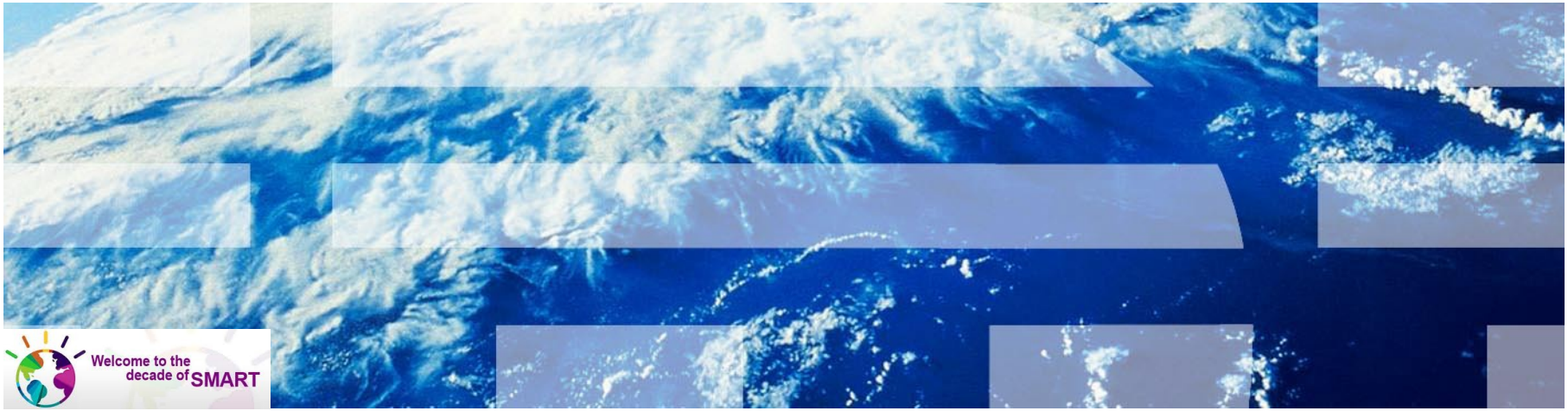
IBM Software

Information Management

DB2 10.1 LUW New Features and Enhancements



June 4, 2012
Burt Vialpando - IBM

DB2 10.1 LUW New Features and Enhancements
Agenda

DB2 10.1 LUW New Features

- Temporal Data Management and Time Travel Query
- Multi Temperature Storage Support
- Ingest Utility
- Row and Column Access Control (RCAC)
- Usage List
- Insert Time Clustering (ITC) Tables
- Zigzag Join

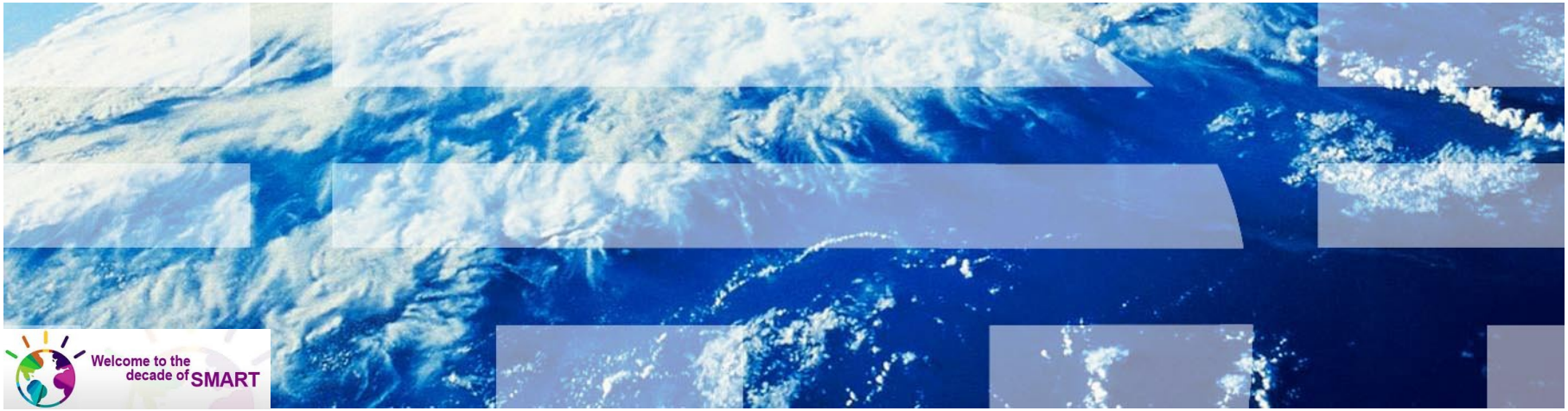
DB2 10.1 LUW Feature Enhancements

- Deep Compression Enhancements
- Workload Manager (WLM) Enhancements
- Event Monitor Enhancements
- Oracle Compatibility Enhancements
- Autonomic Computing Enhancements
- Multi-core parallelism and parallel index scan
- pureScale Enhancements
- HADR Enhancements
- Data Studio Enhancements

IBM Software

Information Management

DB2 10.1 LUW New Features



DB2 Temporal Data Management and Time Travel Query New!

Historical data automatically captured and easily queried

Temporal Data Management gives you:

- Time Travel Query capability without the burden of changing the application
  - or creating complex trigger/function/procedure solutions
- Optimized environments for meeting audit and compliance inquiries
- Standardized support for temporal INSERT, UPDATE and DELETE operations
- Reduced risks, lower costs and DBA time savings

Temporal Data Management

EMPLOYEES_TB

EmpID	Dept	System_start	System_end
12345	M15	05/31/2000	12/30/9999

History automatically maintained by DB2

EMPLOYEES_HISTORY_TB

EmpID	Dept	System_start	System_end
67890	K25	11/15/1995	03/31/2000
12345	J13	11/15/1995	12/31/1998
12345	M24	12/31/1998	05/31/2000

Time Travel Query

```
SELECT Dept FROM employees_tb
    WHERE EmpID=12345;
```

See data as it is right now

```
SELECT Dept FROM employees_tb
    FOR SYSTEM_TIME
    AS OF '12/01/1997'
    WHERE EmpID = 12345;
```

See data as it was in the past

DB2 Temporal Data Management
Temporal data: the basic concepts

- **System Time**
  - Also known as “Transaction Time”
  - Tracks when changes are made to the state of the table itself
    - e.g. when an insurance policy is modified or a when a loan is created
  - Timestamps assigned by DB2
- **Business Time**
  - Also known as “Valid Time”
  - Tracks the effective dates of business conditions
    - e.g. the term dates of an insurance policy or duration period of a loan
  - Timestamps assigned by the application
- **Bitemporal**
  - When both System Time and Business Time are tracked in a single table
- **Temporal tables use an inclusive / exclusive approach**
  - Start times include the dates
  - End times exclude the dates
- **System temporal tables consist of a base table with a matching history table**
  - The history table is automatically updated by DB2 during DML operations to the base table
- **Business temporal tables only consist of a base table**
  - DB2 does “row splits” during DML operations to maintain the business versions of data

System Time is needed to see:
DB2’s physical view of time
System’s validity
Past to present view of the data

Business Time is needed to see:
The application’s view of time
Business validity
Past, present or future view of the data

DB2 Temporal Data Management
When is it needed?

Industry	Typical business cases for temporal data	Type
Healthcare	A pending lawsuit requires a hospital to reassess its awareness of a patient's medical condition just before a new treatment was ordered.	System Time
Financial	An internal audit requires a financial institution to report on all changes made to a client's records during the past 7 years.	System Time
Insurance	A client challenges an insurance firm's resolution of a claim involving a car accident. The insurance firm needs to determine the policy's terms in effect for that client when the accident occurred.	Business Time
Retail	A retailer needs to ensure that no more than one discount is offered for a given product during any period of time.	Business Time
Banking	A client inquiry reveals a data entry error involving the three-month introductory interest rate on a credit card. The bank needs to retroactively correct the error and compute a new balance.	Bitemporal

BV

DB2 Temporal Data Management
System Time: Three easy steps to implemen**New!**

1. **Create the System Time base table**

```
CREATE TABLE EMPLOYEES _TB (
    EmpID        INT primary key not null,
    Dept         CHAR(3),
    Sys_start    TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
    Sys_end      TIMESTAMP(12) GENERATED ALWAYS AS ROW END NOT NULL,
    Trans_start  TIMESTAMP(12) GENERATED ALWAYS
            AS TRANSACTION START ID IMPLICITLY HIDDEN,
    PERIOD SYSTEM_TIME (Sys_start, Sys_end)
);
```

This can be any date or time expression

Tracks when a transaction begins a change to the table

Makes this a System Time table

2. **Create ... history table**

3.
```
CREATE TABLE EMPLOYEES_HISTORY_TB LIKE EMPLOYEES_TB
    WITH RESTRICT ON DROP;
```
... history table

```
ALTER TABLE EMPLOYEES_TB ADD VERSIONING
    USE HISTORY TABLE EMPLOYEES_HISTORY_TB;
```

Links base table to history table

DB2 Temporal Data Management
System Time DML operations temporal ramifications New!

- **INSERT**
  - Base table system times automatically maintained

Done on 01/31/2011

```
INSERT INTO EMPLOYEES_TB (EmpID, Dept),
    VALUES (12345, 'J13');
```

EMPLOYEES_TB

EmpID	Dept	System_start	System_end
12345	J13	01/31/2011	12/30/9999

- **UPDATE**
  - Automatically affect both base and history tables

Done on 05/15/2012

```
UPDATE EMPLOYEES_TB
    SET Dept = 'M15'
    WHERE EmpID = 12345;
```

EMPLOYEES_TB

EmpID	Dept	System_start	System_end
12345	M15	05/15/2012	12/30/9999

EMPLOYEES_HISTORY_TB

EmpID	Dept	System_start	System_end
12345	J13	01/31/2011	05/15/2012

- **DE**
  - Automatically affect both base and history tables

Done on 06/12/2012

```
DELETE FROM EMPLOYEES_TB
    WHERE EmpID = 12345;
```

EMPLOYEES_TB

EmpID	Dept	System_start	System_end

EMPLOYEES_HISTORY_TB

EmpID	Dept	System_start	System_end
12345	J13	01/31/2011	05/15/2012
12345	M15	05/15/2012	06/12/2012

DB2 Temporal Data Management
Business Time: One easy step to implement **New!**

1. Create a Business Time table

```
CREATE TABLE EMPLOYEES_TB (
   EmpID       INT NOT NULL,
   Dept        CHAR(3),
   Salary      DOUBLE,
   Bus_start   DATE NOT NULL,
   Bus_end     DATE NOT NULL,
   PERIOD BUSINESS_TIME (Bus_start, Bus_end),
   PRIMARY KEY(EmpID, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

This can be any date or time expression

Makes this a Business Time table

Prevents overlapping periods.

e.g. Only one salary figure is valid for any time period for any given employee.

Note:

Business Time tables do not require a history table.

All data is “current” information about the past, present or even future state of business.

DB2 Temporal Data Management
Business Time DML operations temporal ramifications New!

- **INSERT**
  - Business dates required, can be close ended

```
INSERT INTO EMPLOYEES_TB  VALUES
   (12345,' J13', 5000, '01-01-2011', '01-01-2012');
```

EMPLOYEES_TB

EmpID	Dept	Salary	Business_start	Business_end
12345	J13	5000	01/01/2011	01/01/2012

- **INSERT**
  - Business dates can also be open ended

```
INSERT INTO EMPLOYEES_TB  VALUES
   (67890,' M15', 7000, '01-01-2011', '12-30-9999');
```

EMPLOYEES_TB

EmpID	Dept	Salary	Business_start	Business_end
12345	J13	5000	01/01/2011	01/01/2012
67890	M15	7000	01/01/2011	12/30/9999

- **UPDATE**
  - Updates (or deletes) cause automatic “row splits” that tracks appropriate business periods

```
UPDATE EMPLOYEES_TB
   FOR PORTION OF BUSINESS_TIME
   FROM '06-01-2011' TO '12-30-9999'
   SET salary = Salary + 500
   WHERE EmpID = 67890;
```

EMPLOYEES_TB

EmpID	Dept	Salary	Business_start	Business_end
12345	J13	5000	01/01/2011	01/01/2012
67890	M15	7000	01/01/2011	06/01/2011
67890	M15	7500	06/01/2011	12/30/9999

DB2 Temporal Data Management
Business Time DML operations temporal ramifications continued **New!**

- **UPDATE**
  - Multiple “row splits” can occur depending upon the implied business time periods
  - Here an employee remains employed, but takes an unpaid leave of absence

```
UPDATE EMPLOYEES_TB
   FOR PORTION OF BUSINESS_TIME
   FROM '07-01-2011' TO '09-01-2011'
   SET Salary = 0
   WHERE EmpID = 12345;
```

EMPLOYEES_TB

EmpID	Dept	Salary	Business_start	Business_end
12345	J13	5000	01/01/2011	07/01/2011
12345	J13	0	07/01/2011	09/01/2011
12345	J13	5000	09/01/2011	01/01/2012
67890	M15	7000	01/01/2011	06/01/2011
67890	M15	7500	06/01/2011	12/30/9999

- **DELETE**
  - Deletes can cause row splits too
  - Here an employee is suspended for 1 year with no pay and no other vesting accruing either

```
DELETE EMPLOYEES_TB
   FOR PORTION OF BUSINESS_TIME
   FROM '01-01-2012' TO '01-01-2013'
   WHERE EmpID = 67890;
```

EMPLOYEES_TB

EmpID	Dept	Salary	Business_start	Business_end
12345	J13	5000	01/01/2011	07/01/2011
12345	J13	0	07/01/2011	09/01/2011
12345	J13	5000	09/01/2011	01/01/2012
67890	M15	7000	01/01/2011	06/01/2011
67890	M15	7500	06/01/2011	01/01/2012
67890	M15	7500	01/01/2013	12/30/9999

Notice the gap of one year here.

DB2 Temporal Data Management
Time Travel Query capabilities – System Time examples **New!**

- **FOR [time-type] AS OF**...
  - Query for data at a certain point in time

```
SELECT Dept FROM employees_tb
   FOR SYSTEM_TIME
   AS OF TIMESTAMP('12/15/1997')
   WHERE EmpID = 12345;
```

- **FOR [time-type] FROM... TO**...
  - Query data from a certain time to a certain time
  - Uses an INCLUSIVE, EXCLUSIVE approach

```
SELECT Dept FROM employees_tb
   FOR SYSTEM_TIME
   FROM TIMESTAMP('12/01/1997')
        TO TIMESTAMP('12/31/1997')
   WHERE EmpID = 12345;
```

12/31/1997 rows NOT returned

- **FOR [time-type] BETWEEN... AND**...
  - Query data from a certain time through a certain time
  - Uses an INCLUSIVE, INCLUSIVE approach

```
SELECT Dept FROM employees_tb
   FOR SYSTEM_TIME
   BETWEEN TIMESTAMP('12/01/1997')
           AND TIMESTAMP('12/31/1997')
   WHERE EmpID = 12345;
```

12/31/1997 rows returned

* Business Time uses the same syntax.

DB2 Temporal Data Management
Time Travel Query additional capabilities New!

- **Views on temporal tables**
  - If they contain a clause: **`FOR SYSTEM_TIME`** or **`FOR BUSINESS_TIME`**
    - Data is restricted to just the rows constrained by that clause
    - Regular queries on the view should not contain that clause
  - If they do not contain a clause: **`FOR SYSTEM_TIME`** or **`FOR BUSINESS_TIME`**
    - Data is un-constrained and all rows are exposed to the view
    - Queries on the view therefore require that clause

- **Registers for temporal time**
  - You can set special registers to constrain queries that do not contain temporal clauses
  - Allows you to use regular queries without the clause to run in a temporal mode.

```
SET CURRENT TEMPORAL BUSINESS_TIME = '2011-01-01 10:00:00';
```

The same as adding `FOR BUSINESS_TIME AS OF` to all queries in the session

```
SET CURRENT TEMPORAL SYSTEM_TIME=  CURRENT TIMESTAMP – 1 MONTH;
```

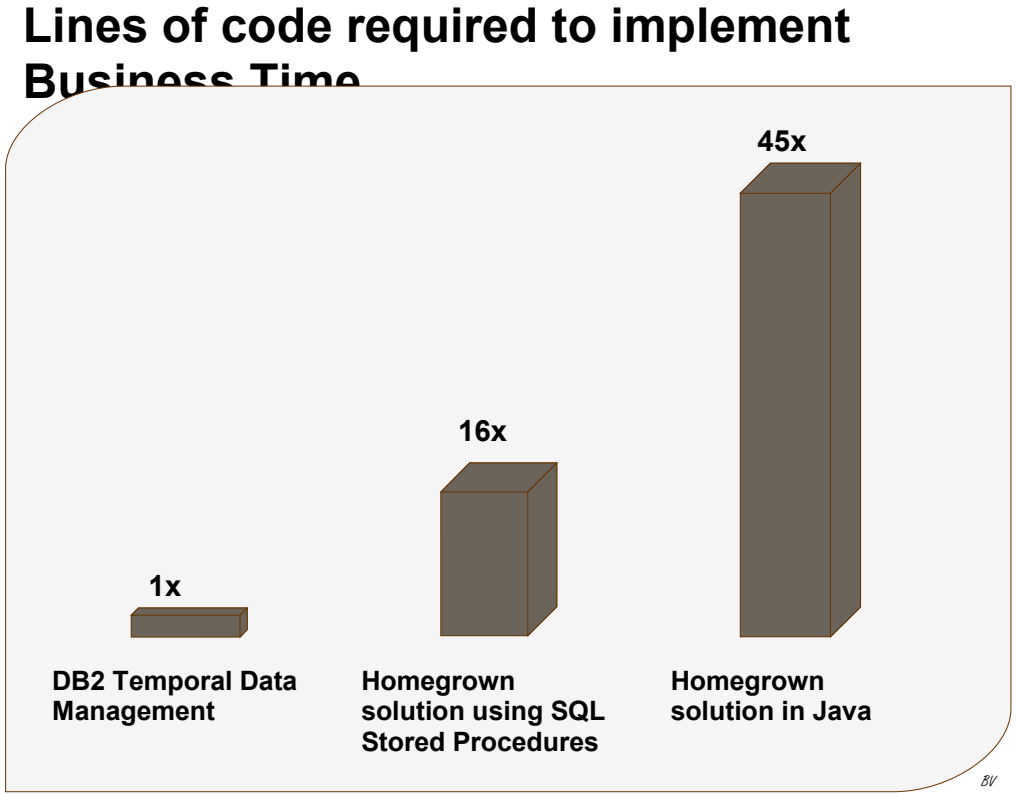
The same as adding `FOR SYSTEM_TIME AS OF` to all queries in the session

DB2 Temporal Data Management
Performance tips and best practices

- **System Time history table performance tips**
  - **`APPEND ON`** should be considered
    - Since mostly **`SELECT`** and **`INSERT`** are used, rarely **`UPDATE`** or **`DELETE`**
  - Range partitioning should be considered
    - Which could be a good way to detach partitions to easily prune older data
  - A primary key index of the base table primary key plus temporal dates should be considered
    - This can often improve the **`FOR SYSTEM TIME AS OF`** queries
    - In lieu of this, consider a non-unique key that is similar to the primary key of the base table

- **Other best practices**
  - Define privileges on the history table with care, it does not require access for normal use
    - Users or applications doing all DML to the base table require only DML access to it
    - Keep regular users from doing any DML directly to the history table to keep its integrity
  - Consider using a global variable for minimum dates or maximum date values
    - Creating these once will ensure a consistency of use from every user or application
    - e.g. **`CREATE VARIABLE MYCONSTANTS.MAX_DATE DATE('12-31-9999');`**
  - Consider using **`RESTRICT ON DROP`** for the history table to avoid accidentally losing it
    - Otherwise, dropping the base table will also drop the history table

DB2 Temporal Data Management
IBM study of benefits for using the DB2 solution

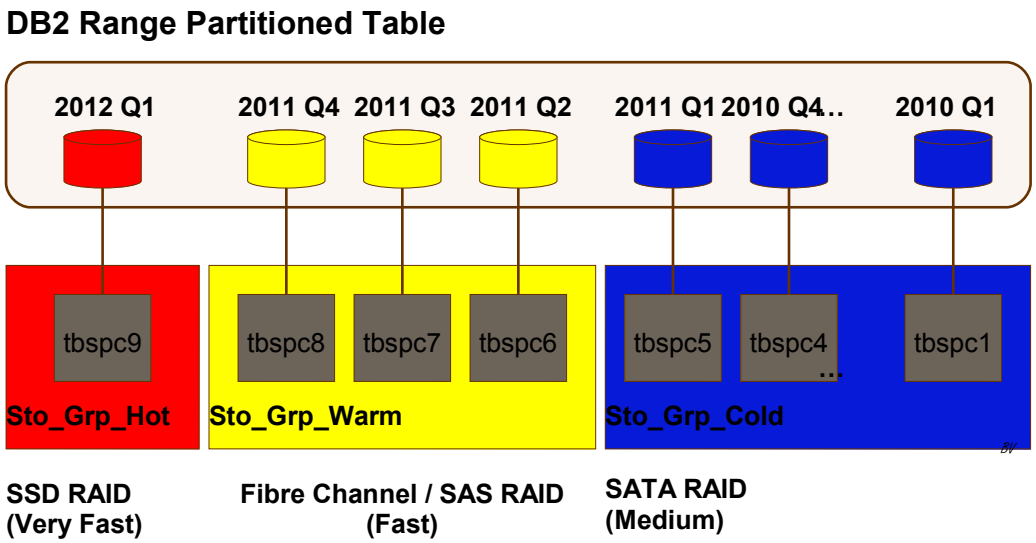


Note: This does not take into consideration the development cost of creating SQL statements for the final solution.

In this study it took only hours to create and test them in Time Travel Query syntax, where it took weeks to create and test them in the home grown solution.

DB2 Multi-Temperature Storage Support * New!
Use storage groups to maximize data placement on disk with ease

- Maximize your newest and fastest storage for your most important data
  - e.g. keep only the data you need for instant access on your solid state storage
- Extend the life of your older storage
  - e.g. keep infrequently used data on it
- Utilize storage groups in the automatic storage pool to manage this with ease
  - Administrative commands make moving data between storage systems easy
  - Prioritize data by what it is and where it resides
  - Integrates fully with DB2 Workload Manager and InfoSphere Optim Data Growth solutions



DB2 Multi-Temperature Storage Support
Two easy steps to implement New!

- Step #1 - Create storage groups (this is all in automatic storage):

```
Create stogroup Sto_Grp_Hot  on '/prodpath1/ssddisk'  overhead  0.75 device read rate 500;
Create stogroup Sto_Grp_Warm on '/prodpath2/sasdisk'  overhead  6.73 device read rate 100;
Create stogroup Sto_Grp_Cold on '/prodpath3/satadisk' overhead 12.75 device read rate 50;
```

- Step #2a – Alter (or create) automatic storage table spaces to use the storage groups:

```
Create tablespace tbspc9 using stogroup Sto_Grp_Hot;
Alter tablespace tbspc8 using stogroup Sto_Grp_Warm;
Alter tablespace tbspc5 using stogroup Sto_Grp_Cold;
```

The table space will inherit overhead and transfer rates from the storage group's overhead and device read rates respectively.

- Step #2b – or alter non-automatic storage table spaces this way:

```
Alter tablespace tbspc_dms managed by automatic storage using stogroup Sto_Grp_Hot;
Alter tablespace tbspc_dms rebalance;
```

Table spaces already in automatic storage will rebalance to the new storage automatically.

Table spaces not in automatic storage still require you to rebalance when moving there.

DB2 Multi-Temperature Storage Support
Rebalance considerations

- Automatic rebalance occurs by default when altering storage groups
  - **MON_GET_REBALANCE_STATUS** table function shows status **New!**

```
select        char(tbsp_name, 20)              as tbsp_name,
       rebalancer_status          as status,
       rebalancer_extents_remaining  as extents_remaining,
rebalancer_extents_processed          as extents_processed
from table    (MON_GET_REBALANCE_STATUS(NULL, -1)) ;
```

```
TBSP_NAME        STATUS  EXTENTS_REMAINING    EXTENTS_PROCESSED
---------------  ------  -------------------- --------------------
SALES_Q1_2011    ACTIVE                  6911                   34

1 record(s) selected.
```

  - Suspend or resume the reba

```
Alter tablespace SALES_Q1_2011 rebalance suspend;
… [let your system run until a better time for the rebalance]
Alter tablespace SALES_Q1_2011 rebalance resume;
```

DB2 Multi-Temperature Storage Support
Other DB2 Storage Group capabilities

DB2 Storage Group Capability	Description
SYSCAT.STOGROUPS SYSIBM.SYSSTOGROUPS SYSIBM.SYSTABLESPACES (SGID column) **New!**	System catalog views to get storage group information in the database
ADMIN_GET_STORAGE_PATHS **New!**	Table function to get storage group information in the database
ADMIN_LIST_DB_PATHS MON_GET_TABLESPACE MON_GET_CONTAINER	Other table functions to help manage storage
db2pd -storagegroup **New!**	Utility option for seeing storage group information in the database
SET STOGROUP PATHS FOR command **New!**	• Use when performing a redirected restore of a database that uses storage groups • Use in the same way you would for setting table space paths
DATA TAG [numeric-value] **New!**	• Create or alter a storage group (or table space) to contain this optional characteristic • A table space can inherit its storage group DATA TAG value if not specified • This value is used by the Workload Manager to influence how queries will run, to give priority by the location of the data (See the WLM slides for more details)

Physical Database Storage Model Enhancement
DB2 table space and storage group creation

- DB2 table space creation now includes these new clauses:
  - **DATA TAG, USING STORAGE GROUP** and **INHERIT**

```
CREATE                  [ LARGE / REGULAR / [SYSTEM / USER] TEMPORARY ]
    TABLESPACE          tablespace-name
    PAGESIZE            integer
    MANAGED BY          [ AUTOMATIC STORAGE][ SYSTEM / DATABASE ]
                  [ system-container / database-container ]
    EXTENTSIZE          [ no-pages / integer ]
    PREFETCHSIZE        [ no-pages / integer ]
    BUFFERPOOL          bufferpool-name
    OVERHEAD            [no-milliseconds / INHERIT]
    TRANSFERRATE        [no-milliseconds / INHERIT]
    DATA TAG            [ integer / INHERIT  / NONE]
    USING STORAGE GROUP [sto-group-name]
    ...
    AUTORESIZE          YES...
          ;
```

New!

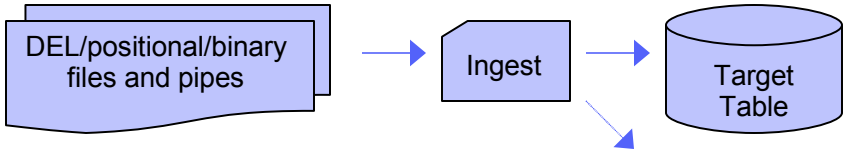
INHERIT means the table space will get its behavior from its storage group

New!

```
CREATE STOGROUP     sto-group-name
   ON                [path]
   OVERHEAD          no-of-milliseconds
   DEVICE READ RATE  no-megabytes-second
   DATA TAG          [integer / NONE]
;
```

The use of storage groups is optional if automatic storage is not used.

DB2 Ingest Utility New!
High-speed, client side Continuous Data Ingest (CDI)



DB2 Ingest Main Characteristics

- All DML operations supported
  - Insert, update, delete, replace and merge
- Moves large amounts of data in real-time
  - Can process a continuous data stream
- Availability and concurrency of data is preserved
  - With low latency, row lock approach
- Supports the following table types:
  - Regular tables, nicknames, MDCs, ITCs, RCTs, MQTs, Range partitioned, temporal and updatable views
- Input types:
  - Delimited, positional and binary files or pipes
- Commit by time or row count
- Uses three different phases
  - Transport, format, flush
- Monitor with `LIST` and `GET STATS` options
  - Show ingest jobs being run by connected ID

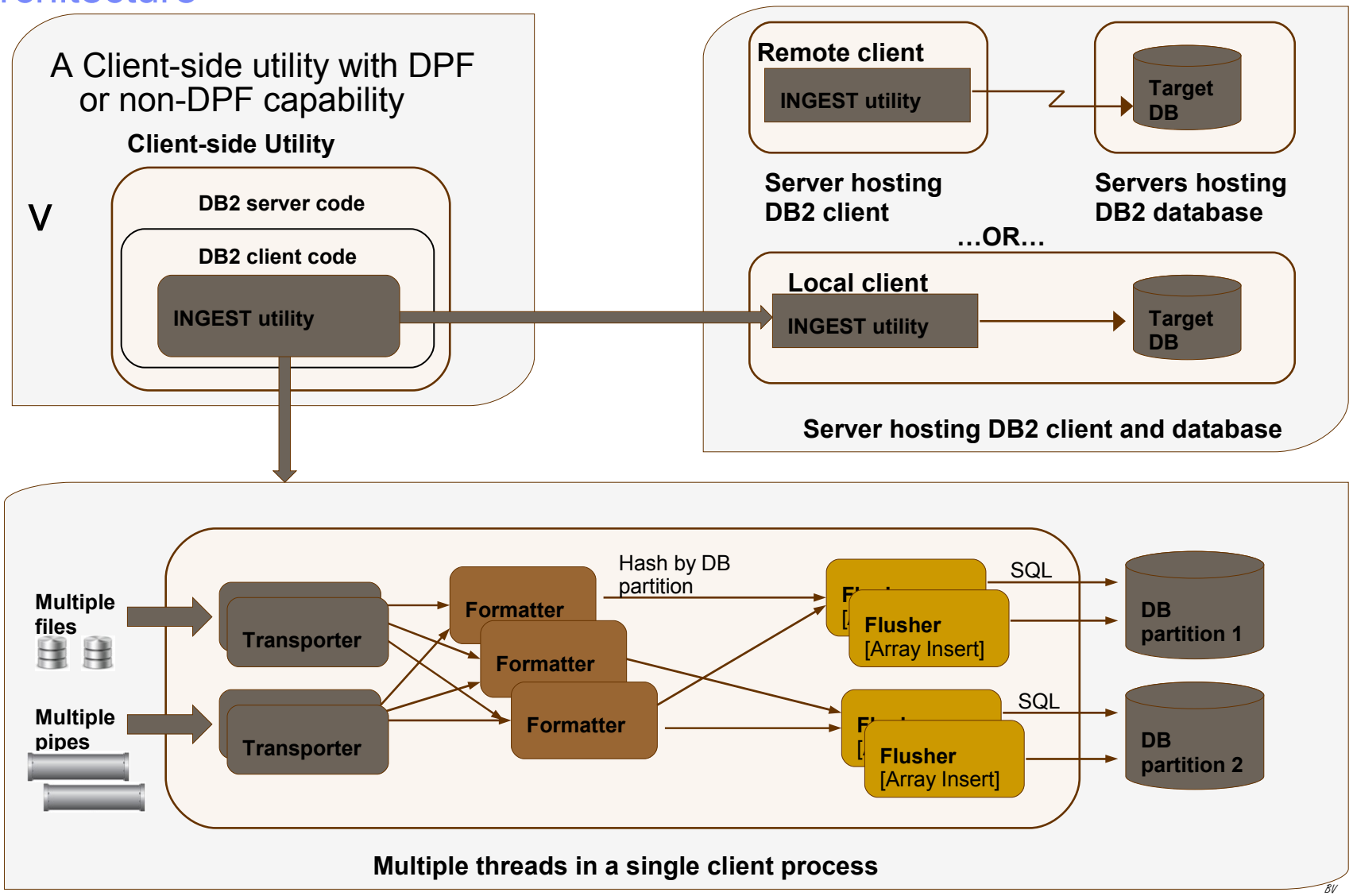
```
INGEST FROM FILE [file-name]
    FORMAT DELIMITED
    RESTART NEW [Job-ID]
    INSERT INTO [table-name] ;
```

The **RESTART NEW** option gives the ability to restart an ingest job and uniquely identifies it with an ID

```
INGEST LIST;
```

```
INGEST GET STATS FOR [ALL/Job-ID]
    EVERY [number] SECONDS;
```

DB2 Ingest Utility
Architecture



DB2 Ingest Utility
Examples New!

Delimited with UPDATE, commit count

```
INGEST SET commit_count 1000;
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED
  (  $key1 INTEGER EXTERNAL,
     $key2 INTEGER EXTERNAL,
     $data1 CHAR(8),
     $data2 CHAR(32),
     $data3 DECIMAL(5,2) EXTERNAL )
   EXCEPTION TABLE except_tb
   MESSAGES ICD_msg.txt
   RESTART NEW 'ICDJob001'
   UPDATE my_table
   SET (data1, data2, data3) =
         ($data1, $data2, $data3)
   WHERE (key1 = $key1) AND (key2 = $key2);
```

Positional with MERGE, multiple files

```
INGEST FROM FILE my_file1.txt,  my_file2.txt
  FORMAT POSITIONAL
  (  $key_fld  POSITION(1:10)   INTEGER EXTERNAL,
     $data_fld POSITION(11:20) INTEGER EXTERNAL )
  MERGE INTO my_table
             ON (key1 = $key_fld)
  WHEN MATCHED AND (action = 'U')
          THEN UPDATE SET data = $data_fld
  WHEN MATCHED AND (action = 'D')
          THEN DELETE;
```

Pipe input with DELETE, commit period

```
INGEST SET pipe_timeout 600;
INGEST SET commit_period 300;
INGEST FROM PIPE DataWarehousePipe1
  FORMAT DELIMITED
 ( $key1 INTEGER EXTERNAL,
   $key2 INTEGER EXTERNAL )
   DELETE FROM my_table
   WHERE (key1 = $key1) AND (key2 = $key2);
```

DB2 Ingest Utility
Parameters

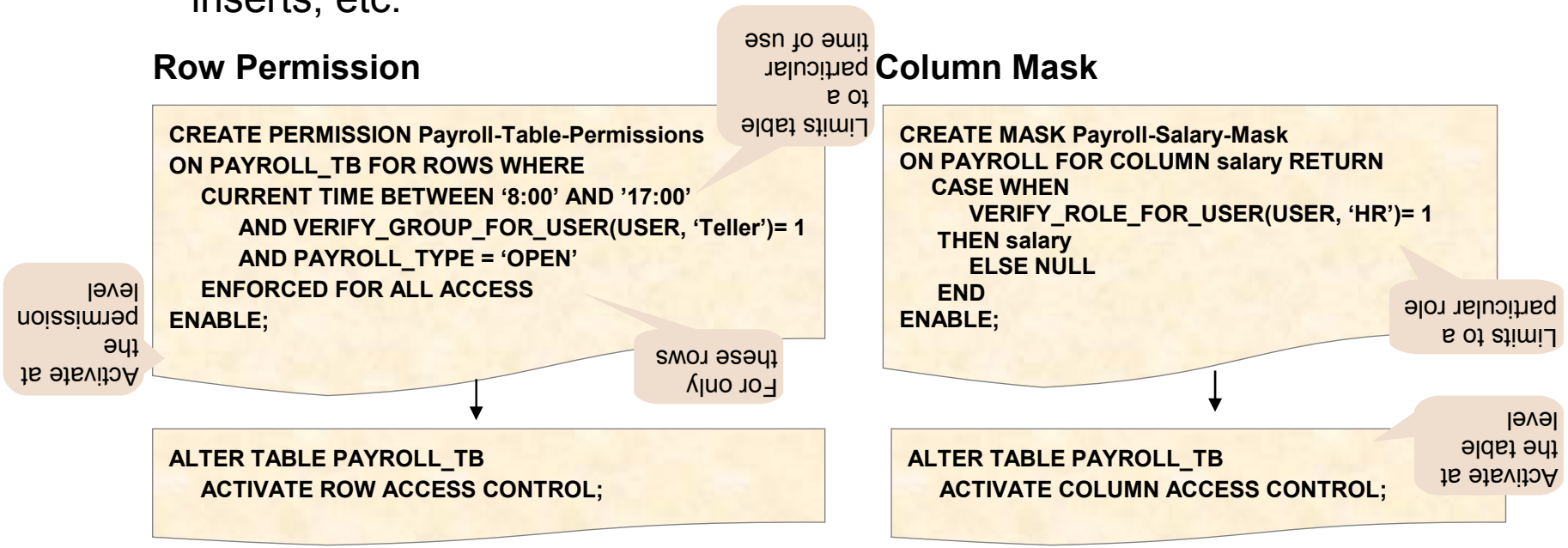
Parameter	Range	Default	Description
commit_count	0 to max	0	Number of rows each flusher writes in a single transaction before issuing a commit
commit_period	0 to 2,678,400 seconds	1 second	Number of seconds between committed transactions
num_flushers_per_partition	0 to system resources	Max(1,((no logical cpus)/2) / no partitions))	Number of flushers to allocate for each database partition (0 means 1 flusher for all partitions)
num_formatters	1 to max no threads	Max(1,(no logical cpus)/2)	The number of formatter threads
pipe_timeout	0 to 2,678,400 seconds	600 seconds	The maximum number of seconds to wait for data when the input source is a pipe (0 means wait indefinitely). If no data arrives within the specified period, the `INGEST` command ends and an error is returned
retry_count	0 to 1,000	0	The number of times to retry a failed (but recoverable) transaction
retry_period	0 to 2,678,400 seconds	0	The number of seconds to wait before retrying a failed (but recoverable) transaction
shm_max_size	1 to available memory	1 GB	Maximum size of IPC shared memory in bytes (allocated on the client where `INGEST` is run)

DB2 Ingest vs. DB2 Load
Usage scenarios

Use Ingest When	Use Load When
Other applications must update the table while the ingest is occurring	Your table contains XML or LOBS
You require **MERGE**, **UPDATE** or **DELETE** operations to be performed	You need to load from a cursor or a device
You require a SQL expression to populate a column during the ingest	You need to load from an IXF source file
You require automatic recoverability (of a failed ingest)	Your table contains a **GENERATED ALWAYS** column and you wish to override it
	Your table contains a **SYSTEM TIME** column and you wish to override it

Row and Column Access Control (RCAC) New!
Also called "Fine-Grained Access Control"

- Row permission and column mask approach has three key advantages:
  - No database user is inherently exempted from them, even users that have DATAACCESS
  - Data protection is assured, regardless of how it is accessed
  - Application changes are not required to take advantage of them
- Does not apply to DB2 applied internal processing
  - Like explain table population, MQT refresh, temporal history table inserts, etc.



Row and Column Access Control (RCAC)
New scalar functions and SECURED keyword New!

New Scalar Functions

```
VERIFY_ROLE_FOR_USER(user, role1, role2, …)
```

Returns 1 if any roles are associated with the user in the list. If not, it returns a 0.

```
VERIFY_GROUP_FOR_USER(user, group1, group2, …)
```

Returns 1 if any groups are associated with the user in the list. If not, it returns a 0.

Note: LDAP groups can be used here as well.

```
VERIFY_TRUSTED_CONTEXT_FOR_USER(user, role1, role2, …)
```

Returns 1 if any role that is acquired through a trusted connection is in or contains any of the roles in the list. If not, returns 0.

New object keyword: SECURED

```
ALTER FUNCTION MYFUNC1  SECURED;
```

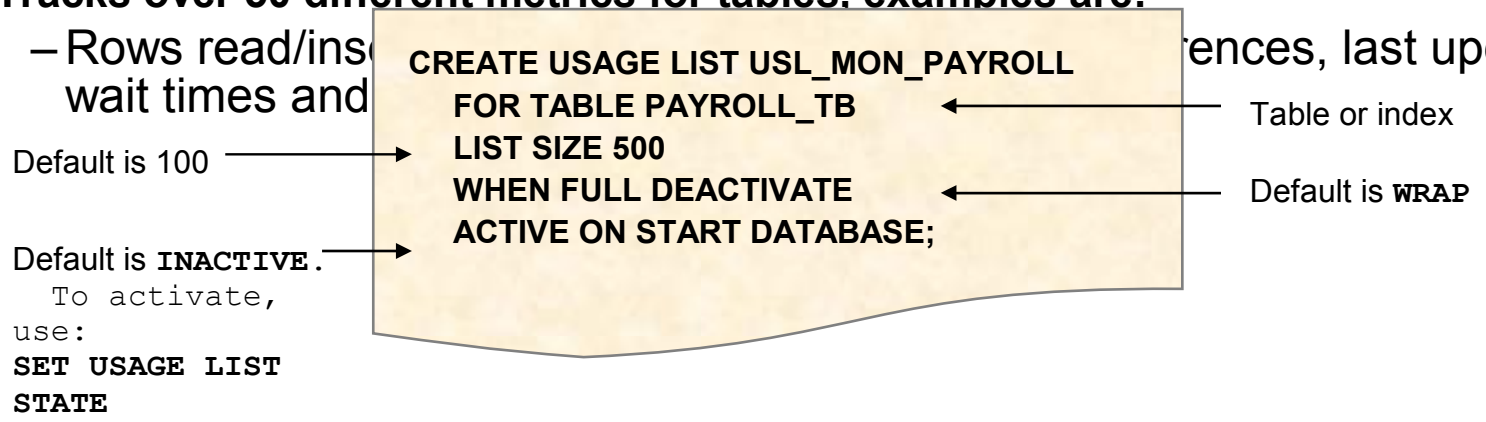
UDFs and triggers that are used in an RCAC environment should be secured. Here is an example of usage.

```
CREATE MASK Payroll-Salary-Mask
ON PAYROLL FOR COLUMN salary RETURN
    CASE WHEN
        VERIFY_ROLE_FOR_USER(USER, 'HR')= 1
    THEN MYFUNC1(salary)
        ELSE NULL
    END
ENABLE;
```

Here SALARY is masked, so a UDF that is invoked on this column needs to be secured.

Usage List – The DML usage recorder New!
A monitoring enhancement

- **A usage list is a database object that**
  - Records each DML statement section that references a particular table or index
  - Also captures statistics about how that section affects each object
- **All table types supported <u>except:</u>**
  - Aliases, nicknames, views, created temporary tables, detached tables, hierarchy tables
- **Only the following index types are supported:**
  - Regular, block, dimension block, clustering
- **Lists are stored in the monitor heap (`mon_heap_sz`)**
  - So increase if necessary
- **Tracks over 50 different metrics for tables, examples are:**
  - Rows read/ins… …ences, last updated, wait times and…



Usage List
Functions for retrieving information **New!**

- **Retrieve information from usage list using the following table functions:**
  - **MON_GET_TABLE_USAGE_LIST** - Table usage list information
  - **MON_GET_INDEX_USAGE_LIST** - Index usage list information
  - **MON_GET_USAGE_LIST_STATUS** - The current status of the usage list

- **Example:**

```
SELECT * FROM
   TABLE( MON_GET_TABLE_USAGE_LIST(NULL, 'USL_MON_PAYROLL', 3));
```

```
USAGELISTSCHEMA USAGELISTNAME   TABSCHEMA TABNAME     MEMBER . . . LAST_UPDATED                  . . .
--------------- --------------- --------- ----------- ------ . . . -------------------------- . . .
ISAYYID         USL_MON_PAYROLL ISAYYID   PAYROLL_TB       3 . . . 2012-03-03-10.20.22.727803 . . .
ISAYYID         USL_MON_PAYROLL ISAYYID   PAYROLL_TB       3 . . . 2012-03-03-10.20.58.202161 . . .

2 record(s) selected.
```

Insert Time Clustering Tables New!
A variation of MDCs – based on a single time dimension

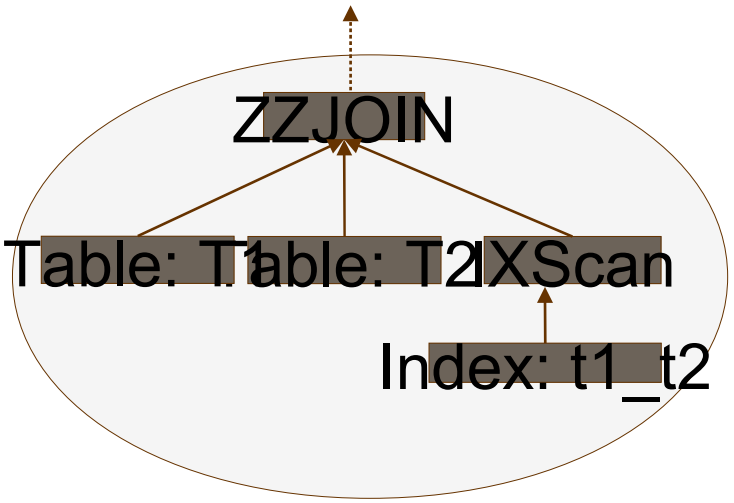
- **Insert Time Clustering Tables (ITCs) are similar to MDCs**
  - They use the `ORGANIZE BY` syntax
  - They use block allocation
  - They use block based indexing
  - They can have indexing on any columns
  - They have similar performance advantages
  - They have similar maintenance advantages
- **ITCs are unique in that they**
  - Always have only one dimension
  - Use a virtual, time-based column to cluster rows
  - Have that column populated by DB2 during an insert, import or load
    - The generated time-based column data is all a similar time determined by DB2

```
CREATE TABLE ITC_TB (COL1 CHAR(100), COL2 INT  )
    ORGANIZE BY INSERT TIME ;
```

Zigzag Join Overview **New!**
A new join type

What is zigzag join?

- A new join method for complex queries on dimensional schemas
- Works for star schema queries in single or multiple subject areas with snowflakes
- Works seamlessly with serial or DPF databases, range-partitioned and MDC tables
- Simple index adviser integrated with db2exfmt and OQWT to help with fact table index design and get best performance from zigzag join



Why do we need zigzag join?

- Improved query performance
- More stable performance that is less sensitive to small query or configuration changes
- Easier logical database design

When can DB2 use zigzag join?

- Joins between fact table(s) and dimension tables on dimension unique keys
- Fact table must have a multicolumn index that contains at least two of the join keys used in the query

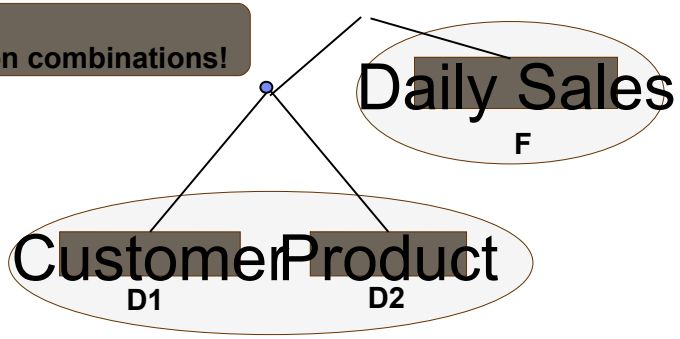
Zigzag Join Details
How does it work?

- First forms the conceptual Cartesian product of dimensions but avoids most non-productive probes from the Cartesian product into the fact table
- Zigzag join skips over non-productive combinations of dimension values to find matches in the fact table with a reduced number of probes
- Fact table index provides feedback to dimensions
- Zigzags through the dimensions and the fact tables

How many items in the "cosmetics" product category were sold to customers in the 39-54 age group?

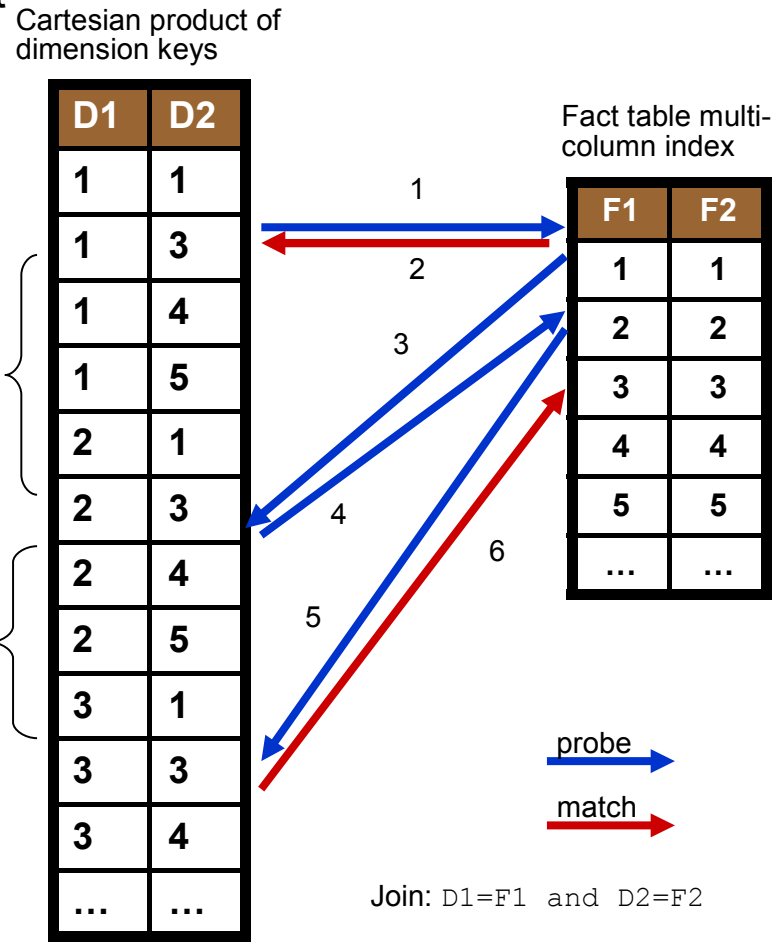
Zigzag join

10 million combinations!



Only 1,000 of them are represented in the fact table

Pre-requisite: A multi-column index on the fact table on columns that join with the dimensions

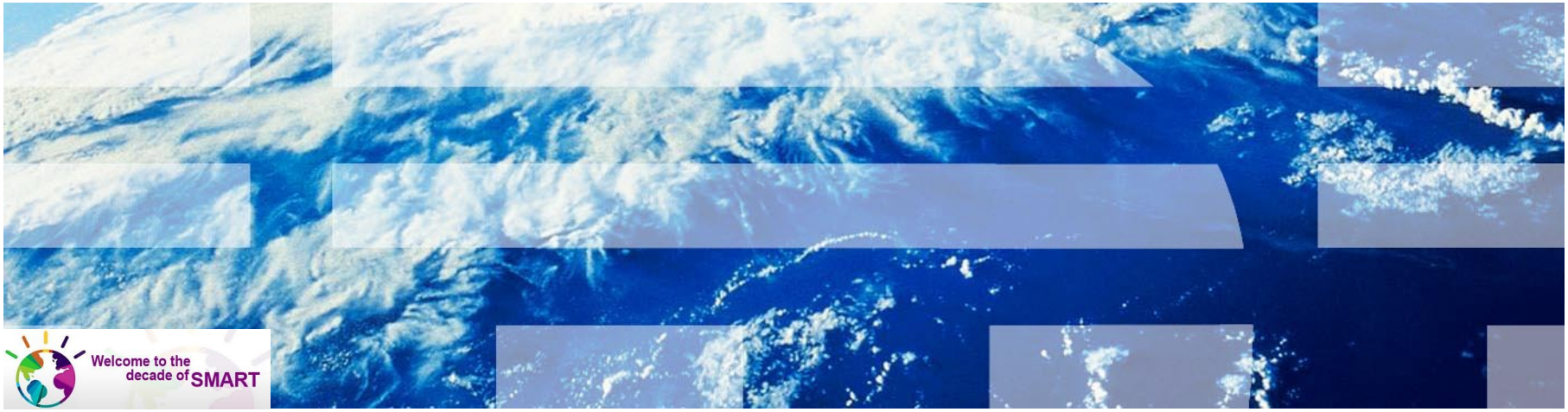


Join: `D1=F1 and D2=F2`

IBM Software

Information Management

DB2 10.1 LUW Feature Enhancements



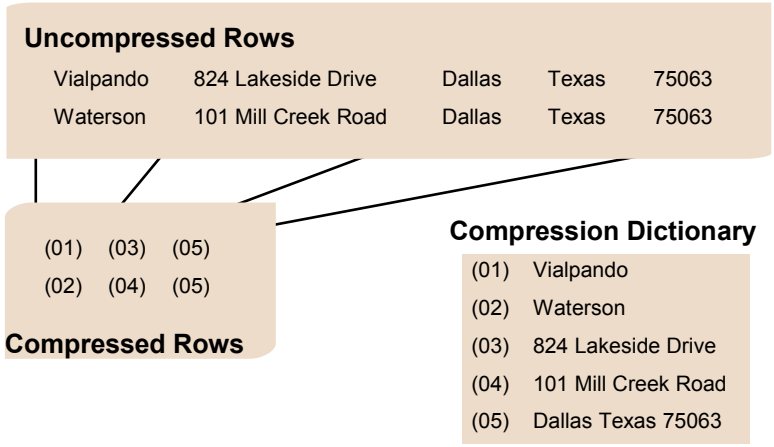
DB2 Deep Compression Enhancements
Feature highlights

- **Dictionary based – symbol for compressing or decompressing data rows**
  - Lempel-Ziv (LZ) based algorithm creates a static dictionary based at the ***table level***
    - A single table dictionary is stored within the table object for recurring large domain patterns **New!**
  - Compression is also done adaptively at the ***page level***
    - Smaller page stored dictionaries are for the ever changing page level patterns
- **Data resides compressed on pages**
  - On the table space disk – giving significant I/O bandwidth savings
  - In buffer pools – giving memory savings & improvement in performance
  - In recovery log disk & archive log tape
- **Compression is easy to implement and use**
  - DBA just uses these keywords: **`COMPRESS YES`**
  - Compression done during insert, update, import or load
  - Table dictionary is built on the fly with Automatic Dictionary Creation
  - Page dictionaries are built adaptively any time repeating patterns are detected
- **Compression done in many ways:**

Data 2 levels | **Indexes** 3 techniques | **XML XDAs** | **Inline LOBs** | **Temp tables** 5 usages | **Replicated tables** | **Log archives** **New!**

DB2 Deep Compression – a Recap
Static (or classic) Row Compression for data

- **Table level row compression for data**
  - There is only one dictionary for the entire table or table partition
  - Contains values that appear frequently across the larger domain of the entire table
- **Uses a Lempel-Ziv algorithm to create a compression dictionary**
- **Is called "static" because the dictionary is created in one of two ways**
  1. By default - with Automatic Dictionary Creation "on the fly" after a 1 MB sample is detected
     - This is the easiest to implement
     - But gives up some compression benefits
  2. Purposefully - with a table REORG or an on line table move
     - This gives the best over all table level, large domain compression
     - But requires DBA work to attain



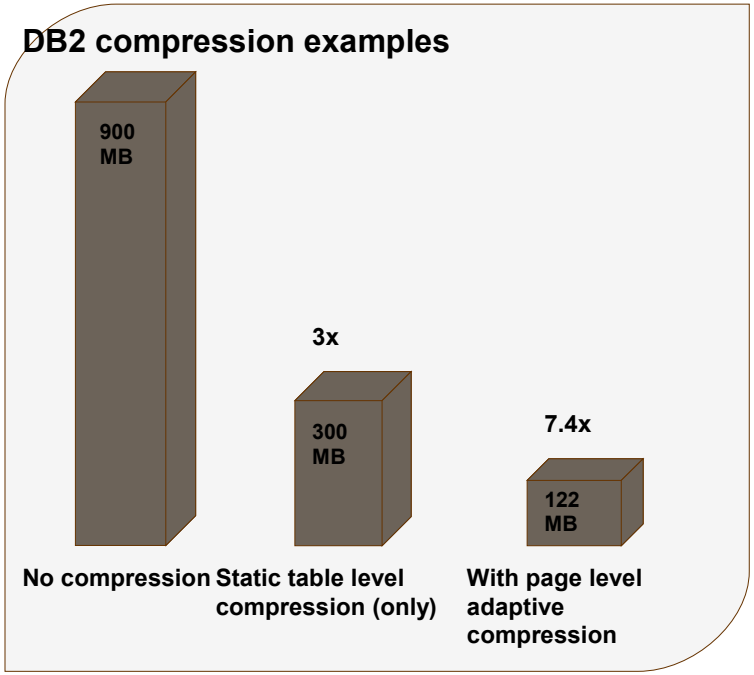
To enable static-only compression on a table:

```
CREATE TABLE CUSTOMER_INFO …
   COMPRESS YES STATIC;

ALTER TABLE CUSTOMER_INFO …
   COMPRESS YES STATIC;
```

DB2 Deep Compression Enhancements
Adaptive Row Compression for data **New!**

- **Page level row compression for data, compliments table level compression**
  - A dictionary per page is created for this smaller domain
    - Adapts for changing data characteristics over time
  - Compression rates improve as data grows and changes
    - Improvements over table level (static only) compression are usually much higher
    - Occurs during: inserts, updates, imports, loads, reorgs, table-moves and redistributes
- **Key point: table REORGs are not required to maintain high compression**



Adaptive compression on a table is the default if using compression. Here are ways to enable this:

```
CREATE TABLE CUSTOMER_INFO …
   COMPRESS YES;

ALTER TABLE CUSTOMER_INFO …
   COMPRESS YES;
```

Or:

```
CREATE TABLE CUSTOMER_INFO …
   COMPRESS YES ADAPTIVE;

ALTER TABLE CUSTOMER_INFO …
   COMPRESS YES ADAPTIVE;
```

DB2 Deep Compression Enhancements
Archived Logs New!

- **Archive logs compressed at time of archive movement**
  - The database manager will automatically compress log extents when they are moved from the active log path to the archive location
  - Upon retrieval (e.g. during ROLLBACK and ROLLFORWARD) the database manager automatically expands compressed log files when they are moved from the archive into the active or overflow log paths
- **Enabled independently for primary or secondary archive methods**
  - Accomplished with database parameters
  - Archive locations must be disk, TSM or vendor
- **Will affect all objects in the archive log**
  - Objects not compressed will be compressed
  - Data already compressed will not necessarily experience more

```
UPDATE DB CFG FOR TDB2 USING LOGARCHMETH1 DISK:/vol_auxillary1/archive/db2
UPDATE DB CFG FOR TDB2 USING LOGARCHCOMPR1 ON
```

```
UPDATE DB CFG FOR TDB2 USING LOGARCHMETH2 DISK:/vol_auxillary2/archive/db2
UPDATE DB CFG FOR TDB2 USING LOGARCHCOMPR2 ON
```

DB2 Deep Compression Comparison
DB2 vs. competition – DB2 delivers superior compression

Compression Feature	DB2 10.1	Oracle 11g R2	SQL Server
Table – general	✓	✓	✓
• Table – large domain pattern matching	✓	✗	✗
• Table – page (block) pattern matching	✓	✓	✓
• Table – column value	✓	✓	✓
• Table – multi-column value	✓	✓	✓
• Table – substring value	✓	✗	✗
• Table – XML (in-line and XDA)	✓	✗	✗
• Table – LOBs in-line	✓	✗	✗
Index – general	✓	✓	✓
• Index – prefix	✓	✓	✓
• Index – variable slot (dynamic prefix)	✓	✗	✗
• Index – rid list	✓	✗	✗
User temp – CGTT & DGTT	✓	✗	✗
System temp – sort overflow	✓	✗	✗
System temp – join temps	✓	✗	✗
System temp – aggregate temps	✓	✗	✗
Replication	✓	✓	✗
Archive log	✓	✓	✗

DB2 Workload Manager (WLM) Enhancements
The WLM Dispatcher - key concepts

- **The WLM Dispatcher** New!
  - Is built-in technology to allocate CPU resources to work being executed in WLM
  - Controls CPU resources by using relative % “shares” and actual % “limits”
    - The “shares” approach maximizes usage of CPU
    - The “limit” approach ensures free CPU for other processing needs
  - Is self contained, platform independent, flexible and easy to implement
- **Soft CPU Shares**
  - Relative CPU % usage that is enforced only when demand exceeds capacity
  - Gives the service class the ability to consume more than its share of CPU resources
    - The “share” amount is a maximum usage by percent, but only when the system is fully utilized
  - For high priority activities with preferential treatment
- **Hard CPU Shares**
  - Relative CPU % usage is always enforced
  - Caps the service class ability to consume CPU resources by share
    - The “share” amount is the maximum usage by percent, regardless of system utilization
  - For lower priority activities (especially if they are high impact)
- **CPU LIMIT**
  - Actual maximum CPU % resource usage, regardless of other activity
  - Can co-exist with Soft CPU Shares or Hard CPU Shares

DB2 WLM Service Class Enhancements
Examples of defining a service class using share New!

```
CREATE SERVICE CLASS "Main_SC"
    HARD CPU SHARES 3000
    PREFETCH PRIORITY MEDIUM
    BUFFERPOOL MEDIUM;
```

```
CREATE SERVICE CLASS "Low_SC"

    UNDER "Main_SC"
    HARD CPU SHARES 1000
    PREFETCH PRIORITY LOW
    BUFFERPOOL LOW;
```

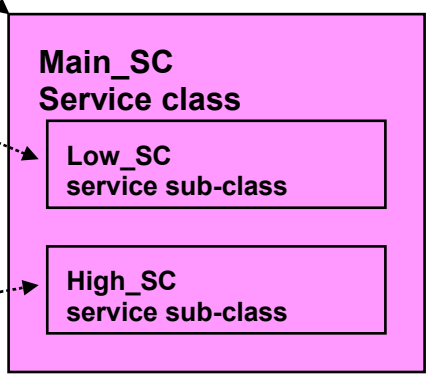
```
CREATE SERVICE CLASS "High_SC"

    UNDER "Main_SC"
    SOFT CPU SHARES 6000
    PREFETCH PRIORITY HIGH
    BUFFERPOOL HIGH;
```

"CPU shares" are controlled via the WLM Dispatcher

NOTE: There are four new DBM configuration parameters that will control the dispatching services in your databases:

WLM_DISPATCHER: Starts/stops dispatcher services
WLM_DISP_CONCUR: Controls thread concurrency
WLM_DISP_CPU_SHARES: Enables/disables CPU shares
WLM_MIN_UTIL: Control minimum CPU utilization



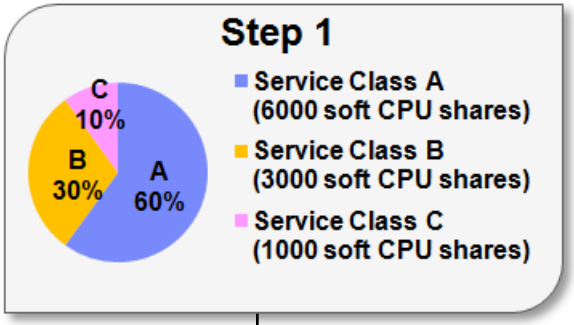
Note: Any activities that end up running in a super service class (in this example Main_SC) will actually run under its default service subclass that uses all of its service class settings.

DB2 WLM Service Class Enhancements
WLM Dispatcher usage example

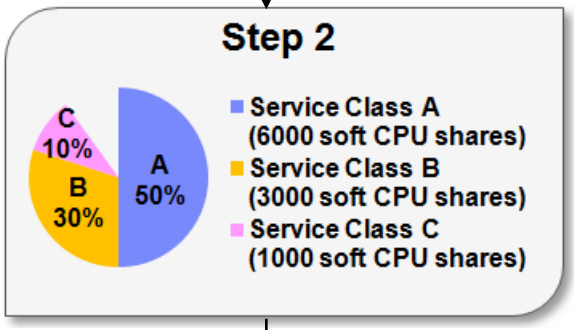
Soft CPU Shares Example

Service class A starts out using its allocation of shares, which proportionally is 60%. The others are 30% and 10%.

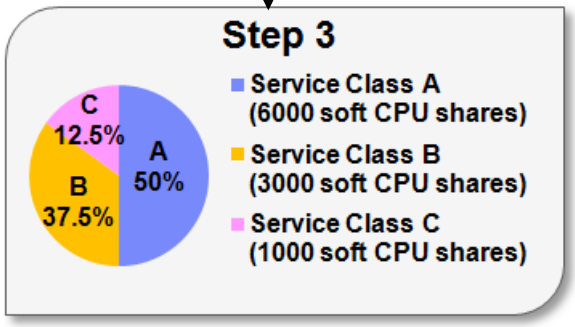
The CPU is fully utilized.



Service class A doesn't have enough work to utilize all its shares and begins to give them up.



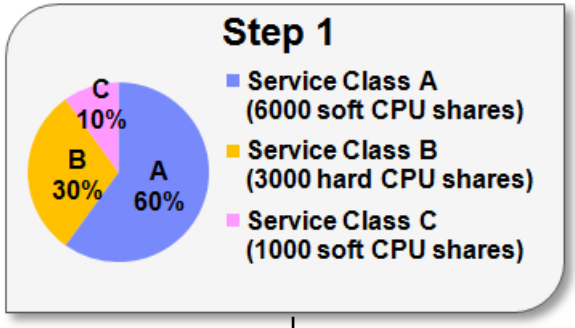
Service classes B and C use soft shares, so they take up the slack and split those shares from A.



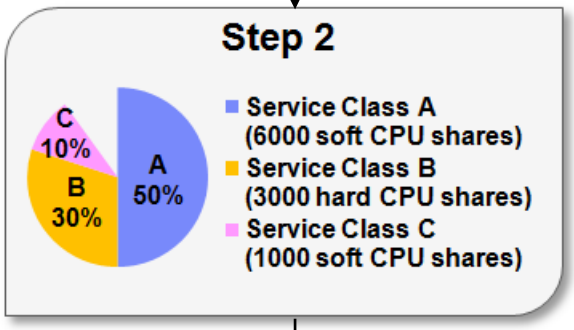
Hard CPU Shares Example

Service class A starts out using its allocation of shares, which proportionally is 60%. The others are 30% and 10%.

The CPU is fully utilized.

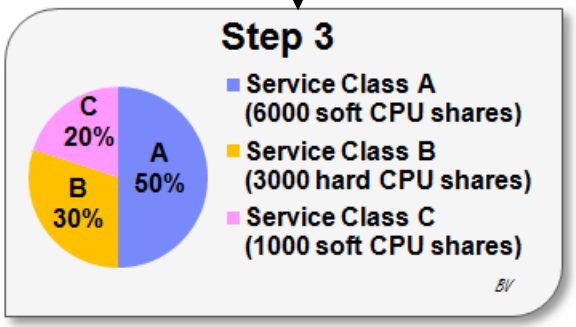


Service class A doesn't have enough work to utilize all its shares and begins to give them up.



Service class C uses soft shares and takes all the shares from A.

Service class B uses hard shares and was already at its limit, so it gets no more.



DB2 WLM Threshold Enhancements
Definition and customization options

- **A DB2 threshold is a way to define and enforce execution boundaries in the database**
  - Oversees different aspects within DB2, e.g.
    - connections, temp space used, SQL cost, rows returned, etc.
  - Provides support for common scenarios, e.g.
    - controlling “rogue” queries or concurrency control
- **Threshold domains:**
  - ☆ Database – applies to any activity in the database
  - ☆ Service Class – applies only to activities in this service class or superclass
  - ☆ Workload – applies to the workload or work action
  - New! → ☆ Statement – applies to a particular statement
- **Threshold enforcement:**
  - ☆ Database
  - ☆ Database partition
  - ☆ Workload
- **Threshold action options:**
  - ☆ Terminate the activity: **`STOP EXECUTION`**
    - An error is returned to the application
  - ☆ Allow activity to keep processing: **`CONTINUE`**
    - No error is returned to the application
  - ☆ Collect detailed activity information: **`COLLECT ACTIVITY DATA`**

A threshold for a particular SQL statement can be created. Example:
Putting a CPU threshold on the SQL statement:
“SELECT A, B from MYTABLE”.
A threshold violation will occur if that statement exceeds the CPU threshold.

DB2 WLM Work Action Set Enhancements
Using table space or storage group Data Tags New!

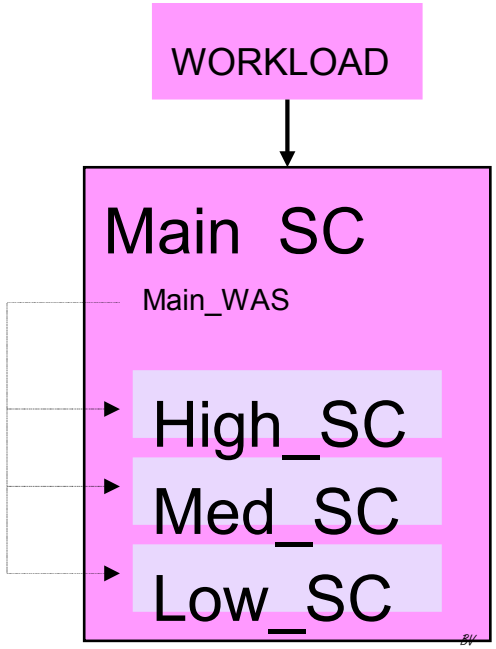
- **Alter or create a table space (or its storage group) to contain a data tag value**
- **Data tags can be used predictively by a work action set to map activities**

```
Alter tablespace TS_HIGH DATA TAG 1;
Alter tablespace TS_MED  DATA TAG 5;
Alter tablespace TS_LOW DATA TAG 9;
```

TS_HIGH TS_MED TS_LOW

```
CREATE WORK CLASS SET WCS_DATATAG
   (WORK CLASS HIGH_WC WORK TYPE DML DATA TAG LIST CONTAINS 1,
    WORK CLASS MED_WC WORK TYPE DML DATA TAG LIST CONTAINS 5,
    WORK CLASS LOW_WC WORK TYPE DML DATA TAG LIST CONTAINS 9);
```

```
CREATE WORK ACTION SET MAIN_WAS FOR SERVICE CLASS MAIN_SC
    USING WORK CLASS SET WCS_DATATAG
      (WORK ACTION MAP_HIGH ON WORK CLASS HIGH_WC
          MAP ACTIVITY TO HIGH_SC,
       WORK ACTION MAP_MED ON WORK CLASS MED_WC
          MAP ACTIVITY TO MED_SC,
       WORK ACTION MAP_LOW ON WORK CLASS LOW_WC
          MAP ACTIVITY TO LOW_SC);
```



The work action set Main_WAC is on the super service class Main_SC.

It maps activities to a different sub-service class depending upon the data tag in the table spaces those activities are accessing.

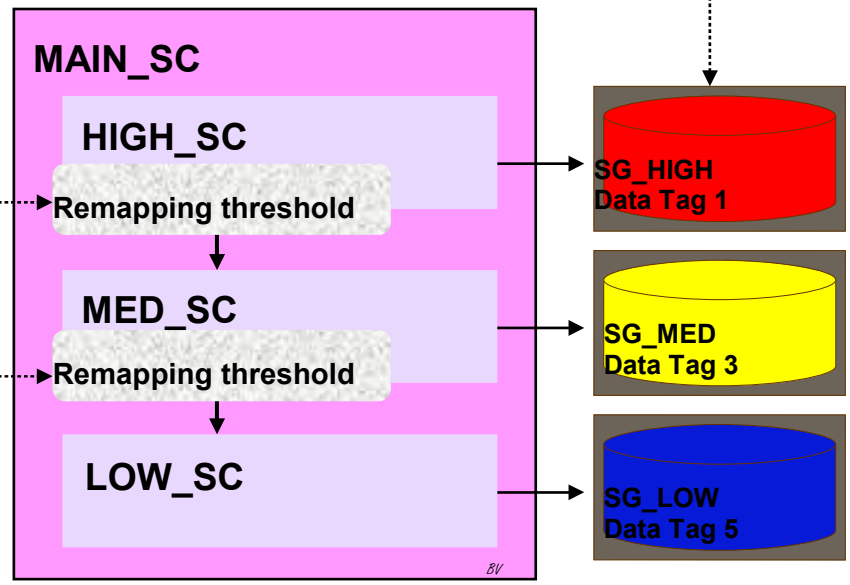
DB2 WLM Threshold Enhancements
Data Tag remapping using thresholds New!

- **Similar to priority aging, only the threshold is not based on aging**
- **Thresholds use "Data Tags" on tables spaces (or their storage groups) to reactively remap running activities**

```
Alter stogroup SG_HIGH DATA TAG 1;
Alter stogroup SG_MED  DATA TAG 3;
Alter stogroup SG_LOW DATA TAG 5;
```

```
CREATE THRESHOLD REMAP_HIGH_TO_MED
    FOR SERVICE CLASS HIGH_SC
    UNDER MAIN_SC ACTIVITIES
    ENFORCEMENT DATABASE PARTITION
    WHEN DATATAGINSC IN (3,5)
    REMAP ACTIVITY TO MED_SC;
```

```
CREATE THRESHOLD REMAP_MED_TO_LOW
    FOR SERVICE CLASS MED_SC
    UNDER MAIN_SC ACTIVITIES
    ENFORCEMENT DATABASE PARTITION
    WHEN DATATAGINSC IN (5)
    REMAP ACTIVITY TO LOW_SC;
```



Oracle Compatibility Enhancements
Objects that run without change in DB2 – new PL/SQL features

Oracle object	DB2 object	Comment
Package	Module	• `CREATE PACKAGE` statement accepted by DB2, but a DB2 module is the result object created • DB2 modules can be altered to add or drop objects within them, unlike Oracle packages
Procedure	Stored Procedure	
Function	User Defined Function (UDF)	
Trigger	Trigger	

Other PL/SQL and SQL features and behavior:

Public synonyms	Parameterized timestamps	Data type anchoring	Call statements with parameters
Cursor data types	Autonomous transactions	Implicit casting	Call statement with defaults
Constants	Create or replace	NEXTVAL / CURVAL	Positional call statements
MINUS	Statement concentrator	Unnamed inline views	Created global temporary tables
FORALL IN	BULKCOLLECT INTO	Declared types *	Declared procedures *
… and more		New!	

* Declared types and declared procedures are local to a compound PL/SQL statement and are not created in the system catalog.

Oracle Compatibility Enhancements
New Oracle built-in functions

Built-in Function Type	Examples
Conversion and Formatting	TO_CHAR, TO_DATE, TO_TIMESTAMP, TO_NUMBER, TO_CLOB, ROUND, VARCHAR_FORMAT, HEX_TO_RAW, INSTRB *, TO_SINGLE_BYTE ** New!
Datetime arithmetic	ADD_MONTHS, DAYNAME, LAST_DAY, MONTHS_BETWEEN,MONTHNAME, NEXT_DAY, ROUND_TIMESTAMP, TRUNC_TIMESTAMP, TIMESTAMP_FORMAT, TO_DATE, TIMESTAMPDIFF *** New!
String manipulation	INITCAP, RPAD, LPAD, INSTR, REVERSE, LOCATE_IN_STRING, TRUNCATE, TRUNC, LISTAGG, SUBSTR2, LTRIM, RTRIM, MOD
Misc.	DECODE, NVL, NVL2, LEAST, GREATEST, BITAND, EXTRACT, UNNEST

Note: DB2 already had a very diverse and rich set of functions. These examples are showing the additional ones created to support Oracle compatibility for functions of a particular name that were previously only in Oracle.

* INSTRB returns the starting position, in bytes, of another string
** TO_SINGLE_BYTE converts multi-byte strings to single-byte strings
*** TIMESTAMPDIFF returns the time interval of the difference between two different timestamps

Autonomic Computing Enhancements
DB2 automatic storage implementation

1. Enabled by default when you create a database in DB2 10.1:

CREATE DATABASE …

Note: The default storage group created in an automatic storage database is called IBMSTOGROUP

For migrations from previous DB2 versions, add a storage group to enable it:

CREATE STOGROUP [sto-grp-name] ON [path/drive]…

New!

New way to add automatic storage to a non-automatic storage database.

2. Create table space with the following:

CREATE TABLESPACE … MANAGED BY AUTOMATIC STORAGE … USING STORAGE GROUP…

- Under the covers, DB2 creates, names and sizes the data files needed to support the table space without any DBA intervention.
- Path or drive can be any number of subdirectories, mount points or drives
  - Path or drive list can be added to later with an **`ALTER or CREATE STOGROUP`** command
- DB2 manages all automatic storage containers within this path
  - Smart enough to make temporary table spaces SMS, all others DMS
  - Automatically stripes containers for you by your number of paths or drives
- Create other DMS or SMS table spaces outside automatic storage if desired
- Health Monitor indicators signal if paths or drives about to become full

Autonomic Computing Enhancements
Automatic Maintenance Queues **New!**

- Table function: **MON_GET_AUTO_RUNSTATS_QUEUE**
  - To monitor RUNSTATS in the maintenance queue, example usage:

```
SELECT QUEUE_POSITION, OBJECT_TYPE, OBJECT_STATUS, VARCHAR(OBJECT_SCHEMA, 10)
        AS OBJECT_SCHEMA, VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME
FROM  TABLE(MON_GET_AUTO_RUNSTATS_QUEUE()) AS T
ORDER BY QUEUE_POSITION ASC
```

```
QUEUE_POSITION OBJECT_TYPE OBJECT_STATUS        OBJECT_SCHEMA OBJECT_NAME
-------------- ----------- -------------------- ------------- -----------
             1 TABLE       JOB_SUBMITTED        TEST          EMPLOYEE
             2 TABLE       EVALUATION_PENDING   TEST          T1
             3 TABLE       EVALUATION_PENDING   TEST          BLAH
3 record(s) selected.
```

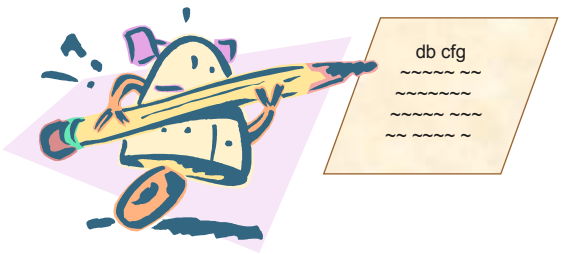
- Table function: **MON_GET_AUTO_MAINT_QUEUE**
  - To monitor all other auto maintenance activities, example output:

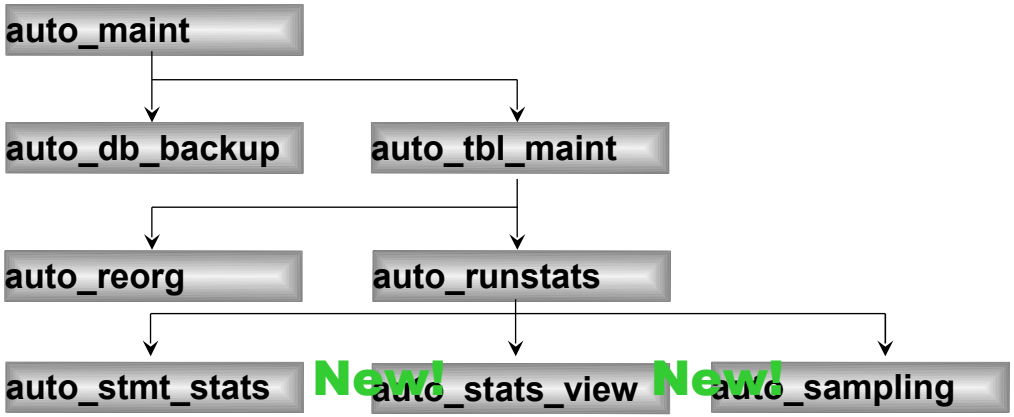
```
JOB_STATUS JOB_TYPE     OBJECT_TYPE OBJECT_NAME
---------- ------------ ----------- -----------
EXECUTING  REORG        TABLE       TP3

JOB_DETAILS
-------------------------------------------------------------
REORG INDEXES ALLOW WRITE CLEANUP ALL; RECLAIM EXTENTS

1 record(s) selected..
```

Autonomic Computing Enhancements
New Auto_% DB configuration parameters





AUTO_MAINT	Parent to all other auto_% parameters. Sets auto maintenance at a global level. Enabled: All recorded values for child parameters take effect.
AUTO_DB_BACKUP	Enabled: Allows for automatic BACKUP operations.
AUTO_TBL_MAINT	Parent parameter to all other (non backup) auto_% parameters. Enabled: All recorded values for child parameters take effect.
AUTO_REORG	Enabled: Allows for automatic table and index REORG operations.
AUTO_RUNSTATS	Enabled: Allows for automatic asynchronous RUNSTATS operations.
AUTO_STMT_STATS	Enabled: Allows for real-time, synchronous statistics gathering.
AUTO_STATS_VIEW	Enabled: Allows for statistical view statistic gathering.
AUTO_SAMPLING	Enabled: Allows for sampling of large tables.

New!
New!

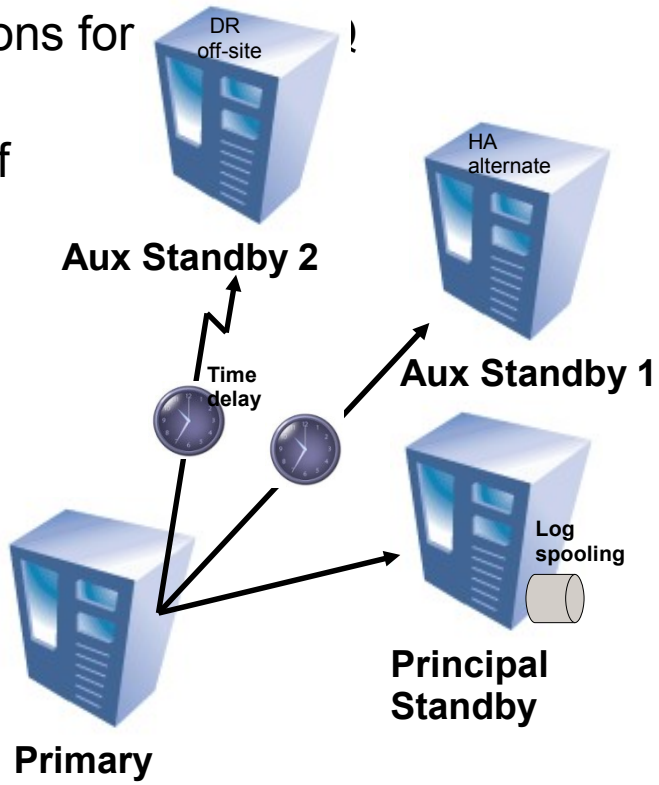
HADR Enhancements - Multiple standby servers
With optional replay time delay and log spooling

New! **Multiple standby**
- Have up to three standby databases in an HADR group
- All can be read only
- System is redundant even after a first failure
  - e.g. have redundant local HA standbys along with a remote DR standby
- Eliminates the need to have additional solutions for replication
- Set DB CFG: **`HADR_TARGET_LIST`**

New! **Replay time delay**
- Optionally introduce a delay into the replay of log records on any HADR standby database.
- This purposefully keeps a standby in an earlier version of the database
- This allows that standby to be used to recover to a point in time using only HADR and DB2 core restore / roll forward to a point in time technique

New!
- Set DB CFG: **`HADR_REPLAY_DELAY`**

Log spooling
- Spool logs on standby to prevent spikes on throughput
- Reduce impact on primary when log replay gets delayed on standby for any reason
- Set DB CFG: **`HADR_SPOOL_LIMIT`**



Event Monitor Enhancements
Change History Monitor

- **An event monitor is a created object in a DB2 database that:**
  - Returns information for events specified
  - Writes output to a TABLE, FILE or PIPE
- **Is "started" manually or automatically upon database activation**

New!

Event Type	When Collected
CHANGE HISTORY	Any changes to: DB & DBM config, registry, DDL or various utility executions
LOCKING	At lock timeout, lock wait and deadlock time
STATEMENTS	End of SQL statement
CONNECTIONS	End of Connection
DATABASE	Database deactivation
BUFFERPOOLS	Database deactivation
TABLESPACES	Database deactivation
TABLES	Database deactivation
Activities	WLM event (like completion of an activity in a Service class)
Statistics	WLM_COLLECT_INT setting
Threshold Violations	At detection of violation
Package cache	At detection of package cache changes
Unit of work	At completion of Unit of Work

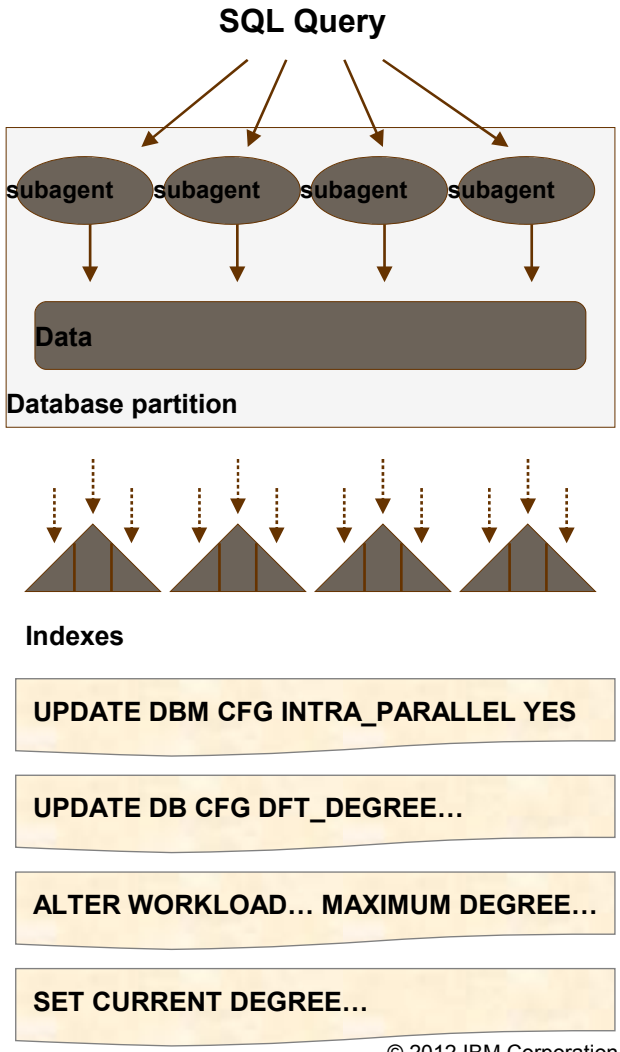
DB2 Optimizer Enhancements
Multi-core parallelism and parallel scan on partitioned indexes New!

Benefits:

- Provides maximum performance and scaling for the industry's ever increasing multi-core processor machines
- Eases set up and maintenance by avoiding the need to partition based on cores
- Improves performance of both OLTP and OLAP workloads on a single server
- Is easily enabled or disabled for particular workloads

How it works:

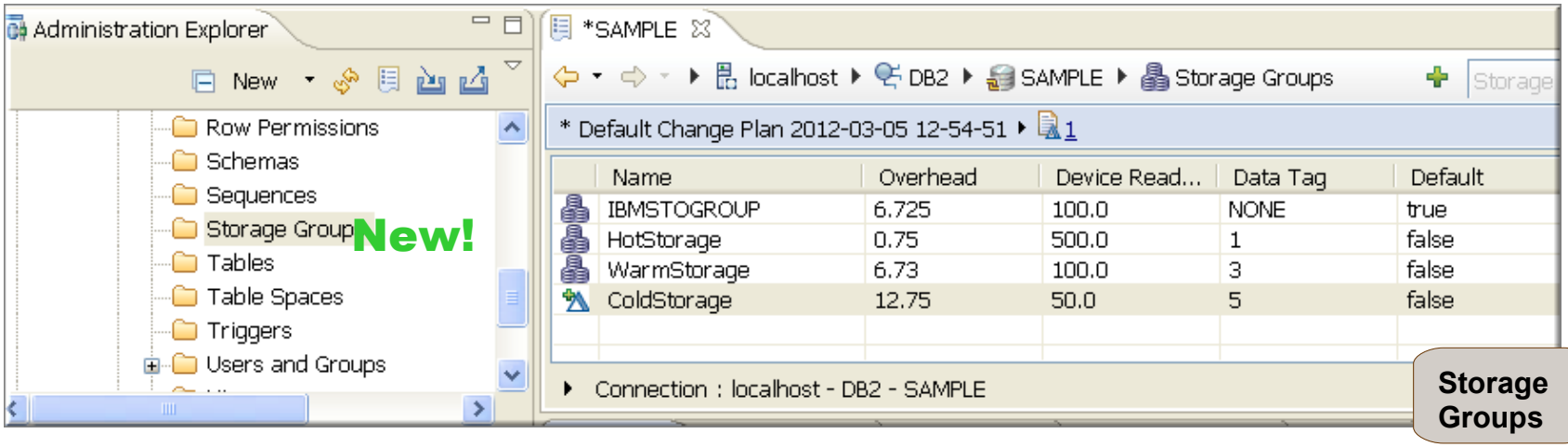
- Data is scanned by multiple subagents per partition regardless of cores per partition mapping
  - will rebalance agent usage automatically
- Partitioned indexes are parallel scanned
  - by key values and numbers of key entries
- A subagent is assigned a new range when it has completed its work on the current range
  - index partitions are scanned sequentially with subagents potentially scanning different index partitions at any point in time without waiting for each other.
  - only the subset of index partitions that is relevant to the query

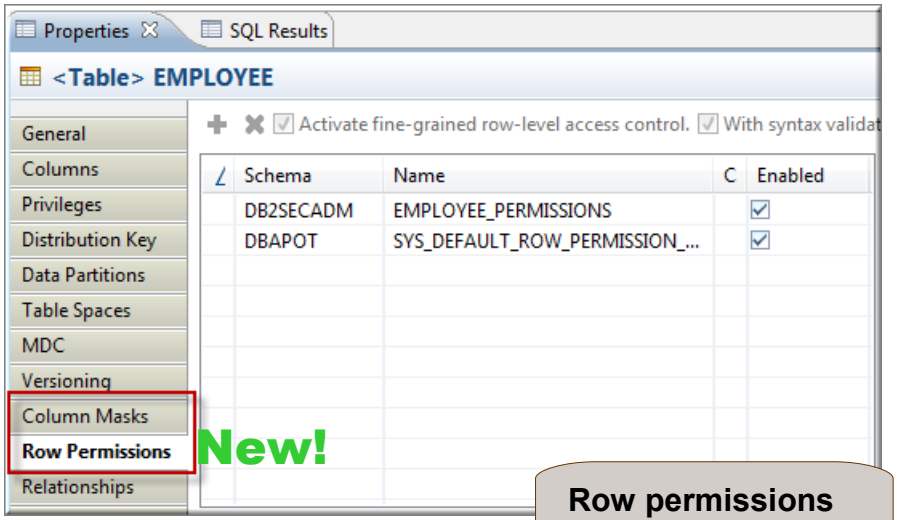


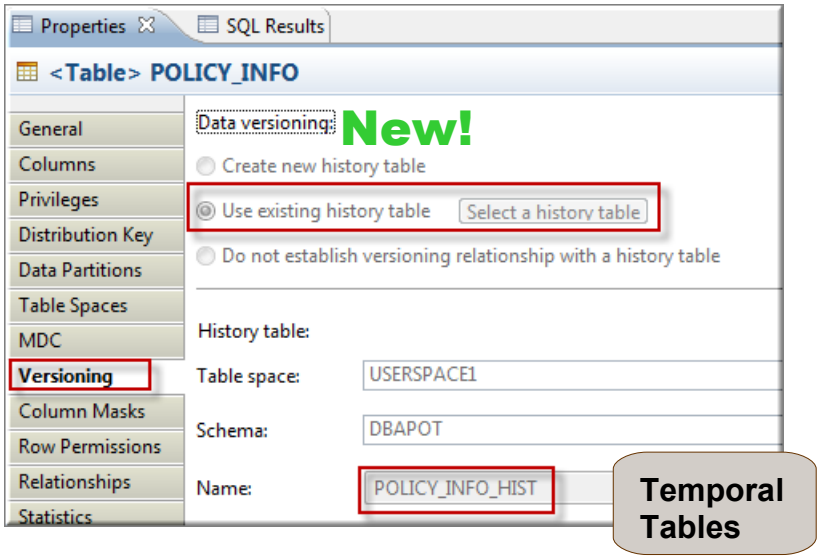
pureScale Enhancements New!
Increases ability to meet SLAs and add or remove capacity

- **pureScale feature included in regular edition install of DB2 10.1**
  - DB2 9.8 was previously necessary for a pureScale database installation
  - Now pureScale is a licensed option is for Workgroup, ESE and AESE editions
- **Features now supported in pureScale for DB2 10.1:**
  - Workload manager
  - Range partitioning
  - db2val command (to verify install validity)
- **Apply fix packs across all hosts with new –p parameter**
- **RoCE networks supported for AIX**
- **New Table function `MON_GET_GROUP_BUFFERPOOL`**
  - Helps determine properly sized group buffer pools
- **`CURRENT MEMBER` value is available for `ALTER TABLE` or `CREATE TABLE`**
  - Can utilize `CURRENT MEMBER` register to better control workloads
- **Other functionality supported in pureScale for DB2 10.1:**
  - 10-gigabit Ethernet
  - Multiple InfiniBand adapters and switches

Data Studio 3.1.1 supports new features from DB2 10.1
Examples: storage groups, RCAC and temporal tables







DB2 10.1 LUW New Features and Enhancements
Presentation by:

Burt Vialpando

IBM Executive IT Specialist, The Open Group Distinguished IT Specialist
IBM IT Certification Board
IBM Certified Database Administrator for DB2 10.1 for Linux, UN
Wi
C dv at ator for DB2 V9.7 fo
Linux, UNIX and Windows