IBM Software
White Paper

October 2011

IBM Informix Warehouse Accelerator

Performance is everything

Keshava Murthy, Senior Technical Staff Member, IBM Informix Development

Contents

Introduction

Imagine getting reports to business analysts and C-level executives within seconds instead of hours. Increasing the speed of analysis and insights. Improving business execution speed with data points available within seconds. Improving warehouse query performance without constantly monitoring and tuning the system.

Imagine doing all this without building cubes, summary tables, indexes, statistics or partitioning strategies.

Change is constant, as is the need for speed. Data analysis can help organizations understand patterns, predict trends and adjust business flow. And performing this analysis quickly and consistently—at a low total cost of ownership (TCO)—can give organizations an advantage over the competition.

But traditionally, data warehouse queries are called complex for a reason. They access millions and billions of rows, produce large intermediate results and perform best when they are parallelized. The star join optimization technique is designed to handle these workloads in a traditional system. It takes an experienced DBA to understand such workloads and tune the system parameters and indexes to suit the workloads.

To improve the performance of warehouse queries while minimizing manual tuning, DBAs can turn to the IBM® Informix® Warehouse Accelerator. The accelerator boosts the speed of warehouse queries to the IBM Informix database server, a scalable and highly available relational database used to drive mission-critical transactional and analysis applications (see Figure 1). Informix is a complete warehouse database server with extract, transform and load (ETL) tools; built-in support for time-cyclic data management; online operations; deep compression; and query optimization and processing techniques designed for complex data warehousing workloads.

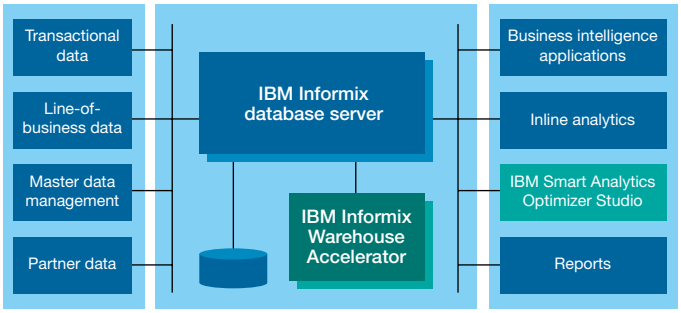

Figure 1: Informix Warehouse Accelerator architecture

Informix Warehouse Accelerator is designed to take advantage of innovations in memory and processor technology. 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. However, while the systems assume low memory configurations, falling memory prices have resulted in affordable systems with multiple-terabyte memory capacity. At the same time, every new generation of processors adds cores and increases on-chip cache sizes.

Informix Warehouse Accelerator exploits these trends in several ways to boost query performance. To take advantage of the increased availability of system memory, it compresses and caches data in memory, eliminating the overhead of disk I/O. To take advantage of the larger caches, the accelerator optimizes its algorithms to parallelize queries and minimize synchronization. Plus, Informix Warehouse accelerator can be deployed on IBM BladeCenter® to scale both capacity and performance. It can also be deployed on virtual and cloud environments, providing additional choices and cost savings.

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

- **Indexes, index advisors and index reorganization.** The accelerator's query processing engine is designed to logically scan billions 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 features like Single Instruction Multiple Data (SIMD) and larger L2 and L3 cache.
- **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 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. Afterward, runtime tuning is avoided through consistent plans, elimination of disk I/O and fast scans and joins.
- **Storage management.** Data is stored in memory, with just a copy of the in-memory image on disk. Administrators do not need 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 datamarts 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.

Deploying Informix Warehouse Accelerator

Informix Warehouse Accelerator is designed to run on commodity hardware—a high-performance Linux operating system on an Intel processor–based server. It integrates with the Informix database server, which supports 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; and Oracle Solaris operating system on Oracle SPARC servers. When running Informix database server and Informix Warehouse Accelerator on Linux, the database server and the accelerator can be installed on the same or different computers.

"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

Informix Warehouse Accelerator is a component of Informix Ultimate Warehouse Edition, which includes the Informix Ultimate 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 Smart Analytics Optimizer Studio, an Eclipse-based administration interface for configuring and managing the accelerator and for defining and deploying datamarts from the Informix database to the accelerator. The Smart Analytics Optimizer Studio comes in two versions: one for Linux and another for the 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.

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.
2. Install, configure and start the accelerator. Key configuration settings include location of the file system for data backup, amount of memory and processor resources.
3. Connect the Smart Analytics Optimizer Studio to the Informix database and add the accelerator.
4. Design, validate and deploy the datamarts.
5. Load data to the accelerator. The accelerator is now ready for queries.

Components of Informix Warehouse Accelerator

Informix Warehouse Accelerator is connected to the Informix database server over a TCP/IP network. If the accelerator and the database server are on the same computer, they communicate by using a TCP/IP loopback connection. The accelerator uses coordinator and worker processes, or nodes (see Figure 2). Informix communicates with the accelerator through the coordinator node. Worker nodes communicate only with the coordinator. Both the coordinator and worker nodes share query processing responsibility and work together to parallelize execution of every query and return results quickly.

After a connection between the accelerator and the database server has been established, Informix adds the accelerator connection information into its SQLHOSTS file:

```
sales_acc      group      -- c=1,a=4b3f3f457d5f552b613b4c587551362d2776496f226
e714d75217e22614742677b424224
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 indicates that the accelerator is running on same computer as the database server. Configurations that include a large number of worker nodes will have multiple coordinator nodes for handling failover; each coordinator node will have its own entry in the SQLHOSTS file.

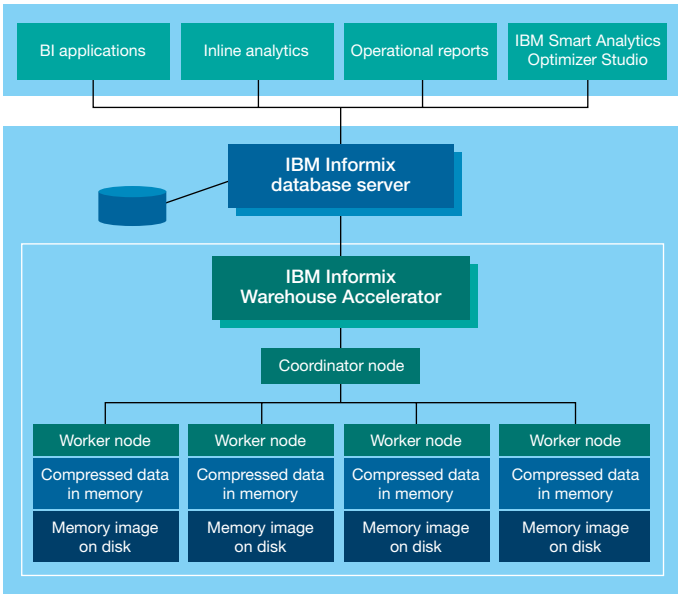

Figure 2: Sample architecture with one coordinator node and four worker nodes

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 connects to the coordinator node to send data and queries and also to retrieve result sets.

During the data loading phase, the coordinator node 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.

"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 1,400 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, Manager, Data Management Services, Skechers USA

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, decompresses the data and sorts the data if necessary before sending the final results to the database server.

The role of the worker node

In the data loading phase, each worker node analyzes the incoming data using frequency partitioning, automatically partitions the data vertically and horizontally into cells and compresses the data using deep columnar techniques (see section "Techniques for enabling peak performance"). Once the data is compressed, it is written to disk for recovery.

The worker node also performs 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 and memory configuration of the nodes. The number of coordinator nodes and worker nodes are automatically determined; for example, specifying five nodes will automatically create one coordinator and four worker nodes.

Consider a sample configuration with four worker nodes and one coordinator node. The DBA deploys a datamart with a SALES fact table and CUSTOMER, STORE and TIME dimension tables. The dimension tables are compressed and kept in private 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. A large number of 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 on the query depends on the query as well.

Given those 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 taken by the SALES table—approximately 33 GB of memory will be 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 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 those factors are difficult to identify precisely, having additional memory with another one-fifth to one-third of the data size is usually sufficient.

The final stage of accelerator query processing is done by the coordinator node. The coordinator node needs memory for merging, decompressing and sorting the result set. Again, the memory 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 1,000… ORDER BY sum(sales.amount)—the coordinator node must sort all the data to get the first 1,000. Of course, once the coordinator node creates the first 1,000 groups, it can replace or reject the new groups.

"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 DataTools

The next section describes how the nodes work with the dimension and fact tables of a datamart.

Designing and deploying datamarts

After the Informix Warehouse Accelerator has been installed and configured, the next step is to set up a *datamart*. A datamart can be defined as a subset of a data warehouse, usually 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.

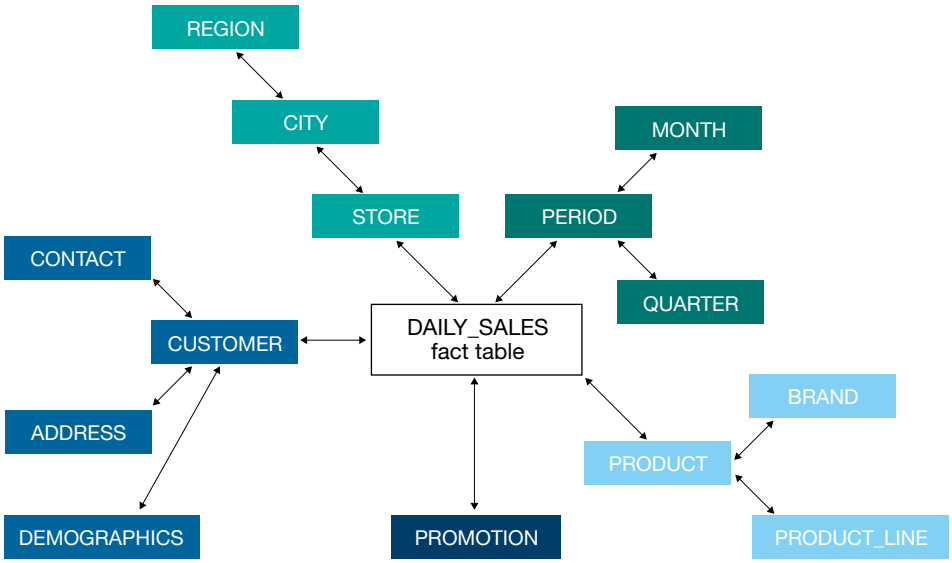

Figure 3: A sample snowflake schema with the DAILY_SALES fact table

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

To create high-value datamarts, enterprises can accelerate them using Informix Warehouse Accelerator. For example, a sales manager may want to analyze sales and inventory data to understand trends and create suitable sales incentives. In this case, only the datamarts with sales and inventory fact tables need to be accelerated.

In the context of the accelerator, a datamart contains one or more snowflake schema, each of which has one fact table and related dimension tables. In the snowflake schema shown in Figure 3, the fact table DAILY_SALES is related to dimensions that describe the business facts.

After identifying the tables to create a datamart, the DBA defines the relationships between the fact table and the dimension tables. The datamart validation step helps ensure that all the relationships between the tables are defined. The DBA should address any errors in this step before deploying the datamart.

As part of datamart deployment, IBM Smart Analytics Optimizer Studio sends the datamart definition in XML format to the accelerator, which sends back the definition in SQL. This definition is saved as a view within Informix system catalogs with a special flag and related information. This special view, known as the accelerated query table (AQT), is later used to match queries and redirect matching queries to the accelerator.

```
 create view “dwa”.”aqt2dbca0d9-509d-434b-9cc9-4a12c6de6b3d” (“COL16”,”COL17”
,”COL18”,”COL19”,”COL20”,”COL21”,”COL22”,”COL23”,”COL24”,”COL25”,”COL26”,”CO
 L27”,”COL28”,”COL29”,”COL30”,”COL31”,”COL32”,”COL33”,”COL34”,”COL35”,”COL3
 6”,”COL37”,”COL38”,”COL39”,”COL40”,”COL41”,”COL42”,”COL43”,”COL44”,”COL45”,
”COL46”,”COL47”,”COL07”,”COL08”,”COL09”,”COL10”,”COL11”,”COL12”,”COL13”,”CO
 L14”,”COL15”,”COL48”,”COL49”,”COL50”,”COL51”,”COL52”,”COL53”,”COL54”,”COL5
 5”,”COL56”,”COL57”,”COL58”,”COL59”,”COL60”,”COL61”,”COL01”,”COL02”,”COL03”,
”COL04”,”COL05”,”COL06”,”COL62”,”COL63”,”COL64”,”COL65”,”COL66”,”COL67”,”CO
 L68”,”COL69”,”COL70”,”COL71”,”COL72”,”COL73”,”COL74”,”COL75”,”COL76”,”COL7
 7”,”COL78”,”COL79”,”COL80”,”COL81”) as
  select x0.perkey ,x0.storekey ,x0.custkey ,x0.prodkey ,x0.promokey
    ,x0.quantity_sold ,x0.extended_price ,x0.extended_cost ,x0.shelf_location
    ,x0.shelf_number ,x0.start_shelf_date ,x0.shelf_height ,x0.shelf_width
    ,x0.shelf_depth ,x0.shelf_cost ,x0.shelf_cost_pct_of_sale
    ,x0.bin_number ,x0.product_per_bin ,x0.start_bin_date ,x0.bin_height
    ,x0.bin_width ,x0.bin_depth ,x0.bin_cost ,x0.bin_cost_pct_of_sale
    ,x0.trans_number ,x0.handling_charge ,x0.upc ,x0.shipping
    ,x0.tax ,x0.percent_discount ,x0.total_display_cost ,x0.total_discount
    ,x1.perkey ,x1.calendar_date ,x1.day_of_week ,x1.week ,x1.period
    ,x1.”year” ,x1.holiday_flag ,x1.week_ending_date ,x1.”month”
    ,x2.prodkey ,x2.upc_number ,x2.package_type ,x2.flavor
    ,x2.form ,x2.category ,x2.sub_category ,x2.case_pack ,x2.package_size
    ,x2.item_desc ,x2.p_price ,x2.category_desc ,x2.p_cost ,x2.sub_category_desc
    ,x3.storekey ,x3.store_number ,x3.city ,x3.state ,x3.district
    ,x3.region ,x4.custkey ,x4.”name” ,x4.”address” ,x4.c_city
    ,x4.c_state ,x4.zip ,x4.phone ,x4.age_level ,x4.age_level_desc
    ,x4.income_level ,x4.income_level_desc ,x4.marital_status
    ,x4.gender ,x4.discount ,x5.promokey ,x5.promotype ,x5.promodesc
    ,x5.promovalue ,x5.promovalue2 ,x5.promo_cost
  from
 ((((“informix”.daily_sales x0 left join “informix”.period x1 on (x0.perkey
    = x1.perkey ) )left join “informix”.product x2 on (x0.prodkey
    = x2.prodkey ) )left join “informix”.”store” x3 on (x0.storekey
    = x3.storekey ) )left join “informix”.customer x4 on (x0.custkey
    = x4.custkey ) )left join “informix”.promotion x5 on (x0.promokey
    = x5.promokey ) );
```

Figure 4: Sample accelerated query table with DAILY_SALES fact table and its dimension tables PRODUCT, STORE, CUSTOMER and PROMOTION

Figure 4 shows an example of an AQT with DAILY_SALES as the fact table. (Note that DBAs do not use this view; it is automatically created and is provided here for information.) SQL from applications or tools works as usual, using tables defined in the schema.

Automatically creating the datamart from workload/query analysis

Informix has a built-in tool that analyzes your workload and creates the most optimal datamart to accelerate your queries. This tool will generate a datamart definition that includes only the tables and columns used in the query, automatically discover the relationships between the tables, and add them to the definition. It helps you to create a datamart for a specific workload on a database with many tables—and it's also an easy way to get started with datamart creation (Figure 5 demonstrates the five steps). For more details and examples, check the Informix documentation.

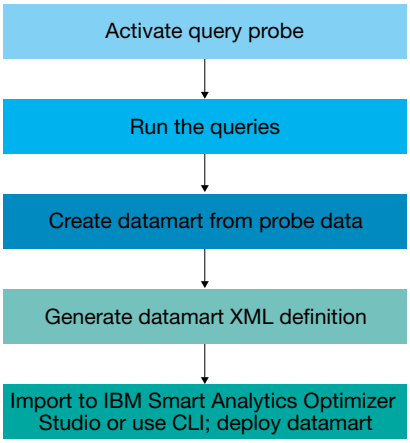

Figure 5: Informix contains a built-in tool to help you create a datamart in five steps.

```
select first 100 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;
```

Figure 6: Sample query that joins the WEB_SALES fact table with four dimension tables

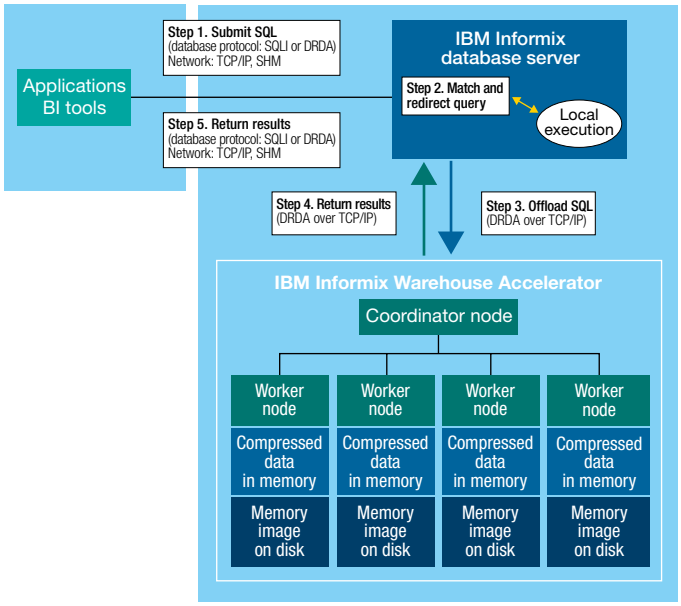

Figure 7: Query flow between Informix database server and Informix Warehouse Accelerator

Loading and querying data

After the datamart has been defined, 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. In this phase, 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 using the deep columnar process described in the "Columnar storage" section of this white paper. Note that in this phase, 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 (for example, every night) from the Informix database server.

As soon as loading is complete, the datamart in the accelerator is ready for use. Informix Warehouse Accelerator includes a command-line utility to design, validate, deploy and load datamarts. This utility is very useful for writing scripts to automate this process.

Querying data typically involves joining the fact table with one or more dimension tables and then looking for specific patterns within the data. Figure 6 shows an example of a query where the WEB_SALES fact table is joined with four dimension tables. Informix matches the query to a specific datamart definition view (AQT) and then sends the query to the accelerator—just like a distributed query from one Informix database server to another server. The result returns over the same connection and is sent back to the client application (see Figure 7). The process is transparent to the client application, except that the client receives the results much faster than during a non-accelerated session.

The accelerator requires queries to join the fact table with zero or more dimension tables and to join each table using the join keys specified when defining the datamart relationship. 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 will finish in a few seconds, compared to the minutes and hours taken by a traditional system.

Techniques for enabling peak performance

To help improve performance and eliminate tuning and maintenance tasks, Informix Warehouse Accelerator uses several techniques developed by IBM research and development. For example, the deep columnar approach goes beyond traditional columnar storage. Extreme compression and query processing on compressed data eliminates disk I/O during query processing and enables large in-memory warehouses. Exploitation of multi-core architectures and single-instruction, 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.

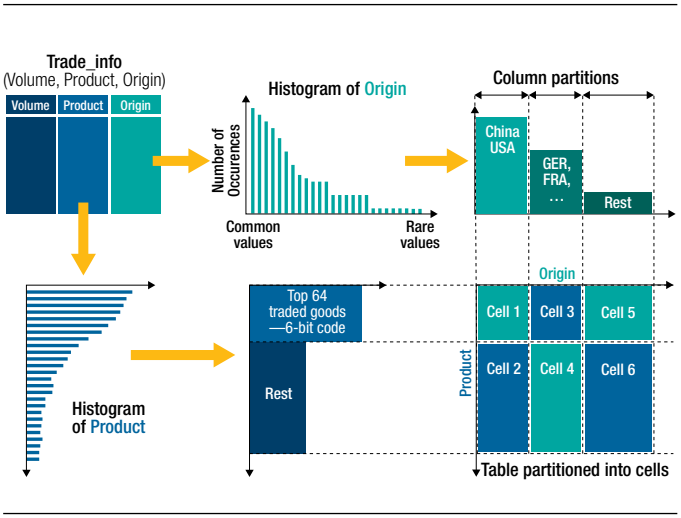

Figure 8: Correlation of product and origin columns during frequency partitioning

Frequency partitioning

Each table in the datamart is analyzed for frequently occurring values in columns and related column groups to determine the optimal columns that will be combined to form a *tuplet*, which is a fraction of a complete row, or *tuple*. In the example shown in Figure 8, the two columns Product and Origin are correlated and hence combined to form a tuplet.

In Figure 8, the top 64 product values are combined *with* the most frequently occurring values in origin (USA, China) to form the smaller cell 1 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 can be 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 the majority of the column values. If the data is compressed in storage, the row is decompressed for every fetch. This process is efficient for transactional workloads accessing few rows per query.

Analytical queries typically access millions or billions of 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 needs to access only three of the columns in the fact table—item, sales and location—and join them with the dimension tables. In this case, accessing and decompressing every row is inefficient.

A columnar database stores all column values together. Every time data is inserted or loaded, each row is unzipped to separate the column values. Every time a row is accessed, the column values are fetched separately and then zipped 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 subsets of the rows. In columnar databases, only pages storing item, sales and location data need to be fetched. For queries accessing a large subset of rows and doing sequential scans, columnar storage helps improves efficiency.

Informix Warehouse Accelerator stores data in columns 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 does not depend on how the column is used in a workload. Instead, the encoding is based on whether the column fits in a bank whose width is some fraction of a memory word. Also, scans need to access only the banks that contain columns referenced in any given 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 needed for scanning and saves considerable 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 9).

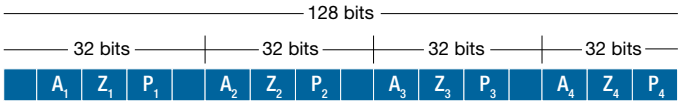

Figure 9: Loading column values into processor registers

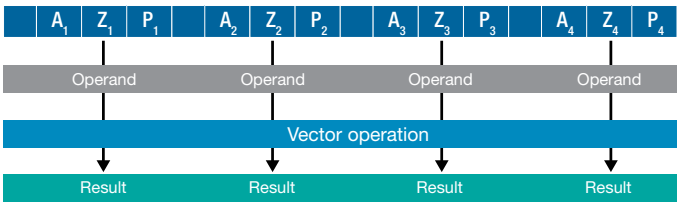

Figure 10: Operating on all values in the registers

SIMD instructions on Intel Xeon processors operate on 128-bit registers. The compression technique 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 10).

Query processing

The previous sections gave an overview of how data is encoded and stored and how processing is done at a micro level. This section describes query processing at a macro level.

Efficient scanning provides the foundation for query processing. In scanning, the *cell* is the unit of processing. The accelerator dynamically spawns appropriate threads to utilize configured processor resources and assigns a cell to each core (see Figure 11). This scan operates on compressed data using SIMD instructions and exploiting the advantages of Huffman encoding. Predicate evaluation GROUP BY is done on compressed data, but aggregation is done on decompressed data.

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

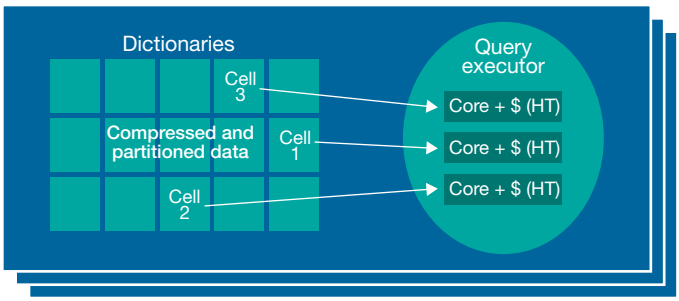

Figure 11: Efficient parallel scanning for query processing

Queries on a datamart will join the fact table with the dimension tables using the join predicates between the fact table and dimension tables and then between dimension tables and 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 acting 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 executes ORDER BY and HAVING instructions before sending 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 come 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

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.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
October 2011

IMW14587-USEN-00