Key SQL Function in DB2 9 for z/OS & Beyond

James Guo
IBM, Silicon Valley Lab

August 2, 2010   4:30 pm – 5:30 pm
Session Number 7967

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

Agenda

- SQL Enhancements in DB2 9 for z/OS
- Selected SQL Overview in DB2 10 for z/OS
  - Hash Access
  - Temporal Data
  - IBM ISAO

SQL Enhancements in DB2 9 for z/OS

Key SQL Functions in V9

- New data types: BIGINT, BINARY, VARBINARY, DECFLOAT
- Instead of Trigger
- Merge
- SELECT FROM MERGE/UPDATE/DELETE
- ORDER BY and FETCH FIRST in subselect
- TRUNCATE
- INTERSECT/EXCEPT
- Index on Expression

Key SQL Functions in V9 - cont'd

- Sparse Index and In Memory Data Caching
- Dynamic Index Anding and Star Schema
- OLAP specification -- RANK, DENSE_RANK, ROW_NUMBER
- RENAME INDEX/COLUMN
- REOPT (AUTO)

New data types: BIGINT, BINARY, VARBINARY, DECFLOAT

- **BIGINT** - big integer.
  - Big integer is a binary integer with a precision of 63 bits. The range of big integers is [-9223372036854775808, 9223372036854775807
- **BINARY** – fixed-length binary string.
  - Fixed-length binary string is in a range of [1,255]. The padding with hexadecimal zeros (X'00'). Not associated with any CCSID
- **VARBINARY** – varying-length binary string.
  - Varying-length binary string is in a range of [1,32704]. No padding is performed. Not associated with any CCSID
- **DECFLOAT** – Decimal float.
  - DECFLOAT(16) = decimal64 format (8 bytes)
  - DECFLOAT(34) = decimal128 format (16 bytes)

DECFLOAT

- Both IEEE and hexadecimal floating point numbers can only approximate common decimal numbers. But DFP can represent decimal number exactly.
- DFP can represent much bigger and smaller number than DECIMAL.

Description	Limit
Smallest DECFLOAT(16) Values	$-9.999999999999999 \times 10^{384}$
Largest DECFLOAT(16) Value	$9.999999999999999 \times 10^{384}$
Smallest positive DECFLOAT(16) Value	$1.000000000000000 \times 10^{-383}$
Largest negative DECFLOAT(16) value	$-1.000000000000000 \times 10^{-383}$
Smallest DECFLOAT(34) Value	$-9.999999999999999999999999999999999 \times 10^{6144}$
Largest DECFLOAT(34) Value	$9.999999999999999999999999999999999 \times 10^{6144}$
Smallest positive DECFLOAT(34) Value	$1.000000000000000000000000000000000 \times 10^{-6143}$
Largest negative DECFLOAT(34) Value	$-1.000000000000000000000000000000000 \times 10^{-6143}$

Instead of Trigger

- A new type of trigger (~ BEFORE, AFTER triggers)
- Defined on VIEWs
  - provides an extension to the updatability of views
  - requested update operation against the view gets replaced by the trigger logic
  - application still believes all operations are performed against the view
  - applicable even for updatable views

Instead of Trigger

```
CREATE TABLE WEATHER (CITY VARCHAR(25), TEMPF DECIMAL(5,2));
CREATE VIEW CELCIUS_WEATHER_V (CITY, TEMPC) AS
      SELECT CITY, (TEMPF-32)*5.00/9.00 FROM WEATHER

CREATE TRIGGER CW_INSERT INSTEAD OF INSERT ON
        CELCIUS_WEATHER_V
REFERENCING NEW AS NEWCW DEFAULTS NULL
FOR EACH ROW MODE DB2SQL
    INSERT INTO WEATHER VALUES (NEWCW.CITY,
                                          9.00/5.00*NEWCW.TEMPC+32)

CREATE TRIGGER CW_UPDATE INSTEAD OF UPDATE ON
         CELCIUS_WEATHER_V
REFERENCING NEW AS NEWCW OLD AS OLDCW DEFAULTS NULL
FOR EACH ROW MODE DB2SQL
    UPDATE WEATHER AS W
      SET W.CITY = NEWCW.CITY,
            W.TEMPF = 9.00/5.00*NEWCW.TEMPC+32
      WHERE W.CITY = OLDCW.CITY
```

Merge

- Combine **Update** and **Insert** operation to a target table or view, from a input source of a list of host-variable-arrays modeled as a source table
  - When source rows **match** to target, **Update** target rows from source
  - When source rows **do not match** to target, **Insert** source rows into target
  - Update/Insert triggers will be fired

```
MERGE INTO account AS T
  USING VALUES (:hv_id, :hv_amt) FOR 5 ROWS AS S(id,amt)
ON T.id = S.id
  WHEN MATCHED THEN
    UPDATE SET balance = T.balance + S.amt
  WHEN NOT MATCHED THEN
    INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

SELECT FROM MERGE/UPDATE/DELETE

- V8 - The INSERT statement was allowed in the FROM clause
- V9 - A searched UPDATE/DELETE is now allowed in the FROM clause

Delete employees at level 'Contractor' and return the total amount of salary:

```
SELECT SUM(Salary)
FROM OLD TABLE
(DELETE FROM Employee WHERE Level =
'Contractor');
```

SELECT FROM MERGE/UPDATE/DELETE

Update salaries of employees at level 'Associate' and return the new salary:

```
SELECT Name, Salary
FROM FINAL TABLE
(UPDATE Employee   SET Salary = Salary *1.1
  WHERE Level = 'Associate');
```

Update salaries of employees at level 'Associate' and return the old salary:

```
SELECT Name, Salary
FROM OLD TABLE
(UPDATE Employee   SET Salary = Salary *1.1
  WHERE Level = 'Associate');
```

ORDER BY and FETCH FIRST in subselect

- ORDER BY clause can be specified in subselect or fullselect
- FETCH FIRST n ROWS ONLY clause can be specified in subselect or fullselect
- ORDER OF table-designator extension to the ORDER BY clause

```
(SELECT * FROM T1
 ORDER BY C1)
UNION
(SELECT * FROM T2
 ORDER BY C2
 FETCH FIRST 2 ROWS)
```

```
(SELECT * FROM T1
 ORDER BY C1)
UNION
SELECT * FROM T2
ORDER BY C2
FETCH FIRST 2 ROWS
```

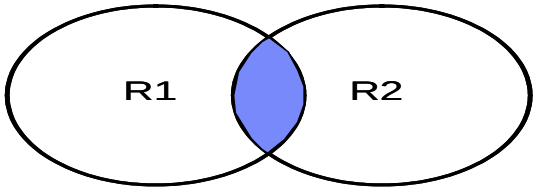
TRUNCATE Table

- A fast way to empty a table
- **DELETE Triggers are ignored**
- Indexes, LOB, XML Tablespaces are also deleted
- X lock on the target table, Mass-delete

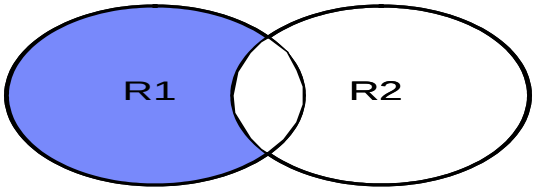
```
TRUNCATE <TABLE> TABLE-NAME

  < DROP STORAGE | REUSE STORAGE>

  < RESTRICT WHEN DELETE TRIGGERS | IGNORE DELETE TRIGGERS>

  < IMMEDIATE>
```

INTERSECT/EXCEPT

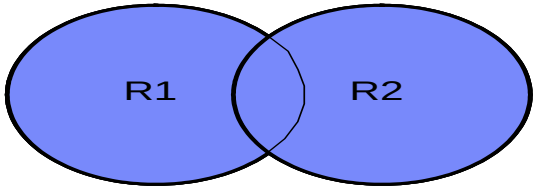
- SET operator: UNION, **INTERSECT, EXCEPT**



INTERSECT



EXCEPT
(Difference)



UNION

```
SELECT c11, c12, …, c1n FROM T1 <set-op>
SELECT c21, c22, …, c2m FROM T2
```

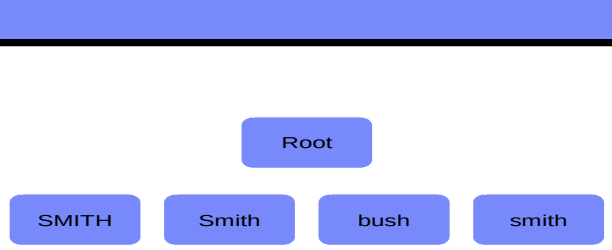
INTERSECT/EXCEPT

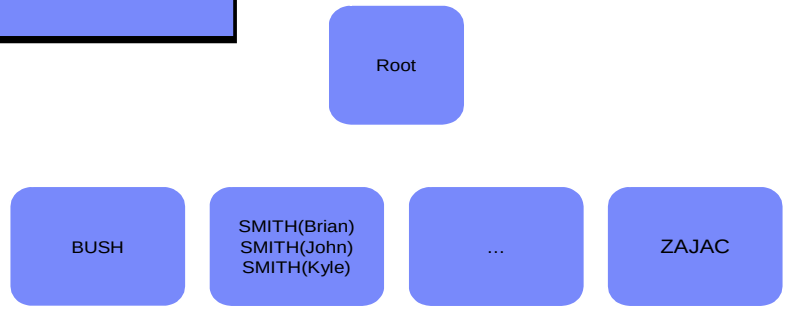
R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTERSECT ALL	INTERSECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

Index on Expression

- Create index on result of Expression
- Enhance Query Performance
- If we want to search for customers whose Upper(Lastname) = ‘SMITH’

```
CREATE INDEX IX_LastName ON CUSTOMER
 ( UPPER (Lastname), CUSTOMER_ID);

SELECT * FROM EMP
    WHERE UPPER (Lastname) = 'SMITH';
```





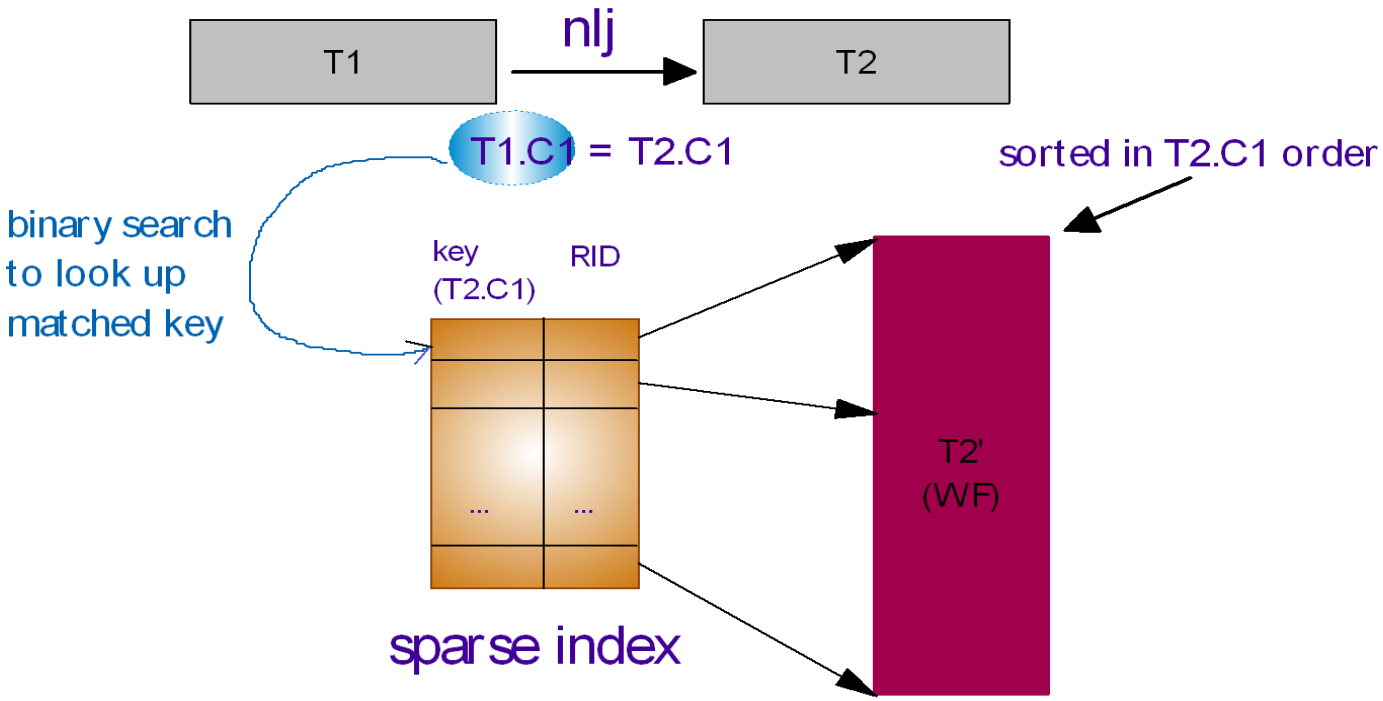
Index on Expression - 2

- CREATE INDEX IX1 ON T1 (HEX(c1), BINARY(LTRIM(c2)));
- CREATE INDEX IX2 ON T2 (SUBSTR(c2, 1, 23), CONCAT(c2, c3));
- CREATE INDEX IX3 ON T2 (salary, bonus/salary, bonus+salary);
- CREATE INDEX IX4 ON T1 (DAYOFYEAR(endship) - DAYOFYEAR(startship));
- CREATE INDEX IX5 ON T2 (GRAPHIC(c3));
- CREATE INDEX IX6 ON T1 (VARCHAR(INSERT(vchar30,1,0,''),20));
- CREATE INDEX IX7 ON T1 (posstr(lvcharx2, '7.2E+02')) ;
- CREATE INDEX IX8 ON T1 (MIDNIGHT_SECONDS(birthday));

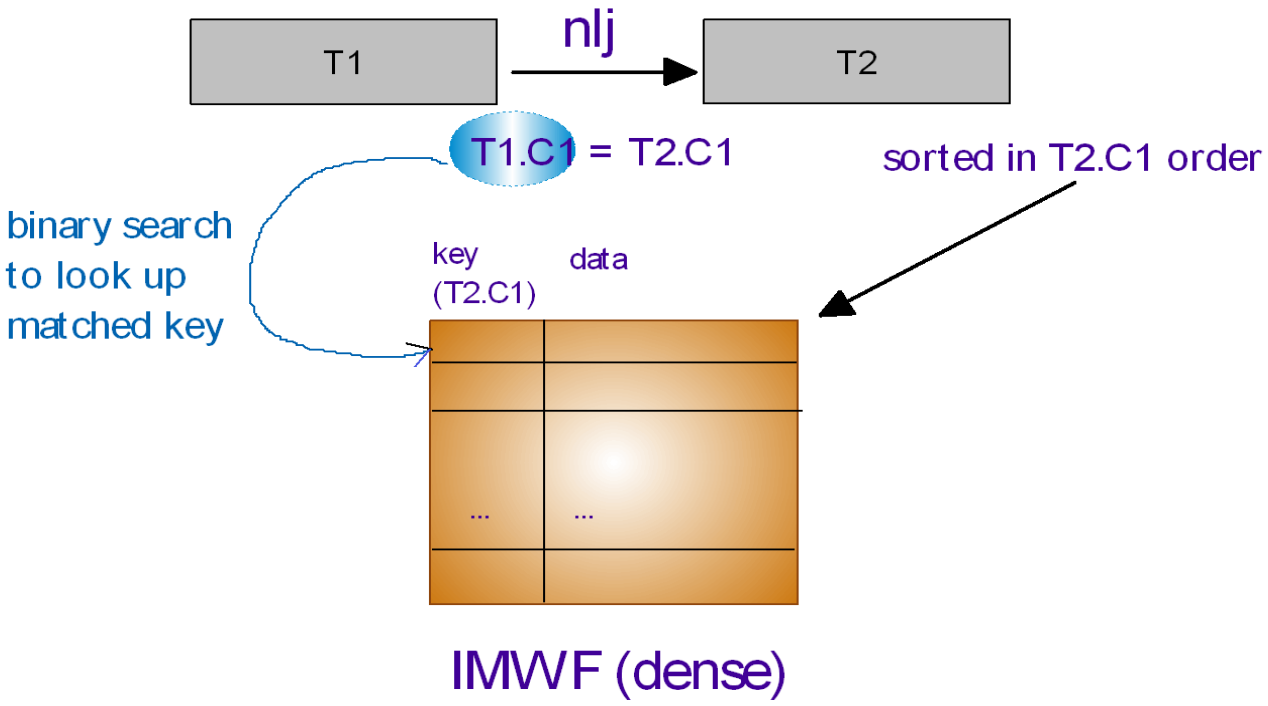
Sparse Index and In Memory Data Caching

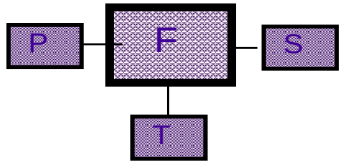
- Tables which are lack of appropriate index or enough statistics can benefit from sparse index / IMWF
- Sparse index / IMWF is used for **Nested Loop Join**
- Whether use sparse index or IMWF is a runtime decision according to storage availability

How does sparse index work?



How does IMWF work?

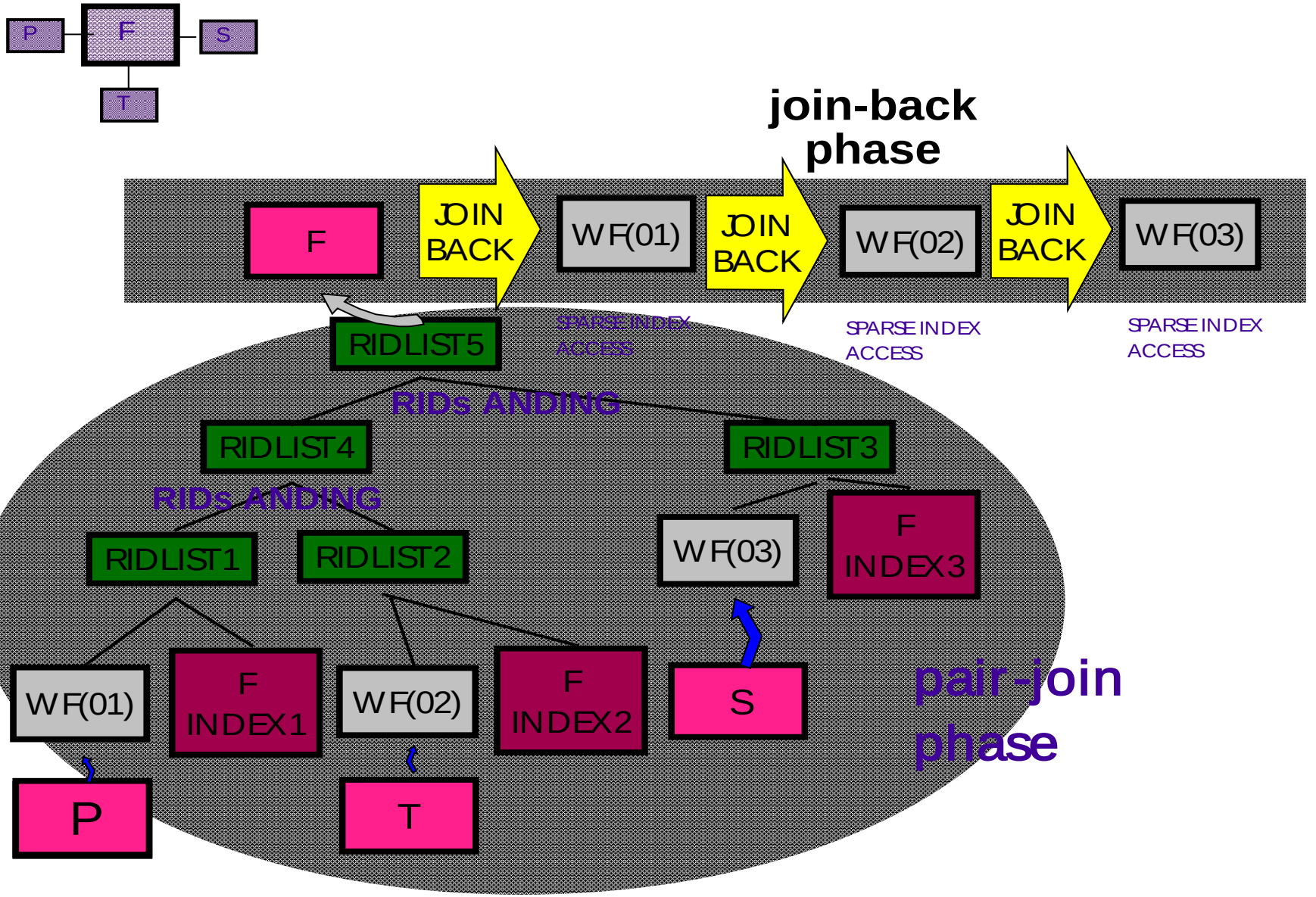




Dynamic Index Anding and Star Schema

- Each dimension table joins with Fact table separately
- The join result is Rid lists of Fact table
- Using Rids Merge (ANDing) to generate final set of Fact table Rid lists

```
SELECT PRODNAME, SUM(SALES), ...
 FROM F, PROD P, CUST C, TIME T, STORES S
WHERE F.PID = P.ID
     AND F.TID = T.ID
     AND F.SID = S.ID
 AND T.MONTH IN ('JAN', 'FEB')
 AND S.LOCATION IN ('SAN JOSE', 'DALLAS')
 AND P.TYPE IN ('FOOD', 'SODA')
GROUP BY ...
```


P
F
S
T
join-back phase
F
JOIN BACK
WF(01)
JOIN BACK
WF(02)
JOIN BACK
WF(03)
RIDLIST5
SPARSE INDEX ACCESS
SPARSE INDEX ACCESS
SPARSE INDEX ACCESS
RIDs ANDING
RIDLIST4
RIDLIST3
RIDs ANDING
RIDLIST1
RIDLIST2
WF(03)
F INDEX3
WF(01)
F INDEX1
WF(02)
F INDEX2
S
P
T
pair-join phase

OLAP specification -- RANK, DENSE_RANK, ROW_NUMBER

- RANK() OVER Window ----> OLAP Function
  - PARTITION BY sh.territory --- row should be assigned to partition according to territory
  - ORDER BY sh.sales --- row sorted in the order of sales amount within each partition
- Apply after Join, Predicates, Group By, Having
- A new class of aggregate functions
  - Rank
  - DENSERANK
  - ROWNUMBER

```
SELECT  sh.territory, sh.sales,
        Rank() over (PARTITION BY sh.territory
                     ORDER BY sh.sales desc) as rank
    FROM sales_history;
```

OLAP specification -- RANK, DENSE_RANK, ROW_NUMBER

```
SELECT EMPNUM, DEPT, SALARY,
    RANK()      OVER (ORDER BY SALARY DESC) as RANK,
    DENSERANK() OVER (ORDER BY SALARY DESC)as DENSERANK,
    ROWNUMBER() OVER (ORDER BY SALARY DESC) as ROWNUM
FROM EMPLOYEE;
```

EMPNUM	DEPT	SALARY	RANK	DENSERANK	ROWNUM
3	-	84000	1	1	1
8	3	79000	2	2	2
6	1	78000	3	3	3
2	1	75000	4	4	4
7	1	75000	4	4	5
12	3	75000	4	4	6
10	3	55000	7	5	7
11	1	53000	8	6	8

RENAME INDEX/COLUMN

- Without having to drop and recreate the object
- Rename Column

```
ALTER TABLE tb1
   RENAME COLUMN   old_columnname
     TO    new_columnname
```

- Rename Index

```
RENAME  INDEX/TABLE
      old_name  TO  new_name
```

REOPT(AUTO)

- We currently have REOPT(NONE), REOPT(ONCE) and REOPT(ALWAYS) for a dynamic query (with parameter marker).
  - **REOPT(NONE)**
    - Generate access path based on default filter factor at BIND time
    - Cache the access path
  - **REOPT(ALWAYS)**
    - Generate access path based on real parameter marker value at each execution
    - No statement cache
  - **REOPT(ONCE)**
    - Generate access path based on real parameter marker value at the first excution
    - Cache the access path

REOPT(AUTO)

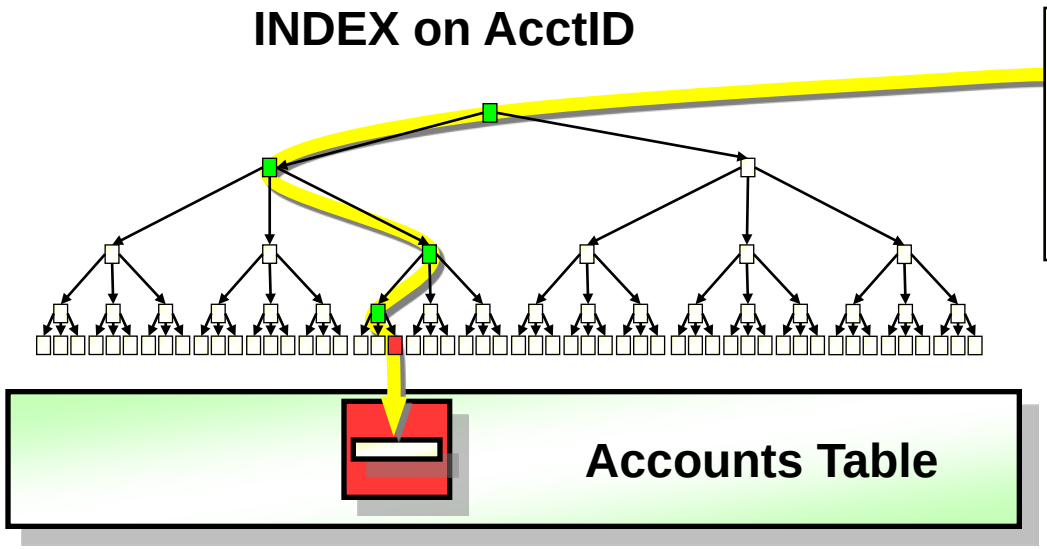
- **REOPT(AUTO)**
  - Generate access path based on real parameter marker value at the first execution.
  - Cache the access path
  - At each execution, DB2 will check if the cache plan is still a good one based on the real value.
  - A new prepare might be automatically invoked
  - Cache the newly generated access path
  - If statement cache is turned off, no reopt occurs

DB2 10 Query Enhancements

- CPU time reductions for queries, batch, & transactions
- SQL enhancements: Moving Sum, Moving Average, temporal, timestamp, implicit cast, SQL PL, …
- pureXML improvements
- Access improvements: Index include columns, hash, index list prefetch, workfile spanned records, …
- Optimization techniques
  - Remove parallelism restrictions and more even parallel distribution. Increased zIIP usage.
  - In-memory techniques for faster query performance
  - Access path stability and control
- Analysis: instrumentation, Data Studio & Optim Query Tuner
- Advanced query acceleration techniques
  - IBM Smart Analytics Optimizer

DB2 10 Hash Access Overview

Index to Data Access Path



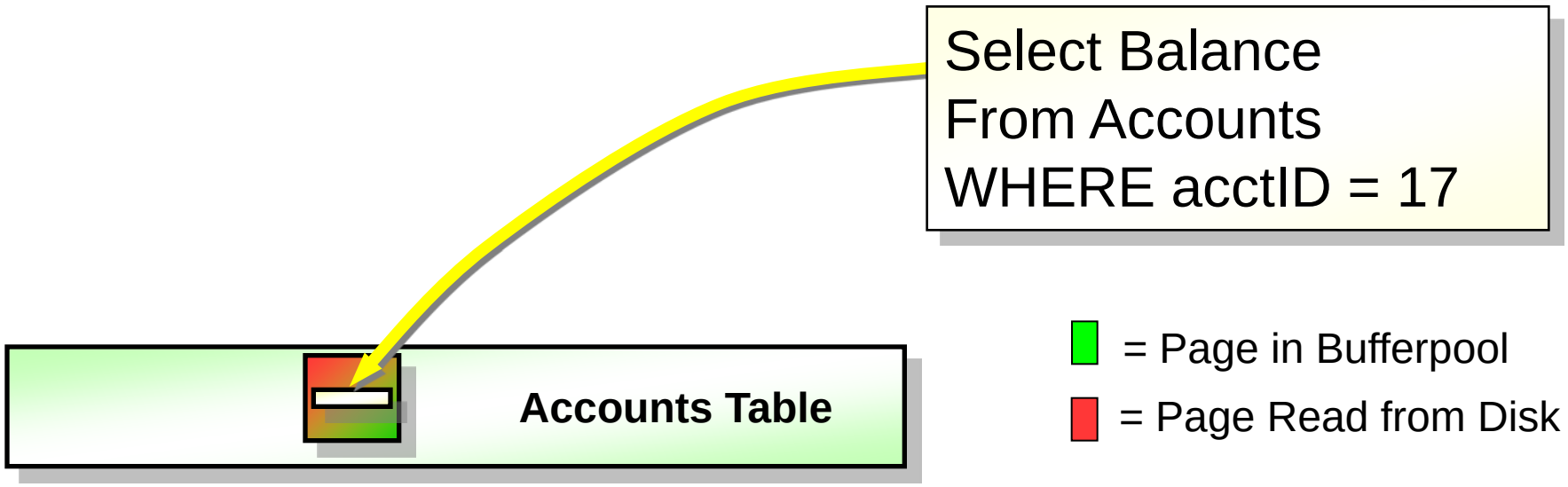
Select Balance
From Accounts
WHERE acctID = 17

= Page in Bufferpool

= Page Read from Disk

- Traverse down Index Tree
  - Typically non-leaf portion of tree in the bufferpool
  - Leaf Portion of the tree requires I/O
  - Requires searching pages at each level of the index
- Access the Data Page
  - Typically requires another I/O
- For a 5 Level Index (6 GETP/RELPs, 2 I/O's, and 5 index page searches)

Hash Access



- Hash Access provides the ability to directly locate a row in a table without having to use an index
- Single GETP/RELP in most cases
- 1 Synch I/Os in common case
  - 0 If In Memory Table
- Greatly reduced Search CPU expense

Using Hash Access

- **Hash Access is good for tables:**
  - With **a unique key**
  - Queried by applications (such as OLTP) needing **single row access via the unique key**
  - With **known approximate size**
  - Tables where **clustering order of data** is not important
    - *Applies to any kind of range processing*
  - When 20%-50% extra space is not an issue
- **Hash Access Path will be chosen when:**
  - The SELECT statement includes an **equal predicate on all hash key columns (or IN-List)**
  - DB2 APS determines via hash availability and the specific query **if hash access is suitable**

- **Follow-up**
  - Run BIND with EXPLAIN option and query the PLAN_TABLE to **check access path**

Examples of Select Statements Using Hash Access

- Simple hash access
  - SELECT * FROM T1 WHERE HASHCOL = :HV
- List prefetch/multi-index access with hash access
  - SELECT * FROM T1 WHERE (HASHCOL = :HV1 OR INDEXCOL = :HV2)
- No query parallelism with hash access
- No hash access with star join or hybrid join

Example of Application for Hash Access

- Typical Online Banking Application
  - SELECT the Account details using account number (Unique Key)
    - *Retrieving account details and maybe account balance*
    - *So far this would be suitable for Hash access*
  - SELECT all transactions for a certain period
    - *Used to show money into and out of the account each day*
    - *This period would probably invoke Sequential processing*
    - *Either using BETWEEN or > and <*
    - *Neither of these are allowed for Hash access*

Example of a Suitable Application For Hash Access

- Car Insurance Application
  - Typical Car Insurance application where customer details are only ever accessed by Policy Number (Unique key)
    - When policy is set up a unique policy number is created
    - Policy number is used if Customer wants details of policy
    - Policy number is used if customer updates policy
    - Policy number is used when customer makes a claim
    - Policy number is used when claim is processed
  - All accesses will use = predicate in t[illegible] SQ[illegible]

Hashing Summary

- Provides fast, direct location of most rows
  - Reduces I/O and CPU in some cases
  - Can replace an existing Primary or Unique Key Index
    - Faster Insertion/Deletion
- Size of Fixed Size Hash Area is important
  - Too small and performance degrades
  - Too large and space is wasted
- DB2 helps you manage the size
  - REORG AUTOESTSPACE YES
  - RTS tracks the number of overflowed entries

DB2 10 Temporal Data Overview

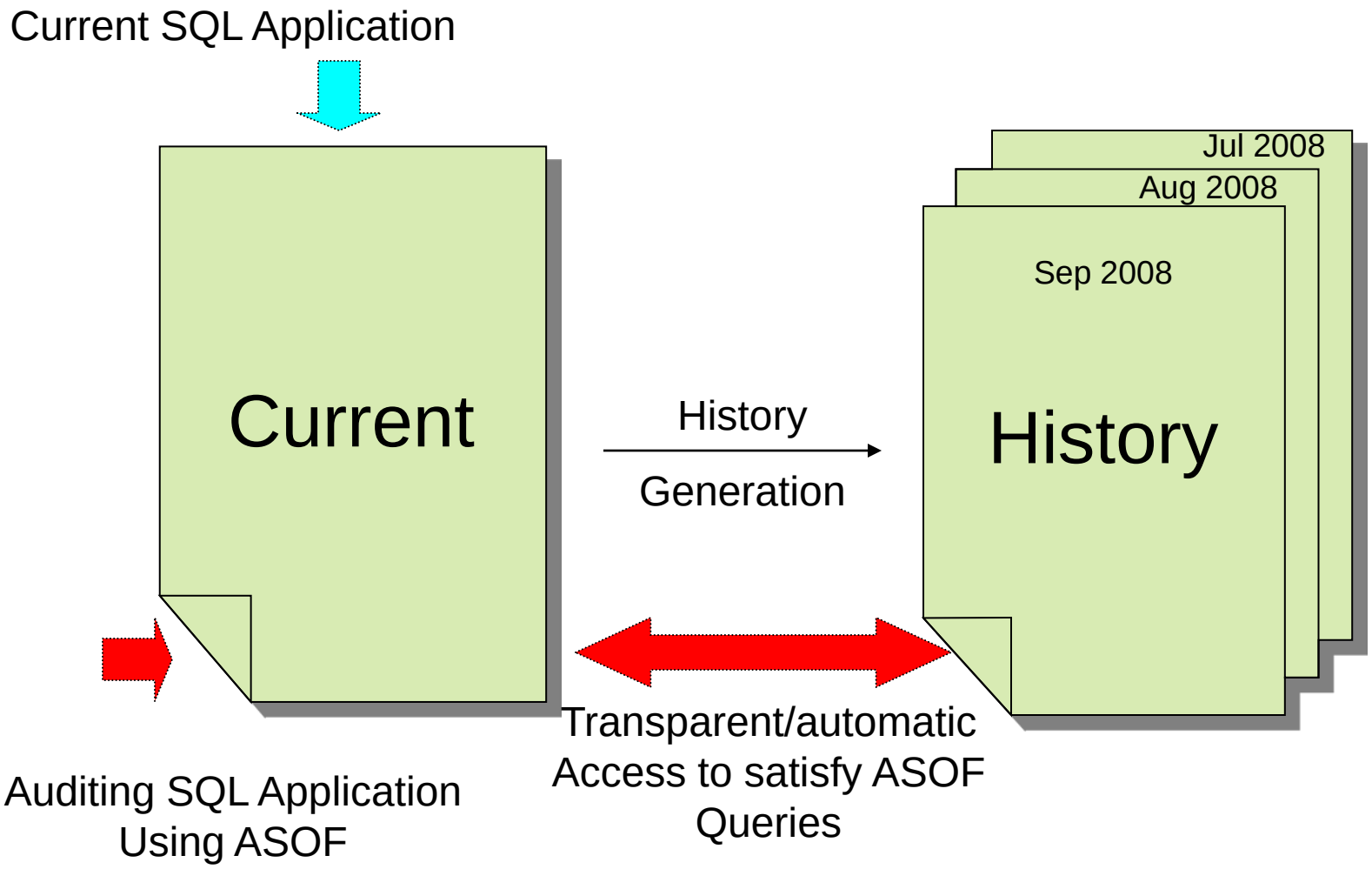
Motivation for Temporal Data

- Dramatic cost savings via Application Simplification
- Dramatic Reduction in Time to Deployment
- Auditing compliance advantages by moving logic from application to database layer
- Provides an easy way for applications to manage time sensitive data
  - Powerful, novel, yet intuitive SQL
  - End users deploy without IT Staff intervention
  - Query past/future data with current queries
  - Same schema name, simply add novel temporal clause
  - Response time for current DML/queries preserved
  - Response time for past/future queries comparable

Versioned data or Temporal Data

- Table-level specification to control data management based upon time
- Two notions of time:
  - System time: notes the occurrence of a data base change
    - “row xyz was deleted at 10:05 pm”
    - Query at current or any prior period of time
    - Useful for auditing, compliance
  - Business time: notes the occurrence of a business event
    - “customer xyz’s service contract was modified on March 23”
    - Query at current or any prior/future period of time
    - Useful for tracking of business events over time, application logic greatly simplified
- New syntax in FROM clause to specify a time criteria for selecting historical data

Current and History



Temporal UPDATE example (business time)

Simple table definition (Policy#, start, end, coverage)

Table has 1 row of (123,'01/01/2001', '12/31/2001', 1000)

```
UPDATE policy p
   FOR BUSINESS_TIME FROM DATE('03/01/2001') TO DATE('03/31/2001')
   SET coverage = 2000;
```

Result of the update statement is 3 rows:

(123,'01/01/2001','03/01/2001',1000)
(123,'03/01/2001','03/31/2001',2000)
(123,'03/31/2001','12/31/2001',1000)

IBM Smart Analytics Optimizer Overview

Why Optimize Smart Analytics?

- **Today, performance of Business Intelligence (BI) queries is too unpredictable**
  - When an analyst submits a query, s/he doesn't know whether to:
    - Wait for the response
    - Go out for coffee
    - Go out for dinner
    - Go home for the night!
  - Response time depends upon "performance layer" of indexes & materializations
  - Depends critically on predicting the workload
  - But BI is inherently ***ad hoc!***

- **Goal of IBM Smart Analytics Optimizer:**

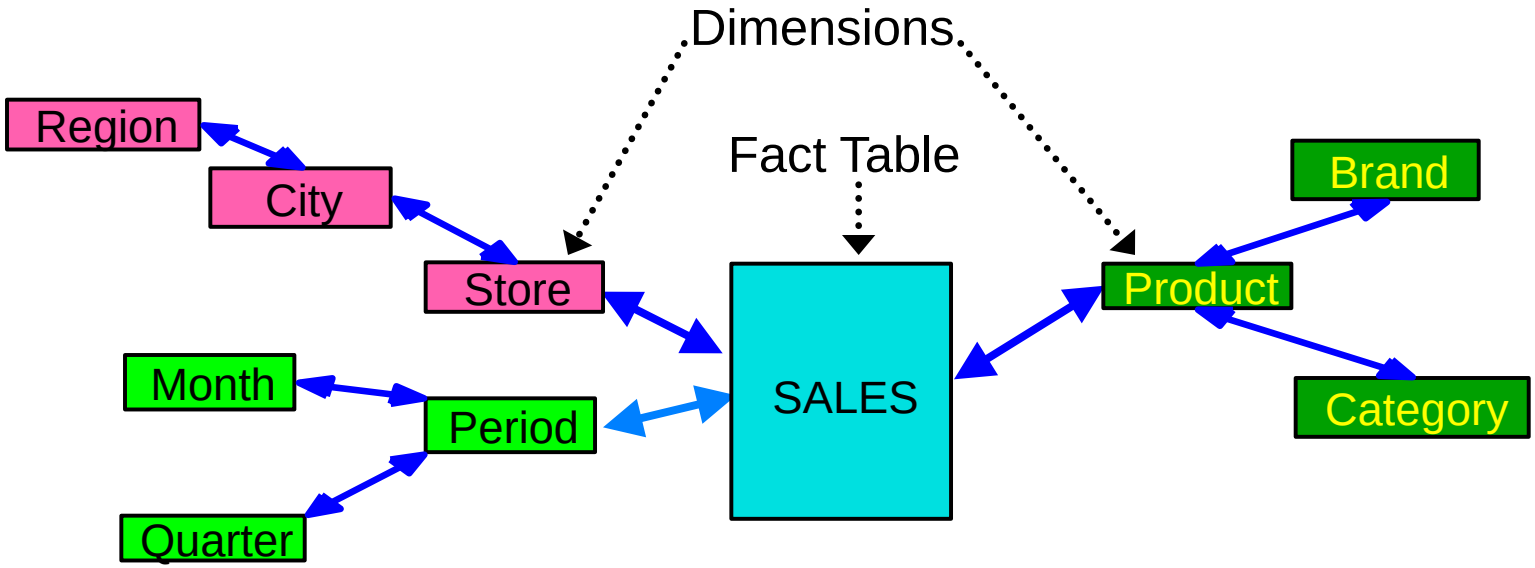
  Predictably Fast (i.e., Interactive) Ad Hoc Querying
  - **Any** query should run in about the same time
  - Permit an Analyst to interact with the data

What Is the IBM Smart Analytics Optimizer?

- **IBM Smart Analytics Optimizer for z/OS (ISAO)** is:
  - Network-attached accelerator to DB2 on z/OS
    - (Future: also DB2 for Linux, UNIX, and Windows and Informix IDS)
  - Exploits:
    - Large main memories
    - Commodity multi-core processors
    - Extreme compression
  - Speeds up typical Data Warehouse / Business Intelligence SQL queries by **10x to 100x**
  - Without requiring tuning of indexes, materialized views, etc.

Target Market: Business Intelligence (BI)

- Characterized by:
  - **“Star” or “snowflake” schema:**



  - **Complex, ad hoc queries that typically**
    - Look for trends, exceptions to make actionable business decisions
    - Touch large subset of the database (unlike OLTP)
    - Involve aggregation functions (e.g., COUNT, SUM, AVG,…)
    - **The “Sweet Spot” for IBM Smart Analytics Optimizer!**

What IBM Smart Analytics Optimizer is Designed For

- **OLAP-style SQL queries:**
  - Relational star schema (large **fact table** joined to multiple **dimensions**)
  - Large subset of data warehouse accessed, reduced significantly by...
  - **Aggregations** (SUM, AVG, COUNT) and optional **grouping** (GROUP BY)
  - **Looking for trends or exceptions**

- **EXAMPLE SQL**:

```
SELECT PRODUCT_DEPARTMENT, REGION, SUM(REVENUE)
FROM FACT_SALES F
     INNER JOIN DIM_PRODUCT P ON F.FKP = P.PK
     INNER JOIN DIM_REGION R ON F.FKR = R.PK
     LEFT OUTER JOIN DIM_TIME T ON F.FKT = T.PK
WHERE T.YEAR = 2007
GROUP BY PRODUCT_DEPARTMENT, REGION
```

Reference

- Redbooks at www.redbooks.ibm.com
  - DB2 9 for z/OS Technical Overview SG24-7330
  - DB2 9 for z/OS Performance Topics SG24-7473
- DB2 for z/OS home page at www.ibm.com/software/db2zos
  - E-support (presentations and papers) at www.ibm.com/software/db2zos/support.html

Key Details About DB2 10: Getting Ready

Prerequisites: migrate from DB2 9 for z/OS or DB2 for z/OS V8

- z/OS V1.10 SMS-managed DB2-managed DB2 catalog
- System z10, z9, z890, z990, and above (no z800, z900)
- DB2 Connect 9 FP1, 9.7 FP3 for many 10 functions, FP2 beta
- IMS 10 & 11 (not 9) CICS compilers (See announcement)
- Info APARs for migration II14477 (9), II14474 (8)
- SPE PK56922 PK69411 PK61766 PK85956 PM04680 PK87280 PK87281 PM08102 PM08105
- Premigration check DSNTIJPA PM04968

Items deprecated in earlier versions eliminated: more for V8 mig.

- Private protocol → DRDA (DSNTP2DP, PK92339, PK64045)
- Old plans and packages V5 or before → REBIND
- Plans containing DBRMs → packages PK62876 PK79925 (V8)
- ACQUIRE(ALLOCATE) → ACQUIRE(USE)
- Old plan table formats → DB2 V8 or 9, Unicode, 59 cols PK85068
- BookManager use for DB2 publications → Info Center, pdf

Disclaimer/Trademarks

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

The information on the new product is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information on the new product is for informational purposes only and may not be incorporated into any contract. The information on the new product is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion. *

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

Trademarks The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the pages of the presentation:

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: AIX, AS/400, DataJoiner, DataPropagator, DB2, DB2 Connect, DB2 Extenders, DB2 OLAP Server, DB2 Universal Database, Distributed Relational Database Architecture, DRDA, eServer, IBM, IMS, iSeries, MVS, Net.Data, OS/390, OS/400, PowerPC, pSeries, RS/6000, SQL/400, SQL/DS, Tivoli, VisualAge, VM/ESA, VSE/ESA, WebSphere, z/OS, zSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Other company, product, or service names may be trademarks or service marks of others.

Thank you

James Guo

IBM

guojw@us.ibm.com