DB2 UDB for z/OS Version 8: Application and SQL Related Enhancements  
NC ITS - February 9, 2006

Charles Lewis  
DB2 UDB for z/OS  
lewisc@us.ibm.com

Disclaimer and Trademarks
Information contained in this material has not been submitted to any formal IBM review and is distributed on "as is" basis without any warranty either expressed or implied. Measurements data have been obtained in laboratory environment. Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion. You should not rely on such information to make business plans. The use of this information is a customer responsibility.

IBM MAY HAVE PATENTS OR PENDING PATENT APPLICATIONS COVERING SUBJECT MATTER IN THIS DOCUMENT. THE FURNISHING OF THIS DOCUMENT DOES NOT IMPLY GIVING LICENSE TO THESE PATENTS.

TRADEMARKS: THE FOLLOWING TERMS ARE TRADEMARKS OR ® REGISTERED TRADEMARKS OF THE IBM CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: AIX, AS/400, DATABASE 2, DB2, e-business logo, Enterprise Storage Server, ESCON, FICON, OS/390, OS/400, ES/9000, MVS/ESA, Netfinity, RISC, RISC SYSTEM/6000, iSeries, pSeries, xSeries, SYSTEM/390, IBM, Lotus, NOTES, WebSphere, z/Architecture, z/OS, zSeries,

The FOLLOWING TERMS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF THE MICROSOFT CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: MICROSOFT, WINDOWS, WINDOWS NT, ODBC, WINDOWS 95

For additional information see ibm.com/legal/copytrade.phtml
Agenda

- New "news" in 2006
- Brief "review" of DB2 v8 migration
- Scalability enhancements
- Availability enhancements
- SQL enhancements
- Utility and other miscellaneous topics
- Additional DB2 info resources

New IBM System z9 Integrated Information Processor (IBM zIIP)

- New specialty engine for the System z9 mainframe (planned for 2006) designed to help:
  - Customers integrate data across the enterprise
  - Improve resource optimization and lower the cost of ownership for eligible data serving workloads
- z/OS manages and directs work between the general purpose processor and the zIIP
  - Number of zIIPs per z9-109 not to exceed number of standard processors
  - No changes anticipated to DB2 for z/OS V8 applications
- DB2 for z/OS V8 will be first IBM exploiter of the zIIP with
  - System z9 109
  - z/OS 1.6 or later
  - DB2 for z/OS V8
DB2 V8 exploitation of IBM zIIP can add value to database workloads

- Portions of the following DB2 for z/OS V8 workloads may benefit from zIIP*
  1. ERP, CRM, Business Intelligence or other enterprise applications
     - Via DRDA® over a TCP/IP connection
  2. Data warehousing applications*
     - Requests that utilize complex star schema parallel queries
  3. DB2 for z/OS V8 utilities*
     - Internal DB2 utility functions used to maintain index maintenance structures

*The zIIP is designed so that a program can work with z/OS to have all or a portion of its enclave Service Request Block (SRB) work directed to the zIIP. The above types of DB2 V8 work are those executing in enclave SRBs, of which portions can be sent to the zIIP.

How does the zIIP work .....  

- The zIIP is designed so that a program can work with z/OS to have all or a portion of its enclave Service Request Block (SRB) work directed to the zIIP. The types of DB2 V8 work listed below are those executing in enclave SRBs, portions of which can be sent to the zIIP.

Example 1 = Distributed SQL requests (DRDA)
Queries that access DB2 for z/OS V8 via DRDA over a TCP/IP connection are dispatched within z/OS in enclave SRBs. z/OS directs a portion of this work to the zIIP.

Example 2 = Complex parallel query (BI)
Complex star schema parallel queries will now use enclave SRBs. z/OS directs a portion of this work to the zIIP.

Example 3 = DB2 utilities for index maintenance
DB2 utilities LOAD, REORG, and REBUILD will now use enclave SRBs for the portion of the processing that is related to index maintenance. z/OS directs a portion of this work to the zIIP.
Specialty engines

The IBM System z9 specialty engines can operate on the same machine together (shown at left, instances where zIIP can be employed)

Post GA Utility Enhancements

- Online CHECK INDEX
  - APARs PQ92749 (DB2 base) and PQ96956 (Utility Suite)
- Cross Loader support for > 32K LOBs
  - APAR PQ90263 (PTF available now for V7 and V8)
- LOAD/UNLOAD support for very large LOBs
  - Running prototype
  - Production code in unit test
  - APAR PK10278 for V7 and V8
- Data first claiming/draining
  - Greatly reduces the chances of a deadlock between SQL and utilities – now SQL and utilities will always claim/drain the data first, and then the index. This doesn't prevent any deadlocks -- if SQL accesses partition M, then tries for partition N while utilities does the reverse, there is still a potential deadlock among data partitions. ZPARM CLAIMDTA (default is NO)
  - APAR PK09781

More detail covered in utility enhancements
Installation and migration

- Migration
  - Compatibility Mode
  - Enabling New Function Mode
  - ENFM Jobs
  - New Function Mode
- Catalog changes

Migration to Version 8

Data sharing co-existence V7 - V8 CM

Fallback possible after successful CATMAINT

NO Fallback to V7 or CM after entering ENFM or NFM !!!
Modes of operation

**DSN1 - Compatibility Mode**
- Catalog and Directory converted to V8
- Catalog and Directory in Unicode
- New V8 functions available

**DSN1 - Enabling New Function Mode**
- Catalog and Directory converted to V8
- Catalog and Directory in Unicode accepting long names
- New V8 functions available

**DSN1 - New Function Mode**
- Catalog and Directory converted to V8
- Catalog and Directory in Unicode accepting long names
- New V8 functions available

---

**Installation Verification Procedure (IVP)**

- **Important** - You cannot run the Version 8 IVP jobs until DB2 is running in Version 8 new-function mode
- Run the Version 7 IVP jobs to verify a successful migration to Version 8 compatibility mode
- If migrating, the recommendation is to run portions of the sample applications from Version 7
  - Verifies the migration
  - Ensures that the old jobs work with Version 8 in new-function mode
- The Version 8 IVP jobs are created by the installation CLIST as part of Version 8 enabling-new-function mode
DB2 for z/OS Version 8 catalog - summary

- Longer names (128 byte varchar Unicode)
  - Table, view and alias
  - Column (30)
  - Schema
  - UDF, stored procedures, triggers, packages

- Some page sizes greater than 4 KB
  - 4K, 8K, 16K, 32K pages in DB2 catalog
  - DB2 allocates BPs for those new page sizes
  - When using data sharing, YOU must allocate GBPs for those BPs

- Indexes longer than 255 bytes
- Contains Unicode data

CCSID – Is it a problem?

- DB2 uses a set of three CCSIDs to describe data stored in DB2
  - EBCDIC, ASCII and Unicode
- DB2 supports specification of CCSIDs at DB2 subsystem level
  - DSNHDECP
    - Consistent across members of data sharing group
    - Once set they should not be changed
- General case, no data conversion for local applications
  - Terminal emulators should have a compatible code page
- Automatic CCSID – Is it a problem? conversion for remote access when needed
  - DRDA receiver
If one encoding scheme is good…
… then two must be better?

- Potential for incorrect character interpretation (i.e., data corruption) if multiple CCSIDs within one encoding scheme
  - Terminal emulator using incompatible code page
  - Incorrect CCSID specification
  - Change of CCSID specification
- Must be fixed and residual corruption cleaned up ahead of V8
  - 28 Step or Zap? It depends…
- V7 APAR PQ56697 introduces checks for non-zero valid CCSIDs and issues warning when trouble found
- V7 APAR PQ89018 will provide CCSID detection if DECP value does not match the emulator value

Unicode Catalog

- Run DSNTIJP8 (V7 APAR PQ84421) or DSNTIJPM (V8) ahead of migration to check that there are not multiple CCSIDs within one encoding
- V8 start up checks for:
  - Non-zero, valid CCSIDs
  - EBCDIC<>Unicode round trip conversion
- Must configure Unicode Conversion Services for z/OS
  - Unicode CCSIDs (367,1208,1200) <> ASCII/EBCDIC CCSIDs
  - Client CCSIDs <-> Unicode CCSIDs (367,1208,1200)
- Must also add to/from Unicode CCSIDs (367,1208,1200)
  - CCSID 37 for DBRMs provided with DB2 install
  - CCSID 500 for DRDA
  - CCSID 1047 for Unix System Services (USS)
- Rebuild conversion image, Define only what you need
- New image picked up by DB2 ‘on the fly’
- Configure for best conversion performance
  - z890 and z990 zSeries (hardware instructions instead of mcode/ucode)
  - z/OS R4
Unicode Catalog cont.

- SQL parsed and metadata derived from SQL in Unicode UTF-8 format
- Most character columns converted to Unicode VARCHAR(128)
- Trailing blanks not stripped
- Special characters expand beyond single byte (e.g., #)
- Increase in space: CHAR -> VARCHAR (length bytes)
- Decrease in space: DB2 defined indexes converted to NOT PADDED
- Recommend converting user defined indexes to NOT PADDED
- Compatibility (COMPAT) mode
  - SQL parsed in Unicode UTF-8 format
  - Metadata derived from SQL converted back to EBCDIC
- New Function Mode (NFM) mode
  - Catalog converted to Unicode UTF-8 format
  - Metadata derived from SQL no longer converted back to EBCDIC

Unicode Catalog cont.

- Use zparm UIFCIDS option to have IFCID trace data in Unicode UTF-8 format
  - Default = YES
  - Moves conversion overhead off-line e.g., DB2PE
- Conversion of SQL statements and derived metadata to Unicode
  - Minor Conversion (fast)
    - Optimization for most common single byte CCSIDs (MIXED=NO)
    - i.e., simple alphanumeric characters
    - Most common case ASCII/EBCDIC->UTF-8, can go other way
    - Internal 256 byte translation tables used by DB2 (TR/TRT)
  - Major Conversion (slower)
    - Offloaded to Unicode Conversion Services for z/OS
    - Used if EBCDIC/ASCII->UTF-8 cannot be performed inline
    - Always for ASCII/EBCDIC->UTF-16 (no optimization)
- Particular conversion not available
  - SQLCODE -332
  - SQL SUBSTR behavior can be inconsistent when accessing Catalog data
  - New character based functions (18) will be added through APAR PQ88784
Unicode Catalog cont.

- If precompile application with NEWFUN(NO)
  - Disables use of new SQL function for static SQL
  - Go back through SQL Parser during BIND processing
    - To get Unicode UTF-8 format
    - Incur extra overhead
  - After entry to NFM, default for NEWFUN changes to YES
    - DBRMs switch to Unicode UTF-8 format
    - Set NEWFUN to NO
    - Get DBRMs in EBCDIC format
- At BIND time
  - Anything from DBRM in Unicode UTF-8 format goes into SYSSTMT and SYSPACKSTMT in Unicode UTF-8 format
  - Anything from DBRM in EBCDIC format goes into SYSSTMT and SYSPACKSTMT in EBCDIC format

Removing growth inhibitors
Exploiting the 64-bit architecture
More partitions
More log data sets
More tables in join
Removing growth inhibitors

**DB2 for z/OS limits:**

<table>
<thead>
<tr>
<th></th>
<th>V7</th>
<th>V8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual storage</td>
<td>$2^{31}$</td>
<td>$2^{64}$</td>
</tr>
<tr>
<td>Active logs</td>
<td>31</td>
<td>93</td>
</tr>
<tr>
<td>Archive logs</td>
<td>1000</td>
<td>10,000</td>
</tr>
<tr>
<td>Partitions</td>
<td>254</td>
<td>4096</td>
</tr>
<tr>
<td>SQL statement length</td>
<td>32 KB</td>
<td>2 MB</td>
</tr>
<tr>
<td>Tables in a join</td>
<td>15</td>
<td>225</td>
</tr>
</tbody>
</table>

Exploiting the 64 bit architecture

**z/Architecture real addressing support:**

- Operating system supporting 24-bit, 31-bit, and 64-bit applications
- Up to 512 GB of central storage on z9-109s (128 GB per book)
- 64 bit registers, etc...
64 bit evolution

- **64 bit zSeries, z/Architecture hardware support (z800, z900, z990, z9)**

- **Operating system support**
  - 64 bit *Real storage* support (OS/390 V2R10 ESAME mode +)
    - Large amounts of real storage (for performance)
    - Improve all versions of DB2
    - V6 data space advantages (outperforms hiperpool)
  - 64 bit *Virtual storage* support (z/OS V1.2+)
    - z/OS 64-bit Virtual Storage Roadmap
    - Exploited by **DB2 V8** (requires z/OS 1.3)
      - Move large memory areas above the bar
    - Exploited by **IRLM V2.2** (that ships with DB2 V8)
      - Locks always in private storage above the bar (PC=YES enforced)

---

Supporting 4096 partitions

- **Maximum number of parts raised from 254 to 4096**
  - Table spaces and indexes
    - Table space use DSSIZE to go beyond 254 parts
  - **ALTER TS ADD PARTITION** adds partitions to the end
  - Maximum table size remains 16 TB for 4 KB pages

- **Data set naming convention expanded**
  - 'A001' - 'A999' partitions 1-999
  - 'B000' - 'B999' partitions 1000-1999
  - 'C000' - 'C999' partitions 2000-2999
  - 'D000' - 'D999' partitions 3000-3999
  - 'E000' - 'E096' partitions 4000-4096

- **Maximum number of parts allowed depends on page size and DSSIZE**
  - 4 KB page size, DSSIZE=1 GB => 4096 parts, 4 TB max table size
  - 4 KB page size, DSSIZE=64 GB => 256 parts, 16 TB max table size
Supporting more log data sets

- Increase the number of active logs from 31 to 93
- Increase the number of archive logs that can be recorded in the BSDS from 1000 to 10000
- Both changes require prior conversion of the BSDS to accommodate for this extra information
  - Run the DSNJCNVB BSDS conversion job
  - Must be in New Function Mode before conversion is allowed

Supporting more tables in a join

- V7: up to 15 tables can be joined in single FROM clause
  - If more SQLCODE -129
  - Does not apply to star join
- Can be tweaked via (hidden) zparm MXTBJJOIN
- Problem is dynamic programming exponential growth of combinations when determining best access path
- New algorithms
  - For storage management, CPU usage
  - Clipping off branches that will not lead to better access path
  - Monitoring and clipping of resource usage during prepare
- Result: in V8
  - Up to 225 tables allowed in join
  - Prepare (ET and CPU) time under control with reasonable chance of still picking the best access path
Availability

- Online schema evolution:
  - ALTER instead of DROP / CREATE
  - Data Partitioned Secondary Indexes (DPSI)
  - System-level point in time recovery
  - Additional online ZPARMs
  - Other availability enhancements

Online schema evolution

- **Partition changes**
  - Add a partition to the end of a table
  - Rotate partitions
  - Rebalance partitions

- **Unbundling partitioning and clustering**
  - Ability to drop the partitioning index
  - Data Partitioned Secondary Indexes
  - Ability to change the clustering index
    - Also applies to non-partitioned tables

- **Index changes**
  - Add columns to indexes

- **Table changes**
  - Ability to change data types and lengths (extensions)
    - Includes column data types referenced within a view
    - Includes column changes for indexed columns
Partitioned table space enhancements

- **Changing partition definitions**
  - Add partition
  - Rotate partition
    - Move first part to last
  - Rebalance partitions
- **Table controlled partitioning**
- **Terminology changes**
- **Data Partitioned Secondary Index**
- **ALTER INDEX - CLUSTER / NOT CLUSTER**
  - PADDED / NOT PADDED
  - ADD COLUMN to index

### ADD partition

```sql
ALTER TABLE ... ADD PARTITION
ENDING AT ("31-12-2005");
```

<table>
<thead>
<tr>
<th>Partition</th>
<th>ts</th>
<th>pi</th>
<th>npsi</th>
<th>dpsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2001 Jan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2001 Feb</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>2005 Oct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>2005 Nov</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>2005 Dec</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rotate partitions

<table>
<thead>
<tr>
<th>Part</th>
<th>ts</th>
<th>pi</th>
<th>npi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>2001 Jan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 2</td>
<td>2001 Feb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 59</td>
<td>2005 Nov</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 60</td>
<td>2005 Dec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rotate partitions - 2

```
ALTER TABLE ... ROTATE FIRST TO LAST
ENDING AT ("31-01-2005") RESET;
```

<table>
<thead>
<tr>
<th>Part</th>
<th>ts</th>
<th>pi</th>
<th>npi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>2006 Jan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 2</td>
<td>2001 Feb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 59</td>
<td>2005 Nov</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 60</td>
<td>2005 Dec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
After using ROTATE, logical ordering on partitions will be different from physical order

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPEPART</th>
<th>PARTITION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsbart</td>
<td>TS</td>
<td>0002</td>
<td>RW</td>
</tr>
<tr>
<td>- - THRU</td>
<td></td>
<td>0060</td>
<td></td>
</tr>
<tr>
<td>tsbart</td>
<td>TS</td>
<td>0001</td>
<td>RW</td>
</tr>
</tbody>
</table>

SYSTABLEPART and SYSCOPY have new LOGICAL_PART column
Version 8 classification of indexes

- An index may / may not be correlated with the partitioning columns of the table
  - Partitioning index (PI)
  - Secondary index

- An index may / may not be physically partitioned
  - Partitioned
  - Non-partitioned

- Clustering index:
  - Any index may be the clustering index
  - The clustering index can be non-unique

Partitioning indexes

- A partitioning index
  - Has the same leftmost columns as the columns which partition the table
  - These columns have the same collating sequence (ASC / DESC)

```
CREATE TABLE CUSTOMER (
  ACCOUNT_NUM INTEGER,
  LAST ACTIVITY_DT CHAR(3),
  CCODE CHAR(2),
  ...
) PARTITION BY (ACCOUNT_NUM ASC)
...
CREATE . . . INDEX part_ix_1 ON CUSTOMER (ACCOUNT_NUM ASC)
```
Partitioning indexes -2

A partitioning index has the same leftmost columns, in the same collating sequence, as the columns which partition the table.

Partitioning index  part_IX_1 (ACCOUNT_NUM ASC)

CREATE TABLE CUSTOMER (ACCOUNT_NUM INTEGER, CUST_LAST_NM CHAR(30), . . . )
PARTITION BY (ACCOUNT_NUM ASC). . .

Secondary indexes

A secondary index is any index which is not a partitioning index.

Secondary Index  SI_1  on LAST_ACTIVITY_DT

Secondary Index  SI_2  on STATE_CD
Partitioned index and non-partitioned index

Partitioned index -- multiple partitions -- 1 per data partition

IX

TB

Partitioned table

IX

Non-partitioned index

Partitioning indexes (Partitioned and Non-Partitioned)

Partitioned Partitioning index  part_ix_1

Partitioned table

Non-partitioned Partitioning index  part_ix_2
Secondary indexes (Partitioned and Non-Partitioned)

Data Partitioned Secondary Index (DPSI) -- data_part_si_1

Partitioned table

Non-Partitioned Secondary Index (NPSI) -- non_part_si_2

Data Partitioned Secondary Indexes - Benefits

- **Online REORG**
  - No BUILD2 phase with DPSIs

- **LOAD PART jobs**
  - No shared pages with DPSIs
  - Removes page contention, and GBP flooding issues
  - More efficient "append" strategy can be used

- **Easier partition level operations**
  - Rolling in/out new partitions much easier, more efficient with DPSIs

- **Media failure**
  - Partition level recovery

- **Data sharing overhead**
  - Member <-> partition affinity routing is effective for DPSIs

- **Allows query parallelism**
DPSI - Design considerations

- DPSIs are a design choice and may not fit ALL situations
- Query predicates that solely reference columns in DPSIs must probe each partition - this will degrade query performance
  
  - SELECT * FROM table WHERE DPSI_col = 'X'
  - SELECT * FROM table WHERE DPSI_col BETWEEN 'X' AND 'Y'
  
- If predicates also restrict the query to a single partition, query performance will benefit
  
  - "Partition pruning"
  - Bind time pruning
    - P_COL = literal
  - Runtime pruning with REOPT(VARS)
    - P_COL = :hv , ? , special register

- Recommendation: Code predicates explicitly in your applications to allow for partition pruning when a DPSI exists

Other index enhancements

- ALTER INDEX ADD COLUMN
- ALTER INDEX PADDED / NOT PADDED
  
  - True varying length index keys (NOT PADDED)
- ALTER clustering index
  
  - For both partitioning and non-partitioning tables
  - Effective immediately / REORG required to rearrange all existing rows to new clustering index
- More index enhancements in the performance section
Supported ALTER TABLE data types in V8

<table>
<thead>
<tr>
<th>FROM DATA TYPE</th>
<th>TO DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallint</td>
<td>integer</td>
</tr>
<tr>
<td>smallint</td>
<td>float(1-21) or real</td>
</tr>
<tr>
<td>smallint</td>
<td>float(22-53) or double</td>
</tr>
<tr>
<td>smallint</td>
<td>&gt;= decimal(5,0)</td>
</tr>
<tr>
<td>integer</td>
<td>float(22-53) or double</td>
</tr>
<tr>
<td>integer</td>
<td>&gt;= decimal(10,0)</td>
</tr>
<tr>
<td>float(1-21) or real</td>
<td>float(22-53) or double</td>
</tr>
<tr>
<td>&lt;= decimal(7,s)</td>
<td>float(1-21) or real</td>
</tr>
<tr>
<td>&lt;= decimal(15.s)</td>
<td>float(22-53) or double</td>
</tr>
<tr>
<td>decimal(p,s)</td>
<td>decimal(p+a,s+b)</td>
</tr>
<tr>
<td>char(n)</td>
<td>char(n+x)</td>
</tr>
<tr>
<td>char(n)</td>
<td>varchar(n+x)</td>
</tr>
<tr>
<td>varchar(n)</td>
<td>char(n+x)</td>
</tr>
<tr>
<td>varchar(n)</td>
<td>varchar(n+x)</td>
</tr>
<tr>
<td>graphic(n)</td>
<td>graphic(n+x)</td>
</tr>
<tr>
<td>graphic(n)</td>
<td>varchar(n+x)</td>
</tr>
<tr>
<td>vargraphic(n)</td>
<td>vargraphic(n+x)</td>
</tr>
<tr>
<td>vargraphic(n)</td>
<td>vargraphic(n+x)</td>
</tr>
</tbody>
</table>

- Within same data type group (character - numeric)
- Smaller -> bigger size

```
ALTER TABLE tab1
ALTER COLUMN cola
    SET DATA TYPE CHAR(20);
```

For decimal data types "a" plus "b" must be greater than zero or there is no change

For character data types "x" can be greater than or equal to zero

What happens to the TABLE and data?

- **Table**
  - New definition is captured in catalog and directory
  - Maximum 255 alters per table space before a REORG is required
  - Table space is placed in AREO* (advisory REORG-pending)
  - Plans, packages and cached dynamic statements referring to the changed column are invalidated
  - Runstats values for columns are invalidated (eg HIGH2KEY, LOW2KEY - treated as if -1) - similar to extending varchar col in V7

- **Data**
  - Existing data remains unchanged
  - On SELECT, data will be materialized into the new format
  - INSERT/UPDATE changes the entire row to latest format
  - REORG changes all rows to the current (latest) version and improve performance (In many cases online Reorg can be used)
Dependent indexes and referencing views

- **Indexes**
  - New index version created for each index that references the altered column (up to 16 versions)
  - Immediate access for character data type extensions
    - Index placed in AREO*
  - Delayed access for numeric data types
    - Index placed in RBDP

- **Views**
  - Views referencing the affected columns are regenerated
  - Errors during regeneration causes the ALTER to rollback
  - Plans and packages dependent on the view are invalidated
  - `ALTER VIEW name REGENERATE` - limited usage only

---

Review of new DBET states

<table>
<thead>
<tr>
<th>Object type</th>
<th>Action taken</th>
<th>Resulting condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Space</td>
<td>ALTER TABLE ALTER COLUMN (any data type)</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of VARCHAR to CHAR</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of CHAR to VARCHAR</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of VARCHAR to VARCHAR (padded only)</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of GRAPHIC to VARGRAPHIC</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of VARGRAPHIC to GRAPHIC</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of CHAR to CHAR</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of GRAPHIC to GRAPHIC</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of VARGRAPHIC to VARGRAPHIC (padded only)</td>
<td>AREO*</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER TABLE ALTER COLUMN of NUMERIC to NUMERIC</td>
<td>RBDP</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER INDEX ADD COLUMN</td>
<td>RBDP (1)</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER INDEX from NOT PADDED to PADDED</td>
<td>RBDP</td>
</tr>
<tr>
<td>Index Space</td>
<td>ALTER INDEX from PADDED to NOT PADDED</td>
<td>RBDP</td>
</tr>
</tbody>
</table>

The AREO* cannot be reset by a START FORCE command, but can be reset using the REPAIR utility.
(1) Can be AREO* if column added to table is done in same UOW as alter ix
Online Schema Evolution

- SQL ALTER leaves tablespace in AREO
  - Up to 30% CPU increase until REORG
    - Fast column processing disabled
    - Data conversion
  - Extra logging
- SQL ALTER can leave index in RBDP
  - Invalidation of plans and packages
  - For dynamic queries, indexes that are in RBDP state are ignored
    - If the index is on a partitioned tablespace
      - All the parts (logical or physical) need to be in RBDP
      - If ANY part is available then the index will not be ignored
- Conversion from non-large index-controlled partitioning leaving partition(s) in REORP

Online Schema Changes cont.

- Rotate performance can be ugly with many NP(S)Is
  - Subject partition must be emptied ready for new data
    - Individual row deleted and NPSIs corrected serially
    - Increased logging
    - Logical drain of NP(S)Is
    - Lock on DBD for extended period
  - PIT recovery not possible for rotated partition
- Recommendations
  - Unload or copy data out as an archive
  - LOAD REPLACE PART 1 with dummy input dataset
  - ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST
SQL enhancements

- Support for long names
- Dynamic scrollable cursors
- Common table expressions
- Recursive SQL
- Multi-row insert
- GET DIAGNOSTICS
- Multi-row fetch

- Identity column enhancements
- Sequences
- SELECT from INSERT
- Scalar fullselect
- DISTINCT enhancements
- .. and much much more

Breaking SQL limitations

- Table name length (*) 18 to 128
- Column name length 18 to 30
- Index key size 255 to 2000
- Character literals 255 to 32704
- Hex literal digits 255 to 32704
- Predicates 55 to 32704
- SQL statement length 32KB to 2MB
- Number of tables in a join 15 to 225

(*) Also applies to (most) other DB2 objects, like views, aliases, index, triggers, synonyms, ...

© 2006 IBM Corporation
Sensitive and insensitive cursors - V7 review

- **DECLARE C1 INSENSITIVE**
  - SCROLL...
  - ... FETCH INSENSITIVE...
  - Read only cursor
  - Not aware of updates or deletes in base table

- **DECLARE C1 SENSITIVE**
  - STATIC SCROLL...
  - ... FETCH INSENSITIVE...
  - Updatable cursor
  - Aware of own updates or deletes within cursor
  - Other changes to base table not visible to cursor
  - All inserts not recognized

- **DECLARE C1 SENSITIVE**
  - STATIC SCROLL...
  - ... FETCH SENSITIVE...
  - Updatable cursor
  - Aware of own updates and deletes within cursor
  - Sees all committed updates and deletes
  - All inserts not recognized

---

**Dynamic scrollable cursors**

- **Scrollable cursor that provides access to base table rather than temp tables** -- allows viewing of updates and inserts done by you or other users

- **Defaults to single row fetch, so DDF applications should use:**
  - Multi-row FETCH
  - Positioned UPDATE/DELETE for multi-row FETCH

```sql
DECLARE C1 SENSITIVE DYNAMIC SCROLL CURSOR
FOR SELECT C1, C2 FROM T1;
```
Declare cursor - new attributes

SENSITIVE DYNAMIC
- Specifies that size of result table is not fixed at OPEN cursor time
- Cursor has complete visibility to changes
  - All committed inserts, updates, deletes by other application processes
  - All positioned updates and deletes within cursor
  - All inserts, updates, deletes by same application processes, but outside cursor
- FETCH executed against base table - no temporary result table created

ASENSITIVE
- DB2 determines sensitivity of cursor
- If read-only...
  - It behaves as an INSENSITIVE cursor
  - Cursor is INSENSITIVE if SELECT statement does not allow it to be SENSITIVE (UNION, UNION ALL, FOR FETCH ONLY, FOR READ ONLY)
- If not read-only, SENSITIVE DYNAMIC is used for maximum sensitivity
- Mainly for client applications that do not care whether or not the server supports the sensitivity or scrollability

Sensitive dynamic scrollable cursor - example

Declare sensitive dynamic scrollable cursor
DECLARE CURSOR ORDERScroll SENSITIVE DYNAMIC SCROLL FOR
  SELECT ORDErnum, CUSTNAME, ORDERAMT, ORDERDATE
  FROM ORDERS WHERE ORDERAMT > 1 FOR UPDATE OF COMMENTS;

Fetch forward from scrollable cursor
LOOP-TO-FILL SCREEN
  DO 1 TIMES
    FETCH FROM ORDERScroll INTO :hv1, :hv2, :hv3, :hv4;
  END

Fetch ABSOLUTE from scrollable cursor
Re-read the third row
  FETCH ABSOLUTE + 3 FROM ORDERScroll INTO :hv1, :hv2, :hv3, :hv4;

Fetch RELATIVE from scrollable cursor
Read the third row from the current position
  FETCH SENSITIVE RELATIVE +3 FROM ORDERScroll INTO :hv1, :hv2, :hv3, :hv4;

Execute a positioned UPDATE through scrollable cursor
Update the current row
  UPDATE ORDERS SET COMMENTS = "Expedite"
    WHERE CURRENT OF ORDERScroll;


Cursor type comparison

<table>
<thead>
<tr>
<th>Cursor type</th>
<th>Result table</th>
<th>Visibility of own changes</th>
<th>Visibility of changes by others</th>
<th>Updatability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-scrollable (SQL contains a Join or Sort, etc)</td>
<td>Fixed, workfile</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Non-scrollable</td>
<td>No workfile, base table access</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSENSITIVE SCROLL</td>
<td>Fixed, declared temp table</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SENSITIVE STATIC SCROLL</td>
<td>Fixed, declared temp table</td>
<td>Yes</td>
<td>Yes (INSERTs not allowed)</td>
<td>Yes</td>
</tr>
<tr>
<td>SENSITIVE DYNAMIC SCROLL</td>
<td>No declared temp table, base table access</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Common table expressions (CTE)

- **Allowed in SELECT, CREATE VIEW, INSERT**
- **Syntax:**

```
WITH common-table-expression AS ( fullselect )
```  

- **Refer to table-name later in SQL**
Common table expressions (CTE) - 2

WITH
  E AS
  (SELECT EMPNO, LASTNAME, SALARY, SUBSTR(CHAR(HIREDATE, ISO), 1, 3) CONCAT '0 - 9' AS HIREDECADE FROM EMPLOYEE)
,
M (HIREDECADE, MINIMUM_SALARY) AS
  (SELECT HIREDECADE, MIN(SALARY) FROM E GROUP BY HIREDECADE)

SELECT E.EMPNO, E.LASTNAME, E.HIREDECADE, E.SALARY, M.MINIMUM_SALARY FROM E INNER JOIN M ON E.HIREDECADE = M.HIREDECADE

Common table expressions (CTE) - 3

SELECT E.EMPNO, E.LASTNAME, E.HIREDECADE, E.SALARY, M.MINIMUM_SALARY FROM E INNER JOIN M ON E.HIREDECADE = M.HIREDECADE

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>LASTNAME</th>
<th>HIREDECADE</th>
<th>SALARY</th>
<th>MINIMUM_SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>200340</td>
<td>ALONZO</td>
<td>1940 - 9</td>
<td>23840.00</td>
<td>23840.00</td>
</tr>
<tr>
<td>000050</td>
<td>GEYER</td>
<td>1940 - 9</td>
<td>40175.00</td>
<td>23840.00</td>
</tr>
<tr>
<td>000340</td>
<td>GOUNOT</td>
<td>1940 - 9</td>
<td>23840.00</td>
<td>23840.00</td>
</tr>
<tr>
<td>000110</td>
<td>LUCCHESI</td>
<td>1950 - 9</td>
<td>46500.00</td>
<td>46500.00</td>
</tr>
<tr>
<td>000120</td>
<td>O’CONNELL</td>
<td>1960 - 9</td>
<td>29250.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000230</td>
<td>JEFFERSON</td>
<td>1960 - 9</td>
<td>22180.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000220</td>
<td>LUTZ</td>
<td>1960 - 9</td>
<td>29840.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000200</td>
<td>BROWN</td>
<td>1960 - 9</td>
<td>27740.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>200010</td>
<td>HEMMINGER</td>
<td>1960 - 9</td>
<td>46500.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>200220</td>
<td>JOHN</td>
<td>1960 - 9</td>
<td>29840.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000320</td>
<td>MEHTA</td>
<td>1960 - 9</td>
<td>19950.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000310</td>
<td>SETRIGHT</td>
<td>1960 - 9</td>
<td>19180.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000280</td>
<td>SCHNEIDER</td>
<td>1960 - 9</td>
<td>26250.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000250</td>
<td>SMITH</td>
<td>1960 - 9</td>
<td>19180.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>200310</td>
<td>SPRINGER</td>
<td>1960 - 9</td>
<td>26250.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>200280</td>
<td>SCHWARTZ</td>
<td>1960 - 9</td>
<td>26250.00</td>
<td>15900.00</td>
</tr>
<tr>
<td>000010</td>
<td>HAAS</td>
<td>1960 - 9</td>
<td>52750.00</td>
<td>15900.00</td>
</tr>
</tbody>
</table>
Recursive SQL with CTE

WITH

RPL (PART, SUBPART, QUANTITY) AS

(  

**Initialization Select**

SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
FROM PARTLIST ROOT
WHERE ROOT.PART = '01'

UNION ALL

**Iterative Select**

SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
FROM RPL PARENT, PARTLIST CHILD
WHERE PARENT.SUBPART = CHILD.PART
)

**Main Select**

SELECT PART, SUBPART, SUM(QUANTITY) AS QUANTITY
FROM RPL
GROUP BY PART, SUBPART

---

Recursive SQL - Initialization select

```
SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
FROM PARTLIST ROOT
WHERE ROOT.PART = '01'
```

<table>
<thead>
<tr>
<th>PART</th>
<th>SUBPART</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>5</td>
</tr>
<tr>
<td>00</td>
<td>05</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>06</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>7</td>
</tr>
<tr>
<td>02</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>07</td>
<td>6</td>
</tr>
<tr>
<td>04</td>
<td>08</td>
<td>10</td>
</tr>
<tr>
<td>04</td>
<td>09</td>
<td>11</td>
</tr>
<tr>
<td>05</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>05</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>07</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>07</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>

RPL

<table>
<thead>
<tr>
<th>PART</th>
<th>SUBPART</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>06</td>
<td>3</td>
</tr>
</tbody>
</table>

PARTLIST Table

© 2006 IBM Corporation
### Recursive SQL - First iteration

```
SELECT CHILD.PART, CHILD.SUBPART, 
CHILD.QUANTITY 
FROM RPL PARENT, PARTLIST CHILD 
WHERE PARENT.SUBPART = CHILD.PART
```

<table>
<thead>
<tr>
<th>PART</th>
<th>SUBPART</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>5</td>
</tr>
<tr>
<td>00</td>
<td>05</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>06</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>7</td>
</tr>
<tr>
<td>02</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>07</td>
<td>6</td>
</tr>
<tr>
<td>04</td>
<td>08</td>
<td>10</td>
</tr>
<tr>
<td>04</td>
<td>09</td>
<td>11</td>
</tr>
<tr>
<td>05</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>05</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>07</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>07</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>

### Recursive SQL - Second iteration

```
SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY 
FROM RPL PARENT, PARTLIST CHILD 
WHERE PARENT.SUBPART = CHILD.PART
```

<table>
<thead>
<tr>
<th>PART</th>
<th>SUBPART</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>5</td>
</tr>
<tr>
<td>00</td>
<td>05</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>06</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>7</td>
</tr>
<tr>
<td>02</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>07</td>
<td>6</td>
</tr>
<tr>
<td>04</td>
<td>08</td>
<td>10</td>
</tr>
<tr>
<td>04</td>
<td>09</td>
<td>11</td>
</tr>
<tr>
<td>05</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>05</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>07</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>07</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>
Recursive SQL - Main select

```sql
SELECT PART, SUBPART, SUM(QUANTITY) AS QUANTITY
FROM RPL
GROUP BY PART, SUBPART
```

<table>
<thead>
<tr>
<th>PART</th>
<th>SUBPART</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>01</td>
<td>06</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>7</td>
</tr>
<tr>
<td>02</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>07</td>
<td>6</td>
</tr>
<tr>
<td>04</td>
<td>08</td>
<td>10</td>
</tr>
<tr>
<td>04</td>
<td>09</td>
<td>11</td>
</tr>
<tr>
<td>06</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>05</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>05</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>06</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>07</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>07</td>
<td>14</td>
<td>8</td>
</tr>
</tbody>
</table>

Final Result Table

Multi-row INSERT

New (third) form of INSERT

- INSERT with FOR "n" ROWS is used to insert multiple rows into the table or view using values provided in host variable array

FOR "n" ROWS

- For static, valid to specify FOR "n" ROWS on INSERT statement
  (for dynamic INSERT, specify FOR "n" ROWS on EXECUTE statement)
- Input provided with host variable array -- each array represents cells for multiple rows of a single column

VALUES clause allows specification of multiple rows of data

- Host variable arrays used to provide values for a column on INSERT
- Example: VALUES (:hva1, :hva2)
INSERT syntax

```
INSERT INTO table-name
    view-name
    OVERRIDING USER VALUE
VALUES
    expression
    DEFAULT
    NULL
    expression
    DEFAULT
    NULL
FROM (SELECT
    RR
    CS
    QUERYNO integer
)
```

Values

```
VALUES
    expression
    DEFAULT
    NULL
```

Full select

```
VALUES
    expression
    DEFAULT
    NULL
```

FOR dynamic statement:
- FOR "n" ROWS specified on EXECUTE statement
- ATOMIC/NOT ATOMIC CONTINUE ON SQLEXCEPTION specified on PREPARE statement
- (FOR SINGLE ROW / FOR MULTIPLE ROWS specified on PREPARE statement)

For dynamic statement:
- FOR "n" ROWS specified on EXECUTE statement
- ATOMIC/NOT ATOMIC CONTINUE ON SQLEXCEPTION specified on PREPARE statement
- (FOR SINGLE ROW / FOR MULTIPLE ROWS specified on PREPARE statement)
ATOMIC/NOT ATOMIC CONTINUE ON SQLEXCEPTION

ATOMIC (default)
- If the insert for any row fails, all changes made to database by that
  INSERT statement are undone

NOT ATOMIC CONTINUE ON SQLEXCEPTION
- Inserts are processed independently
- If errors occur during execution of INSERT, processing continues
- Diagnostics are available for each failed row through

GET DIAGNOSTICS
- SQLCODE indicates if:
  - All successful - SQLCODE 0
  - All were successful but warning(s) occurred - SQLSTATE 01659, SQLCODE +252
  - All failed - SQLSTATE 22530, SQLCODE -254
  - At least one failed - SQLSTATE 22529, SQLCODE -253

GET DIAGNOSTICS
- Returns SQL error information
  - For overall statement
  - For each condition (when multiple errors occur)
- Supports SQL error message tokens greater than 70 bytes (SQLCA limitation)
- Considering replacing DSNTIAR with GET DIAG.

```
INSERT INTO T1 VALUES(:ARRAY) FOR 5 ROWS ;

GET DIAGNOSTICS :ERR_COUNT = NUMBER;
DO II = 1 TO ERR_COUNT;
    GET DIAGNOSTICS CONDITION :II
    :RC = RETURNED_SQLSTATE;
END;
```
GET DIAGNOSTICS examples

To determine how many rows were updated in an UPDATE statement

- GET DIAGNOSTICS :rcount = ROW_COUNT;

To handle multiple SQL errors during a NOT ATOMIC multi-row insert

- GET DIAGNOSTICS :numerrors = NUMBER;
- Then code a loop to execute the following for the number of errors
  - GET DIAGNOSTICS CONDITION :i :retstate = RETURNED_SQLSTATE

To see all diagnostic information for an SQL statement

- GET DIAGNOSTICS :diags = ALL STATEMENT
- Sample output in :diags
  - Number=1; Returned_SQLSTATE=02000;
  - DB2_RETURNED_SQLCODE=+100;
  - Would continue for all applicable items and for all conditions
  - Items are delimited by semi-colons

Multi-row FETCH

- Returns multiple rows on one API crossing
- Significant performance boost
- "wide" cursor with locks on multiple rows
- Supports scrollable and non-scrollable, static and dynamic SQL

DECLARE C1 CURSOR
  WITH ROWSET POSITIONING
  FOR SELECT COL1, COL2 FROM T1;

OPEN C1;

FETCH NEXT ROWSET FROM C1
  FOR :hv ROWS INTO :ARRAY1, :ARRAY2;
Multi-row FETCH - 2
- Allows positioned UPDATE or DELETE (of a single row) on a "wide" cursor

```
UPDATE T1 SET COL1='ABC'
WHERE CURRENT OF C1
FOR ROW :hv OF ROWSET
```

- Using only the WHERE CURRENT OF clause on a cursor with an "active" ROWSET, updates all rows in the ROWSET

Identity column improvements
- ALTER support for identity columns, eg.

```
ALTER TABLE ALTER COLUMN idencol
RESTART WITH 500
```

- GENERATED ALWAYS | BY DEFAULT
- START WITH value
- RESTART WITH value
- INCREMENT BY
- MINVALUE | NO MINVALUE
- MAXVALUE | NO MAXVALUE
- CYCLE | NO CYCLE
- CACHE | NO CACHE
- ORDER | NO ORDER  (Bold keywords are new in V8)
Sequences

What is it?

- A way to generate unique numeric key values and to coordinate keys across multiple rows and tables
- A sequence object is a
  - User-defined object that generates a sequence of numeric values
  - Stand-alone object - NOT TIED TO A TABLE
- Generated values returned to the user can be used for anything
- Once a sequence has been incremented, DB2 does not wait for a commit before allowing the sequence to be incremented again by another transaction

Sequences - benefits

- Useful for porting of applications from other vendors
- Better performance and concurrency than application generated counters
- Recoverability in the event of a DB2 system failure
- Can be shared across data sharing group members
- New SQL to support sequences:
  - CREATE SEQUENCE
  - ALTER SEQUENCE
  - DROP SEQUENCE
  - COMMENT ON SEQUENCE
  - GRANT/REVOKE privileges for the sequence object
  - NEXT/ PREVIOUS VALUE FOR sequence-name
Sequences - examples

- An example:

```
CREATE SEQUENCE order_seq
    START WITH 1 INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE CACHE 2
```

```
INSERT INTO orders (orderno,custno)
    VALUES (NEXT VALUE FOR order_seq,123456);
```

```
UPDATE orders
    SET orderno =NEXT VALUE FOR order_seq
    WHERE custno =123456;
```

```
SELECT NEXT VALUE FOR order_seq
    INTO :hv_seq from sysibm.sysdummy1;
```

Sequences - examples -2

1) Assume sequence order_seq is created with START WITH 1, INCREMENT BY 1

Application starts
```
VALUES (NEXT VALUE FOR order_seq) INTO :hv1;  Generates Value of 1
VALUES (NEXT VALUE FOR order_seq) INTO :hv1;  Generates Value of 2
COMMIT;
```
```
VALUES (PREVIOUS VALUE FOR order_seq) INTO :hv1;  Returns most recently generated value (2)
VALUES (PREVIOUS VALUE FOR order_seq) INTO :hv1;  Returns same value (2)
```

MUST BE WITHIN THE SAME APPLICATION!!

2) Assume same sequence order_seq

Application starts
```
VALUES (PREVIOUS VALUE FOR order_seq) INTO :hv1;
Returns error (SQLCODE -845)
```
```
Must NEXT VALUE before you can use PREVIOUS VALUE
```
Sequence objects vs. identity columns

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Identity columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand-alone object</td>
<td>Tied to a table</td>
</tr>
<tr>
<td>Can use one sequence for many tables or many sequences in one table</td>
<td>One to one relationship between identity column and the table</td>
</tr>
<tr>
<td>Retrieved using NEXT VALUE FOR / PREVIOUS VALUE FOR expressions</td>
<td>SELECT from FINAL TABLE(INSERT..) Retrieved using IDENTITY_VAL_LOCAL function -- within agents scope only</td>
</tr>
<tr>
<td>SELECT from FINAL TABLE(INSERT..)</td>
<td>Can be altered using ALTER SEQUENCE statement Prior to V8 could not be altered</td>
</tr>
<tr>
<td>Can be altered using ALTER SEQUENCE statement</td>
<td></td>
</tr>
</tbody>
</table>

Scalar fullselect

- Is a fullselect, enclosed in parentheses, that returns a single value
- Allows scalar fullselect where expressions were previously supported

```
SELECT C1
FROM T1
WHERE
  C2 > (SELECT COL1 FROM T2 ...);

SELECT C1,
  (SELECT COL1 FROM T2...),
  C3
FROM T1;
```
Scalar fullselect in CASE expression of UPDATE

- Give discount to the parts that have the large inventory and raise price on the parts that have the small inventory.

```sql
UPDATE PARTPRICE N SET PRICE =
CASE
  WHEN((SELECT ONHAND# FROM INVENTORY WHERE PART=N.PART) < 7)
     THEN 1.1 * PRICE
  WHEN((SELECT ONHAND# FROM INVENTORY WHERE PART=N.PART) > 20)
     THEN .8 * PRICE
  ELSE PRICE
END;
```

SELECT from INSERT

- Elegant technique for retrieving values created/modified by DB2 during INSERT
  - Identity columns, sequence values
  - User-defined defaults, expressions
  - Columns modified by triggers
  - ROWIDs, special registers

```sql
SELECT C1, C2, C3, C4, C5
FROM FINAL TABLE (INSERT (C1, C5) INTO T1
VALUES ('ABC', CURRENT DATE));
```
IS NOT DISTINCT FROM

- SQL uses three-valued logic where any given comparison can return: TRUE, FALSE, or NULL
- Applications can use IS NOT DISTINCT FROM to obtain a TRUE result instead of NULL when comparing NULL values

```
SELECT C1 FROM T1 WHERE C1 IS NOT DISTINCT FROM :hv;
```

<table>
<thead>
<tr>
<th>C1 value</th>
<th>:hv value</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>'ABC'</td>
<td>FALSE</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
<tr>
<td>'ABC'</td>
<td>'ABC'</td>
<td>TRUE</td>
</tr>
<tr>
<td>'ABC'</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>'ABC'</td>
<td>'DEF'</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Multiple DISTINCT clause

Version 7 .....  
SELECT DISTINCT C1, C2 FROM T1;  
SELECT COUNT(DISTINCT C1) FROM T1;  
SELECT COUNT(DISTINCT(C1)),SUM(DISTINCT C1)FROM T1; -- same col  
SELECT C1, COUNT(DISTINCT C2) FROM T1 GROUP BY C1;

With Version 8 .....  
SELECT DISTINCT COUNT(DISTINCT C1), SUM(DISTINCT C2) FROM T1;  
SELECT COUNT(DISTINCT C1), AVG(DISTINCT C2) FROM T1  
GROUP BY C1;  
SELECT SUM(DISTINCT C1), COUNT(DISTINCT C1), AVG(DISTINCT C2) FROM T1  
GROUP BY C1 HAVING SUM(DISTINCT C1) = 1;

Not Supported in Version 8 .....  
SELECT COUNT(DISTINCT A1,A2) FROM T1  
GROUP BY A2;  
SELECT COUNT(DISTINCT(A1,A2)) FROM T1  
GROUP BY A2;
SQL statements up to 2 MB

- SQL statements can now be up to 2 MB in length
- Parse tree has been completely rearchitected to reduce the number of cases of SQLCODE -101 (SQL too complex) due to SQL statement length
- Important for SQL procedure language applications since the whole program code is considered as one SQL statement

Other SQL improvements

- SQL procedures language extensions (see later)
- GROUP BY expression
  - SELECT A+B,C FROM T ORDER BY A+B GROUP BY A+B

- Qualified column names on SET clause
  - UPDATE T1 SET T1.COL1...

- Qualified column names on INSERT
  - INSERT T1.COL1 INTO T1 VALUES...
Other SQL improvements -2

- **DROP ON COMMIT**
  - Declared Temporary Table improvement that causes declared temp tables to be dropped automatically at commit
  - When no HELD cursors are open against the temp table
  - Significant improvement for DDF threads that use CMTSTAT= INACTIVE
  - Stops declared temp tables from preventing thread from going INACTIVE

Other SQL improvements -3

- **READ ONLY USE AND KEEP UPDATE LOCKS**
  - Allows WebSphere persistence layer to minimize network traffic when using searched update and pessimistic locking

```
PREPARE stmt 1 FROM "SELECT C1, C2, C3 FROM T1 WHERE C1=? AND C2=? FOR READ ONLY USE AND KEEP UPDATE LOCKS"
```

- **Transparent ROWID column**
  - Eliminates the need to explicitly declare a ROWID column in tables that include LOBs
  - DB2 will generate a "hidden" ROWID column, which is not visible on SELECT *
  - Simplifies porting of LOB applications from other platforms
New built-in functions

- **ENCRIPT_TDES**: encrypt a column in a table with a user-provided encryption password
  - ENCRYPTION PASSWORD special register
- **DECRIPT_BIT, DECRYPT_CHAR, DECRYPT_DB**
- **GETHINT**: obtain hint to help remember the ENCRYPTION PASSWORD
- **GENERATE_UNIQUE** creates a CHAR(13) FOR BIT DATA value, unique across a sysplex
- **GETVARIABLE** (see later)
- **SET ENCRYPTION PASSWORD xx [WITH HINT yy]**

New special registers

Client information for this connection

Provided by sqleseti, Java, RRS SIGNON

- CLIENT ACCTNG accounting string
- CLIENT APPLNAME value of application name
- CLIENT USERID client user ID
- CLIENT WRKSTNNAME workstation name

```sql
select *
from dept
where deptno = current client_userid;
```
Session variables

- Variables set by DB2, connection or signon exit
- Built-in function to retrieve value for a variable
  - For example, use function in views, triggers, stored procedures & constraints to enforce security policy
- Can have more general, flexible access checks
  - Multiple columns, AND/OR logic, ...
- Complements other security mechanisms

```
CREATE VIEW V1 AS SELECT * FROM T1 WHERE COL5 = GETVARIABLE(SYSIBM.SECLABEL);
```
JDBC driver types

- **Java defines 4 types of JDBC drivers**
  - Type 1 -- implements JDBC as a layer on top of ODBC
  - Type 2 -- uses native method calls (JNI) to call DLLs that issue the SQL (the DLLs are specially written for JDBC)
  - Type 3 -- network driver using a communication protocol that is DBMS independent (for example using TCP/IP)
  - Type 4 -- network driver that speaks the DBMS network protocol (for example DRDA)

- **Type 1 drivers generally have more CPU overhead than type 2**
- **Type 3 and type 4 drivers must route through a network layer, so they are not as efficient for local JDBC connectivity**

Objectives for DB2 Universal Driver

- **Single driver for Linux, Unix, Windows, and z/OS**
  - Eliminates major cause of Java porting problems
- **Provide fully compliant JDBC 3.0 driver (soon)**
- **Improved Java driver integration with DB2**
- **Simplify install/deployment of Type 2 driver**
- **Provide 100% Java application development process for SQLJ**
- **Improve JDBC and SQLJ performance significantly**
  - SQLJ is now faster than JDBC on Unix/Windows
- **Type 4 driver for thin clients**
- **Trace improvements**
  - Dynamically turn trace on/off
  - Multiple levels of trace detail
Connectivity with Java DB2 Universal Driver

- Thin client type 4 driver
- DB2 Connect Gateway
- DB2 for OS/390 and z/OS
- DB2 for Unix/Windows

SQLJ application development using Universal Driver

- **100% Java application process**
  - Eliminates DBRM files and .bnd files
- **New SQLJ serialized profile format**
  - Fully portable to all platforms -- user can deploy on any server platform without running db2sqljcustomize on the target system
  - Contains information needed for all BIND operations, without having to recustomize on each BIND
  - Conversion utility for old SQLJ serialized profiles for z/OS
- **Simplifies deployment of applications, but does require changes in existing procedures used by SQLJ users.**
- **WSAD V5.1 integration**
XML publishing functions

- New built-in functions for generating XML
  - `XMLELEMENT` generates XML element from arguments
  - `XMLATTRIBUTES` specify attributes for the XML element within `XMLELEMENT`
  - `XMLFOREST` creates forest of XML elements that share a specific pattern
  - `XMLCONCAT` returns a forest of XML elements that are generated from a concatenation of two or more arguments
  - `XMLAGG` returns a concatenation of XML elements from a collection of XML elements
  - `XML2CLOB` converts the new transient XML data type into UTF-8 CLOB

- Alternative to XML generation functions of XML Extender
- Integrated with DB2 engine leads to better performance

Complex query using XML functions

```sql
SELECT VARCHAR(XML2CLOB(XMLElement(NAME 'TABLE',
XMLATTRIBUTES('1' as "border"),
XMLElement(NAME CAPTION, 'Department-Employee Table'),
XMLElement(NAME TR, XMLFOREST('Dept No' as TH, 'Dept Name' as TH,
'Emp No' as TH, 'Emp Name' as TH, 'Phone' as TH) ),
XMLAGG(
XMLCONCAT(
XMLElement(NAME TR, XMLFOREST(NAME TD,
XMLATTRIBUTES( X.CNT+1 as "rowspan"),
D.DEPTNO),
XMLElement(NAME TD,
XMLATTRIBUTES( X.CNT+1 as "rowspan"),
D.DEPTNAME)
),
(SELECT XMLAGG(XMLElement(NAME TR,
XMLForest(EMPNO as TD, FIRSTNAME || ' ' || LASTNAME as TD,
PHONENO as TD) )
FROM DSN8810.EMP E
WHERE E.WORKDEPT = D.DEPTNO )
) ) ) )
FROM DSN8810.DEPT D, (SELECT WORKDEPT, COUNT(*)
FROM DSN8810.EMP GROUP BY WORKDEPT) X(DEPTNO, CNT)
WHERE D.DEPTNO = X.DEPTNO AND
D.DEPTNO IN ('A00', 'C01')
```
Stored procedures and RRS

- Stored procedures improvements
  - WLM management of TCBs
    - TCBs are added/removed based on WLM recommendations
  - MAX FAILURES on CREATE PROCEDURE and UDF
  - COMPJAVA (HPJ) is replaced by interpretive Java

- Creation of DB2 managed Stored Procedures not allowed
SQL procedure language

- CREATE PROCEDURE up to 2 MB
- RETURN statement
  - Return a 'condition code' for execution of SP without having to define extra parm on SP
- SIGNAL/RESIGNAL support
  - Set SQLSTATE and/or message_text
- ITERATE statement support
- Enhanced support for labels
  - Now valid at the beginning of any statement within an SQL procedure
- GET DIAGNOSTICS for all fields in SQLCA
  - Get values from RETURN

Security enhancements
Multilevel Security (MLS) with row granularity

- Table has column defined AS SECURITY LABEL
- Each row value has a specific security label
- Get security labels from RACF
- Save in rows for INSERT, UPDATE, LOAD, ...
- Check for each new seclabel value accessed
  - If access is allowed, then normal access
  - If access is not allowed, data not returned
- Runtime user-to-data checking
- Seclabel values are cached to minimize cpu
- Requires z/OS V1R5 and Security Server (RACF)
Multilevel Security with row granularity

<table>
<thead>
<tr>
<th>SECLABEL='RAINBOW'</th>
<th>SECLABEL='PASTEL'</th>
<th>SECLABEL='SUNSET'</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAINBOW</td>
<td>RAINBOW</td>
<td>RAINBOW</td>
</tr>
<tr>
<td>56</td>
<td>24</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
<td>6</td>
</tr>
<tr>
<td>76</td>
<td>65</td>
<td>45</td>
</tr>
<tr>
<td>BLUE</td>
<td>INDIGO</td>
<td>VIOLET</td>
</tr>
<tr>
<td>3</td>
<td>113</td>
<td>3</td>
</tr>
<tr>
<td>456</td>
<td>4556</td>
<td>4553</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
<td>7</td>
</tr>
<tr>
<td>RED</td>
<td>ORANGE</td>
<td>RED</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>5455</td>
</tr>
<tr>
<td>78</td>
<td>7</td>
<td>76</td>
</tr>
<tr>
<td>567</td>
<td>567</td>
<td>567</td>
</tr>
<tr>
<td>YELLOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>999</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>45</td>
</tr>
</tbody>
</table>

UNICODE evolution

- UNICODE encoding scheme can represent codepoints of characters of many different geographies and languages
  - May need more than one byte to represent many characters
  - Different languages in the same table
  - CCSID UNICODE
- Unicode for Java and XML
- V7 UNICODE data storage and manipulation
- OS/390, z/OS, z/Architecture and zSeries support (Conversion services- HW +SW)
Unicode enhancements in Version 8

- Able to join Unicode with EBCDIC or ASCII
- SQL in Unicode, statements, literals, object names, mixed CCSIDs / Utility statements can be in Unicode
- Most DB2 character catalog columns in Unicode (Unicode catalog)

www.unicode.org

NEWFUN (YES|NO)

Unicode parsing/precompiler

- Parsing is always done in Unicode in V8 (CM, ENFM, and NFM)

Application source

Modified Source

Compiler

Object Module

Linkage Editor

Load Module

Listing, Messages

EBCDIC DBRM

Unicode DBRM

Unicde Precompiler

NFM?

YES

NO

Parsing always in Unicode

DB2 Bind

Plan/Package

Runnable Program

Compiler

Object Module

Linkage Editor

Load Module
Precompilation is always done after converting the source to Unicode.

If NEWFUN=YES, you can specify the CCSID that the source program is in on the precompile.

Otherwise precompiler makes assumption that source is in CCSID 37.

Utility Unicode parsing

- Utility control statements can be provided as:
  - EBCDIC or
  - Unicode UTF-8, CCSID 1208

- Statements processed as Unicode if starting with:
  - '20'x - Unicode blank
  - '2D'x - Unicode dash
  - '41'x through '5A'x

- All SYSPRINT remains in EBCDIC

- New stored procedure DSNUTILU for Unicode input
Multiple CCSIDs per SQL statement

- A single SQL statement can now intermix EBCDIC, ASCII and Unicode tables and host variables
- Unlike host variables are translated to column CCSID
- Unlike column CCSIDs are "promoted" to Unicode
  - May require query to be materialized/sorted

```
SELECT T1.COL_EBCDIC ||
   T2.COL_ASCII  ||
   T3.COL_UNICODE
FROM T1,T2,T3;
```

Multiple CCSIDs per SQL statement - 2

- Example

```
SELECT  a.name, a.creator, b.charcol, 'ABC',
        :hvchar, X'C1C2C3'
FROM SYSIBM.SYSTABLES a, ebcdictable b
WHERE  a.name = b.name  AND
       b.name > 'B'  AND
       a.creator = 'SYSADM'
ORDER BY  b.name;
```

Result or Evaluated:

- EBCDIC
- Unicode
- Application Encoding Scheme

Assuming a Unicode catalog, the result will contain multiple CCSIDs and the comparisons and ordering will be dependent on the context.
CURRENT PACKAGE PATH

- Allows application to specify search list of package schemas (similar to PKLIST)

```
BIND PLAN(ABC) PKLIST(ALPHA.*,BETA.*,PROD.*))
```

- Important for SQLJ and DDF applications
  - Less need for SET CURRENT PACKAGESET
  - Less network traffic for multiple PKLIST values
  - Easier to switch to/from JDBC and SQLJ
  - Better handling of nested stored procedure packages

Miscellaneous enhancements

- Support for multiple instances of the same cursor

- Allows for nested SP result sets
- Affects SQLJ and remotely invoked stored procedures

- Extended DESCRIBE
  - Enabled through DESCSTAT DSNZPARM
  - Provides lots of descriptive information from prepare, describe, a query or the execution of a stored procedure
MQSeries UDFs

- Integrate MQSeries messaging operations within SQL statements
- Are already available in V7 - come in 3 flavours
  - Basic MQ scalar and table UDFs (eg. MQRECEIVE(), MQSEND() - PQ59549)
  - MQ publish/subscribe UDFs (eg. MQPUBLISH(), MQSUBSCRIBE() - PQ73340)
  - MQXML UDFs (eg. MQReadAllXML() - PQ74445)
- Set of UDFs in MQSeries Application Messaging Interface:
  - UDFs with schema name DB2MQ1C provide only single-phase commit
  - UDFs with schema name DB2MQ2C support two-phase commit
  - Separate WLM environments for each flavor of UDFs

**Application program address space**
*invoke prog A select MQSEND('1') from EMP*

**DB2 System**
*select MQSEND('1')...*

**WLM-established SP address space**
*MQSeries Functions*

---

**DB2 as a Web Services provider**

- Web services Object Relational Framework (WORF)

1. create DB2 WS Provider
2. publish WSDL
3. find WSDL
4. SOAP
5. JDBC

**DB2**
- Tables
- Stored Procedures
- XML Extender

**DB programmer or DBA**

**UDDI registry**

**WAS**

**DB2 WS Provider**

**DADX file**

**SELECT * FROM employee**
DB2 as a Web Services Consumer

- Ships as a set of user-defined functions (UDFs) that act as a SOAP client over HTTP
- SOAP body is constructed according to the WSDL

```
CREATE FUNCTION itso.getStockQuote (symbol VARCHAR(100))
RETURNS DECIMAL(5,2) SPECIFIC xmethods_getQuote
LANGUAGE SQL CONTAINS SQL
EXTERNAL ACTION NOT DETERMINISTIC
RETURN
    db2xml.extractREAL(
        db2xml.xmlclob(db2xml.soaphttpv(
            'http://66.28.98.121:9090/soap',
            'urn:xmethods-delayed-quotes#getQuote',
            varchar('
                <!-- getQuote xmlns:m="urn:xmethods-delayed-quotes" -->
                <m:getQuote>
                    <symbol xsi:type="xsd:string">IBM</symbol>
                </m:getQuote>
            '/*/Result'))
    )
VALUES itso.getStockQuote('IBM')
```

WSAD can help you build this UDF

Invoking the getStockQuote UDF

VALUES itso.getStockQuote('IBM')
Utilities: DB2 for z/OS

- Delimited LOAD and UNLOAD
- RUNSTATS - distribution statistics
- Rebalancing partitions
- Online REORG enhancements
- REORG/REBUILD SCOPE PENDING
- Online schema changes support
- New defaults for better performance
- Backing up and restoring system

LOAD/UNLOAD delimited input/output

- LOAD / UNLOAD utilities will accept / produce delimited files

Benefits:
- Ease the import / export of (large amounts of) data from DB2 for z/OS to other operating system platforms and vice versa
- Eliminates requirement to write programs to convert non-z/OS platform data into positional format for DB2 for z/OS LOAD utility or to use INSERT processing
- Unload/ export data from another DBMS in delimited file format and LOAD it into DB2 for z/OS
Example of UNLOAD statement

```
UNLOAD TABLESPACE  databasename.tablespacename
  DELIMITED CHARDEL '"' COLDEL ';' DECPRT '"'
  PUNCHDDN SYSPUNCH
  UNLDDN SYSREC EBCDIC
  FROM TABLE  tablename

(LNAME   POSITION(*) VARCHAR(15),
  DEPTNO  POSITION(*) CHAR(4),
  SEX     POSITION(*) CHAR(2),
  COUNTRY POSITION(*) DBCLOB(11),
  SALARY  POSITION(*) DECIMAL(8,2),
  SALARYRATE POSITION(*) FLOAT )
```

Unloaded data looks like:
```
#warren#; #D10#; #m #; #U S A#; #6500.00#; #.5E+1#
```

Note that field lists are optional for LOAD / UNLOAD and are primarily used for selecting a subset of columns or selecting data in a different order.

RUNSTATS enhancements

- **Collection of non-uniform distribution statistics on non-index columns**
  - V7 technique is to use a separate program DSTATS
  - V8 RUNSTATS improvement allows collection of frequency and non-uniform distribution statistics on columns that aren't part of an index
  - Used by optimizer and can lead to significant performance improvements of certain queries
  - Not supported for inline statistics

- **Example:**
  ```
  RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
  TABLE(DSN8810.DEPT)
  COLGROUP(EMPLEVEL,EMPGRADE,EMPSALARY)
  FREQVAL COUNT 10
  ```

© 2006 IBM Corporation
RUNSTATS enhancements -2

- Can also collect LEAST, MOST, BOTH distribution and correlation statistics
- RUNSTATS with UPDATE NONE REPORT NO
  - This invalidates the DSC for objects referenced in the RUNSTATS statement
- RUNSTATS only updating HISTORY tables

Rebalancing a tablespace or partition range

Sets new partition boundaries for even distribution of rows across the partitions being reorganized

**Before REORG TABLESPACE**

![Partitioning Index](image)
![Data Partitions](image)

**After REORG TABLESPACE ... REBALANCE**

![Partitioning Index](image)
![Data Partitions](image)

LKa='50000'  LKa='80000'
LKa='30000'  LKa='80000'
### Online REORG Enhancements

- No BUILD2 phase for REORG PART when using DPSIs

- Support for DISCARD processing with OLR SHRLEVEL CHANGE

- Support for doing OLR SHRLEVEL REFERENCE of all catalog tables (including those with links)

### REORG TABLESPACE - SCOPE PENDING

Example

<table>
<thead>
<tr>
<th>Part</th>
<th>ts</th>
<th>DBET</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jan_2001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Feb_2001</td>
<td>AREO*</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Mar_2001</td>
<td>AREO*</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Oct_2005</td>
<td>REORP</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Nov_2005</td>
<td>REORP</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Dec_2005</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parts 2, 3, 13, 14 will be REORG'd
COPY enhancements - system pages

- Whether the dictionary and version system pages of the object are copied at the beginning of output data set
  - YES - ensures that the dictionary and version system pages are at the beginning
  - NO - copy pages as they appear in the object (pre-V8)
- Improvements over V7
  - Header page will always be included no matter what SYSTEMPAGES option is used
  - With SYSTEMPAGES YES, compression dictionary will be included (independent of new online schema function)
  - V8 version data will also be copied with SYSTEMPAGES YES
- Note that for SYSTEMPAGES YES, the UNLOAD utility can process image copies with data versioning
COPY, QUIESCE, LISTDEF, TEMPLATE

COPY
- DSNUM may specify a partition of a DPSI

QUIESCE
- Claim classes and restrictive states for DPSIs mirror PIs:
  - WRITE YES: Partitions are DW / UTRO
  - WRITE NO: No claims or restrictive states
- NPSIs are DW / UTRO during WRITE YES for the table space or any partition of the table space

LISTDEF
- PARTLEVEL keyword specifies the partition granularity for partitioned objects -- now extended to DPSIs

TEMPLATE
- Templates created for DPSIs may wish to make use of the &PA OBJECT variable

New defaults for better performance and usability

- RESTART is new default for Utilities
- SORTKEYS for LOAD, REORG, and REBUILD
- SORTDATA for REORG
  - SORTDATA now allowed for 32K records
- REORG will use implicit clustering index
  - if no clustering index, first index defined is used
  - If table space has no indexes, SORTDATA operates as in pre-V8 releases
Online CHECK INDEX

- V7 CHECK INDEX causes data and indexes to be unavailable for update for the duration of CHECK operation
- Online CHECK INDEX different design than Online REORG
- Claim as reader for target data and indexes
- Create shadow datasets
  - same dataset naming convention as Online REORG
  - cannot run Online CHECK INDEX on two logical parts of NPI
- Drain writers for target data and indexes
- Flash data and indexes from target to shadows
- After copy logically complete for ALL
  - dedrain target data and indexes
  - run parallel check index on shadow data and indexes
    - same parallel design as REBUILD INDEX
- At utilterm delete shadow datasets when DB2 managed

Single NPI on partitioned
Cross Loader support for > 32K (rows with) LOBs

- Architectural limits within LOAD/UNLOAD did not allow for a record greater than 32K to be loaded or unloaded
- New buffering scheme for LOB values to bypass the 32K limit
- Will be constrained by region size
- Cross Load of 2GB LOBs will still not be possible
- Cross Loader will also allow for conversion between CLOBs and DBCLOBs
  - not currently supported when loaded from file
VOLATILE table support

- Tables whose contents can vary from empty to very large at runtime
- Encourages index access for tables that have unpredictable cardinality
- Significant performance improvement for some SAP applications (cluster tables)
- Can avoid locking conflicts caused by different access paths from different queries accessing the same table

```
CREATE TABLE XYZ ...... VOLATILE
```

Table function improvements

- **Cardinality in catalog (V7)**
  - Table UDF cardinality in SYSIBM.SYSROUTINES
  - Populated by CREATE TABLE UDF
- **Specifies estimate of expected number of rows returned by UDF reference**

- **V8 enhancement:**
  - CARDINALITY clause
    - SELECT * FROM TABLE (TABUDF(C2) CARDINALITY 30) AS ABC
  - CARDINALITY multiplier
**REOPT(ONCE)**

- Bind option that controls when the optimizer builds the access path information for dynamic SQL applications
  - By default, access path is calculated at PREPARE
  - Using REOPT(ONCE)
    - Defers access path selection until OPEN
    - Values of host variables on OPEN are used to calculate access path
    - Resulting access path is cached in the global (prepared) statement cache
  - REOPT(NONE) = NOREOPT(VARS)
  - REOPT(ONCE)
  - REOPT(ALWAYS) = REOPT(VARS)

---

**EXPLAIN improvement**

- Enhancements to the EXPLAIN statement allow you to obtain EXPLAIN information for entries in the DB2 global statement cache.
- Visual Explain is enhanced to exploit this new function.
Intelligent Visual Explain

- Significant improvements to Visual Explain tool:
  - Much more detailed information about access path
  - More statistical details for each node in the graph
  - XML document describing query access path selected
  - Easier collection of information to send for help

DB2 for z/OS Version 8 material
Latest DB2 Redbooks

- DB2 UDB for z/OS V8 Everything you ever wanted to know, ...and more, SG24-6079
- DB2 UDB for z/OS Version 8 Performance Topics, SG24-6465
- DB2 UDB for z/OS V8: Through the Looking Glass & What SAP Found There, SG24-7088
- DB2 for z/OS and OS/390: Ready for Java, SG24-6435
- Distributed Functions of DB2 for z/OS and OS/390, SG24-6952
- Moving Data Across the DB2 Family, SG24-6905
- Implementing PeopleSoft 8.4 on zSeries, SG24-6549
- Siebel 7 with DB2 for z/OS: Database Implementation & Administration, SG24-6868
- SAP on DB2 for z/OS: Multiple Components in One Database, SG24-6914
- SAP on DB2 for z/OS: High Availability Using System Automation, SG24-6836
- Squeezing the Most Out of Dynamic SQL, SG24-6418
- Large Objects with DB2 for z/OS and OS/390, SG24-6571

DB2 for z/OS on the Web

- Full URL for DB2 for z/OS page
- DB2 for z/OS Version 8 related material
- Direct URL for Support page
  http://www.ibm.com/software/data/db2/xos/support.html
- Presentations page:
- DB2 Developer Domain articles
  http://www.ibm.com/developerworks/db2/