IBM Information Management Software
Technical white paper
Updated:March 2013

IBM Informix Warehouse Accelerator

Performance is everything

Keshava Murthy, Senior Technical Staff Member, IBM Informix Development

Introduction

Imagine analyzing your store sales and returns data quickly to create actionable intelligence. Imagine Business analysts and C-level executives running interactive dashboards and reports in seconds instead of hours. Imagine these reports are based on the latest business transactions instead of yesterday's data. Improving business execution speed with data points available within seconds; Imagine analyzing terabytes of smart sensor data in seconds; Imagine improving analytical query performance without constantly monitoring and tuning the system. Imagine running your transactional application and analytical application on the same hardware and same data; Imagine doing all this without building cubes, summary tables, indexes, statistics, partitioning strategies or changing your application.

Change is constant, as is the need for higher speed in business. Data analysis helps organizations understand patterns, predict trends and quickly adjust business flow. You can gain business advantage by analyzing quickly and consistently on the latest data—at a low total cost of ownership (TCO).

Data analysis queries are called complex for a reason. They access millions or billions of rows, can produce large intermediate results and perform best when they are parallelized. The star schema and star join optimization techniques are designed to handle these workloads. To meet SLAs, it takes experienced data architects and DBAs to understand such workloads, and tune the system parameters and indexes to improve and maintain the workload performance.

Informix takes a novel approach. IBM® Informix® Advanced Enterprise Edition, which includes IBM® Informix® Warehouse Accelerator (IWA), is a complete warehouse database server with extract, transform and load (ETL) tools; built-in support for advanced online time-cyclic data management; online operations; deep compression; and query optimization and processing techniques designed for complex data warehousing workloads. IWA boosts analytical query performance using in-memory database techniques, without requiring physical schema redesign or application redesign.

"Before using Informix Warehouse Accelerator, complex inventory and sales analysis queries on the enterprise warehouse with more than a billion rows took anywhere from a few minutes to 45 minutes to run. When we ran those same queries using Informix Warehouse Accelerator, they finished in 2 to 4 seconds. That means they ran from 60 to 1400 times as quickly, with an average acceleration factor of more than 450—all without any index or cube building, query tuning or application changes."
–Ashutosh Khunte, Senior Database Architect, Skechers USA

Traditional warehouse database systems were designed to maximize disk I/O throughput. Because processor-to-memory access is orders of magnitude faster than processor-to-disk access, traditional database systems optimize operations to minimize disk I/O, using buffering to keep recently accessed data in memory.

Memory prices have fallen – a terabyte of memory costs USD 20,000; Multi-core processors are common – IBM System X ships with 4 or 8 Intel Westmere processors amounting to 40 and 80 physical cores. New generations of processors add cores and increase on-chip cache sizes. Hence, large memory and CPU configurations are affordable.

"It offers highly impressive performance with queries running 30 times faster than previously. The columnar technology saves a lot of processing time; it reduced our workload time from 9.5 hours to 15 minutes, all without any database tuning or need to manage the physical storage."
– Lester Knutsen, Informix Data champion, Advanced Data Tools

---

Informix Warehouse Accelerator is an In-Memory database (IMDB), designed to exploit these affordable innovations in memory and processor technology and trends in novel ways to boost query performance. To take advantage of the increased availability of system memory, it compresses the data and keeps all of the compressed data in memory, eliminating disk I/O overhead. To take advantage of the larger caches, the accelerator optimizes its algorithms to parallelize queries and minimize synchronization. The explain details of novel approaches for improving query performance are explained later in this paper.

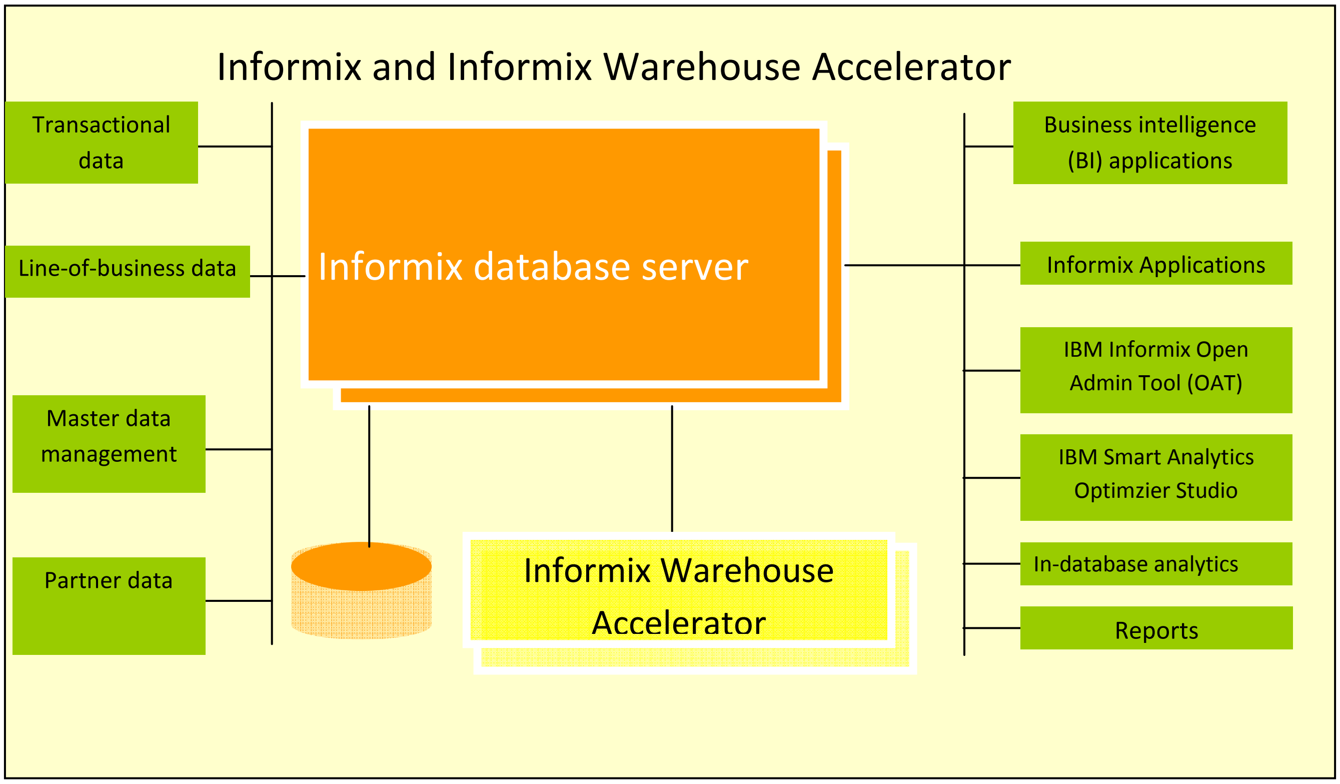

Informix and Informix Warehouse Accelerator
Transactional data
Line-of-business data
Master data management
Partner data
Informix database server
Informix Warehouse Accelerator
Business intelligence (BI) applications
Informix Applications
IBM Informix Open Admin Tool (OAT)
IBM Smart Analytics Optimzier Studio
In-database analytics
Reports

Figure 1: Using Informix and IWA with BI applications

These techniques enable Informix Warehouse Accelerator to provide breakthrough query performance while eliminating or minimizing the following tuning tasks which are required for traditional data analysis:

- **Indexes, index advisors and index reorganization.** The accelerator's query processing engine is designed to logically scan millions of rows in seconds or milliseconds without the use of indexes—enabled by features such as deep columnar data representation, query processing on compressed data and innovative algorithms that exploit modern processor trends.
- **Statistics collection and advisors.** Traditional optimizers rely on regular statistics collection to improve query plans. In contrast, Informix Warehouse Accelerator automatically determines join orders and consistently uses star join plans. Runtime optimization and the lack of indexes mean that statistics collection and statistics collection advisors are not needed.
- **Manual partitioning schemes.** The accelerator automatically partitions data both vertically and horizontally. Queries also benefit from the vertical and horizontal partition pruning (also known as *fragment or partition elimination*) because of cell-based deep columnar storage.
- **Manual tuning for each query or workload.** During installation of the accelerator, administrators provide basic memory and storage configurations. Afterwards, runtime tuning is avoided through consistent plans, elimination of disk I/O and incredibly fast scans and joins.
- **Storage management.** Data is stored in memory, with a copy of the in-memory image on disk (for reading the compressed data on restart or recovery only). There is no need for administrators to plan and create storage spaces for tables and indexes.
- **Database changes.** The accelerator exploits the logical schema in the existing data warehouse.
- **Application changes.** The accelerator plugs into the Informix database server as a resource. Informix knows which data marts are stored in the accelerator and automatically routes relevant queries to the accelerator.
- **Summary tables and related advisors.** Accelerator table scans and joins are at least an order of magnitude faster than those of traditional databases without the use of summary tables.
- **Page- or block-size configuration.** A deep columnar approach automatically determines and optimizes in-memory cell size.
- **Temporary space allocation.** The intermediate result set is compressed and stored in memory, avoiding the need for allocating temporary disk space.
- **Query and optimizer hints for accelerated queries.** The accelerator uses star join plans consistently. The accelerator query processor adjusts the join orders depending on the runtime statistics.

"Informix Warehouse Accelerator brings the capabilities of data warehousing and online transaction processing (OLTP) on a single platform. The queries can be run in a matter of seconds without having to make any additional tool investments."
– Thomas Gemesi, ATG IT Consulting GmbH

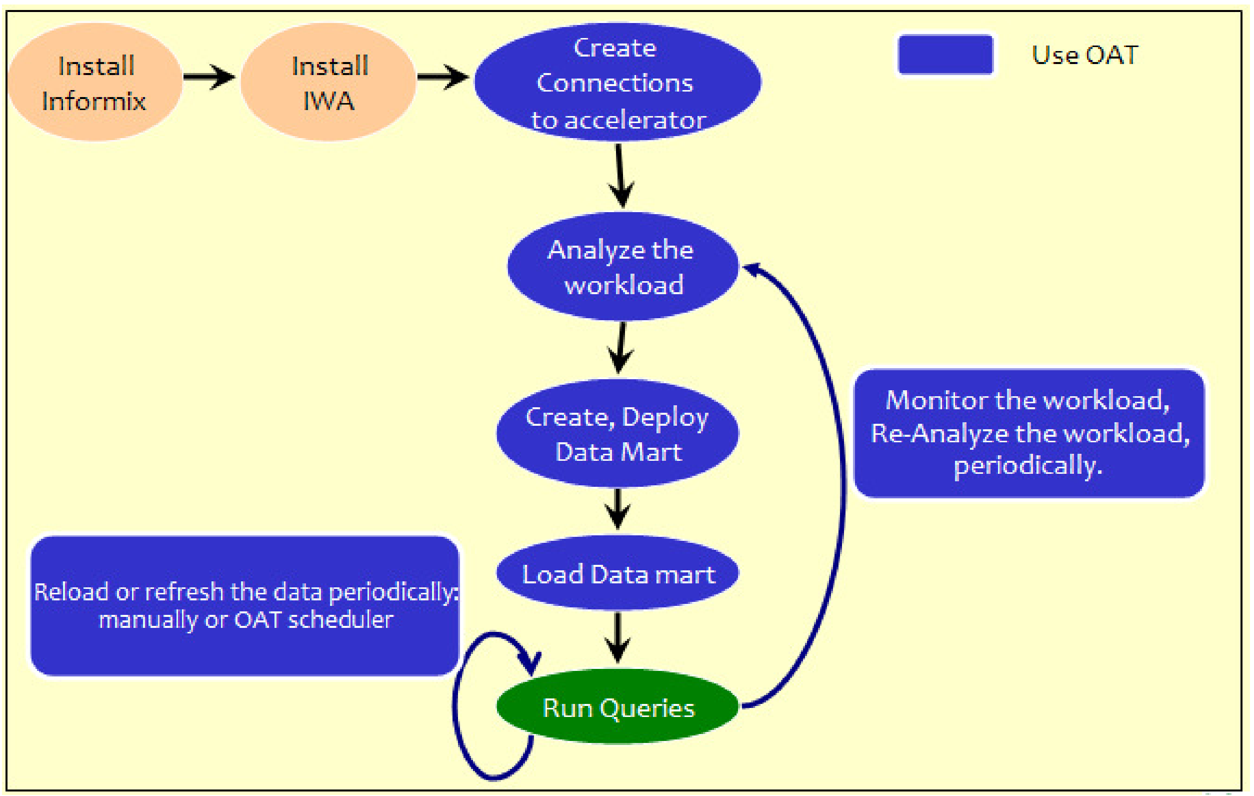
Overview of Informix Warehouse Accelerator

Informix Warehouse Accelerator is designed to run on commodity hardware—a high-performance Linux operating system on an Intel processor–based single server or a cluster. It tightly integrates with the Informix database server, running on a single server or a cluster. This combination is supported when an Informix database server is on any of the following platforms:

- Linux operating system on Intel processor–based servers;
- IBM AIX® operating system on IBM POWER7® processor–based servers;
- HP-UX operating system on Intel Itanium processor–based servers;
- Oracle Solaris operating system on Intel processor based server or Oracle SPARC servers.

The database server and the accelerator can be installed on the same or different computers.

Informix Warehouse Accelerator is a component of IBM Informix Advanced Enterprise Edition, which includes the Informix Enterprise Edition database server as well as Quick Start and Administration Guides that provide comprehensive documentation on installing and configuring the Informix database server and the accelerator. The package also includes IBM Open Admin Tool (OAT), a web application that can be used to administer the accelerator and define and deploy data marts from the Informix database to the accelerator. The package continues to include IBM Smart Analytics Optimizer Studio for administration and design. The Smart Analytics Optimizer Studio is available for Linux and Microsoft Windows operating system. Administrators can use the Linux version on the same computer as the Informix database server or on different computer. To use the Windows version, administrators can transfer the self-extracting binary to a Windows-based computer and install it there.


Install Informix
Install IWA
Create Connections to accelerator
Use OAT
Analyze the workload
Create, Deploy Data Mart
Monitor the workload, Re-Analyze the workload, periodically.
Reload or refresh the data periodically: manually or OAT scheduler
Load Data mart
Run Queries
Figure 2: Installing, deploying and using Informix and IWA.

Deploying the database server and the accelerator requires five basic steps (which are detailed in the *Informix Warehouse Accelerator Administration Guide*):

1. Install, configure and start the Informix database server as usual.
2. Install, configure and start the accelerator. Key configuration settings include location of the file system for retaining a copy of the memory image, amount of memory allocated for IWA and number of nodes.
3. Using OAT or Studio, connect to the Informix database and add the accelerator connection information.
4. Design and create the data marts on to IWA. You can create an optimal data mart by analyzing an existing workload (using OAT) or design the mart manually (using Studio).
5. Load data to the accelerator. The accelerator is now ready for queries.

Scenarios for using Informix Warehouse Accelerator.

1. **Accelerating data warehouse**: Data warehouses contain one or more data marts, defined to handle a particular subject area. Each data mart is typically a star or snow-flake schema. You can choose to accelerate a specific high value data mart and only the workload using this particular data mart will get accelerated without changing anything for the rest of the schema or workload. Typically, data is loaded to the warehouse nightly. You can update the IWA data with partition refresh or full load.
2. **Accelerating Operational Analytics**: As businesses try to get more responsive, analysis of transactional data is becoming common. You can create data marts on your transactional schema and deploy them on IWA. Data update from the transactional system can be trickle fed into IWA for near real time analysis.
3. **Accelerating timeseries data**: Time series data type and access method provides efficient storage and query processing of time series data. Use IWA if your analytical queries have many joins and ad hoc aggregations of this data. Create a VTI (Virtual Table Interface) table on the base Timeseries table to act as the fact table and then create a data mart.

Components of Informix Warehouse Accelerator

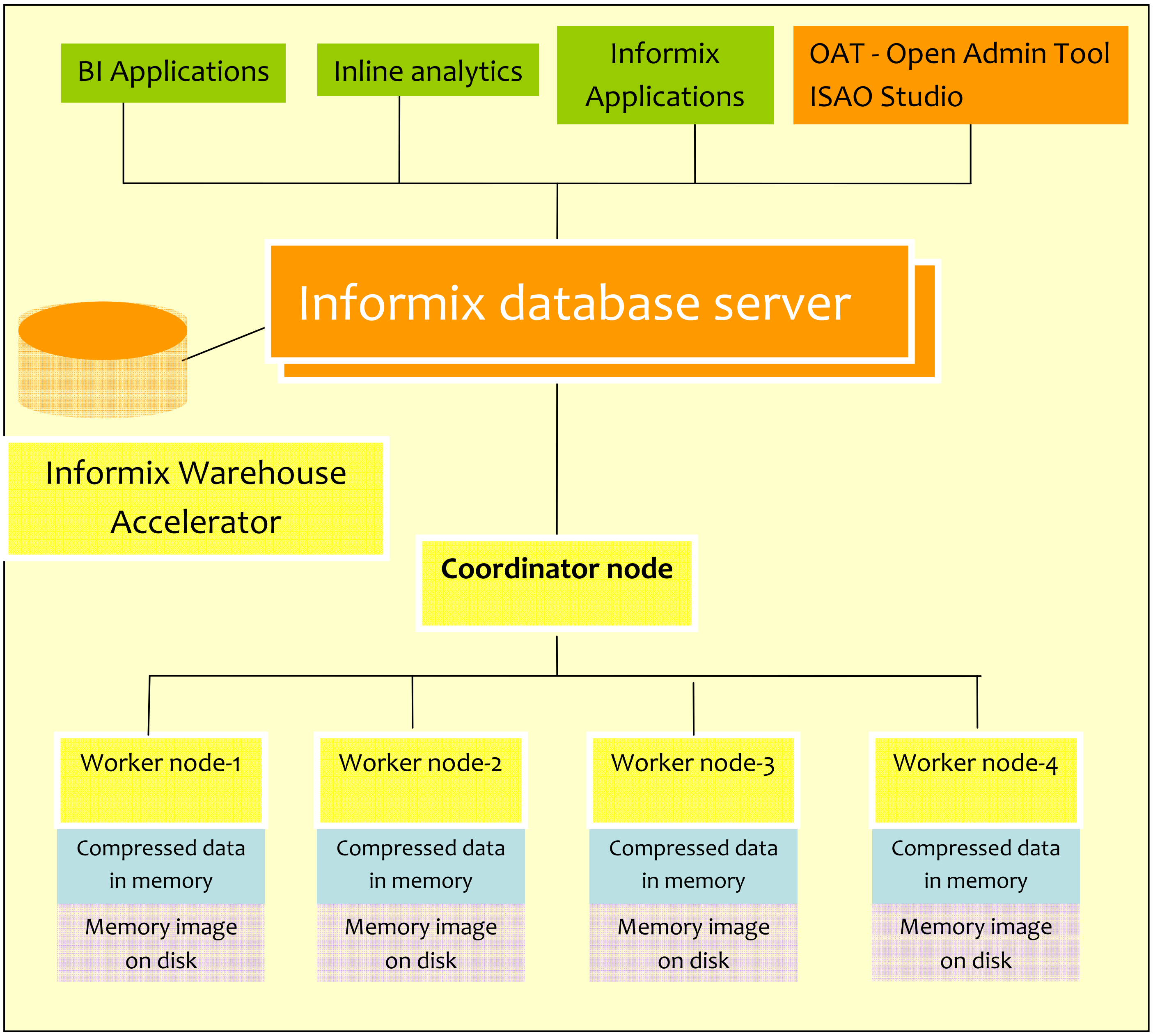
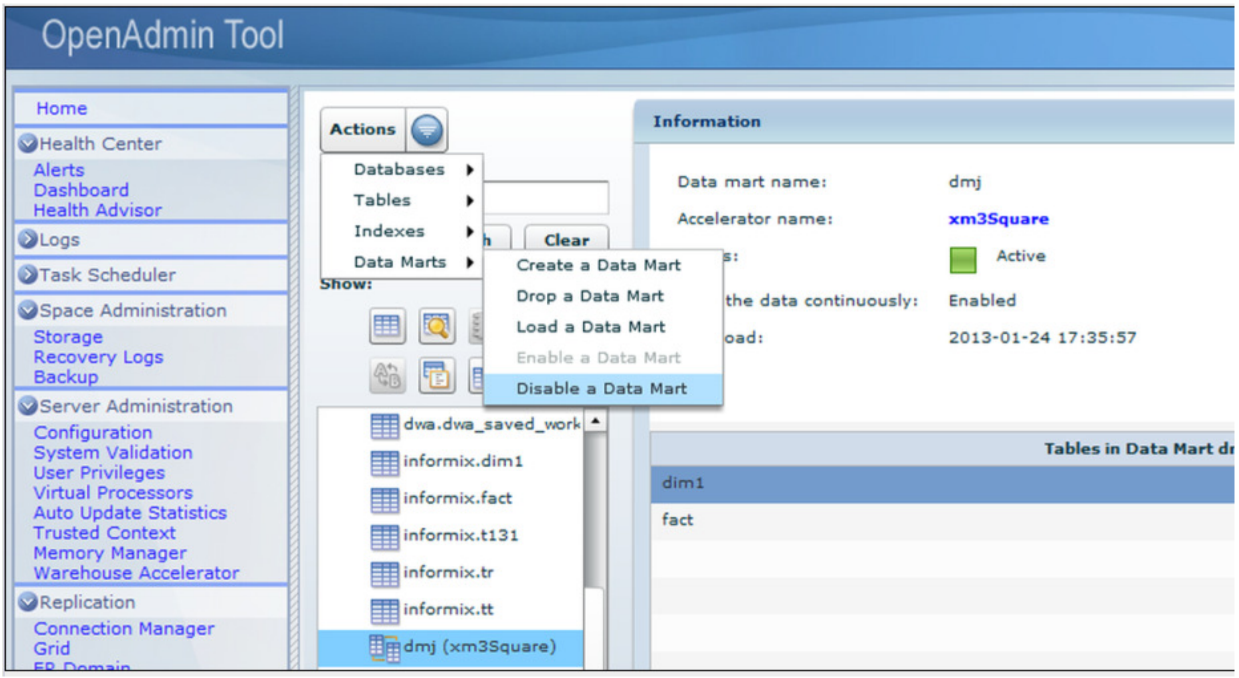


Figure 3: Components of 5-node IWA: one coordinator node and four worker nodes.

The role of Open Admin Tool (OAT)

OAT is a browser-based Informix administration tool. With Informix 12.10, you can use OAT to configure and administer IWA and manage data marts on it.



Data mart name: dmj
Accelerator name: xm3Square
s: 🟩 Active
the data continuously: Enabled
oad: 2013-01-24 17:35:57

dwa.dwa_saved_work
informix.dim1
informix.fact
informix.t131
informix.tr
informix.tt
dmj (xm3Square)

Tables in Data Mart dr
dim1
fact

Figure 4: Open Admin tool supports Informix Warehouse Accelerator version 12.10.

Using OAT, you can do the following tasks:

1. Configure Informix to connect to one or more IWAs
2. Design a data mart by analyzing the workload
3. Create or drop the data mart
4. Load the data mart
5. Full reload or partition based refresh of the data mart
6. Setup automatic periodic refresh by using full reload or partition based refresh or trickle feed
7. Enable and disable the data mart

Configuring Informix to IWA connection is simple: specify the name, IP address, socket number and a pin you get from ondwa utility. Informix communicates with the accelerator through the coordinator node. Worker nodes communicate only with the coordinator node.

After you configure the connection from Informix database server to the accelerator, en entry to the SQLHOSTS file is created:

```
sales_acc    group  --
      c=1,a=4b3f3f457d5f552b613b4c587551362d2776496f226e714d75217e22614742677b424224
sales_acc_1  dwsoctcp     127.0.0.1    21022  g=sales_acc
```

In this example, the name of the accelerator is **sales_acc**. Informix creates a new group with that name. The hexadecimal value is the authentication code used to ensure that only the Informix database communicates with the accelerator sales_acc. The name of the coordinator node is sales_acc_1. The database protocol **dw**, which is very similar to the Distributed Relational Database Architecture (DRDA) protocol, is used for communication over TCP/IP and is optimized for database server and accelerator communication. The TPC/IP loopback address 127.0.0.1 and coordinator node is listening on the socket 21022.

IBM Smart Analytics Optimizer studio can be used with Data Studio for integrated design, development and deployment. In IBM Smart Analytics Optimizer studio, you can also design the data mart graphically by choosing table by table and drawing the relationships to create the star or snowflake schema.

Informix 12.10 comes with a set of built-in stored procedures to manage data marts on IWA. These stored procedures can be started from SQL scripts or stored procedures, making the automation easy.

Informix Dynamic Server

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

The role of the coordinator node

The coordinator node provides the main point of communication between Informix Warehouse Accelerator and the Informix database server. The database server uses to the coordinator node to administer IWA, send data and submit queries, retrieve result sets.

During the data loading phase, the coordinator node receives the data from Informix, distributes the data among multiple worker nodes. The coordinator node then collects the entire compression dictionary, merges it and redistributes the dictionary so all nodes are using the same reference dictionary. During the query processing phase, the coordinator node receives a query from the database server, sends the query to each worker node, gets the intermediate result set, merges the groups, uncompresses the data and sorts the data if necessary before it sends the results to the database server.

Both the coordinator and worker nodes share query processing responsibility and work together to parallelize execution of every query and return results quickly.

The role of the worker node

In the data loading phase, each worker node analyzes the incoming data by using frequency partitioning, automatically partitions the data vertically and horizontally into cells and compresses the data by using deep columnar techniques (see section "Techniques for enabling peak performance"). After the data is compressed, it is written to disk for recovery. During the partition refresh and trickle feed task, workers handle the data update tasks as well.

The worker node also runs query processing 100 percent in memory, on compressed data. Each worker node maintains a compressed copy of the dimension tables and a portion of the fact table, and each returns intermediate results to the coordinator node.

Configuring memory for nodes

During installation of Informix Warehouse Accelerator, the DBA configures the number of the nodes and memory for them. The number of coordinator nodes and worker nodes are automatically determined. For example, specifying two nodes automatically creates one coordinator and one worker node. When IWA is running on a single computer, it is typical to use two-node configuration.

Consider a sample cluster configuration with four worker nodes and one coordinator node. The DBA deploys a data mart with a SALES fact table and CUSTOMER, STORE, and TIME dimension tables. The dimension tables are compressed and kept in memory of each worker; thus there are four copies of the dimension tables. The rows of the fact table SALES are evenly divided among the four worker nodes. As a result, each worker node maintains the dimension tables and 25 percent of the fact table's rows in main memory. The data transfer rate typically increases as the number of worker nodes increases, assuming that there is enough processor capacity. Many worker nodes also helps increase query processing speed, although less dramatically than it does the data transfer rate. The effect of the number of workers depends on the query as well. Given these considerations, how much memory should be allocated for each worker node? How much memory is needed by the system?

Generally, there is a 3:1 compression ratio for the uncompressed Informix data to the compressed Informix Warehouse Accelerator data in memory. If the fact table SALES and the dimension tables CUSTOMER, STORE, and TIME total approximately 100 GB in size, with most of that space taken by the SALES table, approximately 33 GB of memory is needed for the workers to store the data. DBAs can easily determine table sizes by using the Open Admin Tool (OAT) or by directly querying the catalogs.

Each worker node needs sufficient runtime memory to store intermediate results at runtime. Worker nodes dynamically allocate and release the memory that is required for query processing. Planning memory allocation for nodes is similar to planning temp space for the Informix database server. How much sorting of intermediate results is expected from the workload? How related is the data and how many groups are in each category? Although these factors are difficult to identify precisely, having extra memory with another one-fifth to one-third of the data size is sufficient.

The next stage of accelerator query processing is done by the coordinator node. The coordinator node needs memory for merging, decompressing and, when required, sorting the result set. Again, the memory that is required here depends on the expected size of the result set. Keep in mind that when the query is issued with the FIRST clause, for example SELECT FIRST 1000… ORDER BY sum(sales.amount), the coordinator node must sort all the data to get the first 1000. However, after the coordinator node creates the first 1000 groups, it can replace or reject the new groups.
The subsequent section describes how the nodes work with the dimension and fact tables of a data mart.

Flexible grid, high availability and IWA.

IBM Informix offers one of the industry's most comprehensive sets of high availability options. Informix on cluster distributes the workload and provides failover nodes to avoid downtime. Informix HDR (high availability data replication) and RSS (remote secondary servers) provide more insurance by copying the data to distinct storage across the network. In many cases, Informix customers use the additional nodes for analytics and reporting while they use cluster deployment for OLTP workload. You can find more information about this technology at: http://www-01.ibm.com/software/data/informix/.

After setting up the high availability solution, adding up IWA to work with your high availability topology is easy:

1. Install and configure IWA as usual.
2. From one of the nodes, set up the connection from Informix to IWA.
3. Copy the full IWA group entry in the SQLHOSTS file into the respective SQLHOSTS file of each Informix node.

You can now use and operate on IWA from any node.

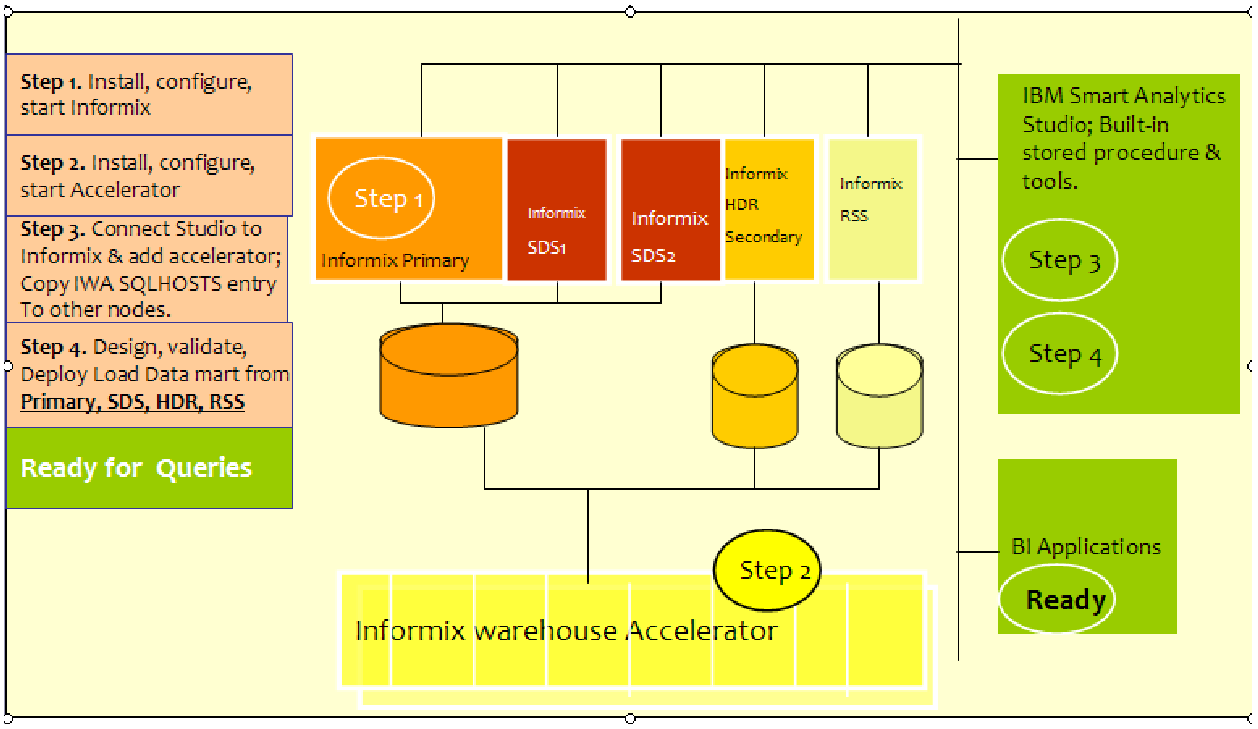

Step 1. Install, configure, start Informix
Step 2. Install, configure, start Accelerator
Step 3. Connect Studio to Informix & add accelerator; Copy IWA SQLHOSTS entry To other nodes.
Step 4. Design, validate, Deploy Load Data mart from Primary, SDS, HDR, RSS
Ready for Queries
Step 1
Informix Primary
Informix SDS1
Informix SDS2
Informix HDR Secondary
Informix RSS
IBM Smart Analytics Studio; Built-in stored procedure & tools.
Step 3
Step 4
Step 2
Informix warehouse Accelerator
BI Applications
Ready

Figure 5: Deploying IWA with Informix high availability.

Designing and deploying data marts

After the Informix Warehouse Accelerator is installed and configured, the next step is to set up a *data mart.* A data mart is defined as a subset of a data warehouse or your schema, oriented to a specific business line or team. An enterprise's data warehouse on the Informix database server can contain information from sales, inventory, customer service, market data, and the like. Your transactional schema is designed for maximizing the transactional throughput. You must select the subset that is required for analysis and specify star schema relationships.

A data mart defines fact and dimension tables and their relationships. The dimension tables typically contain fewer rows than fact tables, for example, product information and customer information. In some cases the dimension table can be large, for example, one that contains data on all California residents.

You can selectively accelerate subsets of the data marts and data by using Informix Warehouse Accelerator. For example, a sales manager might want to analyze sales and inventory data to understand trends and create suitable sales incentives. In this case, only the data marts with sales and inventory fact tables must be accelerated.

In the context of the accelerator, a data mart contains one or more snowflake schemas, each of which has one fact table and related dimension tables. In the snowflake schema that is shown in Figure 6, the fact table DAILY_SALES is related to dimensions that describe the business facts.

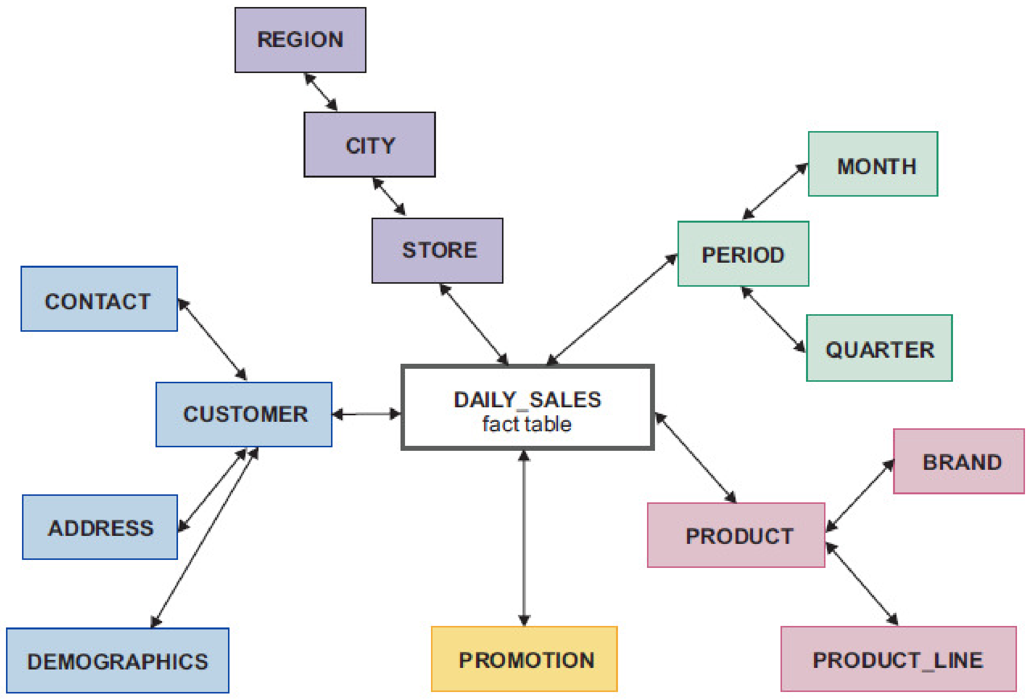


Figure 6: A sample snowflake schema with the DAILY_SALES fact table

After the tables to create a data mart are identified, the DBA defines the relationships between the fact table and the dimension tables. The data mart validation step helps ensure that all the relationships between the tables are defined. The DBA must address any errors in this step before the data mart is deployed.
As part of data mart deployment, the administration tools (OAT or Smart Analytics Optimizer Studio) send the data mart definition to the accelerator, which sends back the definition in SQL. This definition is saved as a special accelerated query table (AQT) view within Informix system catalogs with a special flag and related information. The AQT view is later used to match queries and redirect the matching queries to the correct data mart and accelerator.

Data marts can be designed by using the following methods:

- IBM Informix Open Admin Tool (OAT) can analyze a specified analysis workload and create an optimal data warehouse to accelerate the workload. The data mart definition only includes the tables and columns that are used to accelerate the workload.
- Built-in stored procedures automatically create the data mart from workload and query analysis Informix has built-in support to analyze and create the most optimal data mart for your workload. The generated data mart definition includes only the tables and columns that are used in the query, and automatically discovers the relationships between the tables. It helps you to create a data mart for a specific workload on a database with many tables, and it is an easy way to get started with data mart creation. For examples, see the Informix documentation.
- With IBM Smart Analytics Optimizer Studio, the data mart can be designed manually, table by table. You can explicitly define the relationships, deploy, and load the data mart.

Loading the data

After the data mart is created (deployed), the next step is to load the data. In this step, a snapshot of data from the database server tables is sent to the accelerator. The accelerator distributes the data to each of its worker nodes. The worker node analyzes the data for frequently occurring values and the relationships between the columns, and then it partitions the data vertically and horizontally. After partitioning, it compresses the data by using the deep columnar process that is described in the "Columnar storage" section of this white paper. After the compression, the worker node keeps the compressed data in memory with a copy on disk for persistence; no indexes, summary tables, or cubes are created. The data can be refreshed periodically from the Informix database server.

After the loading is complete, the data mart is ready for queries on IWA.

Data Refresh

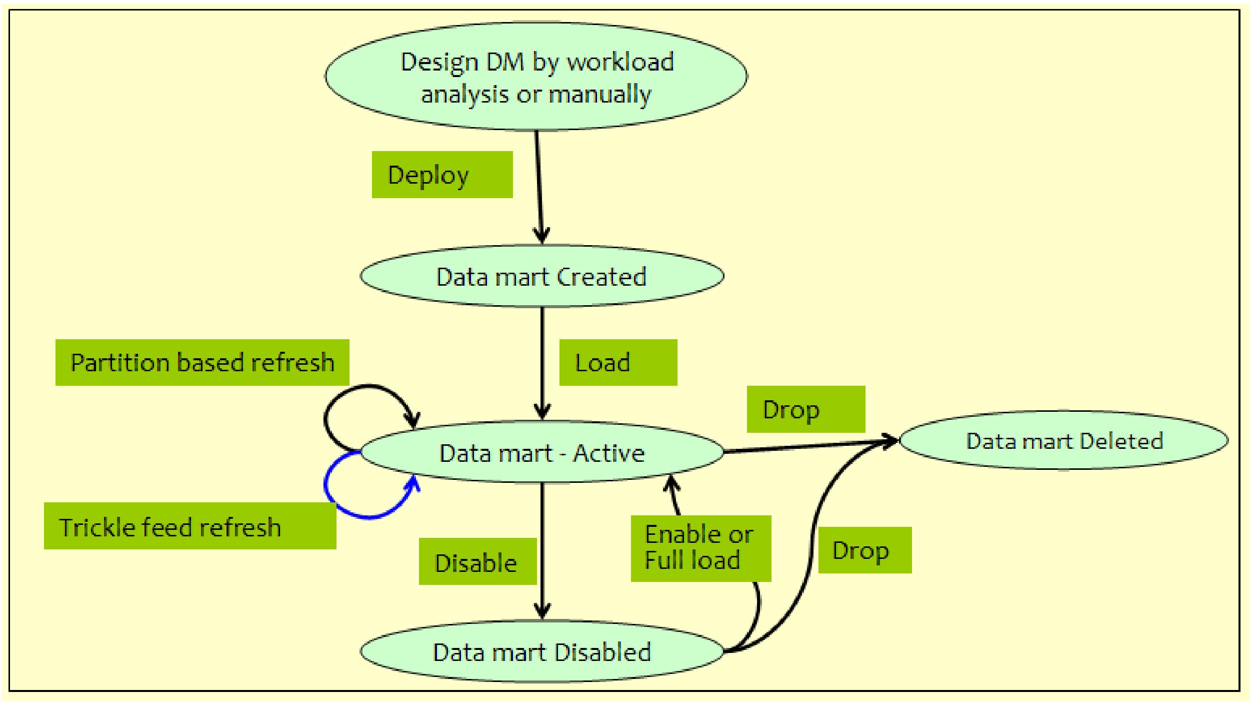


Figure 7: Data mart life cycle.

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

- Full data reload
- Refresh the partitions that are modified since the last refresh
- Trickle feed the data from Informix to IWA as the data is being inserted

Full data reload requires you to disable the data mart and then do a complete load of the data mart. The new data is transferred just like the first load, then analyzed to create new compression dictionary, and then compressed. Automatic partition refresh detects the fact and dimension table partitions modified since the last refresh. It then removes those partition data from IWA and, when necessary, reloads the data from respective partitions on to IWA. Automatic partition refresh also handles partition add, drop, attach, or detach to fact and dimension tables. Automatic partition refresh is useful when your data mart is designed to handle time cyclic data management.

Partition refresh is incredibly fast compared to full data reload since it reuses the existing compression dictionary. Because of this reuse, a slight loss of compression efficiency is possible. IWA creates new versions of the modified rows and hence can run query concurrent to partition refresh and trickle feed.

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

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

Query acceleration

Querying data typically involves joining the fact table with one or more dimension tables and then looking for specific patterns within the data. The query uses the WEB_SALES fact table is joined with 4 dimension tables. Informix matches the query to a specific data mart definition view (AQT) and then sends the query to the accelerator, in the same way a distributed query is sent from one Informix database server to another server. The result is returned over the same connection and is sent back to the client application (see Figure 8). The process is not apparent to the client application, except that the client receives the results much faster than during a non-accelerated session.

```
SET ENVIRONMENT USE_DWA ‘1’;  -- allow acceleration
SELECT FIRST 1000
       i_item_id,
       AVG(ws_quantity)    avg_quantity,
       AVG(ws_list_price)  avg_list_price,
       AVG(ws_coupon_amt)  avg_coupton_amt,
       SUM(ws_sales_price) sum_sales_price
FROM   web_sales,
       customer_demographics,
       date_dim,
       item,
       promotion
WHERE  ws_sold_date_sk = d_date_sk
       AND ws_item_sk = i_item_sk
       AND ws_bill_cdemo_sk = cd_demo_sk
       AND ws_promo_sk = p_promo_sk
       AND cd_gender = 'F'
       AND cd_marital_status = 'M'
       AND cd_education_status = 'College'
       AND ( p_channel_email = 'N'
              OR p_channel_event = 'N' )
       AND d_year = 2001
GROUP  BY i_item_id
ORDER  BY SUM(ws_sales_price) DESC;
```

The accelerator requires queries to join the fact table with zero or more dimension tables and to join each table by using the join keys that are specified when the data mart relationship is defined. It also

supports inner joins and left joins with the fact table on the dominant side. (For more information about query qualification, see the *Informix Warehouse Accelerator Administration Guide*.)
The accelerator runs each query on a first-come, first-serve basis without interruption. All the data and intermediate results are stored in memory. Each worker node has a copy of the dimension table to join with, and each join thread tries to cache the hash table in the level 2 (L2) cache to optimize memory access. The worker nodes scan and join independently with little data exchange and synchronization with other worker nodes. Typically, each query finishes in a few seconds, in comparison to the minutes and hours that are taken by a traditional system.

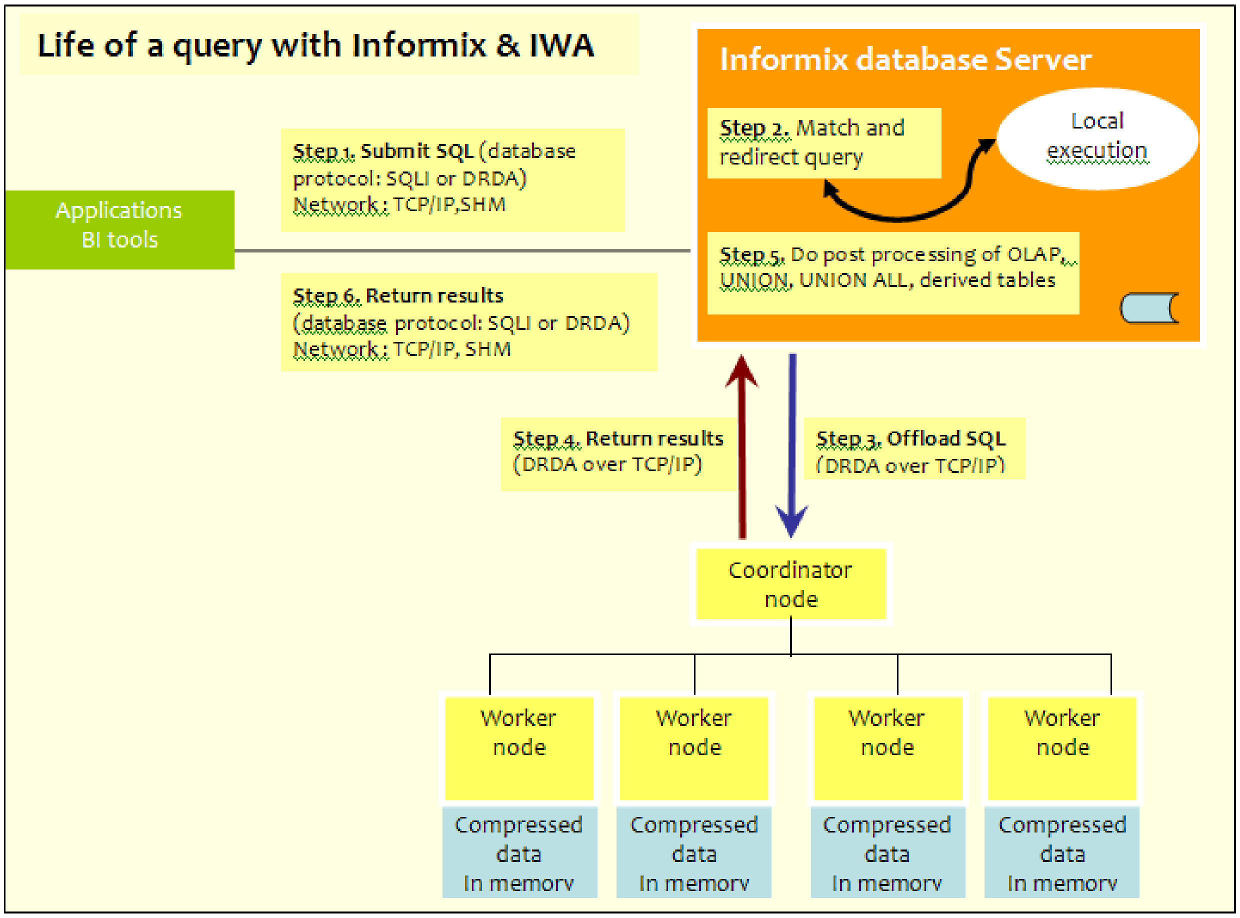

Life of a query with Informix & IWA
Informix database Server
Step 2. Match and redirect query
Local execution
Step 5. Do post processing of OLAP, UNION, UNION ALL, derived tables
Step 1. Submit SQL (database protocol: SQLI or DRDA) Network: TCP/IP,SHM
Applications BI tools
Step 6. Return results (database protocol: SQLI or DRDA) Network: TCP/IP, SHM
Step 4. Return results (DRDA over TCP/IP)
Step 3. Offload SQL (DRDA over TCP/IP)
Coordinator node
Worker node
Compressed data In memory
Worker node
Compressed data In memory
Worker node
Compressed data In memory
Worker node
Compressed data In memory

Figure 8: Query flow between Informix database server and Informix Warehouse Accelerator

IWA is an in-memory database system that is designed to maximize the SJP (select-join-project) performance. Sometimes, when the complete cannot be run by IWA, Informix maximizes query performance by using it for SJP processing.

The following example shows the plan for a SQL with OLAP window functions in Informix 12.10.

```
QUERY: DWA executed: (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) 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: sitaramv.d1.d_year = 2001

  2) dwa_ds.store_sales: INDEX PATH

    (1) Index Name: sitaramv. 125_170
        Index Keys: ss_sold_date_sk   (Serial, fragments: ALL)
        Lower Index Filter: sitaramv.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: sitaramv. 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
```

The first two sections give you the plan for query execution on IWA and Informix. The execution itself is discussed in the next section. For this query, Informix runs part of the query on IWA – represented by the "dwa" iterator that is shown in the query statistics section of the example. IWA does the SJP processing, 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.

Similarly, Informix 12.10 evaluates and accelerates each query that is connected by UNION or UNION ALL set operator, applicable derived table queries to maximize analytical performance.

Techniques for enabling peak performance

To help improve performance and eliminate tuning and maintenance tasks, Informix Warehouse Accelerator uses several techniques that are developed by IBM research and development. For example, the deep columnar approach goes beyond traditional columnar storage to provide:

- Extreme compression and query processing on compressed data, and elimination of disk I/O during query processing, which enables large in-memory warehouses.
- Exploitation of multi-core architectures and single-instruction, and multiple-data (SIMD) processing enables incredible speed without indexes or summary tables.

The "Additional reading" section lists papers that provide further details of underlying theory and techniques.

Frequency Partitioning

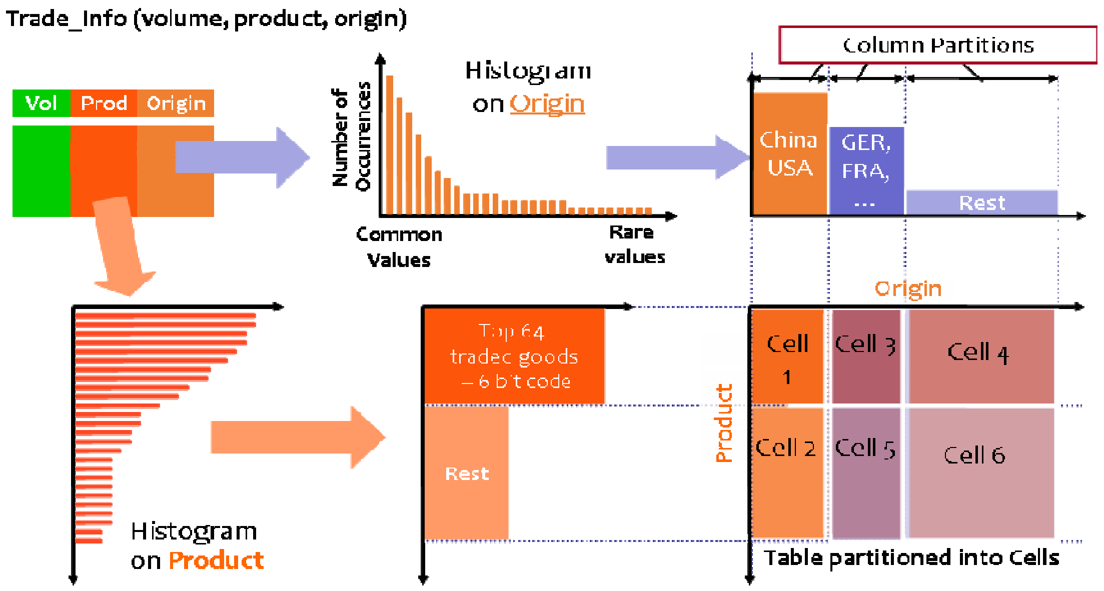


Figure 9: Correlation of product and origin columns during frequency partitioning

Each table in the data mart is analyzed for frequently occurring values in columns and related column groups to determine the optimal columns that are combined to form a *tuplet*, which is a fraction of a complete row, or *tuple*. In the example that is shown in Figure 9, the 2 columns product and origin are correlated and hence combined to form a tuplet.

In Figure 9, the top 64 product values are combined with the most frequently occurring values in origin (USA, China) to form the smaller cell 1 by using Huffman encoding. The benefit of Huffman encoding is that most frequently occurring values are encoded with the least number of bits. This technique increases compression efficiency and is used to evaluate both equality and range predicates. Since query processing is done on compressed data, fewer bits translate to higher speed.

Columnar storage

Traditional row-wise databases store a complete row in a page followed by another row. The design optimizes row I/O efficiency, assuming the query is interested in most of the column values. If the

data is compressed in storage, the row is uncompressed for every fetch. This process is efficient for transactional workloads that access a few rows per query.
Analytical queries typically access a million or more rows, but each query analyzes the relationship between subsets of columns in the fact table. For example, to find out the total sales of items per location in 2010, the required query must access only 3 of the columns in the fact table, item, sales, and location, and join them with the dimension tables. In this case, accessing and uncompressing every row is inefficient.

A columnar database stores all column values together. Every time data is inserted or loaded, each row is extracted to separate the column values. Every time a row is accessed, the column values are fetched separately and then compressed to form the row. Because the column values are stored together, better compression can be achieved. The previous example showed that analytical queries are typically interested in a subset of the rows. In columnar databases, only pages that store item, sales, and location data must be fetched. For queries that access a large subset of rows and doing sequential scans, columnar storage helps improve efficiency.

Informix Warehouse Accelerator stores data in column groups, which are vertical partitions of the table called *banks*. The assignment of columns to banks is cell-specific because column lengths vary from cell to cell. The assignment uses a bin-packing algorithm that is based on whether the column fits in a bank whose width is some fraction of a word rather than usage in a workload. Also, scans must access only the banks that contain columns that are referenced in any query, which avoids scanning banks with no columns referenced in the query. This projection is similar to the way pure column stores minimize disk I/Os. The accelerator stores all data in memory; even though there is no disk I/O, this technique minimizes the amount of memory that is needed for scanning and saves a considerable number of processor cycles.

Single-instruction, multiple-data parallelism

Consider the following query:
```
SELECT SUM(s.amount)
FROM   sales AS s
WHERE  s.prid = 100
GROUP  BY s.zip;
```

If the columns amount (A), prid (P) and zip (Z) are from the same bank, multiples of these values can be loaded into a 128-bit processor register at the same time. In this case, there are 12 column values at a time (see Figure 10).

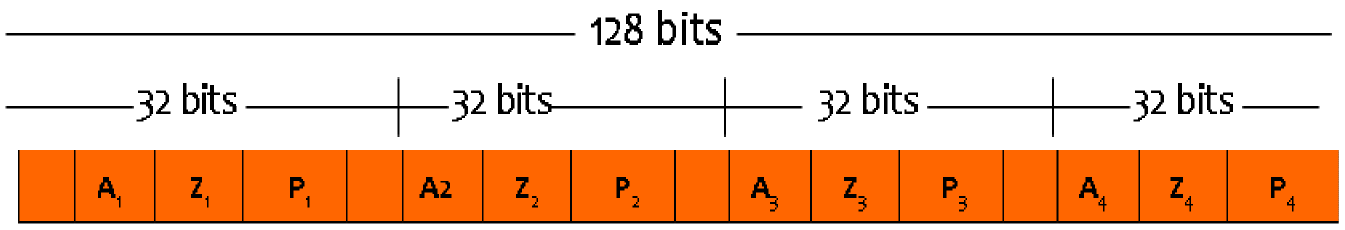


Figure 10: Loading column values into processor registers

SIMD instructions on Intel Xeon processors operate on 128-bit registers. The compression technique that is used by the accelerator typically requires few bits for each column and hence can load many fields in each 128-bit register. The technique applies predicates on all columns simultaneously. During query processing, this overloaded operation occurs on all the allocated cores, resulting in extreme parallelism for the query. All 12 values can be operated on simultaneously with a single processor instruction, resulting in significant performance gain (see Figure 11).

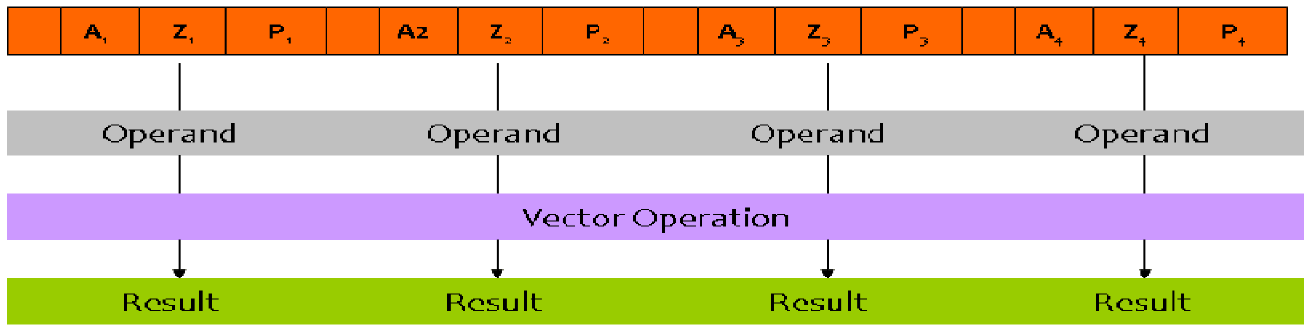


Figure 11: Operating on all values in the registers

Query processing

The previous sections gave an overview of how data is encoded and stored and how processing is done at a micro level. Query processing at a macro level is described here.
Efficient scanning provides the foundation for query processing. In scanning, the *cell* is the unit of processing. The accelerator dynamically creates appropriate threads to use configured processor resources and assigns a cell to each core (see Figure 12). This scan operates on compressed data by using SIMD instructions and using the advantages of Huffman encoding. Predicate evaluation GROUP BY is done on compressed data, but aggregation is done on decompressed data.

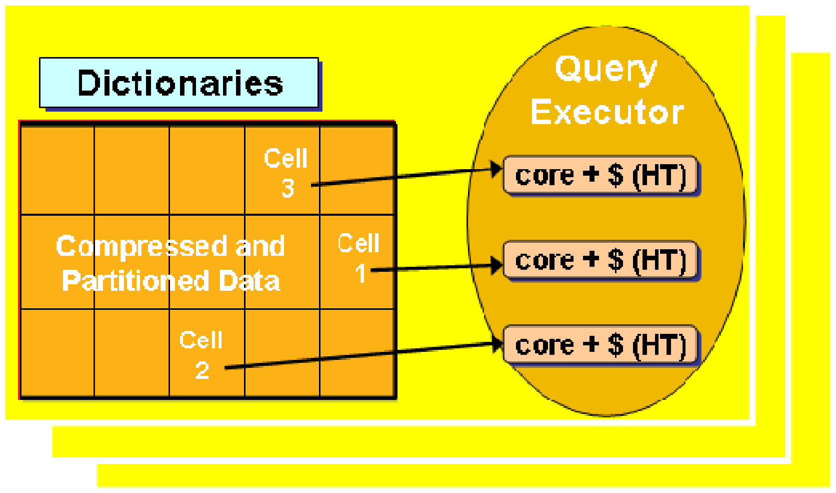


Figure 12: Efficient parallel scanning for query processing

The encoding technique also works as a dense hash function that allows caching of the hash table in the processor's L2 cache and a quick look-up. The technique enables quick `GROUP BY` on compressed data. Joins between two densely encoded hash tables can result in a sparse set of values. The accelerator detects these situations and switches over to linear probing dynamically. A combination of these techniques enables query processing on compressed data.

Queries on a data mart join the fact table with the dimension tables by using the join predicates between fact table to dimension tables and then dimension tables to other dimension tables. For each query, each worker node creates a snowflake model for the tables. It then starts at the outermost edge of the branch and works its way into the fact table. Each snowflake branch is processed, and its result acts as dimensional input for next level. First, the local predicates are applied to dimension tables to create a list of qualified keys. These keys from the dimension table are then joined with fact table (or the dimension table that acts as a fact table at the snowflake branch) to form the next level of aggregations and relations. This process is applied recursively until the complete join is processed at each worker node.

The coordinator node gets the intermediate result set from each worker node, merges the intermediate results into appropriate groups, decompresses the data, and then runs `ORDER BY` and `HAVING` instructions before it sends the data to the Informix database server over the DRDA protocol. The Informix database server then routes the data to the application program. Because there is only one representation of the data, compressed columnar table data, the accelerator follows the same code path for each table every time, which enables accelerator performance to be consistent. All efficiencies of cell and block elimination for the query comes from compression encoding instead of indexes or summary tables.

Conclusion

The innovative approaches to complex query processing taken by Informix Warehouse Accelerator can help improve the productivity of an enterprise by providing quick answers without increasing the amount of manual work or budget required. Because the accelerator is tightly integrated with the Informix database server, DBAs can divide the load between the database server and the accelerator as necessary.

Fast response time means quick answers, quick insights, and an agile business. Using Informix Warehouse Accelerator, enterprises can plan to accelerate the high-value aspect of their warehouses and dynamically evolve their infrastructures to suit business needs.

Further information

To learn more about Informix Warehouse Accelerator and Informix Ultimate Warehouse Edition, please contact your IBM Representative or IBM Business Partner, or visit the following websites:
ibm.com/informix
ibm.com/informix/warehouse

Additional reading

Allison L. Holloway, Vijayshankar Raman, Garret Swart, David J. DeWitt: How to barter bits for chronons: compression and bandwidth trade offs for database scans. SIGMOD 2007: 389–400
Ryan Johnson, Vijayshankar Raman, Richard Sidle, Garret Swart: Row-wise parallel predicate evaluation. PVLDB 1(1): 622–634 (2008)
Lin Qiao, Vijayshankar Raman, Frederick Reiss, Peter J. Haas, Guy M. Lohman: Main-memory scan sharing for multi-core CPUs. PVLDB 1(1): 610–621 (2008)
Vijayshankar Raman, Garret Swart: How to wring a table dry: Entropy compression of relations and querying compressed relations. PVLDB: 858–869 (2006)
Vijayshankar Raman, Garret Swart, Lin Qiao, Frederick Reiss, Vijay Dialani, Donald Kossmann, Inderpal Narang, Richard Sidle: Constant-Time Query Processing. ICDE 2008: 60–69
Knut Stolze, Vijayshankar Raman, Richard Sidle, O. Draese: Bringing BLINK Closer to the Full Power of SQL. BTW 2009: 157–166
Ronald Barber et al: Business Analytics in (a) Blink. IEEE Data Eng. Bull. 35(1): 9-14 (2012)
Ronald Barber et al: Blink: Not Your Father's Database! BIRTE 2011

Acknowledgements

Informix Warehouse Accelerator was developed through the collaboration of IBM Almaden Research, IBM Böblingen lab and the IBM Informix team. Thanks to the Informix team for reviewing and improving this paper.