DB2 10 FOR LUW
NEW FEATURES
A QUICK OVERVIEW FROM
A CUSTOMER PERSPECTIVE

JEAN-MARC BLAISE
IT Solution Architect

blaise@deltadb.com
About DELTA DB

- 15 consultants' company based in France (suburb of Paris).
- IBM Business Partner, reselling DB2 software & providing training and expertise in DB2 for z/OS and DB2 for LUW. Our goal: “to help you get the most of new DB2 features”.
- Active beta participant on IBM Programs:
  - DB2 LUW 10
  - OPM 5.1
  - OQT/OQWT 5.1
  - Data Studio 3.1.1
  - Merge Backup, Recovery Expert ...
- Jean-Marc BLAISE was recognized by IBM as an Information Champion for Data Management in 2009, 2010, 2011 and 2012 and is a member of DB2 and Optim Customer Advisory Councils.
AGENDA

- Brief recap on hardware evolution
- DB2 10 for LUW positioning
- Storage reduction
- Performance increase
- Multi-temperature storage
- Temporal tables and security
- Multi-standby HADR
In 20 years, CPU speed, Disk and Memory capacity have dramatically increased

- But Hard Drive (HD) latency stays limited to 70 ~ 80 Mb/s. Solid State Disk (SSD) emerging to supplement HD.
About Hardware Evolution (2/2)

- In 20 years, CPU speed, Disk and Memory capacity have dramatically increased

**RAM / CPU**

- CPU is now multi-sockets, multi-core, common to have a 4 core server for DB2 LUW
DB2 10 for LUW positioning (1/2)

Increase ROI
- Increase storage reduction: adaptive compression
- Increase CPU usage, decrease Memory usage: optimizer benefits from multi-core and reduces I/Os and sorts

Ease maintenance and reduce coding
- Increase Oracle compatibility from 95% to 98%
- Implement Temporal tables and Time Travel Query
- Implement Multi-temperature storage and increase WLM functionality
- New Security offer: RCAC – Row and Column Access Control
**DB2 10 for LUW positioning (2/2)**

**Merge code and offer extended architecture**

- pureScale renamed pureCluster as DB2 10 component in WSE, ESE and AESE, supports WLM and RP tables
- TSA as DB2 10 component on Linux/Unix
- HADR multi-standby – 1 primary standby + 2 auxiliary standby

**Very Attractive Packaging**

- RCAC and Temporal tables (Time Travel Query) in all releases
- DB2 pureCluster in WSE included (max 16 cores)
- DB2 AESE including 10 Infosphere Data Architect licences
- HADR in DB2 Express, extended to 8 Gb RAM
- Oracle Compatibility now in DB2 Express-C
AGENDA

INCREASE ROI

- Storage reduction
- Performance increase
Compression intended for Storage Reduction
Used mainly for I/O bound systems: Reporting, BI

Compression becomes “Adaptive”
- If compressed before DB2 10, need to `ALTER TABLE ... COMPRESS YES ADAPTIVE`
- `REORG TABLE ... RESET DICTIONARY` (preferably)

Principle is:
- To generate static dictionary as in 9.x
- Complement table compression with page level compression
Benefits: reduce I/Os (and cope with 70~80 Mb/s limit on HD), pages are compressed on disk and bufferpool

Biggest reduction is Data: 1500 Gb Data, 200 Gb Index from a test on a PeopleSoft DB using DB2 9.7

If you divide by 2~2.5 your DB size in 9.7, can expect 4~5 or more in DB2 10, depending on Data.

Need to REORG drastically reduced due to page level compression (example):
- MDC table with 80% compression ratio ... then 75% with next month data in 9.7, …
- 90% stable ratio in DB2 10 !!!
DB2 10 Storage Reduction (3/5)

Percentage Storage Space Saved

Table sizes in MB
More accurate ADMIN_GET_COMPRESS_INFO function

```sql
select PCTPAGESSAVED_CURRENT, PCTPAGESSAVED_STATIC,
PCTPAGESSAVED_ADAPTIVE from table (admin_get_tab_compress_info
('DB2INST1','CUST_NO_COMPRESSION'))
```

<table>
<thead>
<tr>
<th>PCTPAGESSAVED_CURRENT</th>
<th>PCTPAGESSAVED_STATIC</th>
<th>PCTPAGESSAVED_ADAPTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>82</td>
<td>91</td>
</tr>
</tbody>
</table>

```sql
select tabname, PCTPAGESSAVED, compression, rowcompmode, npages ... 
```

<table>
<thead>
<tr>
<th>TABNAME</th>
<th>PCTPAGESSAVED</th>
<th>COMPRESSION</th>
<th>ROWCOMPMODE</th>
<th>NPAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER_NO_COMPRESSION</td>
<td>0</td>
<td>N</td>
<td></td>
<td>1729</td>
</tr>
<tr>
<td>CUSTOMER_ST_COMPRESSION</td>
<td>83</td>
<td>R</td>
<td>S</td>
<td>295</td>
</tr>
<tr>
<td>CUSTOMER_AD_COMPRESSION</td>
<td>93</td>
<td>R</td>
<td>A</td>
<td>131</td>
</tr>
</tbody>
</table>

Additional storage reduction:

- Can compress archive logs (**logcompr1** DB CFG)
- Can eliminate potentially multi-column index not used anymore due to jump scan technology (see next)
COMPRESSION BENEFITS

- Great storage reduction. Reduced I/O, less pages to process
- Needs for REORGs dramatically reduced
- For Reporting on OLTP or BI databases
- Make an I/O application profile and plan reduction with a simple statement using table functions
- Activate compression on archive logs to gain further
Statistics maintenance improved using reduced resources

- Index sampling

  \[
  \text{index-sampling-options:} \\
  \text{INDEXSAMPLE \{BERNOULLI | SYSTEM\} (numeric-literal)}
  \]

- Automatic statistics on statistical views and automatic sampling (default OFF)

  - Automatic maintenance (AUTO_MAINT) = ON
  - Automatic table maintenance (AUTO_TBL_MAINT) = ON
  - Automatic runstats (AUTO_RUNSTATS) = ON
  - Real-time statistics (AUTO_STMT_STATS) = ON
  - Statistical views (AUTO_STATS_VIEWS) = OFF
  - Automatic sampling (AUTO_SAMPLING) = OFF
**DB2 10 Optimizer Features (2/8)**

- **Transparent** enhancements to query access plans (reduced I/Os and sortheap memory)
  - Partial Early Aggregation to start pre-aggregation asap
  - Partial Early Distinct to eliminate some duplicates
    
    ```sql
    SELECT DISTINCT ...
    FROM Newton.Table1, Newton.Table2
    WHERE c1 = c2
    ```

- More HSJOIN conditions (WHERE `c1 = Upper(c2)`)

![Diagram showing query execution plan with pUnique and HSJOIN conditions]
**Transparent** better use of composite indexes: Jump Scan

```
SELECT * FROM orders WHERE product_id = 10 AND order_amount = 898;
```

![Diagram showing the use of Jump Scan in DB2 10 Optimizer Features](image)
**Benefits** on JUMPSCAN technology

- Reduced I/Os
- Reduced storage
- Reduced performance impacts on writes
- Control index usage in access plans and in monitoring and potentially remove unused indexes

```sql
SELECT ... FROM TABLE(MON_GET_INDEX('DB2INST1','\', -2)) as T, SYSCAT.INDEXES AS S WHERE T.TABSCHEMA = S.TABSCHEMA AND T.TABNAME = S.TABNAME AND T.IID = S.IID
```

<table>
<thead>
<tr>
<th>INDSCHEMA</th>
<th>INDNAME</th>
<th>INDEX_SCANS</th>
<th>INDEX_JUMP_SCANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2INST1</td>
<td>PRD_I1</td>
<td>249</td>
<td>30</td>
</tr>
<tr>
<td>DB2INST1</td>
<td>STM_I1</td>
<td>332</td>
<td>0</td>
</tr>
</tbody>
</table>
DB2 10 Optimizer Features (5/8)

- **Transparent ZZJOIN technology for star schema queries**
  - Reduced I/Os if 2 dimensions at least used, with equi-joins and composite index on fact table

<table>
<thead>
<tr>
<th>Cartesian product of dimension keys</th>
<th>Fact table multi column index</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1 1</td>
<td>d2 1</td>
</tr>
<tr>
<td>1 1</td>
<td>1 3</td>
</tr>
<tr>
<td>1 4</td>
<td>1 5</td>
</tr>
<tr>
<td>2 1</td>
<td>2 3</td>
</tr>
<tr>
<td>2 4</td>
<td>2 5</td>
</tr>
<tr>
<td>3 1</td>
<td>3 3</td>
</tr>
<tr>
<td>3 4</td>
<td></td>
</tr>
</tbody>
</table>

Join: d1 = f1 and d2 = f2
**DB2 10 Optimizer Features (6/8)**

- **Explain plan displays ZZJOIN operators**
  
  ```
  2.6623e+06
  ZZJOIN
  (  5)
  7620.42
  5.37556
  +------------------+------------------+
  | 292.2 | 40000 | 0.227781 |
  | TBSCAN | TBSCAN | FETCH |
  | (  6) | (  9) | ( 13) |
  | 56.2251 | 7596.78 | 11.8222 |
  | 1 | 2.92 | 1.22778 |
  | /----++---\ |
  | 292.2 | 40000 | 0.227781 | 6.65576e+08 |
  | TEMP | TEMP | IXSCAN | TABLE: POPS |
  | (  7) | ( 10) | ( 14) | DAILY_SALES |
  | 30.4233 | 4235.52 | 9.93701 | Q3 |
  | 1 | 2.92 | 1 |
  ```

- **Explain can recommend composite index creation**

  Analysis of the query shows that the query might execute faster if an additional index was created to enable zigzag join. Schema name:... Column list:...

- **ZZJOIN can be combined with Jump Scan, access plan can use Partial Early ... Distinct or Aggregation ... all TRANSPARENT!**
- Multi-core parallelism can be activated – in precedence and dynamically - at workload, by AMIN_SET_INTRA_PARALLEL proc. DB2 rebalances between agents to make them process data equally

- Table data or index partitions can be scanned in parallel

- Index prefetching enhanced even if index not well organized, benefits is reduced need to REORG TABLE ... INDEXES ALL

- Profile enhancement: inexact matching and registry options

```xml
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>
  <!--Global section -->
  <STMTMATCH EXACT='FALSE'/>
  <!-- Statement level profile -->
  <STMTPROFILE ID='S1'><STMTMATCH EXACT='FALSE'/>
  <STMTPROFILE ID='S2'>
    <!--STMTKEY--><![CDATA[select * from T1 where c1 in( 10,20)]]></STMTKEY>
    <OPTGUIDELINES>
      <REGISTRY><OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='YES'/></REGISTRY>
    </OPTGUIDELINES>
  </STMTPROFILE>
</OPTPROFILE>
```
OPTIMIZER BENEFITS

- Less resources used on I/O, memory
- Faster statistic maintenance
- More flexible for mixed workloads (Reporting on OLTP), enhanced parallelism, at core level
- Transparent to the user – no need to rewrite statements, help in explain plans for indexes
- Better control of optimization profiles – flexibility, if needed
AGENDA

EASE ADMINISTRATION AND DEVELOPMENT

- Multi-temperature storage
- Temporal tables and RCAC security
DB2 Multi-Temp. Storage (1/7)

- Capability to store data/index/Lobs and XML on different groups of storage
- Storage should reflect data aging (hot, warm, cold data, …). Multi-Temperature storage is for high volume data, DSS and can be even used for OLTP.
- Storage could be different type, could use SSD – much faster transfer rate – for hot data
- Data rotation and placement is smooth, DB2 is responsible for that, not the storage
- Is very simple for the DBA to setup, flexible, based on database model
DB2 Multi-Temp. Storage (2/7)

- ACCESSIBLE, ONLINE
- ARCHIVED DATA Optim Data Growth

Volume of data:
- Hot data: Current Month
- Warm data: 2 previous years
- Cold data: 8 previous years
- Archived data: Dormant data

Age of data
CREATE STOGROUP SG_HOT ON <pathSSD> ...
CREATE STOGROUP SG_WARM ON <pathHD> ...
CREATE TABLESPACE TS2011Q1 USING SG_HOT
CREATE TABLESPACE TS2010Q4 USING SG_WARM
CREATE TABLE SALES ... PARTITION BY RANGE (DATE) ...
DB2 Multi-Temp. Storage (4/7)

- Uses fully Automatic Storage and Range Partition Tables (ESE minimum).
- Rotation as simple as
  \[ \text{ALTER TABLESPACE TS2011Q1 USING SG_WARM} \]
- Automatic rebalance (extent movement) can be controlled
  \[ \text{ALTER TABLESPACE TS2011Q1 REBALANCE SUSPEND/RESUME} \]
- However, cannot yet change dynamically bufferpool assigned to tablespace, to control storage access in bufferpools and give more memory to hot data.
- Another question is, how to avoid changing oldest data? HOT or WARM Data might be updated, COLD data should NOT be modified.
Answer is use WLM that has been enhanced and can use DATA TAB priorities (1 – high priority - to 9 – low priority).

CREATE STOGROUP SG_HOT ... DATA TAG 1
CREATE TABLESPACE TS2011Q1 (inherits stogroup DATA TAG)
ALTER STOGROUP SG_WARM DATA TAG 4
ALTER TABLESPACE TS2010Q4 DATA TAG 9 (tablespace can have specific DATA TAG)

Activate new WLM features at instance level (WLM integrates CPU resource control, independently from OS WLM features CPU usage – limits and hard shares, soft shares).

WLM dispatcher enabled (WLM_DISPATCHER) = YES
CREATE THRESHOLD REMAP_MEDIUM_TO_LOW
FOR SERVICE CLASS WLM_MEDIUM UNDER APPLI_SC ACTIVITIES
ENFORCEMENT DATABASE PARTITION
WHEN DATATAGINSC IN (7, 8, 9) REMAP ACTIVITY TO WLM_LOW
MULTI-TEMP. STORAGE BENEFITS

- Store data based on priority
- Ease of Maintenance
- Can leverage SSD and gain extra-speed for the most important data
- Can control with WLM resource priority (prefetch priority, bufferpool priority, cpu control, prevent execution, ...)

blaise@deltadb.com © 2012
DB2 Temporal Tables & RCAC (1/7)

- Capability to keep history of table rows, managed internally by the DB2 LUW engine
- Business application can manage dates and DB2 LUW engine split rows to track changes, ex. Pricing policy
- Reduced application coding, simplified, replace triggers that could have been used by the application
- Capability to address rows in tables at a point in time in your SQL, get the vision of What and When
- Security greatly enhanced, row modifications “tracing” but also access permission introduced
- Simpler than LBAC, more flexible and enhanced control access to rows
- Column masking depending on the user profile
DB2 Temporal Tables & RCAC (2/7)

**SYSTEM TEMPORAL TABLES**

1) CREATE OR ALTER TABLE with additional columns

   system_begin TIMESTAMP(12) NOT NULL
   GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
   system_end TIMESTAMP(12) NOT NULL
   GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
   trans_start TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION
   START ID IMPLICITLY HIDDEN,
   PERIOD SYSTEM TIME (system begin, system end)

2) CREATE TABLE loan_accounts_stt_history
   LIKE loan_accounts_stt

3) Link the TABLE with its history
   ALTER TABLE loan_accounts_stt ADD VERSIONING USE HISTORY
   TABLE loan_accounts_stt_history

**APPLICATION TEMPORAL TABLES**

1) CREATE OR ALTER TABLE with additional columns

   bus_begin DATE NOT NULL,
   bus_end DATE NOT NULL,
   PERIOD BUSINESS_TIME (bus_begin, bus_end)

2) CREATE or RECREATE UNIQUE index to avoid insertions of overlapping business times

   CREATE UNIQUE INDEX index_loan
   ON loan_accounts_att (account_number,
   BUSINESS_TIME WITHOUT OVERLAPS)
DB2 Temporal Tables & RCAC (3/7)

SYSTEM TEMPORAL TABLES

UPDATE loan_accounts SET balance = 603750
WHERE account_number = 2111

APPLICATION TEMPORAL TABLES

UPDATE loan_accounts_all
FOR PORTION OF BUSINESS_TIME FROM '2010-03-01' TO '2010-03-31'
WHERE account_number = 2111
DB2 Temporal Tables & RCAC (4/7)

SYSTEM TEMPORAL TABLES

![Query Example]

```sql
SELECT account_number, rate_of_interest, system_begin, system_end
FROM loan_accounts;
WHERE account_number = 2111;
```

<table>
<thead>
<tr>
<th>account_number</th>
<th>loan_type</th>
<th>rate_of_interest</th>
<th>balance</th>
<th>system_begin</th>
<th>system_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>2111</td>
<td>A21</td>
<td>9.00</td>
<td>559500</td>
<td>2009-11-30</td>
<td>2011-01-31</td>
</tr>
<tr>
<td>2111</td>
<td>A22</td>
<td>8.00</td>
<td>E55555</td>
<td>2009-03-01</td>
<td>2009-11-30</td>
</tr>
</tbody>
</table>

APPLICATION TEMPORAL TABLES

![Query Example]

```sql
SELECT account_number, bus_begin, bus_end
FROM loan_accounts_att;
WHERE bus_begin < '2010-01-01'
AND bus_end > '2009-08-31';
```

<table>
<thead>
<tr>
<th>account_number</th>
<th>loan_type</th>
<th>rate_of_interest</th>
<th>balance</th>
<th>bus_begin</th>
<th>bus_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>2111</td>
<td>A21</td>
<td>9.00</td>
<td>559500</td>
<td>2009-11-01</td>
<td>2013-11-01</td>
</tr>
<tr>
<td>2112</td>
<td>A10</td>
<td>12.00</td>
<td>450320</td>
<td>2010-01-02</td>
<td>2013-02-02</td>
</tr>
<tr>
<td>2113</td>
<td>A21</td>
<td>9.00</td>
<td>100000</td>
<td>2010-02-06</td>
<td>2010-12-30</td>
</tr>
<tr>
<td>2114</td>
<td>A15</td>
<td>10.00</td>
<td>200000</td>
<td>2010-02-07</td>
<td>2011-08-31</td>
</tr>
</tbody>
</table>
DB2 Temporal Tables & RCAC (5/7)

- Security greatly enhanced, row modifications “tracing” but also data control access introduced
- Simpler than LBAC, more flexible and easy control access to rows, column masking for column protection
  
  ```sql
  CREATE PERMISSION ...
  CREATE MASK ...
  ```

- All depending on the user profile - new functions. Complements Dictionary control, roles and authorities (DATAACCESS)

- User can update/insert data that he can SELECT back or delete data than he can SELECT
CREATE PERMISSION ROW_ACCESS ON PATIENT
FOR ROWS WHERE
    (VERIFY_ROLE_FOR_USER(SESSION_USER,'PATIENT') = 1
    AND PATIENT.USERID = SESSION_USER) OR
    (VERIFY_ROLE_FOR_USER(SESSION_USER,'PCP') = 1
ENFORCED FOR ALL ACCESS
ENABLE;

ALTER TABLE PATIENT ACTIVATE ROW ACCESS CONTROL;

CREATE MASK ACCT_BALANCE_MASK ON PATIENT
FOR COLUMN ACCT_BALANCE
RETURN
    CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1
    THEN ACCT_BALANCE ELSE 0.00
    END
ENABLE;

ALTER TABLE PATIENT ACTIVATE COLUMN ACCESS CONTROL;
Temporal Tables & RCAC BENEFITS

- Temporal tables enable Time Query Travel, managed by DB2 LUW and application (bus_begin/bus_end). Can have both System and Application Temporal tables at the same time.

- Reduced application logic, simpler queries.

- RCAC leverages roles, implementation is simple and DATA is protected whatever interface & tools you use.

- Remember this is in all releases from DB2 Express Edition!!!
AGENDA

EXTENDED ARCHITECTURE

- Multi-hadr standby
- HADR can use now up to 3 standby servers
- The primary standby server can be automated by TSA
- The 2 other auxiliary standby servers can only be in SUPERASYNC mode (introduced in 9.5 FP8, 9.7 FP5). The standby PULL the transactions from the production server, whereas the primary PUSH transactions to the primary standby
- Can READ on ALL standby servers, `db2set DB2_HADR_ROS=ON`
- Can DELAY replay, only in SUPERASYNC mode, `hadr_replay_delay = N s (DB CFG)`
DB2 Extended Architecture (2/3)
HADR Multi-Standby BENEFITS

- Can offload primary server by doing reads on multiple-standby
- Can have HA, fully automated by TSA, on primary site, and DR on standby sites
- Some new parameters in DB CFG, but things still SIMPLE

Remember this is in all releases from DB2 Express Edition !!!
DB2 10, a GREAT release !!!

- Lower administration requirements
- Better performance per core, Better compression, Better Workload Management
- Improved business critical reliability and availability, take care of your business, not the database server
- Easy to use, Easy to develop
- Broad solution platform, from DB2 10 Express-C to DB2 10 AESE … and Infosphere Warehouse 10 Advanced Enterprise Edition
DB2 10, some links !!!

DB2 10 Trial

Data Studio 3.1.1

Toad 5.1 for DB2 LUW
http://toadfordb2.com/index.jspa

THANK YOU !!!