IBM Information Management software

IBM Informix Warehouse Accelerator

Performance is everything

March 2011

Technical White Paper

by Keshava Murthy

Senior Technical Staff Member

IBM Informix Development

Imagine your business analysts, C-level executives getting their reports within seconds instead of hours. Imagine the increasing speed of analysis and insights. Imagine improving business execution speed with the data points available within seconds. Imagine improving your warehouse query performance without constantly monitoring and tuning the system. Imagine doing this without building cubes, summary tables, indices, statistics or creating partitioning strategy.

After extensive tests of Informix Warehouse Accelerator, Ashutosh Khunte, Senior Database Architect, of Skechers, USA says: "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!"

Lester Knutsen of Advanced Data Tools, Informix Data champion, took the data from his customer, United States Department of Agriculture, and ran his workload. He says: "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."

Thomas Gemesi, ATG IT Consulting GmbH says: "Informix Warehouse Accelerator (IWA) brings the capabilities of Data warehousing and OLTP (Online Transaction Processing) on a single platform. The queries can be run in a matter of seconds without having to make any additional tool investments."

Change is constant. Need for speed is constant. You have to understand the patterns, predict trends, and adjust the business flow. Data analysis help you to understand the trends and shifts in business. Doing this faster than anyone else, doing this consistently, doing this at lower TCO will give you an advantage over competition.

Informix Warehouse Accelerator provides extreme performance while removing most of the tuning required for traditional data warehouses. It is designed to process large data within seconds. It's designed to provide the business with the right data at right time without increasing the maintenance or changing application infrastructure.

Innovation is the key. The performance gap between processor to memory access speed and processor to disk access speed is increasing with time. Traditional database systems assume low memory configurations and optimize the operations to minimize disk IO and use buffering to keep recently accessed data in memory. Falling memory prices and affordable systems with terabyte memory capacity require new design. IBM has systems supporting 3TB of RAM and this capacity is expected to increase. The accelerator exploits this trend by compressing and caching all the data in-memory and eliminating disk IO. Processors are adding more cores and bigger on-chip cache with every release. The accelerator exploits this trend by parallelizing, minimizing synchronization and optimizing the algorithms to exploit on chip cache. This scales the query performance to new heights and leads to simple and elegant design.

Power of Simplicity

Perfection is achieved, not when there is nothing more to add,
but when there is nothing left to take away.
-- Antoine de Saint-Exupery

The accelerator has three key principles:

- Acceleration without manual tuning for each workload
- Support existing business tools and applications.
- Work with existing warehouse infrastructure

If you already have Informix in your environment, the accelerator plugs into your existing environment. For new deployments, you have flexible platform options and open integration with Informix database server.

Tuning
Indexes
Statistics
Storage mgmt Partitioning
Expensive Hardware
Database Changes
Application Changes
Summary Tables

Just give the data to the accelerator; it's ready to show its peak performance.

NO index to create, no index advisor needed, no index reorganization required. The accelerator's query processing engine logically scans billions of rows in milliseconds or seconds. The deep columnar data representation, query processing on compressed, innovative algorithms to exploit modern processor features voids the need for indices.

NO statistics to collect. Traditional optimizers rely on regular statistics collection to create better query plans. With the accelerator, join orders are determined automatically, star join plans are used consistently. Lack of indices simplifies the options and runtime optimization (explained later in the paper) voids the need for statistics collection and statistics collection advisors.

NO partitioning (fragmentation) schemes to create. The data is automatically partitioned both vertically and horizontally. The queries also benefit from vertical and horizontal partition pruning (aka fragment elimination) due to cell based deep columnar storage. This eliminates planning for table partitioning schemes.

NO Tuning for each query or workload. During the installation of the accelerator, you provide the basic memory and storage configuration. Consistent plans, elimination of disk IO, fast scans and joins eliminate runtime tuning.

NO storage management. All the data is store in-memory with just a copy of in-memory image on the disk. There is noo need to plan and create storage spaces for tables, indices.

NO expensive hardware. The accelerator is designed to run on commodity hardware and to evolve with your infrastructure. The accelerator runs on Linux/Intel computers. It integrates with Informix database server on Linux/Intel, AIX/Power, HP-UX/Itanium, Solaris/Sparc.

NO database changes. The accelerator exploits the existing logical schema in your data warehouse.

NO application changes. The accelerator plugs into the Informix database server as a resource. The Informix database server knows the data marts that are stored in the accelerator and automatically routes relevant queries to the accelerator. You application or tools require no change.

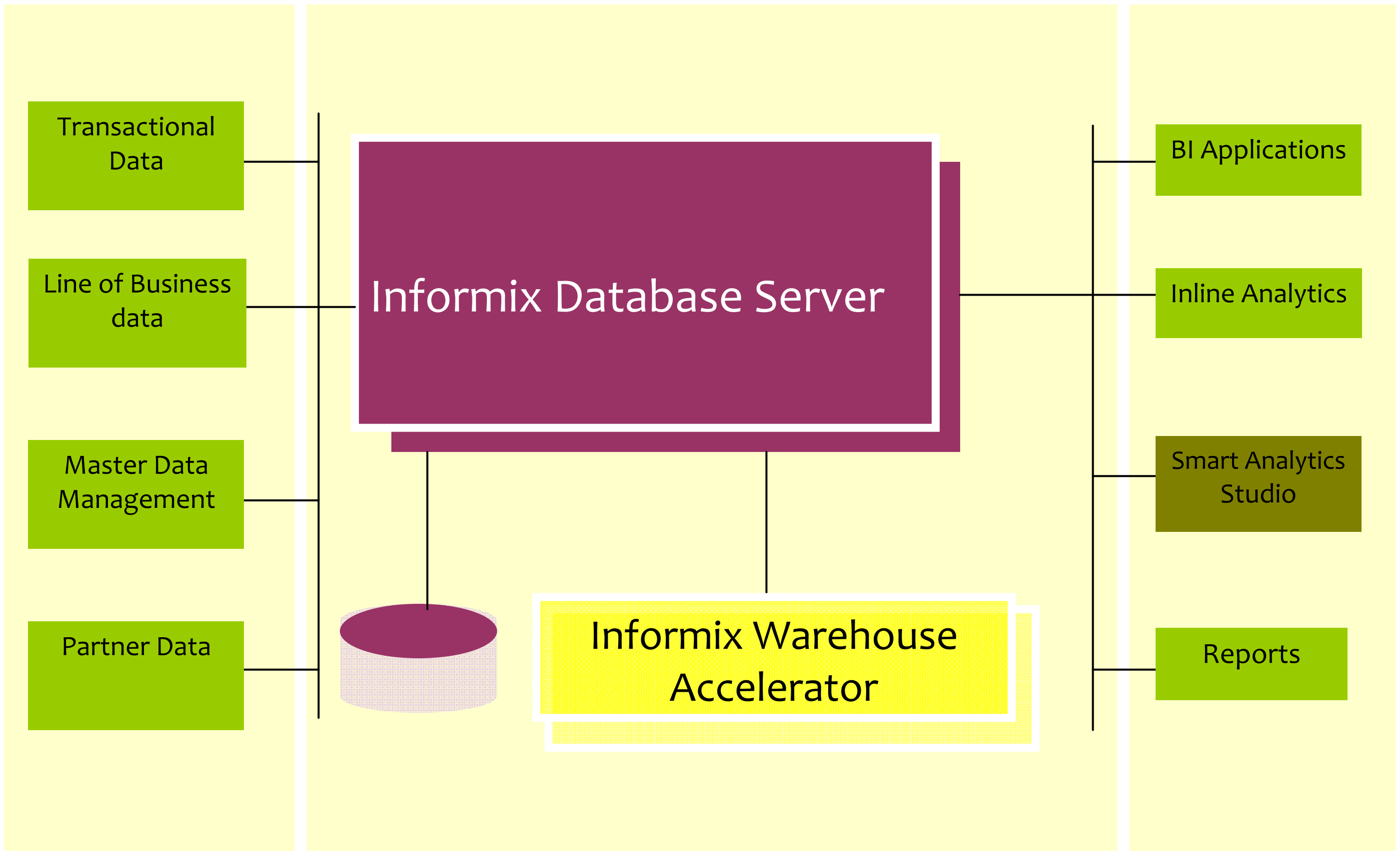
NO summary tables or materialized views to create. The accelerator table scans and joins are at least an order of magnitude faster than the traditional databases. This eliminates creation and maintenance of summary tables and the use of related advisors.

NO page size or block size configuration required. The deep columnar technology automatically determines and optimizes the in-memory cell size.

NO temp space allocation or monitoring. The intermediate result set is also compressed and takes less space.

NO query and optimizer hints for accelerated queries. The optimizer uses star join plans consistently. The accelerator query processor adjusts the join orders depending on the run time statistics.

Informix Warehouse architecture



Informix database server is a scalable and highly available database server used by tens of thousands of customers to drive mission critical transactional and analysis applications. Informix database server is a complete warehouse database server with 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 workload. Informix customers can use BI tools like Cognos, Microstrategy, etc., to analyze and improve their business.

Traditionally, data warehouse queries are called *complex* for a reason. These queries access millions and billions of rows, intermediate results are quite large, queries perform best when they are parallelized. The star join optimization technique is designed to handle this in traditional system. All this takes an experienced DBA to understand the workload and tune the system parameters and indexes to suite the workload.

Informix Warehouse Accelerator provides breakthrough warehouse query performance, eliminates the database tuning tasks and does all this without requiring any changes to your existing application and tooling environment. The remainder of this paper explains the accelerator technology, its usage and its integration with Informix.

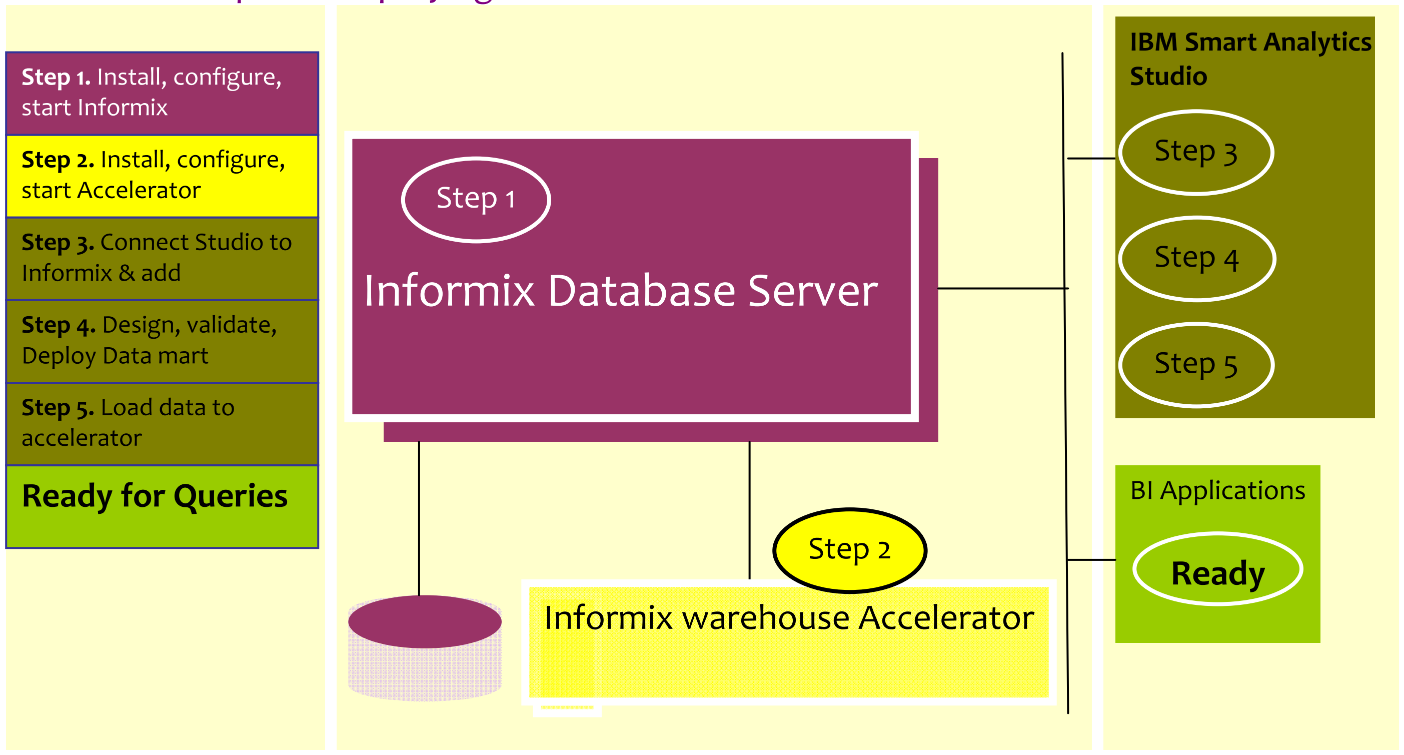
Deploying Informix Warehouse Accelerator

Informix Warehouse Accelerator always runs on a high performance Linux operating system on an Intel server. The Informix database server and the accelerator can run on the same server or different servers. You can run Informix database server on Linux/Intel, AIX/ Power, HPUX/HP Itanium, Solaris/Sparc and run the accelerator on a different Linux on Intel server.

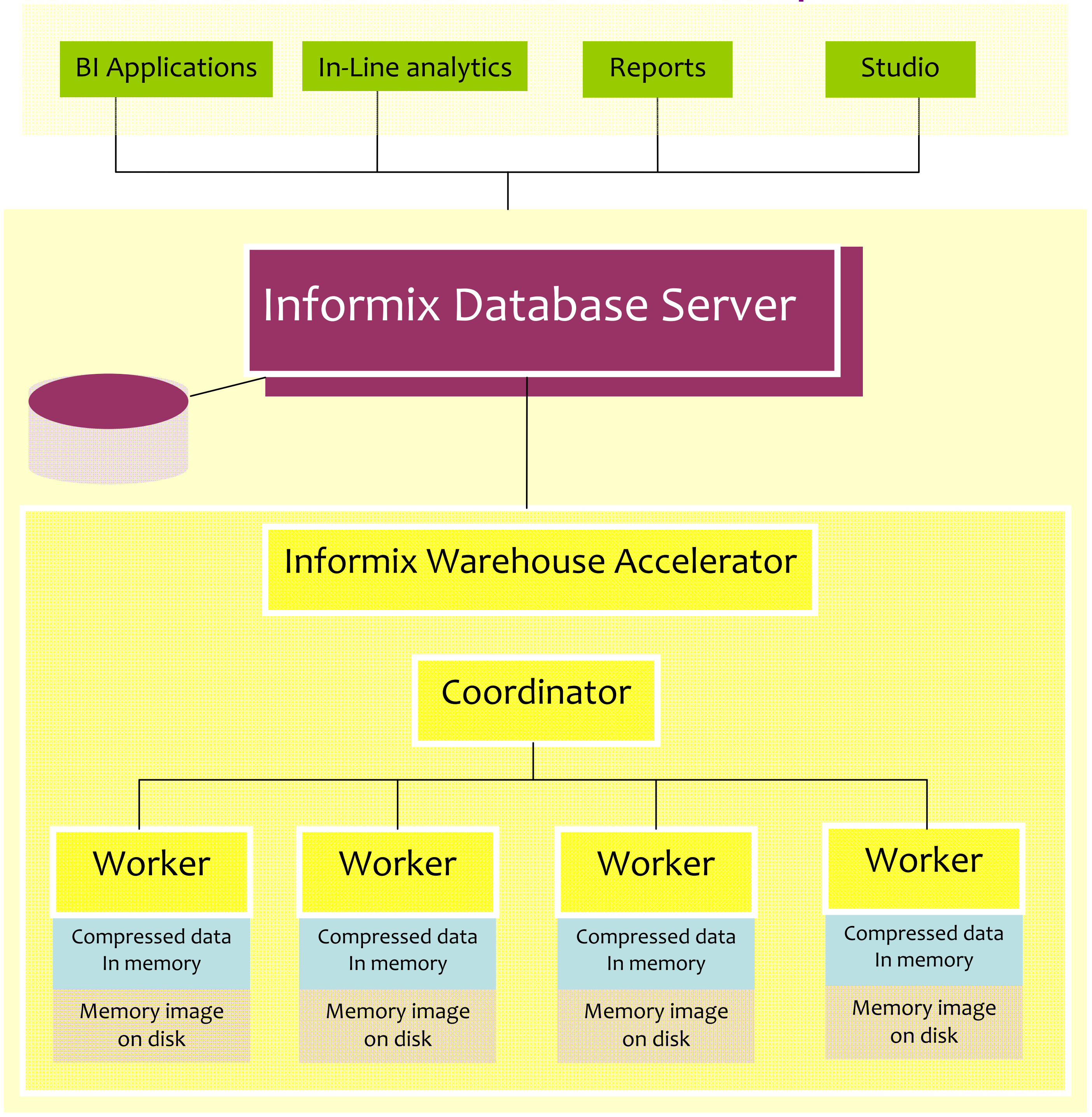
The accelerator is part of Informix Ultimate Warehouse Edition (IUWE) which includes Informix database server (Ultimate Edition) and Informix Warehouse Accelerator. The accelerator media includes the accelerator binary, and IBM Smart Analytics Optimizer Studio, a tool for configuring and managing the accelerator with Informix database server. Below is the sequence of operations to go from zero to deployment quickly. The product includes a Quick Start Guide and an Administration Guide with comprehensive documentation on installing and configuring Informix and the accelerator. You will see that accelerator requires only a few simple configuration steps: location of the file system for data backup, amount of memory, and CPU resource. Information about the configuration is discussed later in the document and in the Administration Guide.

IBM Smart Analytics Optimizer Studio is an eclipse based tool used to manage the accelerators, and define and deploy the data marts from Informix to the accelerator. The Studio comes in two versions: One version for Linux and another for Windows. You can use the Linux version on the same computer as the Informix database server or on different computer. To use the Windows version, transfer the self-extracting binary to a windows computer and install it.

Steps for Deploying Informix database server and the accelerator.


Step 1. Install, configure, start Informix
Step 2. Install, configure, start Accelerator
Step 3. Connect Studio to Informix & add
Step 4. Design, validate, Deploy Data mart
Step 5. Load data to accelerator
Ready for Queries
Step 1
Informix Database Server
Step 2
Informix warehouse Accelerator
IBM Smart Analytics Studio
Step 3
Step 4
Step 5
BI Applications
Ready

Informix Warehouse Accelerator Components



The 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 consists of coordinator and worker processes. Informix communicates with the accelerator using the coordinator processes. Coordinator and worker processes share the query processing responsibility. The configuration you specify determines number of these processes, and the amount of memory and CPU resource used.

Add the accelerator information via Studio after Informix and accelerator are running [See manual for details]. On success of this step, Informix adds accelerator connection information into its SQLHOSTS file. The SQLHOSTS file entry for the accelerator will look like this:

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

The name of the accelerator is sales_acc. Informix creates new group with that name. The hexadecimal value is the authentication code used to ensure only the Informix database communicates with the accelerator, sales_acc. The name of the coordinator is sales_acc_1. The database protocol dw is used for communication over TCP/IP. The database protocol is very close to the DRDA protocol, optimized for Informix and accelerator communication. 127.0.0.1 (TPC/IP loopback address) indicates that the accelerator is running on same computer as Informix. With a large number of workers, you will have multiple coordinators for handling failover. In those configurations, there will be an entry for each coordinator.

Coordinator has three important functions.

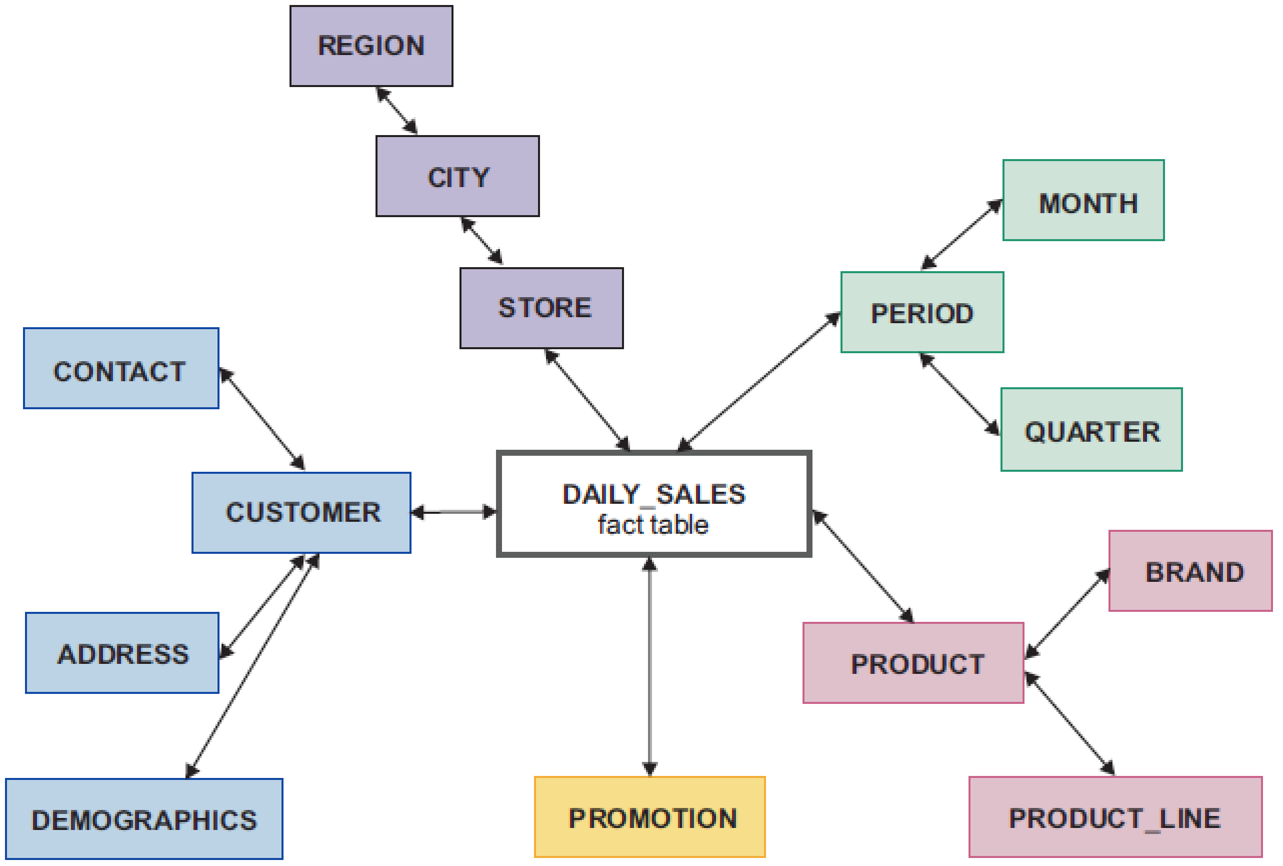
1. This is the main point of communication for Informix database server. Database server connects to the coordinator to send over the data and it's also the main contact to send the queries and retrieve result set.
2. During the data loading phase (Step 8), it distributes the data among multiple workers. The coordinator then collects the entire compression dictionary, merges it and redistributes the dictionary so everyone is using the same reference dictionary.
3. During the query processing phase, the coordinator receives the query from the database server, sends the query to each worker, gets the intermediate result set, merges the groups and then uncompresses the data, and sorts the data if necessary before sending the final results to the Informix database server.

Worker processes have two important functions.

1. In the data loading phase, each worker analyzes the incoming data using the frequency partitioning, automatically partitions the data vertically and horizontally into cells and compresses the data using Deep Columnar techniques. Once the data is compressed, the data is written to disk for recovery. We'll revisit Deep Columnar techniques later in the paper.
2. Second function of the worker processes is query processing on compressed data. Each worker maintains compressed copy of the dimension tables and portion of the fact table. Each worker returns intermediate results to the coordinator. The query processing is done 100% in-memory.

Coordinators and workers work together to parallelize execution of every query and return results quickly.

Wikipedia defines Data Mart as a subset of data warehouse, usually oriented to a specific business line or team. We'll use this definition in this paper. Your enterprise data warehouse on Informix could contain everything from sales, inventory, customer service, and market data. Out of this, you accelerate the data marts providing very high value. For example, the sales manager may want to analyze sales and inventory data to understand trends and create suitable sales incentives. So, you only need to accelerate the data marts needed for this with the sales and inventory fact tables. In the context of the accelerator, each data mart contains one or more snowflake schema. Each snowflake schema has one fact table and related dimension tables. In the following snowflake schema, DAILY_SALES is the fact table and is related to dimensions which describe the facts.



The following section explains the steps from designing the data mart to using the accelerator. These steps refer to diagram in the *Deploying the Informix Warehouse Accelerator* section.

Use the IBM Smart Analytics Optimizer Studio design and deploy data marts. A data mart defines fact and dimension tables and their relationships. The dimension tables typically contain significantly fewer rows than fact tables, e.g. product information, customer information. In some cases the dimension tables could be quite large, e.g. all California residents. Once you have identified the tables to create a data mart, you have to define the relationship between the fact and all the dimension tables.

The data mart validation step ensures that all the relationships between the tables are defined. Make sure to address any errors in this step before deploying the data mart. In the deployment step, the IBM Smart analytics Optimizer studio sends the data mart 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 view is known as Accelerated Query Table (AQT). This AQT is later used to match the queries and redirect matching queries to the accelerator.

Here's an example AQT with daily_sales as the fact table and its dimensions: product, store, customer, promotion. This is automatically created and is given here for your information. The user or DBA does not use this view. SQL from application or tools works as usual, using tables defined in the schema.

```
create view "dwa"."aqt2dbca0d9-509d-434b-9cc9-4a12c6de6b3d"
("COL16","COL17","COL18","COL19","COL20","COL21","COL22","COL23","COL24","COL25","COL2
6","COL27","COL28","COL29","COL30","COL31","COL32","COL33","COL34","COL35","COL36","CO
L37","COL38","COL39","COL40","COL41","COL42","COL43","COL44","COL45","COL46","COL47","
COL07","COL08","COL09","COL10","COL11","COL12","COL13","COL14","COL15","COL48","COL49"
,"COL50","COL51","COL52","COL53","COL54","COL55","COL56","COL57","COL58","COL59","COL6
0","COL61","COL01","COL02","COL03","COL04","COL05","COL06","COL62","COL63","COL64","CO
L65","COL66","COL67","COL68","COL69","COL70","COL71","COL72","COL73","COL74","COL75","
COL76","COL77","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 ) );
```

In the Loading 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 workers. The worker analyzes the data for frequently occurring values, and the relationships between the columns and then partitions the data vertically and horizontally. Once the partitioning is decided, the data is compressed using deep columnar process described later in this paper. Note that in this phase, the data is compressed and kept in memory with a copy on disk for persistence. No indexes are created; no summary tables or cubes are created. The data can be refreshed periodically (e.g., every night) from the Informix database server.

As soon as loading is complete, the data mart in the accelerator is ready for use. Just give the data to the accelerator and start experiencing the peak performance right away. Informix warehouse accelerator includes command line utility to design, validate, deploy and load. This is very useful to write scripts to automate this process.

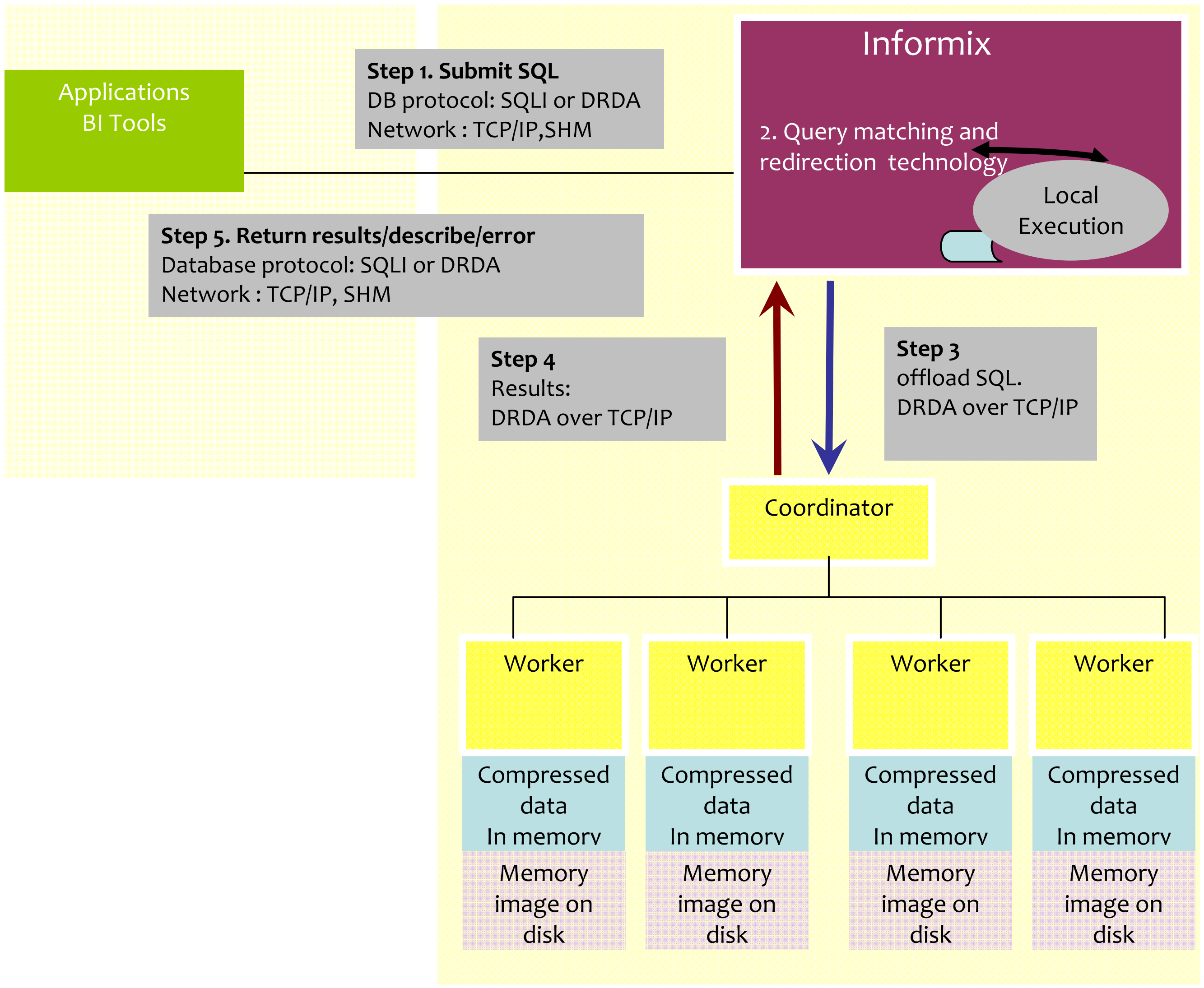
The Queries on data typically involves joining the fact table with one or more dimensions and then looking for specific patterns within the data. In the example below, the web_sales fact table is joined with 4 other dimensions. If you have created and deployed a data mart that includes these tables, Informix will match the query to a specific data mart definition view (Accelerated Query Table) and then send the query to the accelerator. This works just like a distributed query from one Informix database server to another server. The result comes back over the same connection and is sent back to the client application. For the client application, this is transparent, except that the client gets the results back much faster.

```
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;
```

The accelerator requires the queries to join fact table with zero or more dimensions and join each table using the join keys specified when defining the data mart relationship. It also supports inner joins and left joins with fact table on the dominant side. See the Administration Guide for details about query qualification. The accelerator runs each query on a first come first serve basis without interruption. All the data and intermediate results are in memory. Each worker has copy of the dimension to join with. Each join thread tries to cache the hash table in L2 cache to optimize memory access. The worker processes scan and join independently with little data exchange and synchronization with other workers. It's typical for each query to complete in a few seconds compared to minutes and hours in a traditional system. Hence, the accelerator queues the query requests and runs one query after another.

Query Flow with the Accelerator



Notes on Configuration

During the installation of Informix Warehouse Accelerator, you'll be asked to configure the number of nodes (coordinators, workers), memory for workers, and memory configuration for the coordinator node. The number of coordinators and workers will be automatically determined. E.g. defining 5 nodes will automatically create 1 coordinator and 4 worker nodes. Details are described in the Administration Guide. Consider a configuration with 4 workers and a coordinator. Let's say, you deploy a data mart with a tables sales fact table and customer, store, time dimensions. The dimension tables customer, store, and time are compressed and kept in private memory of each worker. So, there will be 4 copies of the dimension tables; the rows of the fact table sales will be evenly divided among the 4 workers. Each worker will maintain the dimension tables customer, store, time and then 25% of rows in fact table – sales in this case—in main memory.

The data transfer rate typically increases as you increase the number of workers, assuming there's enough CPU capacity. Larger number of workers does help query processing speed, although less dramatically. The effect of the number of workers on the query depends on the query as well.

Given this, how much memory should you allocate for each worker? How memory is need by the system?

Generally, we have seen a 3:1 compression ratio for the uncompressed Informix data to the compressed Informix Warehouse Accelerator data in memory. If the sum of fact table sales and the dimension tables customer, store, time is about 100GB and most of that is consumed by sales table, you'll need about 33 GB of memory to store the data. You can easily find out the size of the tables by using the Open Admin Tool (OAT) or directly querying the catalogs.

Each worker needs sufficient runtime memory to store intermediate results at runtime. Workers dynamically allocate and release the memory required for query processing dynamically. Planning here is similar to temp space planning in Informix. How much intermediate results, sorting do you expect from the workload? How related is your data and how many groups would be there in each category? Although it is difficult to tell this precisely, we have found having another 1/5th to 1/3rd of the data size is usually sufficient.

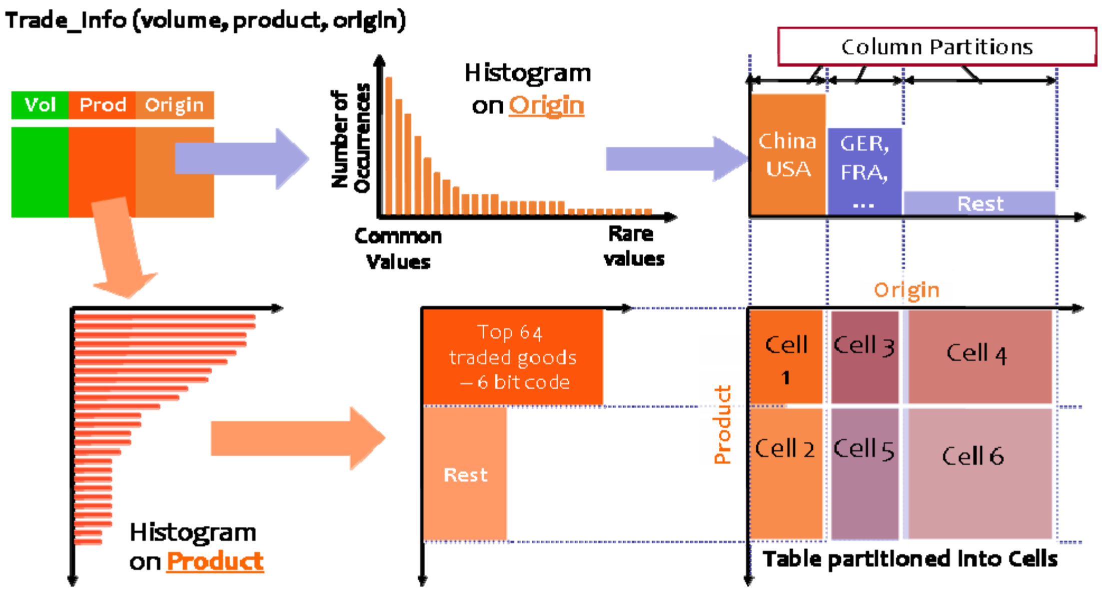
The final stage of accelerator query processing is done at the coordinator. The coordinator needs memory for merging, decompressing, and sorting the result set. Again, the memory required here depends on the size of the result set you expect. Remember, when you issue the query with FIRST clause, "e.g. SELECT FIRST 1000… ORDER BY sum(sales.amount)", the coordinator will have to sort all the data to get the first 1000. Of course, once the coordinator creates the first 1000 groups, it can replace or reject the new groups.

Peak Performance

The extreme performance with the accelerator is the cumulative result of technologies developed by IBM research and development. This paper provides an overview of the technologies and techniques used in the accelerator to improve the performance and eliminate tuning and maintenance tasks. Papers listed in the reference section have further details of underlying theory and techniques. These papers are published in well known, peer reviewed journals or conferences. IBM has filed patent applications on these techniques.

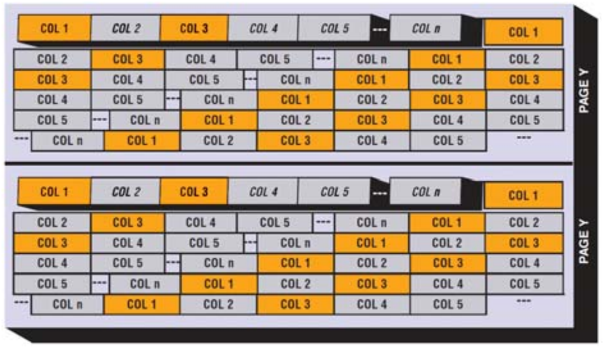
Deep Columnar technology goes beyond the traditional columnar storage. Extreme compression and query processing on compressed data eliminates disk IO during query processing and enables large in-memory warehouse. Exploitation of multi-core architectures and SIMD technology gets you incredible speed without indexes or summary tables. Let's look into each technique now.

Frequency Partitioning



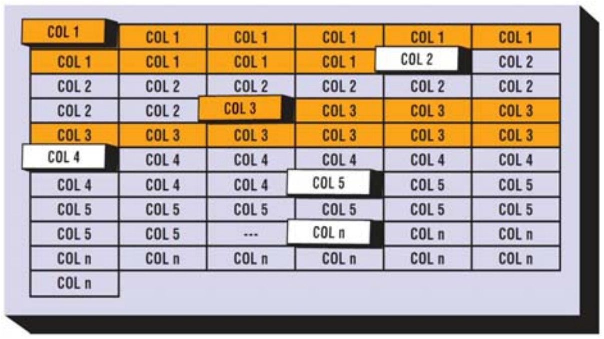
Each table in the data mart is analyzed for frequently occurring values in columns and related column groups to determine the most optimal columns to combine to form a tuplet. In the example above, the two columns product and origin are correlated and hence combined to form a tuplet. A tuplet is a fraction of a tuple. A tuple is the complete row. The benefit of Huffman encoding is that most frequently occurring values are encoded with least number of bits. In the figure above, the top 64 product values are combined with the most frequently occurring values in origin (USA, China) to form the smaller cell1 using Huffman encoding. This technique increases the compression efficiency and can be used to evaluate both equality and range predicates. Since the query processing is done on compressed data, less bits translate to higher speed.

Columnar Storage in accelerator

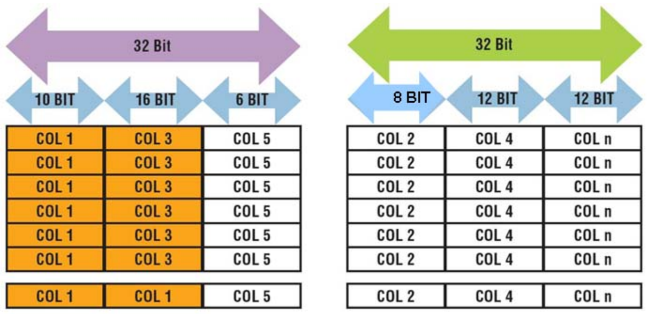


Traditional row-wise databases store a complete row (aka tuple) in a page followed by another row. The design optimizes the row IO efficiency, assumes the query is interested in majority of the column values. If the data is compressed in storage, the row is uncompressed for every fetch. This is efficient for transactional workloads accessing few rows per query.

In analytical queries, you typically access millions or billions of rows, but each query analyzes the relationship between subset of columns in the fact table. For example, you could be interested in total sales of the items per location in 2010. This query only needs to access three of the columns in the fact table: item, sales amount and location, and join with dimensions. In this case, accessing and uncompressing every row is inefficient.



Columnar database stores every column values together. Every time you insert or load the data, each row is unzipped to separate the column values and every time you access the row, the column values are fetched separately and then zipped to form the row. Because the column values are stored together, you can get better compression. In the example above, we saw that analytical queries are typically interested in subset of the rows. For this query, in columnar storage databases, you can only fetch the pages storing item, sales and location. For the queries accessing large subset of the rows and doing sequential scans, columnar storage improves the efficiency.



The Informix Warehouse Accelerator stores data in columns groups, or vertical partitions of the table, called “banks”. The fraction of a row (or tuple) that fits in each bank is called a “tuplet”. The assignment of columns to banks is cell-specific, because the column lengths vary from cell to cell.

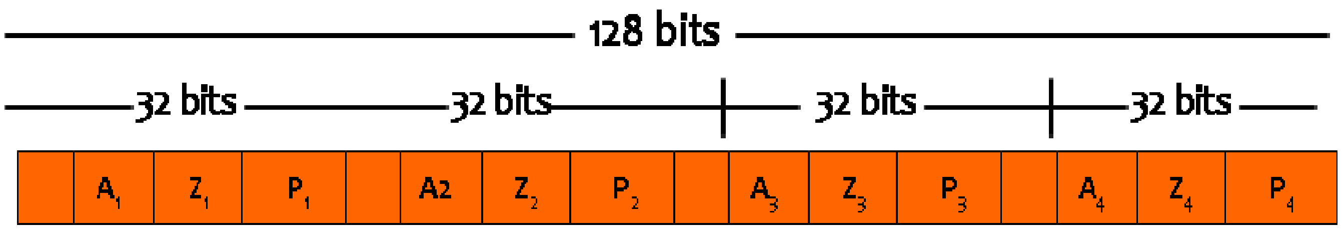
The assignment uses a bin-packing algorithm that is based upon whether the column fits in a bank, whose width is some fraction of a word, rather than usage in a workload. Also, scans need only access the banks that contain columns referenced in any given query, saving scanning those banks with no columns referenced in the query. This projection is similar to the way pure column stores minimize disk I/Os. The accelerator stores all the data in memory, so there is no disk IO. Even though, Informix Warehouse accelerator does not have disk IO, this technique minimizes the amount of memory to scan and saves considerable CPU cycles.

Single Instruction Multiple Data parallelism

Consider a query:

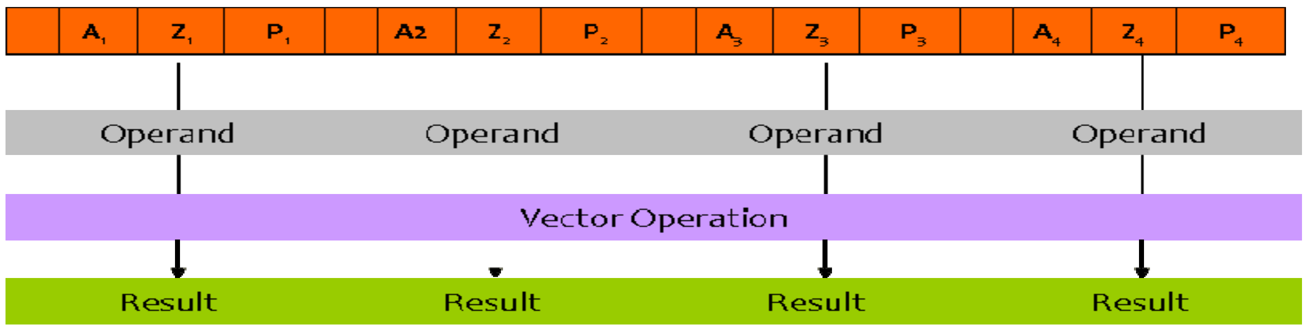
```
SELECT SUM(s.amount) FROM sales AS s WHERE s.prid = 100 GROUP BY s.zip;
```

If the columns amount (A), prid (P), zip(Z) are from the same bank, multiple of these values can be loaded into 128 CPU register at the same time. In this case, we have 12 column values at a time.



SIMD instructions on Intel Xeon processors operate on 128-bit registers. The compression technique in the accelerator typically requires few bits for each column and hence can load many fields in each 128-bit register. We can load multiple values and apply predicates on all columns simultaneously. During query processing, this overloaded operation is happening on all the allocated cores resulting in extreme parallelism for the query.

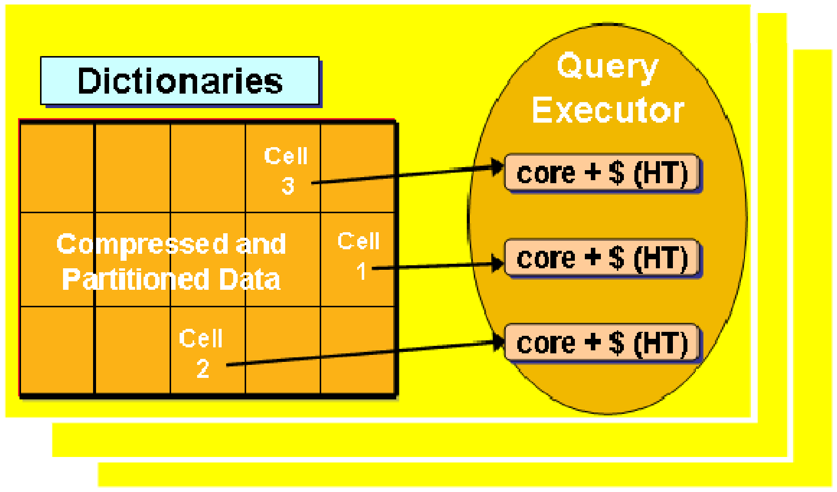
Once you have that, you can operate on all the 12 values simultaneously with a single CPU instruction resulting in significant performance gain.



Query Processing

So far, we have looked at how the data is encoded and stored, and how the processing is done at micro level. Let's explore the query processing at macro level.

Efficient Scanning provides the foundation for query processing. We know the data is organized into tuplets, banks and cells. For scanning, Cell is the unit of processing. The accelerator dynamically spawns appropriate threads to utilize configured CPU resource and assigns cell to each core. This scan operates on compressed data, using SIMD instructions, 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 allowing caching of the hash table in the processor's L2 cache and a quick look up.



This enables very quick GROUP BY on compressed data. Joins between two densely encoded hash tables can result in sparse set of values. The accelerator detects these situations and will switch over to linear probing dynamically. Combination of these techniques enables query processing on compressed data.

We know the data mart is a star or snowflake schema containing a large fact table and number of dimension tables. The queries will join the fact tables with number of dimension tables using the join predicates between fact to dimensions and then dimensions to other dimensions. For each query, each worker creates a snowflake model for the tables. It then starts at the outer most edge of the branch and works its way into the fact table. Each snowflake branch is processed and its result acts as a dimensional input to 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 joined with fact table (or the dimension table acting as a fact at the snowflake branch) to form the next level of aggregations and relations. This process is applied recursively till the complete join is processed at each worker. Since each worker has copy of dimension table, there's little data exchanged between the workers during query processing and each proceed with maximum performance possible. Worker then sends the intermediate result set to the coordinator.

Since there is only one representation of the data – compressed columnar table data – accelerator follows the same code path for each table every time. All the efficiencies of cell, block elimination for the query comes from the compression encoding. Basically the accelerator does not have indexes, summary tables to worry about. It pretty much follows the same process every time. Because of this, the accelerator performance is also consistent.

The Coordinator node gets the results from each worker, merges the intermediate result into appropriate groups, decompresses the data, and then executes ORDER BY and HAVING instructions before sending the data to the Informix Database server over DRDA protocol. Informix database server then routes the data to the application program.

Conclusion

Informix Warehouse Accelerators' innovative approaches to complex query processing improves productivity of your business by providing answers faster without increasing your work or breaking your budget. Because it's tightly integrated with Informix database server, you can exploit the environment to divide the load between Informix and accelerator as necessary.

Faster response time means quicker answers, quicker insights and a business that can adjust faster. You can plan to accelerate the high value part of your warehouse and dynamically evolve the infrastructure to suit business needs.

Further information

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

Additional reading

- **VLDB 2008:** *"Main-Memory Scan Sharing for Multi-core CPUs",* Lin Qiao, Vijayshankar Raman, Frederick Reiss, Peter Haas, Guy Lohman
- **VLDB 2008:** *"Row-Wise Parallel Predicate Evaluation",* Ryan Johnson, Vijayshankar Raman, Richard Sidle, Garret Swart
- **VLDB 2006**: *"How to wring a table Dry: Entropy Compression of Relations and Querying Compressed Relations"*, Vijayshankar Raman, Garret Swart
- **SIGMOD 2007**: *"How to barter bits for chronons: compression and bandwidth trade offs for database scans",* Allison L. Holloway, Vijayshankar Raman, Garret Swart, David J. DeWitt
- **ICDE 2008**: *"Constant-time Query Processing"*, Vijayshankar Raman, Garret Swart, Lin Qiao, Frederick Reiss, Vijay Dialani, Donald Kossmann, Inderpal Narang, Richard Sidle
- **BTW 2009**: Bringing BLINK Closer to the Full Power of SQL. Knut Stolze, Vijayshankar Raman, Richard Sidle, Oliver Draese

Acknowledgements

This product was developed by collaboration of IBM Almaden Research, IBM Böblingen lab and IBM Informix team. Thanks to the Informix team for reviewing and improving this paper.
