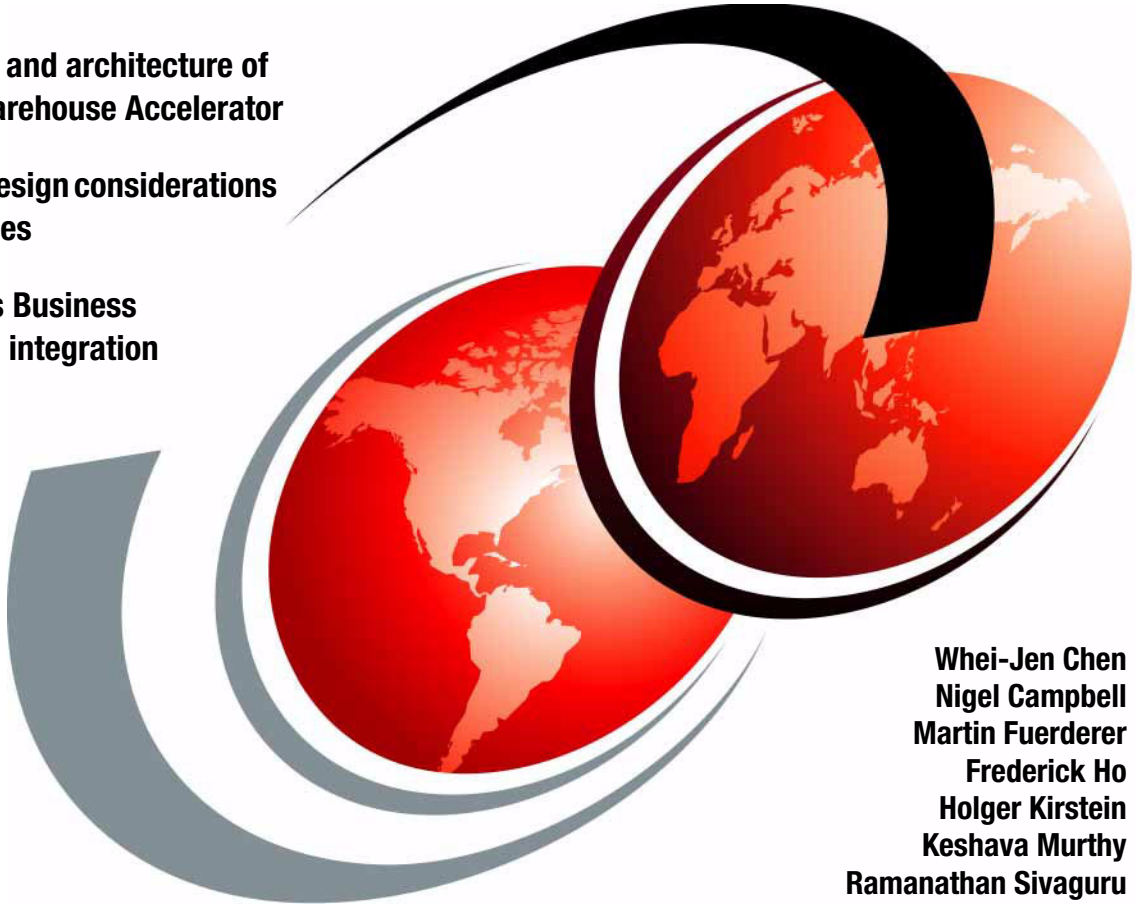
IBM® Information Management Software

Query Acceleration for Business Using IBM Informix Warehouse Accelerator

- Technology and architecture of Informix Warehouse Accelerator
- Data mart design considerations and examples
- IBM Cognos Business Intelligence integration



Whei-Jen Chen
Nigel Campbell
Martin Fuerderer
Frederick Ho
Holger Kirstein
Keshava Murthy
Ramanathan Sivaguru

Redbooks

ibm.com/redbooks

International Technical Support Organization

Query Acceleration for Business Using IBM Informix Warehouse Accelerator

November 2013

SG24-8150-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (November 2013)

This edition applies to IBM Informix Version 12.10.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	GPFS™	Red Brick®
Cognos®	IBM®	Redbooks®
DB2®	IMS™	Redbooks (logo)  ®
developerWorks®	Informix®	SPSS®
DRDA®	Optim™	System p®

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® Informix® Warehouse Accelerator is a state-of-the-art in-memory database that uses affordable innovations in memory and processor technology and trends in novel ways to boost query performance. It is a disruptive technology that changes how organizations provide analytics to its operational and historical data. Informix Warehouse Accelerator uses columnar, in-memory approach to accelerate even the most complex warehouse and operational queries without application changes or tuning.

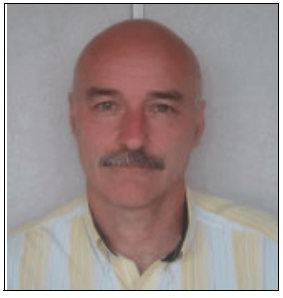
This IBM Redbooks® publication provides a comprehensive look at the technology and architecture behind the system. It contains information about the tools, data synchronization, and query processing capabilities of Informix Warehouse Accelerator, and provides steps to implement data analysis by using Informix Warehouse Accelerator within an organization.

This book is intended for IBM Business Partners and clients who are looking for low-cost solutions to boost data warehouse query performance.

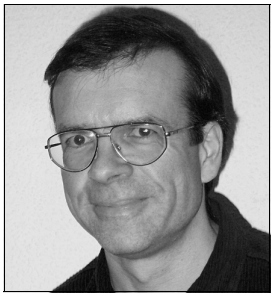
Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and IBM DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development, and an IBM Certified IT Specialist.



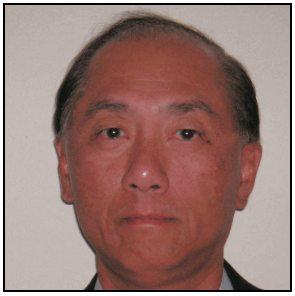
Nigel Campbell is a Senior Developer with IBM Cognos® specializing in the data access and query engine technologies that are used by IBM Cognos Business Intelligence products. He has more than 25 years in the industry, building products and applications that span relational and non-relational platforms.



Martin Fuerderer joined Informix in 1996, working for three years in Advanced Support before changing to Platform Engineering to port the Informix database server to different UNIX platforms. When IBM acquired Informix, Martin joined Informix Product Development, where he enhanced the backup and restore functions. Since its beginning, Martin has been involved in the development of Informix Warehouse Accelerator, which the team in Munich, Germany, started early in 2009.

When the first release of Informix Warehouse Accelerator became available in 2011, Martin started a technical IBM developerWorks® blog that is dedicated to Informix Warehouse Accelerator, and he is still the main author of this blog, which can be found at the following website:

http://tinyurl.com/the-iwa-blog



Frederick Ho is a Program Director for Informix Development and the Chief Technologist for Informix Warehouse. He leads the current initiative for Informix Warehouse Accelerator. Before he joined IBM, he was an Executive Director of Engineering at Informix/Red Brick Systems since 1999. Fred has held engineering and management positions with Tandem Computers, HP, and Sun Microsystems. He holds a Bachelor of Science degree in Computer Engineering from the University of Illinois and a Master of Science degree in Electrical and Computer Engineering from the University of California.

Holger Kirstein is a resolution team engineer with the European Informix support team. He joined the Informix support team in 1996 and has over 17 years experience in application development and support for Informix database servers and Informix clients. He holds a Master of Applied Computer Science degree from Technische Universität, Dresden.

Keshava Murthy is a senior technical staff member and architect for the IBM Informix query processing component. He has worked on Sybase, Illustra, Informix database servers, and Informix Warehouse Accelerator. Keshav developed features in SQL, NoSQL, RTREE, distributed queries, and object-relational components of Informix. Keshav has advised enterprise clients and partners on designing, developing, and deploying enterprise and embedded applications. He is a two-time recipient of the IBM Outstanding Technical Achievement award, for enabling Manhattan logistics applications in 2006 and for query processing innovations in 2011. He holds a Bachelor's degree in Computer Science and Engineering from University Of Mysore, India.

Ramanathan Sivaguru is a senior member of the Informix Competitive Technologies and Enablement team, which is based in the US. He has a Bachelor's degree in Physics and a Master's degree in Computer Applications. Ram has more than 25 years of industry experience and has held positions in software development, technical services, and has leadership experience with support teams. During his career as an IBM customer, Ram was an active customer advisory council member and was instrumental behind getting most of Informix Extended Parallel features into Informix. Ram's job responsibilities include, but are not limited to, conducting customer and IBM Business Partner boot camps on various aspects of Informix technology worldwide, and conducting proofs of concept and proofs of technology. Ram conducts hands-on labs and presentations on various topics at user group meetings and at conferences.

Acknowledgements

The authors thank **Joel Hamill** for editing the book.

Joel Hamill is a senior Information Developer on the Informix team and is based in San Francisco, CA. He has extensive experience authoring technical content for IBM database products, including IMS™, DB2, and Informix. He holds a Bachelor of Science degree in Technical Communication from the University of Washington in Seattle, WA.

The authors also thank the following people for their contributions to this project:

Sandor Szabo
Development Manager, IBM Informix Warehouse Accelerator, IBM Information Management

Andreas Weininger
Leading Technical Sales Professional, IBM Information Management

Karl Ostner
Software Developer, IBM Informix Warehouse Accelerator, IBM Information Management

Andreas Breitfeld
Software Developer, IBM Informix Warehouse Accelerator, IBM Information Management

Sumanth Rajagopal
Software Developer, IBM Informix Open Admin Tool, IBM Informix Management

Sapna Ramesh
Software Quality Assurance Lead, IBM Informix Warehouse Accelerator, IBM Information Management

Veronica Gomes
Client Technical Specialist, Informix Warehousing, Informix Development, IBM Information Management

Andreas Dworsky
IBM Senior Level Accredited Informix Advanced Support, IBM Information Management

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction to IBM Informix Warehouse Accelerator

Informix Warehouse Accelerator (IWA) is a state-of-the-art in-memory database (IMDB) that provides a Google-like experience for analytic queries. It is a disruptive technology that changes how organizations provide analytics to its operational and historical data. This book provides a comprehensive look at the technology and architecture behind the system. It contains information about the tools, data synchronization, and query processing capabilities of Informix Warehouse Accelerator, and steps to implement data analysis by using Informix Warehouse Accelerator within an organization.

1.1 Problems that Informix Warehouse Accelerator solves for its customers

The traditional approach to providing analytics in an enterprise is to separate the transactional system (OLTP) from the data warehouse. Whether the data warehouse is an Enterprise Data Warehouse (EDW) or a data mart, the performance of analytic queries often dictates the separation of transactional versus data warehousing processing. The analytic queries are often complex SQL that is accessing and joining millions or billions of rows.

To provide what is considered “reasonable” performance on the data warehouse, vendors build a “performance layer” that consists of indexes, materialized views, cubes, and even dedicated hardware to pre-fetch and pre-summarize data. In addition, a SQL expert who optimizes queries and a DBA who optimizes the placement of data to achieve performance are needed. All of this overhead is often prohibitively costly to organizations and minimizes the availability of such high-speed access to analytical data.

Informix Warehouse Accelerator provides access to this type of analytical data with an increase of speed by several orders of magnitude, all without the need for the “performance layer”. It provides organizations with unprecedented access to massive amounts of data and can perform interactive and ad hoc queries at a fraction of the cost that is associated normally with data warehouse implementations. Imagine 6 - 8 hours of a “reporting window” per day that is reduced to a few minutes or less. Imagine deploying solutions to users that were never possible before.

Discover how Informix Warehouse Accelerator can solve these problems for your organization.

1.2 History of Informix Warehouse Accelerator

The genesis of Informix Warehouse Accelerator dates from the latter part of 2008 when IBM Informix development manager Sandor Szabo in Munich first learned about a research project that was called BLINK at a meeting at the Boblingen Research Lab in Germany. The BLINK project, led by Namik Hrle at Boblingen and Guy Lohman at Almaden Research, is documented in various publications, notably *Business Analytics in (a) Blink* from the Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2012.

During the same period, another Informix development manager, Frederick Ho, from the San Jose Lab in the US, started an initiative to bring data warehousing capabilities to the Informix Data Server product line, which was called Informix Dynamic Server (IDS) at the time. Informix has a heritage of data warehousing products from its Extended Parallel Server (XPS), an early implementation of massively parallel processing (MPP), and IBM Red Brick® Warehouse, a pioneer of star schema data mart and data warehouse. The flagship product in the Informix product line, IDS, with its large and loyal set of customers, concentrated on providing high performance OLTP in a highly available and minimally administered environment. Before the IBM acquisition of Informix Software in 2001, a project was started called “Arrowhead” that combined the code lines between IDS and XPS. It would have provided comprehensive OLTP and Data Warehousing capabilities in a single Informix product. As a result of reprioritization from the IBM acquisition, this project was put on hold.

In the ensuing years after the IBM acquisition, Informix customers using Informix for OLTP often found analytic solutions on competing vendor platforms, such as Oracle, SQL Server, or Teradata. Given the loyal customer base, there was an opportunity for IBM and Informix to provide a comprehensive data warehouse offering, preferably tightly coupled with the Informix product itself. Together with the BLINK technology, there was extra motivation to “leap-frog” current data warehousing capabilities and bring Informix into a state-of-the-art database that takes advantage of the best of Informix in performance, reliability, and ease-of-use.

With key members of the Informix technical staff, including Kevin Brown and Keshava Murthy, realizing the potential of such a disruptive technology, a prototype was built in May 2009 to demonstrate the viability of deploying the BLINK engine from Informix. Jerry Keesee, the Lab Director that is responsible for the Informix business, and Les King, IM Product Management Director, provided the critical management support to turn this prototype into a product.

There were milestones along the way that included successful early testing at two customer sites: one at a government agency in Germany, and another at a leading retailer in the US. With the release of Informix 11.7 (the Panther release) in October 2010, Informix added significant data warehousing capabilities, including Star Join optimization, multi-index scan, and time-cyclic data management services. This release effectively completed the “Arrowhead” project and provided a database for OLTP and data warehousing on a single platform that is on par with other such products in the industry.

Informix Warehouse Accelerator accelerated analytic queries to Informix. Informix Warehouse Accelerator was released as part of Informix 11.7 in March 2011, and can accelerate analytical queries that are processed by Informix. Informix Warehouse Accelerator requires no change to SQL queries that are submitted to Informix and requires no indexing, partitioning, query optimization, summary tables, and so on, to obtain significant improvement in query time.

Informix Warehouse Accelerator serves the small and medium business (SMB) segment and large enterprises with raw data sizes ranging from a few hundred gigabytes to tens of terabytes. It is a software appliance that is not tied to particular hardware vendors. Configurations for memory and processors can be customized easily. Two editions of the software are available to meet different price points in the marketplace.

1.3 Key Informix Warehouse Accelerator technologies

To achieve its near-instantaneous response time for large scale analytic queries, Informix Warehouse Accelerator takes advantage of commodity multi-core processors and inexpensive Dynamic Random Access Memory (DRAM) main memory that can easily reach multiple terabytes in a single system. However, Informix Warehouse Accelerator is not merely a large buffer pool. It uses a proprietary dictionary encoding method, called approximate Huffman encoding, and cache-conscious algorithms to run most SQL query processing on the encoded data, and to enable single-instruction multiple-data (SIMD) operations on vectors of those compressed values.

1.3.1 Informix Warehouse Accelerator architecture

Figure 1-1 shows the architecture of the information system with Informix Warehouse Accelerator.

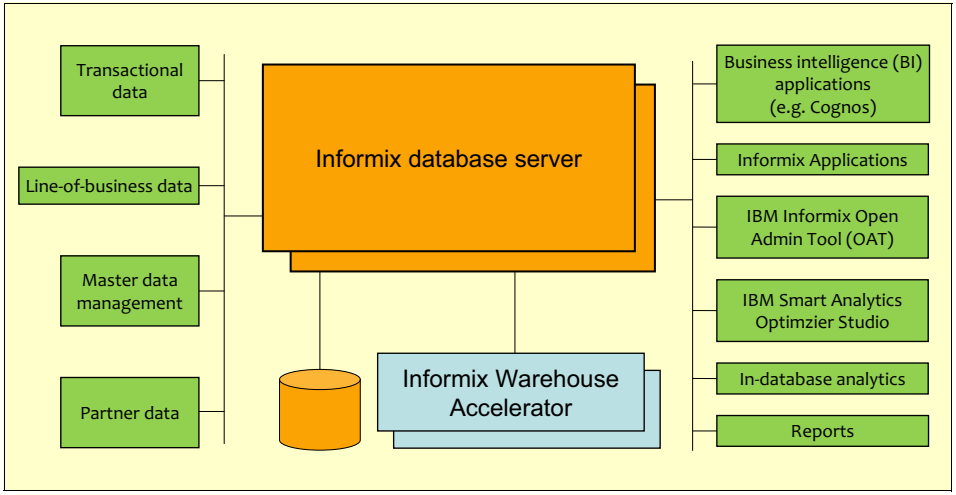


Figure 1-1 Informix and Informix Warehouse Accelerator architecture

Informix Warehouse Accelerator is an attachment to the Informix database server. Although the Informix database server runs on a number of different platforms, for example, IBM AIX®, HP-UX, Solaris, and Linux, the Informix Warehouse Accelerator always runs on a Linux system.

The disk subsystem belongs to the Informix database server itself, not Informix Warehouse Accelerator, which means that Informix maintains the database from which data is loaded into Informix Warehouse Accelerator. Depending on the workload that this Informix server is servicing, the data can be transactional data in third normal form (3NF) or higher, or it can be a data warehouse using de-normalized data, that is, dimensional modeled. All external interfaces, such as business intelligence (BI) or administration tools, are directed to the Informix database server, not Informix Warehouse Accelerator.

1.3.2 Informix and Informix Warehouse Accelerator data mart definition and data loading

A data mart in Informix Warehouse Accelerator is defined as a set of tables, columns within those tables, and the relationship between the tables, for example, fact and dimensions. This data mart might be the entire database schema that is stored in Informix or a subset of the schema.

According to the data mart definition, data is loaded from Informix to Informix Warehouse Accelerator. Because Informix Warehouse Accelerator is an in-memory database, a snapshot of the data that is stored on Informix is taken. The loading process uses sophisticated encoding techniques, such as data compression, and creates a format that takes full advantage of multiple cores within the system.

When the data is loaded into Informix Warehouse Accelerator memory, the system is ready for query execution.

1.3.3 Query execution in Informix and Informix Warehouse Accelerator

SQL queries can be submitted either directly to Informix or through a BI tool. The Informix Optimizer looks at each query and decides on whether the query can be accelerated by Informix Warehouse Accelerator. This determination is made by matching the columns that are referenced in each table in the query to the definition of the data mart. If there is a match, the query is routed to Informix Warehouse Accelerator for execution. If there is no match, the query is run (in its entirety) by Informix. The query is not routed between Informix and Informix Warehouse Accelerator multiple times. The query is either accelerated or not.

After execution, the results are returned by Informix to the query submitter. To the submitter, the experience is the same, except for the much faster execution time.

1.3.4 Data synchronization in both Informix and Informix Warehouse Accelerator

A snapshot of the data in Informix is initially taken and loaded in to Informix Warehouse Accelerator for query processing. When data is inserted, updated, or deleted in Informix, there are several ways to synchronize or refresh the data in Informix Warehouse Accelerator without taking another complete snapshot and reloading the data into Informix Warehouse Accelerator.

You can reload a partition of a table from Informix to Informix Warehouse Accelerator by using the Informix Warehouse Accelerator partition refresh facilities, which can add or update only affected partitions. This facility applies to both fact and dimension tables. Automatic detection and refresh can also be set up to eliminate administration. Depending on how the partitions are defined in Informix, entire tables or fragments of tables can be replaced with this facility.

Another facility that is called *trickle feed* is available for even finer granularity of data refresh in Informix Warehouse Accelerator. It provides what is often called *real-time data warehousing*.

Typically, an enterprise manages data in a “time-cyclic” fashion, where data, for example, sales history, is maintained for two or three years, and when a new month of data is added, the oldest month is removed. By using the trickle feed facility in Informix Warehouse Accelerator, newer additions to partitions are appended to Informix Warehouse Accelerator in a continuous “trickling” of incoming data. Trickle feed provides the lowest granularity because trickle feeding can be at the record level as opposed to the partition level and the effect of “real-time” queries in a data warehouse can be achieved.

1.4 Informix Warehouse Accelerator architectural options

Informix Warehouse Accelerator is supported on Linux x86_64 (EM64T and AMD64). Supported Linux distributions are the current versions of Red Hat and SUSE. Because Informix is supported on many operating systems, it is possible to have both Informix and Informix Warehouse Accelerator run on the same Linux system. This situation is advantageous when you use TCP/IP loop back optimization between Informix and Informix Warehouse Accelerator, which provides a seamless experience to the customer. Queries that are submitted to Informix are accelerated without having to administer a second system.

Although main memory (DRAM) on current Intel Symmetric Multi-processing (SMP) systems can exceed 3 TB at the time of writing, horizontal scaling might be wanted as more data is added. For this reason, Informix Warehouse Accelerator is supported on a blade server running as Massively Parallel Systems (MPP) and on a single SMP machine. The blade server can be from any vendor of choice, assuming it is based on the Intel x86-64 architecture. Configuration considerations are described in Chapter 3, “Designing and sizing an IBM Informix Warehouse Accelerator environment” on page 35.

Informix is known for its replication facilities. It provides for platform independent failover capabilities, such as High Availability Data Recovery (HADR) and Remote Standalone Server (RSS), and a shared-disk cluster that is known as Shared-Disk Server (SDS).

Enterprises that want to provide a single environment that handles both OLTP and data warehousing can submit analytic queries from any of the Informix secondary servers, including HDR, RSS, or SDS, to Informix Warehouse Accelerator. This situation for uninterrupted OLTP response times that are driven through the primary Informix server, and data loading and complex analytic queries are submitted through any of the secondary servers to Informix Warehouse Accelerator for acceleration.

1.5 Components of Informix Warehouse Accelerator

As shown in Figure 1-2, there are two main components within Informix Warehouse Accelerator: the coordinator node and a configurable set of worker nodes. This architecture is the same regardless of whether Informix Warehouse Accelerator is running on an SMP or MPP. The default configuration on an SMP system is a single coordinator node and a single worker node, and an MPP system contains a single worker node on each blade.

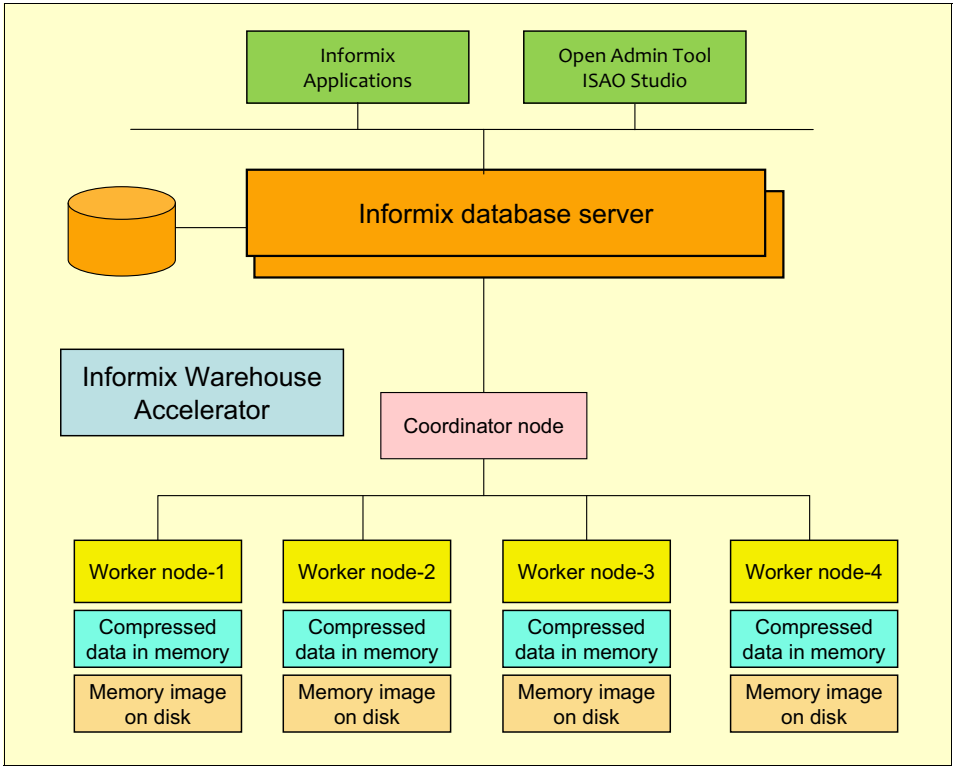


Figure 1-2 Components of the Informix Warehouse Accelerator

1.5.1 Coordinator node

As the name suggests, the coordinator node of Informix Warehouse Accelerator serves as the main point of communication between the Informix database server and Informix Warehouse Accelerator. During the loading phase, the coordinator receives data from the Informix database server and distributes data to the worker nodes. As part of the loading process, data is compressed by using dictionaries in each worker node. The coordinator collects these dictionaries, merges them, and redistributes the final dictionary so all nodes are using the same reference dictionary. During the query processing phase, the coordinator receives a query from the database server, sends the query to each node, collects results from each worker node, and performs necessary merging and sorting before returning the final result to the database server.

1.5.2 Worker node

During the loading phase, a worker node receives data from the coordinator and analyzes the data by using frequency partitioning. The worker node partitions data vertically and horizontally into cells and compresses data by using deep columnar techniques. After the data is compressed, the worker node writes the data out (in compressed format) onto internal disks for recovery purposes.

Each worker node contains a full (compressed) copy of all dimension tables that are defined in the data mart. As part of query processing, each worker node works on a portion of the entire (compressed) fact table, maximizing parallelism among all worker nodes. Intermediate results are returned to the coordinator node.

1.6 Informix Warehouse Accelerator tools

The following tools provide administration and monitoring, data mart design, and automatic schema generation for use with Informix Warehouse Accelerator:

- IBM OpenAdmin Tool (OAT)

  OAT is a browser-based Informix administration tool. OAT is the most comprehensive single tool to administer all activities with the Informix server, including a health center, task scheduler, space administration, server administration, replication, and Informix Warehouse Accelerator.

  With OAT, the Informix Warehouse Accelerator tasks, such as configuration, creating or dropping a data mart, loading and reloading Informix Warehouse Accelerator, and enabling or disabling the data mart, can be accomplished in an easy-to-use, graphical user interface (GUI) environment.

OAT is part of the Informix Warehouse Accelerator bundle and is included at no additional cost.

- IBM Smart Analytics Optimizer Studio

  IBM Smart Analytics Optimizer Studio is an Eclipse-based GUI interface for designing data marts. Users can define a complete database schema, including table and column definitions, data types, key specification, and the relationships between different tables, for example, primary and foreign key specification. It is an easy-to-use drag-and-drop interface that is suitable for designers who are familiar with the overall Enterprise Data Warehouse (EDW) schema and knowledgeable about what part of the overall database can be offloaded to Informix Warehouse Accelerator.

  By using the IBM Smart Analytics Optimizer Studio, you can use the schema for the data mart to load data into Informix Warehouse Accelerator.

- Workload Analyzer

  For DBAs who are not familiar with the overall EDW schema, Informix Warehouse Accelerator provides another method to create the data mart for use in Informix Warehouse Accelerator. Here, the Workload Analyzer, also provided as part of the Informix Warehouse Accelerator bundle, is turned on before submitting a set of queries against Informix. The Analyzer works to capture the table names and column names that are referenced in the submitted queries. When the set of queries completes, the Analyzer generates the data mart in XML format and submits the script as input to the loading process of Informix Warehouse Accelerator.

  This method takes the guesswork out of the data mart design process and ensures that the most problematic queries are accelerated by Informix Warehouse Accelerator.

1.7 Business intelligence tools that are used with Informix and Informix Warehouse Accelerator

SQL queries can be submitted to Informix either directly from SQL programs or from a BI tool. Informix is compatible with most popular BI tools on the market, including Cognos, Microstrategy, Oracle BI (OBI), and Pentaho. Because Cognos is part of IBM, there is more opportunity to collaborate with Cognos to obtain the most optimal queries. These considerations are described in Chapter 9, “Use of IBM Cognos Business Intelligence with IBM Informix Warehouse Accelerator” on page 183.

1.8 Informix Warehouse Accelerator editions

At the time of writing, Informix Warehouse Accelerator is available in two different editions:

- Advanced Enterprise Edition

  Together with the Informix 12.10, the Advanced Enterprise Edition consists of the Informix data server, the data compression facility in Informix, and Informix Warehouse Accelerator. This edition is targeted towards enterprise customers because there is no limit on data size, memory size, or the number of cores in the machine. The same version of the software must be installed on both Informix and Informix Warehouse Accelerator, that is, both must be from the Version 12.10 Advanced Enterprise Edition. It cannot be a mix of two versions of Informix. The Advanced Enterprise Edition is bundled with five licenses of Cognos BI and one license of IBM SPSS®. The five licenses of Cognos BI can be in user or administrator mode.

- Advanced Workgroup Edition

  The second edition of Informix that contains Informix Warehouse Accelerator is called Informix Advanced Workgroup Edition. This edition is targeted towards departmental or workgroup level enterprises, with a limit of 16 GB of DRAM in Informix and 48 GB of DRAM in Informix Warehouse Accelerator. There is also a limit of four sockets and 16 cores for the processors in Informix and 16 cores in Informix Warehouse Accelerator. The price point for the Advanced Workgroup Edition is lower than the Advanced Enterprise Edition. It also contains five licenses of Cognos BI. This edition is most suitable for smaller scale enterprises. 48 GB of main memory can be deployed for enterprises with raw data sizes of well over 100 GB given the compression ratio that is built into Informix Warehouse Accelerator.

The editions and bundling of products and licensing restriction can change from one release to the next.

1.9 Informix Warehouse Accelerator positioning

There are many competing products in the BI space today. Where does Informix Warehouse Accelerator fit in the marketplace?

We attempt to position Informix Warehouse Accelerator by using categories that are defined by market research companies that are published for the BI space:

- Data warehouse appliance

  Data warehouse appliances are characterized by database software that is installed on specific hardware that is optimized for a data warehousing workload. It processes large amounts of data efficiently by using various algorithms, including data compression, pre-fetching, and pre-filtering of data. The benefit is that these highly “workload-optimized systems” are configured with software and hardware that are integrated, without the need for further customization, thus working as an “appliance”. When you implement a data warehouse appliance, consider the lack of customization as part of the total cost of the packaged system.

  Examples of data warehouse appliances are DataAllegro (Microsoft), Netezza® (IBM), Greenplum (EMC), Exadata (Oracle), Dataupia, and Kognito.

- Columnar database

  Traditional relational DBMSs over the last 20 years are designed with a row-oriented page layout where a row (with all columns in a row) is stored after another row in the table. Most OLAP queries tend to access only a few columns in a row, making the I/O of entire rows on database pages inefficient. Columnar databases store all values of a column contiguously before storing the values of the next column of the same table. This allows retrieval of entire columns of a table to be efficient. The columnar storage method allows for easy compression as column values tend to be similar and reduced sizes of these columns mean even faster I/O. Columnar databases have become popular as analytical marts, but they are not suitable for an OLTP workload.

  Examples of columnar databases are Calpont, Exasol, Infobright, ParAccel (Actian), Sand Technology, Vertica (Hewlett Packard), and Sybase IQ (SAP).

- In-memory OLAP tools

  In-memory OLAP tools provide excellent analytic performance on data in the client layer. They deploy analytic algorithms for various business applications by using in-memory techniques, often without exposing the underlying database layer to the user. Data in the client layer must be refreshed when data changes in the database layer.

Examples of In-memory OLAP tools are QlikView/QlikTech, Applix TM-1 (IBM-Cognos), Exalytics (Oracle), and PALO.

- In-memory data warehouse

  In-memory data warehouses process OLAP queries efficiently by using a combination of in-memory and compression algorithms. It can also use columnar storage (in-memory) to provide vertical partitioning to achieve partition elimination and increase parallelism throughout. These systems scan base table data without further need for indexing or summarization. They can also spill to disk as data volume increases beyond what is contained in main memory.

  Examples of In-memory data warehouses are HANA (SAP), IBM DB2 BLU (IBM), and Informix Warehouse Accelerator (IBM).

2

Designing data marts

This chapter provides an overview of database designs for transactional and analytical database systems. It describes the techniques of dimensional modeling for data warehouses, including star and snowflake database schemas. Also covered are the specific design aspects for data marts in Informix Warehouse Accelerator and practical considerations.

2.1 Relational database design overview

The purpose of a database determines what the schema for the database looks like. When you design the schema for a relational database, consider how the data is segmented into different tables and how these tables are related to each other.

The relationships between tables are defined by primary and foreign keys. A parent table has a primary key that identifies its data records. A child table establishes the relationship of its own data records to the data records of the parent table by a foreign key that references the primary keys of the parent table. Ideally, each parent table primary key uniquely identifies a single data record. This then constitutes a 1:n relationship between the parent and the child table; otherwise, it is a n:m relationship.

At a high level, we distinguish between databases that are designed for online transaction processing (OLTP) and databases that are designed for online analytical processing (OLAP). Because Informix Warehouse Accelerator is a data warehouse accelerator, database designs that follow the dimensional modeling method for analytics are preferred. However, it is useful to review the transactional database designs to understand how the design methodologies differ.

2.1.1 Designs for OLTP

OLTP systems manage transaction-oriented applications. For example, an automated teller machine (ATM) is an application that accesses and changes the data of bank accounts by using transactions. The term OLTP can be used to refer to commercial business transactions, but in this book the term refers to the computational processing of transactions by a database system.

To facilitate OLTP for a database, it is important to correctly and efficiently process the changes that are applied to single data records. For example, when you withdraw cash at an ATM, the data record of a single bank account is changed. The requirement is that the data is always consistent, even for concurrent access of several users, where the concept of transactions ensures this state. Relational databases that are designed for OLTP generally follow the Entity-Relationship (ER) model and provide Atomicity, Consistency, Isolation, and Durability (ACID) properties. Usually, OLTP is also required to provide the response for the completion of a transaction to the user within an acceptable time window.

The concept of database normalization was introduced by Edgar E. Codd in 1970 as a method for structuring the tables and their columns in a relational database while minimizing data redundancy and dependency. This concept evolved from the early First Normal Form to its present day incarnation. For more information about database normalization, see the following resources:

- Date, C. J. *An Introduction to Database Systems* (8th ed.), Addison-Wesley Longman, 1999, ISBN 0321197844
- Kent, W., *A Simple Guide to Five Normal Forms in Relational Database Theory*, Communications of the ACM, vol. 26, pp. 120–125, 1983

Highly normalized database schema designs are best suited for fulfilling the requirements of an OLTP system. By avoiding data redundancy, a normalized schema design delivers high data consistency. Changes to the data must be applied only to the single occurrence of the data. There is no possibility of accidentally omitting the update of any duplication of the same data in different places. The normalization also increases the overall performance of change processing because only a small amount of unique data must be changed in a single place.

An example for a normalized database design is the schema that is developed for the TPC-E benchmark. Figure 2-1 shows a portion of that database schema. Although it is only a portion of the TPC-E database schema and the illustration is simplified, it is obvious that considerable effort is required to understand the schema and take full advantage of the data.

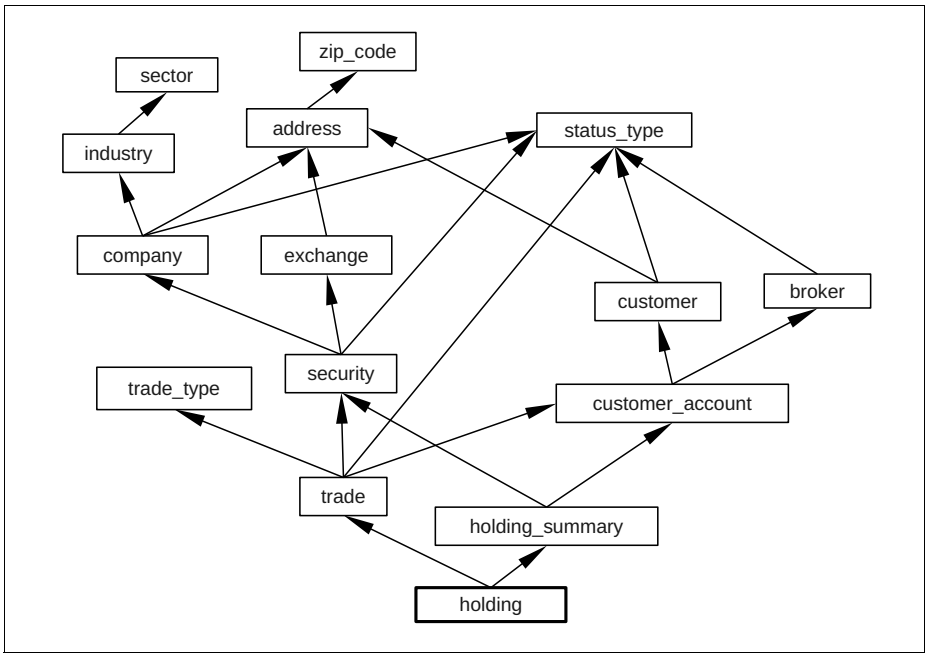


Figure 2-1 Partial TPC-E database schema

The figure shows the complexity of the schema. The *holding* table contains information about the security holdings in a customer account. The *holding_summary* table is an aggregation of the holding table. The *customer_account* table stores the customer account information. A customer can have more than one account, so customer information is not stored in the customer_account table because this duplicates the customer data within each of their accounts. Instead, the customer data is normalized into the separate *customer* table. This principle is continued for the address of a customer, which is normalized into a separate *address* table, with the postal code stored in a separate *zip_code* table. For example, if you want to know the postal code of the customer, you must join together all of these tables: holding with holding_summary with customer_account with customer with address with zip_code. To formulate the needed SQL query statement, you need the names of all of the join columns that define the relationships between these tables.

2.1.2 Designs for OLAP and dimensional modeling

OLAP is used in the area of business intelligence as a method to do multidimensional data analysis efficiently. Applications include business reporting, trend analysis, and forecasting. When data is aggregated, it is referred to as roll-up or consolidation. When you review the details of the data, it is referred to as drill-down. When a subset of data is taken out and viewed separately from different angles, it is referred to as slicing and dicing. For more information about OLAP, see *Management information systems (9th ed.)*, by O'Brien, et al.

You can collect the data from several different sources, but for analysis it is stored in a central data warehouse (DW). Before the data can be inserted into the data warehouse, it is often necessary to apply some changes so that it conforms to the required format. The process of getting the data and putting it into the data warehouse is referred to as extract, transform, and load (ETL).

For the database system, the most important task of data analysis is efficiently processing large amounts of data. Tracking changes to the data is not important because the data is used for analysis and therefore is only read. Data is loaded in bulk, rather than applying small units of data change at a time. You can denormalize the database, which causes data redundancy, but increases the processing performance by eliminating joins between the tables. As the database schema becomes simpler, it is easier to understand and the data can be used better, with fewer complicated join specifications.

Dimensional modeling

When designing warehouse databases, the dimensional modeling technique is used because it emphasizes understanding and performance. According to warehousing consultant Ralph Kimball, the transaction-oriented ER model should be avoided for delivering the data to users. For more information about dimensional modeling, see *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, *3rd edition,* by Kimball, et al.

The *fact tables* contain the measurements or metrics, for example, of a business process, such as sales orders or material shipments. The *dimension tables* contain descriptive fields, often with textual or discrete numerical values that are used with filters in queries or as labels for the results. The fact tables are the child tables, and their foreign keys reference the primary keys of dimension tables, which take on the role of parent tables. Dimension tables can reference other dimension tables, thus building a hierarchy of dimension tables.

Depending on the degree of denormalization, the schemas with dimensional modeling are referred to as *star* or *snowflake* schemas.

Star schema

The star schema is more denormalized than the snowflake schema, and is the simplest design for data warehouses. It consists of one fact table, which directly references any number of dimension tables. The dimension tables do not reference any other tables because they are denormalized. Therefore, the star schema often is considered a special case of the snowflake schema. Denormalized dimension tables are larger in size because of data redundancy, but if they are small when compared to the fact table, this is not a problem. Theoretically, the dimension tables can be further denormalized into the fact table, leaving a single table that contains everything. However, the fact table normally has an order of magnitude more data records than the dimension tables, and such denormalization increases the size of the single table beyond usability. The star schema is a practical schema design of a data warehouse for analytic processing and is easy to understand by the users. The relationship between the fact table and the dimension tables are rather simple and you can join the tables easily when writing queries.

The natural primary keys of dimension tables cannot always be easily joined with the fact table because the primary keys must be unique. For example, because names of people are not unique, a primary key for a customer table might require a combination of a given name, surname, birth date, and possibly the address or something like a Social Security number to be unique. In many cases, it is easier to use artificially created unique values as a surrogate key, for example, a serial number. The database system can provide this number automatically and it is a simple integer. Including only the surrogate key as a foreign key in the fact table provides better performance when processing the join on this key and it saves space in the fact table. Often, the fact table by itself does not have a unique primary key, but generally all foreign keys of a fact table combined together can serve as unique (foreign) key, if needed.

An example of star schema database design is the database schema that was developed for the TPC-DS benchmark. A partial aspect of the TPC-DS database schema with the fact table store_sales is shown in Figure 2-2.

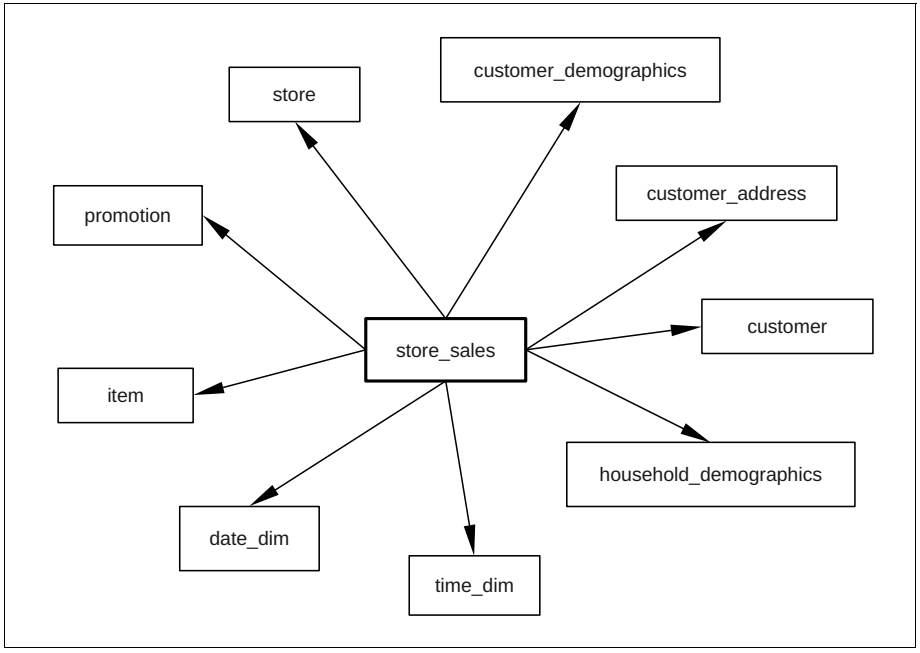


Figure 2-2 TPC-DS star schema for table store_sales

In the example star schema that is shown in Figure 2-2, the fact table store_sales directly references table customer_address, even though (not shown in Figure 2-2) table customer also references customer_address. With this denormalization, you can get information about a customer's address, for example, the postal code, without first joining table store_sales with table customer, even though you are not interested in any of the information in table customer. The direct reference from the fact table store_sales to table customer_address makes formulating the respective SQL query statement much easier. Furthermore, postal code information is contained not only in table customer_address. The postal code information for stores is stored directly in table stores. If you want to correlate the postal code of the store with the customer address postal code for all the sales in stores, you must join the fact table store_sales with the table customer_address and the table stores. This join makes the join conditions of the SQL statement for the query rather simple. Furthermore, dimension tables customer and customer_address are sometimes combined into a single dimension table that contains all of the customer data, which makes the schema even simpler.

Examples of the measurement values that are stored in the TPC-DS fact table store_sales are quantity, wholesale cost, list price, net profit, and sales price. Examples of the attributes of dimension table customer are salutation, given name, surname, login, and email address.

Snowflake schema

When you add new kinds of data to an existing star schema, the new data must either be added as new columns to the existing dimension tables or as new dimension tables that are referenced directly by the fact table at the center of the star schema. However, this action can lead to excessive data redundancy, which causes the dimension tables to become large. Very large dimension tables are problematic with certain configurations of Informix Warehouse Accelerator, which is described in 2.3.3, “Dimension tables and multiple Informix Warehouse Accelerator worker nodes” on page 33.

To keep dimension tables from growing too large, normalize the schema to a certain extent. The result is a snowflake schema with a hierarchy of dimension tables. An example of this situation is the introduction of a date dimension table named date_dim in the TPC-DS schema. Such a date dimension table is useful for storing more information about specific dates, for example, the day of the week, whether it is a holiday or the day before a holiday, and in which quarter of the year the date is.

Because each date is unique in the calender, there is no need to store all of this information multiple times in different tables for different purposes. Instead, the other tables can refer, by a key, to a date dimension table whenever they must store a date value. For customers, this can be the date of their first purchase, and for a promotion, the start and end dates of the promotion are relevant, and for a sale, the date of the actual transaction. The tables store, customer, and store_sales can all refer to a table date_dim. For each date, possibly referenced by several other tables, only a single record is stored in table date_dim, which saves space, especially when dimension tables are rather large compared to the fact tables. But, it also adds another level of joins whenever such a date value is needed for a query. The corresponding SQL statements become more complex, but this is still a good trade-off compared to the otherwise increased data size. The more complex a snowflake schema is, the more sophisticated the users or software applications must be when exploring the database.

A snowflake schema is a compromise between a pure star schema and a fully normalized schema, allowing for some data redundancy while avoiding excessive data duplication. Figure 2-3 shows an example of the logical view of a snowflake schema that is based on the TPC-DS database design. A few more dimension tables (without further specifying them) are added to the figure to better illustrate why such a schema is called a snowflake.

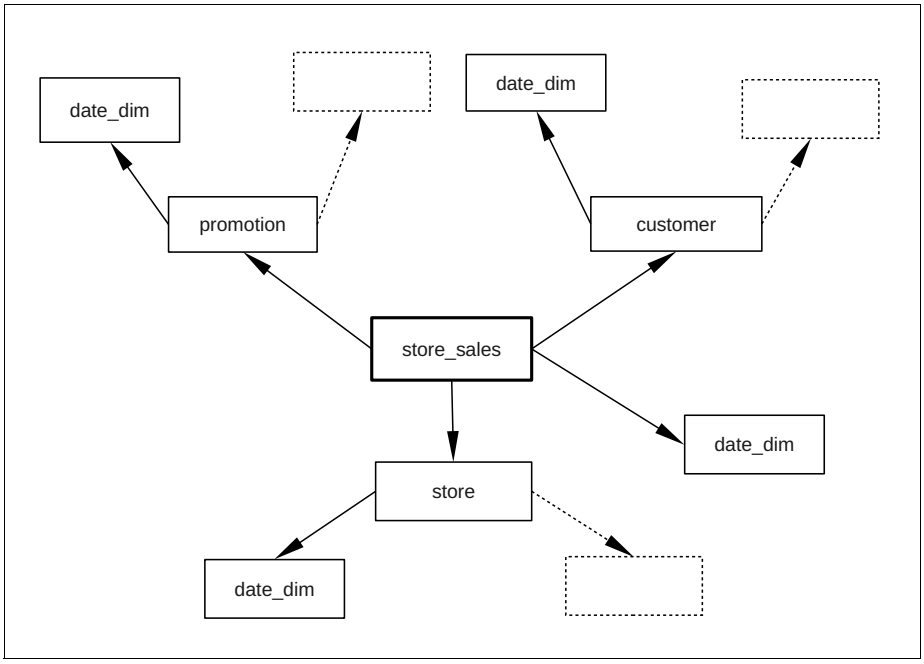


Figure 2-3 Logical view of a more complex snowflake schema

2.2 Dimensional modeling for Informix Warehouse Accelerator

In data warehousing terminology, a *data mart* is defined as a subset of the data in the warehouse database. Often, this subset is tailored to serve a specific business need or department working with the data. In this book, the term *data mart* is defined as an entity of data from a warehouse database that is pushed into the Informix Warehouse Accelerator to accelerate queries. Even though several data marts can exist in the same accelerator server, they are separated logically and physically. Therefore, they do not share any memory for storing their data.

2.2.1 Degree of denormalization

The data mart in the accelerator server must be defined based on existing tables in the warehouse database. You should consider how denormalized you want the star or snowflake schema in the warehouse database to be before you design the data mart. For Informix Warehouse Accelerator, it is best when dimension tables are relatively small compared to the fact table. Denormalization from a snowflake schema to a star schema can cause the resulting dimension tables in the star schema to become rather large. Large dimension tables can later be problematic when you design and create the data mart based on such a star schema (as explained in 2.3.3, “Dimension tables and multiple Informix Warehouse Accelerator worker nodes” on page 33). Therefore, a slightly more complex snowflake schema might be preferred to the simpler star schema, especially if it reduces the overall size of the dimension tables. It is difficult to provide specific numbers as recommendations, but if the number of data records in a dimension table is equal to or greater than the number of data records in the fact table, you should carefully consider the effects of further denormalizing. A dimension table that has considerably more data records than the fact table indicates that the overall design, including the data marts that will be built, can probably benefit from normalization.

2.2.2 Composing a data mart

Informix Warehouse Accelerator always uses a single data mart to accelerate a specific query. Therefore, a data mart must contain all of the data that is needed for the query. If the query requires data from a table or column that is not included in the data mart, then this data mart cannot be used to accelerate the query. When you manually design a data mart, you must include all of the required tables and columns for the queries for the data mart to accelerate. However, including everything from the warehouse database also might not be optimal because it increases the size of the data mart unnecessarily.

Informix Warehouse Accelerator uses columnar organization for the data of a data mart, which means that if you have a large data mart that contains tables or columns that are never needed, it does not slow query performance, but only wastes memory resources. A data mart, even though its data is compressed, is kept in memory, including parts of the data mart that are never used by any query. It is better to exclude unneeded tables or table columns to conserve memory for other data marts or to keep memory available for temporary usage during query execution. For example, wide character columns that contain long textual descriptions or comments are good candidates for exclusion. This process is described in Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69.

In general, the data marts that are defined for Informix Warehouse Accelerator are snowflake schemas, even though they do not necessarily reflect the complete schema of a data warehouse. They are often only a subset of the data warehouse schema, such as a simpler star schema or even some degenerate form of a snowflake schema (such as a single, linear 'branch' of a snowflake). For example, a degenerate snowflake could comprise only a single fact table, without any dimension tables.

Here are two important prerequisites for queries to be accelerated by the accelerator:

- The fact table of the query is a fact table in a data mart.
- The joins in the query are reflected as references in that data mart.

The Informix server checks these prerequisites when deciding whether a query can be accelerated.

Figure 2-4 shows a rather simple snowflake schema that is a subset of the TPC-DS database schema. The store_sales table is the fact table at the center of the snowflake. All other tables are dimension tables. The columns with names that end with "_sk" are surrogate key columns that define the primary-foreign key relationships between the tables. The other columns are attributes that contain user data. (For simplicity, not all columns of these tables are shown in the figure.)

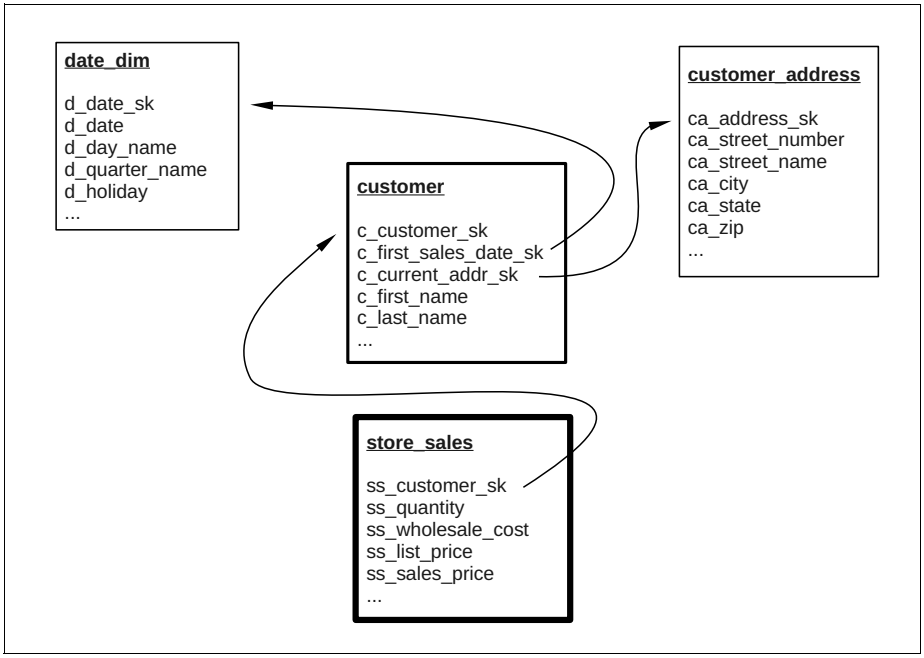


Figure 2-4 Simple snowflake schema

If a data mart with the structure shown in Figure 2-4 on page 25 is built, queries that have store_sales as their fact table can be accelerated with this data mart. Queries can include data on customers, addresses, and the first sale to each customer by joining the fact table with the respective dimension tables. You are not required to use all joins in a query. A query that only joins the store_sales table with the customer table can also be accelerated with this data mart. Even a query that selects data only from the store_sales fact table, without joining it with any other table, can be accelerated.

In general, a snowflake schema contains a single fact table. This fact table references the dimension tables, but it is not referenced by any other table. When modeling a data mart from a snowflake schema, the resulting data mart also has only a single fact table.

2.2.3 Subsets in a snowflake schema

In the data mart that is shown in Figure 2-4 on page 25, the store_sales table is the fact table and Informix Warehouse Accelerator can use this data mart to accelerate queries with store_sales as their fact table. However, you can also issue a query that joins only the customer table with the customer_address table and the date_dim table. For example, this query can be used to analyze a customers age in correlation with the area where they live. This is sometimes referred to as *dimension browsing*; however, it actually reveals the recursive nature of the hierarchy of dimensions in a snowflake schema.

Figure 2-5 shows a schema for queries with the customer table as the fact table.

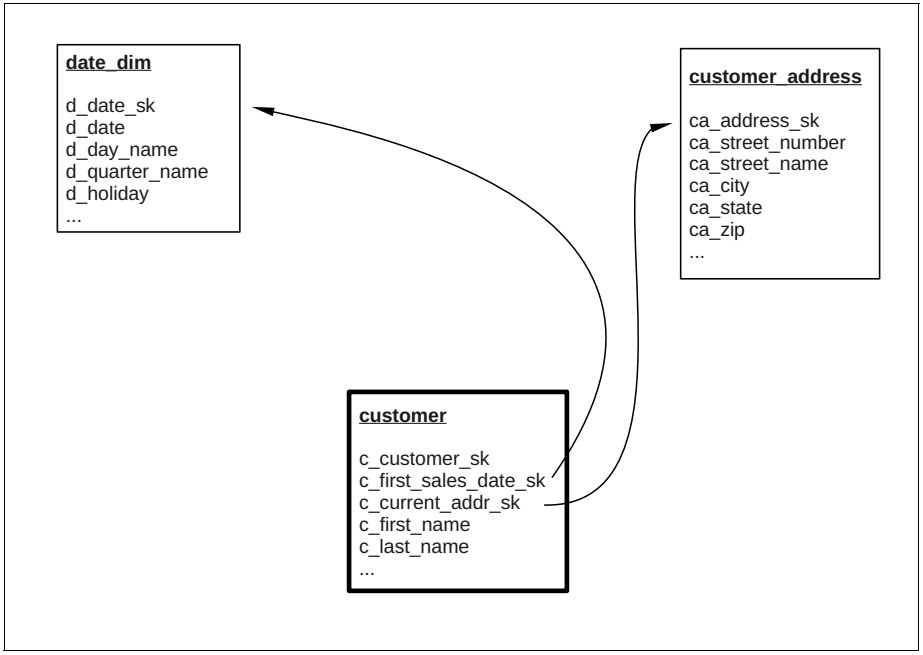


Figure 2-5 Query with the customer table as a fact table

For this query, the interesting facts are the customer names, birthdate, age, and sex. For the purposes of this query, the customer table is the fact table. The dimensions are the customer addresses and the dates. The fact table store_sales of the data mart that is shown in Figure 2-4 on page 25 does not appear at all. However, this query does not meet the prerequisite that states the fact table of the query must be a fact table in the data mart. Therefore, this query cannot be accelerated with the data mart, even though the required tables are a subset of the data mart.

To solve this problem you can create a separate data mart with a subset of the first snowflake schema, which contains only the tables customer, customer_address, and date_dim, matching the structure that is shown in Figure 2-5. This data mart by itself is a small star schema. The customer table is the single fact table and the customer_address and date_dim tables are the dimension tables. A query that joins the customer and customer_address tables can now be accelerated with this separate data mart.

Because a data mart in Informix Warehouse Accelerator is its own entity, if you define a data mart that contains a subset of tables that are part of an existing data mart, the data is duplicated in the accelerator server. This is not an effective way of using memory resources.

Informix Warehouse Accelerator provides a solution by allowing more than one fact table in a single data mart, even though such fact tables are not fact tables in the corresponding snowflake schema. In Figure 2-5 on page 27, the customer table is a prime candidate for being both a dimension table to some queries and a fact table to others. Tables that play both roles are *hybrid dimension tables*. You make the customer table a hybrid dimension table by flagging it as a fact table in the data mart definition. A hybrid dimension table is still a dimension table, which means that the customer table still acts as dimension table when a query joins it with the store_sales fact table. But, the data mart can now also be used to accelerate a query that uses hybrid dimension table customer as the fact table. The data mart has the same size as though the store_sales table is the single fact table. You do not have to define two separate data marts, each with only one of the two tables as fact table, and therefore the table data does not need to be duplicated in subsequent data marts. All queries that use one of the two tables as a fact table (and respective joins to dimension tables) can be accelerated with this single data mart.

A single fact table can be considered a (degenerate) snowflake schema, meaning that in a data mart every table can be a fact table. In Figure 2-5 on page 27, even the *leaf node dimension tables* of the snowflake schema, customer_address and date_dim, can be designated as fact tables in the data mart definition. However, designating all tables as fact tables might not always make sense. This action allows acceleration of queries that use just one of these tables without any join, but it is not guaranteed that an actual acceleration takes place. If the dimension tables are relatively small, the amount of work that is required to send the query to the accelerator server and to send the result set back to the Informix server might outweigh any acceleration effect.

2.2.4 Overlapping snowflake schemas

Warehouse databases often contain more than one fact table. Each of these fact tables is the center of a star or snowflake schema. With Informix Warehouse Accelerator, it is possible to include several star or snowflake schemas in a single data mart. However, whether this type of data mart setup works for you depends on many factors.

From the perspective of an administrator, it makes sense to have separate data marts for different projects or departments. This type of data mart setup provides individual data mart management that is tuned to the needs of the users or projects. This setup is useful when you schedule times of unavailability during a data load or when you rebuild a data mart with a new definition after structural changes in the warehouse database are made.

To maximize memory usage, you can include several star or snowflake schemas into a single data mart if they share some of the dimension tables. Data marts are separate entities so they cannot share memory. For example, two data marts for two different star or snowflake schemas cannot share the data of a dimension table that is part of both schemas. The dimension table must be stored in each data mart, where it occupies the full space that is required. You can combine several star or snowflake schemas into a single data mart to avoid this space issue. For a single data mart, the shared dimension tables exist only once within the data mart and this avoids the duplication of their data into separate data marts. The result is a data mart that contains several star or snowflake schemas that partially overlap at the edges.

Figure 2-6 shows a fictional data mart that contains three snowflake schemas that partially overlap by sharing some dimension tables.

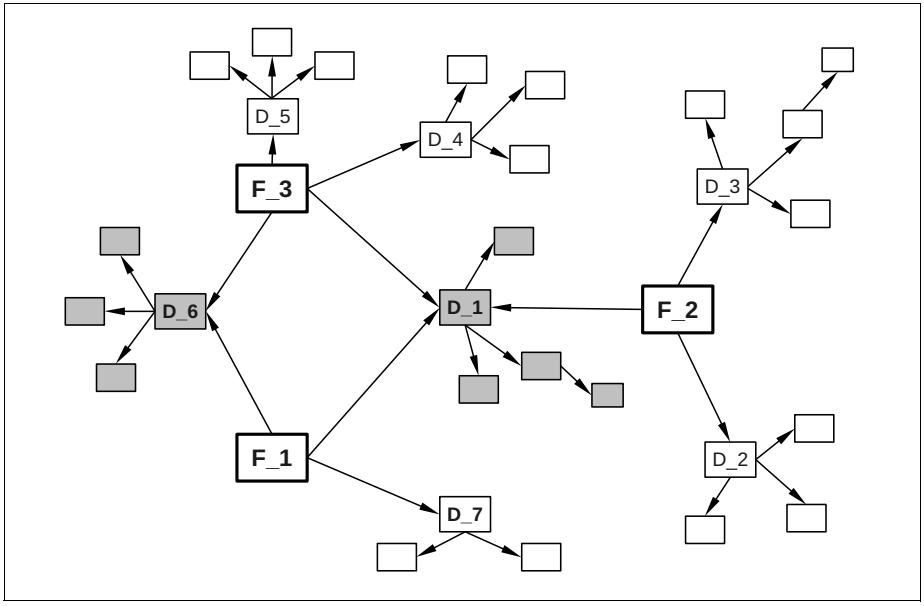


Figure 2-6 Data mart with three partially overlapping snowflake schemas

In Figure 2-6 on page 29, tables F_1, F_2 and F_3 are the fact tables, each at the center of its own snowflake schema. Dimension table D_1 is referenced by all three fact tables and is shared by all three snowflake schemas. The four dimension tables referenced (indirectly) by table D_1 are also shared among the three snowflake schemas. Dimension table D_6 is referenced only by fact tables F_1 and F_3. Combined with the dimension tables D_6 references, it is part of the snowflake schema of fact tables F_1 and F_3, but is not part of the snowflake schema that is formed by fact table F_2.

The goal is to find the appropriate balance between separating each star or snowflake schema into a separate data mart versus combining them into a single data mart. There are no generic rules to provide for this exercise. The final design depends on the specific demands and available resources in each environment.

2.2.5 Types of relationships between tables

The relationships between tables in a data mart are defined by their references. In a relational database, a reference between two tables is established by the columns in the child table that refer to columns in the parent table. The referring columns in the child table are defined as a foreign key for the child table. The referenced columns in the parent table are defined as a primary key in the parent table. You do not have to define primary and foreign keys to establish references between the tables in a data mart. You must define all of the necessary relationships between tables as references in the data mart so that the queries can later join the tables. If a query attempts to join two tables where this join is not established as a reference in the data mart definition, Informix Warehouse Accelerator cannot use this data mart to accelerate this query. The type of relationship between tables in the database determines what the corresponding reference is in the data mart. The two kinds of relationship are *one-to-many* and *many-to-many*.

One-to-many relationship

In a one-to-many relationship, each parent table record can have zero, one, or more child records in the child table. Each child record has exactly one parent record. Each parent record must be uniquely identifiable by the parent table key.

If the parent key is a composite key, meaning a combination of several columns in the parent table, then the key values can potentially grow large. To reduce the space requirements for the key in the child table, you can use a surrogate key, for example, a serial value that is provided by the database server. This surrogate key is an integer that requires only 4 or 8 bytes of space.

A one-to-many relationship between two tables is reflected in the data mart definition as a reference with the cardinality attribute for the parent table as ‘1’ and for the child table as ‘n’.

One-to-many relationships that are defined as a 1:n reference in the data mart are the preferred type of relationship because they increase the processing performance of the query joins. If you cannot establish a one-to-many relationship between two tables in the database, you can create a surrogate key. However, because of the nature of the data, this is not always possible.

Many-to-many relationship

If a one-to-many relationship cannot be established between two tables, then it is a many-to-many relationship. In this case, a data record in the child table is related to more than one data record in the parent table. This multiple relationship is not reflected by a primary-foreign key relationship in a relational database because the primary key implies that the key values in the parent table must be unique. In the data mart, a reference between the two tables can be defined only if it has a cardinality of ‘n’ for the parent table and a cardinality of ‘m’ for the child table.

Avoid joins between tables that are related with an ‘n:m’ reference because they are more costly to run during query acceleration. Because ‘n:m’ references in a data mart require more Informix Warehouse Accelerator internal data structures, the maximum number of ‘n:m’ references in a data mart is limited to 10.

Prerequisites for relationships

Here are the prerequisites for defining a reference in a data mart:

- The data types of the corresponding columns that are used to establish the relationship between the tables must be equivalent.
- The parent key must be unique when you define a ‘1:n’ reference. At a minimum, the parent table key in the database must have a unique constraint, a unique index, or a primary key index. The condition is verified when a data mart is created because a ‘1:n’ reference without a unique parent key causes incorrect query results.
- You cannot exceed 10 ‘n:m’ references.

2.3 Designing Informix Warehouse Accelerator data marts

This section describes Informix Warehouse Accelerator data mart design, including supported data types, data compression and storage methods, and metadata on data marts.

2.3.1 Supported data types

Many of the Informix server data types are supported by Informix Warehouse Accelerator with some exceptions. For a complete list of supported data types in a specific version of Informix Warehouse Accelerator, see *IBM Informix Warehouse Accelerator Administration Guide*, found at:

https://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/acc.htm

Here is a condensed list of supported data types (as of Informix Warehouse Accelerator V2.10.xC1):

- INTEGER, SMALLINT, SERIAL, BIGINT, and BIGSERIAL
- CHAR, VARCHAR, and LVARCHAR
- DATE and some variations of DATETIME
- DECIMAL(p,s), MONEY, FLOAT, and SMALLFLOAT
- TimeSeries Data

2.3.2 Data compression

To maximize memory usage, the data of a data mart is compressed by Informix Warehouse Accelerator before the data is stored in memory. (The data is also written to disk in its compressed format for recovery purposes. Query acceleration uses only the data that is in memory.) To achieve good compression rates, Informix Warehouse Accelerator combines a variation of the Huffman compression with the columnar organization of the data. You can expect compression rates of 1:3 - 1:5, with even better rates possible. However, there is no guarantee of universally valid numbers because the actual compression rates depend on the nature of the data.

The compression technique is dictionary-based. The dictionary is built during a full load of the data into a data mart. Each full load operation on the data mart rebuilds the dictionary from scratch, which provides the best compression over time. Because a partial data refresh of an existing data mart does not rebuild the compression dictionary, it can result in less efficient compression over time. The advantage of a partial refresh is the fast performance and continuous availability of the data mart for query acceleration. For more information about the compression technique that is used and its benefits, see *IBM Informix Warehouse Accelerator: Performance is everything*, found at:

http://public.dhe.ibm.com/common/ssi/ecm/en/imw14587usen/IMW14587USEN.PDF

The complete data of a data mart, in its compressed format, must fit into the available memory. In addition to data mart storage, Informix Warehouse Accelerator temporarily uses memory when running query acceleration (for more information, see Chapter 3, “Designing and sizing an IBM Informix Warehouse Accelerator environment” on page 35). Even with the benefits of compression factored into the size estimates for data marts, keep in mind the overall size when you are designing a data mart.

2.3.3 Dimension tables and multiple Informix Warehouse Accelerator worker nodes

This section focuses on configurations with multiple worker nodes. However, consider these design aspects as well if your current single worker node configuration might change in the future to include multiple worker nodes.

Because of the unique architecture of Informix Warehouse Accelerator (see Chapter 3, “Designing and sizing an IBM Informix Warehouse Accelerator environment” on page 35), it can support shared-nothing hardware nodes of a cluster system. This environment requires multiple worker node support and the data mart dimension tables must be available to every worker node. Table data is distributed among multiple worker nodes. The dimension tables must be available in their entirety to each worker node so that the worker nodes can do all of the join operations independently of each other. Running the join operations independently allows Informix Warehouse Accelerator to process single queries in parallel, where each worker node joins its share of the fact table data with all of the dimension tables. Each worker node produces its final query result independently of the other worker nodes. To accomplish this independence, each worker node requires its own copy of the complete dimension table data. All dimension tables must be duplicated to the memory of each worker node. This condition is also true for hybrid dimension tables, as described in 2.2.3, “Subsets in a snowflake schema” on page 26.

2.3.4 Data mart metadata in the Informix server

The Informix server stores pieces of metadata on every data mart in the database for which the data mart was created. This metadata consists of special views that are created in the database when a data mart is created. A separate view is automatically created for each fact table and hybrid dimension table in the data mart. Any ‘n:m’ reference that occurs in the data mart might cause more of these views to be created. The views are called *Accelerated Query Tables* (*AQTs*), and they are primarily used by the Informix server optimizer to decide whether a specific query can be accelerated by Informix Warehouse Accelerator.

When query acceleration is activated, the optimizer takes a query and attempts to match it against the AQTs. If a match is found, the query can be accelerated and the Informix server sends it to Informix Warehouse Accelerator for execution by using the corresponding data mart. If several matches are found, the Informix server optimizer chooses the data mart that was most recently loaded with data. If no match is found, the Informix server runs the query locally. There are multiple options to control this behavior, as described in Chapter 6, “Query execution” on page 103.

When a data mart state changes, the AQTs are automatically updated. For example, when a data mart is not available for query acceleration during a full data reload, the corresponding AQTs are updated and the optimizer in the Informix server is aware that this data mart is not available. When a data mart is dropped, the corresponding AQTs are also dropped from the database.

3

Designing and sizing an IBM Informix Warehouse Accelerator environment

This chapter describes the unique architecture of the Informix Warehouse Accelerator and its implications on supported runtime environments, including symmetric multiprocessor (SMP) systems, cluster systems, and virtual machines (VMs). It describes how to implement continuous availability of data marts and how to estimate the sizing requirements for the different environments.

3.1 Architectural considerations

Informix Warehouse Accelerator can be installed on different hardware environments, including symmetric multiprocessor (SMP) systems or cluster systems. Additionally, you can install it on a virtual machine. The main requirement is that Informix Warehouse Accelerator runs on a Linux x86_64 (EM64T or AMD64) system with processors that support the Streaming SIMD Extensions 3 (SSE3) instruction set.

3.1.1 SMP environment

Informix Warehouse Accelerator runs in a shared-everything environment of an SMP system. This is a single machine with multiple processors and a matching amount of memory. Because the Informix Warehouse Accelerator environment is based on the inexpensive Intel architecture, it is common to have systems with 16 - 24 processor cores and main memory of 500 GB - 1 TB.

Informix Warehouse Accelerator implements a multi-threaded process architecture that takes advantage of an SMP system by using a coordinator node and a single worker node. Each Informix Warehouse Accelerator node consists of a single multithreaded process on the Linux operating system. You can specify configuration parameters to set the number of physically available processor cores that these nodes can use when running multiple threads in parallel. The work units of the different tasks, such as loading data into a data mart and running queries, are split up into portions that can be run in parallel by these threads. Query execution benefits from this architecture because the work of each singular query is run with maximum parallelism, rather than running many separate queries concurrently on their own single thread. The operating system schedules which threads are run on which processor cores. The cache conscious algorithms that are implemented in Informix Warehouse Accelerator use the different levels of processor memory caches and avoid unnecessary context switches, which can require the individual threads to migrate from one processor to another.

The coordinator and worker node configuration parameters control the maximum amount of memory that can be used for keeping structures and data permanently in memory. This memory is used for the data marts in their compressed format. The remaining memory is used temporarily on an as-needed-basis by all coordinator and worker node threads. In an SMP environment, all of the memory requirements must be satisfied by the physically available memory of the machine. You must find an adequate memory usage balance for the coordinator and worker nodes.

Figure 3-1 schematically shows Informix Warehouse Accelerator in a single worker node configuration running in an SMP environment.

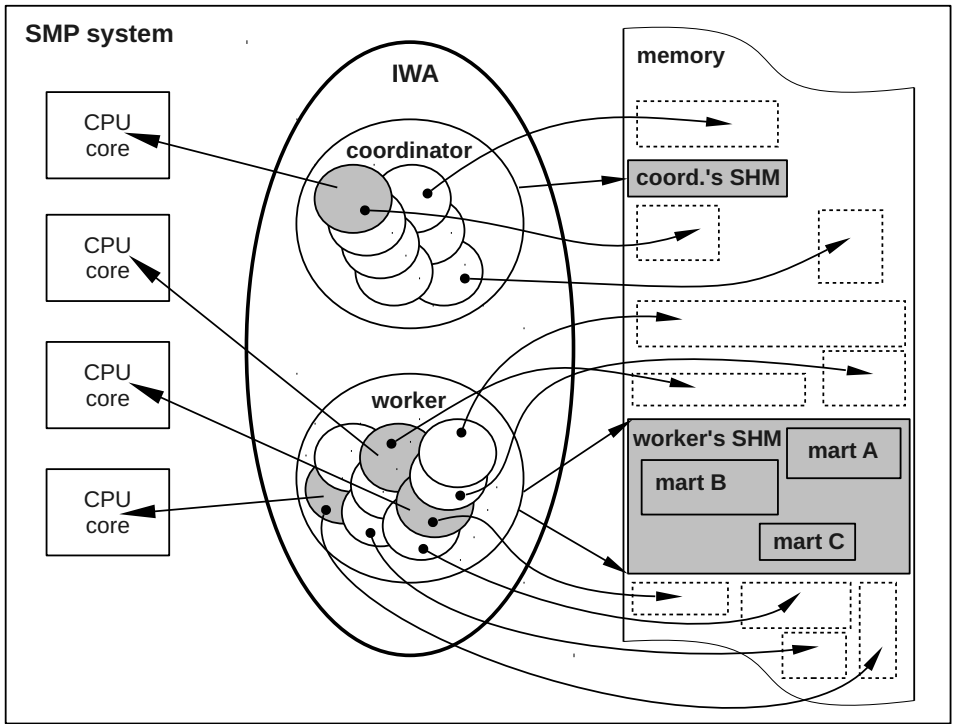


Figure 3-1 Example of Informix Warehouse Accelerator single worker node configuration in an SMP environment

Figure 3-1 shows a four processor core SMP system with three threads of the worker node and one thread of the coordinator node running on the four processor cores. The coordinator and the worker node each have their shared memory (SHM) in the main memory. The SHM of the worker node is large because it contains the complete data of all data marts (in their compressed format). Several smaller pieces of memory are temporarily allocated for private use (shown with dotted line boxes) by different threads. Individual threads privately allocate and release pieces of memory as needed for the different tasks they are working on. For example, tasks can include building hash tables from scanned and filtered table data or joining data that is scanned from different tables to previously built hash tables, both during query execution, and building histograms or the data dictionary and compressing the data, both during data load. For more information about how memory is used, see 3.3.1, “General resource considerations” on page 45.

Multiple worker nodes

The multiple threads of the nodes share access to the SHM of their node, which requires coordination among these threads (within their respective node). With more than 24 processor cores available in an SMP environment, it is possible for a single worker node to start so many threads to use all of these processor cores that performance cannot scale any further. In this situation, processor time is unnecessarily used for coordinating the many threads rather than getting the work done. To avoid this situation, multiple worker nodes can be configured. Multiple worker nodes are independent from each other (SHM is not shared among several nodes), so the individual nodes have fewer threads to coordinate among themselves. However, when the threads for all worker nodes are combined, there are enough threads to use the processor cores. With the reduced thread coordination, the threads use more processor time to get the work done, resulting in better parallelism. The increased performance is especially true for data loading, where building the histograms and the dictionary, and compressing the data means that much of the shared memory access by the threads is write access.

In a multiple worker node configuration, the fact table data is distributed among the worker nodes so that each worker node has its own portion of fact table data to independently work with. However, this configuration requires that each worker node has the complete data of the dimension tables to join it with the respective portion of the fact table. The worker nodes do not share SHM with each other (as they also can run separated on different cluster hardware nodes), so the dimension table data must be duplicated into the SHM of each worker node. Dimension tables are much smaller than the fact tables, and so this duplication is not a problem. In cases where the dimension tables are large, multiple worker node configuration can cause a large increase in the amount of memory required. Therefore, in an SMP environment, Informix Warehouse Accelerator has a single worker node by default.

Figure 3-2 schematically shows an Informix Warehouse Accelerator configuration that consists of two worker nodes running in an SMP environment.

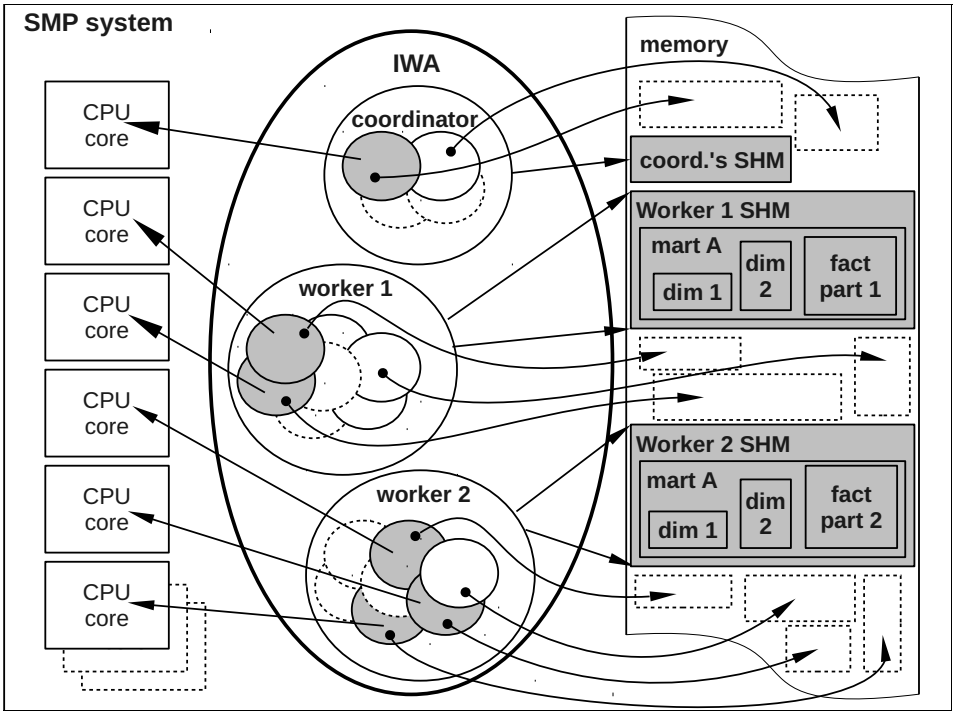


Figure 3-2 Informix Warehouse Accelerator two worker node configuration in an SMP environment

Figure 3-2 shows Informix Warehouse Accelerator with two worker nodes that are running on an SMP system with many processor cores. Two threads of worker node 1 and three threads of worker node 2 are running, and one thread of the coordinator node. Each worker node has its own SHM, which contains a single data mart. The fact table data is partitioned between the two worker nodes, but the dimension table data is duplicated. Several threads are temporarily allocated an amount of memory for their private use (dotted line boxes).

3.1.2 Cluster environment

Although in an SMP environment everything is shared and accessible by everyone, a cluster environment is the opposite. In a cluster environment, each cluster node has its own processor cores and memory, with exclusive access to these resources. Communication between cluster nodes is implemented by using rapid network connectivity (for example, fiber optic channels) and applications that use the TCP protocol. Conceptually, you can think of a cluster environment as being a set of separate computers with high-speed connections between them.

In a cluster environment, Informix Warehouse Accelerator is configured to run a single accelerator server node on each cluster node. One cluster node runs the coordinator node and the remaining cluster nodes each run one worker node. Memory and processor resources are not shared between cluster nodes. Each individual Informix Warehouse Accelerator node is given all of the resources of its respective cluster node.

Figure 3-3 schematically shows Informix Warehouse Accelerator in a four-node cluster environment.

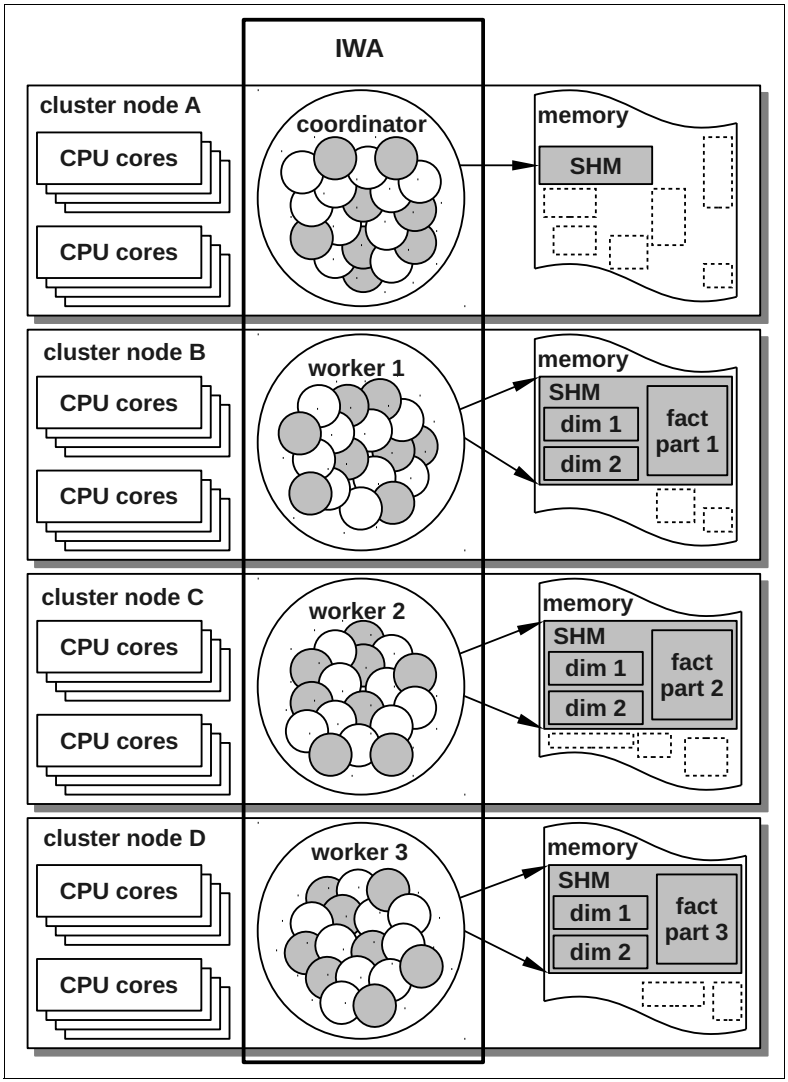


Figure 3-3 Informix Warehouse Accelerator in a four-node cluster environment

Figure 3-3 on page 41 shows Informix Warehouse Accelerator running in a cluster environment with four cluster nodes that each have two quad-core processors. There is a single accelerator server node on each cluster node. The coordinator node runs on cluster node A, where eight of its threads (shaded circles) run on the eight processor cores of this cluster node. The coordinator node has its SHM in the memory of this cluster node, and threads are temporarily allocating private memory (dotted line boxes). The accelerator server worker node 1 runs on cluster node B. Its SHM in the memory of this cluster node has a single data mart that contains a portion of the fact table data and the complete data of the two dimension tables. Informix Warehouse Accelerator worker node 2 and 3 run on cluster nodes C and D. Each of the nodes has its portion of the fact table data in its SHM and the complete data of both dimension tables.

For recovery purposes, Informix Warehouse Accelerator writes essential data structures and the data marts in their compressed format to disk. In a cluster environment, portions of the disk must be accessible by all of the accelerator server nodes. You must configure a shared file system for the cluster nodes.

Only the cluster node of the Informix Warehouse Accelerator coordinator requires a connection from the Informix server. The remaining cluster nodes require only connections to each other and with the cluster node where the Informix Warehouse Accelerator coordinator node runs. Informix Warehouse Accelerator worker nodes do not require connectivity to the Informix server.

3.1.3 Virtual machine environment

For Informix Warehouse Accelerator, a virtual machine environment is similar to an SMP environment in that the accelerator server is not aware of the virtual machine and Informix Warehouse Accelerator does not do anything special on the virtual machine. The virtual machine must have properties similar to a physical machine. Specifically, the virtual machine for Informix Warehouse Accelerator must be assigned a fixed amount of physical resources with exclusive access. You must assign physical resources for processor cores and memory.

A virtual machine that runs Informix Warehouse Accelerator must be granted exclusive access to a fixed amount of physical resources or problems can occur. For example, the accelerator server memory could be swapped out to disk by the virtual machine manager. When Informix Warehouse Accelerator attempts to access the data, the virtual machine manager intercepts the memory access of Informix Warehouse Accelerator and swaps the appropriate memory region from disk back into main memory. This action causes disk I/O. Avoiding such disk I/O is the main purpose of Informix Warehouse Accelerator as an in-memory database accelerator.

Another problem that could occur without exclusive physical resources is that the virtual machine manager could temporarily remove processor resources from the Informix Warehouse Accelerator VM and give it to another VM. This removal of processor resources requires Informix Warehouse Accelerator to do a processor context switch when its VM is granted processor access the next time round. The advantages of the Informix Warehouse Accelerator cache conscious algorithms are lost without exclusive access because another VM can preempt all of the cache data that was previously in use by Informix Warehouse Accelerator. Upon regaining processor access, Informix Warehouse Accelerator must again fill the caches with its own data. It is not useful to run Informix Warehouse Accelerator in a production environment on a VM that does not have exclusive access to a fixed amount of resources. A test environment is the only situation where non-exclusive access can be useful, when stable performance is not important.

3.1.4 Changing an existing configuration

You can change certain configuration parameters for an existing Informix Warehouse Accelerator installation. To make the changes effective, you must shut down and restart Informix Warehouse Accelerator. However, other configuration parameters are static by nature, and to change them you must do a new setup.

The **NUM_NODES** parameter is an important static configuration parameter that specifies the number of accelerator nodes. You cannot change the **NUM_NODES** parameter in an existing configuration because the fact table data is distributed to the worker nodes at the data load time. This data remains static for the lifetime of the data mart in the respective worker nodes, according to this distribution. Even if you change the number of worker nodes, the fact table data is not redistributed. If you add worker nodes, it does not affect the existing data marts because the added worker nodes do not contain any data from the existing data marts. Without such fact table data, the added worker nodes cannot contribute to query acceleration by using those data marts. If you remove worker nodes, their portion of fact table data disappears and the queries that use an affected data mart produce incorrect results.

To change the number of worker nodes, you must do a new setup and you must re-create all data marts and load them with data. This process ensures that the fact table data is distributed appropriately among the new number of worker nodes.

The configuration change requirements always apply, regardless of the type of environment on which Informix Warehouse Accelerator is running. You must take particular care when deciding on the number of Informix Warehouse Accelerator nodes in a production system. To change the number of nodes later requires downtime for Informix Warehouse Accelerator and must be planned carefully.

3.2 Continuous data mart availability

At times during the lifecycle of a data mart, you must refresh the complete data set. For example, if you want to change the data mart definition by adding a column for a table, you must re-create the data mart with the new definition and load the complete data. During the data loading process, which can take a considerable amount of time, the data mart is not available for query acceleration. Continuous data mart availability offers a solution to this downtime.

To achieve continuous data mart availability, a second data mart with the same or new definition and a different name, is created and loaded. While this new data mart is being loaded, the original data mart remains available for query acceleration. When the new data mart is finished loading, it becomes active and is used for all future query acceleration. The data mart switch occurs automatically in the optimizer of the Informix Server. The optimizer decides whether to accelerate a query and which data mart is used. The moment the final query that uses the original data mart is complete, you can manually drop the original data mart.

By using this method, you can accelerate queries continuously even while loading data. However, this method requires twice the amount of data mart memory, although only for a short amount of time. You must configure the appropriate amount of shared memory for the worker nodes based on this method. You can specify the memory allocation with the **WORKER_SHM** parameter.

For more information, see the “Replacing a data mart” section in *IBM Informix Warehouse Accelerator Administration Guide”*, found at:

https://pic.dhe.ibm.com/infocenter/informix/v121/index.jsp?topic=%2Fcom.ibm.acc.doc%2Fids_acc_099.htm

Additional methods for partial data refresh in a data mart without rendering the data mart unavailable are described later.

3.3 Sizing

Although Informix Warehouse Accelerator can share machine resources with other applications, this section assumes that Informix Warehouse Accelerator is running exclusively in the environment that is being described and does not share its resources with other applications.

3.3.1 General resource considerations

The most important resources are memory and processor cores. The disk space resource (or shared disk space resource in a cluster environment) is less problematic because it is less costly.

Memory resources

Allocate a large portion of memory to the worker node to use it as shared memory (SHM) for storing the data of data marts. This memory is configured by the **WORKER_SHM** configuration parameter. Unlike the Informix server, Informix Warehouse Accelerator does not pre-allocate the complete size of the configured SHM. Instead, SHM is allocated as needed to accommodate the data of the data marts as they are created and loaded with data. The value that is specified for the **WORKER_SHM** configuration parameter sets the maximum amount of SHM that Informix Warehouse Accelerator allocates.

Allocate memory to the coordinator node by specifying the **COORDINATOR_SHM** configuration parameter. The coordinator node uses much less SHM than the worker node. You can allocate about 1/10th or less the amount of worker node memory to the coordinator node.

Because of the compression, you cannot predetermine the exact size of a data mart because the compression ratio varies with the nature of the data itself. You can start by using a rough estimate of a compression ratio of 1:3. As you gain more experience with how the specific data is compressed, you can provide more accurate estimates. Informix Warehouse Accelerator does not require indexes. When you calculate the size estimate of the data mart, you need to include only the raw data in your sizing. Do not include any unnecessary columns or whole tables in the data mart estimate; deduct their size when you determine the size of the raw data in the database.

The worker node threads also require extra memory temporarily when they are loading data into the data mart (for example, to build the histograms and the dictionary) and when they are running a query (for example, to build internal hash tables to store intermediate results and join them with more data while the threads are working on the different tables that are accessed in the query). Informix Warehouse Accelerator coordinator and worker nodes allocate and use extra memory as needed. The architectural design assumes that Informix Warehouse Accelerator is running exclusively on the machine or cluster node, so temporary memory allocation by the threads cannot be configured or otherwise restricted. If necessary, the threads attempt to allocate all available memory from the operating system.

After a thread no longer requires the allocated memory, it releases it. Informix Warehouse Accelerator does not necessarily give the memory back to the operating system immediately. Informix Warehouse Accelerator has its own, internal memory management, which retains an amount of memory in internal pools for imminent reuse. This memory pooling is done to avoid frequent memory allocation and deallocation system calls, which can be costly operations.

For query execution, the worker node passes the results to the coordinator node after it finishes its work on the query. If the coordinator node must do some final processing before it hands the final result to the Informix server, this processing might require extra memory. While the coordinator node is processing the final result and transferring it back to the Informix server, the worker node can start working on the next query. This means that while the worker node is using temporary memory for its purposes, at the same time the coordinator node threads can also require extra memory for processing the final result of the previous query. If the available memory is not sufficient for a specific query, then this particular query individually fails.

The coordinator node stores the query result set in temporary memory to transfer it to the Informix server, which then passes it to the user. The coordinator node requires this temporary memory until the result transfer is complete, or until the user interrupts the receiving of the result transfer (for example, by prematurely closing a cursor). If the user is receiving the result slowly, for example, by a cursor, then the completion of the result transfer can take a long time, during which the coordinator holds on to that temporary memory. Queries that produce large result sets can cause the coordinator to use a considerable amount of memory. If several memory-consuming queries queue up in this execution phase, which can cause the threads of the coordinator node to allocate a significant amount of memory.

To avoid failures that are caused by memory shortage during query processing, you must ensure that sufficient memory is available for temporary, private allocation by the multiple threads of Informix Warehouse Accelerator nodes.

If queries that require the maximum amount of temporary memory during execution are rare or not expected, then you can use swap space. You should not plan to use swap space regularly, but an occasional slow down because of swapping might be more tolerable than a failing query because of a temporary memory shortage.

Processor resources

The Informix Warehouse Accelerator coordinator and worker node processes are multi-threaded and can take advantage of multiple processor cores. For memory allocation, there is a specific configuration parameter for coordinator and worker nodes. However, processor usage is not separated by coordinator and worker nodes, but rather by the task being run. The two different task types are loading data into a data mart and running a query. The amount of processor resources that are used for each of these task types is specified as a percentage of available processor cores in the configuration parameters. The coordinator and worker nodes use these parameters to determine how many parallel threads can be run for the respective task.

The loading data task includes receiving the data from the Informix server, building the histograms for the dictionary, and compressing the data that is based on that dictionary. The processor usage of this task is controlled by the **CPU_PERCENTAGE_LOAD** configuration parameter.

The query execution task scans the table data in memory, performs the table joins that are part of the query, decodes the compressed data, processes the final results (mainly sorting and grouping), and transfers the results to the Informix server. The processor usage of this task is controlled by the **CPU_PERCENTAGE_SCAN** configuration parameter.

Disk space

Although Informix Warehouse Accelerator stores all of the data marts entirely in memory for rapid data access, Informix Warehouse Accelerator also requires disk space. A portion of this disk space is local to each Informix Warehouse Accelerator node, for private use, but some of it is also shared among all Informix Warehouse Accelerator nodes. At a minimum, in a cluster environment, the latter part must be on a file system or disk subsystem that can be accessed from all the cluster nodes where the Informix Warehouse Accelerator nodes run. You can implement this configuration with the IBM General Parallel File System (GPFS™). You can also create similar file systems that allow parallel read and write access, such as clustered file systems or shared disk file systems.

The local and the shared disk portions are both in subdirectories of the directory that are specified by the **DWADIR** configuration parameter. The local portion is in the `local` subdirectory and is for temporary, private use by the individual Informix Warehouse Accelerator nodes. The shared portion is in the `shared` subdirectory and is used to store copies of the data marts in their compressed format for recovery purposes. For example, when a machine is rebooted, the Informix Warehouse Accelerator nodes read the data from the `shared` subdirectory and load the data directly into memory, without having to pull the data from the Informix server.

To estimate the amount of disk space that is required, use these rules:

- Local: 50% of the size of data marts in memory for temporary use
- Shared: 150% of the size of data marts in memory for the recovery copy and additional statistics information, and 150% of the largest table in any data mart for temporary storage (histogram and dictionary building) for use during data load

For a simple setup, you do not have to differentiate between the `shared` and `local` subdirectories. Both subdirectories can be on the same file system. In this type of setup, you can add the two estimates to get a total size estimate.

When you load a data mart, the data is loaded sequentially, table after table. For each table, the data is received from the Informix server, processed, and builds the data dictionary. The data is temporarily stored on disk. After the dictionary is complete, the table data is read from disk and compressed for storage in memory by using the dictionary. After the processing of one table completes, the data is received for the next table. The space requirement for the largest table is normally sufficient for the load processing of the entire data mart.

These guidelines assume that data mart loads are not done for several data marts concurrently. Concurrent data mart loads can occur in a larger production system, where several disjointed Informix Servers use the same Informix Warehouse Accelerator for their own data marts. The different Informix Servers, and even the different administrators, might not be aware of each other and can independently schedule their data mart loads. As a result, several data marts from different Informix Servers can be loaded concurrently. In such a case, you must adjust the disk space estimate.

3.3.2 Sizing for an SMP environment

In an SMP environment, all of the resources are shared among the processes that are running, including memory and processor cores. Each process has access to all of the memory, and the threads of a process can run on any of the processor cores. The standard configuration of Informix Warehouse Accelerator in this environment is with a single worker node.

Single worker node configuration

In an SMP environment, Informix Warehouse Accelerator must be configured so that the coordinator and worker node can start enough threads to use the available processor cores efficiently. With the assumption that Informix Warehouse Accelerator is the only application running on the machine, the resources of the machine must be given only to these two processes of Informix Warehouse Accelerator: the coordinator node and the worker node.

Memory

To avoid failures during query processing that are caused by a memory shortage, keep about twice the size of the largest data mart as free memory for temporary, private memory allocation by the worker and coordinator node. The remaining memory can be allotted for data mart data by specifying the **WORKER_SHM** configuration parameter. You should also allocate a smaller portion of memory to the coordinator node by specifying the **COORDINATOR_SHM** parameter. The coordinator node amount is much less than the SHM of the worker node.

Table 3-1 shows an example for estimating the minimum memory size for three data marts.

Table 3-1 Example steps to estimate minimum memory size for three data marts

Step	Description	Mart A	Mart B	Mart C	Total
1	Size of raw data in the database	1 GB	3 GB	2 GB	6 GB
2	Size of compressed data for data marts	1/3 GB	1 GB	2/3 GB	2 GB
3	Size for worker's SHM (total size of compressed data for data marts)				2 GB
4	Size of coordinator's SHM (10% of worker's SHM)				0.2 GB
5	Size of temporary memory for worker node threads (size of biggest data mart, which is data mart B)				1 GB
6	Size of temporary memory coordinator node threads (equals size of temporary memory for worker node threads)				1 GB
7	**Total size of memory that is required for Informix Warehouse Accelerator in this example**				**4.2 GB**

This example calculates the minimum memory requirements. For a production environment, do not base your sizing on minimum values. If the example was a sizing for a production system, the estimate requires at least 50% more memory to be allocated in case of failures that are caused by temporary memory shortages. This is because there is uncertainty about the compression ratio, which is assumed to be 1:3. In addition, a production system must accommodate for data growth over time and allocate more memory according to the expected data mart growth rate.

Processor cores

The worker node threads run well in parallel and the overall performance scales almost linearly, to a certain point. For environments with up to 24 processor cores, the scaling is good with the default single worker node.

In an Informix Warehouse Accelerator environment where data marts are always loaded at times when there are no queries to be run, data load and query execution can use 100% of the available processor resources. However, if data loading and query execution occur concurrently, more processor resources might be needed to maintain the same level of performance for the query execution. The additional processor resources are then used by the load processing.

It is difficult to provide an example with actual numbers for sizing processor resources because the performance of query execution depends on the queries in the workload.

Multiple worker node configuration

If Informix Warehouse Accelerator has access to many processor resources, it can start too many threads. In this situation, merely adding more processor resources does not provide better performance. In this type of situation, you can configure multiple worker nodes that share the processor resources between them. The configuration parameters that control processor usage must be adjusted accordingly. For example, when you upgrade from a 16 processor core machine with a single-worker Informix Warehouse Accelerator to a 64 processor machine with a four worker Informix Warehouse Accelerator, then the two processor percentage parameters must specify values of around 20%.

The dimension table data is duplicated into the SHM of every worker node, even on a shared-everything SMP system. The overall size of memory that is required to store the data of data marts increases according to the size of the dimension tables and the number of worker nodes that are configured. Because fact table data is distributed among the worker nodes, the SHM size for individual worker nodes might actually decrease.

The example that is shown in Table 3-2 calculates the minimum memory requirements with a four worker node configuration (based on the example in Table 3-1 on page 49 for single worker node configuration, with three data marts of different size characteristics).

Table 3-2 Example calculation for four worker node configuration

Step	Description	Mart A	Mart B	Mart C	Total
1	Size of compressed data for data marts	0.3 GB	1 GB	0.6 GB	-
2	Size of fact tables	0.25 GB	0.5 GB	0.25 GB	-
3	Size of dimension tables	0.05 GB	0.5 GB	0.35 GB	-

Step	Description	Mart A	Mart B	Mart C	Total
4	Size of partial fact table on single worker node	0.0625 GB	0.125 GB	0.0625 GB	-
5	Size of data mart on a single worker node	0.1125 GB	0.625 GB	0.4125 GB	-
6	Size of data marts on all worker nodes combined	0.45 GB	2.5 GB	1.65 GB	4.6 GB
7	Size of worker nodes' SHM (all worker nodes combined)				4.6 GB
8	Size for coordinator's SHM				0.5 GB
9	Size of temporary memory for threads of single worker node (equals size of biggest data mart)				0.625 GB
10	Size of temporary memory for threads of all worker nodes combined				2.5 GB
11	Size of temporary memory for threads of coordinator node				1 GB
12	**Size of the memory that is required for Informix Warehouse Accelerator in this example**				**8.6 GB**

The example calculation shows that changing from a single worker configuration to a four worker configuration increases the minimum memory that is required from 4.2 GB to 8.6 GB. This is roughly twice the amount. As before with the estimate for the single worker node configuration, this example calculates the minimum memory requirements. You might need to allocate more memory to avoid swapping or query failures that are caused by temporary memory shortage. To plan for growth, you can allocate even more memory. If substantial dimension table growth is expected, you can allocate more memory to account for the dimension data duplication to the multiple worker nodes.

To avoid the increased memory requirements that are caused by multiple worker nodes, the default configuration of Informix Warehouse Accelerator for an SMP environment is the coordinator node and a single worker node. The advantages and disadvantages of increasing the number of worker nodes must be considered carefully.

3.3.3 Sizing for a cluster environment

For the shared nothing environment of a cluster system, all cluster nodes must be equipped with the same amount of memory and processor cores. Although, the Informix Warehouse Accelerator coordinator node might not use all of the resources of its cluster hardware node. This section focuses on the sizing of an individual cluster node rather than the cluster system as a whole.

It is possible to run more than one Informix Warehouse Accelerator node on a single cluster hardware node, but this is not recommend. At the time of writing, individual cluster nodes normally do not have more than 24 processor cores, and therefore a single Informix Warehouse Accelerator node is able to use the available resources. Each Informix Warehouse Accelerator node gets all of the available resources of the cluster hardware node where it is running.

Memory

Each worker node can allocate temporary memory equal to the size of the largest data mart. You can configure the remaining memory as SHM so that the worker node can use this to keep data marts in memory. The fact table data is distributed evenly among the worker nodes, and the dimension table data is duplicated to each worker node.

For a worker node in a cluster environment, you do not need to account for memory that is used by the coordinator node because the coordinator node is running on a separate cluster hardware node. The coordinator node does not require much SHM, so you can configure a smaller amount of SHM for the coordinator node than for a worker node. This ensures that the coordinator node has enough memory available for temporary allocation, as all cluster nodes have the same amount of physical memory.

The example that is shown in Table 3-3 calculates the minimum memory requirements for a cluster with five hardware nodes, one coordinator node, and four worker nodes (based on the previous example with three data marts of different size characteristics).

Table 3-3 Example calculation for a five node cluster system

Step	Description	Mart A	Mart B	Mart C	Total
1	Size of compressed data for data marts	0.3 GB	1 GB	0.6 GB	-
2	Size of fact tables	0.25 GB	0.5 GB	0.25 GB	-
3	Size of dimension tables	0.05 GB	0.5 GB	0.35 GB	-
4	Size of partial fact table on single worker node	0.0625 GB	0.125 GB	0.0625 GB	-
5	Size of data mart on a single worker node	0.1125 GB	0.625 GB	0.4125 GB	1.15 GB
6	Size of temporary memory for threads of single worker node (equals size of biggest data mart)				0.625 GB
7	**Size of the memory that is required for each worker node**				**1.8 GB**

The example shows the minimum size of SHM for each worker node is 1.2 GB. The coordinator node has a 0.3 GB SHM, and with the remaining 1.5 GB, it has enough memory available for temporary allocation.

For a cluster environment, the **WORKER_SHM** configuration parameter specifies the amount of SHM for all worker nodes combined. In this example, it is a minimum of 4.6 GB. The combined memory that is required for all five cluster nodes is a minimum of 9 GB.

Add more memory on top of the estimated minimum requirement, for example, to avoid query failures that are caused by a temporary memory shortage. Also, you must consider the potential growth of data marts and allocate more memory.

Processor cores

Each Informix Warehouse Accelerator node that runs on its own cluster hardware node can be given 100% of the available processor cores. If the data mart load operations do not coincide with query execution, the two parameters that specify processor resource usage can both be set to 100%. If data mart load occurs while queries are running, the two parameter values can be balanced according to the preferences of users and the administrator.

If you add or remove cluster nodes, you must perform a new setup. In this situation, the **NUM_NODE** configuration parameter must be changed, but is static by nature.

3.3.4 Sizing for a VM environment

For Informix Warehouse Accelerator, there is no difference between running in a VM environment or in an SMP environment. The process of estimating the amount of memory that is required and the processor resources is the same. For Informix Warehouse Accelerator to deliver the expected performance in a VM environment, the VM must be granted exclusive access to the physical resources. This applies to both memory and processor resources. For more information, see 3.1.3, “Virtual machine environment” on page 42.

3.3.5 Network connectivity

In production environments, Informix Warehouse Accelerator often runs on its own machine that is separate from the Informix server. The Informix server must be able to contact Informix Warehouse Accelerator directly for all the interactions between the two servers. The Informix server sends requests to accelerate queries and performs administrative operations on data marts. To achieve the best performance for all interactions, a dedicated network connection with high bandwidth is required.

Sending queries for acceleration from the Informix server to Informix Warehouse Accelerator usually does not require much bandwidth because only the SQL statement text is sent to the accelerator server. However, to return the query results from Informix Warehouse Accelerator back to the Informix server can require more bandwidth. The process of returning a large result set from the accelerator server to the Informix server can require so much extra resources that the performance gain by the acceleration itself is noticeably diminished. The optimal balance for query acceleration with Informix Warehouse Accelerator is complex queries that produce a small result sets.

When you load data into Informix Warehouse Accelerator data marts, a high-speed connection between the Informix server and Informix Warehouse Accelerator is even more important. All of the tables and their selected columns that form a data mart, and all of the rows, are read by the Informix server and transferred to Informix Warehouse Accelerator during the data mart load operation. With a well-tuned Informix server and an Informix Warehouse Accelerator that has enough processor resources to process loads, the network connection can become the bottleneck for the data load. Informix Warehouse Accelerator compresses the data to the final format and stores it in memory. During data load, the raw data is transferred from the Informix server to Informix Warehouse Accelerator. Usually, this raw data is much larger in size than the final data mart in memory.

When you run Informix Warehouse Accelerator in a cluster environment, only the cluster node of the coordinator requires a connection to the Informix server. The Informix server never directly communicates with any of the worker nodes and so they do not require a connection to the Informix server. For the worker nodes, a high-speed connection between them and to the coordinator node is required, for example, a 10 Gigabit Ethernet or faster connection.

4

IBM Informix Warehouse Accelerator installation and configuration

With the appropriate Informix Warehouse Accelerator data mart design and sizing considerations in place, you are now ready to deploy Informix Warehouse Accelerator. The process of deploying Informix Warehouse Accelerator involves a series of installation and configuration steps that are relatively straightforward compared to the installation and configuration of the Informix server. Informix Warehouse Accelerator is an accelerator to the Informix server and you must consider parameters that connect the two components.

4.1 Installation of the Informix Warehouse Accelerator server

You can install the Informix Warehouse Accelerator server, which we refer to as *accelerator server*, on a single computer or on a cluster, such as a blade server. Informix Warehouse Accelerator runs on the latest Linux operating system, either the Red Hat (RHEL) or SUSE distribution. The underlying processor architecture of the hardware must be Intel x86-64. It can be from any vendor, for example, IBM, HP, Fujitsu, or others.

The decision to deploy on a single computer or a cluster depends on a number of factors, including the expected database growth rate, the amount of available DRAM on a single computer, and the ease of administering a single computer versus a cluster. For more information about the sizing and layout of the hardware environment, see Chapter 3, “Designing and sizing an IBM Informix Warehouse Accelerator environment” on page 35.

4.1.1 Installation of Informix Warehouse Accelerator: Single computer

The installation of Informix Warehouse Accelerator on one computer means that all of the specific Informix Warehouse Accelerator coordinator and worker nodes share the memory and processor resources. You can use either the computer where your Informix server is running (if the computer is running on 64-bit Linux) or establish a new computer that is dedicated for Informix Warehouse Accelerator.

To install Informix Warehouse Accelerator, you must obtain a copy of the Informix Advanced Edition (either Workgroup or Enterprise). Each edition consists of the Informix server, Informix Warehouse Accelerator, and IBM Smart Analytics Optimizer Studio. Advanced Enterprise Edition contains an additional license to use the Storage Optimization Feature on the Informix server to compress data and indexes.

To install Informix Warehouse Accelerator, complete the following steps:

1. Define the user where you want Informix Warehouse Accelerator to run on, that is, “Informix” or “root”.
2. Download the installation media.

   In case you want to manage the Informix Warehouse Accelerator by the “informix” user, create the user on the installation computer.
3. Create an installation directory.

4. Extract your media file into a temporary directory, for example, `/tmp/IWA`.
5. Run **`iwa_install`** from the directory where you extracted the media:
   - Specify your installation directory during the installation process.
   - During the installation, you can adjust various configuration parameters.

- Add the <*`your_installationdir`*>`/bin` directory to your PATH variable.

You are not required to install the IBM Smart Analytics Optimizer Studio, which is also packaged with the installation media. This package contains a tool that you can use for interactively designing Informix Warehouse Accelerator data marts. If you want to use IBM Smart Analytics Optimizer Studio, install this tool either on a Windows based computer or on a Linux 32-bit computer separately.

After a successful installation, the required executable files, libraries, and configuration files are deployed. If you specified the Informix Warehouse Accelerator configuration parameters during the execution of **`iwa_install`**, you can continue with the setup of the accelerator server. If you did not specify the configuration parameters, you must manually apply the required changes in the configuration files before continuing with the setup.

4.1.2 Installation of Informix Warehouse Accelerator: Cluster

By using a cluster, you can spread the Informix Warehouse Accelerator nodes across different computers. By doing so, you can dedicate memory and processors to the Informix Warehouse Accelerator coordinator and worker nodes and avoid possible concurrency issues. There are requirements when installing on a cluster that are in addition to those required for installation on a single computer.

You must consider the following items when you are installing on a cluster:

- You must have a cluster file system and make sure that the computers in the cluster have access to the installation directory. Currently, the most common cluster file systems on Linux are *ocfs2* or *gfs2*. Either file system is acceptable.
- You must set up an SSH connection that does not require a password. The communication between the Informix Warehouse Accelerator nodes across the used systems in the cluster is run by **`ssh`**.
  - A common setup is to use the public-private key pair.
  - You can generate a private or public key with **`ssh-keygen`**.
  - You can distribute the public key to the appropriate target systems and the `.ssh/authorized_keys` file.

After you successfully set up the cluster file system and the SSH password-less communication, you can continue with the installation routine as described for installation on a single computer.

4.2 Configuration tasks

After you install Informix Warehouse Accelerator, you must update the Informix server and Informix Warehouse Accelerator configuration files.

4.2.1 Configuring Informix Warehouse Accelerator

You must apply all configuration parameter changes to the `dwainst.conf` file, which is in the installation subdirectory `dwa/etc`. Because there is a single installation directory for either an SMP or cluster configuration, you are required to apply the changes only to a single file.

General configuration parameter settings

You must customize the `dwainst.conf` file parameters that are described here.

Storage-related parameters

Informix Warehouse Accelerator requires a directory where the distribution is installed and another directory that is specified by the **`DWADIR`** parameter. It is important to create the `DWADIR` directory before the initial setup of the accelerator server.

The **`DWADIR`** parameter specifies the storage directory for Informix Warehouse Accelerator. The accelerator server uses the directory to maintain the following items:

- Log files
- Configuration files for the local nodes
- Configuration files for the cluster
- Current data mart definitions
- Information that is exchanged between coordinator and worker nodes

Processor and workload-related parameters

The accelerator server uses different processes to accomplish specific tasks. These processes can be configured by setting the value for these parameters:

- **NUM_NODES**
  - This parameter specifies the number of nodes in this accelerator server.
  - The default value is two nodes: a coordinator and a worker node.
  - The underlying processes are multithreaded so a single worker node can parallelize data load and query acceleration.
  - Increasing the value of **NUM_NODES** increases the number of worker nodes. An increased number of worker nodes has the following impacts:
    - More memory is required because dimension tables are duplicated in each worker node.
    - The workflow between the coordinator and worker nodes is increased.
    - Each worker node process creates more threads that affect applications that are running parallel.
- **CPU_PERCENTAGE_LOAD**
- **CPU_PERCENTAGE_SCAN**

For more information about sizing and parameter settings, see 3.3, “Sizing” on page 44.

Memory-related parameters

Memory usage is a critical component of the accelerator server. All data mart definitions and data are stored in shared memory. Additional private memory is also required for other processes. It is important to distinguish the usage of the accelerator server memory from the usage of Informix server memory.

To calculate an initial sizing estimate for the Informix Warehouse Accelerator memory parameters, complete the following steps:

1. Determine the total available memory on your computer.
2. Determine the amount of memory that is required for non-accelerator server operations and application requirements.
3. Determine whether there is sufficient memory for accelerator server data and processing requirements.

After you determine the sizing requirements, apply these changes to the following parameters:

- **WORKER_SHM**
  - This parameter specifies how much shared memory can be used by all worker nodes.
  - A general guideline is that you can assign 60% of the memory that is available to the accelerator server.
- **COORDINATOR_SHM**
  - This parameter specifies the amount of shared memory that the coordinator node can use.
  - A general guideline is that you can assign 7% of the memory that is available to the accelerator server.

Keep the remaining memory available as private memory space for other accelerator server-related processes.

Network-related parameters

To facilitate operations between the Informix server and accelerator server, you must specify the **START_PORT** and **DRDA_INTERFACE** network parameters.

- **START_PORT**
  - This parameter specifies the starting port number for the coordinator and worker nodes.
  - The number of ports that are used depends on how many nodes are configured in the accelerator server environment.
- **DRDA_INTERFACE**
  - This parameter specifies the network interface name that is used for the connection from the Informix database server to the accelerator server.
  - On a local computer that has the Informix server and accelerator server, use the `lo` (local loopback) interface.
  - If the accelerator server is installed separate from the Informix server, either in a cluster or on a single computer, the value depends on which network interface is configured.
  - Use the Linux **ifconfig** system command to determine the name. For example, the name can be `eth0`.

    Here is the possible result of running an **ifconfig** command:

```
root@iwa_host:/# ifconfig
eth0     Link encap:Ethernet  HWaddr f0:de:f1:08:86:41  .....
lo       Link encap:Local Loopback  .....
```

Additional requirements in a cluster environment

In a cluster environment, you must set the **CLUSTER_INTERFACE** configuration parameter in the `dwainst.conf` file.

Network-related parameter

The **CLUSTER_INTERFACE** parameter specifies the network interface name for the connection between the cluster nodes. Run **ifconfig** to determine the name that is configured on your computer.

Memory and processor usage

In a cluster environment, the guidelines for the number of processes (nodes) that must be configured is different from the guidelines for a single computer. The number of nodes is defined by the number of computers in your cluster. Follow these guidelines to configure the **NUM_NODES** parameter for a cluster environment:

- Specify the number of worker nodes that are spread across the defined computers in the cluster. (This includes the coordinator node.)
- Each worker node has a unique set of processor and memory resources.
- You might require additional memory because the dimension tables of your data marts are maintained on each worker node.
- If you add or remove nodes after the initial configuration, you must rebuild your accelerator server configuration and reload of all of your data marts.
- If you must change the number of workers in your configured accelerator server environment because of changes in hardware, complete the following steps:

  a. Change the **NUM_NODES** parameter to a more appropriate number.

  b. Modify the `cluster.conf` file.

  c. Drop all data marts.

  d. Stop the accelerator server by running **ondwa stop**.

  e. Clean the current setup by running **ondwa clean**.

  f. Set up and start the accelerator server by running **ondwa setup** and **ondwa start**. Re-create and load all of the data marts.

Configuring the cluster topology by using the cluster.conf file

After you change the `dwainst.conf` file, you must create a `cluster.conf` file in the `dwa/etc` subdirectory of your accelerator server installation directory. This file contains the computer names that are used in your cluster environment. You can either start with the existing skeleton `cluster.conf.std` file or create a file. Add the host names of the computers to this file. The first entry in the file is the host name for the coordinator node in the new accelerator server. Subsequent computers that are listed in the file are used by worker nodes.

To add a computer to an existing cluster accelerator server installation, you must add the name of the new computer to the file. Additionally, you must change the **NUM_NODES** parameter in `dwainst.conf` file. To make this change available for the accelerator server, follow the procedure in 4.2.1, “Configuring Informix Warehouse Accelerator” on page 58.

Additional information

For more information about the accelerator server configuration, memory requirements, and communication flow between the server components, go to the following website:

https://www.ibm.com/developerworks/community/blogs/2fa81a5c-cb30-4873-b775-1370151e3614/entry/informix_warehouse_accelerator_new_configuration_tips_2?lang=en

4.2.2 Operating system setup requirements

Informix Warehouse Accelerator provides the **ondwa** utility to set up and work with the accelerator server. Before you initially run the **ondwa** utility, you must apply some changes to your operating system. The specific changes that you apply depend on the available memory and the requirements of your environment.

Setting up kernel parameters

You must change the default Linux kernel settings for the **SHMMAX** and **OVERCOMMIT_MEMORY** parameters:

- **SHMMAX**
  - This parameter defines the maximum size in bytes of a single shared memory segment.
  - To permanently set the value, add the following line to `/etc/sysctl.conf` file:

    ```
    kernel.shmmax = <your_value>
    ```

– To temporarily set this value for the current kernel environment, which is valid until the next restart, run one of the following commands:

  - **cat <your_value> > /proc/sys/kernel/shmmax**
  - **sysctl -w kernel.shmmax=<your_value>**

- **OVERCOMMIT_MEMORY**

  – This parameter controls the overcommitment of system memory. Linux allows over allocating memory by using the **malloc()** library call. Processes can allocate more memory than what is available as defined by physical memory and swap space. The accelerator server requires the allocated memory for storing data and schema definition, and for query processing. Therefore, you must ensure that the allocated memory is available.

  – To ensure that the allocated memory is available, set the **OVERCOMMIT_MEMORY** to a value of 2:

  ```
  vm.overcommit_memory = 2
  ```

  You can instead set this value temporarily for the current kernel environment, which is valid until the next restart, by running one of the following commands:

  - **cat 2 > /proc/sys/vm/overcommit_memory**
  - **sysctl -w vm.overcommit_memory=2**

Changing the ulimits in your administrator user environment

The maximum locked memory limit of your administrator user must be set to `unlimited`. However, the default setting usually is `64`. To run the accelerator server in a root environment, you must temporarily change this value as follows:

```
#set the user limit temporarily as root with
ulimit -l unlimited
```

If you want to permanently run the accelerator server as user root or as user informix, you must modify the `/etc/security/limits.conf` file as follows:

- Using informix as administrator:

  ```
  informix soft memlock -1
  informix hard memlock -1
  ```

- Using root as administrator:

  ```
  root soft memlock -1
  root hard memlock -1
  ```

After you apply this change for permanent use, you must restart the system. You can view the new limits by running the following command:

```
informix@node1:/> ulimit -a  | grep "max locked memory"
max locked memory       (kbytes, -l) unlimited
```

Setting up the file system

The shared memory that is allocated by the worker and coordinator nodes is maintained by creating files in the `/run/shm` or `/dev/shm` directories, depending on the Linux distribution. These are file systems that are mounted in your Linux environment. The default settings might not be sufficient for your needs.

You can view the current size by running the following command:

```
informix@node1:/> df
Filesystem           1K-blocks      Used Available Use% Mounted on
devtmpfs              24710580       156  24710424   1% /dev
tmpfs                    38934528      5792  38928736    1% /dev/shm
```

You can apply the required size to `/etc/fstab` by adding a line or replacing the size in the existing one. For example:

```
cat /etc/fstab | grep shm
tmpfs                /dev/shm             tmpfs      size=38022m 0 0
```

A sizing guideline for `/dev/shm` is roughly 70% of the physically available memory in the system.

4.2.3 Configuring the Informix server

A smart blobspace (sbspace) is required for the interaction between the Informix server and accelerator server. If you have a sbspace that is defined in your system, check if the available space is sufficient for the additional use by the accelerator server.

If you do not have a sbspace that is defined, create one. You can use the following statement to adjust the name, size, and path name:

```
onspaces -c -S iwa_space -p /dev/sbspace1 -o 0 -s 200000
```

After adding a sbspace, change your `ONCONFIG` file and modify the **`SBSPACENAME`** parameter. Assign the name of your new sbspace to this parameter. You must restart the Informix server to make the changes effective.

4.3 Setting up and starting the accelerator server

After you change the configuration file, you can set up and start your new accelerator server. The Informix product distribution provides an executable Bash shell script that is called **ondwa**, which supports all of the necessary administration tasks.

4.3.1 Setting up Informix Warehouse Accelerator

You are required to run only setup for an accelerator server after the installation and configuration of a new accelerator server or after you apply changes to a parameter in the `dwainst.conf` file. For example, if you change the number of nodes in the accelerator server, you must run **ondwa**.

You can initially set up your new accelerator server by running **ondwa setup** on the command line while using your informix or root administration account.

After the setup is complete, the storage directory that is specified by the **DWADIR** configuration parameter contains the definition files for your configured nodes and the subdirectories that are required for the communication between the nodes. If you did not create the storage directory, you get an error. To correct this error, create the storage directory and rerun the command.

4.3.2 Starting Informix Warehouse Accelerator

You must start the accelerator server (after you complete the setup) by running **ondwa startup**. This command starts all the necessary accelerator server processes.

You can run **ondwa status** to monitor the status of your new accelerator server. For example, after successfully starting a small accelerator server, the output might look like the following output:

```
root@iwa_host:/# ondwa status

ID  | Role        | Cat-Status | HB-Status | Hostname | System ID
----+-------------+------------+-----------+----------+-----------
0   | COORDINATOR | ACTIVE     | Healthy   | iwa_host | 1
1   | WORKER      | ACTIVE     | Healthy   | iwa_host | 2

Cluster is in state    : Fully Operational
Expected node count    : 1 coordinator and 1 worker nodes
```

In a clustered environment, the output might look like the following output:

```
informix@node1:~> ondwa status
Retrieving from DWA_CM_node0 on node1:
ID   | Role        | Cat-Status  | HB-Status    | Hostname | System ID
-----+-------------+-------------+--------------+----------+------------
0    | COORDINATOR | ACTIVE      |   Healthy    | node1    | 1
1    | WORKER      | ACTIVE      |   Healthy    | node2    | 2
```

If you review the list of processes that are running on your computer, you see several new processes that are running, indicating that the accelerator server successfully started. There are two types of new processes started: the DWA_CM_node processes that represent the several nodes that you configured, and a DWA_watchdog process that monitors the other node processes. The DWA_watchdog process restarts nodes in case of an exception.

```
informix@iwa_host:/$ ps -eaf | grep DWA
informix  7795     1  0 20:34 pts/0    00:00:00 ./DWA_CM_node0 --no-console
informix  7821     1  0 20:34 pts/0    00:00:00 ./DWA_CM_node1 --no-console
informix  7840     1  0 20:34 pts/0    00:00:00 DWA_watchdog -daemon
```

Potential resource problems when initially running ondwa

When you run initially run **ondwa**, you might encounter the resource limitation settings of your operating system. If your Linux kernel parameter for shared memory is set too low, you might see the following error:

```
informix@iwa_host:/$ ondwa start
WORKER_SHM (1153MB = 1209008128B) is greater than value of kernel
parameter shmmax (33554432B), please increase value of shmmax
```

If you encounter a ulimit problem while running **ondwa start**, you might see the following message:

```
informix@iwa_host:/$ ondwa start
Limit for max locked memory is 64.
```

You must set the reported limits to `unlimited`.

You can make the appropriate changes in the operating system environment.

4.3.3 Connecting Informix server and Informix Warehouse Accelerator

The Informix server and the accelerator server are now configured and ready to be connected.

Creating the accelerator

The final setup task is to establish a link between the servers by creating an *accelerator*. For more information about the differences between the *accelerator server* and an *accelerator*, see A.2, “The ondwa utility” on page 219.

To create the accelerator, run **`ondwa getpin`** in the accelerator server environment to get an output similar to the following one:

```
root@iwa_host:/# ondwa getpin
127.0.0.1 23002 7357
```

Then, create the accelerator in the Informix server environment by using values that are taken from the output of the **`ondwa getpin`** command. To create the accelerator, you can use the OAT tool, IBM Smart Analytics Optimizer Studio, or a built-in procedure in the Informix server. For example, here is how you can create the accelerator by using the built-in procedure.

```
dbaccess -e sysadmin << EOF
   execute function  ifx_setupdwa('My_accel','127.0.0.1','23002','7357');
EOF
```

In the example, `My_accel` is the name that is assigned to your accelerator, `127.0.0.1` is the IP address of the computer that hosts the coordinator node, `23002` is the port number that is assigned to the coordinator on the accelerator server host computer, and `7357` is the pairing code that is used for the initial connection from the Informix server to the accelerator server. This process establishes the accelerator and creates an authentication token that is assigned to the accelerator. This token enables communication between the Informix server and the accelerator server. This step is required only once in the lifetime of the accelerator.

After you establish the accelerator, you see some additional lines in the `sqlhosts` file of the Informix server.

```
My_accel group   -       -
c=1,a=6961422d6e697c727b624d3c27286147753c716b68406b4655665d524f21312e6c683d203
52f59696e35232d5d292043595764495c5e4b5b422e2e5834434b4d53367b3d6c4d4d797035
My_accel_1   dwsoctcp        127.0.0.1       21022   g=My_accel
```

These new entries in the `sqlhosts` file enable the Informix server to talk with the accelerator server. Your Informix server and the newly installed accelerator server are now connected.

There are some limitations:

- After you run **`ondwa getpin`**, you must create the accelerator by using one of the Informix Tools within 30 minutes. After 30 minutes, the pairing code expires and you are required to run the command again to get a new pairing code.
- Database client applications are never directly connected to the accelerator server. Do not use the `sqlhosts` entry of the accelerators in your application. The Informix server always initiates the statement executions to the accelerator server by using the accelerator connection information.

Coexistence of accelerators

You can connect different Informix Servers to your new accelerator server by creating more accelerators. To create more accelerators, you must run (in the accelerator server environment) separate **`ondwa getpin`** commands for each new accelerator. Create the accelerator on the dedicated Informix server by using the new connection information that is taken from the output of the **`ondwa getpin`** command. This type of topology can maximize the hardware usage where the accelerator server is running.

Additionally, you can create multiple accelerators by using the same accelerator server with the same Informix server. The additional accelerators are shown as additional entries in the same `sqlhosts` file. This environment provides you with the flexibility to create independent name spaces for data marts that are maintained by the accelerator. In this scenario, you can use the same name for data marts but in different accelerators. This environment provides the function to logically group all data marts. You can create an accelerator for each unit of work that you operate on. For example, you can logically group databases, production job chains, or projects. Some of these groups might have a limited duration. If they expire, it is easy to identify the associated data marts and drop them.

5

Creating IBM Informix Warehouse Accelerator data marts

This chapter contains detailed steps for creating an operational data mart. You can create an operational data mart by using the workload analysis method or the interactive design method. With the workload analysis method, you capture and record the SQL statements of an application and then use tools to define a data mart that is based on this recording. With the interactive design method, you work with an existing database schema by using the IBM Smart Analytics Optimizer Studio GUI application. This chapter also describes how to use built-in functions to create a data mart and run the initial data load.

5.1 Data marts as objects in Informix Warehouse Accelerator

Data marts are the most important objects in an accelerator server environment. From a logical standpoint, data marts are objects in an accelerator within Informix Warehouse Accelerator. An accelerator is a logical entity in Informix Warehouse Accelerator that establishes the connectivity between the Informix server and the accelerator server. On the Informix server, a data mart is always directly associated with the database where it originated. The Informix server uses the data mart definition to determine which queries are accelerated. After you create the data mart, you cannot change the structure.

The accelerator enables the Informix server to connect to the accelerator server, but the existence of the accelerator alone does not accelerate queries. In your data mart definition, you must define which of the database tables and their columns are loaded into the accelerator server, and their relationship. When you are finished defining your data mart, it is created and the data is loaded into the data mart. The creation and data loading processes together are referred to as the deployment of a data mart. For more information about dimensional modeling, and star and snowflake schemas, see Chapter 2, “Designing data marts” on page 15.

The data mart definition is an XML document. Informix Warehouse Accelerator handles XML documents internally so that you do not need to understand or learn the structure of this XML document. The data mart definition is stored internally in the accelerator server. The Informix server keeps metadata about the data mart in the form of Accelerated Query Tables (AQTs), which are described in “Representation of a data mart in the Informix server” on page 81.

All objects (tables and their columns) in a data mart must belong to the same database. However, you can create several data marts from the same database. Because data marts are logical objects in an accelerator, they must have a unique name within that accelerator. The accelerator defines a name space for its data mart objects. If you want to create different data marts with the same name, then you must create them in different accelerators. In this case, you must first create several accelerators for the same Informix server.

The workload analysis method is considered the easiest approach to creating a data mart, and is the method that is most often used. A workload is a collection of SQL statements that are issued to the Informix server that are usually part of a specific job that accesses the database. These jobs are often run by a third-party product such as Cognos, which is used in a business intelligence environment. A job can also be run by an application that you implemented yourself, a script or a report that issues SQL statements, or ad hoc queries.

Here are the major steps in the workload analysis method for deploying a Informix Warehouse Accelerator data mart:

- Identify the jobs that issue star join or snowflake join queries on the database.

  Tip: You might be able to easily rewrite existing jobs to include star join or snowflake join queries.

- Run workload analysis:
  - Capture and analyze all the statements that are run by your selected jobs.
  - Identify all of the SQL statements that are not acceleration candidates.
  - Apply changes to these statements.
  - Restart the capture and analysis.
- Create the data mart:
  - You can automatically create the data mart by using the probe data that is generated as a result of the workload analysis
  - You can create the data mart by using IBM Smart Analytics Optimizer Studio.
- Run the initial data load.

5.2 Identifying workloads in your business environments

Database servers are one of the central resources in the IT infrastructure of companies. Often, they host many different database applications that are used for multiple purposes. These applications can be web-based applications, SQL scripts, reporting applications, ad hoc queries from various sources, managed cubes for OLAP, warehouse operations, and all OLTP-oriented applications that support business operations. For example, hotlines, retail stores, customer relation activities, and workflow processing.

Out of the many different database applications, you must identify the applications that can benefit most from using Informix Warehouse Accelerator acceleration for their workload. In general, Informix Warehouse Accelerator can accelerate only **SELECT** type queries. Changes to data, including **INSERT**, **UPDATE**, and **DELETE**, cannot be accelerated. Therefore, applications that meet any of the following requirements are good candidates for acceleration:

- Applications for business intelligence (BI). For example, BI applications that are used for reporting purposes, which include some star join or snowflake join queries, as described in Chapter 2, “Designing data marts” on page 15.
- Applications that do not require the most up-to-date data. Informix Warehouse Accelerator data marts are loaded with a snapshot of the data in the database and can be refreshed in intervals, but they do not reflect data changes of OLTP in real time.
- Applications that do OLAP in an OLTP environment. You can transform the OLTP database schema to a dimension schema to enable acceleration for these applications.
- Applications that rely on purpose-built summary tables and cubes, for example, to aggregate data. With Informix Warehouse Accelerator, these applications can be accelerated without the need for cubes or summary tables.
- Applications with an urgent need for performance improvements. This can be an application with a run time that exceeds your schedule or a query that never finishes. For example, a query might not finish because it is run with an ineffective query plan or requires too much I/O or resources of memory or processor.

Applications that benefit most from the acceleration capabilities of Informix Warehouse Accelerator are queries with star or snowflake joins that involve fact and dimension tables. Because creating data marts is simple when you use workload analysis, you can try it in environments that are not strictly in this realm. Such applications might still benefit from the acceleration capabilities of Informix Warehouse Accelerator.

5.3 Data mart creation by using workload analysis

After you identify all of your database applications that meet the acceleration requirements for Informix Warehouse Accelerator, you can create the data mart.

5.3.1 Workload analysis methods

Database applications contain a set of SQL statements that process your data. To accelerate these SQL statements with Informix Warehouse Accelerator, you must capture all of the statements so that the analysis tool can determine the following information:

- The query statements that meet the criteria for acceleration (described in Chapter 6, “Query execution” on page 103.
- The tables, their columns, and their relationships that are used by these query statements.

With this information, the analysis tool then creates the data mart definition and the data mart. Here are two techniques for using the workload analysis method and tool:

- Before you create the data mart, determine which statements in the workload cannot be accelerated. Change these statements so that they can be accelerated, and then repeat the workload analysis so that you have a complete set of information. The resulting data mart contains all of the required tables and columns for the queries that can be accelerated. This approach is considered the easiest.
- Deploy the data mart that is based on the analysis of the initial workload capture. When the data mart is ready for acceleration, you rerun the application to check the performance of the workload after being accelerated. It is possible that some of the SQL statements in a workload cannot be accelerated because they do not meet the query acceleration criteria.

  You can review these non-accelerated statements and change them so that they can be accelerated with the existing data mart. However, this approach might not produce optimal results because the data mart definition remains fixed according to the initial workload analysis. Even after you change the statements, it is possible that they still cannot be accelerated because some of the required tables or columns are not in the data mart.

Using the OpenAdmin Tool (OAT), you can view the list of captured query statements. This list indicates which statements can be accelerated. In OAT, this list is available before you create the data mart. Using this list, you can identify statements that cannot be accelerated and rewrite them. You can rerun the workload capture and analysis up to this point and then verify that the changed statements can be accelerated by viewing the updated statement list. You can repeat this cycle several times. When you are satisfied with the number of statements that can be accelerated in your workload, continue on to creating the data mart.

5.3.2 Workload Analysis by using OAT

OAT is a browser-based GUI that supports all major database administration tasks on local and remote database servers. In addition, you can now maintain your Informix Warehouse Accelerator data mart objects by using OAT.

Creating a data mart by using OAT is done by using the workload analysis method. In OAT, you can initiate the capture of SQL statements in the workload with a simple click of a button. Internally, this enables the SQL Trace function of the Informix server and defines the starting point of the workload in the trace. After you initiate the SQL statement capture in OAT, you can run the workload independently, whether it is an entire application, a specific action within an application, or a set of SQL statements in a SQL script that you can run by using the **`dbaccess`** utility. When the workload is finished running, you can stop the SQL statement capture in OAT with a simple click of a button. Internally, this action defines the stopping point in the tracing and stops the data capture. OAT uses built-in functions of the database server to determine for each captured SQL statement whether it is possible to accelerate it by using Informix Warehouse Accelerator. This process might take a few moments. When finished, OAT provides a table of all statements that are captured, and indicates which of these statements can be accelerated.

To analyze your workload with OAT, complete the following steps:

1. In the OAT menu, click **SQL Toolbox** → **Schema Manager**.
2. Select the database that you want to use to create the new data mart.

3. In the Actions menu, click **Data Marts** → **Create a Data Mart**, as shown in Figure 5-1.

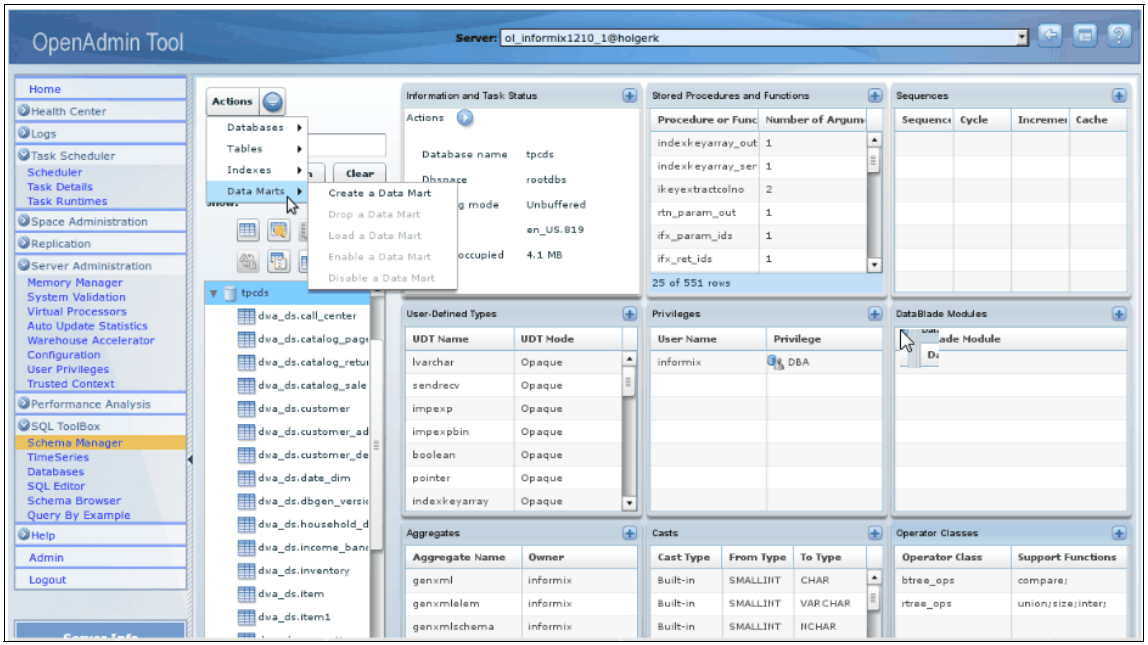


Figure 5-1 First step to create a data mart object

4. Specify the details for the workload capture, as shown in Figure 5-2.

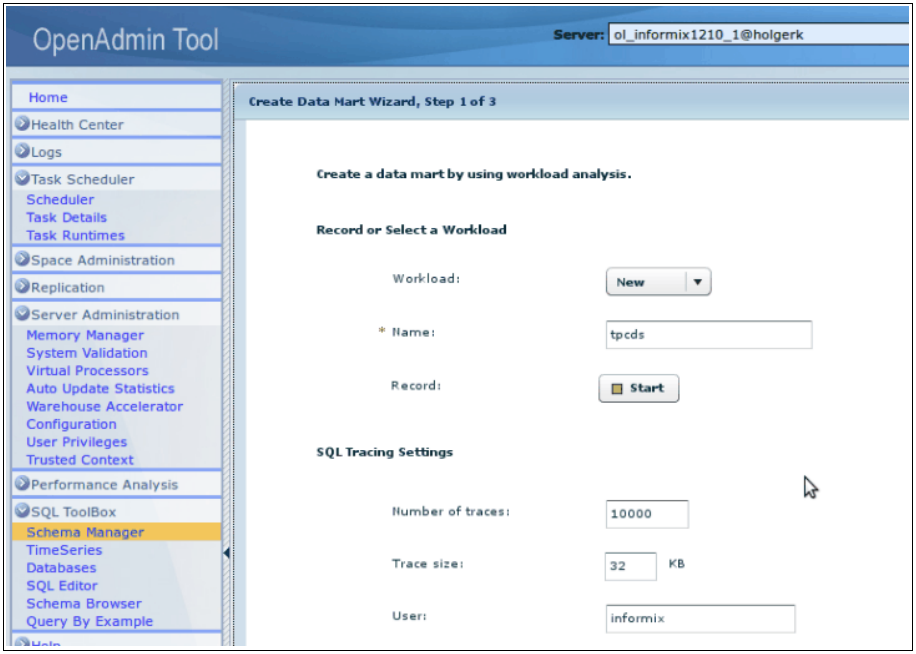


Figure 5-2 Specifying details for the workload capture

This is the starting point for your workload analysis process. The analysis of the captured statements is based on the Informix server SQL Trace function. You can check whether SQL tracing is already active in your Informix server instance. When SQL tracing is active, there is a risk of overwriting active settings by the workload capture process. If SQL Tracing is active, you can save the current settings and re-establish them after you finish creating the data mart.

5. Click **Start** to begin SQL Tracing by using the options. Then, run your workload by using your database client application in whichever mode is required by the application.

You can monitor how many statements are captured during the execution of the database client application. Figure 5-3 shows a sample output of the progress tracking.

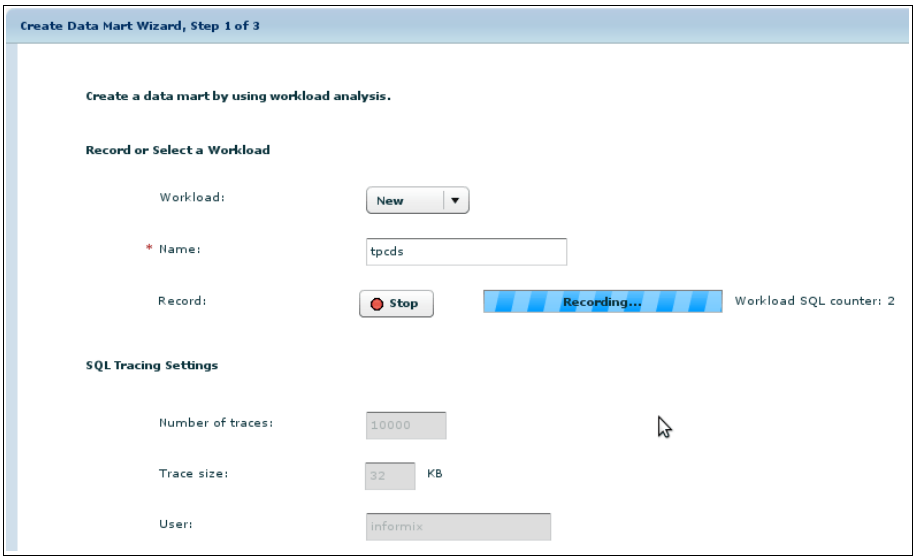


Figure 5-3 Trace in progress

If you want more detail about which particular statements are run by the workload, run **`onstat -g his`** in the database server environment. The output of this command shows the statements themselves, their iterator tree, statistics for time, memory consumption, I/O and logical logs, the number of rows that are processed, and so on.

6. Click **Stop** when your workload run is finished. The analysis of the SQL statements determines which query statements can be accelerated and is referred to as *query probing*.

7. Click **Next** to review the workload. You are shown a summary of all statements that were captured, and whether the statements are candidates for acceleration. Figure 5-4 shows an analysis of a sample workload that contains three SQL statements.

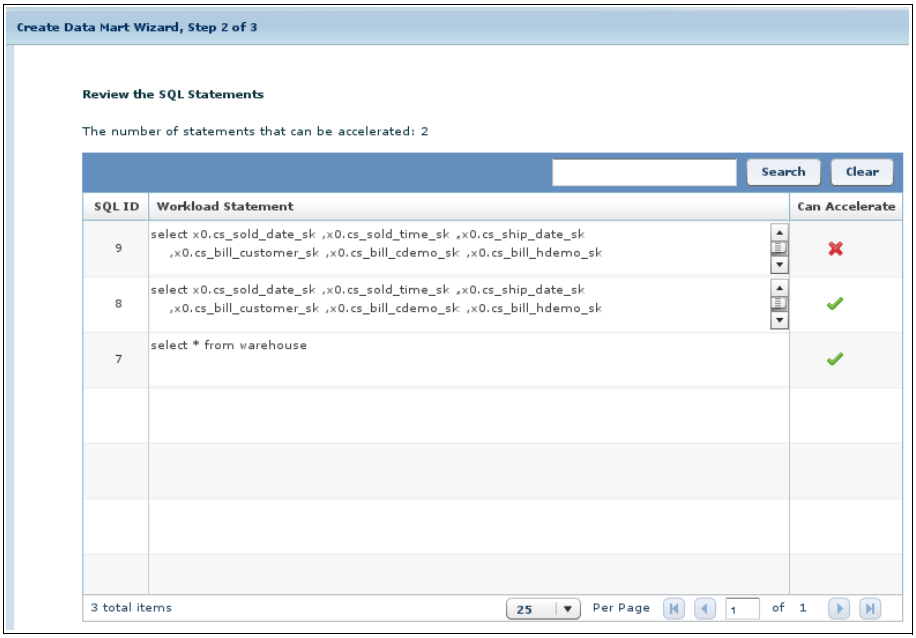


Figure 5-4 Summary of a workload analysis

8. If there are queries that cannot be accelerated, determine the reason why and make modifications. Here are some reasons why queries cannot be accelerated:
   - Does the query statement use temporary tables? Temporary tables cannot be part of a data mart object.
   - Does the query statement reference objects that are in other databases? The data mart is bound to a particular database and cannot reference objects in other databases.
   - Does the query use a join form besides equijoin? Only queries that use equijoins are supported. For example, `table1.column1= table2.column2`, can be accelerated.
   - Do the base tables of the query contain unsupported data types such as BLOBs or BOOLEAN? Use only supported data types.

   For more information about accelerable queries versus non-accelerable queries, see Chapter 6, “Query execution” on page 103.

9. After you apply the changes to the statements or database schema, drop the table from the previous workload run. This data is kept in your target database in a standard table that is named `dwa_saved_workloadtab_<your_workload_name>`. If you do not drop this table, the data from the new workload run is appended to the existing data.

The process of the workload analysis is followed by identification and implementation of changes in the database schema, the SQL statement logic, and job logic. This process usually requires multiple iterations. The time that you spend in this cycle depends on the number of statements in the workload. It might not be possible to get all statements into a form that can be accelerated. However, you can still achieve good overall acceleration if most of the long runtime statements in the Informix server are accelerated.

If you run third-party applications, there are not many options for changing the SQL statements themselves. Some third-party application environments offer you the flexibility to influence the flow characteristics, for example, you can set a preference to use persistent objects rather than temporary tables. However, the process here creates a data mart that is based on the collected data, rather than a repeated actual analysis process.

5.3.3 Data mart deployment by using OAT

The Informix server collects a set of query statements during workload analysis that are stored in the workload table within the database. The workload table should contain only statements from the last iteration of workload analysis. In the next step, OAT uses this data and several built-in functions of the Informix server to generate an internal definition of the data mart. Later, OAT creates the data mart from this internal definition.

Creating the data mart

Continue from the workload analysis in OAT to create the data mart definition. When you create a data mart, you must decide whether you want to create only the data mart or if you want to create the data mart and load the associated data immediately. If you create only the data mart, the Informix server does not yet consider this data mart for query acceleration because it is in a “Load Pending” state. While the data is not loaded, the data mart remains empty and is not used for query acceleration. See Figure 5-5.

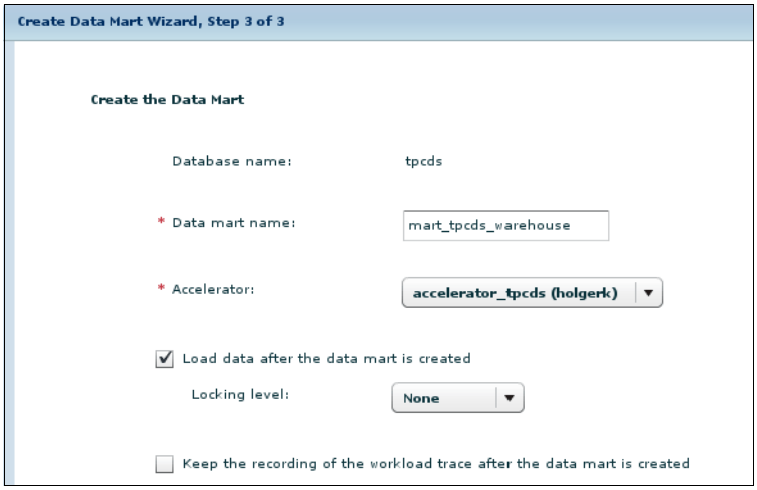


Figure 5-5 Deploy the data mart using OAT

If the Informix server is processing other work, separating the data mart creation from the data load can be useful. For example, while you are creating the data mart, the data in the Informix server database can change continually. To load a consistent state of data into a data mart, you must lock all of the tables that are part of the data mart for the entire duration of the load. However, locking all tables can cause intolerable wait times for other users. Another consideration is that the data load process requires the Informix server to read and transfer all of the data from database tables to the accelerator server. This process can require an intensive amount of I/O and might slow down other operations. Therefore, schedule the data load during non-peak times or when no concurrent workloads are changing the relevant tables.

In the Create the Data Mart step of the wizard, you can specify whether to retain the captured workload data after you create the data mart. The retained data is used as a template for creating data marts in the future. The future workload analysis appends new query statements to the existing data. If you plan to start with a new and empty base, do not select the **Keep the Recording** option.

Figure 5-6 shows, in the schema manager of OAT, that the data mart is created and the data load is deferred. The data mart is in the “Load Pending” state.

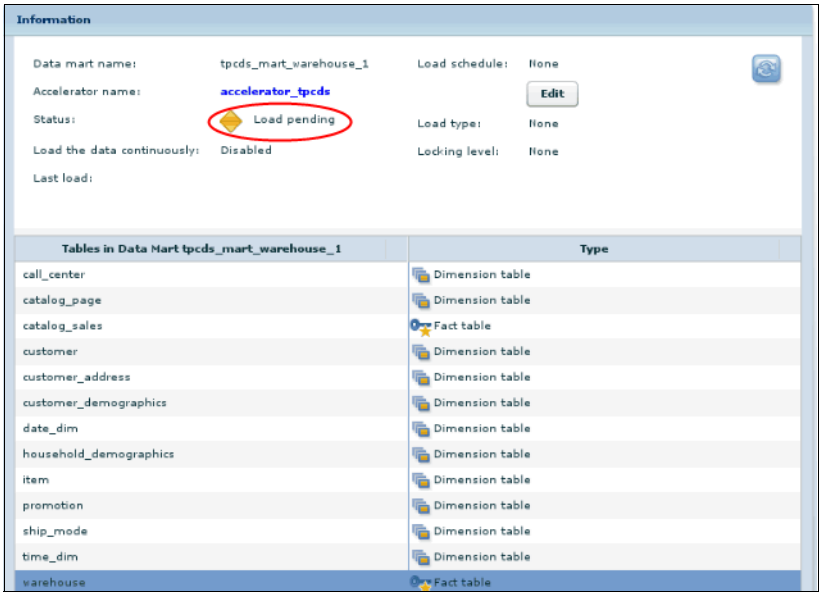


Figure 5-6 The new data mart without data in the schema manager of OAT

Representation of a data mart in the Informix server

The Informix server stores metadata from the new data mart in the database where the data mart was created. The Informix server uses the metadata to make decisions about whether to accelerate a particular query and which data mart to use for the acceleration. The metadata is created as internal views in the system catalog of the database. The names of these views are prefixed by `aqt`, which stands for Accelerated Query Tables (AQTs). After the creation of a data mart, you can check the systables table in the system catalog of the database on which you created the data mart. There are one or more such views for each data mart. For more information about the view definition, you can review the sysviews table. These views are only for internal use and they cannot be used in a query.

Example 5-1 shows a view of the metadata of a data mart:

Example 5-1 Data mart metadata

```
select a.tabname,b.viewtextfrom systables a, sysviews b
where a.tabname matches "a*" and a.tabid=b.tabid

tabname   aqtf820d692-8d7a-4304-b7c7-27dee28b964a
```

```
viewtext  create view "informix"."aqtf820d692-8d7a-4304-b7c7-27dee28b964a"
("COL02","COL03","COL04","COL05","COL06","COL07","COL08","COL09","COL10",
          "COL11","COL12","COL13","COL14","COL15") as
select x0.w_city,x0.w_country ,x0.w_county ,x0.w_gmt_offset ,x0.w_state ,
```

Metadata for data mart administration that is independent of a single database is stored in specific tables in the sysadmin database. You can view the data in the tables, but the structure is subject to change because these tables are used internally by the Informix server.

Initial load of the associated data into a data mart

If you do not load the data at the same time as the data mart creation, the data mart can exist and be viewed in OAT, but cannot be used for query acceleration by the Informix server. You must first run a full data load into the data mart before you can use the data mart for query acceleration. While you are loading the data, the Informix server begins an unload of the appropriate table data to an external table. The data is then transferred into the memory of the accelerator server. For more information about the additional functions that identify data changes in the database and refresh the data mart, see Chapter 7, “Managing and refreshing an IBM Informix Warehouse Accelerator data mart” on page 143.

To load a data mart that is in “Load Pending” state, complete the following steps:

1. From the schema manager in the OAT main window, select your database and the appropriate data mart object. Verify that the data mart object is in the “Load pending” state. Select **Actions** → **Data marts** → **Load a Data Mart** (Figure 5-7 on page 83).

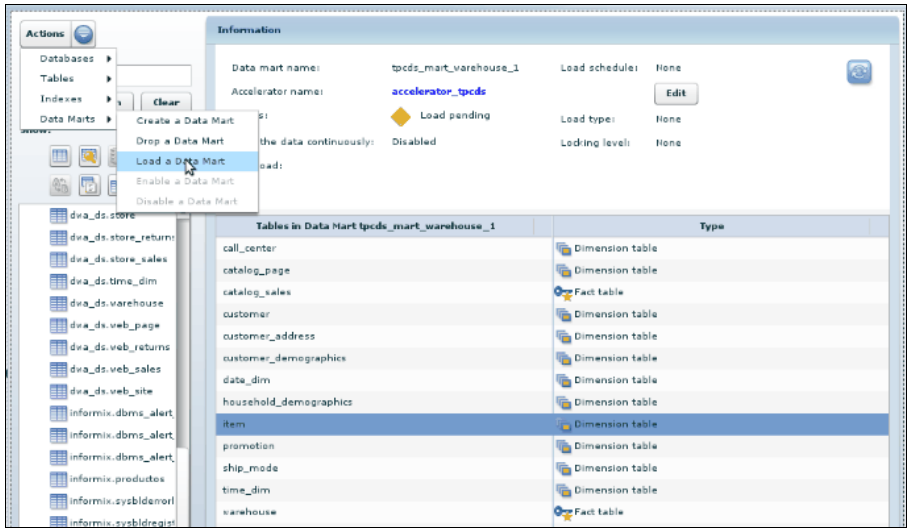


Figure 5-7 Initiate the first load of data into a new data mart

2. Select the **Load the data mart now** check box (Figure 5-8). Loading a data mart can take a considerable amount of time, depending on the size of the data that is loaded.

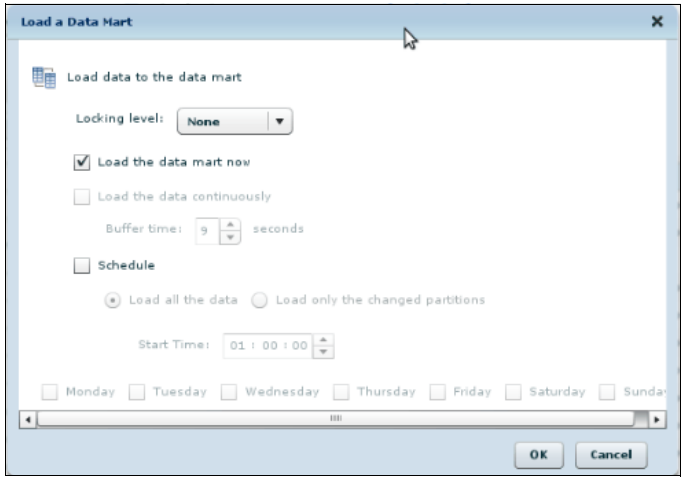


Figure 5-8 Option window for the initial load configuration

3. After the load mart operation finishes, the status of the data mart is changed from “Load Pending” to “Active”, as shown in Figure 5-9.

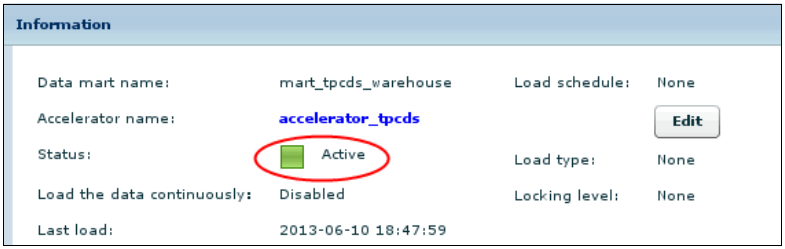


Figure 5-9 Data mart object ready to use for accelerating queries

After you are done with the initial data load, the data mart is operational. The Informix server can begin matching new incoming queries against the AQTs of the data mart. If a query matches one of the data marts AQTs, that data mart can be used to accelerate the query. For a complete description of how to run your queries in an operational data mart environment, see Chapter 6, “Query execution” on page 103.

5.4 Interactive data mart design with IBM Smart Analytics Optimizer Studio

When you do not know which tables and columns are used in your application environment, you generate your data mart definition based on captured workload, as described in 5.3, “Data mart creation by using workload analysis” on page 72. If you know the database schema well and how the application queries the database, you can design a data mart that is based on an existing database schema by using IBM Smart Analytics Optimizer Studio.

The IBM Smart Analytics Optimizer Studio tool is distributed with Informix Ultimate Warehouse Edition. You must install this tool on a Windows system or a Linux 32-bit system. From the installation medium, you extract the subdirectory named `IBM_Smart_Analytics_Optimizer_Suite`. This directory contains two binary files. Depending on your operating system, use the `*.exe` or `*.bin` file for the installation.

Setting up the connection to the database server

To set up a connection to your Informix database server, complete the following steps:

1. Start IBM Smart Analytics Optimizer Studio and close the welcome window. The welcome window is shown the first time that you start IBM Smart Analytics Optimizer Studio.
2. Right-click **Database Connections** in the Data Source Explorer area, as shown in Figure 5-10.

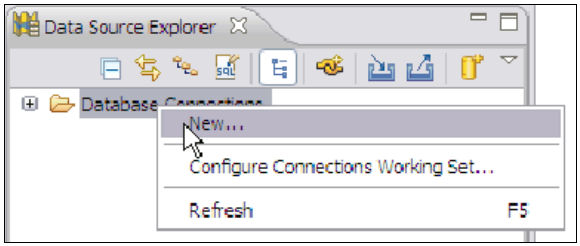


Figure 5-10 Start a new database connection using IBM Smart Analytics Optimizer Studio

3. In the dialog box, specify the connection parameters for your database server. Select a supported JDBC driver type. In this example, **Informix 11.5 - Informix JDBC Driver Default** is used (Figure 5-11).

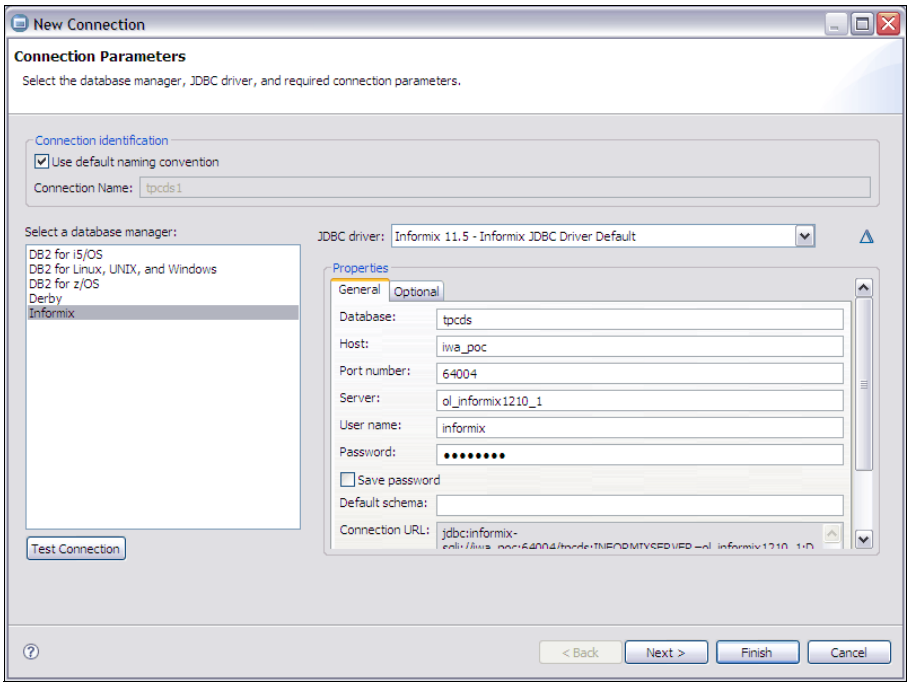


Figure 5-11 Connection parameters for an Informix database server connection

4. After you apply all of the required parameters, IBM Smart Analytics Optimizer Studio establishes a new connection to the database. You can begin designing a new data mart.

Setting up a data mart project in IBM Smart Analytics Optimizer Studio

To set up a data mart project in IBM Smart Analytics Optimizer Studio, complete the following steps:

1. From the new project icon, select **Accelerator Project** and specify the project name (Figure 5-12 on page 87).

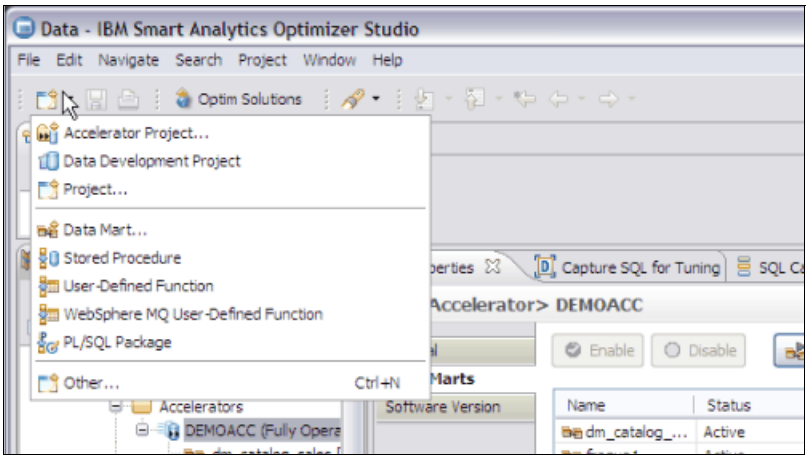


Figure 5-12 Create an accelerator project to start designing a data mart

2. From the Data Project Explorer tab, right-click the new project name and select **New** → **Data Mart**.

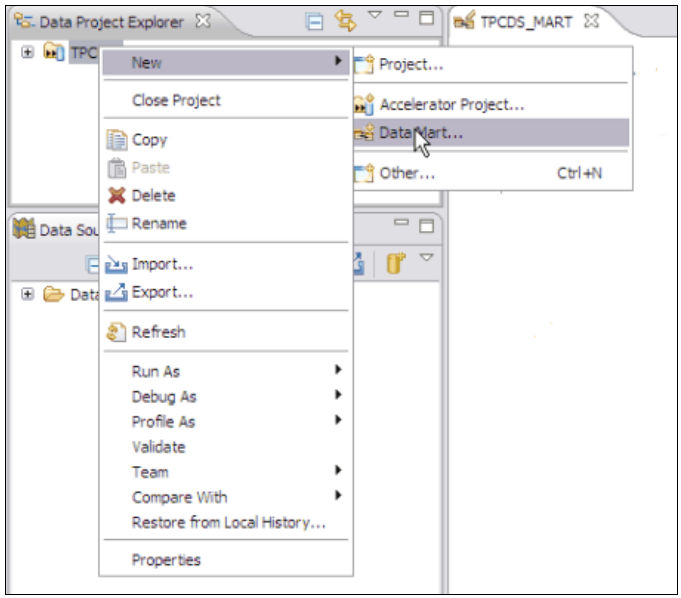


Figure 5-13 Start working on a data mart in your current accelerator project

3. Select the base tables that you want to include in the new data mart. The tables are not immediately visible because the initial listing shows only the various owners of the objects in the database, as shown in Figure 5-14.

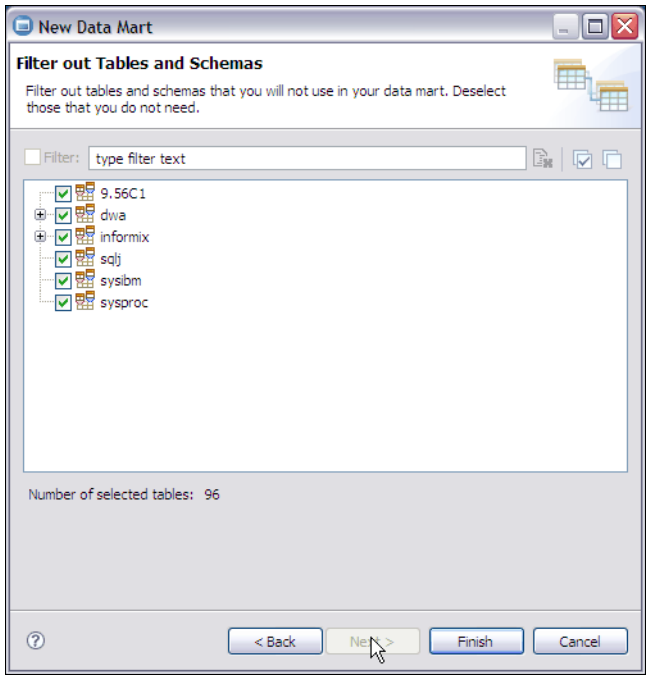


Figure 5-14 Owners of objects in your target database

To view the tables that you want, clear all owners whose objects you do not want to see. Click **+** to expand the list of objects. Clear tables that are not required in the data mart. See Figure 5-15.

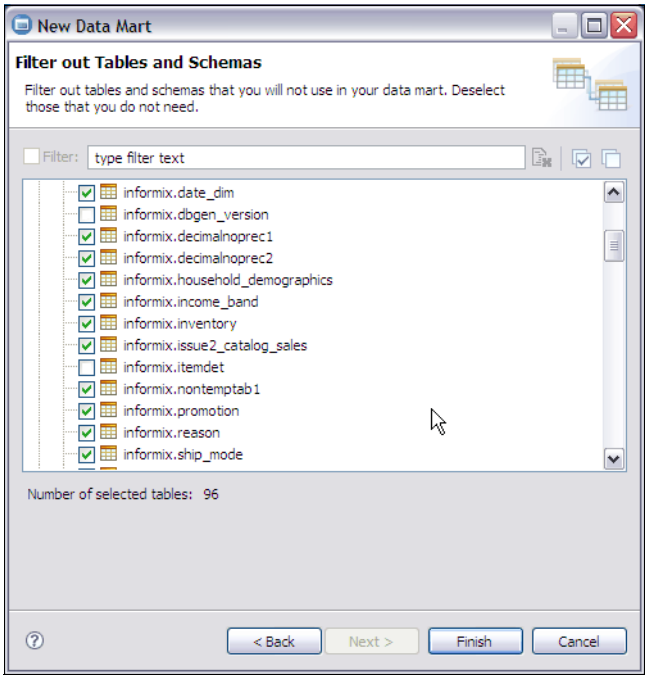


Figure 5-15 Check only the required tables for your mart

The new data mart project is started and the tables that you selected are filtered.

Managing tables, columns, and table relationships

To add the tables to the data mart and define the relationships between tables, complete the following steps:

1. From the project space in the middle of your IBM Smart Analytics Optimizer Studio window, right-click to add one or more tables to the project. See Figure 5-16

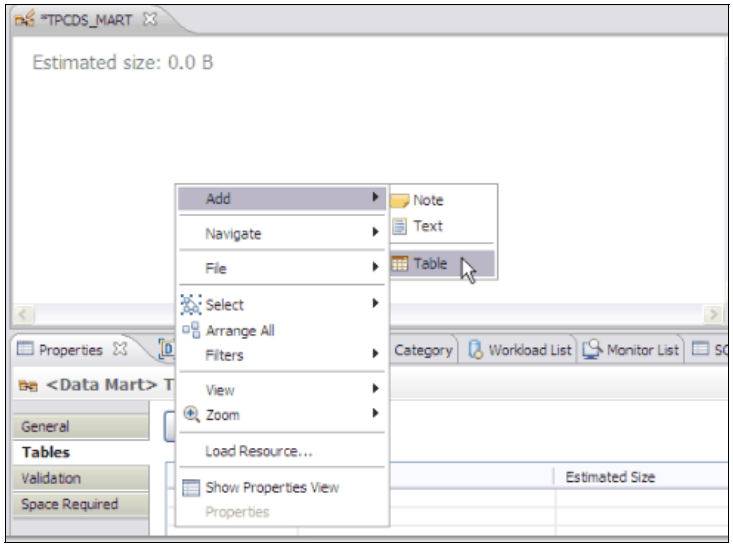


Figure 5-16 Add new tables to the data mart project space

Figure 5-17 shows the project space after tables are added:

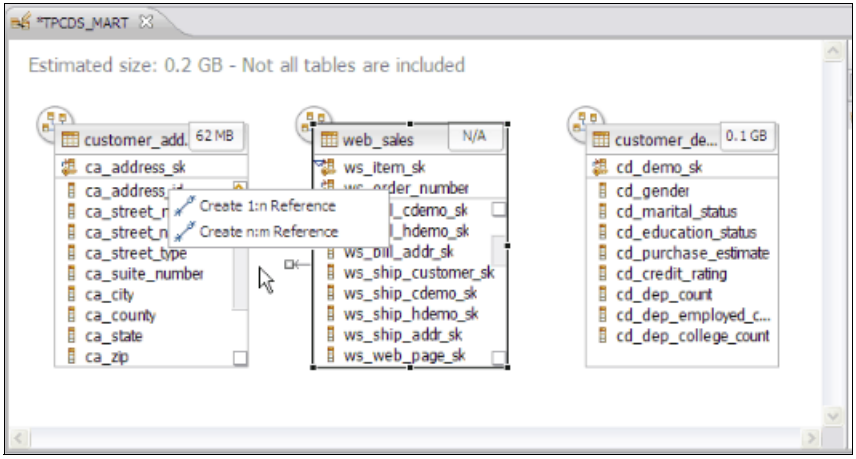


Figure 5-17 New tables added to the data mart

You can now define the relationships. Initially, all of the tables are fact tables, but this status changes when you apply the relationships.

2. Drag the columns from one table to the corresponding columns in the other table to indicate a join. You are required to define each relationship as one-to-many or many-to-many.

   A separate dialog box opens where you connect the columns, as shown in Figure 5-18. Click **Finish** when you are done.

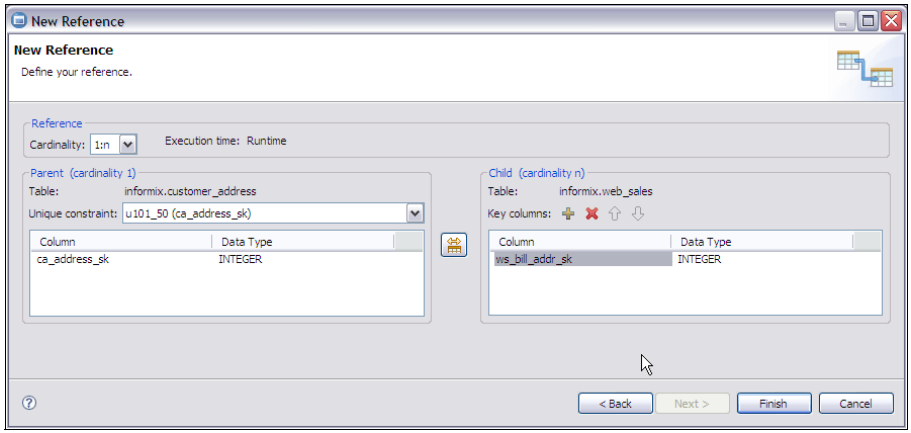


Figure 5-18 Define the relationship between columns of two different tables

3. View the relationships as links between the tables in the data mart project space, as shown in Figure 5-19. The table classifications changed from fact to dimension. In this example, only the web_sales table remains a fact table, and the rest are changed to dimension tables.

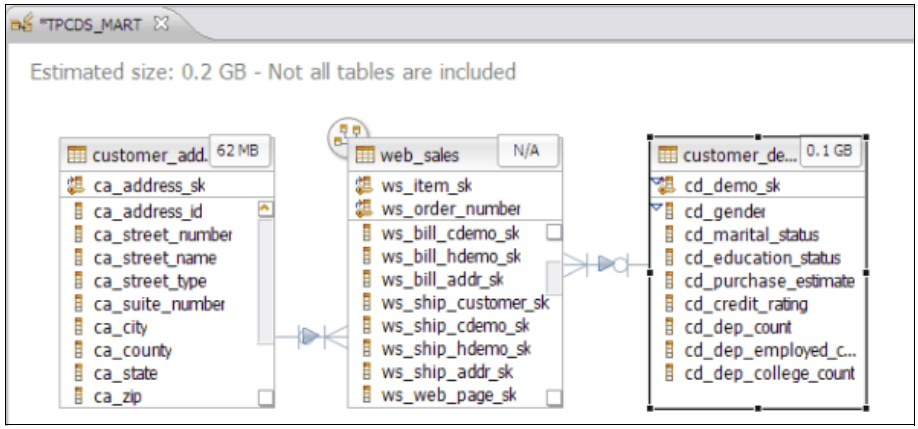


Figure 5-19 Table relationships in IBM Smart Analytics Optimizer Studio

4. Repeat this procedure to add more tables to the current design.

To reduce the size of the data mart that is loaded in to the accelerator server, remove unnecessary columns from the tables in the data mart definition. These columns are not deleted from the base tables in the database, but are removed from the data mart definition.

After you complete the design of your data mart, you can deploy the data mart.

Deploying the data mart and loading the data

To deploy the data mart, complete the following steps:

1. You can either right-click the data mart name in the Data Project Explorer or right-click the accelerator in Data Source Explorer, as shown in Figure 5-20.

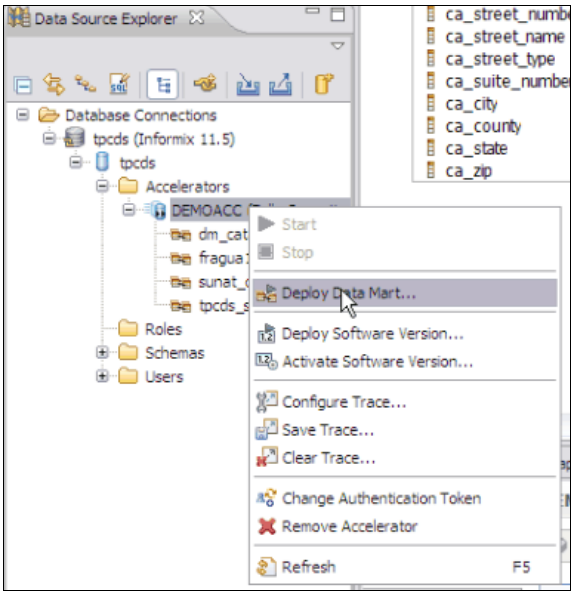


Figure 5-20 Deploy the data mart to the database server

2. Specify the accelerator project, the data mart project, and the accelerator in the Informix server. The accelerator project contains the data mart designs and is an entity of IBM Smart Analytics Optimizer Studio. The data mart project contains the design for a specific data mart. You can save a data mart project locally in IBM Smart Analytics Optimizer Studio before you create the actual data mart in Informix Warehouse Accelerator. The accelerator in the Informix server is an entity that is used for the connection from the Informix server to the accelerator server to administer new and existing data marts in Informix Warehouse Accelerator.

3. In the Deployment dialog, indicate whether to include the initial data load to the deployment process, as shown in the Figure 5-21. You might want to separate the two steps, as explained in 5.3.3, “Data mart deployment by using OAT” on page 79.

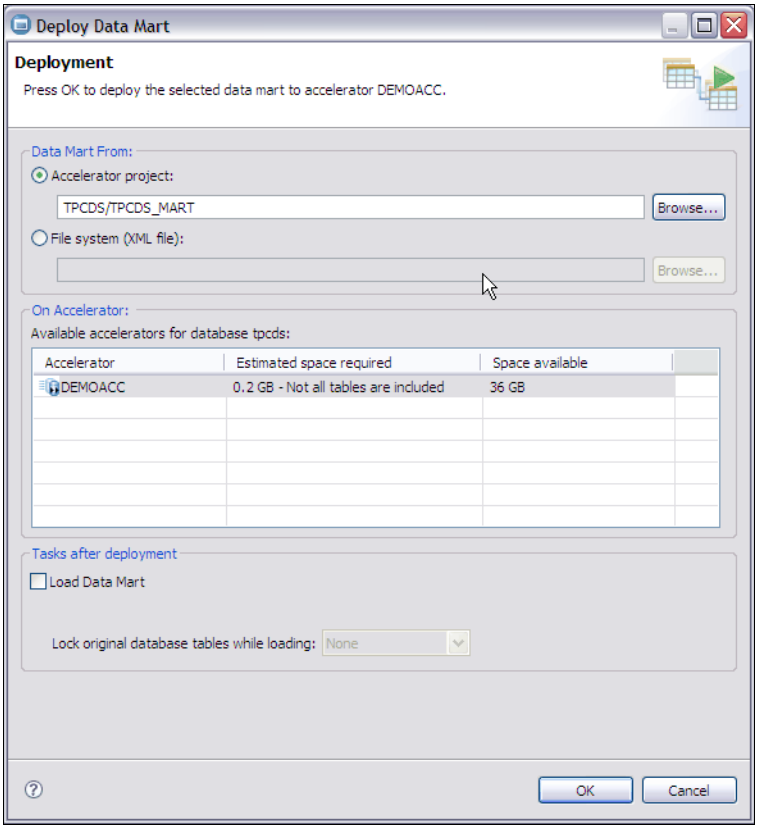


Figure 5-21 Deploy the data mart with an option to load the data

After you create the data mart, the new objects appear in the Data Source Explorer window. Depending on the options that you specified for the data load, the data mart is either “Active” or “Inactive”. For more information about the status of the data mart, see 5.3.3, “Data mart deployment by using OAT” on page 79.

4. Optional: To load the data, right-click the data mart object that is in the “Inactive” status and click **Load**, as shown in Figure 5-22. When the data is finished loading, the status of the mart is changed to “Active”. The database server now considers this data mart as operational and ready for query acceleration.

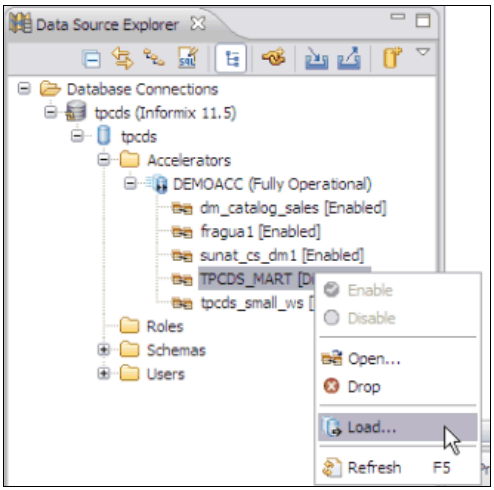


Figure 5-22 Initiate the first load of the data mart

5.5 Embedded data mart creation by using sysdbopen()

The workload analysis and interactive design methods for creating data marts are not suitable for automation and embedding because they have step dependency and iteration. To automatically deploy the data mart and load the data, use the following embedded method:

1. Identify two database users that are not involved with the production environment: user1 and user2.
2. Create user-specific **`sysdbopen()`** and **`sysdbclose()`** stored procedures for user1 to contain calls to the new Stored Procedure API for data mart administration (introduced in Informix Server 12.10). In these stored procedures, specify the start and the stop of the workload capture, followed by the analysis, data mart deployment, and initial data load.

   When the application connects to the database on behalf of the user, the **`sysdbopen()`** stored procedure automatically cleans up the old query probing data and starts the query probing.

When the application disconnects from the database, the **sydbclose()** stored procedure creates the data mart definition from the query probing data, and deploys the data mart and loads the data.

3. As user1, run your database application and optionally measure the performance.
4. Drop the stored procedures **sysdbopen()** and **sysdbclose()** for user1.
5. Create a **sysdbopen()** for user2 to enable query acceleration when the user connects to the database.
6. As user2, run the application again and optionally measure the performance. The new **sysdbopen()** stored procedure automatically enables query acceleration when the application connects to the database.

With the embedded method, you cannot view the list of query statements that can be accelerated or not. However, you can compare the performance measurements, which are taken before and after you created the data mart, to know whether the queries are accelerated.

Example 5-2 shows the **sysdbopen()** and **sysdbclose()** stored procedures for creating a data mart automatically.

Example 5-2 Using sysdbopen() and sysdbclose() for workload analysis

```
-- Stored procedure for user one: probe queries for possible
acceleration:
-- * cleanup previously collected probing data.
-- * start the collection of probing data for the remainder of the
session.
create procedure “user1”.sysdbopen()
   SET ENVIRONMENT use_dwa “probe cleanup”;
   SET ENVIRONMENT use_dwa “probe start”;
-- make the workload probing faster by not executing the statements:
-- set explain on avoid_execute
end procedure;

-- Stored procedure for user one:
-- * create the data mart definition from the collected probing data.
-- * create the data mart using the definition.
-- * load the data mart with data.
create procedure “user1”.sysdbclose()
   execute procedure ifx_probe2mart(“database_name”,”mart_name”);
   execute function ifx_createmart(“accelerator_name”,”mart_name”);
   execute function
ifx_loadmart(“accelerator_name”,”mart_name”,”NONE”);
end procedure;
```

```
-- User two run the application with query acceleration enabled:
create procedure “user2”.sysdbopen()
   SET ENVIRONMENT use_dwa “accelerate on“;
end procedure;
```

If you do not need to compare the run time for your application before and after the data mart creation, you can speed up the generation of the query probing data by using the **avoid_execute()** keyword. This keyword prevents the database server from running the application SQL statements.

5.6 Designing a data mart that is based on SQL trace filtering

In Example 5-2 on page 95, the built-in **ifx_probe2mart()** procedure was used to by the embedded method to create the data mart. The **ifx_probe2mart()** procedure transfers probe data into an internal, relational presentation of the data mart definition. This definition is then used as the basis for the subsequent data mart creation. Normally, this function is used to transfer all of the collected query probing data by specifying the database name and the data mart name in the function call. However, you can transfer only the probing data of a specific SQL statement into the data mart predefinition by using the optional statement ID parameter of this function. As a prerequisite, SQL Tracing must be activated during the query probing. To activate SQL Tracing, use a statement similar to the following one:

```
dbaccess -e sysadmin << EOF
execute function task
(“ set sqltracing on”,1000,1,”medium”,”application_user”);
EOF
```

You can run **onstat -g probe** to view information about individual query statements. Here is an example output:

```
onstat -g probe
DWA probing data for database tpcds:

statement 1778:
columns: tabid[colno,...]
         100[1,2,3,4,5,6,7,8,9,10] f
joins:   tabid[colno,...] = tabid[colno,...] (type) {u:unique}
```

By using the statement ID (1778 in the example), you can obtain the SQL statement and you can get more information by running **onstat -g his** or querying the syssqltrace table in the sysmaster database. The syssqltrace table is a virtual table that points to the internal memory of the Informix server where the SQL Tracing content is stored. You can apply filters when you are querying the table. For example, to create a data mart definition from the probing data of statements that had a run time of more than 10 seconds, use one of the following filters:

```
-- Use only probing data of statements with a run time > 10 seconds
SELECT probe2mart('tpcds','warehouse',sql_id)
   FROM sysmaster:syssqltrace
   WHERE sql_runtime > 10
      and sql_database = “tpcds”;
```

or

```
-- Use only probing data from statements that select from
fact_table_name
SELECT probe2mart('tpcds','warehouse',sql_id)
   FROM sysmaster:syssqltrace WHERE
      sql_statement matches “fact_table_name” and
      sql_statement matches “SELECT”
      and sql_database = “tpcds”;
```

After you apply all of the SQL statements that meet your new data mart requirements, you can create the data mart itself by using the built-in **ifx_createMart()** function.

5.7 Using Informix TimeSeries data in an Informix Warehouse Accelerator environment

In general, time series data is generated by measurement devices, for example, meters that measure and record consumption data of water, electricity, or heating at regular time intervals. Another example is GPS devices that record position data and speed every few seconds. Each data record is associated with a time stamp that indicates when data values were recorded. Storing time series data in a standard relational database requires a large amount of disk space and processor time because each record is stored as a separate data row, along with the ID of the device, and the time stamp of the recording.

With Informix TimeSeries data types, you can store the measurements as a list of values in a single data record for each device. The device ID is stored once in the data record. If the measurements are taken at fixed intervals, it is not necessary to store the time stamp of each measurement. Instead, it is sufficient to store a single time stamp of the first measurement in the data record. The length of the value list is defined by time-based definitions of how many measurements you want to store in a single data record.

Because of their unique structure, TimeSeries data types contain a set of specific functions to process the data, for example, aggregation functions. These functions are optimized to process specific meter values within the same data record. For example, the aggregation of values from a single device over a certain amount of time. However, these functions can take a significant amount of time when processing all values of all meters. This is an area where Informix Warehouse Accelerator can accelerate the processing of TimeSeries data.

You can include TimeSeries data in an Informix Warehouse Accelerator data mart. You must create a representation of the TimeSeries data that conforms to the relational model rather than the TimeSeries internal format. To create a relational representation, you can use the Virtual Table Interface (VTI) that is provided by the Informix database server. VTI tables are a layer in the Informix server that you can use to present data in a different way, similar to a view.

VTI tables do not store the original data in a different format; instead, they provide access to the data as through it was stored in relational tables. Thus, you can use VTI tables for TimeSeries data in data mart definitions and transfer data from the Informix server to the accelerator when the data mart is loaded. This method is efficient because the VTI tables transform the TimeSeries data into the relational format and load it directly into the data mart, where it is compressed again. This technique avoids storing the data in a relational format, which temporarily requires an enormous amount of storage capacity.

TimeSeries provides two functions for creating a VTI table on the base tables:

- **`TSCreateExpressionVirtualTab()`**
- **`TSCreateVirtualTab()`**

The only difference between the two functions is that with **`TSCreateExpressionVirtualTab()`** you can specify specific expressions to represent the data, but **`TSCreateVirtualTab()`** is used to represent the data as is.

Using the example from the demonstration database that is created with the **`dbaccessdemo`** utility, the following example shows how TimeSeries data is different from normal data. The example shows how a VTI table can be created so that you can use the TimeSeries data in a data mart. First, Example 5-3 on page 99 shows the definition of the base table that contains TimeSeries data.

Example 5-3 TimeSeries base table

```
-- Data type definition using TimeSeries.
create row type meter_data
  (
  tstamp datetime year to fraction(5),
  value decimal(14,3)
  );

-- Base table using the TimeSeries data type defined above.
create table ts_data
  (
    loc_esi_id char(20) not null ,
    measure_unit varchar(10) not null ,
    direction char(1) not null ,
    multiplier timeseries(meter_data),
    raw_reads timeseries(meter_data),
    primary key (loc_esi_id,measure_unit,direction)
  );
```

Example 5-4 shows the data that is selected from the TimeSeries base table.

Example 5-4 Example data

```
loc_esi_id    4727354321000111
measure_unit  KWH
direction     P
raw_reads     origin(2010-11-10 00:00:00.00000), calendar(cal15min),
              container(raw_container), threshold(0), regular,
              [(0.092), (0.084), (0.090), (0.085), (0.088), (0.088),
               (0.085), (0.091), (0.083), (0.094), ...    , (1.412)]

loc_esi_id    4727354321046021
measure_unit  KWH
direction     P
raw_reads     origin(2010-11-10 00:00:00.00000), calendar(cal15min),
              container(raw_container), threshold(0), regular,
              [(0.041), (0.041), (0.040), (0.041), (0.041), (0.041),
               (0.055), (0.073), (0.071), (0.068), ...    , (0.023)]

...

28 row(s) retrieved.
```

There are 28 data rows in the example. Each data row contains thousands of values, denoted by the ellipses within the square brackets. This data format cannot be used in an Informix Warehouse Accelerator data mart, so you must define a VTI table.

Example 5-5 shows how to create the VTI table with the **dbaccessdemo** utility. The creation of the calendar that is used for the procedure call in Example 5-5 is not shown here. You can also use the predefined calendars that come with TimeSeries data types, for example, calendar ts_15min for events that occur every 15 minutes.

Example 5-5 Creating a VTI table

```
EXECUTE PROCEDURE TSContainerCreate('raw_container',
  'rootdbs', 'meter_data', 100, 50);

EXECUTE PROCEDURE TSCreateVirtualTab('ts_data_v', 'ts_data',
  'origin(2010-11-10 00:00:00.00000),calendar(cal15min),
   container(raw_container),threshold(0),regular', 0,
  'raw_reads');
```

Selecting data from the created VTI table ts_data_v produces data in a format that is usable in an Informix Warehouse Accelerator data mart. Example 5-6 shows data record samples.

Example 5-6 VTI data

```
loc_esi_id    4727354321000111
measure_unit  KWH
direction     P
tstamp        2010-11-10 00:00:00.00000
value         0.092

loc_esi_id    4727354321000111
measure_unit  KWH
direction     P
tstamp        2010-11-10 00:15:00.00000
value         0.084

...

loc_esi_id    4727354321046021
measure_unit  KWH
direction     P
tstamp        2010-11-10 00:00:00.00000
value         0.041
```

```
loc_esi_id    4727354321046021
measure_unit  KWH
direction     P
tstamp        2010-11-10 00:15:00.00000
value         0.041

...

241920 row(s) retrieved.
```

The VTI table provides the same data in a normal format, one measurement value with each data row. Instead of 28 data rows for the **SELECT** on the table that uses TimeSeries data, there are now 241920 data rows for the **SELECT** on the VTI table.

After you create the VTI tables, you can create Informix Warehouse Accelerator data marts. After you create the data mart, you can load the data mart. The data load uses the VTI tables, extracting the TimeSeries data from the Informix database server. In environments that use TimeSeries data, the amount of data often grows fast because of the continuous inflow of new measurement data. To keep up with this new data, the data load and subsequent refreshes with new data into a data mart is parallelized for TimeSeries data.

6

Query execution

The main goal of deploying Informix Warehouse Accelerator into an Informix environment is to provide dramatically improved query performance. To transparently accelerate the query to Informix Warehouse Accelerator, Informix must qualify the query and then match the query to the correct data mart. Some queries are sent to Informix Warehouse Accelerator fully, some partially, and others not at all. This chapter describes how the query flows from the application to Informix and Informix Warehouse Accelerator, how to enable the application to use Informix Warehouse Accelerator without changing the application, and describes the types of queries that are supported for acceleration by using Informix Warehouse Accelerator.

6.1 Overview

Queries that are accelerated by orders of magnitude with Informix Warehouse Accelerator have the following characteristics:

- Complex, ad hoc queries that look for trends or exceptions that are used to make business decisions.
- Queries that operate on a large subset of the database (unlike OLTP), often by using table scans.
- Queries that contain aggregation functions (for example COUNT, SUM, and AVG) and OLAP window functions.
- Queries that often group by time, product, geography, customer group, or market.
- Queries that involve star-joins of large fact tables with zero, one, or more dimension tables.

Informix can accelerate these queries without requiring applications to change or even recompile. Informix accomplishes this task by verifying the following information before it sends the query to Informix Warehouse Accelerator:

- The tables and their relationships within the query are a correct subset of the data mart that is deployed to Informix Warehouse Accelerator.
- The joins and expressions that are used by the query are of the type that is supported by Informix Warehouse Accelerator.

The Informix optimizer checks for these conditions and then creates a query plan to use Informix Warehouse Accelerator. In some cases, for example, queries that contain OLAP window functions and UNION operators, Informix sends a portion of the query to Informix Warehouse Accelerator for select-join-project processing, gets the intermediate result, and then evaluates the applicable UNION clause, OLAP window functions, ORDER BY, SKIP, and FIRST clauses before it sends the results to the application. Query plan and statistics in the Informix EXPLAIN file show which queries and query blocks were run on Informix Warehouse Accelerator and on Informix.

6.2 Query execution flow

Each part of the query execution is described, from preparing the query on the client to getting the query results back. Figure 6-1 shows the query acceleration sequences with Informix and Informix Warehouse Accelerator.

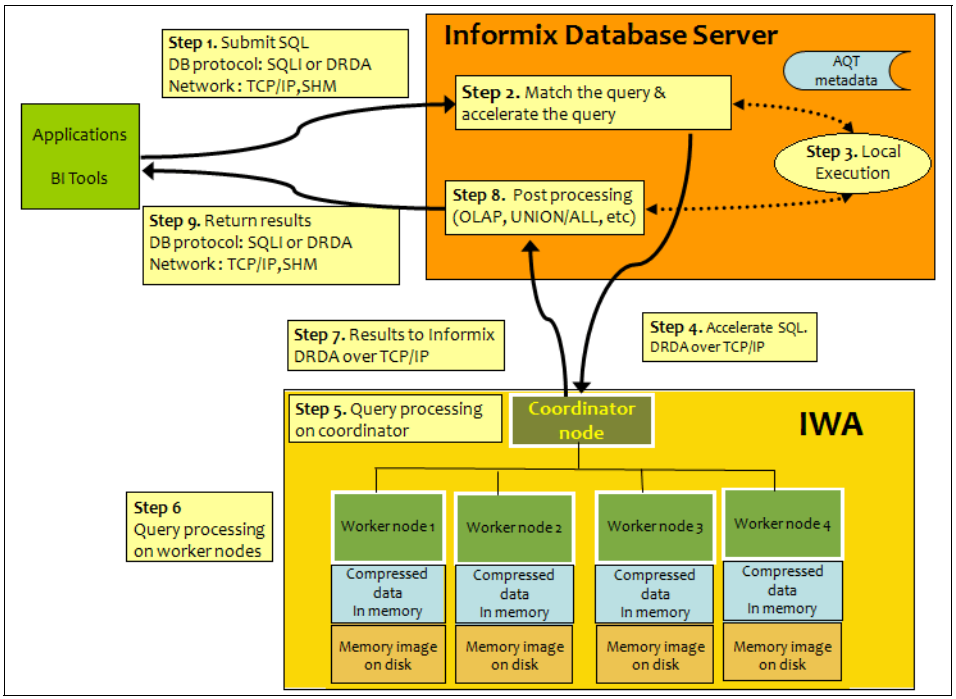


Figure 6-1 Life of an accelerated query with Informix and Informix Warehouse Accelerator

Example 6-1 shows a query that is accelerated by Informix Warehouse Accelerator and its EXPLAIN file.

Example 6-1 Query and EXPLAIN file

```
-- Executed query
> SET EXPLAIN ON;
> SET ENVIRONMENT use_dwa 'accelerate on';
> select count(*) from store_sales;

-- Content of the sqexplain.out file:

QUERY: DWA executed:(OPTIMIZATION TIMESTAMP: 06-09-2013 02:16:49)
------
```

```
select count(*) from store_sales

Estimated Cost: 1
Estimated # of Rows Returned: 1

1) tpcds@FASTR:informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d: DWA
REMOTE PATH

Remote SQL Request:
{QUERY {FROM  informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d}
{SELECT {COUNT_BIG(*) } } }


QUERY: IDS FYI:(OPTIMIZATION TIMESTAMP: 06-09-2013 02:28:39)
------
select count(*) from store_sales

Estimated Cost: 1
Estimated # of Rows Returned: 1

  1) dwa_ds.store_sales: INDEX PATH

    (1) Index Name: (count)
        Index Keys: (count)


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------

  type     rows_prod  est_rows  time       est_cost
  ------------------------------------------------
  dwa      1          0         00:00.00   0
```

6.2.1 Step 1: Submitting SQL on the application or BI tools

Application and business intelligence (BI) tools connect to the Informix database server by using one of the client APIs: ESQL/C, ODBC, JDBC, .NET, 4GL, stored procedure, or other APIs. The applications prepare and run the statements by using the standard programming interfaces.

The application can use any of the network or shared memory protocols and the Informix drivers or IBM DRDA® based IBM common drivers for JDBC, ODBC(CLI), or .NET.

Example 6-2 shows an ESQL/C example.

Example 6-2 ESQL/C

```
$PREPARE s1 FROM
"SELECT id.itemname, sum(inv.stock) \
       FROM item id, inventory inv        \
    WHERE id.itemid = inv.itemid      \
       group by id.itemname order by id.itemname";

EXEC SQL PREPARE s2 FROM
        "SELECT id.itemname, sum(inv.stock) \
          FROM item id, inventory inv       \
       WHERE id.itemid = inv.itemid      \
          group by id.itemname order by id.itemname";
```

Example 6-3 shows a JDBC example.

Example 6-3 JDBC

```
PreparedStatement pstmt =
conn.prepareStatement("SELECT id.itemname, sum(inv.stock) \
          FROM item id, inventory inv       \
       WHERE id.itemid = inv.itemid      \
          GROUP BY id.itemname ORDER BY id.itemname";
```

6.2.2 Step 2: Optimizing and matching the query

The Informix server is enhanced to seamlessly use the Informix Warehouse Accelerator in-memory database server to accelerate analytical queries. Applications and tools can connect to and use Informix. After the data mart is set up, analytical queries get the benefit of Informix Warehouse Accelerator performance. The Informix server is aware of the data marts that are deployed on the Informix Warehouse Accelerator and its capabilities. The Informix optimizer accelerates as much of the query as possible. For example, if your query uses OLAP window functions, the Informix server uses the Informix Warehouse Accelerator for select-join-project processing, retrieving the intermediate result set, and running the OLAP window function processing and ORDER BY processing before Informix returns the results to the application. The Informix server does similar post processing when it accelerates UNION, UNION ALL, and derived table and view queries.

Query matching takes effect only if the acceleration is turned on by using the **use_dwa** session environment setting. This can be set automatically by using the **sysdbopen()** procedure to avoid application changes and recompilation. This process is explained later in 6.3, “Enabling query execution” on page 113.

For each query, the Informix optimizer first determines whether the query matches any of the data marts by using the metadata within its database system catalogs. Metadata on the data mart definitions is stored as special views called Accelerated Query Tables (AQTs) within `informix.systables` and `informix.sysviews`. After the qualifying AQT is determined, Informix validates the joins, join types, and join keys, scalar and aggregate expressions.

Example 6-4 shows a snippet of the EXPLAIN file. When the acceleration is on, the EXPLAIN output contains two plans: *DWA executed* and *IDS FYI*. The DWA executed plan contains the information about the accelerator that is used, and the part of the query that is accelerated. The IDS FYI plan is for information purposes only and is used only in case of query fall-back.

Example 6-4 EXPLAIN file snippet

```
QUERY: DWA executed:(OPTIMIZATION TIMESTAMP: 06-09-2013 02:16:49)
------
select count(*) from store_sales

…

QUERY: IDS FYI:(OPTIMIZATION TIMESTAMP: 06-09-2013 02:28:39)
------
select count(*) from store_sales
```

6.2.3 Step 3: Local execution

If the query or query block is unqualified, it runs on Informix. The execution plan can combine local execution with Informix Warehouse Accelerator execution. In a hybrid query plan, the Informix Warehouse Accelerator query acts as a remote iterator that feeds the rows to its parent.

Informix also runs the query if the query acceleration to Informix Warehouse Accelerator fails. For example, query acceleration can fail if there is large number of queries in the Informix Warehouse Accelerator queue.

6.2.4 Step 4: Accelerating SQL

When the query is matched and qualified to run on Informix Warehouse Accelerator, Informix creates an execution plan to run that query block on Informix Warehouse Accelerator. During execution of the query block, it automatically transforms the query block into a distributed query (remote query) and sends it to Informix Warehouse Accelerator.

Example 6-5 shows the snippet of the EXPLAIN output for a query sent to Informix Warehouse Accelerator.

Example 6-5 EXPLAIN snippet

```
1) tpcds@FASTR:informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d: DWA
REMOTE PATH

Remote SQL Request:
{QUERY {FROM  informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d}
{SELECT {COUNT_BIG(*) } } }
```

The query was sent to an accelerator named FASTR from the database. The query matches the AQT "`informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d`" and uses it to generate the column references within the Remote SQL Request. DWA REMOTE PATH indicates the acceleration of the query to Informix Warehouse Accelerator. Remote SQL Request shows the query that was sent to Informix Warehouse Accelerator. The Remote SQL Request text is a SQL dialect specific to Informix Warehouse Accelerator. This SQL dialect is generated by the Informix server and sent to the Informix Warehouse Accelerator for queries that qualify for acceleration.

Each connection from the Informix server to the Informix Warehouse Accelerator supports one SQL statement. After the cursor is closed on a statement, the connection from Informix to Informix Warehouse Accelerator is used by another accelerated statement. However, Informix uses a unique Informix Warehouse Accelerator connection for each active statement that is accelerated to Informix Warehouse Accelerator.

From each session, Informix opens a connection to the Informix Warehouse Accelerator and then generates the SQL equivalent for the query block and sends the query. Informix then opens the cursor on the statement and retrieves the result set. The result set block size is 64 KB.

In this example, the query "`select count(*) from store_sales`" was sent over from the database `tpcds` to the accelerator named FASTR. The actual query sent to Informix Warehouse Accelerator is the following SQL text:

```
{QUERY {FROM  informix.aqtbced10c7-6a05-40a4-812b-3e1427a2f15d} {SELECT
{COUNT_BIG(*) } } }
```

This query uses the Informix distributed query system, and you can see the REMOTE SYSTEM reference to this query and in the EXPLAIN file.

The connection between the Informix session and Informix Warehouse Accelerator persists beyond a single query, and with the same session, Informix reuses the connection when queries are later run on Informix Warehouse Accelerator. However, even within the same session, Informix opens a new connection to Informix Warehouse Accelerator when multiple queries are concurrently active. The statement is active until all rows are retrieved from the result set.

6.2.5 Step 5: Query processing on the coordinator node

The coordinator node can run on the same system as Informix, on a different system, or on a cluster node. The coordinator node maintains connections and sessions with the Informix server. It is also responsible for query processing before and after the worker node query processing.

The coordinator node maintains one or more connections to the Informix server. The coordinator node also maintains and prepares the queue of the queries from the Informix server that are ready for execution. The Informix Warehouse Accelerator runs one query at a time. While one query is running on the worker nodes, other queries are waiting in the queue.

By default, 20 queries can wait in this queue. You can change this value by modifying the **MAXIMUM_NUMBER_QUEUED_QUERIES** parameter in `dwainst.conf` and redo the setup of the accelerator server by using the **ondwa stop**, **ondwa setup**, and **ondwa start** commands. If you submit a query when the queue is full, the coordinator node rejects the query request and the query execution is handled locally by Informix. If the query is rejected, the application does not receive any errors, but the performance is slow. Similarly, if there is an error while preparing the query, the coordinator node returns an error to Informix and the query processing is handled by Informix.

The Informix Warehouse Accelerator processes queries that are based on the massive parallel processing (MPP) architecture. The query execution is managed by the coordinator node. The select-join-predicate processing of the query is delegated to worker nodes.

After the query is processed, the worker nodes return their intermediate result set to the coordinator node. When necessary, the coordinator node merges the result set. Depending on the query, the coordinator node does ORDER BY and FIRST N processing before it returns results to the Informix server. Although the Informix Warehouse Accelerator runs a single query at a time, there is bit of overlap between the query processing at the coordinator and the worker nodes. After the coordinator node receives the complete intermediate results from the worker nodes, it submits the next query on the queue to the worker nodes and concurrently merges the result sets, sorts data, and so on. All of this processing is done in-memory.

6.2.6 Step 6: Query processing on worker nodes

Each worker node maintains a full copy of the dimension tables and a partial copy of the fact table in a compressed form. On each worker node, Informix Warehouse Accelerator runs one query at a time without relinquishing the processor for other queries or yielding for disk I/O. Informix Warehouse Accelerator has all of the data in memory and uses memory for intermediate results. After the result set is ready to be sent to the Informix server, the Informix Warehouse Accelerator picks up the next query in the queue.

For each query, each worker node runs a portion of the query corresponding to its part of the fact table and sends the intermediate results to the coordinator node.

6.2.7 Step 7: Results sent to Informix and DRDA over TCP/IP

For the Informix session, the coordinator node collects all of the results from the other coordinator node tasks and then returns the results. The context and the result set are available until the cursor on the result set is closed. The close cursor statement by the application closes the cursor, ends the statement in the Informix Warehouse Accelerator, and frees all of the memory that is associated with the statement.

6.2.8 Step 8: Post processing on the Informix server

The Informix optimizer maximizes the usage of the Informix Warehouse Accelerator. For each query, the Informix server sends one or more query blocks to the Informix Warehouse Accelerator. If the entire query can be sent to the Informix Warehouse Accelerator, the Informix server does that and the result set from the Informix Warehouse Accelerator is sent to the application without further query processing by the Informix server. If the query is sent partially (for example, OLAP, UNION, and UNION ALL within the query), the Informix server does the post processing of the result set.

When using the UNION and UNION ALL set operators, each query block (arm) can be run on the Informix server or the Informix Warehouse Accelerator, depending on the qualification. The Informix server receives results from the execution of each query block and then combines them according to the query specification.

When using queries with OLAP window functions and aggregates, the Informix server sends a portion of the query with **`SELECT`**, **`JOIN`**, and **`PROJECT`** operations to the Informix Warehouse Accelerator. The non-OLAP aggregates are sent to the Informix Warehouse Accelerator as part of the query.

If the Informix Warehouse Accelerator returns errors during connection or query preparation, the query execution is returned to local Informix server processing and the results are returned to the application. If the Informix Warehouse Accelerator returns errors during query execution, the errors are propagated back to the application.

6.2.9 Step 9: Returning results

The Informix server sends the results of the query to the application by using the configured network and database protocol.

Example 6-6 shows the statistics for the query. In this case, the entire query was sent to the Informix Warehouse Accelerator, so there is only one iterator on the plan. The new type of iterator, `dwa` on Informix, represents the fetching of the result set from the Informix Warehouse Accelerator.

Example 6-6 Query statistics

```
Query statistics:
-----------------
  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------
  type     rows_prod  est_rows  time       est_cost
  ------------------------------------------------
  dwa      1          0         00:00.00   0
```

6.3 Enabling query execution

In a single Informix deployment, there can be transactional, analytical, and hybrid applications that are running at the same time. Analytical and hybrid applications use Informix Warehouse Accelerator more often than OLTP applications. You can transparently enable Informix to accelerate queries from certain applications and not others, without requiring application changes or recompilation.

Here are the prerequisites for query acceleration:

- The data mart is deployed and enabled on Informix Warehouse Accelerator.
- The session environment setting (**`use_dwa`**) is on.
- The query is a dynamic **`SELECT`** statement or **`SELECT`** query block within another statement from an application or stored procedure. A static query within stored procedure does not accelerate.
- The query passes the qualification requirements for acceleration.

By default, all of the queries are run on the Informix server. To accelerate your queries to Informix Warehouse Accelerator, set the following setting for each session with the Informix server:

```
SET ENVIRONMENT use_dwa 'accelerate on';
```

By default, queries that cannot be accelerated by Informix Warehouse Accelerator are run by the Informix server. In some cases, for example, if you are testing or using external tables in data marts, you might not want the query to run on the Informix server. You can turn off the ability for queries to run on the Informix server by running the following command:

```
SET ENVIRONMENT use_dwa 'fallback off';
```

You can issue these commands from your application or by using the **SYSDBOPEN()** procedure, which is implicitly run when you connect to a database. By issuing these commands in the **SYSDBOPEN()** procedure, you avoid changes to the application. You can create a **SYSDBOPEN()** procedure for everyone (**PUBLIC.SYSDBOPEN()**) and for a specific user as **USERNAME.SYSDBOPEN()**. Example 6-7 shows the procedures to set acceleration on or off and fall back on or off for various users in the same database.

Example 6-7 Set acceleration and fallback

```
CREATE PROCEDURE PUBLIC.SYSDBOPEN();
-- Enable acceleration for all users public.
SET ENVIRONMENT use_dwa 'accelerate on';
END PROCEDURE;

CREATE PROCEDURE jsmith.SYSDBOPEN();
-- set acceleration on, fallback off for jsmith.
-- This overrides PUBLIC.SYSDBOPEN().
SET ENVIRONMENT use_dwa 'accelerate on';
SET ENVIRONMENT use_dwa 'fallback off';
END PROCEDURE;

CREATE PROCEDURE jdoe.SYSDBOPEN();
-- Avoid acceleration for jdoe.  This overrides PUBLIC.SYSDBOPEN().
SET ENVIRONMENT use_dwa 'accelerate off';
END PROCEDURE;
```

These settings are effective until the session (connection) is ended or the value is explicitly changed.

Informix also accelerates dynamic statements within your SPL stored procedure, C, and Java user-defined routines. The decision whether to evaluate these dynamic queries for acceleration depends on the **use_dwa** setting at run time, not at compile time. The following two examples each show two sessions: the first session creates a stored procedure and the second session calls this stored procedure.

In Example 6-8, session 2 does not benefit from Informix Warehouse Accelerator because **use_dwa** is not on. You must explicitly or implicitly, by using **sysdbopen()**, set the **use_dwa** session environment setting to "accelerate on" within session 2.

Example 6-8 Query cannot be accelerated - example 1

```
Session 1:
SET ENVIRONMENT use_dwa 'accelerate on';
create procedure state_sales(stid char(2)) returns varchar(24), bigint,
decimal(9,2);
define s lvarchar;
let s = "select state, product, sum(s.netsales) from sales s, product p,
location l";
let s = s || "where s.pid = p.pid and s.lid = l.lid and l.stid = ||  stid;
prepare from stmt s;
end procedure;

Session 2:
execute procedure state_sales("CA");
```

A slightly different implementation is shown in Example 6-9. In this example, the procedure **state_sales2()** uses a static statement. Even though the **use_dwa** session environment setting was set to "accelerate on" during its creation and execution, the query is not accelerated with Informix Warehouse Accelerator. Only statements that are prepared dynamically in the query plan are accelerated.

Example 6-9 Query cannot be accelerated, example 2

```
Session 3:
SET ENVIRONMENT use_dwa 'accelerate on';
create procedure state_sales2(stid char(2)) returns varchar(24), bigint,
decimal(9,2);
define s lvarchar;
define pr_pid bigint;
define pr_sum_sales decimal(9,2);
define pr_state varchar(24)
foreach cursor c1 for select state, product, sum(s.netsales)
into pr_state, pr_pid, pr_sum_sales
from sales s, product p, location l
where s.pid = p.pid and s.lid = l.lid and l.stid = stid
return
end procedure;

Session 4:
SET ENVIRONMENT use_dwa 'accelerate on';
execute procedure state_sales2("NV");
```

C and Java user-defined routines allow only dynamic statements. Therefore, all the statements from C and Java user-defined routines are considered for acceleration.

The acceleration of static statements is being considered as a feature. Your feedback to IBM is helpful and much appreciated. Use the IBM Request For Enhancement (RFE) tool to provide the feedback.

The usage of the session environment setting **use_dwa** and the **sysdbopen()** procedure is documented in Informix documentation. The developerWorks article *SYSDBOPEN: A flexible way to change session behavior in Informix* expands the usage information about **sysdbopen()**. It can be found at the following website:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1109sysdbopen/

6.4 Query matching

The following questions are addressed in this section:

- Can this query run on Informix Warehouse Accelerator?
- How does Informix match the query to the correct data mart?
- Which subset of the queries does it accelerate?
- How are the queries that require both Informix and Informix Warehouse Accelerator processing run?

Figure 6-2 on page 117 depicts a typical flow of SQL query execution within the Informix server. Each query is parsed to ensure that it conforms to the Informix SQL syntax. The semantic analyzer uses the parsed statement together with the table, column, and procedure metadata in the system catalog to ensure that the query is meaningful. The Informix optimizer, whenever it is required and allowed, rewrites the query into a semantically equivalent query. The optimizer then considers the available access paths (table or index), join paths (nested loop or hash join), physical schema of the table and indexes, available system resources, table and index statistics, query directives, and system and session level settings to create an optimal plan to run the query. When EXPLAIN is on, the server writes the EXPLAIN plan, estimate, and runtime statistics in to the EXPLAIN file.

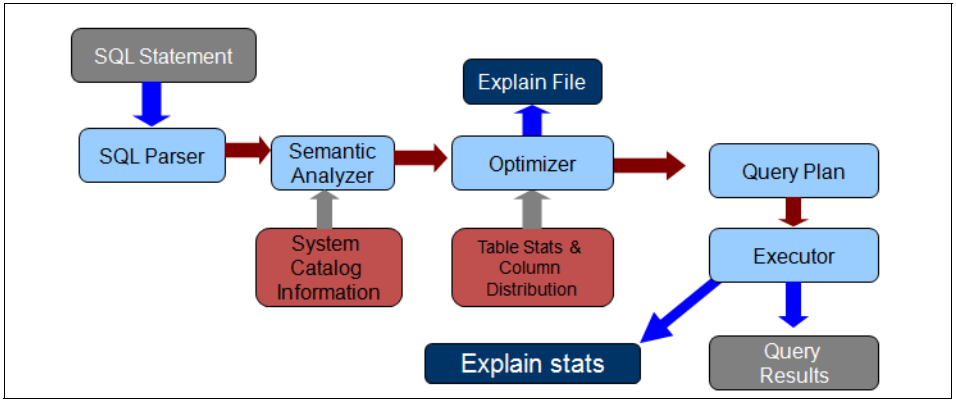


Figure 6-2 Life of a query with Informix

Figure 6-3 shows the query flow when you set **use_dwa** to "accelerate on".

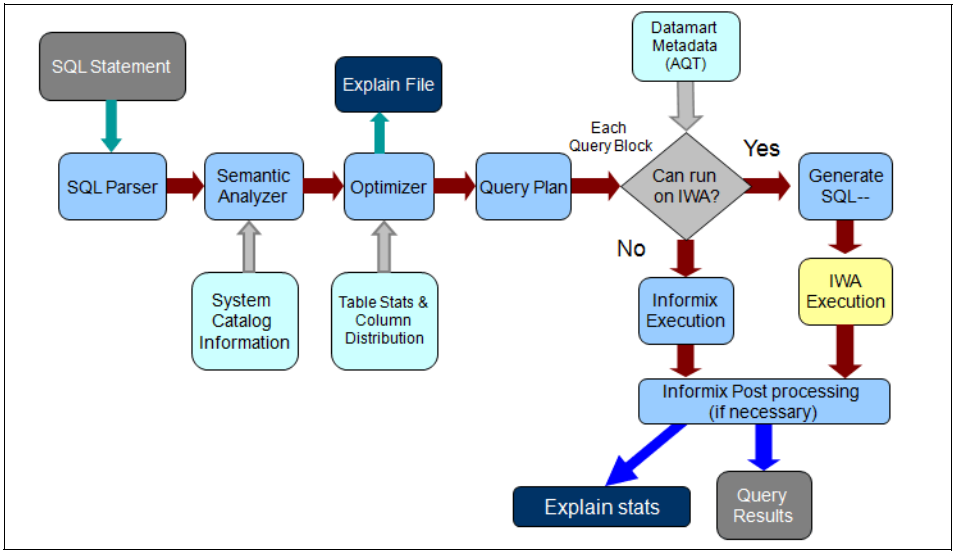


Figure 6-3 Life of a query with Informix and Informix Warehouse Accelerator

After you define the data mart and load the data, the query processing goes along the familiar path. The optimizer creates the query plan for execution on Informix. Then, the optimizer qualifies each query block for acceleration and creates a query plan that maximizes the usage of Informix Warehouse Accelerator. Each qualified query block is run on the Informix Warehouse Accelerator. If needed, the Informix server does post processing for these query blocks. Other query blocks are run on the Informix server. Finally, the Informix server combines the results from all query blocks and sends the result to the application. The planning and execution between the Informix server and the Informix Warehouse Accelerator is transparent and tightly integrated.

The data marts contain one or more snowflake schemas. Each snowflake schema is characterized by a central fact table that is related to one or more dimensions by using one or more keys. The snowflake schema forms a simple star schema when all of the dimensions are directly related to the fact table. In the simplest form, you can have a single fact table without any dimensions.

Informix stores a logical representation of the deployed Informix Warehouse Accelerator data mart in special views called Accelerated Query Tables (AQTs). Each AQT contains the logical definition of a snowflake schema that is part of a data mart in the Informix Warehouse Accelerator. For each query block, the Informix server attempts to create a snowflake schema for the tables by using joins and join keys that are given in the query block. For qualification, the Informix server attempts to verify that the query block snowflake schema is a connected subset of one of the data mart snowflake schemas. The Informix server then verifies that the query uses the Informix Warehouse Accelerator supported expressions. This process is also called *query matching*.

6.4.1 Checklist for query (query block) qualification

The query in Example 6-10 is used for a description of the requirements for query acceleration in the checklist that is shown in Table 6-1 on page 120.

Example 6-10 Example query

```
SELECT Sum(ss_net_profit) / Sum(ss_ext_sales_price) AS gross_margin,
       i_category,
       i_class,
       Rank()
         OVER (
           ORDER BY Sum(ss_net_profit)/Sum(ss_ext_sales_price) ASC)
            AS rank_within_parent
FROM   store_sales,
       date_dim d1,
       item,
```

```
       store
WHERE  d1.d_year = 2001
       AND d1.d_date_sk = ss_sold_date_sk
       AND i_item_sk = ss_item_sk
       AND s_store_sk = ss_store_sk
       AND s_state = 'TN'
GROUP  BY i_category,
          i_class
ORDER  BY rank_within_parent;
```

Table 6-1 show a checklist for qualifying a query or a query block for acceleration.

Table 6-1 Checklist for query (or query block) qualification

Rule	Mnemonic	Description

1	TABLES	The FROM clause of the query block must contain a subset of tables and columns from a single snowflake schema in a data mart. The Informix server applies this filter to select the candidate AQTs to match and analyze further.
		Our example query uses the tables store_sales, date_dim, item, and store. The four tables for this query must be in a single snowflake schema of the data mart.
		Temporary tables cannot be part of a data mart definition.
		In a query block that fills a temporary table, the part of the query block that generates the data by selecting it from regular tables is considered for acceleration.
		Referencing a temporary table in the FROM clause disqualifies the query block.
		More examples:
		► Qualified query
		`INSERT INTO temp_tab (a,b,c)`
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d where f.id = d.id;`
		► Qualified Query
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d`
		`WHERE f.id = d.id INTO temp temp_tab2;`
		► Disqualified query.
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d, temp_tab t`
		`where f.id = d.id and d.c1 = t.c1;`
		All the objects in the data mart definition must be in a single database. Therefore, queries with a direct or synonym reference to a table that is not in the same database as the data mart are disqualified:
		`> database sales;`
		`> SELECT c1, c2, SUM(c3)`
		`FROM fact f, corp:dim d`
		`where f.id = d.id;`

Rule	Mnemonic	Description

2	COLUMNS	The column references within the query must be part of the data mart definition. Informix Warehouse Accelerator supports a subset of the Informix data types for its data marts. Therefore, the queries and expressions must use the supported Informix Warehouse Accelerator data types.
		Here is an example query with column references:
		`store-sales(ss_net_profit, ss_ext_sales, ss_sold_date_s, ss_item_sk, ss_store_sk, ss_ext_sales_price)`
		`date_dim(d_year, d_date_sk)`
		`item(i_category, i_class)`
		`store(s_store_sk, s_state)`

3	FACT	The identified fact table of the query block and the fact table in the snowflake schema of the data mart must match.
		In our example query, store_sales is the fact table in the data mart design. Based on the query structure, the Informix optimizer must identify the store_sales table as the fact table of the query.
		Informix identifies fact tables in each query by using the following rules:
		► For LEFT OUTER JOIN queries, Informix chooses the left most (outer most) table as the fact table.
		► For INNER JOINS, Informix estimates the number of qualified rows after it applies all filters, by using available statistics (collected by using **`UPDATE STATISTICS LOW`**, **`MEDIUM`**, or **`HIGH`**).
		► Then, Informix designates the table with the largest number of estimated rows as a candidate fact table.
		You can use the following statements to force the selection of particular tables as fact tables. Setting PDQPRIORITY is a prerequisite to setting the FACT table.
		`SET PDQPRIORITY 5;`
		`SET OPTIMIZATION ENVIRONMENT FACT "f1";`
		`SET OPTIMIZATION ENVIRONMENT FACT "f1", "f2", "f3";`

Rule	Mnemonic	Description

4	JOIN TYPE	Query joins must include equality predicates that match the table relations in the data mart definition.
		The tables in the query block must be joined (LEFT OUTER JOIN or INNER JOIN) in the same way as in a data mart. For example, the fact table must be the left most dominant side if a LEFT OUTER JOIN is used. The usage of FULL OUTER JOIN disqualifies the query.
		Our example query uses inner joins, but the joins are implied through join keys in the WHERE clause:
		`d1.d_date_sk = ss_sold_date_sk`
		`AND i_item_sk = ss_item_sk`
		`AND s_store_sk = ss_store_sk`
		The tables (date_dim with store_sales, item with store_sales, and store with store_sales) are joined with equality predicates.
		When you use ANSI joins, the join keys are explicitly specified in the ON-clause.
		Here are some more examples:
		► Qualified:
		`SELECT count(*) FROM fact f, dim d where f.id = d.id;`
		`SELECT count(*) FROM fact f INNER JOIN dim d on (f.id = d.id);`
		`SELECT count(*) FROM fact f LEFT OUTER JOIN dim d on (f.id = d.id);`
		► Unqualified:
		`SELECT count(*) FROM fact f, dim d where f.id >= d.id;`
		`SELECT count(*) FROM dim d FULL OUTER JOIN fact f on (f.id = d.id);`
		`SELECT count(*) FROM fact f RIGHT OUTER JOIN dim d on (f.id = d.id);`
		`SELECT count(*) FROM fact f FULL OUTER JOIN dim d on (f.id = d.id);`
		Joining the table to itself with multiple references, either dimension or fact, disqualifies the query because it does not match the snowflake schema of the data mart that you created. To fix this issue, create and copy your table into a separate table and use it in the data mart definition and query.
		The following query joins the fact table `store_sales` to itself:
		`SELECT count(*)`
		`FROM store_sales s1, store_sales s2`
		`WHERE s1.ss_store_sk = s2.ss_store_sk`
		`AND s1.ss_net_qty > s2.ss_net_qty`
		`AND s2.ss_store_sk = 1234;`

Rule	Mnemonic	Description

5	JOIN KEYS	The join keys in the query must contain the join keys and relationships in the data mart definition.
		After the fact table is identified, the Informix server searches for the dimension tables that the fact table is directly joined to and the equality join keys. Then, for each dimension table, Informix uses the equality join keys to recursively search for tables to which the dimension tables are joined.
		Here are the table join keys in our example query:
		`d1.d_date_sk = ss_sold_date_sk`
		`AND i_item_sk = ss_item_sk`
		`AND s_store_sk = ss_store_sk`
		Informix uses the information in AQT views to match the join keys in the query to the join keys in the data mart definition.
		Here are some more examples:
		► Qualified:
		`SELECT count(*) FROM fact f LEFT OUTER JOIN dim d on (f.id = d.id);`
		► Unqualified:
		`SELECT count(*) FROM fact f INNER JOIN dim d on (f.id < d.id);`
		`SELECT count(*) FROM fact f LEFT OUTER JOIN dim d on (f.id > d.id);`

6	EXPRESSIONS	The query block must use only the supported operators, expressions, scalar, aggregate, and OLAP functions.
		For a detailed list of the supported expressions, see the Informix documentation.
		Our example query uses the following expressions:
		Projection list: SUM, RANK() OVER(), equality predicates. Informix supports the acceleration of queries with these expressions.
		Additional examples:
		`Sum(ss_net_profit) / Sum(ss_ext_sales_price) AS gross_margin,`
		`  i_category,`
		`  i_class,`
		`  Rank()`
		`  OVER (`
		`  ORDER BY Sum(ss_net_profit)/ Sum(ss_ext_sales_price) ASC) AS`
		`  rank_within_parent`
		
		`WHERE clause:`
		
		`d1.d_year = 2001`
		`AND d1.d_date_sk = ss_sold_date_sk`
		`AND i_item_sk = ss_item_sk`
		`AND s_store_sk = ss_store_sk`
		`AND s_state = 'TN'`

Rule	Mnemonic	Description

7	SUBQUERIES	The query block must not contain any subqueries.
		Queries that use the **LATERAL** keyword contain subqueries with correlated reference, and therefore are not accelerated.
		The query in our example does not use subqueries.
		Here is an example of unsupported queries:
		`SELECT count(*) FROM fact f, dim d`
		`where f.id = d.id where f.skid in`
		`(select d2.skid from d2 where d.skid = d2.sid);`

8	CONSTRAINTS	Dimension tables can be related to fact tables by either 1:n or n:m.
		In a 1:n relationship, for every row in the dimension table, there are zero or more rows in the fact table. This requires a unique constraint on the join keys in the dimension table.
		In a n:m relationship, there can be duplicate rows in the dimension table with matching join keys in the fact table.
		When you have n:m relationships, there are multiple AQTs created in the system catalog that represent the same star schema. If you have debug mode on (**use_dwa** set to '`debug on`'), you are shown these types of messages:
		`03:58:31  SQDWA: Identified 4 candidate AQTs for matching`
		`03:58:31  SQDWA: matched:`
		`aqta5337ff9-7208-4b5d-986b-e95d0a124506`
		`03:58:31  SQDWA: matching successful (8 msec)`
		`aqta5337ff9-7208-4b5d-986b-e95d0a124506`
		`03:58:31  SQDWA: offloading successful (152 msec)`
		To avoid returning the wrong results, the Informix optimizer must match the query to the correct AQT. For example, for AQTs that are created for a 1:n relationship, the tables in the queries can be a subset of the tables in the AQT. For AQTs that are created for a n:m relationship, the tables in the query must match the tables in the AQT.

Rule	Mnemonic	Description

9	STATEMENTS	The query block is a **SELECT** statement. This SELECT query block must be part of a **SELECT**, **SELECT INTO**, **VIEW** reference, or **INSERT** statement.
		Qualified SELECT query blocks:
		Our example query contains a **SELECT** statement and benefits from query acceleration. The **SELECT** query blocks in the **INSERT** statements are evaluated for acceleration. Here are additional examples:
		`INSERT INTO temp_tab (a,b,c)`
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d where f.id = d.id;`
		
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d`
		`WHERE f.id = d.id INTO temp temp_tab2;`
		The queries are matched by using the internal query plan after the views are translated into actual table references. If they meet the remaining requirements, the queries that reference fact and dimensions tables of an AQT are matched and accelerated:
		`CREATE VIEW sales_view(c1, c2, c3) AS`
		`SELECT c1, c2, SUM(c3)`
		`FROM fact f, dim d where f.id = d.id;`
		
		`SELECT * from sales_view WHERE c2 = 'CA';`
		UPDATE and **DELETE** statements do not benefit from query acceleration.
		For a list of supported statements and types that can be accelerated, see the Informix documentation.

10	DYNAMIC	The **SELECT** must be a dynamic statement that is submitted directly by the application, an SPL stored procedure, C, or Java function. For examples, see 6.3, “Enabling query execution” on page 113. Our example query is accelerated if it is submitted dynamically, but is not accelerated as a static statement inside an SPL stored procedure.

Rule	Mnemonic	Description

11	OPERATORS	Only the UNION or UNION ALL set operators are supported.
		Each query block of UNION and UNION ALL operators is evaluated for acceleration.
		Queries with MINUS, EXCEPT, and INTERSECT set operations are run in the Informix server. Queries with MINUS, EXCEPT, and INTERSECT operators are internally rewritten as subqueries. You cannot accelerate query blocks with subqueries.
		Our example query does not use MINUS, EXCEPT, or INTERSECT operators.
		Here are different examples:
		► Qualified:
		`SELECT d.a, f.x FROM fact f, dim d where f.id = d.id`
		`UNION`
		`SELECT d2.p, f.x FROM fact f, dim d2 where f.id = d2.id`
		
		`SELECT d.a, f.x FROM fact f, dim d where f.id = d.id`
		`UNION ALL`
		`SELECT d2.p, f.x FROM fact f2 INNER JOIN dim d2 on (f.id = d2.id);`
		► Unqualified:
		`SELECT d.a, f.x FROM fact f, dim d where f.id = d.id`
		`INTERSECT`
		`SELECT count(*) FROM fact f, dim d2 where f.id = d2.id`
		
		`SELECT d.a, f.x FROM fact f, dim d where f.id = d.id`
		`MINUS`
		`SELECT d2.p, f.x FROM fact f2 INNER JOIN dim d2 on (f.id = d2.id);`

Example 6-11 shows the query plan for our example query.

Example 6-11 Query plan

```
QUERY: DWA executed: (OPTIMIZATION TIMESTAMP: 6-05-2013 13:12:33)
------
SELECT Sum(ss_net_profit) / Sum(ss_ext_sales_price) AS gross_margin,
       i_category,
       i_class,
       Rank()
         OVER (
           ORDER BY Sum(ss_net_profit)/ Sum(ss_ext_sales_price) ASC) AS
       rank_within_parent
FROM   store_sales,
       date_dim d1,
       item,
       store
WHERE  d1.d_year = 2001
       AND d1.d_date_sk = ss_sold_date_sk
       AND i_item_sk = ss_item_sk
       AND s_store_sk = ss_store_sk
```

```
       AND s_state = 'TN'
GROUP  BY i_category,
          i_class
ORDER  BY rank_within_parent;


Estimated Cost: 4030817
Estimated # of Rows Returned: 19380
Temporary Files Required For: Order By  Group By

  1) ds2@SALESACC:dwa.aqtcdcddf39-4cb1-44db-a797-f647e41ffb3b: REMOTE PATH

    Remote SQL Request:
    {QUERY {FROM  dwa.aqtcdcddf39-4cb1-44db-a797-f647e41ffb3b} {WHERE (((({=
COL334 2001 } {ISNOTNULL COL328 } ){ISNOTNULL COL056 } ){ISNOTNULL COL158 } ){=
COL182 "819 TN" } )} {SELECT {SYSCAST {/ {SYSCAST {SUM COL044 }  AS DOUBLE} {
CASE  OF {WHEN {= {SUM COL037 }  0 } NULL } ELSE {SUM COL037 } }} AS DOUBLE}
{SYSCAST COL068 AS CHAR 50 819} {SYSCAST COL066 AS CHAR 50 819} {SYSCAST AS
BIGINT} } {GROUP COL068 COL066 } }


QUERY: IDS FYI:(OPTIMIZATION TIMESTAMP: 1-05-2013 17:02:29)
------
SELECT Sum(ss_net_profit) / Sum(ss_ext_sales_price) AS gross_margin,
       i_category,
       i_class,
       Rank()
         OVER (
           ORDER BY Sum(ss_net_profit)/ Sum(ss_ext_sales_price) ASC) AS
       rank_within_parent
FROM   store_sales,
       date_dim d1,
       item,
       store
WHERE  d1.d_year = 2001
       AND d1.d_date_sk = ss_sold_date_sk
       AND i_item_sk = ss_item_sk
       AND s_store_sk = ss_store_sk
       AND s_state = 'TN'
GROUP  BY i_category,
          i_class
ORDER  BY rank_within_parent

Estimated Cost: 4030817
Estimated # of Rows Returned: 19380
Temporary Files Required For: Order By  Group By

  1) dwa_ds.d1: SEQUENTIAL SCAN
```

```
        Filters: dwa_ds.d1.d_year = 2001

  2) dwa_ds.store_sales: INDEX PATH

    (1) Index Name: dwa_ds. 125_170
        Index Keys: ss_sold_date_sk   (Serial, fragments: ALL)
        Lower Index Filter: dwa_ds.d1.d_date_sk =
dwa_ds.store_sales.ss_sold_date_sk
NESTED LOOP JOIN

  3) dwa_ds.store: SEQUENTIAL SCAN

        Filters:
        Table Scan Filters: dwa_ds.store.s_state = 'TN'


DYNAMIC HASH JOIN
    Dynamic Hash Filters: dwa_ds.store.s_store_sk =
dwa_ds.store_sales.ss_store_sk

  4) dwa_ds.item: INDEX PATH

    (1) Index Name: dwa_ds. 110_55
        Index Keys: i_item_sk   (Serial, fragments: ALL)
        Lower Index Filter: dwa_ds.item.i_item_sk =
dwa_ds.store_sales.ss_item_sk
NESTED LOOP JOIN


Query statistics:
-----------------

  Table map :
  ----------------------------
  Internal name     Table name
  ----------------------------

  type     rows_prod  est_rows  time       est_cost
  ------------------------------------------------
  dwa      154        0         00:01.25   0

  type     rows_sort  est_rows  rows_cons  time
  ------------------------------------------------
  sort     154        0         154        00:01.25

  type     it_count   time
  ---------------------------
  olap     154        00:01.25
```

```
  type     rows_sort  est_rows  rows_cons  time       est_cost
  ------------------------------------------------------------
sort     154         19380     154         00:01.25   10690
```

First, `DWA executed` shows the plan that is run with Informix Warehouse Accelerator acceleration. The Informix server also prints the local plan (`IDS FYI`) for the query, for informational purposes only.

The query contains the OLAP window function:

```
Rank() OVER (ORDER BY Sum(ss_net_profit)/Sum(ss_ext_sales_price) ASC) AS
       rank_within_parent
```

The rest of the query is matched with the data mart definitions and the following query is sent to Informix Warehouse Accelerator:

```
1) ds2@SALESACC:dwa.aqtcdcddf39-4cb1-44db-a797-f647e41ffb3b: REMOTE PATH
Remote SQL Request:
    {QUERY {FROM  dwa.aqtcdcddf39-4cb1-44db-a797-f647e41ffb3b} {WHERE (((({=
COL334 2001 } {ISNOTNULL COL328 } ){ISNOTNULL COL056 } ){ISNOTNULL COL158 } ){=
COL182 "819 TN" } )} {SELECT {SYSCAST {/ {SYSCAST {SUM COL044 }  AS DOUBLE} {
CASE  OF {WHEN {= {SUM COL037 }  0 } NULL } ELSE {SUM COL037 } }} AS DOUBLE}
{SYSCAST COL068 AS CHAR 50 819} {SYSCAST COL066 AS CHAR 50 819} {SYSCAST AS
BIGINT} } {GROUP COL068 COL066 } }
```

The final section shows statistics for each iterator in the query plan. The DWA iterator represents execution on Informix Warehouse Accelerator. Informix Warehouse Accelerator does the select-join-project processing, and returns 154 rows into Informix. Informix then sorts the result set, evaluates the RANK() window functions and then runs the final ORDER BY clause before it returns the final result set to the application. The statistics also show the amount of time that is taken by each iterator, including DWA. In this case, the DWA iterator took 1.25 seconds and an insignificant amount of time was used to process the rest of the query, including OLAP window processing and final ORDER BY. The most expensive portion of the query processing was sent to the Informix Warehouse Accelerator, which helps run a four-table star join query with an estimated Informix cost of 4030817 in 1.25 seconds.

6.4.2 Examples for query qualification

To illustrate how queries are matched, consider a snow flake schema with one fact table, ten dimensions, and three levels of relationships. Assume that there are 1:n relationships between dimensions and dimension to fact (Figure 6-4).

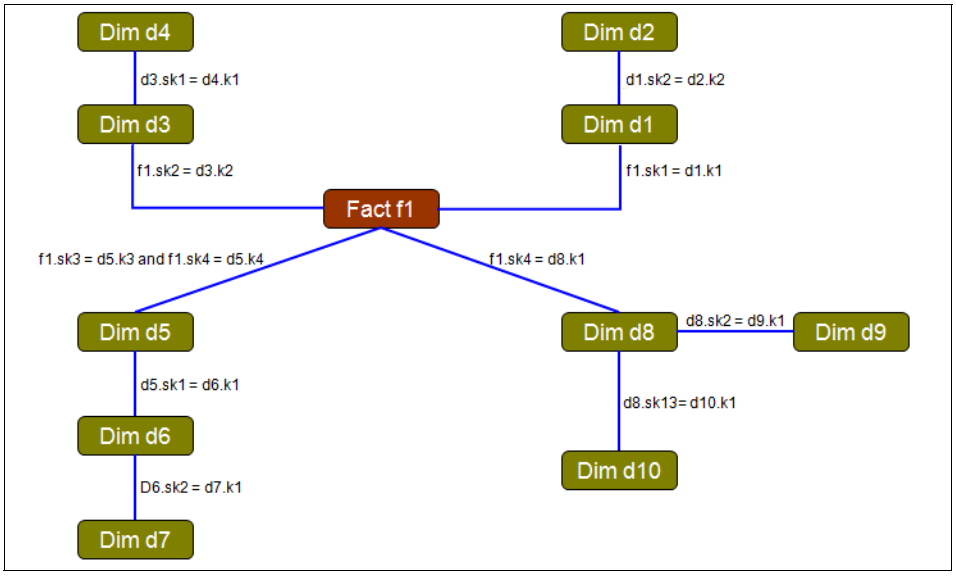


Figure 6-4 Snowflake schema of the data mart

The examples that follow provide seven different queries that all use the underlying database schema of the data mart that is shown in Figure 6-4. Each example shows the SQL statement and how the Informix server matches the query against the data mart definition.

Query 1

The SQL statement in Example 6-12 queries fact table f1 and two dimension tables, d1 and d8:

Example 6-12 Query 1

```
SELECT d2.v3,
       SUM(f1.v2)
FROM   f1,
       d1,
       d2,
       d8
WHERE  f1.sk1 = d1.k1
       AND f1.sk4 = d8.k1
```

```
        AND d1.sk2 = d2.k2

        AND d1.v1 = "tires"
        AND d2.v2 = "july"
GROUP   BY d2.v3
ORDER   BY SUM(f1.v2);
```

Informix creates an internal snowflake representation of the query (Figure 6-5) after it identifies the fact table and analyzes the join keys.

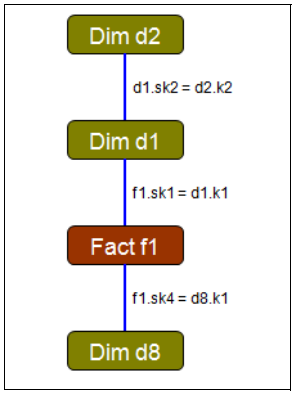


Figure 6-5 Snowflake representation of the query

Informix then maps this snowflake query representation on to the snowflake of the candidate data mart (AQT), as shown in Figure 6-6. The yellow box indicates the part of the data mart schema that is matched by the schema of the query. The query matches the AQT and the rest of the rules and therefore can be accelerated.

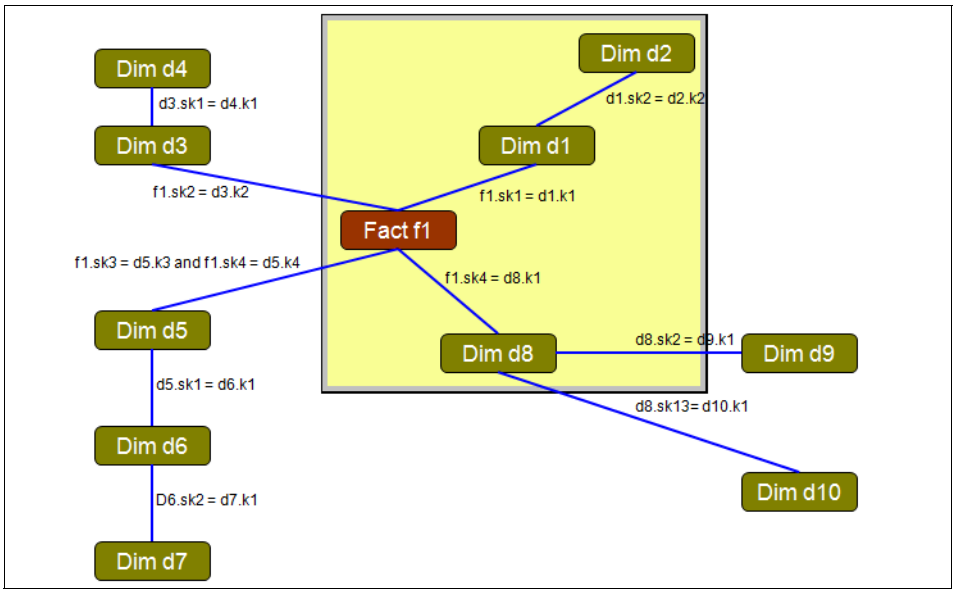


Figure 6-6 Projection of snowflake in Figure 6-5 on to the data mart definition

Query 2

The SQL statement in Example 6-13 joins fact table t1 with dimension table d1, and dimension table d1 with dimension table d2.

Example 6-13 Query 2

```
SELECT COUNT(*)
FROM   f1,
       d1,
       d2
WHERE  f1.sk1 = d1.k1
       AND d1.sk2 = d2.k2
```

Figure 6-7 shows the projection of query 2 on to the data mart definition.

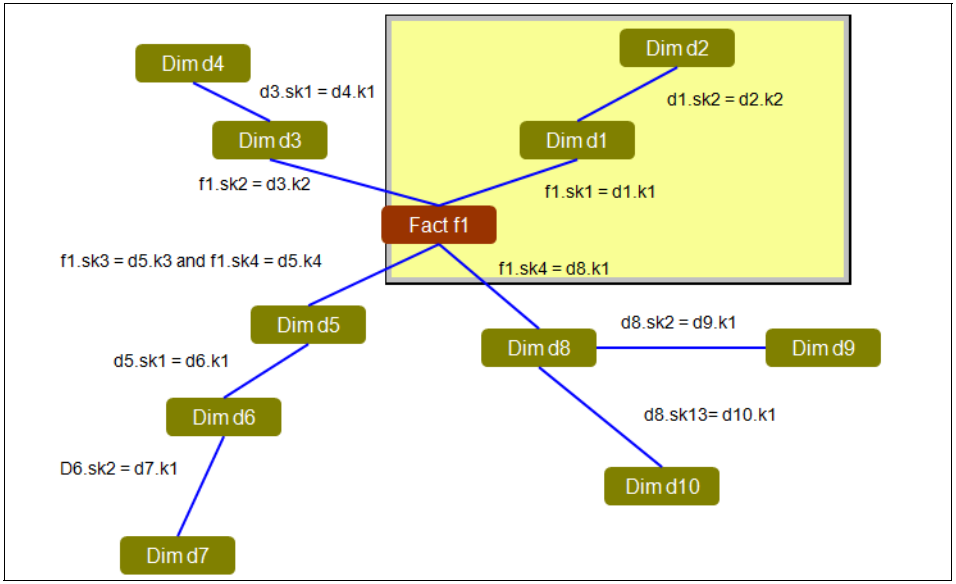


Figure 6-7 Projection of query 2 on to the data mart's snowflake

Query 2 matches the data mart definition, and the query can be accelerated.

Query 3

Example 6-14 shows the SQL statement of query 3. This query joins fact table f1 with dimension tables d3 and d5. The join between fact table f1 and dimension table d5 is on two table columns.

Example 6-14 Query 3

```
SELECT COUNT(*)
FROM   f1
       LEFT OUTER JOIN d5 ON
      ( f1.sk3 = d5.k3 AND f1.sk4 = d5.k4 )
       INNER JOIN d3 ON ( f1.sk2 = d3.k2 )
WHERE  d5.v1 BETWEEN 10 AND 19
       AND d3.v1 = "10-10-2012";
```

Figure 6-8 shows how the query schema is matched on to the data mart definition.

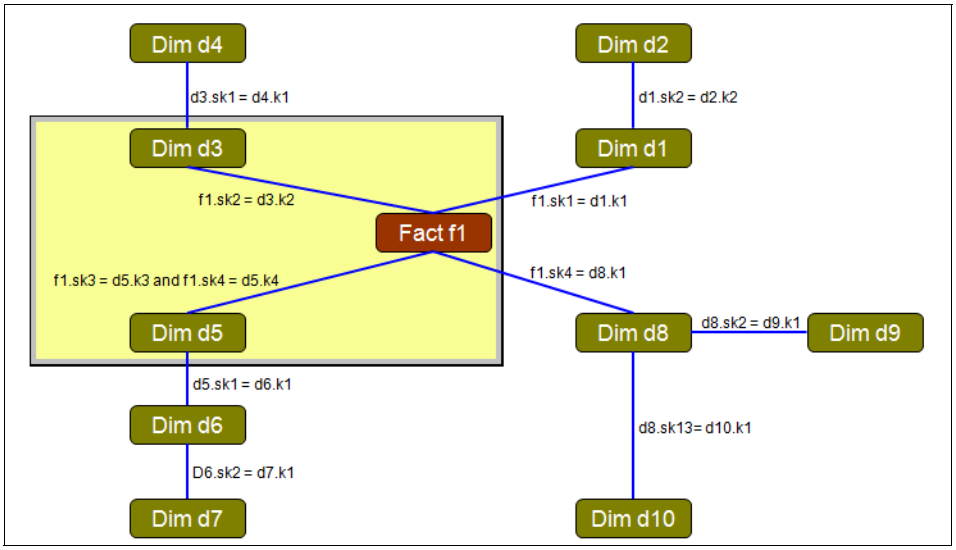


Figure 6-8 Projection of query 3 onto the data mart's snowflake

Query 3 can be accelerated. The two table column join of fact table f1 to dimension table d5 is done according to the data mart definition.

Query 4

Example 6-15 shows the SQL statement of query 4, which joins fact table f1 with dimension tables d3 and d10.

Example 6-15 Query 4

```
SELECT COUNT(*)
FROM   f1
       LEFT OUTER JOIN d10 ON ( f1.sk4 = d10.sk2 )
       INNER JOIN d3
               ON ( f1.sk2 = d3.k2 )
WHERE  d5.v1 BETWEEN 10 AND 19
       AND d3.v1 = 5;
```

Figure 6-9 shows the projection of the query schema on to the data mart definition.

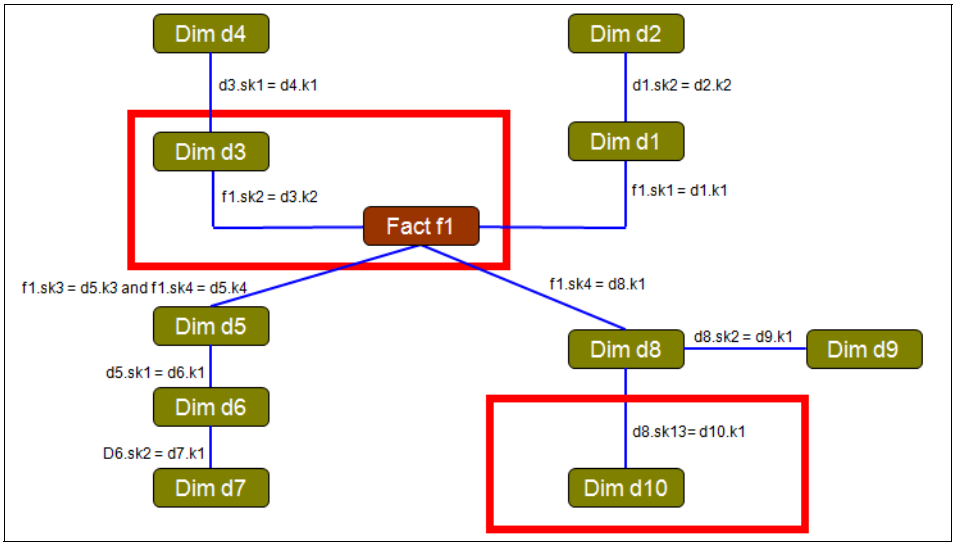


Figure 6-9 Projection of query 4 on to the data mart's snowflake

All of the tables in the query are present in the data mart. However, tables f1 and d10 are joined directly in the query, but not in the data mart definition. Therefore, the query cannot be accelerated.

Query 5

Example 6-16 shows the SQL statement that joins fact table f1 with dimension table d5 on the two columns. Then, it joins dimension table d5 with d6 and d6 with d7.

Example 6-16 Query 5

```
SELECT COUNT(*)
FROM   f1
       LEFT OUTER JOIN d5
                    ON ( f1.sk3 = d5.k3
                         AND f1.sk4 = d5.k4 )
       INNER JOIN d6
               ON ( d5.sk1 = d6.k1 )
       INNER JOIN d7
               ON ( d6.sk2 = d7.k1 )
WHERE  d5.v1 BETWEEN 10 AND 19
       AND d3.v1 = 5;
```

Figure 6-10 shows how the query schema is projected on to the data mart definition.

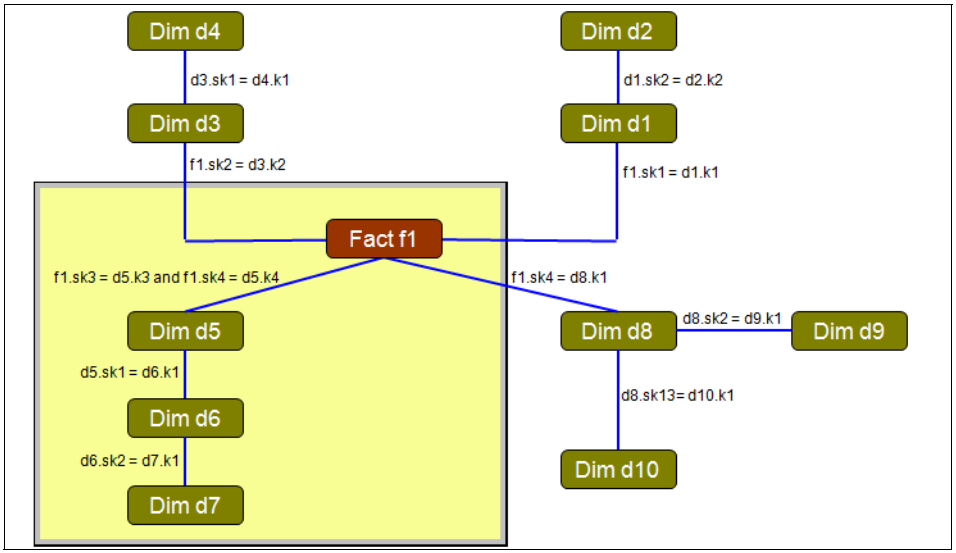


Figure 6-10 Projection of query 5 on to the data mart's snowflake

This query matches the data mart definition for the fact table, join, and other requirements, and can be accelerated.

Query 6

Example 6-17 shows the SQL statement that combines two query blocks by using UNION ALL. Both query blocks select from the fact table f1 and dimension tables d5, d6, and d7.

Example 6-17 Query 6

```
--Query block 1
SELECT COUNT(*)
FROM   f1
       LEFT OUTER JOIN d5
                    ON ( f1.sk3 = d5.k3
                         AND f1.sk4 = d5.k4 )
       INNER JOIN d6
               ON ( d5.sk1 = d6.k1 )
       INNER JOIN d7
               ON ( d6.sk2 = d7.k1 )
WHERE  d5.v1 BETWEEN 10 AND 19
       AND d3.v1 = "10-10-2012";
```

```
UNION ALL

--Query block 2
SELECT COUNT(*)
FROM   f1
       LEFT OUTER JOIN d5
                    ON ( f1.sk3 = d5.k3
                         AND f1.sk4 = d5.k4 )
       INNER JOIN d6
               ON ( d5.sk1 = d6.k1 )
       INNER JOIN d7
               ON ( d6.sk2 = d7.k1 )
WHERE  d5.v1 BETWEEN 10 AND 19
       AND d7.v1 = 5
       AND d5.v2 IN (SELECT max(d6.v2)
                     FROM   d6
                     WHERE  d6.v3 = d5.v3);
```

Figure 6-11 shows the projection of both query blocks on to the data mart schema. The projections of both query blocks are congruent because they both select from the same tables; only the filtering from the tables is different.

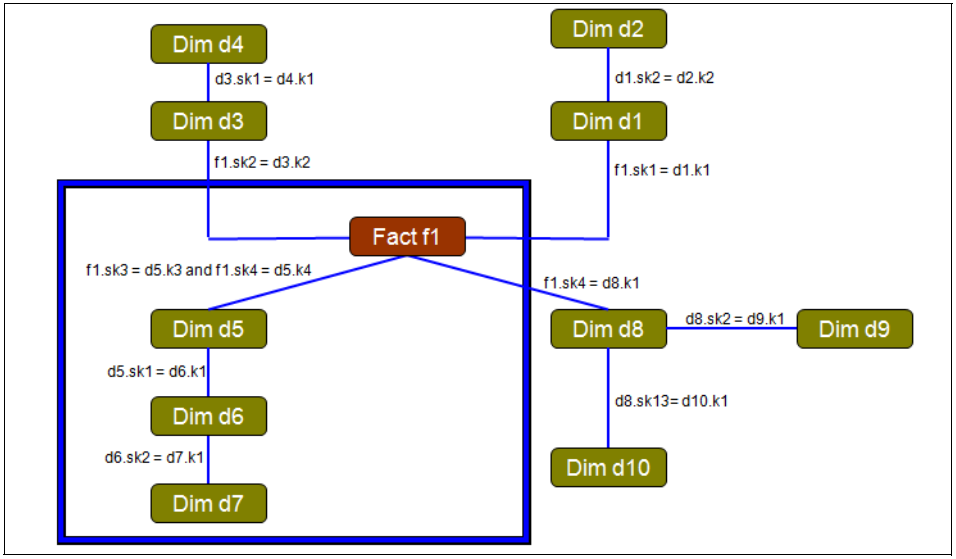


Figure 6-11 Projection of query 6 on to the data mart's snowflake

Query block 1 and query block 2 both match the table and join key requirements, but query block 2 has a subquery and therefore is not accelerated, but is run locally by Informix. Query block 1is accelerated.

Query 7

Example 6-18 shows the SQL statement that has three query blocks that are combined by UNION ALL. Each query block selects from the fact table, but joins it with different dimension tables.

Example 6-18 Query 7

```
SELECT COUNT(*)
FROM f1, d5
WHERE f1.sk3 = d5.k3
    AND f1.sk4 = d5.k4
    AND d5.v1 BETWEEN 10 AND 19

UNION ALL

SELECT COUNT(*)
FROM f1, d1
WHERE f1.sk1 = d1.k1
    AND d1.v1 = "tires";

UNION ALL

SELECT COUNT(*)
FROM f1, d8, d9
WHERE f1.sk1 = d8.k1
   AND d8.sk2 = d9.k1
   AND d8.v1 = "Mazda"
   AND d9.v2 = "100k";
```

Figure 6-12 shows the overlapping projections of the three query blocks on to the data mart definition.

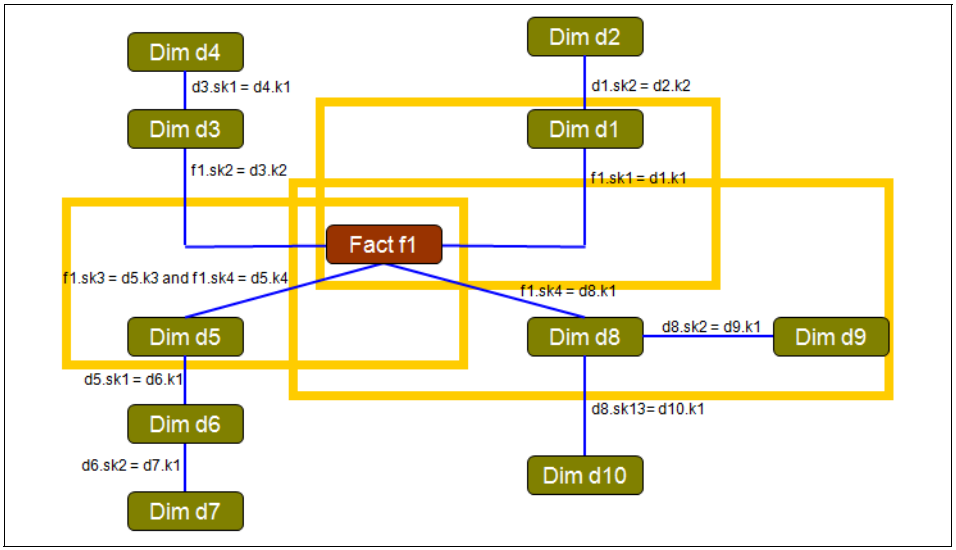


Figure 6-12 Projection of the query blocks of query 7 on to the data mart's snowflake

All three branches of the UNION query are matched to different parts of the snowflake schema and are qualified for acceleration. The Informix server sends each query block and fetches the results from Informix Warehouse Accelerator serially.

6.5 Monitoring query execution

After you design the data mart, enable the query execution, and test queries for acceleration, you deploy Informix server and Informix Warehouse Accelerator in to production. If the application or the reports generate only a predefined or predictable set of queries, it is easy to test them. However, if the application generates queries dynamically, it is difficult to test acceleration for all possible combinations.

The session environment setting **use_dwa** can get debugging information about a query:

```
SET ENVIRONMENT use_dwa 'debug on';
```

Example 6-19 is an example of a debug message of a successful acceleration in `online.log` when this option is on.

Example 6-19 Debug messages - successful

```
03:15:03  SQDWA: select sum(ss_net_profit) from store_sales
03:15:03  SQDWA: Identified 1 candidate AQTs for matching
03:15:03  SQDWA: matched: aqtbced10c7-6a05-40a4-812b-3e1427a2f15d
03:15:03  SQDWA: matching successful (61 msec)
aqtbced10c7-6a05-40a4-812b-3e1427a2f15d
03:15:03  SQDWA: offloading successful (1 msec)
```

Example 6-20 shows an example of a debug message of an unsuccessful acceleration in `online.log` when this option is on. This error message indicates the reason that the query was disqualified from acceleration. In this case, column ss_sold_date_sk, which was used in the query, was not part of the data mart definition.

Example 6-20 Debug messages - unsuccessful

```
03:27:51  SQDWA: select max(ss_sold_date_sk) from store_sales
03:27:51  SQDWA: Identified 1 candidate AQTs for matching
03:27:51  SQDWA: not matched: aqtbced10c7-6a05-40a4-812b-3e1427a2f15d
-26552: column is not contained in mart definition (table
'dwa_ds'.store_sales, column ss_sold_date_sk, projection list)
03:27:51  SQDWA: matching failed (-26552): column is not contained in
mart definition (table 'dwa_ds'.store_sales, column ss_sold_date_sk,
projection list)
03:27:51  SQDWA: reverting to original plan
```

In a production environment, it is convenient to use the **`AVOID_EXECUTE`** option in EXPLAIN to avoid sending the queries to Informix Warehouse Accelerator. If the queries are disqualified from acceleration, you see only the query plan for Informix because a plan for Informix Warehouse Accelerator could not be created. To avoid execution of the query by Informix server, you set **`use_dwa`** `'fallback off'`. Finally, to enable the debug output, you also set **`use_dwa`** `'debug on'`. With this combination, you can check, where a query can accelerate or not, without waiting for the statement to return results and potentially putting workload on the Informix server. If the query is not accelerated to Informix Warehouse Accelerator, Informix aborts query execution after printing the debug messages in `online.log`. See Example 6-21.

Example 6-21 Abort message

```
SET ENVIRONMENT use_dwa 'accelerate on';
SET ENVIRONMENT use_dwa 'debug on';
```

```
SET ENVIRONMENT use_dwa 'fallback off';
SET EXPLAIN ON AVOID_EXECUTE;
select max(ss_sold_date_sk) from store_sales;

03:30:07  SQDWA: select max(ss_sold_date_sk) from store_sales
03:30:07  SQDWA: Identified 1 candidate AQTs for matching
03:30:07  SQDWA: not matched: aqtbced10c7-6a05-40a4-812b-3e1427a2f15d
-26552: column is not contained in mart definition (table
'dwa_ds'.store_sales, column ss_sold_date_sk, projection list)
03:30:07  SQDWA: matching failed (-26552): column is not contained in
mart definition (table 'dwa_ds'.store_sales, column ss_sold_date_sk,
projection list)
03:30:07  SQDWA: Aborting query execution (-26404)
```

6.6 Summary

Understanding how queries are run helps you design data marts more efficiently and manage your existing data marts to meet the changing requirements of your application. By using the **use_dwa** session environment setting and the **SYSDBOPEN()** procedure, you can accelerate the queries to Informix Warehouse Accelerator without changing your application or business intelligence tool. For queries with OLAP window functions and aggregates, UNION or UNION ALL operators, and **INSERT** or **SELECT INTO** statements, Informix accelerates query blocks or partial queries to Informix Warehouse Accelerator. The Informix EXPLAIN file and **use_dwa** `'debug on'` setting can help you understand and debug issues.

7

Managing and refreshing an IBM Informix Warehouse Accelerator data mart

Informix Warehouse Accelerator provides incredible query performance by using a snapshot of the data from the Informix database. Business processes can require an analysis of operational data from a few minutes ago to a few days ago. Other applications, and extract, transform, and load (ETL) jobs, can modify the data in the Informix database. You must determine how to keep the Informix Warehouse Accelerator data synchronized with the Informix data and schema. You can determine the timing, frequency, and mechanism to reflect these changes in the Informix Warehouse Accelerator data mart. The data mart is the unit of acceleration from Informix to Informix Warehouse Accelerator.

You can refresh the Informix Warehouse Accelerator data mart or refresh only a portion, for example, the modified tables and partitions. You can also use trickle feed to get the updates. These refresh processes can be automated. The tools and methods for managing and automating the data mart refresh tasks and data mart lifecycle are described here, including how to keep the data synchronized with the Informix data and schema.

7.1 Overview

Here is an overview of how the data mart was created and deployed for query acceleration for our example. Figure 7-1 shows the steps to create an Informix Warehouse Accelerator data mart.

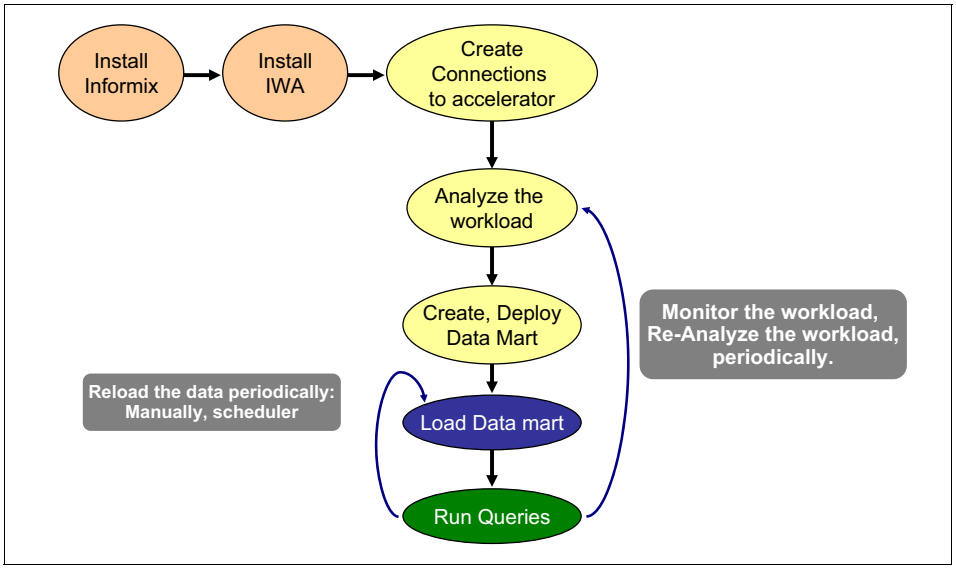


Figure 7-1 Steps for Informix Warehouse Accelerator data mart creation

Five steps are required to deploy the database server and the accelerator. These steps are explained in greater detail in the earlier chapters and the *Informix Warehouse Accelerator Administration Guide*:

1. Install, configure, and start the Informix database server.
2. Install, configure, and start the accelerator. The key configuration settings are the location of the file system for retaining a copy of the memory image, the amount of memory that is allocated for Informix Warehouse Accelerator, and the number of nodes.
3. Using IBM Informix Open Admin Tool (OAT) or IBM Smart Analytics Optimizer Studio, connect to the Informix database and add the accelerator connection information.

4. Design and create the data marts on Informix Warehouse Accelerator. You can create an optimal data mart by analyzing an existing workload by using OAT or stored procedures, or design the mart manually by using Smart Analytics Optimizer Studio.
5. Load data to the Informix Warehouse Accelerator data mart. The Informix and Informix Warehouse Accelerator are now ready for query acceleration.

After the data mart is deployed, many things can change in the system. For example, data on the Informix tables can change or the logical or physical schema of a table can change.

Here are the tools that are available for creating and managing a data mart:

- IBM Informix Open Admin Tool (OAT)
- Informix and Informix Warehouse Accelerator built-in Stored Procedure API
- IBM Smart Analytics Optimizer Studio

7.1.1 IBM Informix Open Admin Tool

IBM Informix Open Admin Tool (OAT) is a web application that is an open source graphical tool for administering and analyzing the performance of IBM Informix database servers (Figure 7-2).

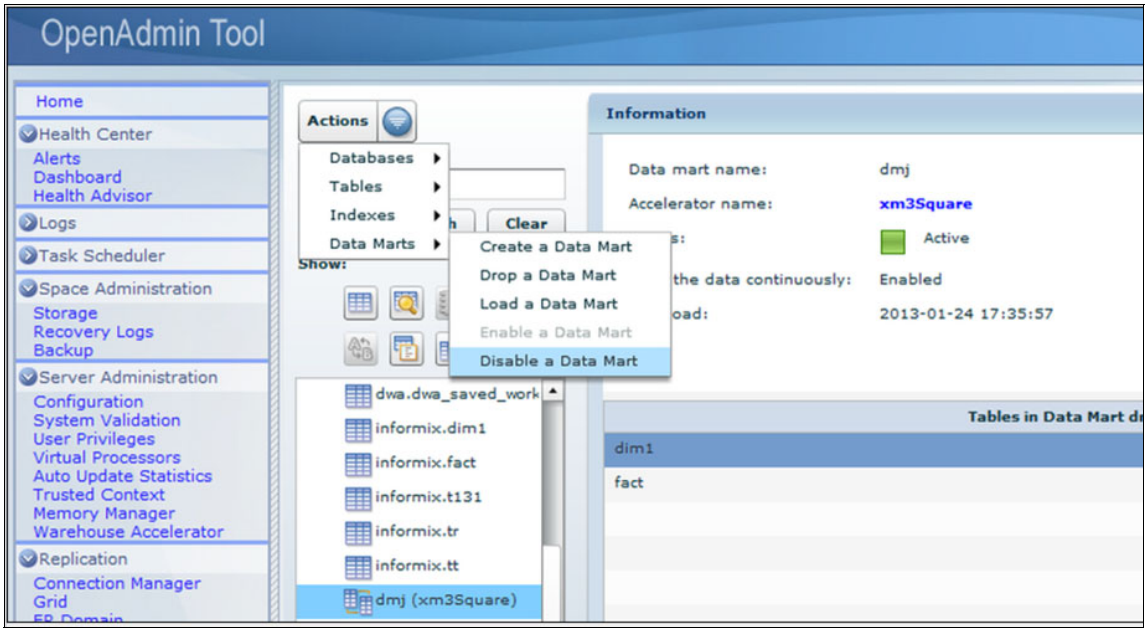


Figure 7-2 OAT supports the administration of Informix Warehouse Accelerator Version 12.10

OAT is also available as a native mobile application on Android and iOS. With OAT, you can accomplish the following tasks:

- Administer one or more databases and servers.
- Set up and administer simple to complex topology for Enterprise Replication and flexible grid.
- Monitor and manage storage.
- Monitor and tune query performance.

Starting with Informix 12.10, you can use OAT to set up and manage Informix Warehouse Accelerator and Informix Warehouse Accelerator data marts.

You can complete the following Informix Warehouse Accelerator tasks by using OAT:

- Create, drop, and monitor accelerators.
- Configure Informix to connect to one or more Informix Warehouse Accelerators.
- Design a data mart by analyzing the workload.
- Create or drop the data mart.
- Load the data mart.
- Perform a full reload or partition-based refresh of the data mart.
- Set up an automatic periodic refresh by using full reload or partition-based refresh or trickle feed.
- Enable and disable the data mart.

If you are upgrading from Informix 11.70 to Informix 12.10, you must drop and re-create your data marts so that all of the related data mart metadata is in place. This metadata is stored in the sysadmin database. You must grant the administrator the access privileges for this metadata as follows:

- `database sysadmin;`
- `execute function task (GRANT "admin","informix", "warehouse");`

7.1.2 The Informix and Informix Warehouse Accelerator Stored Procedure API

The Informix and Informix Warehouse Accelerator Stored Procedure API is built into every database on Informix 12.10. Use these stored procedures to manage Informix Warehouse Accelerator and its data marts. The stored procedures are the same as the one that are used by OAT for Informix Warehouse Accelerator management.

For usage and examples of the stored procedures, see the *IBM Informix Warehouse Accelerator Administration Guide*. These stored procedures can be used from any SQL client, for example, DB-Access scripts, ESQL/C, JDBC, ODBC, and .NET. You can program and automate administration and data synchronization tasks to fit your business requirements.

7.1.3 IBM Smart Analytics Optimizer Studio

IBM Smart Analytics Optimizer Studio (Figure 7-3) is an Eclipse-based administration interface that is available on Linux and Windows and is used to manage Informix Warehouse Accelerator data marts for Informix. It is integrated in to IBM Optim™ Studio, which provides you with an integrated interactive environment for data management functions with Informix. IBM Smart Analytics Optimizer Studio provides a graphical interface that you can use to interactively design and deploy a data mart. You can also use IBM Smart Analytics Optimizer Studio for basic administration of existing data marts and mart design, although not all of the functions are available. For detailed information about how to use IBM Smart Analytics Optimizer Studio, see Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69.

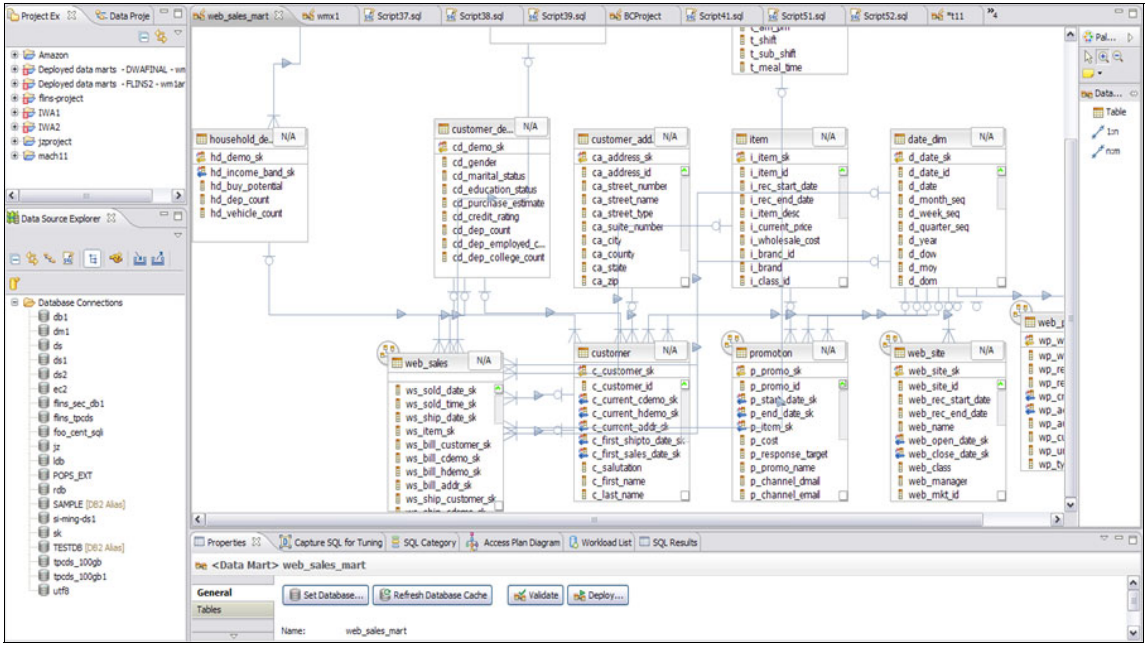


Figure 7-3 IBM Smart Analytics Optimizer Studio

7.2 Data synchronization methods

After the initial data load, the data on Informix database can change. You can refresh the data in Informix Warehouse Accelerator by using one of these methods:

- Perform a full data reload.
- Refresh the partitions that were modified since the last refresh.
- Trickle feed the data from Informix to Informix Warehouse Accelerator as the data is being inserted.

To do a full data reload, you must disable the data mart and then do a complete reload of the data mart. After you load the data, the data mart is enabled automatically. The new data is transferred just like the first data load, analyzed to create a new compression dictionary, and then compressed. Automatic partition refresh detects the fact and dimension table partitions that were modified since the last refresh and then removes those data partitions from Informix Warehouse Accelerator. When necessary, automatic partition refresh reloads the data from the affected partitions on to Informix Warehouse Accelerator. Automatic partition refresh also performs partition add, drop, attach, or detach activities to fact and dimension tables. Automatic partition refresh is useful when your data mart is designed for time cyclic data management.

Partition refresh is fast compared to a full data reload because it reuses the existing compression dictionary. Because of this reuse, a slight loss of compression efficiency is possible. Informix Warehouse Accelerator creates new versions of the modified rows and can run queries concurrent to partition refresh and trickle feed.

Trickle feed helps you achieve near real-time analysis of the data. Typically, you collect transaction data and append the new data to the fact table, for example, the sales table, and update on the dimension table, for example, and customer, product pricing. After you set up the trickle feed, Informix collects all of the data that is inserted to the fact table into a staging area. At the designated interval, Informix sends the data that was inserted to the fact table to Informix Warehouse Accelerator and does a partition-based refresh on all of the dimension tables. Dimension tables are typically smaller and the refresh is fast.

Note: When trickle feed is active, calling **`ifx_refreshMart()`** does not refresh fact tables.

You can run and set up all of the data refresh options by using OAT or built-in stored procedures.

7.3 Data mart states and transitions

After you load data to Informix Warehouse Accelerator, it can be analyzed immediately. If you are using Informix for a traditional data warehousing application, the data in Informix table might change in regular intervals as part of your ETL process. In an operational data store environment, applications continuously update the data. Either way, the data in Informix Warehouse Accelerator becomes out of sync with data in the Informix database. Because of this situation, Informix Warehouse Accelerator is most useful for seeing trends and patterns in the data by using aggregations. Informix Warehouse Accelerator is not useful for viewing a bank balance or the stock market.

Figure 7-4 shows the various stages of data mart creation, synchronization, and maintenance.

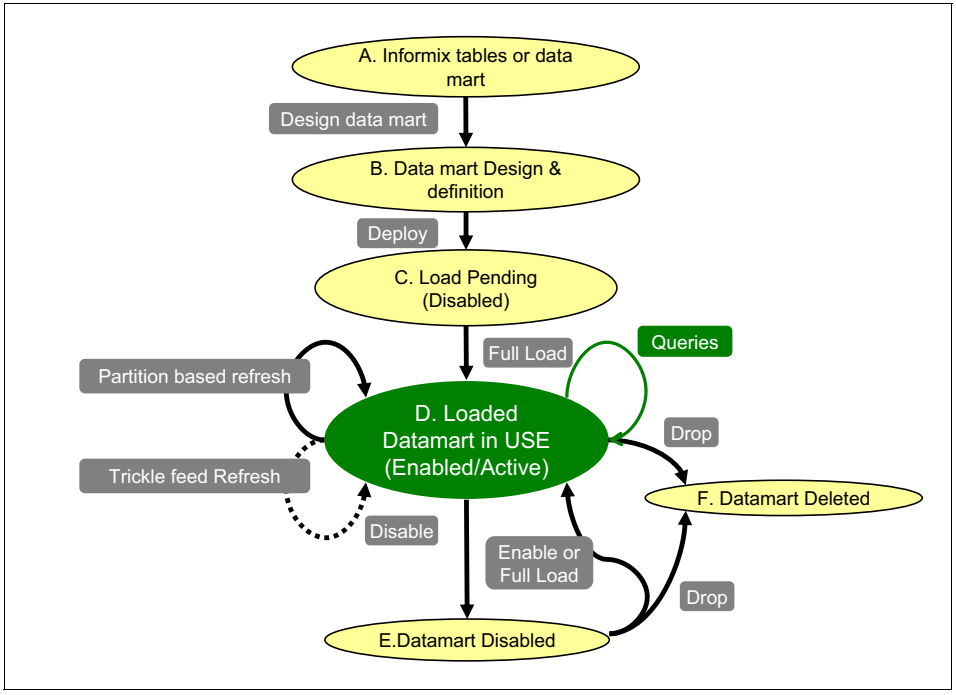


Figure 7-4 Data mart states, operations, and state transitions

To run each action or to change the state of the data mart, you can use the built-in stored procedure. Figure 7-5 shows the stored procedures to use for each respective action.

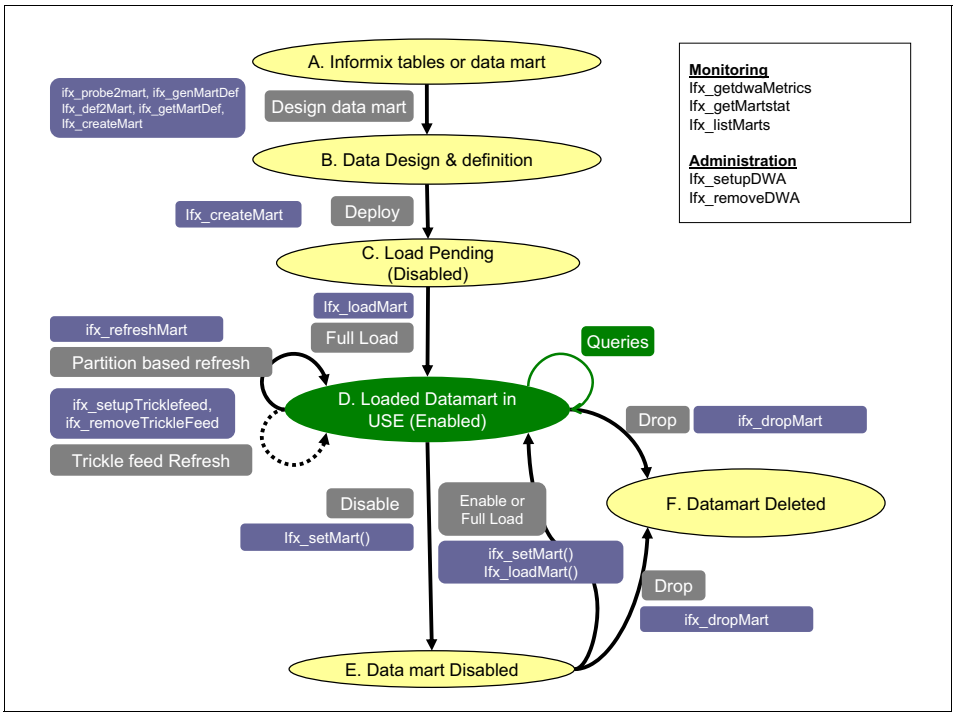


Figure 7-5 Informix and Informix Warehouse Accelerator stored procedures that are used for each action

7.3.1 Stage A: Informix tables or data marts

This stage consists of the following items:

- Action to take: Design the data mart.
- Tools to use: Open Admin Tool (OAT), Informix stored procedures, and IBM Smart Analytics Optimizer Studio
- Next stage: Data mart design with XML definition.

Begin with existing tables, a logical data mart, or logical data warehouse on an Informix database. You must identify a subset of tables, data marts to design, and the data mart to deploy on Informix Warehouse Accelerator. Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69 introduces you to designing the data marts in two ways:

- Design optimized data marts by using workload analysis.

  You can probe a set of queries to define an exact set of tables, columns, and relationships. The result of this probing is an XML document that is used for deploying the data mart on Informix Warehouse Accelerator. You can use OAT and the Informix stored procedures and tools for this task.

- Manually design data marts by selecting tables, columns, and relationships.

  You can interactively add tables and columns into a canvas, draw the relationships, and specify the join keys by using the IBM Smart Analytics Optimizer Studio.

Either approach produces a well-designed data mart definition.

Figure 7-6 shows a simple data mart with t as the fact table, and t1 and t2 as dimensions. They are related by n:m relationship of t2.k to t.k, and t1.k to t.k.

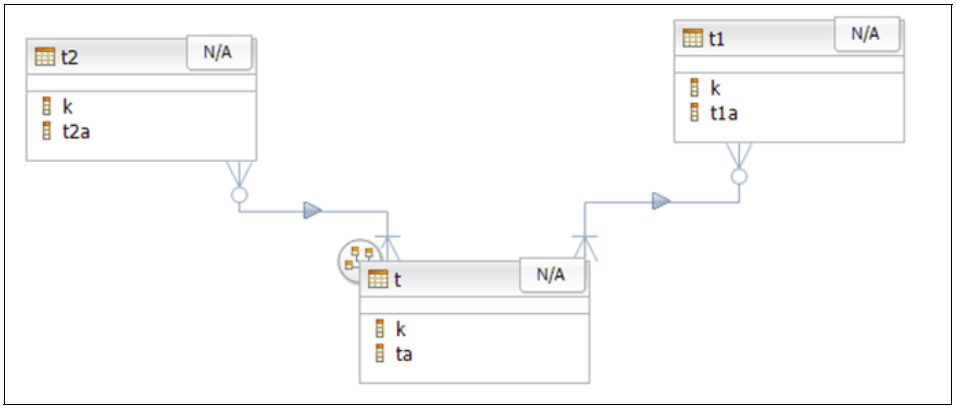


Figure 7-6 Simple data mart design

Example 7-1 shows the XML data mart that is created from the manual design by using IBM Smart Analytics Optimizer Studio.

Example 7-1 Data mart XML

```
<?xml version="1.0" encoding="UTF-8"?><aqtmart:martModel
xmlns:aqtmart="http://www.ibm.com/xmlns/prod/dwa" version="1.0">
  <mart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="t11">
    <table name="t" schema="dwa" isFactTable="true" >
```

```
      <column name="k"/>
      <column name="ta"/>
    </table>
    <table name="t2" schema="dwa">
      <column name="k"/>
      <column name="t2a"/>
    </table>
    <table name="t1" schema="dwa">
      <column name="k"/>
      <column name="t1a"/>
    </table>
    <reference dependentCardinality="n" dependentTableName="t"
dependentTableSchema="dwa" isRuntimeJoin="true" parentCardinality="n"
parentTableName="t1" parentTableSchema="dwa" referenceType="LEFTOUTER">
      <parentColumn name="k"/>
      <dependentColumn name="k"/>
    </reference>
    <reference dependentCardinality="n" dependentTableName="t"
dependentTableSchema="dwa" isRuntimeJoin="true" parentCardinality="n"
parentTableName="t2" parentTableSchema="dwa" referenceType="LEFTOUTER">
      <parentColumn name="k"/>
      <dependentColumn name="k"/>
    </reference>
  <guiConfig> … </guiConfig>
  </mart>
</aqtmart:martModel>
```

7.3.2 Stage B: Data mart design and definition

This stage consists of the following items:

- Input: Data mart design and its XML definition.
- Output: A data mart definition that is deployed on Informix Warehouse Accelerator (disabled) and data mart metadata (AQTs) that are created in the Informix database.
- Action to take: Deploy the data mart.
- Tools to use: Open Admin Tool (OAT), Informix stored procedures, and IBM Smart Analytics Optimizer Studio.
- Next stage: Deployed data mart.

You can export (by clicking **File** → **Export**) the data mart definitions that are created interactively from IBM Smart Analytics Optimizer Studio in to an XML document. You can also import an XML document that is created from query probing into IBM Smart Analytics Optimizer Studio (by clicking **File** → **import**) to manage it interactively. After you have the data mart design, you can use the definition in any of the tools.

In IBM Smart Analytics Optimizer Studio, you can click any area on the data mart canvas, then use the tabs in the **Properties** → **General** window to validate and deploy. Validate simply ensures that the tables and relationships are valid and can be sent to Informix Warehouse Accelerator. The deployment action sends the XML data mart definition to Informix Warehouse Accelerator. IBM Smart Analytics Optimizer Studio asks if you want to automatically LOAD the data after the deployment. For detailed examples about how to deploy a data mart in IBM Smart Analytics Optimizer Studio, see Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69.

In OAT, after you stop probing, analyze the data to get the data mart definition. Next, you can use the data mart definition to deploy to Informix Warehouse Accelerator.

You can also deploy the data mart with the stored procedure and the data mart XML definition. This call creates a data mart named saleseast by using the `sales_east_mart.xml` file, which is available on the current directory of the client machine. **`FILETOCLOB`** is a Informix routine that sends the `sales_east_mart.xml` file to the server and converts it to a CLOB.

```
EXECUTE FUNCTION
ifx_createMart('saleseast',FILETOCLOB('sales_east_mart.xml',
'client'));
```

If the definition is on the server, change the **`FILETOCLOB`** parameter as follows:

```
EXECUTE FUNCTION
ifx_createMart('saleseast',FILETOCLOB('sales_east_mart.xml',
'server'));
```

After a successful deployment, Informix Warehouse Accelerator registers the snowflake schema in your data mart definition, creates metadata to store within Informix Warehouse Accelerator, and sends additional metadata on the data mart to Informix as a set of **`SELECT`** statements. These **`SELECT`** statements are the accelerated query tables (AQTs) and represent the logical definition of the data mart. Within Informix, this is saved as specially marked views on the **`SELECT`** statements that join the tables in the star schema and project the accelerated columns.

Here is a SELECT statement for the AQT that represents the three table data mart that was shown earlier:

```
CREATE VIEW "informix"."aqt064c56e4-a8d4-4edf-9a9b-e7b8717cc3e9" ("CO
         L2","COL3","COL8","COL9","COL5","COL6") AS SELECT x0."k" ,x0.ta ,x1."
         k" ,x1.t1a ,x2."k" ,x2.t2a from (("usr1"."t" x0 LEFT JOIN "usr1".
         t1 x1 ON (x0."k" = x1."k" ) )LEFT JOIN "usr1".t
```

You can view the defined AQTs manually by selecting them from the system catalog tables SYSTABLES and SYSVIEWS:

```
SELECT * from "informix".SYSTABLES where tabname like 'aqt%';

SELECT b.tabname, b.tabid, a.viewtext
FROM SYSVIEWS a, SYSTABLES b
WHERE a.tabid = b.tabid
AND b.tabname LIKE 'aqt%'
ORDER BY b.tabid, a.seqno;
```

After a successful deployment and before the data load, the Informix Warehouse Accelerator considers this data mart to be deployed, but disabled for acceleration. The data mart status is shown as disabled.

7.3.3 Stage C: Loading the pending data mart (disabled)

This stage consists of the following items:

- Input: A deployed data mart on Informix Warehouse Accelerator (disabled).
- Output: A loaded data mart that can be used for acceleration (enabled).
- Action to take: A full data mart load.
- Tools to use: Open Admin Tool (OAT), Informix stored procedures, and IBM Smart Analytics Optimizer Studio.
- Next stage: An enabled data mart that is used for query processing.

To use the query processing power of Informix Warehouse Accelerator, you must copy the data from Informix to Informix Warehouse Accelerator using a process that is called data load. During a full load, the complete set of data for each table and accelerated column in the data mart is loaded to Informix Warehouse Accelerator. During this full load, Informix Warehouse Accelerator analyzes the data and creates a compression dictionary by using frequency partitioning and modified Huffman encoding. Informix Warehouse Accelerator uses all of the available worker and coordinator resources to create the compression dictionary and encode all of the fact and dimension tables.

From OAT, you can load the data mart once or set up a full reload of the data mart continuously. For more information, see Figure 7-7.

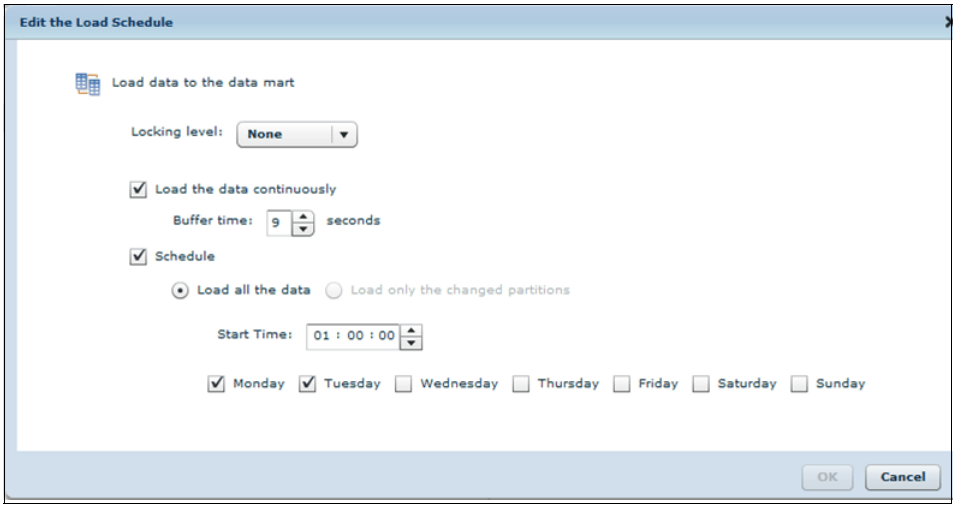


Figure 7-7 Load the data mart

For each successfully enabled data mart, Informix records when the data was loaded, and what was the update, delete, and insert (UDI) counter on the base Informix table and partition. This information is stored in the metadata tables of the sysadmin database.

You can use the built-in stored procedure to load the data mart as well:

```
EXECUTE FUNCTION ifx_loadMart('acmeacc', 'saleseast', 'MART');
```

The accelerator name is acmeacc, the data mart name is saleseast, and the locking mode on the tables in the data mart is ‘MART’. In this example, all of the tables in the mart are locked in the database during the load. A shared lock is placed on all these tables. Concurrent applications can view the data in the database but cannot modify it. The ‘`TABLE`’ value places the shared lock on each table as the tables are read and `'NONE'` does a dirty read on the tables. You can choose a locking mode that is based on your concurrency and consistency requirements.

7.3.4 Stage D: (Enabled) Loaded data mart in use

In this stage, the data mart is active and operational. Several different actions can be performed on the data mart, each bringing the data mart to a different next stage, as shown in Figure 7-8.

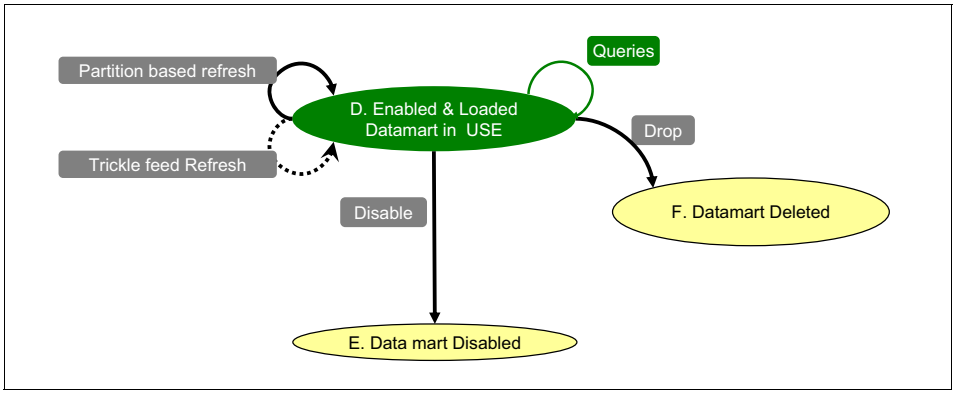


Figure 7-8 Data mart actions and transitions from the active state

This stage consists of the following items:

- Input state: (Active/Enabled) Loaded data mart in use
- Action to take: Do one of the following actions:
  - Query processing
  - Configure a partition based refresh
  - Configure a trickle feed
  - Disable the data mart
  - Drop the data mart
- Tools to use: Open Admin Tool (OAT), Informix stored procedures, and IBM Smart Analytics Optimizer Studio
- Next stage: Depending on the action that is taken, the next stage is one of the following ones:
  - Enabled data mart used for query processing
  - Disabled data mart
  - Deleted data mart

Query processing

Immediately after loading the data, applications can query the data mart on Informix Warehouse Accelerator. The queries can originate from one or more Informix sessions and Informix Warehouse Accelerator data is in continuous use.

If you define multiple data marts that involve a common set of tables, a query can match multiple AQTs. For example, one data mart can contain sales and stores and another data mart can contain sales and customers. A sales table query can use either the data mart with sales and stores, or sales and customers. In this situation, the Informix server chooses the data mart that was most recently fully loaded.

Configuring partition-based refresh

Although you can always use the complete data load option, it can take a long time to load the entire data mart and during that load time, the data mart is unavailable for analysis. To avoid a complete data load, you can examine the patterns of usage in applications and how their data is updated. Fact tables usually are large and partitioned, and on a daily basis, few of the partitions in the fact table receive new data. In addition, the partitions in the fact tables are attached, detached, added, or dropped for efficient data management. Dimensions tables often are smaller and updates to dimension tables can be anywhere. Informix monitors and saves statistical data on each partition of the data mart in the sysadmin database.

Figure 7-9 shows a data mart with a partitioned sales fact table with two dimensions: Customer and stores.

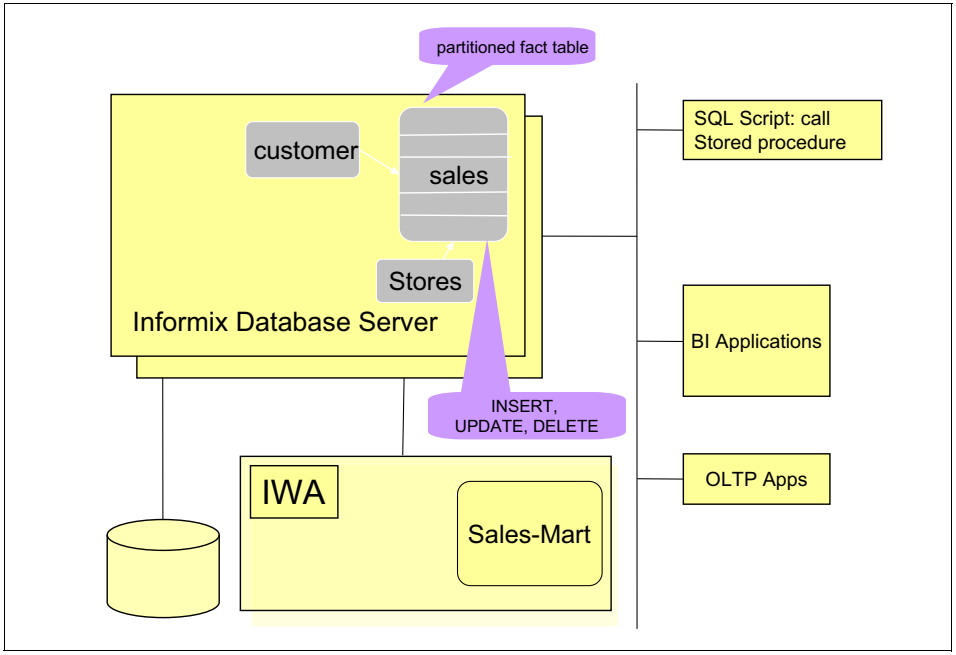


Figure 7-9 Partition-based refresh for the fact table

Automatic data mart refresh uses the statistical information on each table and partitions in the data mart to reload the partitions that are updated in the database since the last load or refresh. In addition, it recognizes the partitions added, dropped, attached, or detached to tables and reflects these changes on the Informix Warehouse Accelerator data mart.

The data refresh process reuses the data dictionary on the Informix Warehouse Accelerator, which avoids the resource-intensive task of creating the compression dictionary for the data. However, new values that are added since the dictionary was created are not compressed. Therefore, you can fully reload the data when you expect significant data skew or when you can afford the downtime.

Another advantage of automatic data mart refresh is that queries can run concurrently during the data refresh. Informix Warehouse Accelerator has an internal system that creates versions for the data, so your queries see a consistent picture of the data. For this reason, you must do a full load of a data mart after 32767 refreshes. This requirement means that you must fully reload the data after more than hundred days, even if you refresh every five minutes. This is a reasonable situation for data analysis. Here is an example of this situation:

```
ifx_refreshMart('accelerator_name', 'data_mart_name', 'locking_mode', NULL);
```

Where:

- `accelerator_name`: The name of the accelerator on which you want to refresh the mart.
- `data_mart_name`: The name of the data mart to refresh.
- `locking_mode`: This can be **"**`TABLE`**"**, **"**`MART`**"**, or **"**`NONE`**"**. If you pass `NULL`, Informix reuses the locking mode that is used by the previous full load.
- The fourth parameter is reserved and unused now. Pass `NULL` for now.

Note: When trickle feed is active, calling **`ifx_refreshMart()`** does not refresh fact tables.

Calling **`ifx_refreshMart()`** refreshes the mart only once. Using Informix DBScheduler or a system cron job, you can schedule **`ifx_refreshMart()`** to run at regular intervals.

When you use the Open Admin Tool (OAT), a simple way to set up automatic data mart refresh is to edit the load schedule within the data mart information page. Then, select the **Load only the changed partitions** check box, time of day, and days of refresh, as shown in Figure 7-7 on page 155.

Configuring the trickle feed

Trickle feed helps you to achieve near real-time analysis of the data. Most applications append or INSERT data to the fact table, for example, sales, and update data in dimension tables, for example, customer and stores.

Figure 7-10 shows the data flow in a trickle feed from Informix to Informix Warehouse Accelerator.

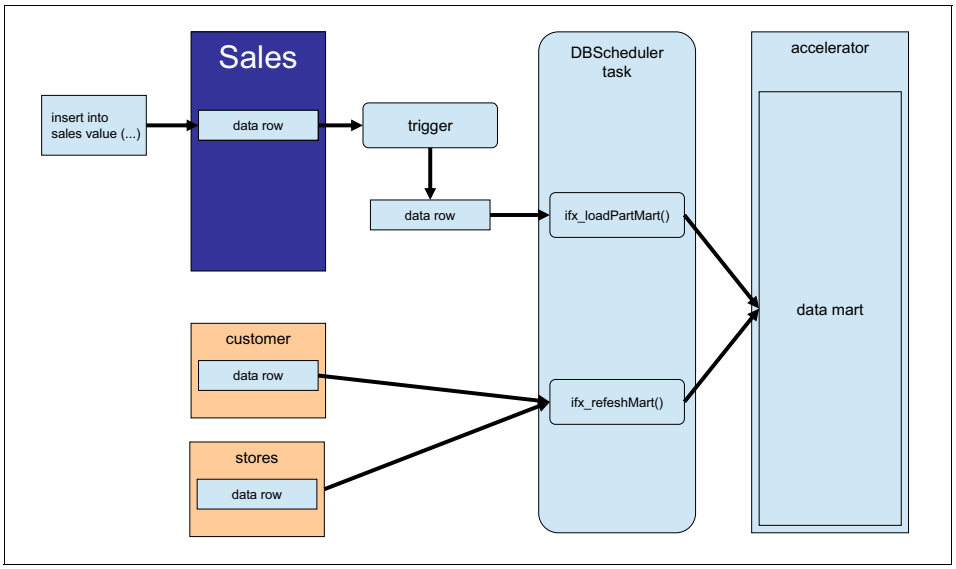


Figure 7-10 Data flow in a trickle feed from Informix to Informix Warehouse Accelerator

After you set up the trickle feed, Informix installs a trigger on the fact table that collects all of the inserted data in the fact table in to a staging area. At the designated interval, Informix sends the data that is inserted to the fact table to Informix Warehouse Accelerator and does a partition-based refresh on all of the dimension tables. Dimension tables are typically small and the refresh is fast.

If the trickle feed is active on the data mart, calling **`ifx_refreshMart()`** does not refresh fact tables.

You can set up the trickle feed by using OAT. Edit the load schedule within the data mart information, select the **Load the data continuousl**y check box, and specify an appropriate time for the interval, as shown in Figure 7-7 on page 155.

You can also set up a trickle feed by using the stored procedure:

```
ifx_setupTrickleFeed( 'accelerator_name', 'data_mart_name', buffertime)
```

Where:

- `accelerator_name`: The name of the accelerator that contains the data mart.
- `data_mart_name`: The name of the data mart.
- `buffertime`: An integer that represents the time interval between refreshes and whether dimension tables are refreshed. A positive number sets up the refresh of all the tables in the data mart and a negative number sets up the trickle feed for fact tables only.

Here are two trickle feed examples:

- Trickle feed for all the tables in the data mart partsmart:

  ```
  execute procedure ifx_setupTrickleFeed('salesacc', 'partsmart', 60);
  ```

- Set up trickle feed for *only* the “fact table” in the carmart data mart:

  ```
  execute procedure ifx_setupTrickleFeed('salesacc', 'carmart', -300);
  ```

Disabling a data mart

There are two reasons to disable a data mart:

- To do a full reload of the data mart, to sync up the data, and to get a better compression rate.
- When multiple data marts have the same table, to disable the data marts that have older data.

All of the tools provide a simple interface to disable the data mart. Figure 7-11 on page 161 shows how to disable a data mart in OAT.

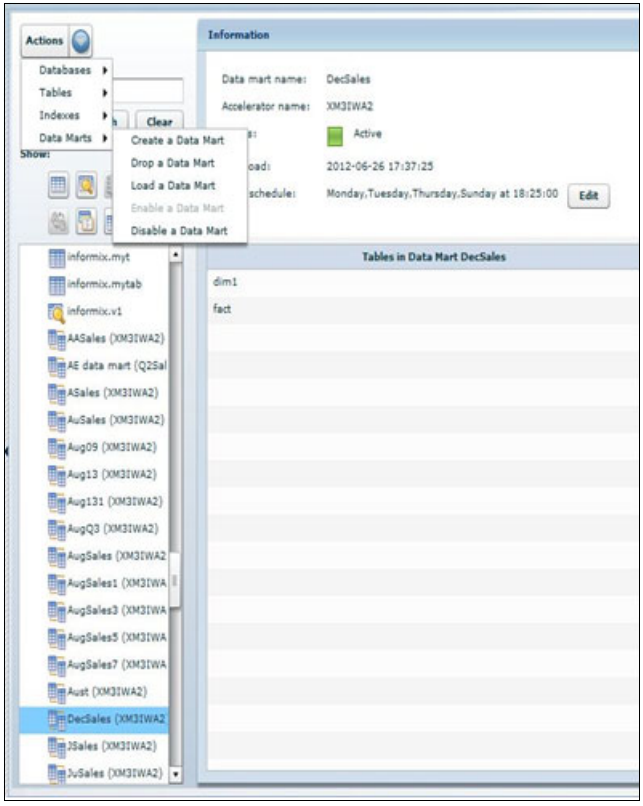


Figure 7-11 Disabling data mart in OAT

Here is the stored procedure example:

```
EXECUTE FUNCTION ifx_setMart('salesacc', 'sales_east_mart', 'OFF');
```

Dropping a data mart

When you want to redesign the data mart, you can drop and re-create the data mart with the required columns, tables, and relationships. All of the tools provide an interface to drop the data mart. Here is a stored procedure example:

```
EXECUTE FUNCTION ifx_dropMart('salesacc', 'sales_east_mart');
```

7.3.5 Stage E: Data mart disabled

Figure 7-12 illustrates the data mart in the disabled state.

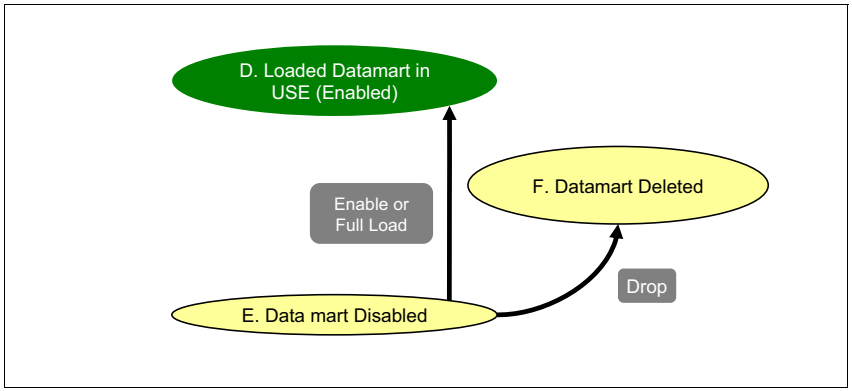


Figure 7-12 Data mart actions and transitions from the disabled state

This stage consists of the following items:

- Actions: Perform one of the following actions:
  a. Enable the data mart.
  b. Perform a full reload of the data.
  c. Drop the data mart.
- Tools to use: Open Admin Tool (OAT), Informix stored procedures, and IBM Smart Analytics Optimizer Studio
- Next stage: Depending on the action that is taken, the next stage is one of the following ones:
  a. Enabled data mart ready to use
  b. Dropped data mart

From this stage, you can enable the data mart to make the data mart available for query processing. New queries immediately consider this data mart for acceleration. Data that is previously loaded is used for query processing. You must use one of the data synchronization options to get Informix Warehouse Accelerator data in sync.

Here is an example of a statement for enabling a data mart:

```
EXECUTE FUNCTION ifx_setMart('salesacc', 'sales_east_mart', 'ON')
```

You can do a full data reload when your business can tolerate the data mart downtime. A full data reload reuses the data mart definition and analyzes the data to create a compression dictionary and encodes all the data again.

Here is an example of a statement for a full data reload of a data mart:

```
EXECUTE FUNCTION ifx_loadMart('salesacc', 'sales_east_mart', 'NONE' )
```

You can also drop the data mart from this stage by using the following statement:

```
EXECUTE FUNCTION ifx_dropMart('salesacc', 'sales_east_mart', 'OFF')
```

7.3.6 Stage F: Data mart deleted

When the data mart is deleted from Informix Warehouse Accelerator, its metadata from the Informix databases is also deleted. You can redeploy or re-create the data mart by using a previous design or XML definition.

Here is an example of a statement for dropping a data mart:

```
EXECUTE FUNCTION ifx_dropMart('salesacc', 'sales_east_mart', 'OFF')
```

7.4 Schema changes on Informix

The Informix schema is loosely connected to the Informix Warehouse Accelerator data mart. If a table in an Informix data mart is accelerated by using Informix Warehouse Accelerator, you must manage the logical and physical schema changes to this table as part of your change process to minimize or eliminate downtime and maximize availability.

The logical schema changes include adding or dropping or a column, changing the type of the column, or changing the constraints. Before you modify the schema for tables that are in a data mart, drop the data mart. After applying the schema changes in the database, redesign or redeploy the data mart to Informix Warehouse Accelerator.

Physical schema changes include changes to partitioning or fragmentation schemes, and adding, dropping, attaching, or detaching partitions to the tables. Some examples of changes to partitioning schemes are changing the expression in expression-based partitioning, or changing from round robin to expression-based, interval, or list partitioning schemes. In these situations, you can fully reload the data mart so that later data refreshes are efficient. Adding, dropping, attaching, and detaching partitions to tables is handled by automatic data mart refresh.

8

IBM Informix Warehouse Accelerator server: Administration tasks

Here are the main administration tasks in an Informix Warehouse Accelerator environment:

- Maintaining the Informix Warehouse Accelerator objects, the data marts, and their data
- Administering Informix Warehouse Accelerator

The monitoring focus on Informix Warehouse Accelerator is different from the Informix server because tasks such as disk space management are not required. The Informix Warehouse Accelerator distribution provides the **ondwa** utility for all of your administration purposes. The **ondwa** utility provides a single point of access to administer and monitor Informix Warehouse Accelerator. The Informix Warehouse Accelerator management tasks include resource assignment and continuous assignment validation on the host system.

This chapter contains an overview of the operating system resources that are used and how to monitor resource usage. Additional administration activities are described, including how to monitor data mart objects.

8.1 Administering the Informix Warehouse Accelerator server by using the ondwa utility

Certain administration tasks are required to provide efficient and accurate information, including operating system resource usage and the server component monitoring. Because Informix Warehouse Accelerator is an in-memory data server, some of the common database administration tasks are not required.

You can use the **ondwa** command to set up a new Informix Warehouse Accelerator server (*accelerator server*), start and stop an accelerator server, or remove an accelerator server that is no longer used.

Here are the **ondwa** options for maintaining an accelerator server:

- **setup**
- **start**
- **stop**
- **reset**
- **clean**

8.1.1 ondwa setup

The **ondwa setup** command creates the files and subdirectories that are required to run an accelerator server. You can also use this command to reinstantiate a stopped accelerator server to apply any configuration parameters changes, for example, if you increased the **NUM_NODES** parameter to add more hardware in a cluster environment.

The **ondwa setup** command creates the required disk structures for the Informix Warehouse Accelerator nodes, their configuration files, log files, and the directories for their communication. However, the command does not start the accelerator server. After you run **ondwa setup**, the directory that is specified by the **DWADIR** configuration parameter contains the files and subdirectories. New log and configuration files are created for each node. A structure of subdirectories is put in place. For example:

```
informix@node1:/IBM/informix/IWA> ls
DWA_CM_node0  DWA_CM_node2      local       node0.log   node1.log node2.log
DWA_CM_node1  DWA_watchdog.log  node0.conf  node1.conf  node2.conf  shared
```

The **ps** command shows that the accelerator server is not started because there are no Informix Warehouse Accelerator processes:

```
ps -eaf | grep DWA
/* nothing running */
```

8.1.2 ondwa start

The **ondwa start** command starts all of the accelerator server nodes. The accelerator server becomes operational when all the required processes for the coordinator node, the worker node, and the watchdog process are running. You must run **ondwa start** after the Linux system that hosts Informix Warehouse Accelerator is restarted or after you run **ondwa setup**. After you start the accelerator server, you see the new processes running. The processes are spread across the underlying system resources whether Informix Warehouse Accelerator is in an SMP environment or cluster environment.

This example shows the Informix Warehouse Accelerator-related processes in an SMP system:

```
#single machine
informix@node1:/> ps -eaf | grep DWA
informix 23168     1  0 Jun06 ?        00:05:03 ./DWA_CM_node0 --no-console
informix 23171     1  0 Jun06 ?        00:04:52 ./DWA_CM_node1 --no-console
root     23208     1  0 Jun06 ?        00:00:01 DWA_watchdog -daemon 0
```

This example shows the Informix Warehouse Accelerator-related processes in a cluster environment:

```
#cluster
#local machine
informix@node1:/> ps -eaf | grep DWA
informix   768 1  0 Jun06 ?        00:03:13 ./DWA_CM_node0 --no-console
root       892 1  0 Jun06 ?        00:00:01 DWA_watchdog -daemon 0
#Node2
informix@node1:/> ssh node2 ps -eaf | grep DWA
informix 21512     1  0 Jun06 ?        00:19:29 ./DWA_CM_node1 --no-console
root     21546     1  0 Jun06 ?        00:00:01 DWA_watchdog -daemon 1
```

8.1.3 ondwa stop

The **ondwa stop** command stops the accelerator server, including all Informix Warehouse Accelerator processes. After you run this command, the Informix server cannot access any of the accelerators that were created on the stopped accelerator server.

You can run the following command to verify that the accelerator server was stopped:

```
ps -eaf | grep DWA
/* nothing running */
```

After you stop the accelerator server, you can restart it by running **ondwa start**. To apply any configuration parameter changes, you must run **ondwa setup** before you restart the accelerator server.

8.1.4 ondwa reset

This command removes all data that is related to data marts from the shared and local subdirectories of your **DWADIR**. After you run **ondwa reset**, the accelerator server is in the state where it was after the initial **ondwa setup** command. To avoid leftover data mart metadata (for example, AQTs) in the Informix server databases, it is better to first drop all data marts individually before running **ondwa reset**.

8.1.5 ondwa clean

The **ondwa clean** command removes all accelerator server data that is maintained in the directory that is specified by the **DWADIR** configuration parameter. After this operation, the accelerator server does not physically exist. This operation is required for removing an accelerator server or for preparing an existing instance to apply a change in the number of nodes in the configuration file. When you change the value of the **NUM_NODES** parameter in `dwainst.conf`, you bring down the Informix Warehouse Accelerator server, clean the files by running **ondwa clean**, set up a new instance by running **ondwa setup**, and start the instance again by running **ondwa start**. As with the **ondwa reset** command, it is better to first drop all existing data marts individually before running **ondwa clean** because it is better to avoid leftover data mart metadata in the Informix server databases.

You can run **ondwa clean** only after the accelerator server is stopped.

8.2 Access token generation

The Informix server and the accelerator server must be enabled to exchange data. This enablement operation is also known as “creating an *accelerator*”. For more information about the difference between an accelerator server and an accelerator, see A.2, “The ondwa utility” on page 219. When you create an accelerator, information for the initial connection from the Informix server to the accelerator server is obtained by running **ondwa getpin**. The output of the command contains the IP address and port number of the accelerator server coordinator node that receives connection requests from Informix Servers. It also contains a pairing code that controls access to the accelerator server. This pairing code is valid for 30 minutes after you run **ondwa getpin**. You must create the accelerator on the Informix server side within 30 minutes by using this pairing code. When the new accelerator is created, a new authorization token is generated and stored in the `sqlhosts` file of the Informix server, along with the connection information for the new accelerator.

To create an accelerator, complete the following steps:

1. To get the initial connection information, including the pairing code, run **ondwa getpin** in the accelerator server environment.

   The first column of the output is the IP address where the coordinator node accepts connection requests. The second column is the port number where the coordinator node accepts connection requests. The third column is the paring code.

   ```
   informix@node1:~> ondwa getpin
   192.168.0.1 21022 5840
   ```

2. You can create the accelerator on the Informix server by using OAT or by running the built-in **ifx_setupDWA()** function of the stored procedure (SP) API. Use a unique arbitrary name for the new accelerator and the output from the **ondwa getpin** command to specify the parameter values.

   ```
   #Informix server
   dbaccess -e sysmaster << EOF
   execute function ifx_setupDwa(“DWA”,”192.168.0.1”,“21022”, “5840”);
   EOF
   ```

8.3 Monitoring Informix Warehouse Accelerator

You can use the **ondwa** utility interface to monitor an accelerator server. You can monitor the following items:

- Availability and accelerator server layout
- Resource utilization

To monitor availability and layout, use the **ondwa status** command.

The output of the **ondwa status** command contains your recent startup and shutdown activities. In addition, you can view all necessary information about the number and availability of nodes. Here is an example output from the **ondwa status** command:

```
ID    | Role        | Cat-Status  | HB-Status    | Hostname | System ID
------+-------------+-------------+--------------+----------+-----------
0     | COORDINATOR | ACTIVE      |   Healthy    | node1    | 1
1     | WORKER      | ACTIVE      |   Healthy    | node2    | 2
2     | WORKER      | ACTIVE      |   Healthy    | node3    | 3

Cluster is in state    : Fully Operational
Expected node count    : 1 coordinator and 2 worker nodes
```

To monitor workload and resource usage on your configured nodes in the accelerator server, use the **ondwa tasks** command.

The output of the **ondwa tasks** command contains a detailed list of workload and memory usage of the accelerator server. You can monitor the current workload and view the usage and status of the nodes. You also can track the memory usage of the nodes. Memory is the most critical resource for the accelerator server. Example 8-1 shows the output of two different runs of the **ondwa status** command when queries are being processed by the accelerator server.

Example 8-1 ondwa tasks output

```
-----------------+------------+---------+---------+--------+------------
  Location       | Status     | Progr.  | Upd. ms | Memory | Monitor
-----------------------------+---------+---------+--------+------------
  Primary Node 0 | OPNQRY     |       0 |      15 |   194K | Fine
  -> Node 2      | Dim 26140  |       0 |       4 |    18M | Fine
  -> Node 4      | Dim 18120  |       0 |       4 |   118M | Fine
  -> Node 5      | Fact       |       0 |       6 |    42M | Fine
  -> Node 7      | Dim 14140  |       0 |       4 |   112K | Fine
  (Total Memory) |            |         |         |   179M |
-----------------+------------+---------+---------+--------+------------
```

```
------------------+--------------+---------+---------+--------+-----------
  Location        | Status       | Progr.  | Upd. ms | Memory | Monitor
------------------+--------------+---------+---------+--------+----------
  Primary Node 0  | OPNQRY       |       0 |      14 |    66K | Fine
  -> Node 2       | <Initialized> |       0 |       2 |    75K | Fine
  -> Node 4       | Dim 9658     |       0 |       8 |  7309K | Fine
  -> Node 5       | <Initialized> |       0 |       4 |   111K | Fine
  -> Node 7       | Dim 9659     |       0 |       7 |    36M | Fine
  (Total Memory)  |              |         |         |    43M |
---------------------------------+---------+---------+--------+----------
```

8.4 Monitoring operating system usage

Memory, processor, network throughput, and storage are key resources that an operating system provides to applications. The goals of resource monitoring are to detect resource bottlenecks, identify the root cause of the bottlenecks, and define a recovery strategy to bring the system performance to a wanted level. To define a successful strategy to reach these goals, you must understand which resources are used by Informix Warehouse Accelerator and what they are used for.

8.4.1 Monitoring memory usage

Informix Warehouse Accelerator is an in-memory database server, so memory is an important resource. The Informix Warehouse Accelerator processes use various types of memory, including shared memory. Informix Warehouse Accelerator uses shared memory to store data mart metadata and user data. The maximum size of the shared memory is defined by the **COORDINATOR_SHM** and **WORKER_SHM** configuration parameters in Informix Warehouse Accelerator. The Informix Warehouse Accelerator processes also privately allocated memory that is used for various purposes, depending on the tasks. For data loading, the processes use private memory for building the dictionary and compressing the data. For running queries, the processes use private memory for intermediate results, and grouping and sorting operations.

It is important to monitor memory usage because when you build an Informix Warehouse Accelerator environment, you start with an initial number of data marts and a projected data storage size. Eventually, the system grows over time, new data marts are created, and older applications are replaced. Old data mart definitions must be adjusted to meet your current requirements. All these factors influence the allocation and use of shared and private memory.

You can use various methods to check the current Informix Warehouse Accelerator memory usage. As a starting point, you can monitor the amount of memory that is used by using the **ondwa tasks** command. For an example output of this command, see Example 8-1 on page 170. Use the memory column of the output for your analysis.

You can also use Linux system utilities to look at the memory usage of the Informix Warehouse Accelerator processes. Different Linux distributions offer slightly different methods. For example, on a Ubuntu distribution, you can use the **pmap** command.

Example 8-2 shows how to monitor the memory usage of a particular Informix Warehouse Accelerator node. Carefully review the private memory statistics in the summary section at the end of the output.

Example 8-2 pmap command output on a Ubuntu distribution

```
#identify the process id
informix@node1:/> ps -eaf | grep DWA
informix 23168     1  0 Jun06 ?        00:05:22 ./DWA_CM_node0 --no-console
#use pmap
informix@node1:/> pmap 23168
23168: DWA_CM_node0
START               SIZE     RSS     PSS   DIRTY    SWAP PERM MAPPING
0000000000400000  38172K  10948K  10948K      0K      0K r-xp DWA_CM
0000000002b4d000 169900K 145508K 145508K 145508K      0K rw-p [heap]
000041ec84100000   5688K   5688K   5688K   5620K      0K rw-s
/dev/shm/ibm_aqt_data.inst_informix.node_0.mart_55.type_0.addr_0x41ec841000
000041ec8468e000 37553608K      0K      0K      0K      0K rw-s
/dev/shm/ibm_aqt_data.inst_informix.node_0.mart_55.type_0.addr_0x41ec841000
.....
0007fff26a1e000     84K     28K     28K     28K      0K rw-p [stack]
Total:           39216676K 542612K 541334K 527424K      0K

1495600K writable-private, 161780K readonly-private, 37559296K shared, and
542612K referenced
```

On SUSE and Red Hat distributions, the values VmHWM and VmRSS of the pseudo file /proc/<pid>/status give useful information about the total memory usage of an individual Informix Warehouse Accelerator node, which is shown in Example 8-3.

Example 8-3 Information from pseudo file /proc/<pid>/status

```
#identify the process id
informix@node1:/> ps -eaf | grep DWA
informix 23168     1  0 Jun06 ?        00:05:22 ./DWA_CM_node0 --no-console
#get information from /proc/<pid>/status
```

```
informix@node1:/> cat /proc/23168/status | egrep 'VmHWM|VmRSS'
VmHWM:   3223156 kB
VmRSS:   2844324 kB
```

VmHWM shows the peak size (high water mark) of resident memory that is used by the process. VmRSS shows the size (resident set size) of memory currently in use by the process. Both values include shared and privately allocated memory.

There are several methods to figure out how much memory Informix Warehouse Accelerator uses to store the data marts. This amount corresponds to the shared memory that is allocated by the worker nodes.

The easiest method is to get the list of data marts in an accelerator by using the listMarts operation. This list includes the size information about each data mart. If there are several different accelerators in Informix Warehouse Accelerator, then you must repeat this operation for each accelerator. Example 8-4 shows the list of data marts in an accelerator named iwa1.

Example 8-4 List of data marts in accelerator iwa1

```
cat << EOF | dbaccess db1 -
> EXECUTE FUNCTION ifx_listMarts('iwa1');
> EOF

Database selected.

(expression)  ROW('mart1','Active',     2    ,...)
(expression)  ROW('mart2','LoadPending',NULL,NULL)
(expression)  ROW('mart3','Active',     19   ,...)
(expression)  ROW('mart4','Active',     1535,...)
(expression)  ROW('mart5','Active',     12   ,...)
(expression)  ROW('mart6','Active',     3    ,...)

6 row(s) retrieved.

Database closed.
```

Six data marts are listed in Example 8-4, but data mart 2 has not loaded the data yet and therefore no size information is available. The other five data marts are active and their size is shown in megabytes (MB). For example, 2 MB for mart1 and 1535 MB for mart4.

Another method that you can use is to view the shared memory in use on the system from the Linux perspective. Unlike the Informix server, Informix Warehouse Accelerator uses memory mapped files to allocate shared memory. Therefore, the shared memory that is used by Informix Warehouse Accelerator is not shown in the output of the **ipcs** system utility. Instead, the memory mapped files are shown in directory `/dev/shm` or `/run/shm`, depending on the Linux distribution.

However, the sizes of files that are shown by running **ls -l** on this directory reflect the allocated shared memory size, which usually is not equal to the size of shared memory in use. To get the size of shared memory that is used, you must look at the memory map of the processes. You can do this either by using the **pmap** command for Ubuntu, or by using the information in pseudo file `/proc/<pid>/smaps`, as shown in Example 8-5.

Example 8-5 Information from pseudo file smaps on shared memory that is used by a process

```
#identify the process id
informix@node1:/> ps -eaf | grep DWA
informix 23168     1  0 Jun06 ?        00:05:22 ./DWA_CM_node0 --no-console
#get information from /proc/<pid>/smaps
informix@node1:/> cat /proc/23168/smaps| egrep 'ibm_aqt_data|Size:'
...
4335ff400000-4335ff604000 ... /dev/shm/ibm_aqt_data...mart_225...
Size:              2064 kB
4335ff604000-4339a6800000 ... /dev/shm/ibm_aqt_data...mart_225...
Size:          15321072 kB
43c0d2c00000-43c132b06000 ... /dev/shm/ibm_aqt_data...mart_263...
Size:           1571864 kB
43c132b06000-43c47a000000 ... /dev/shm/ibm_aqt_data...mart_263...
Size:          13751272 kB
43f79f800000-43f79fb05000 ... /dev/shm/ibm_aqt_data...mart_278...
Size:              3092 kB
43f79fb05000-43fb46c00000 ... /dev/shm/ibm_aqt_data...mart_278...
Size:          15320044 kB
43feee000000-43feef30e000 ... /dev/shm/ibm_aqt_data...mart_280...
Size:             19512 kB
43feef30e000-440295400000 ... /dev/shm/ibm_aqt_data...mart_280...
Size:          15303624 kB
440295400000-44029601c000 ... /dev/shm/ibm_aqt_data...mart_281...
Size:             12400 kB
44029601c000-44063c800000 ... /dev/shm/ibm_aqt_data...mart_281...
Size:          15310736 kB
...
```

In the output that is shown in Example 8-5 on page 174, the lines that show a shared memory allocation, immediately followed by the corresponding line that contains the size, are of interest. The example was trimmed to remove lines that show sizes of different memory objects. Each shared memory allocation for a data mart is shown twice: once with the memory size in use, and once with the memory size theoretically still available for the data mart. If you add up the numbers of each pair, the result is always 15323136 KB. This number corresponds to the byte size of the files that are shown by running **ls -l**, which is run on the directory `/dev/shm`, for example:

```
... 15690891264 ... ibm_aqt_data...mart_225...
```

(15323136 KB * 1024 = 15690891264 B)

This value closely corresponds to the specification of the **WORKER_SHM** configuration parameter, which in the example is 15360 MB. Because all data marts must fit into the maximum size of shared memory that is specified by **WORKER_SHM**, the remaining size that is shown for individual data marts in the `smaps` pseudo file is theoretical only.

The amount of memory that is used corresponds to the size of the data marts that are shown in the output of the listMarts operation in Example 8-4 on page 173.

All of these complications originate from the Linux concept of memory over commitment, which means applications can allocate more memory than is physically available. The amount of physical memory that is available is the combination of random access memory (RAM) and configured swap space. The Linux operating system assumes that applications never use all their allocated memory. However, Informix Warehouse Accelerator can use of all its allocated shared memory if the data marts require all of the space that is specified by **WORKER_SHM**. Therefore, it is important to monitor memory usage and not overcommit memory on the Linux system.

Do not overcommit memory even though it might be allowed by the default setting in the UNIX kernel. Restrict the allocated memory within the available memory that is based on the RAM and swap space.

8.4.2 Monitoring processor usage

In the default configuration with one coordinator node and one worker node, the accelerator server starts three processes. These processes are coordinating the work (coordinator node), doing the work (worker node), and monitoring availability and restarting worker processes (watchdog). You can add more processes by changing the **NUM_NODES** configuration parameter. Because the coordinator and worker nodes are multithreaded, high processor usage is normal, even with a few processes. Unrelated operations that run on the same machine can also impact the processor usage.

If your Informix Warehouse Accelerator is on a single machine, you must monitor the processor usage as you add more worker nodes to your system. Do not focus only on data load improvements. In an accelerator server, you must balance the processor usage between query execution (scans, joins, and aggregations) and data load (dictionary building and data compression).

More considerations are required when you have the Informix server and Informix Warehouse Accelerator on the same system. You can offload some work to Informix Warehouse Accelerator, but the Informix server still might process a different workload, such as OLTP. The two servers compete for processor resources. If you separate the Informix server and Informix Warehouse Accelerator on to different systems, it is much easier to dedicate the correct amount of resources to each of them without interfering with each other.

If your system is overloaded, consider changing the single server topology to a cluster environment. If your Informix Warehouse Accelerator is already in a cluster environment, consider adding more Informix Warehouse Accelerator worker nodes for workload partitioning. For workload balancing, consider moving some accelerators and their data marts to a different Informix Warehouse Accelerator on a different system.

Depending on your Linux distribution and the packages that you have installed, you can use the **mpstat**, **top**, or **sar** commands to monitor processor usage.

8.4.3 Monitoring the network

If the Informix server and Informix Warehouse Accelerator are on separate systems, you must use a network connection to exchange data between them. When data marts are loaded, the data must be transferred from the database to the Informix Warehouse Accelerator. Queries are sent from the Informix server to Informix Warehouse Accelerator, and the result sets must be returned.

In an Informix Warehouse Accelerator environment, you must monitor the network traffic in both directions. For the network traffic from the Informix server to Informix Warehouse Accelerator, the usage primarily depends on the data mart size and how often they are loaded or refreshed. For the network traffic from Informix Warehouse Accelerator to the Informix server, the usage depends on the number of queries that are run and, more importantly, the size of the result sets. Normally, the network traffic in either direction is not high, but during data loads the network can become the bottleneck.

You can run **`netstat -s -t`** or **`ifconfig`** to check the statistics on the network interface that is used. You can monitor network latencies by using graphical tools from the operating system that hosts the Informix server or Informix Warehouse Accelerator.

8.4.4 Monitoring disk space usage

Informix Warehouse Accelerator keeps all data in memory. The data marts are written to disk in their compressed format only for recovery purposes. Only one working directory is used, which is specified by the configuration parameter **`DWADIR`**, and this directory does not grow significantly after you create all your data marts. Some data that is used for process communication is kept temporarily on disk and no backup is required.

8.5 Informix database server administration objectives

Accelerators and data marts are also maintained in the Informix server databases to manage the access and the query acceleration to Informix Warehouse Accelerator. You must include these Informix Warehouse Accelerator objects in your monitoring and administration strategy for the Informix server.

8.5.1 Accelerated query tables

Metadata on data mart definitions is stored as special views in the system catalog. These views are called Accelerated Query Tables (AQTs) and are listed as objects in the system catalog tables systables and sysviews of a database. The systables table defines the header information and identifies which accelerator and data mart an AQT belongs to. You can easily identify AQTs by their name. All data mart related AQTs in systables have a table name with the prefix “aqt”, followed by a unique cryptic identifier. The relationships between the data tables, used columns, and the join criteria of a data mart are stored as the view definition of the AQT in the corresponding entry in the sysviews table.

The Informix server uses AQTs to determine which queries can be accelerated by Informix Warehouse Accelerator and which data mart to use.

Investigating the definition of a data mart by using SQL

Here is an example of a query that retrieves details from AQTs about the definition of a particular data mart:

```
dbaccess -e <mart_database> << EOF
select tabname, site,dbname from systables where tabtype="V"
and tabname matches "aqt*"
EOF
```

Here is an example of the query output:

```
tabname  aqtf820d692-8d7a-4304-b7c7-27dee28b964a
site     accelerator_tpcds
dbname   mart_tpcds_warehouse

tabname  aqt50c27b5e-cfbe-42f0-a0eb-db0c52d94025
site     accelerator_tpcds
dbname   mart_tpcds_warehouse
```

The tabname is the name of the AQT, the site is the name of the accelerator, and the dbname is the name of the data mart. The AQTs are stored in the system catalog of the database for which the data mart was created.

You can also get data mart details by viewing the tables that are stored in the sysadmin database. These tables, named iwa_datamarts, iwa_marttables, iwa_martcolumns, and iwa_martpartitions, describe the data mart objects, their tables, and the columns. You can query these tables by using SQL statements to obtain more information. By using these tables, you are not required to look through different databases for the mart objects because all the information is available in one location.

Monitoring the AQT usage

You can use the **onstat** command on the Informix server to obtain statistics about which data marts and which AQTs are used in the running system. These statistics are available in various granularities. To get a general overview about the cached AQTs and their usage counts in the system, run **onstat -g aqt**, as shown in Example 8-6. The number of matches indicates how often a particular AQT was used to accelerate queries.

Example 8-6 Overview of the cached AQTs

```
onstat -g aqt
IBM Informix Dynamic Server Version 12.10.FC1 -- On-Line -- Up 20:10:13 --
AQT Dictionary Cache for database tpcds:
```

```
mart: tpcds_mart
accelerator: accelerator_tpcds
last load: 2013/06/13 21:37:11
AQT name                                FactTab #tab #matched address
------------------------------------------------------------------------------
aqt320e32e6-f1f5-449f-81ff-133a92f3fb6c     100    1      235 0x4d888650
aqt7dcd9a9c-3ed2-46a6-a544-3559ecad9d6b     103    2        7 0x4d8889b0
aqtef7c3e71-38d4-4b05-8cc4-89aebde8c042     100    2     1267 0x4d8887a8
aqt5428ac0e-25ff-40b6-8787-10ab811a0589     112    3    86168 0x4d888bb8
```

If you want to check a particular AQT, run **onstat -g aqt <name>**. In addition to the general output, you can see which tables are joined and which columns are used for the join.

Example 8-7 Showing one AQT information

```
$onstat -g aqt aqtfbc02804-d578-4219-ba9a-f8845d101499
IBM Informix Dynamic Server Version 12.10.FC1 -- On-Line -- Up 20:12:37 --
AQT: aqtfbc02804-d578-4219-ba9a-f8845d101499
database: tpcds
AQT tabid: 121
Fact table: 103
Number of times matched: 467
Join structure: alias(tabid)[colno,...] = alias(tabid)[colno,...] {u:unique}
0(103)[1] = 2(104)[1]
0(103)[1] = 1(105)[1]
```

The most critical information that is shown in the **onstat** output is the number of AQTs that are used on your system, which AQTs match the incoming queries, and the frequency. When you examine the matching statistics of the AQTs, use this information to answer the question: *Do the current data mart definitions reflect the current workload?* If the matching numbers are stable for a while but then decrease significantly, it can indicate that the applications were replaced, are no longer used, or that the SQL statement flow changed. If none of the AQTs of a specific data mart are matched anymore, it can indicate that the data mart is not used anymore and can be removed. However, consider that some data marts might be used only in certain intervals, for example, with monthly or quarterly reports.

8.5.2 Cleanup of data mart metadata

There are some minor cleanup operations on the Informix server that you can apply to your databases. The cleanup is required only in specific situations. This section describes which database objects to look for and how to clean them up.

Cleaning up orphaned AQTs

The AQTs for a data mart are stored in the systables and sysviews catalog tables in the Informix database for which the data mart was created. It is possible for the data mart to be dropped, but the AQTs remain in the system catalog. The remaining AQTs are orphaned and must be cleaned up. The main cause is that the drop mart operation was not run on the database for which the data mart was created. In this case, only the data mart itself is dropped, but not the AQTs. You must drop orphaned AQTs manually. For a script that you can use to drop orphaned AQTs and background information about the situation, its cause, and possible effects, see the Informix Warehouse Accelerator website, found at:

https://www.ibm.com/developerworks/community/blogs/2fa81a5c-cb30-4873-b775-1370151e3614/entry/informix_warehouse_accelerator_leftover_aqts_revisited3?lang=en

Cleaning up captured workload data

The captured workload is the result of the workload analysis that is run at the beginning of the data mart creation process. A captured workload is stored persistently in a table. The table contains the captured SQL statements and some internal information. Normally, the table is dropped after the data mart is created. Occasionally, only the workload analysis is done and the captured workload data remains in place in the database for which the workload was captured. In this rare case, you must drop the table manually. The name of the table is `dwa_saved_workloadtab_<workload name>`. In a smaller system, you can find this table by running **`oncheck -pe`**. In larger systems, you can find the remaining workload tables in the system catalog table systables.

Cleaning up query probing data

Probing data from queries is generated when a session issues the SQL statement **`SET ENVIRONMENT use_dwa 'probe start'`**. The probing data is stored only in memory, not persistently on disk. Therefore, the probing data disappears when the Informix server is restarted. You can manually clear the probing data by issuing the SQL statement **`SET ENVIRONMENT use_dwa 'probe cleanup'`**.

Cleaning up iwa_* tables

The Informix server built-in function **`ifx_probe2mart()`** transforms the internally generated probing data into a relational, internal data mart definition. This definition is used either to create an XML file that contains the data mart definition or to create the data mart itself. The relational, internal data mart definition is stored in tables in the specified logging database:

iwa_marts	Names of the data mart definitions
iwa_tables	All of the tables that are used in any data mart definition
iwa_columns	All columns that are used in any data mart definition

iwa_mtabs	Tables for a specific data mart definition
iwa_mcols	Columns for a specific data mart definition
iwa_mrefs	Join descriptors of a specific data mart definition
iwa_mrefcols	Join predicates of a specific data mart definition

These tables are not dropped after the data mart or the XML file is created. You can use these tables to add more probing data from other SQL statements to an existing relational, internal data mart definition by calling the **ifx_probe2mart()** procedure. After you are done creating your data marts in the accelerator, these tables are no longer required. You can drop these tables manually.

Tip: The iwa_* tables are created in the database that you are connected to when you run the **ifx_probe2mart()** function. The iwa_* tables can be in any logging database. These tables can be spread across several of your databases on the Informix server. Run the **ifx_probe2mart()** function calls on a single logging database so that all of the iwa_* tables are in one place. You can then occasionally run a cleanup procedure to clean either the entire contents or just the expired data mart definitions. To clean only the definition of a specific data mart from the tables, issue a statement similar to the following one:

```
DELETE FROM iwa_marts WHERE mart_name = 'your data mart name';
```

This statement cleans all the definition data from all the iwa_* tables for the specified data mart name by the cascading delete that is defined for the tables.

9

Use of IBM Cognos Business Intelligence with IBM Informix Warehouse Accelerator

This chapter gives a short introduction to IBM Cognos Business Intelligence and its potential benefits to your organization. The major part of the chapter describes, from the perspective of the Cognos user, how Cognos works with Informix Warehouse Accelerator and how specific tasks in Cognos can be accomplished to get the maximum performance improvement from using Informix Warehouse Accelerator.

9.1 IBM Cognos Business Intelligence

IBM Cognos Business Intelligence provides a unified workspace for business intelligence and analytics that the entire organization can use to answer key business questions and outperform the competition.

With IBM Cognos Business Intelligence, you can do the following tasks:

- View, assemble, and personalize information.
- Explore all types of information to assess the current business situation.
- Analyze facts and anticipate the tactical and strategic implications, with the ability to shift from views to more advanced predictive or what-if analysis.
- Collaborate to establish decision networks to share insights and drive toward a collective intelligence.
- Provide transparency and accountability to drive alignment and consensus.
- Communicate and coordinate tasks to engage the correct people at the correct time.
- Access information and take action from anywhere, taking advantage of mobile devices and real-time analytics.
- Integrate and link analytics from everyday work to business workflow and process.

Business intelligence applications frequently present data that is summarized from the details that are contained in a database. When you are using an application, you want to access data from different business perspectives and to quickly browse various levels of detail.

For the best performance, you can clean and structure data by using a collection of conformed dimensional star schemas. Depending on your business requirements, the granularity of your fact tables can be fine or rolled up by the processes that publish data to the star schemas. A key component of your database performance is how quickly it can select and summarize the sets of rows that are identified by users when they run ad hoc analysis or consume pre-authored reports.

You can review the models and reports of new or existing applications to ensure that the maximum amount of complete or significant derived tables in SQL statements are being routed to Informix Warehouse Accelerator for query acceleration. Many of these topics are applicable to both the current and older versions of the IBM Cognos Business Intelligence releases, which are supported by Informix through the E/SQL-C and Informix JDBC interfaces.

9.2 Metadata model

IBM Cognos Business Intelligence applications use a metadata model that presents databases by using business terms. A model contains objects or database query subjects, which map to Informix database tables and views. More layers of query subjects combine and transform database query subjects into business entities or model query subjects. These business concepts are further modeled into dimensions, hierarchies, and measures that support explorations, analysis, ad hoc queries, and reporting.

As you perform gestures in studios or run pre-authored reports, the IBM Cognos Business Intelligence query engine uses the report and model metadata to plan the SQL statements that are sent to the Informix database.

A frequently generated SQL statement represents the query subjects as derived tables. These derived tables are joined and filtered with other derived tables. Various levels of aggregation are applied to the intermediary derived tables in the statement by using traditional aggregates, such as sum, along with analytical aggregates, such as rank and cumulative totals.

You can review some of the design patterns that are used in the models and reports to ensure that Informix is able to route as many queries as possible to Informix Warehouse Accelerator.

9.3 Relationships

In a star schema model, applications frequently define a relationship between a dimension and fact table that requires an equi-inner join. A dimension can be related to multiple fact tables and can have more than one relationship or role to the same fact table.

Figure 9-1 shows two business subject areas, store sales and inventory, which are designed as two star schemas that share a conformed dimension (date).

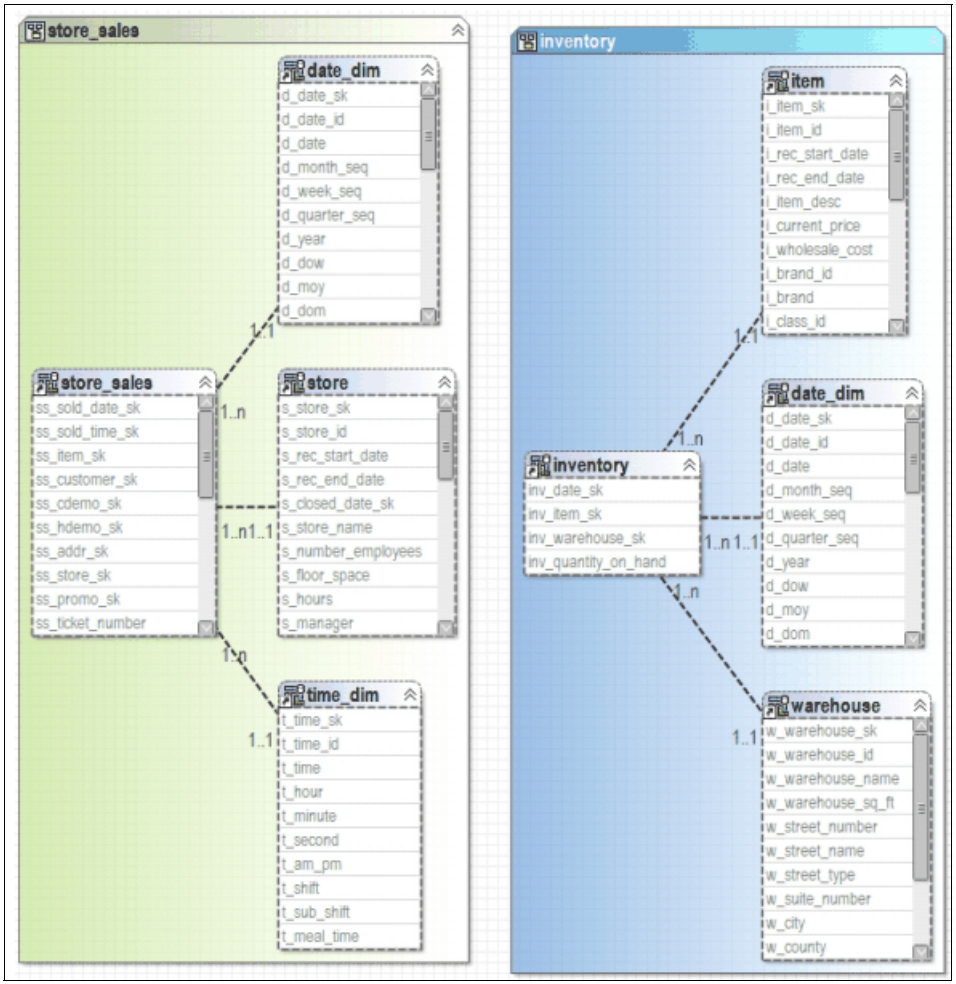


Figure 9-1 Conformed star schemas

In Figure 9-1, query subjects are defined with the relationship cardinality of one to many. Relationships with a cardinality of one-to-many or zero-to-many require SQL statements that use an outer join. A small result set is produced from queries that join one or more dimensions to a fact table with an equi-inner join. A small result set contains only those dimension members that have at least one associated fact table.

Relationships that are defined to preserve dimension members, but do not contain fact tables, can cause problems with queries that use Informix Warehouse Accelerator.

Figure 9-2 shows a model where the relationships for two dimensions are defined by using the zero-to-many cardinality on the fact table.

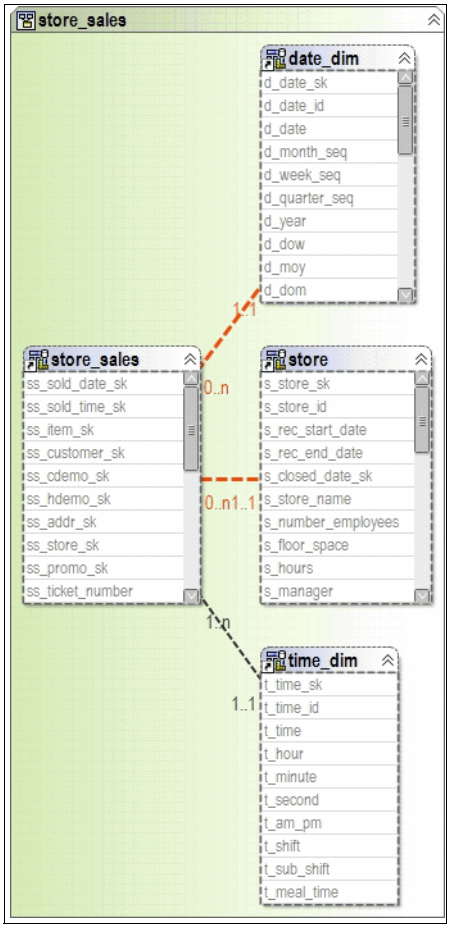


Figure 9-2 Optional relationships

At run time, business queries that use these dimensions can generate either one left outer join or a full outer join, if both dimensions are included. Figure 9-3 shows the SQL statement that is required to project members from the date and store dimensions, while preserving members without any store sales.

```
SELECT
    date_dim.d_date AS d_date,
    store.s_store_name AS s_store_name,
    SUM(store_sales.ss_quantity) AS ss_quantity
FROM
    TPCDS.tpcds.informix.date_dim date_dim
        LEFT OUTER JOIN TPCDS.tpcds.informix.store_sales store_sales
        ON date_dim.d_date_sk = store_sales.ss_sold_date_sk
            FULL OUTER JOIN TPCDS.tpcds.informix.store store
            ON store.s_store_sk = store_sales.ss_store_sk
GROUP BY
    date_dim.d_date,
    store.s_store_name
```

Figure 9-3 Full outer joins in a star query

SQL statements that are generated by the query engine use the SQL joined table syntax when outer joins are required. Queries that require only inner joins are expressed as predicates in the where clause of a SQL statement, or with joined table syntax if the model governor is configured to use explicit joins. Predicates in relationships appear in the join on clause when the explicit joined table syntax is used.

Figure 9-4 shows the package governors in a Framework Manager model, where explicit joins are requested for all join scenarios.

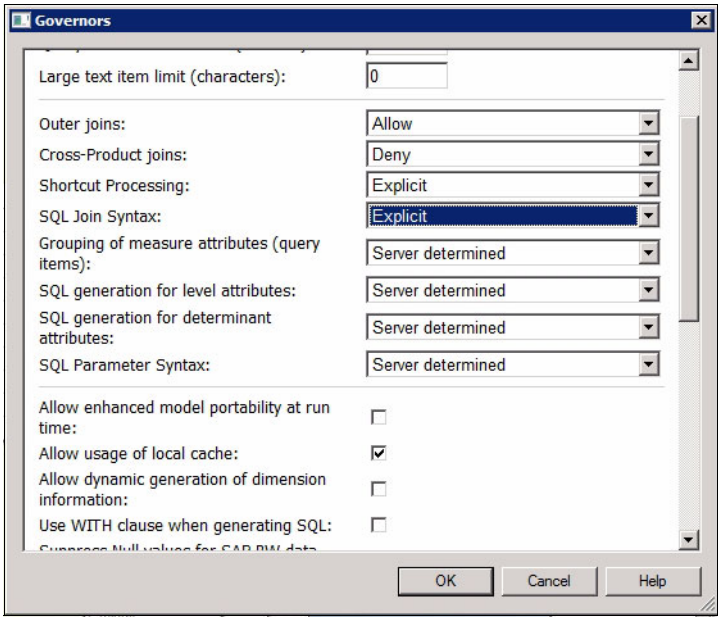


Figure 9-4 Join governor

9.3.1 Dimensions and member preservation

A model that defines dimensions, hierarchies, and measures over an Informix database, which is a dimensionally modeled relational database, can preserve members that do not have fact tables without requiring left outer joins in an SQL statement. The query engine uses separate streams, or queries, to retrieve members along with a query that retrieves the required facts.

Many common relative time business calculations, such as periodsToDate, parallelPeriod, lag, and lead, require members to be preserved to ensure the correct results. For example, an expression that computes the sales for the fourth quarter (Q4) versus the prior quarter. If there is no sales activity for Q3 and members are not preserved, an expression that references the prior quarter by one computes the value for Q2 and not Q3.

Figure 9-5 shows a dimension in a model where the OLAP compatible option is enabled to denote that dimension members are preserved for relative member calculations.

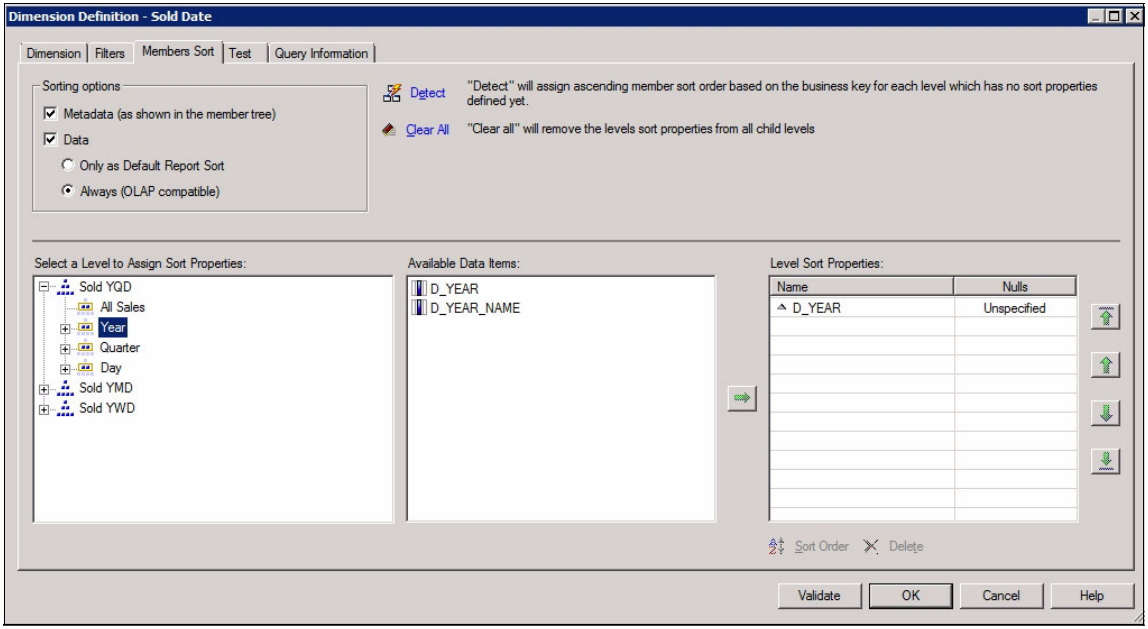


Figure 9-5 Relative members

To show query details within Report Studio, complete the following steps in Report Studio:

1. Select **Tools** and set Validate options to **information**.
2. Select **Tools** and validate the report.
3. Select and expand the information that is returned for the queries that are required by the layouts in the report.

Figure 9-6 shows information that is displayed within Report Studio for a report. The details include the SQL statements, which independently retrieve the dimension members and facts without an outer join.

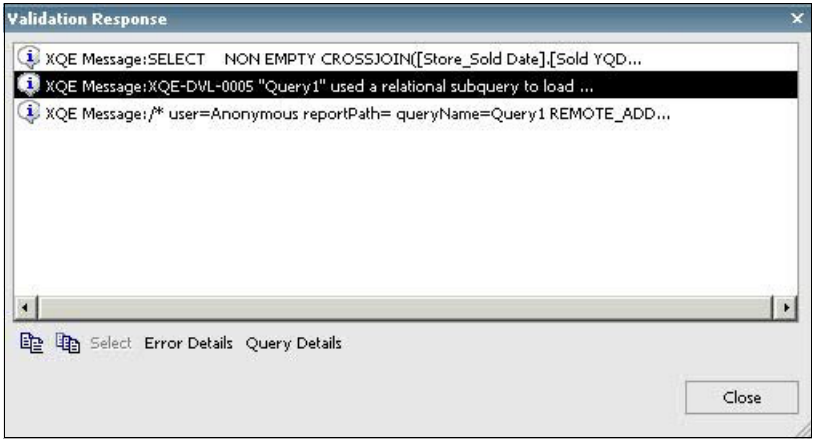


Figure 9-6 Information in Report Studio for a report

9.4 Multiple fact queries

A model normally contains two or more star schemas that can be concurrently accessed in an analysis. SQL statements are generated by using derived tables that reference the separate fact and dimension tables. These tables are later joined in the statement by using full outer joins. When the SQL statement uses a full outer join, Informix determines whether the derived tables can be sent to Informix Warehouse Accelerator, with the full outer join performed in the Informix server.

Figure 9-7 shows a query that combines measures from three fact tables that use derived tables (FS1, FS2, and FS3). The derived tables access each star schema and merge the results with full outer joins. Although the statement uses full outer joins, the inner derived tables can still be considered for acceleration by Informix Warehouse Accelerator.

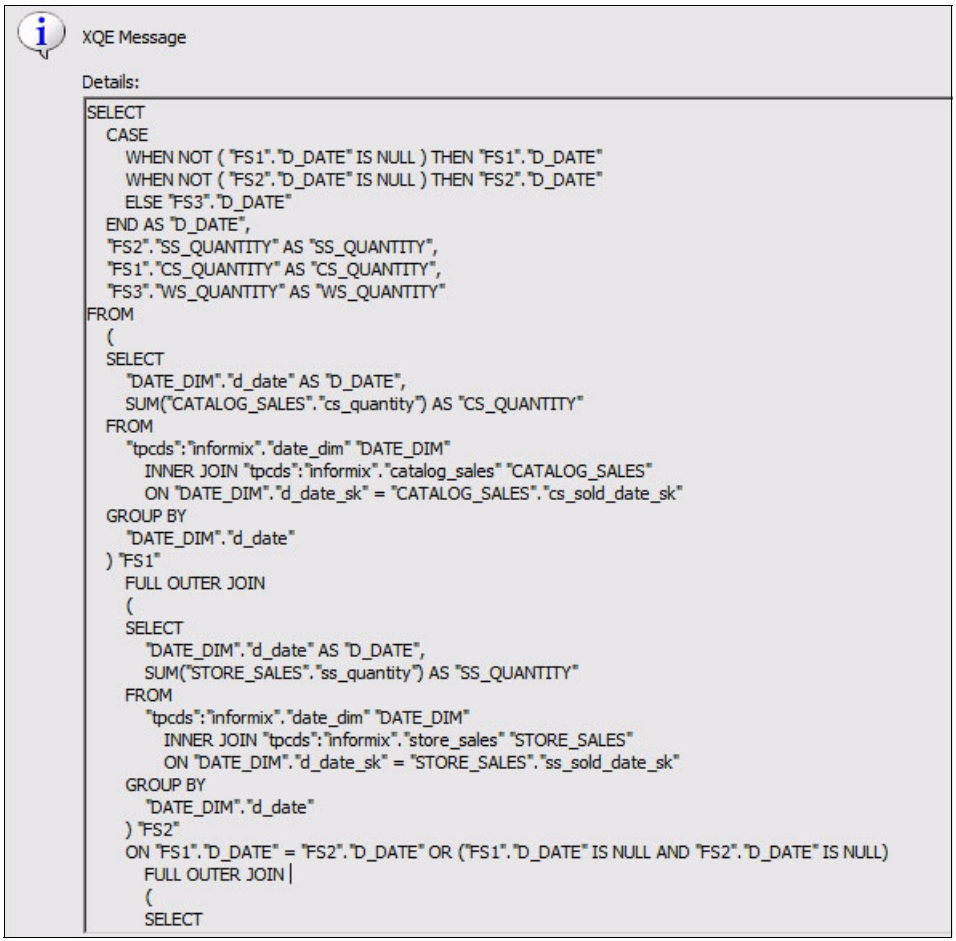


Figure 9-7 Multiple fact query

Another technique that a model or report can use combines multiple fact tables or slices of the same fact table by using a UNION set operator, as shown in Figure 9-8. Informix determines whether the derived tables on either side of the set operator can be sent to Informix Warehouse Accelerator with the union run at the Informix server.

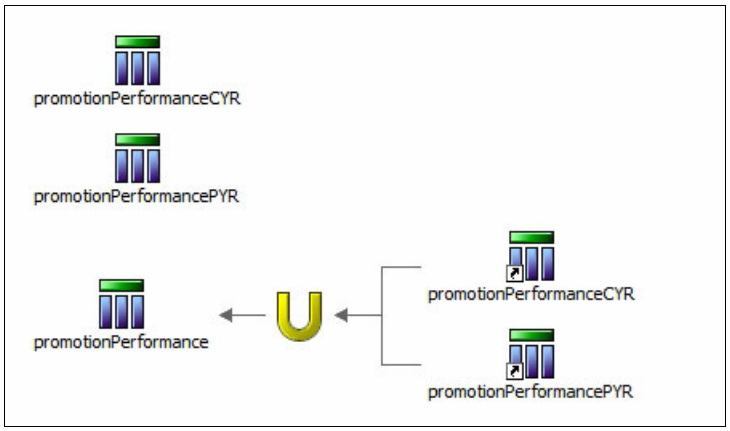


Figure 9-8 Report set operations

9.5 Filtering data

Query subjects can include filters (detail, security, or design time) that use expressions of varying complexity that reference columns of the query subject or other query subjects. These filters are shown as predicates in the where clause of a derived table or the SQL statement. Relationships between query subjects can be defined by simple column references or with complex expressions. If the SQL statement uses the join table syntax, these predicates can form part of the join on clause. Informix might not be able to accelerate some statements or derived tables because of the join predicate restrictions in Informix Warehouse Accelerator.

Figure 9-9 shows a relationship between the dimension and fact table that uses surrogate keys, which are extended to include expressions that reference the columns of both tables.

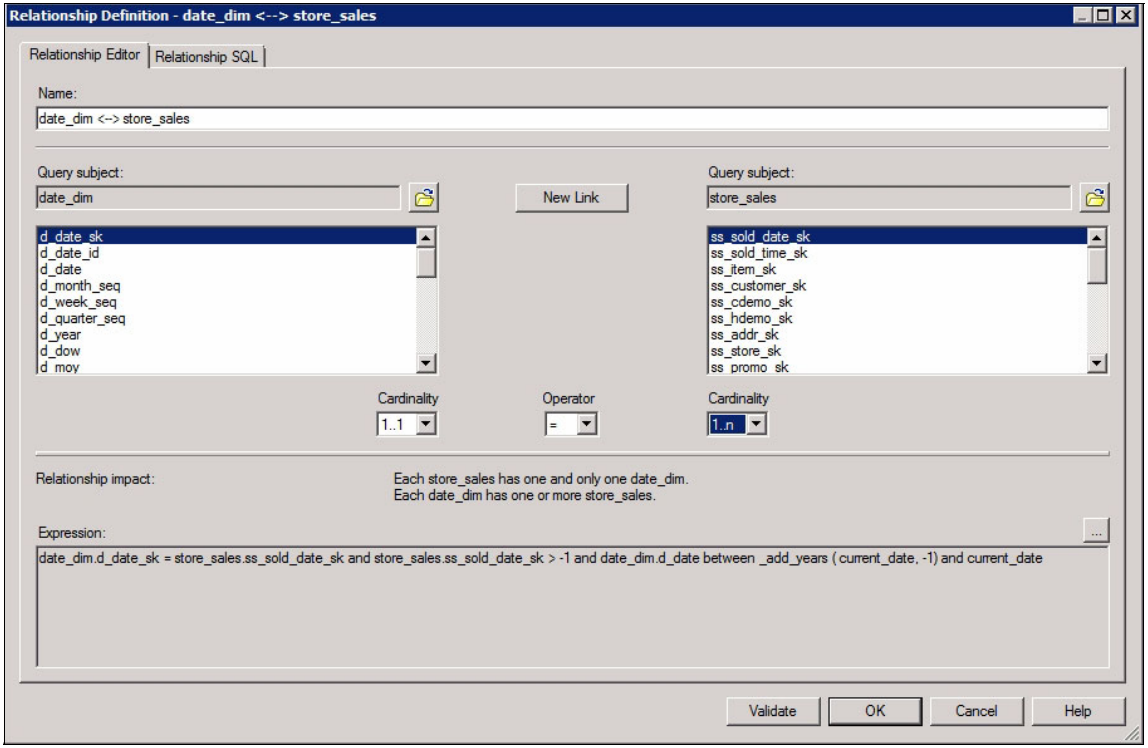


Figure 9-9 Expressions in relationships

9.6 Data types

IBM Cognos Business Intelligence supports the Informix numeric, character, and temporal data types. SQL statements that include column references or expressions of an interval or large object type cannot not be accelerated by Informix Warehouse Accelerator.

Figure 9-10 shows the model search facility within Framework Manager that can be used to locate items of specific data types.

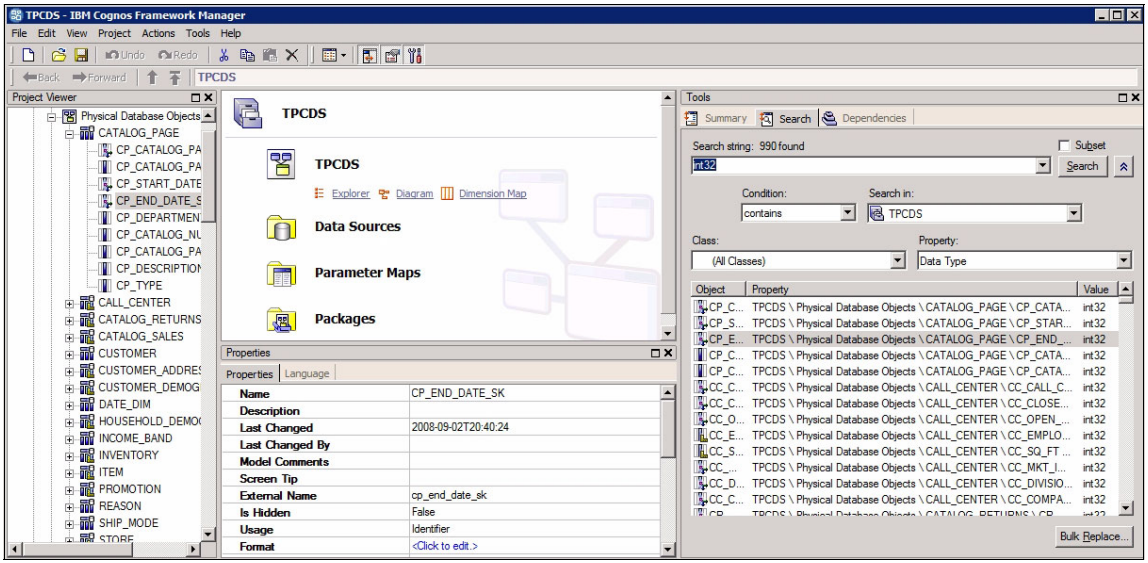


Figure 9-10 Model search

Expressions in a model or report that use interval data types can be changed to use the available Business Date, Time, or Macro functions. These functions enable you to perform common temporal operations, such as _add_days, _days_between, or _years_between by using integer values instead of intervals.

Figure 9-11 shows a filter expression that decrements a date by an integer amount and computes an integer number of days between dates.

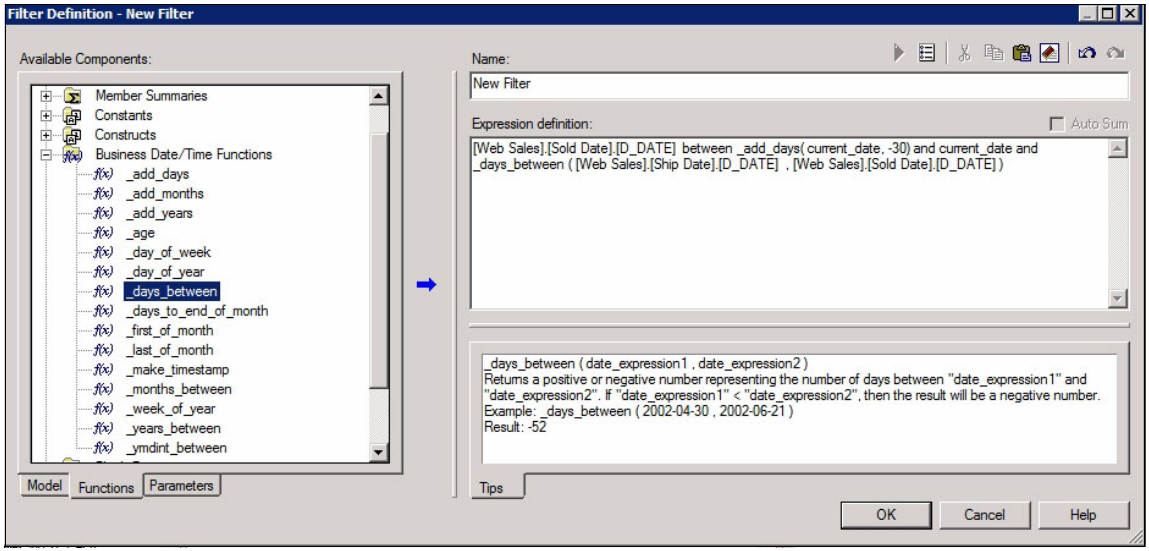


Figure 9-11 Business functions

9.7 Data driven prompts and dimension browses

Data driven prompts, metadata trees, and reports that reference only dimension tables might not be routed to Informix Warehouse Accelerator. The SQL statements that are generated by these scenarios frequently collapse duplicate values by using a distinct or group by clause and order the set by using an order by clause. You can improve the performance of these queries with the appropriate indexing in Informix. Large concurrent user workloads that use many data driven prompts can reduce the load on the Informix server by caching lists of values in the content store.

9.8 Connection command blocks

The IBM Cognos Business Intelligence portal allows you to define database connections to Informix database servers that are used by one or more applications. Optionally, a database connection can define a set of commands that are used to change the session state of a database connection.

To set the database connections to the Informix database server, complete the following steps:

1. Start the IBM Cognos Administration portal page.
2. Click the **Configuration** tab and click **Data Source Connections**.
3. Select the relevant connection, which shows a list of data sources.
4. Select the applicable Informix defined data source (using JDBC).
5. Click **Properties** for the appropriate connection.
6. Find and expand the Commands list to show the database events that can have an XML command block entered.

Figure 9-12 shows a connection command block that changes the environment USE_DWA to enable or disable the server that is attempting to route queries to Informix Warehouse Accelerator.

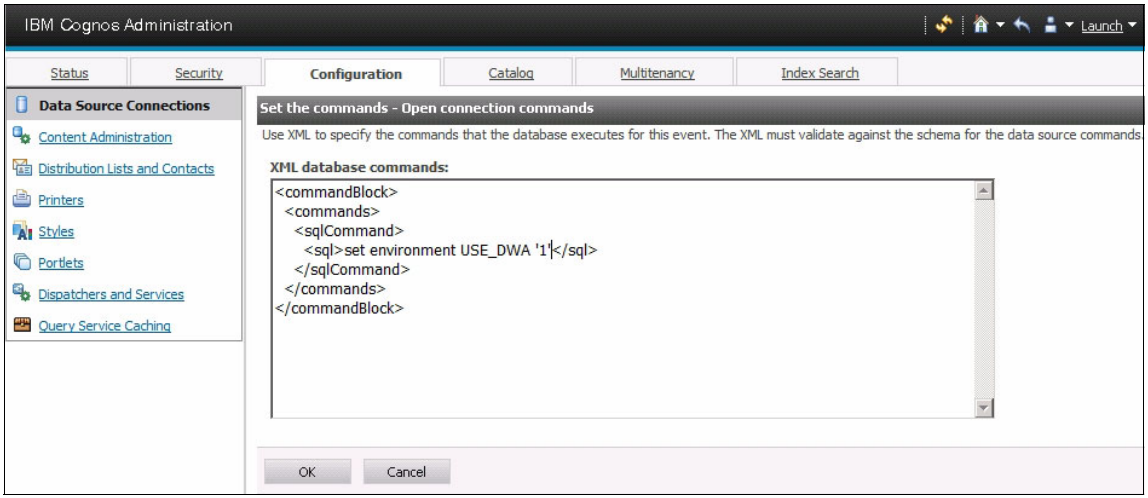


Figure 9-12 Connection command block

The body of a connection command block contains one or more database statements. Each statement can be a static string, as shown in Figure 9-12 on page 197, or dynamically generated by using various macro functions. If the statements do not change how the result sets are computed, or how SQL statements and the Dynamic Query engine is used, you can clear the governor option **Cache is sensitive to Connection Command blocks** (Figure 9-13) in a model. This option can minimize database connections that are concurrently opened with Informix.

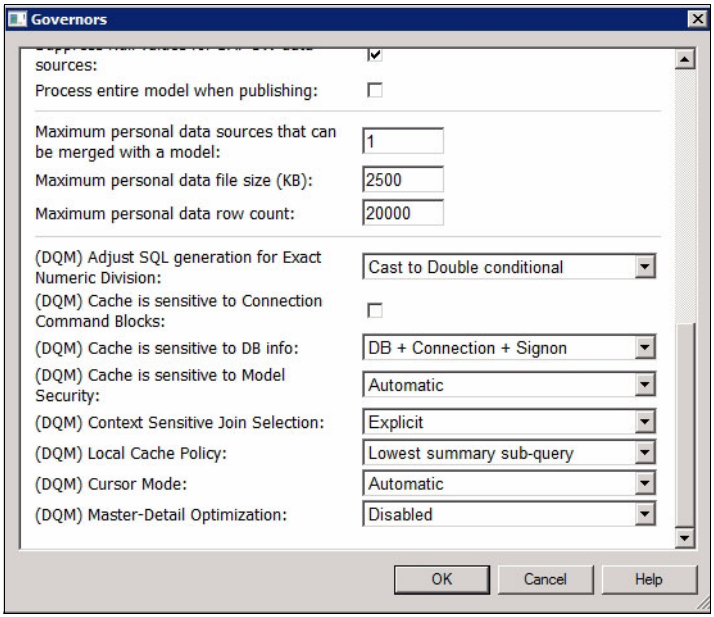


Figure 9-13 Cache manager and command blocks

9.9 Monitoring SQL statements

IBM Cognos Business Intelligence provides an audit facility that logs information about requests that it runs internally and any SQL statements that are run against a database. Optionally, you can add comments that include application context to dynamic SQL statements. You can use various macro functions to customize the information about the application artifacts, for example, package, report and query, and who is running it, for example, name and session ID.

Many applications run previously authored analysis and reports that use user-defined names. However, ad hoc analysis and query gestures are assigned system generated names for reports and queries.

IBM Cognos Business Intelligence applications and the Informix administrators can use both of these facilities to monitor workload that is sent to Informix and, as applicable, Informix Warehouse Accelerator.

The SQL in Figure 9-14 shows a comment string that includes the following items:

- The authenticated user, or anonymous when no authentication is used.
- The location and name of the report that was run.
- The name of the business query in the report that was run.
- An internal request identifier that can be used to link to the audit database data.

```
/* user=Anonymous reportPath= queryName=channelPerformance requestID=sd9vj2829Cqh9qv9hsy8984yMI2jdIlG9j24IMs4 */
SELECT
  CASE
    WHEN NOT ( "FS1"."DATE0" IS NULL ) THEN "FS1"."DATE0"
    WHEN NOT ( "FS2"."DATE0" IS NULL ) THEN "FS2"."DATE0"
    WHEN NOT ( "FS4"."DATE0" IS NULL ) THEN "FS4"."DATE0"
    WHEN NOT ( "FS3"."DATE0" IS NULL ) THEN "FS3"."DATE0"
    WHEN NOT ( "FS6"."DATE0" IS NULL ) THEN "FS6"."DATE0"
    ELSE "FS5"."DATE0"
  END AS "DATE0",
  "FS4"."SS_QUANTITY" AS "SS_QUANTITY",
  "FS3"."SR_RETURN_QUANTITY" AS "SR_RETURN_QUANTITY",
  "FS2"."CS_QUANTITY" AS "CS_QUANTITY",
  "FS1"."CR_RETURN_QUANTITY" AS "CR_RETURN_QUANTITY",
  "FS6"."WS_QUANTITY" AS "WS_QUANTITY",
  "FS5"."WR_RETURN_QUANTITY" AS "WR_RETURN_QUANTITY"
FROM
  (
  SELECT
    "FS6_inner"."DATE0" AS "DATE0",
    "FS6_inner"."WS_QUANTITY" AS "WS_QUANTITY",
    SUM(1)
      OVER(
```

Figure 9-14 Comments in SQL

To enable or disable the logging of comments by the Dynamic Query service in SQL statements, complete the following steps:

1. Start the IBM Cognos Administration portal page (Figure 9-15).
2. Click the **Configuration** tab and click **Dispatchers and Services**.
3. Select the **Query Service** and select the properties action.
4. From the Settings tab, select the logging category and change the **Generate comments in native SQL** value.

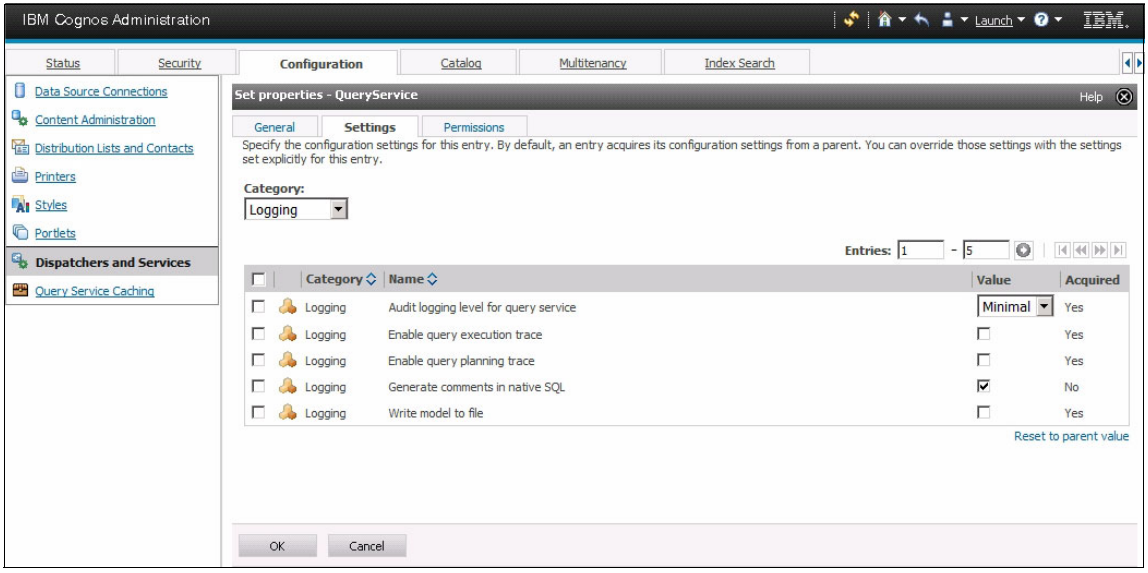


Figure 9-15 Query server configuration

10

IBM Informix Warehouse Accelerator proof of concept

This chapter describes how to conduct a successful proof of concept (POC) for IBM Informix Warehouse Accelerator. This chapter includes information about how to identify queries for acceleration, set goals for a successful POC, and determine the operating system, hardware, and environment. This chapter also describes how to configure the environment, process the queries, and document the results.

10.1 Assessing qualification and compatibility for Informix Warehouse Accelerator

It is important to understand the existing behavior of queries that run on an IBM Informix database server, including the query types, and the workload that they represent, such as OLTP, OLAP, or warehousing. You can gather information about query processing times to help you identify the candidate queries for Informix Warehouse Accelerator. Because you can integrate Informix Warehouse Accelerator with IBM Informix server, you might consider an architecture that combines OLTP and warehousing workloads into a single IBM Informix server environment.

Typically, queries that benefit from the use of Informix Warehouse Accelerator are the following ones:

- Aggregate queries
- Business intelligence queries that involve multi-table joins
- Queries that involve large volumes of data
- Queries that analyze data patterns that are not easily identified with smaller data sets
- Queries that allow data analysis to gather intelligence about the data, patterns, and trends
- Queries that forecast future trends by using data that was captured in the past

Often, the result sets that are returned from these queries are not large. They are summarized in multiple levels.

Capturing and analyzing the current Informix server workload helps you understand the queries and make informed decisions about whether queries can be accelerated:

- Capture query explain plans.
- Isolate queries that contain a high cost for the query.
- Review high-cost queries. Are they caused by poor update statistics? Whether you use Informix Warehouse Accelerator, good update statistics or not, information is key to an efficient Informix database server environment.

Typically, organizations conduct a proof of concept before they make decisions about whether to purchase new hardware or software technology. A successful proof of concept can provide your organization with confidence in your investment. For example, proof of concept results can be used to validate to management why they should invest in new technology.

With internal tests and POCs performed in the field, IBM has seen highly successful results with Informix Warehouse Accelerator. The query processing speed is often orders of magnitude faster. With Informix Warehouse Accelerator, you can reduce the time that it takes to run long queries on the Informix database server. The return on investment of Informix Warehouse Accelerator is easily shown by performing a POC.

Informix Warehouse Accelerator alleviates many of the common concerns of a data warehouse environment:

- Poor query response times
- Inability to scale to large data volumes
- Unsuited for near real-time or on-demand (ad hoc) workloads
- Inability to run mixed workloads
- Minimal in-memory capabilities
- Cannot support advanced analytics

A proof of concept exercise gives your organization the opportunity to validate the performance benefits of Informix Warehouse Accelerator before you make an investment.

In the world of big data, data is analyzed much differently than how it was in the past. Legacy data warehouses of the past processed queries over hours and days; now queries are almost real time. With minimal investment in hardware and software, you can gain a leading edge by making the database environments adapt to new technology. Best of all, implementing a solution that uses Informix Warehouse Accelerator does not require changes to existing applications. Existing applications can continue to attach and connect to an Informix database server and submit the queries to the Informix database server. The Informix database server routes the queries to the accelerator and returns the results in orders of magnitude faster.

10.2 Assessing customer pain for analytic queries

For a proof of concept, you must understand the current environment and the nature of queries:

- Do the aggregation queries populate the resulting data into summary tables?
- Are queries being run to generate cubes?
- Are online analytical processing query types used (OLAP, ROLAP, and MOLAP)?

Make sure that you understand how critical these types of queries are and the time constraints for these queries. Generally, the amount of time it takes to process these queries is the biggest obstacle.

After you identify the queries, you can make informed decisions about whether a particular query is a good fit for Informix Warehouse Accelerator. Typical Informix Warehouse Accelerator queries contain fact and dimension tables. You can make all columns of the fact and dimension tables available to Informix Warehouse Accelerator, or you can make only relevant columns of the fact and dimension tables available to Informix Warehouse Accelerator, which allows for processing of specific queries. If you are using the IBM Smart Analytics Optimizer Studio, you can create the data mart in a way that involves all the columns of the fact and dimension tables. If you are using workload analysis, by probing the Informix server, only the specific columns that are involved in the query are made available to the Informix Warehouse Accelerator.

When using Informix Warehouse Accelerator, you are limited by the amount of memory that is available and the number of processors that are available for coordinator and worker threads. Keep this in mind when you are deciding whether to make a table and all of its columns available to the data mart on Informix Warehouse Accelerator. By having only the required columns from the fact tables, you can best use the available memory and create a comprehensive data mart that can address various queries.

From the perspective of a proof of concept, you must first identify a set of queries to run on the Informix server to obtain a baseline of processing times. These queries should be representative of a production workload. Then, run this same set of queries by using the Informix Warehouse Accelerator and compare the query processing times against the baseline. By doing so, you can decide whether Informix Warehouse Accelerator sped up the queries.

For the queries that you select to run on the Informix Warehouse Accelerator proof of concept, there are no limits on how simple or how complex the queries can be. The more complex the queries are, the better suited they are for processing by using Informix Warehouse Accelerator.

10.2.1 Feasibility assessment

To decide whether Informix Warehouse Accelerator is suited for your organization, you must understand some query characteristics.

Typically, data warehousing environments use the star schema database design. The star schema design is the simplest form of a data mart schema. It has one or more fact tables and any number of dimension tables. The fact and dimension tables are related through primary and foreign key definitions. The fact tables contain the detail-level transactional data and the dimension tables contain the attributes of the transactional data. Informix Warehouse Accelerator typically performs well in these types of database design implementations.

You must consider how the data warehouse gets its data. How often is new data added to the database? What mechanisms are in place, such as tools to extract, transform, and load data into the data warehouse? For example, a nightly batch process can extract data from front-end OLTP databases by using a customized application program, or a constant trickle feed of data from the OLTP databases that occurs at periodic intervals. In some cases, data is delivered to the organization and then parsed by using a batch process.

When you determine the workload in the data warehouse environment, you must identify the complex long running queries. Not all queries can be classified as long running queries. How do you decide that a particular query is a long running query? For example, is a query that runs for one minute classified as a long running query, or is a query that runs for 10 minutes classified as a long running query? To make these determinations, you must understand how complex certain queries are and how long they take to run.

Are many queries run to generate aggregate and summary level data? In data warehousing environments, this practice is common. For example, when transactional data that comes from an OLTP environment is loaded into the data warehouse, queries are run to summarize and aggregate results, and then these results are stored in other tables. This process allows you to start data analysis from a high level and eventually drill down to the detailed data. In some cases, a mid-level summary might satisfy what a business analyst is looking for. With Informix Warehouse Accelerator, summary tables and tables that store aggregates are no longer required. The in-memory transactional data is processed so rapidly by Informix Warehouse Accelerator that it can be summarized or aggregated at run time.

Consider the amount of time that it takes to build the summary and aggregate tables. These tables also require space allocation in the Informix database server environment. If the amount of data is large and the data retention requirements are for more than a year, the summary tables eventually use a significant amount of storage space and might use data partitioning. Data partitioning requires additional planning and management. With Informix Warehouse Accelerator, these tasks are no longer a concern.

The data warehousing environment might provide a set of known reports, referred as “canned” reports. The queries are pre-determined and coded to produce a report with specific aspects of the data. These reports often contain highly optimized SQL queries that are created by developers and DBA teams to improve the query execution time.

However, the “canned” reports might not satisfy everyone. For example, business analysts might want to run ad hoc queries on the data. This ad hoc capability is provided by using business intelligence software tools such as Cognos, Business Objects, Pentaho, and Yellowfin. These tools provide access to the underlying schema objects within a data warehouse, and allow you to select what data elements to use for data analysis. The business intelligence tools can provide a rich graphical interface that includes a dashboard, graph, and charts.

Satisfying ad hoc queries can be a difficult task for a DBA. A query might not be known before execution, and therefore might not be optimized for the database engine. You might have to provide the optimizer with hints to make the queries run faster. These types of queries can take a significant amount of time to run and produce results. Because all of the scanning occurs in-memory when using Informix Warehouse Accelerator, these types of ad hoc queries are much faster.

When they run ad hoc queries with Informix Warehouse Accelerator, DBA teams are not required to specifically tune the database server, build indexes, or optimize the queries. In most data warehousing environments, a large amount of data can be used for business analysis and business intelligence types of queries. Without leading-edge technology, these types of queries can consume many processor cycles and much memory. Query processing times can last anywhere from minutes, to hours and days. With the Informix Warehouse Accelerator, the limiting factors are hardware resources, such as the number of processors available, and the amount of available memory.

IBM can help customers and Business Partners conduct a proof of concept to determine whether Informix Warehouse Accelerator is a good fit. IBM subject matter experts (SMEs) can help configure the hardware and software, and identify queries that benefit from using Informix Warehouse Accelerator. To get help with your proof of concept, contact your IBM sales representative.

10.2.2 Business value assessment

IBM offers a business value assessment (BVA) and return on investment (ROI) service. Typically, this assessment involves an IBM technical team member who works with a technical team member from your organization to conduct the assessment. This assessment normally takes about two to three weeks to complete, with about 10 days of work. During that time, interviews are conducted on site or through teleconference with key stakeholders. After the information is gathered and analyzed, IBM develops a strategy and shares it with your organization. The BVA and three-year ROI assessment can help get buy-in from key stakeholders. To get help with the BVA and ROI assessment, contact your IBM sales representative.

The BVA objectives include the following ones:

- Provide a business case that can be taken to upper management to justify the investment.
- Demonstrate the business value.
- Demonstrate the impact and benefits of these capabilities versus doing nothing.
- Provide a strategy that can easily scale as your business grows.

The BVA approach includes the following items:

- Conduct tests with several customers.
- Conduct interviews with key personnel and stakeholders.
- Understand the current environment.
- Identify business goals.
- Define current issues and problems.
- Define ROI and the payback period.

10.3 Document of understanding

The document of understanding (DOU) briefly explains the nature and the specifics and the goals for the proof of concept. The DOU document describes the objectives and goals of the proof of concept that are agreed to by IBM and the customer representatives. This document outlines the effort, time, and material that is required. It also defines the success criteria for the proof of concept. The DOU clearly defines the expectations from all of the parties that are involved and can help your organization get the necessary approvals to purchase hardware and software for a production deployment.

10.4 Operating system, environment, and client application readiness

The basic hardware requirement for the Informix Warehouse Accelerator is a server that is based on an Intel 64-bit architecture with Linux as the operating system. The Informix database server can be on any of five supported 64-bit platforms, which include a Linux and Intel 64-bit architecture. The Informix Warehouse Accelerator hardware environment can be stand-alone or the same Linux Intel 64-bit system along with the Informix database server.

If you plan to set up the Informix database server and Informix Warehouse Accelerator on the same server, you must consider the available resources on this server. Ensure that adding new functions, such as the Informix Warehouse Accelerator, does not slow down the existing database server function and capability.

If the Informix server is installed on other chip architectures such as IBM System p®, HP-Itanium, Sun, or any other supported chip architecture, Informix Warehouse Accelerator must have a server that is configured on Linux Intel 64-bit. The number of processors and amount of memory are important factors when considering the suitability of this type of hardware configuration.

Because the Informix Warehouse Accelerator uses the processor and memory components, separate storage is not required. The storage requirements are used by the operating system (Linux) and by the image of the compressed data that is stored in memory. All of the defined data marts are stored in compressed format for recovery purposes. Typically, the amount of storage that is required depends on the size of the data mart that is deployed. For example, for a 1 TB data mart, the amount of storage that is required is also 1 TB. The operating system also consumes some storage space.

You are encouraged to acquire your own hardware after the planning phase of the proof of concept. This hardware is then used in a production environment after you successfully complete the proof of concept. The hardware costs are minimal because Informix Warehouse Accelerator uses commodity hardware such as blade servers.

IBM can provide loaner hardware for the proof of concept. This hardware might take some time to procure and might cause a delay in completing the proof of concept. If you are interested in pursuing this path, contact your IBM sales representative.

Alternatively, you can also use the IBM Innovation Centers to test the proposed solution at dedicated IBM facilities. These centers are meant for conducting proofs of concept. You can work with IBM sales representatives to decide which IBM Innovation Center to use and get access to an environment. However, to use this option, you must have an Informix database server that contains all of your data. Depending on the size of the data warehouse, this can be a complicated task. For example, special care might be needed to secure the data before, during, and after the proof of concept. Although this is not a problem for IBM, you might face several challenges in your organization to get the proof of concept accomplished at an IBM facility.

Doing a proof of concept within your own organization is less complicated and can provide greater data security. The process is simple: add another piece of hardware to the existing data center, establish network connectivity, and configure the system to communicate with an existing Informix database server.

10.5 Preparing the server for Informix Warehouse Accelerator

After the hardware is in place and connected to the network, you must verify that the operating system kernel parameter settings are correct, including the shared memory settings, swap parameters, and name resolution. The operating system must also support large pages.

To install Informix Warehouse Accelerator software, complete the following steps:

1. Install Informix Warehouse Accelerator. The Informix Warehouse Accelerator software is installed by using the standard IBM Informix Java-based installation package. You can use command-line options to install Informix Warehouse Accelerator with default values for the configuration file. This configuration file is named `dwainst.conf` and is in `$INFORMIXDIR/dwa/etc/`.
2. Install the IBM Smart Analytics Optimizer Studio. The IBM Smart Analytics Optimizer Studio is available for the Microsoft Windows operating system and for the Linux 32-bit architecture. The IBM Smart Analytics Optimizer Studio must be installed on its own supported operating system. For novice users of Informix Warehouse Accelerator, the IBM Smart Analytics Optimizer Studio is a great tool to configure the Informix Warehouse Accelerator, define the database schema, define the data mart, and initiate the data movement from the Informix database server to the Informix Warehouse Accelerator.

The configuration file contains information about:

- The storage location of the image of the compressed data mart
- The number of worker and coordinator nodes, and the memory that is available for each worker and coordinator node
- Network protocol information
- In a clustered Informix Warehouse Accelerator setup, the configuration file contains cluster information.

10.6 Sizing the environment

Although there is no tool or application that can determine the amount of memory that is required for the Informix Warehouse Accelerator, it is not difficult to come up with a size on your own. To determine the amount of memory that is required, analyze the data, data types, and the amount of data to be stored in the Informix Warehouse Accelerator, and the data retention period. Because the Informix Warehouse Accelerator does not have indexes, there are no memory allocation requirements for the indexes. Because there are no update statistics, you are not required to allocate memory or storage for histograms for the optimizer. The only memory consideration is the raw data. Because the Informix Warehouse Accelerator uses technology such as deep columnar compression and frequency partitioning, the amount of memory that is required is drastically reduced. Typically, a minimum 1:3 compression ratio is possible, if not greater.

When you are calculating the size of a system, this environment can be used in a production environment. During the proof of concept phase, all of the available hardware and memory resources might not be used because the scope of the proof of concept is limited. However, when you implement a solution in a production environment, use of the hardware and memory to the fullest possible extent.

10.6.1 Informix database server requirements

Informix Advanced Workgroup Edition or Informix Advanced Enterprise Edition are required to use Informix Warehouse Accelerator. In most cases, there are not many Informix database server-side requirements.

There are Informix database server configuration-related steps that must be completed during the installation and configuration of Informix Warehouse Accelerator. You can use the **ondwachk** command to check the Informix server environment and identify settings that need to be adjusted. Then, you can set up the trusted connection between Informix Warehouse Accelerator and the Informix database server by using the **ondwa getpin** command. After this trusted communication is activated, the `sqlhosts` file in the Informix database server is modified to include the relevant entries.

The Informix database server is used during the query probing process. Query probing captures the queries for acceleration. To capture probing data, the SQL tracing function is used. SQL tracing is available only on the Informix database server side.

10.7 Defining an accelerator and creating a data mart

Here are the available methods for defining an accelerator and creating a data mart:

- IBM Smart Analytics Optimizer Studio
- Informix Open Admin Tool (OAT)
- SQL stored procedures API to the Informix database server

The IBM Smart Analytics Optimizer Studio is an Eclipse-based administration interface that can be used by novice users to perform administration tasks for Informix Warehouse Accelerator. You can create and define the project and the data mart. The data mart consists of tables with explicit primary and foreign key relationships defined. The data mart is a subset of the database schema as defined in the Informix database server.

The IBM Smart Analytics Optimizer Studio user interface provides a simple way to define the accelerator and data mart, and load the data from the Informix database server to the Informix Warehouse Accelerator.

For advanced or expert users, most of these same tasks are also available by using the SQL-based functions of the stored procedures API. You can enable SQL tracing and query probing. After query probing, gather the probe results and create an XML file from the probe data. This XML file is the data mart definition and contains the tables and columns that are indicative of the query workload that the Informix database server normally encounters. You can use the XML file, along with the **ifx_createMart()** function of the stored procedure API, to create the data mart in the Informix Warehouse Accelerator. With the functions of the stored procedure API, you can script certain management aspects of Informix Warehouse Accelerator. For example, you can automatically refresh the data mart nightly by creating a task in the sysadmin database.

10.7.1 Probing queries

The probing function is useful in specific scenarios. If you have an environment where it is difficult to identify queries that can benefit from query acceleration, you can use the probing feature to identify these queries. To use the probing feature, you must turn on both the SQL tracing and probing capabilities. After the SQL trace and the probing capability are on, each query that is submitted is evaluated by the Informix database engine for its ability to be accelerated. You can run this activity during peak or heavy workloads to capture long running and complex queries. Consider any batch processing situations that occur during the SQL tracing window. After a certain period or when all of the queries are captured, you can turn off SQL tracing and stop the probing. Now, you can use the probe data to define the data mart.

If you build a data mart from the probe data, only the columns that are directly involved in a query are part of the data mart. These are columns from the projection list and the predicates of the query. Building a data mart from probe data decreases the data mart size. For example, if a table contains 32 columns but only 12 columns are part of a particular query, when you create the data mart with the probe data, only the 12 columns are part of the data mart. However, in situations where a query involves any of the omitted columns from the table, the Informix Warehouse Accelerator cannot accelerate the query. The query processing falls back to the Informix database server, if the fallback option is turned on.

When you create a data mart by using probing, you can identify queries that perform poorly or take long processing times. Most of these queries are I/O bound and depend on the storage infrastructure to produce results. Most data warehousing environments are I/O bound and take a long time to process the queries. This is a typical use case scenario for Informix Warehouse Accelerator. You can use OAT to analyze the tracing output to isolate slow I/O bound queries.

After you create a data mart, certain catalog information is gathered and stored in the system catalog tables on the Informix database server. These system catalog tables are called Accelerated Query Tables (AQTs). The Informix database server uses AQTs for query matching and routing operations. After a query is matched to the AQTs, it is then routed to and processed by the Informix Warehouse Accelerator.

Queries or query blocks that meet this criteria are sent to Informix Warehouse Accelerator:

- The query or query block must refer to a subset of the tables in the AQT.
- The table references or joins that are specified in the query or query block must be equijoins and match the references in the data mart definition.
- The query or query block must include only one fact table.
- The query or query block must have an INNER JOIN or LEFT JOIN with the fact table on the left-dominant side.
- The scalar and aggregate functions in the query or query block must be supported by Informix Warehouse Accelerator.

10.8 Accelerating SQL queries

After you create the data mart, you can prepare the queries. Typically, for a proof of concept, there is a set of queries that are identified as candidate queries. You can begin with a simple query, for example, a query against a single table. Then, you can focus on the successful acceleration of the query. You can gather query processing times at a later stage.

You can focus on making sure that all candidate queries are processed error free by the Informix Warehouse Accelerator. If the query returns just a few rows, you must verify that the results are identical to the queries running on the Informix database server. There can be serious problems if the query results do not match, negating the success criteria of the proof of concept.

10.8.1 Controlling query acceleration

You can use the **`set environment use_dwa`** SQL statement to control whether the query is sent to the Informix Warehouse Accelerator or run locally by the Informix database server. You can use the fall back option to ensure that the query is processed by the Informix database server if it cannot be processed by the Informix Warehouse Accelerator.

Figure 10-1 shows the options that are available for the **set environment use_dwa** SQL statement.

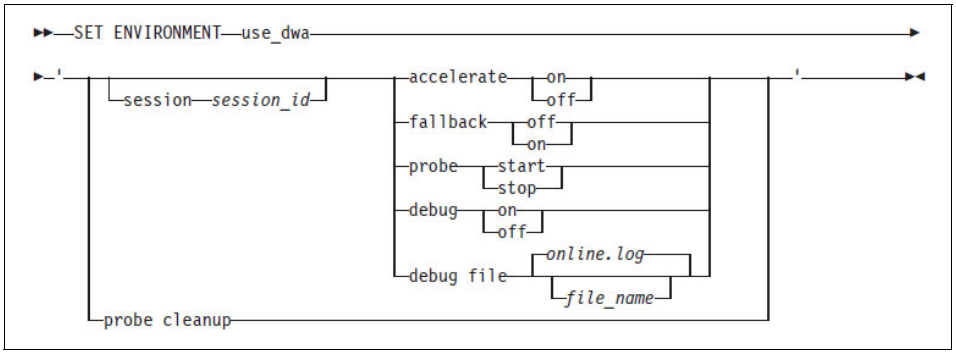


Figure 10-1 Set environment use_dwa command syntax

For the purpose and meaning of the various options, see the Informix information center that is found at the following website:

http://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/ids_acc_056.htm

You can turn on the debug option with the **set environment use_dwa 'debug on'** SQL statement to log messages to the Informix database server online log file. With this option, you can check that all of the queries are processed by the Informix Warehouse Accelerator. Optionally, you can send these debug messages to a separate log file.

It is important to review the log file contents to ensure that there are no error messages and that all of the queries are processed by the Informix Warehouse Accelerator.

10.8.2 Query performance

The proof of concept environment is now fully ready and functional. You can run every query in the baseline query set, one after another, and capture the query processing times. The proof of concept environment must be a controlled environment.

You can record the query processing times for later analysis. You can expect to see orders of magnitude performance improvements. Rerun the set of queries at least 5 - 10 different iterations, recording the query processing time for each iteration. The repetition is required to ensure that results are consistent for each query. The speed of your query results might be hard to believe, but that is the power of Informix Warehouse Accelerator. You must thoroughly verify the results, comparing the accelerated results to those achieved when there is no query acceleration.

10.8.3 Compiling results

To establish the performance improvements, you can compare the timings of the queries on the Informix database server versus when they are run on the Informix Warehouse Accelerator. You can prepare a document that summarizes the environment for the Informix database server, Informix Warehouse Accelerator, and provides the timings in a tabular format with the query performance improvement factor. This summary document can help management understand the value proposition of this proof of concept.

Informix Warehouse Accelerator performance depends primarily on two crucial components: processor power and memory performance. You must configure an environment with enough processor power and memory to get the maximum performance for SQL queries. The administrative tasks are minimal, without the need for query tuning, storage management, partitioning of data, and indexing of data. You must have a well-tuned operating system kernel and enough storage space to save a copy of the compressed memory image.

If the Informix Warehouse Accelerator proof of concept environment was sized for production use, it is easy to implement it as a production solution. Most of the initial work, such as setting up the infrastructure and configuring Informix Warehouse Accelerator, was completed as part of the proof of concept. You might have to drop and rebuild the data mart to support the production requirements.

10.9 Proof of technology

A proof of technology is an exercise that shows a feature function that is based on a pre-determined set of data in a constrained environment, such as a notebook or desktop computer. The IBM Technical Sales team or the Competitive Technical Enablement team can do either an onsite visit or conduct a web conference to discuss Informix Warehouse Accelerator. They can show a demonstration of the Informix Warehouse Accelerator capability. These demonstrations are built on a virtual image, running on a desktop or a notebook. With the constrained resources such as memory and processor, the sales team can show the product and its function, and the ease of use. The Informix Warehouse Accelerator virtual image is a custom virtual image that is built by Informix engineers on top of the IBM Informix Virtual Appliance image. The IBM Informix Virtual Appliance image is distributed by IBM and uses SUSE Linux 11 and Informix 12.10FC1 in a 64-bit architecture.

The virtual image contains a TPC-DS sample database that contains approximately 20 GB of data. The engagement involves a presentation that describes the core technology of the Informix Warehouse Accelerator and a demonstration of the Informix Warehouse Accelerator. In the demonstration, some queries are run against the sample TPC-DS database. Queries are not run on the Informix database server because it takes too long. Internal tests show that query processing times for some of the queries on the Informix database server take 35 - 90 minutes. However, the same queries when run on the Informix Warehouse Accelerator take 1 - 16 seconds. These results are in a constrained setup in a virtual image.

If you want to implement a solution that uses Informix Warehouse Accelerator, a proof of concept exercise is where to begin. This type of engagement allows you to use your own Informix database server, schema, and data. You can run your application queries or even your ad hoc queries and truly experience the power of Informix Warehouse Accelerator.

A

Tools for IBM Informix Warehouse Accelerator

This appendix provides an overview of the tools and interfaces that are used with Informix Warehouse Accelerator.

A.1 Quick guide to Informix Warehouse Accelerator tools and interfaces

With various tools and interfaces that are available to accomplish the same tasks, choosing the correct tool can be confusing. Table A-1 provides an overview of the appropriate tools and interfaces for each task.

Table A-1 Tool and interface usage summary

Task / Tool	OAT	SP API	ondwa	ondwachk	IBM Smart Analytics Optimizer Studio
Set up, start, stop, reset, and clean the accelerator server.	4	4	1	4	4
Get pairing code to connect Informix server with the accelerator server.	4	4	1	4	4
Monitor the accelerator server (status and tasks).	4	3	1	4	4
Check the Informix server configuration for the Informix Warehouse Accelerator connection.	4	4	4	1	4
Create or remove the accelerator.	1	1	4	4	3
Create a data mart with an existing XML definition file.	4	1	4	4	3
Create a data mart by using workload analysis.	1	2	4	4	4
Create a data mart by using interactive design.	4	4	4	4	1
Administer a data mart by embedding or scripting.	4	1	4	4	4
Administer a data mart interactively.	1	4	4	4	3
Refresh data.	1	1	4	4	4
Refresh TimeSeries data.	4	1	4	4	4

Table key:

1 Best

2 Good

3 Applicable

4 Not applicable

A.2 The ondwa utility

You can use the **ondwa** utility to set up and work with the accelerator server. The **ondwa** utility is delivered and installed with the IBM Informix Warehouse Accelerator binary files on the Linux 64-bit server. The **ondwa** utility provides the same control mechanism for the basic administration tasks of an accelerator server. The **ondwa** utility is versatile and supports a cluster or an SMP environment, and multiple worker nodes. The **ondwa** utility does not operate on individual objects within an accelerator server. An *accelerator server* is composed of the physically running Informix Warehouse Accelerator node processes in the Linux operating system. An *accelerator* is a logical entity for a connection from an Informix server and contains data marts.

Figure A-1 shows an example scenario that explains the difference between an accelerator server and accelerator, and how they are related to an Informix server instance. Two separate Informix server instances are connected to the same accelerator server, but each Informix server instance has its own accelerator, “dwa1” and “dwa2”. Each Informix instance has access only to its own data marts within its own accelerator, which are controlled by the entry and authorization tokens in their `sqlhosts` files. Each database contains some meta information, for example, AQTs, about the data mart that was created for the database.

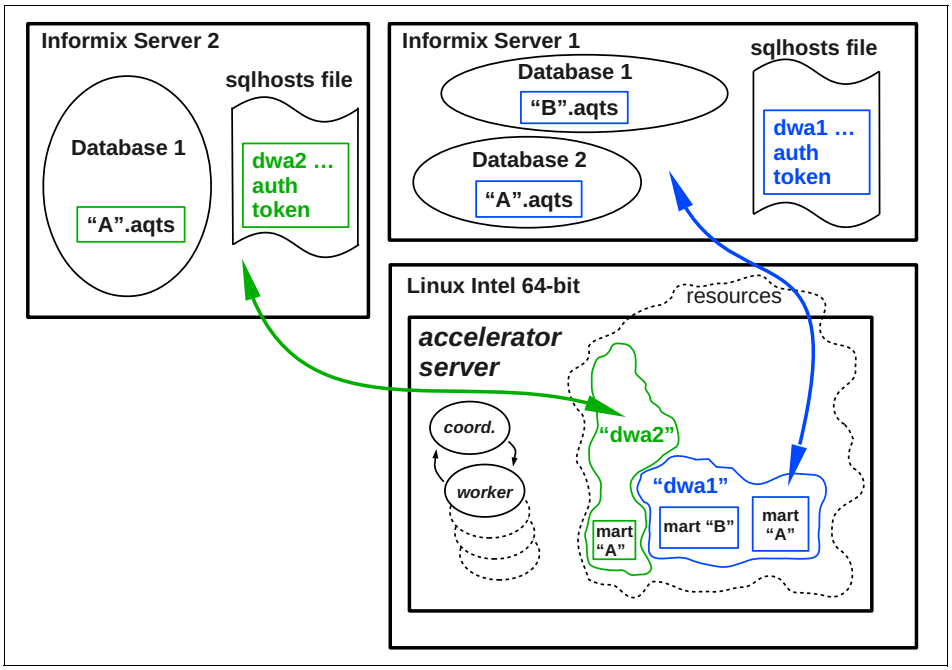


Figure A-1 Relationship between the accelerator server, Informix server, and accelerators

The Informix Warehouse Accelerator tasks that are performed with the **ondwa** utility correspond to the tasks of the Informix database system administrator. Normally, only the informix or root users can run the **ondwa** utility.

The tasks for the **ondwa** utility are specified by a single command-line parameter:

-setup Sets up an environment for an accelerator server and create a configuration that is based on the parameters that are specified in the `dwainst.conf` configuration file.

-reset Puts the accelerator server into the state where it was immediately after running the initial **ondwa -setup** command.

-clean Removes all files that are generated by the accelerator server. To use Informix Warehouse Accelerator again, you must run the **ondwa -setup** command.

-start Starts the accelerator server.

-stop Stops the accelerator server. Optionally, you can specify an extra **-f** command-line parameter to forcefully shut down the accelerator server.

-status Gets a summary on the status of the accelerator server, with one line for each Informix Warehouse Accelerator node.

-tasks Gets the status of the different tasks that are ongoing.

-getpin Gets the pairing information. You use this information when you create the initial connection between an Informix server and the accelerator server.

For more information about usage and precautions, see the section "*T*he ondwa utility" in the *IBM Informix Warehouse Accelerator Administration Guide*, found at:

https://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/ids_acc_601.htm

A.3 The ondwachk utility

The **ondwachk** utility is installed with the Informix server and is in the $INFORMIXDIR/bin directory. You can use the **ondwachk** utility to confirm that the Informix database server environment is set up correctly for Informix Warehouse Accelerator and that the database server can connect to the accelerator server. To verify the Informix database server environment, run the **ondwachk** command as the informix user in the Informix server environment. The output of the command describes which checks were run successfully, which checks were skipped because of a specific configuration, and which checks were not successful. Furthermore, it might provide a warning for checks that it did not run and how to check these errors manually.

Here is an example of the output from the utility:

```
WARNING: The check for compatible versions is not done.
The Informix server is version 12.10.FC2.
The accelerator server version is unknown.
Check the version for the accelerator server by running the command
"ondwa -V" in its respective environment.
PASSED: An sbspace is configured for the Informix server.
```

```
PASSED: The SBSPACENAME parameter is set to the name of the default
sbspace in
the ONCONFIG file.
PASSED: A network connection type of sockets with TCP/IP protocol is
defined in
the sqlhosts file.
SKIPPED: This is a standard server and the Scheduler is running. The
check for
a dwavp virtual processor is skipped.
SKIPPED: This is a standard server. The checks for a secondary server
configuration are skipped.
PASSED: The user informix has write access to the sqlhosts file and the
directory that the file is in.
PASSED: An entry for an accelerator is found in the sqlhosts file.
```

A.4 Stored procedure API

Except for the tasks that are described in A.2, “The ondwa utility” on page 219, all administration of the Informix Warehouse Accelerator data marts is done by functions that are implemented within the Informix server. These functions are accessible through the stored procedure (SP) API. As the name suggests, this API provides a set of stored procedures that can be used by any database client.

These three stored procedures of the API are used to establish or remove an initial connection, or to gather monitoring information about the accelerator server:

- **`ifx_setupDWA()`** initiates a connection from an Informix server to an accelerator server and creates an accelerator for the Informix server instance.
- **`ifx_removeDWA()`** removes an existing accelerator from the `sqlhosts` file of the Informix server instance. This procedure does not remove data marts from Informix Warehouse Accelerator or metadata on those data marts from the databases.
- **`ifx_getdwaMetrics()`** retrieves statistics about the accelerator server.

The rest of the stored procedures manage data marts or information that is related to data marts. These stored procedures are divided into groups, according to their tasks:

- Stored procedures are used to create or drop a data mart, and to change the status of an existing data mart:
  - **`ifx_createMart() type 1`** creates a data mart in an accelerator, according to the specified XML definition.

- **ifx_createMart() type 2** creates a data mart from an internal, relational representation of the data mart definition.
- **ifx_dropMart()** removes the specified data mart from an accelerator.
- **ifx_setMart()** activates or deactivates an existing data mart.

- Stored procedures that work with data in an existing data mart:
  - **ifx_loadMart()** does a full load of the complete data into a data mart. This procedure is required for the initial data load after the data mart is created. You can run this procedure regularly for the lifetime of a data mart.
  - **ifx_loadPartMart()** and **ifx_dropPartMart()** load or drop a specific data partition of a specific table. You must specify the name of the table or the name or partition number of the fragment to be dropped or loaded.
  - **ifx_refreshMart()** determines which data partitions are changed in the database, including fragments that were newly dropped or attached. **ifx_refreshMart()** uses this collected information to run the necessary load or drop for each partition.
  - **ifx_setupTrickleFeed()** and **ifx_removeTrickleFeed()** start or stop feeding a continuous trickle of newly inserted data rows to the fact tables of a data mart. To enhance performance, this process is done in short user-specified time intervals. You can also refresh dimension tables based on the changes in their partitions.
  - Stored procedures that manage TimeSeries Virtual Table Interface (VTI) tables and refresh their data:
    - **ifx_TSDW_createWindow()** creates a time window to limit the amount of TimeSeries data in a TimeSeries virtual table of a data mart. This data is loaded into the data mart.
    - **ifx_TSDW_dropWindow()** removes time windows from a TimeSeries virtual table in a data mart. The data is removed from the data mart.
    - **ifx_TSDW_moveWindow()** moves time windows in a TimeSeries virtual table within a data mart. The old data is dropped and the new data is loaded.
    - **ifx_TSDW_setCalendar()** assigns a TimeSeries calendar to a TimeSeries virtual table in a data mart.
    - **ifx_TSDW_updatePartition()** refreshes the TimeSeries data within a TimeSeries virtual table in a data mart.

- Stored procedures to gather information about existing data marts:
  - **ifx_listMarts()** returns a set of unnamed rows that contain the status for all the data marts in an accelerator, regardless of which database they were created for. The status information for each data mart includes its size and the most recent load or refresh time.
  - **ifx_getMartStat()** returns the status of a single data mart as an unnamed row.
  - **ifx_getMartdef()** retrieves from Informix Warehouse Accelerator the complete definition for a specific data mart in XML format.
- Stored procedures that manage data mart definition information. A corresponding data mart might or might not exist:
  - **ifx_probe2Mart()** converts the data that is gathered from probing into an internal, relational data mart definition.
  - **ifx_genMartDef()** returns a character large object (CLOB) that contains the data mart definition in XML format.
  - **ifx_xml2Mart()** converts a data mart definition from XML format into an internal, relational data mart definition. This procedure is basically the reverse **ifx_genMartDef()**.
  - **ifx_def2Mart()** uses the **ifx_getMartdef()** function to retrieve the data mart definition from an accelerator. It then uses the **ifx_xml2Mart()** procedure to convert the data mart definition from the XML format to an internal, relational data mart definition.

The concept for data mart definition information is modeled on the different stages where this information is used.

To create a data mart by using the workload analysis method, as described in Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69, complete the following steps:

1. Gather information about your query workload.

   In the query probing phase, the data on the probed queries is collected internally by the Informix server and stored in memory. You can view this probing data by running **onstat -g probe**. This probing data is not written to disk. Each probing action adds new data on the newly probed queries. However, you can explicitly clear the probing data by running **set environment use_dwa 'probe cleanup'** or by shutting down and restarting the Informix server.

2. Build a data mart definition from the probing data.

   You can build a data mart definition by running the **ifx_probe2Mart()** procedure. This stored procedure uses the database and data mart name as parameters and uses the current connection to a logging database. It analyses the probing data, builds an internal data mart definition, and stores it under the data mart name in the logging database. Although the data mart definition is permanent on disk, it is still in an internal, relational format, which is stored in a set of tables. If these tables do not yet exist in the logging database, **ifx_probe2Mart()** creates them. If new query probing data is collected, you can run the **ifx_probe2Mart()** procedure again with the same parameters to add the data to the existing relational data mart definition.

3. Create an XML file from the internal, relational data mart definition.

   You can create an XML file from the internal, relational data mart definition by running the **ifx_genMartDef()** procedure. If there is an existing relational data mart definition in the tables of the logging database, this step can be repeated. However, if there is an existing XML file, it is either overwritten or it might cause an error. An existing XML file cannot be augmented with more information. The XML file contains the minimum definition of the data mart, consisting of the tables and their columns, the relationships between the tables and their columns, and which tables are fact tables.

4. Create a data mart that is based on the definition in the XML file.

   You can create a data mart that is based on the definition in the XML file by using the **ifx_createMart() type 1** stored procedure. This stored procedure augments the XML data mart definition with information about the data types of the table columns, and so on. This augmented XML definition is then sent to Informix Warehouse Accelerator for the data mart creation. The Informix Warehouse Accelerator internally stores this data mart definition, whereas the Informix server does not store the XML definition. Instead, the Informix server stores only relevant metadata. You can retrieve the XML data mart definition from Informix Warehouse Accelerator by using the **ifx_getMartdef()** stored procedure. A data mart definition that is retrieved by **ifx_getMartdef()** contains more information, such as the table column data types.

The **ifx_createMart() type 2** stored procedure provides a shortcut for data mart creation. As input parameters, this procedure uses the name of the data mart to be created and the name of the logging database that contains the internal, relational data mart definition. The **ifx_createMart() type 2** stored procedure uses the definition that was previously created when **ifx_probe2mart()** was run. When **ifx_createMart() type 2** is run, the **ifx_genMartDef()** procedure is run internally, and then **ifx_createMart() type 1** is called internally. With this shortcut, you can avoid the requirement to create the intermediate XML data mart definition.

The **ifx_xml2Mart()** stored procedure takes an XML data mart definition as input and converts it to the internal, relational format, stripping present table column data type information. It then stores this information in the tables of the logging database. If a data mart definition exists with the data mart name, then this data mart definition is augmented with the information from the input XML definition.

The **ifx_def2Mart()** stored procedure provides a shortcut. Internally, **ifx_def2Mart()** runs the **ifx_getMartdef()** stored procedure to retrieve the XML definition of an existing data mart from Informix Warehouse Accelerator. Then, it internally calls the **ifx_xml2Mart()** stored procedure to store the definition as an internal, relational definition in the tables of the logging database.

Figure A-2 shows the different stages of a data mart definition and the stored procedures to pass the definition between them.

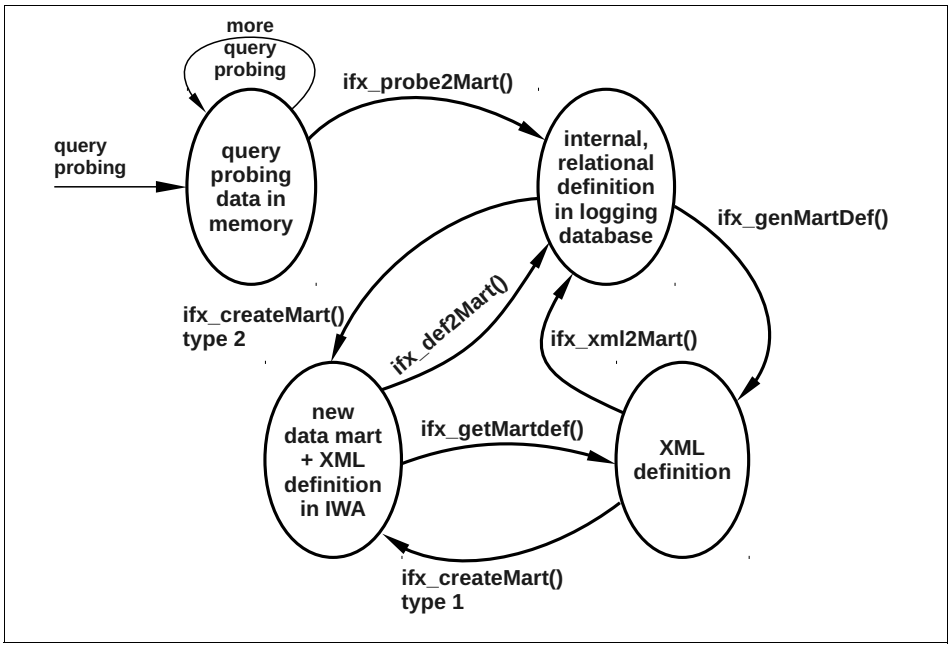


Figure A-2 Stages of a data mart definition and corresponding stored procedures

For more information, see the “SQL administration routines” section in the *IBM Informix Warehouse Accelerator Administration Guide*, found at:

https://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/ids_acc_073.htm

A.5 IBM OpenAdmin Tool for Informix (OAT)

The OpenAdmin Tool is a web application for administering and analyzing the performance of IBM Informix database servers. OAT is used in a web browser and connects to multiple Informix server instances by using the PHP-protocol. Although OAT is available as open source, the main development activities for OAT are done at IBM. With Version 3.11, released in March 2013, OAT provided the capabilities to administer and monitor Informix Warehouse Accelerator data marts. These capabilities are delivered as a plug-in, which is the common way to realize different features and extensions for OAT.

Data mart administration with OAT is done by connecting to an Informix server instance. If the Informix server does not have an accelerator yet, you can create an accelerator within OAT by setting up the initial connection from the Informix server to Informix Warehouse Accelerator. To create an accelerator, you must have the connection information and the pairing code. You can get the pairing code by running the **`ondwa getpin`** command. After the accelerator is created, you can monitor its status by using OAT.

Data mart operations are run from the Schema Manager in OAT. Here, existing data marts are seen as objects in the database, and you can create data marts by using workload analysis, as described in Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69. You can use OAT to activate, deactivate, and drop data marts as well. Furthermore, data load and data refresh is possible. OAT provides tools to easily schedule data mart loads and refreshes.

As of the writing of this book, OAT Version 3.11 does not support TimeSeries data in data marts.

For more information about OAT, see the following resources:

- The official web page is at `http://www.openadmintool.org`. From this page, click **Demonstrations** and then **IWA** to see an introduction video about how to administer data marts with OAT. Alternatively, you can see the same video on the OAT YouTube channel at the following website:

  `http://www.youtube.com/user/OpenAdminToolChannel`

- *Creating data marts with the IBM OpenAdmin Tool (OAT) for Informix*, found at:

  `http://www.ibm.com/developerworks/data/library/techarticle/dm-1304datamart/`

A.6 IBM Smart Analytics Optimizer Studio

The IBM Smart Analytics Optimizer Studio is an Eclipse-based graphical interface for administering data marts in Informix Warehouse Accelerator. It is included with the Informix Ultimate Warehouse Edition and must be installed either on a Windows system or on a 32-bit Linux system. Like OAT, Smart Analytics Optimizer Studio must be connected to an Informix server for data mart administration tasks. Some data mart operations, such as data refresh, trickle feed, and TimeSeries data, are not supported in Smart Analytics Optimizer Studio. However, the unique feature of Smart Analytics Optimizer Studio is the graphical interface for designing a data mart interactively. This interface offers drag operations for assembling a data mart with tables and connecting them by using specific table columns. For more information, see Chapter 5, “Creating IBM Informix Warehouse Accelerator data marts” on page 69.

To learn more about IBM Smart Analytics Optimizer Studio, see these two video tutorials on the IBM developerWorks website:

- *Exploring the accelerator interface* is an introduction to IBM Smart Analytics Optimizer Studio:

  http://www.ibm.com/developerworks/offers/lp/demos/summary/im-iwaexploreinterface.html

- *Creating a data mart* also shows how to include subsets of table columns in a data mart:

  http://www.ibm.com/developerworks/offers/lp/demos/summary/im-iwacreatedatamart.html

For a description about how to use IBM Smart Analytics Optimizer Studio, see the *Informix Warehouse Accelerator Administration Guide*, found at:

https://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/ids_isao1.htm

For Informix Warehouse Accelerator administration tasks besides interactive design of data marts, use OAT instead of Smart Analytics Optimizer Studio.

Related publications

The publications that are listed in this section are considered suitable for a more detailed discussion of the topics that are covered in this book.

Other publications

These publications are also relevant as further information sources:

- Date, C. J. *An Introduction to Database Systems* (8th ed.), Addison-Wesley Longman, 1999, ISBN 0321197844
- *IBM Informix Warehouse Accelerator Administration Guide*
- Kent, W., *A Simple Guide to Five Normal Forms in Relational Database Theory*, Communications of the ACM, vol. 26, pp. 120–125, 1983
- Kimball, Ralph, et al, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd edition,* Wiley, 2013, ISBN 1118530802
- O'Brien, J. A., et al, *Management information systems* (9th ed.), McGraw-Hill/Irwin, 2009, ISBN 0073376817

Online resources

These websites are also relevant as further information sources:

- *Creating a data mart*, found at:

  http://www.ibm.com/developerworks/offers/lp/demos/summary/im-iwacreatedatamart.html
- *Creating data marts with the IBM OpenAdmin Tool (OAT) for Informix*, found at:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1304datamart/
- *Exploring the accelerator interface*, found at:

  http://www.ibm.com/developerworks/offers/lp/demos/summary/im-iwaexploreinterface.html

- IBM OpenAdmin Tool (OAT):

  http://www.openadmintool.org
  http://www.youtube.com/user/OpenAdminToolChannel

- *Informix Warehouse Accelerator*, found at:

  https://www.ibm.com/developerworks/community/blogs/2fa81a5c-cb30-4873-b775-1370151e3614/entry/informix_warehouse_accelerator_leftover_aqts_revisited3?lang=en

- *IBM Informix Warehouse Accelerator Administration Guide*, found at:

  https://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.acc.doc/acc.htm

- *IBM Informix Warehouse Accelerator: Performance is everything*, found at:

  http://public.dhe.ibm.com/common/ssi/ecm/en/imw14587usen/IMW14587USEN.PDF

- Informix information center:

  http://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.welcome.doc/welcome.htm

- Publications for IBM Informix Warehouse Accelerator 12.10:

  http://www-01.ibm.com/support/docview.wss?uid=swg27037872

- *SYSDBOPEN: A flexible way to change session behavior in Informix,* found at:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1109sysdbopen/

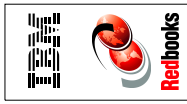
Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

Query Acceleration for Business Using IBM Informix Warehouse Accelerator

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Query Acceleration for Business Using IBM Informix Warehouse Accelerator

Technology and architecture of Informix Warehouse Accelerator

Data mart design considerations and examples

IBM Cognos Business Intelligence integration

IBM Informix Warehouse Accelerator is a state-of-the-art in-memory database that uses affordable innovations in memory and processor technology and trends in novel ways to boost query performance. It is a disruptive technology that changes how organizations provide analytics to its operational and historical data. Informix Warehouse Accelerator uses columnar, in-memory approach to accelerate even the most complex warehouse and operational queries without application changes or tuning.

This IBM Redbooks publication provides a comprehensive look at the technology and architecture behind the system. It contains information about the tools, data synchronization, and query processing capabilities of Informix Warehouse Accelerator, and provides steps to implement data analysis by using Informix Warehouse Accelerator within an organization.

This book is intended for IBM Business Partners and clients who are looking for low-cost solutions to boost data warehouse query performance.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8150-00 ISBN 0738438928