DB2 11 is Great!
Why Should You Care?

Sheryl M. Larsen
IBM WW DB2 for z/OS Evangelist
smlarsen@us.ibm.com
Sheryl M. Larsen
smlarsen@us.ibm.com

Sheryl Larsen is an internationally recognized researcher, consultant and lecturer, specializing in DB2 and is known for her extensive expertise in SQL. She co-authored a book, DB2 Answers, Osborne-McGraw-Hill, 1999.

She now works for IBM, but this material was developed before joining IBM and is the culmination of 25 years as an external consultant specializing in superhuman efforts to tune customer workloads.

Her new title is IBM’s DB2 for z/OS Evangelist. Other titles include:
President of the Midwest Database Users Group,
IDUG Hall of Fame Speaker,
IDUG Hall of Fame Volunteer, and
Northern Illinois University Computer Science Alumni Council

Sheryl has over 25 years experience in DB2, has published articles, white papers, webcasts: bmc.com, ca.com, softbase.com, ibm.developerworks.com
Disclaimer/Trademarks

© Copyright IBM Corporation 2013. All rights reserved.  
U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

THE INFORMATION CONTAINED IN THIS DOCUMENT HAS NOT BEEN SUBMITTED TO ANY FORMAL IBM TEST AND IS DISTRIBUTED AS IS. THE USE OF THIS INFORMATION OR THE IMPLEMENTATION OF ANY OF THESE TECHNIQUES IS A CUSTOMER RESPONSIBILITY AND DEPENDS ON THE CUSTOMER’S ABILITY TO EVALUATE AND INTEGRATE THEM INTO THE CUSTOMER’S OPERATIONAL ENVIRONMENT. WHILE IBM MAY HAVE REVIEWED EACH ITEM FOR ACCURACY IN A SPECIFIC SITUATION, THERE IS NO GUARANTEE THAT THE SAME OR SIMILAR RESULTS WILL BE OBTAINED ELSEWHERE. ANYONE ATTEMPTING TO ADAPT THESE TECHNIQUES TO THEIR OWN ENVIRONMENTS DO SO AT THEIR OWN RISK.

ANY PERFORMANCE DATA CONTAINED IN THIS DOCUMENT WERE DETERMINED IN VARIOUS CONTROLLED LABORATORY ENVIRONMENTS AND ARE FOR REFERENCE PURPOSES ONLY. CUSTOMERS SHOULD NOT ADAPT THESE PERFORMANCE NUMBERS TO THEIR OWN ENVIRONMENTS AS SYSTEM PERFORMANCE STANDARDS. THE RESULTS THAT MAY BE OBTAINED IN OTHER OPERATING ENVIRONMENTS MAY VARY SIGNIFICANTLY. USERS OF THIS DOCUMENT SHOULD VERIFY THE APPLICABLE DATA FOR THEIR SPECIFIC ENVIRONMENT.

Trademarks
IBM, the IBM logo, ibm.com, and DB2 are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

IBM - A global computer products, services and solutions company based in the United States, IBM operates in more than 170 countries and has nearly 427,000 employees. With thousands of business partners and computer products suppliers worldwide, IBM generates more than USD99 billion in annual revenue. The company was founded in 1911 in Endicott, New York.
DB2 11 for z/OS Enhancements for Developers

DB2 Engine Components
Predicate Processing Intelligence
Optimizer Details
DB2 11 Performance Sweet Spots
Easier DB2 Version Upgrade – application compatibility
Big Data Capabilities

Performance Improvements
- no REBIND needed
- REBIND required (with or without APREUSE)
- REBIND required (without APREUSE)
- DBA or application effort required

DDF Performance Improvements
SQL changes/new features for DB2 V11
Expanded Analytics Capabilities in DB2 11
DB2 Engine Components

SQL Execution

Stage 2 – z/OS
- Optimizer
- Dynamic Statement Cache
- RID Pool

Stage 1 – z/OS
- Meta Data
- Access Plans

Catalog

Directory

Buffer Manager

Data

Index

Work Files

Result

Buffer Pool
Page Processing – z/OS

Buffer Pool

Buffer Manager

Data

Index

Stage 1

Stage 2

Work Files

4K, 8K, 16K, 32K
### Indexable Stage 1 Predicates

<table>
<thead>
<tr>
<th>Predicate Type</th>
<th>Indexable</th>
<th>Stage 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL = value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL = noncol expr</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS NULL</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL op value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL op noncol expr</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL BETWEEN valuel AND value2</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL BETWEEN noncol expr 1 AND noncol expr 2</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL BETWEEN expr-1 AND expr-2</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL LIKE 'pattern'</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL IN (list)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS NOT NULL</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL LIKE host variable</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL LIKE UPPER 'pattern'</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL = T2.COL</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL op T2.COL</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL = T2.col expr</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL = (noncor subq)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL = ANY (noncor subq)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>(COL1,...COLN) IN (noncor subq)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS NOT DISTINCT FROM value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS NOT DISTINCT FROM noncol expr</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL1 IS NOT DISTINCT FROM T2.COL2</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL1 IS NOT DISTINCT FROM T2 col expr</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### New Stage 1 Predicates

<table>
<thead>
<tr>
<th>Predicate Type</th>
<th>Indexable</th>
<th>Stage 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSTR(COLX, 1, n) = value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DATE(TS_COL) = value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>YEAR(DT_COL) = value</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>CASE WHEN THEN ELSE END = value</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### Stage 1 Predicates

<table>
<thead>
<tr>
<th>Predicate Type</th>
<th>Indexable</th>
<th>Stage 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL &lt;&gt; value</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL &lt;&gt; noncol expr</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL NOT BETWEEN value1 AND value2</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL NOT IN (list)</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL NOT LIKE 'char'</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL LIKE '%'char'</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>T1.COL &lt;&gt; T2.col expr</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL op (noncor subq)</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL op ANY (noncor subq)</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL op ALL (noncor subq)</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS DISTINCT FROM value</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COL IS DISTINCT FROM (noncor subq)</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

### Four Points of Filtering

1. **Indexable** = The predicate is a candidate for Matching Index access. When the optimizer chooses to use a predicate in the probe of the index, the condition is named Matching (matching the index). This is the first point that filtering is possible in DB2.

2. **Stage 1 Index Screening** = The Stage 1 predicate is a candidate for filtering on the index leaf pages. This is the second point of filtering in DB2.

3. **Stage 1 Data Screening** = The Stage 1 predicate is a candidate for filtering on the data pages. This is the third point of filtering in DB2.

4. **Stage 2** = The predicate is not listed as Stage 1 and will be applied on the remaining qualifying pages from Stage 1. This is the fourth and final point of filtering in DB2.
DB2 11 New Indexable Stage 1 Predicates

- No longer worry about these:
  - WHERE value BETWEEN COL1 AND COL2
  - WHERE SUBSTR(COLX, 1, n) = value
  - WHERE SUBSTR(COLX, 1, n) \( op \) value
  - WHERE DATE(TS_COL) = value
  - WHERE DATE(TS_COL) \( op \) value
  - WHERE YEAR(DT_COL) = value
  - WHERE YEAR(DT_COL) \( op \) value
  - WHERE value BETWEEN col expr AND col expr
  - WHERE CASE expr = value

In the DB2 11 documentation
Summary of Predicate Processing

The predicate \textit{might} be indexable if \textit{expr} contains one of the following scalar functions:
\begin{itemize}
\item DATE
\item YEAR
\item SUBSTR (if the start value for the substring is 1.)
\end{itemize}
Four Points of Filtering – DB2

1. Indexable Stage 1 Probe
2. Stage 1 Index Filtering
3. Stage 1 Data Filtering
4. Stage 2

WHERE C.LAST_NM LIKE ?
AND C.TOKEN_NR = B.TOKEN_NR
AND C.ROLE_CD > ?
AND CASE C.SEX WHEN ‘X’ THEN ‘ABCDE’ END = ?
Visual Plan Graphs - Simple
Visual Plan Graphs - Busy
DB2 11 OLTP/Batch Performance Expectations

- These are results from IBM testing
- Performance expectations vary depending on many factors, including:
  - Access path selection, Read/Write ratio, Number of rows returned
  - Number and type of columns returned, Number of partitions touched
  - Schema - Number of partitions defined, DPSI, etc
  - RELEASE option, data compression
  - Query costing, query rewrite, index skipping

![DB2 11 CPU saving in OLTP/Batch (% of Total DB2 CPU reduction)]
DB2 11 Performance Sweet Spots

- Write Intensive Batch
- Queries
  - With compressed tables
  - With access path improvement
  - With sort intensive workload
  - Accessing multiple DPSI partitions
  - IDAA with large result sets
- Online transactions
  - Write intensive transactions
  - With large # of partitions (>200 partitions) with REL(COMMIT)
  - With large buffer pools
  - With queries returning a large number of columns
  - Chatty DDF applications with z/OS Communications Server
- Cost saving from zIIP eligible address space SRB time
  - DBM1 in data sharing
  - MSTR address space for update intensive workloads
Impressive DB2 11 Performance Results!

DB2 11 % CPU Improvement From DB2 10

TPC-H queries
TPC-H like queries
Customer queries 3
Customer queries 2
Customer queries 1
SAP BW queries
Cognos BI-Day Long
Cognos BI-Day short
TPC-H executed in IDAA

TSO Batches DSHR extended RBA
TSO Batches non-SHR
High Insert Seq

SAP Banking (60 M) dshr 2way
Local OLTP
TPC-E Brokage (rel com ) CM
IRWW DS (rel com) DSHR
IRWW DS extended RBA
High Insert Random
Dist IRWW
Dist IRWW sproc

XML scenario

© 2013 IBM Corporation
DB2 11 The Foundation for Business Critical Analytics

- OLTP Transactions
- Real time data ingestion
- High concurrency
- Operational analytics
- Standard reports
- OLAP
- Complex queries

DB2 Native Processing

- DB2 11 CPU savings benefit query workloads with or without IBM DB2 Analytics Accelerator
DB2/Accelerator Sweet Spots Do Not Overlap

DB2 11 and Big Insights provide this extended capability.
Enhancing DB2 11 Analytics on “z” with Big Data

- Providing the connectors – User Defined Functions
- Providing capability to allow DB2 applications to easily and efficiently access these data sources

FS results on IBM BigInsights platform

Hadoop Distributed File System

• New user-defined functions
• New generic table UDF capability
DB2 11 Support for Big Data Details

- Analytic jobs can be specified using JSON Query Language (Jaql)
  - Submitted to IBM’s Hadoop based BigInsights platform
  - Results stored in HDFS format
- A new generic, polymorphic, TABLE User Defined Function (HDFS_READ) reads the Bigdata analytic result from HDFS
  - For subsequent use in an SQL query
- Must have a variable shape of HDFS_READ output table
  - Its output schema is determined at query run-time
  - DB2 11 supports generic table UDFs, enabling this function
- A new function, JAQL_SUBMIT, enables invocation of IBM BigInsights Jaql from a DB2 application
  - Returns the correct URL string for the HDFS result in VARCHAR(512) or ‘’
Typical Customer Big Data Use Case

1.) Email from all clients sent to an insurance company ingested into IBM’s BigInsights platform

2.) DB2 kicks off a Hadoop job on BigInsights - analyzes the emails and identifies customers who have expressed dissatisfaction

3.) BigInsights creates a file of results (names and email addresses of customers at risk)

4.) DB2 reads the BigInsights result file

   ```sql
   SELECT A.* FROM AGENT A JOIN TABLE(HDFS_Read(JAQL_SUBMIT result,'')) AS .....;
   ```

DB2 joins the result with the Agent table and sends list of at-risk customers to their mobile device
More HDFS_Read Examples

- INSERT INTO T1 SELECT a, b FROM TABLE
  (HDFS_Read('http://9.30.11.27/data/controller/dfs/file1.csv', ''))
  AS X(A INT, B DOUBLE, C INT);

- SELECT T1.*, hd.a, hd.b
  FROM T1 JOIN TABLE
  (HDFS_Read('http://9.30.11.27/data/controller/dfs/file1.csv', '')) AS X(A INT, B DOUBLE, C INT)
  ON T1.x = X.c;

- CREATE VIEW hadoopData AS
  SELECT * FROM TABLE (HDFS_Read('http://9.30.11.27/data/controller/dfs/file1.csv', ''))
  AS X(A INT, B DOUBLE, C INT);
DB2 and IBM zIIP Add Value to Database Work

Portions of the following DB2 workloads in enclave SRB mode are eligible for zIIP*

   DB2 9 in blue    DB2 10 in green    DB2 11 in red

1. DRDA over TCP/IP connections: up to 60% of the processing
   - DB2 9 for zOS remote native SQL procedures
   - DB2 9 XML parsing, schema validation

2. Requests that use parallel queries: up to 80% of the processing after reaching a CPU usage threshold
   - DB2 9 and DB2 10 remove restrictions for query parallelism enabling more queries to run with parallelism and therefore to potentially increase zIIP eligibility

3. DB2 utilities: up to 100% of the processing
   - LOAD, REORG and REBUILD functions used to maintain index structures and sort
   - DB2 10 RUNSTATS – options other than column group, inline
   - DB2 11 RUNSTATS column group and inline

4. Asynchronous processing that is charged to a DB2 address space (introduced in DB2 10, expanded in DB2 11): up to 100% of the processing
   - DB2 10 buffer pool prefetch and deferred write
   - All other such asynchronous processing, except for P-lock negotiation

* NOTE: This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs) (“SEs”). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.
Easier DB2 Version Upgrade – application compatibility

- New DB2 releases can introduce SQL behavior changes which can break existing applications
  - For example, changes for SQL standards compliance
  - Example: DB2 10 CHAR function with decimal input no longer returns leading zeros when there is a decimal point

- Application Compatibility (APPLCOMPAT) – new option for enforcement
  - APPLCOMPAT(WARN)
  - Provide mechanism to identify applications affected by SQL changes
  - Provide seamless mechanism to make changes at an application (package) level or at a system level
  - APPLCOMPAT(VnnR1) – nn is the DB2 Version Number. V10R1 is the lowest release of DB2 catered for

- CURRENT APPLICATION COMPATIBILITY – Dynamic SQL special register
  - This mechanism will enable support for up to two back level releases (N-2)
  - The release after DB2 10 will be the initial deployment of this capability
  - DB2 10 will be the lowest level of compatibility supported
Migration  DB2 10 → DB2 11 (V11R1)

DB2 10 New Function Mode (NFM) With SPE

DB2 10 Catalog

DB2 10 Libraries

CATMAINT UPDATE (DSNTIJTC)

DB2 11 Conversion Mode

CATENFM START (DSNTIJEN)

DB2 11 Enabling New Function Mode

CATENFM COMPLETE (DSNTIJNF)

DB2 11 New Function Mode (NFM)

DB2 11 Catalog

DB2 11 Libraries

1 – 2 months

1 week

Data Sharing Coexistence

Bind with APPLCOMPAT(V10R1) option only

Bind with APPLCOMPAT(V10R1) or APPLCOMPAT(V11R1)

2 hours
Migration DB2 11 → DB2 11 + 1 (VnnR1)

DB2 11 New Function Mode (NFM) With SPE

DB2 11 Catalog

DB2 11 Libraries

CATMAINT UPDATE
(DSNTIJTC)

DB2 11+1 Conversion Mode

CATENFM START
(DSNTIJEN)

DB2 11+1 Enabling New Function

DB2 11 New Function Mode (NFM)

CATENFM COMPLETE
(DSNTIJNF)

DB2 11+1 Catalog

DB2 11+1 Libraries

1 – 2 months

1 week

2 hours

Data Sharing Coexistence

Bind with APPLCOMPAT(V10R1)

Bind with APPLCOMPAT(V10R1) or APPLCOMPAT(V11R1)

Or APPLCOMPAT(VnnR1)
Easier DB2 Version Upgrade...

- Faster ENFM processing
  - Lab measurement showed 16x faster in V11 vs. V10 using a large customer catalog
  - Note: V11 ENFM performance is sensitive to size of SYSLGRNX. Consider running MODIFY RECOVER to clean up old entries if SYSLGRNX is very large

- Access path stability improvements
- Higher code quality stability levels
- New workload Capture/Replay tooling can help testing of DB2 version upgrades
Performance Improvements – no REBIND needed

- DDF performance improvements
  - Reduced SRB scheduling on tcp/ip receive using new CommServer capabilities
  - Improved autocommit OLTP performance
  - DRDA package based continuous block fetch

- xProcs above the bar
  - 31-bit Vstor relief enabled by RMODE 64 support in z/OS 1.13 and above
  - Enables other internal performance improvements

- zIIP enablement for all SRB-mode DB2 system agents that are not response time critical

- Avoid cross-memory overhead for writing log records

- INSERT performance
  - Latch contention reduction for classes 6, 14, 19
  - CPU reduction for Insert column processing and log record creation
  - Data sharing LRSN spin avoidance
  - Page fix/free avoidance in GBP write
Performance Improvements – no REBIND needed…

- Automatic index pseudo delete cleanup
  - DBA work would be required for fine tuning
- ODBC/JDBC type2 performance improvements
  - Stored procedure invocation
- Java stored procedure multi-threading improvements
- Sort performance improvements
- DPSI performance improvements for merge
- Performance improvements with large number of partitions
- XML performance improvements
- Optimize RELEASE(DEALLOCATE) execution so that it is consistently better performing than RELEASE(COMMIT)
  - Monitor # parent locks and cleanup internal structures when threshold is hit
- IFI 306 filtering capabilities to improve Replication capture performance
- Utilities performance improvements
Performance Improvements – no REBIND needed…

- ACCESS DATABASE command performance
- DGTT performance improvements
  - Avoid incremental binds for reduced cpu overhead
- P-procs for LIKE predicates against Unicode tables
- Improved performance for ROLLBACK TO SAVEPOINT
- zEC12 integration for performance improvements
  - Pageable 1M frames for DB2 CPU savings (requires Flash Express)
    - Buffer pool control structures (retrofit to V10)
    - DB2 executable code (requires z/OS 2.1)
  - 2G page frame size to position for extremely large main memory sizes
  - Optimizer CPU and I/O cost balancing improvements (can also benefit z196 and z10)
- Latch contention reduction and other high n-way scalability improvements
- Data sharing performance improvements
- LRSN spin reduction with extended LRSNs
- Castout performance
- GBP write-around
- Index split performance
Performance Improvements – REBIND required (with or without APREUSE)

- Query transformation improvements – less expertise required to write performance SQL
  - Enhanced query rewrite to improve predicate indexability
    - new situations where non-indexable predicates can be rewritten by Optimizer to be indexable
    - Convert some common stage 2 predicates to indexable (YEAR(), DATE(), SUBSTR(col,1,x), value BETWEEN COL1 AND COL2)
    - Improved indexability for OR COL IS NULL predicates
    - Push complex predicates inside materialized views/table expressions
  - Enhanced pruning of "always true" and "always false" predicates

- Enhanced duplicate removal
  - Lots of queries require duplicate removal: e.g. DISTINCT, GROUP BY, etc.
  - Dup elimination via sorting can be expensive
  - New techniques: Index duplicate removal, early out
  - Will not show in Explain table, need to look at IXSCAN_SKIP_DUPS column in DSN_DETCOST_TABLE to determine if sort avoided

- DDF and RDS runtime result set optimizations
  - Reduced DB2 CPU for IDAA queries
Performance Improvements – REBIND required (with or without APREUSE)...

- In-memory techniques
  - In-memory, reusable workfile
  - Sparse index (limited hash join support)
  - Non-correlated subquery using MXDTCACH
  - Correlated subquery caching

- Non correlated subquery with mismatched length

- Data decompression performance improvement

- Select list do-once
  - Non column expressions in the select list can be executed once rather than per-row

- Column processing improvements
  - Xproc (generated machine code) for output column processing
  - Optimized machine instructions for input/output column processing
Performance Improvements – REBIND required (with or without APREUSE)…

- RID overflow to workfile handled for Data Manager set functions
  - DB2 10 added RID overflow to workfile
  - DB2 11 adds support for set functions (COUNT, MAX, MIN etc) which was excluded in DB2 10

- Performance improvements for common operators
  - MOVE, CAST, output hostvar processing, CASE, SUBSTR, DATE, others

- DECFLOAT data type performance improvements
  - Up to 23% CPU reduction for conversion to/from decfloat
  - Approx. 50% cpu reduction in INSERT, FETCH for decfloat columns
  - Helped further by zEC12 hw improvements for decimal floating point
Performance Improvements – REBIND required (without APREUSE)

- DPSI and page range performance improvements
  - Page range screening for join/correlation predicates
  - Parallelism optimization for DPSI access

- Optimizer CPU and I/O cost balancing improvements
  - Measured results: 3% to >30% performance improvement for query workloads
Performance Improvements – DBA or application effort required

- Suppress-null indexes
  - Index entries not created when all values for indexed columns are NULL
  - Reduced index size, improved insert/update/delete performance, compatibility with other DBMSes
  - Improved utility CREATE INDEX performance

- New PCTFREE FOR UPDATE attribute to reduce indirect references

- DGTT performance improvements
  - Non logged DGTTs

- Global variables
  - Easier, more efficient sharing of data between SQL statements
Performance Improvements – DBA or application effort required

- Extended optimization - selectivity overrides (filter factor hints)
  - Improve optimizer’s ability to find the cheapest access path
  - Collect filter factors for predicates in a Selectivity Profile
  - Selectivity Profile is populated via BIND QUERY

- Open dataset limit raised to 200K

- Optimizer externalization of missing/conflicting statistics
  - Identify missing statistics during bind/prepare/explain
  - DBA or tooling to convert output to RUNSTATS input
Current Process for Populating Statistics

Which statistics to collect?

TABLE(ALL)
INDEX(ALL)
KEYCARD*
HISTOGRAMS
COLGROUP
MCARD
Etc.

Optimizer

Real-time Statistics
(RTS)

SYSTABLE-SPACESTATS

SYSINDEX-SPACESTATS

RUNSTATS

TS1
TS2
IX1
IX2

STATSINT
DSNZPARM - minutes

in memory statistics

© 2013 IBM Corporation
DB2 11 – Optimizer Externalization of Missing Statistics

- Statistics in Catalog Tables
- SYSTABLE-SPACESTATS
- SYSINDEX-SPACESTATS
- Real-time Statistics (RTS)

Optimizer

- Missing stats?
- Conflicting stats?

- in memory recommendations
- STATSINT
- DSNZPARM - minutes
- SYSSSTAT- FEEDBACK
- Tooling
- RUNSTATS
IBM SQL History - DB2 V1 vs. DB2 9 SQL

Table Expressions, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions, Limited Fetch, Scrollable Cursors, UNION Everywhere, MIN/MAX Single Index Support, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Table Support, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, IS NOT DISTINCT FROM, ON COMMIT DROP, Transparent ROWID Column, GET DIAGNOSTICS, Stage1 unlike data types, Multi-row INSERT, Multi-row FETCH, Dynamic Scrollable Cursors, Multiple CCSIDs per statement, Enhanced UNICODE, and Parallel Sort, TRUNCATE, DECIMAL FLOAT, VARBINARY, optimistic locking, FETCH CONTINUE, MERGE, call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, client special registers, long SQL Object names, SELECT FROM INSERT, UPDATE, DELETE, MERGE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY IN subselect and fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables
Stage1 unlike data types, Multi-row INSERT, FETCH, Multi-row cursor
UPDATE, Dynamic Scrollable Cursors, Multiple CCSIDs per statement, GET
DIAGNOSTICS, Enhanced UNICODE, IS NOT DISTINCT FROM, VARBINARY,
FETCH CONTINUE, MERGE

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation,
Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited
Fetch, Insensitive Scrollable Cursors, UNION Everywhere, MIN/MAX Single Index Support,
Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row
Expressions 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect,
Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT
PACKAGE PATH, VOLATILE Table Support, Star Join Sparse Index, Qualified Column
names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, call
from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT
SCHEMA, client special registers, long SQL Object names, SELECT FROM INSERT,
UPDATE, DELETE, MERGE, INSTEAD OF TRIGGER, Native SQL Procedure Language,
BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY IN subselect and
fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables, DECIMAL
FLOAT, XQuery, TRUNCATE, OLAP Functions, Session variables, OmniFind, Spatial, ROLE
GROUPING SETS, ROLLUP, CUBE, Many Built-in Functions, SET CURRENT
ISOLATION , multi-site join, MERGE, ARRAY data type, global variables, Oracle
syntax, XML enhancements
DB2 for z/OS Best Practices

DB2 10 z/OS vs. DB2 10 LUW

Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encoding schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, MERGE, SELECT from MERGE, routine versioning, timestamps w/timezone

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables, OmniFind, spatial, range partitions, data compression, session variables, DECIMAL FLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, date/time changes, currently committed, moving sum & average, index include columns, row and column access control, time travel query, XML enhancements

Updateable UNION in Views, GROUPING SETS, ROLLUP, CUBE, more Built-in Functions, SET CURRENT ISOLATION, multi-site join, MERGE, MDC, XQuery, additional data type (array, row, cursor), global variables, even more vendor syntax, temp table compression, MODULEs
DB2 11 SQL – Standard SQL support  
(not exhaustive, some features may be missing)

DB2 11 for z/OS and DB2 10.5 Linux, Unix & Windows

Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encoding schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, SELECT FROM MERGE, MERGE, routine versioning, transparent archive query

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY, KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, SQL PL in routines, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, MERGE not logged tables, OmniFind, spatial, range partitions, data compression, DECFLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, increased timestamp precision, currently committed, moving sum & average, index include columns, row and column access controls, time travel query, GROUPING SETS, ROLLUP, CUBE, global variables, Text Search functions, accelerated tables, DROP COLUMN, array data type, XML enhancements

Updateable UNION in Views, more Built-in Functions, SET CURRENT ISOLATION, multi-site join, full MERGE, MDC, XQuery, additional data type (row, cursor), even more vendor syntax, temp table compression, MODULEs
Expanded Analytics Capabilities...

- **SQL Grouping Sets, including Rollup, Cube**
  - Rollup is helpful in providing subtotaling along a hierarchical dimension such as time or geography
  - CUBE is helpful in queries that aggregate based on columns from multiple dimensions

- **DB2 support for IDAA V3 and V4 (rolled back to V10)**
  - Support for static SQL
  - Propagating DB2 changes to the accelerator as they happen – V11 improved CDC capture performance with new IFI 306 filtering capabilities
  - Detect staleness of data via RTS
  - Reducing disk storage cost by archiving data in the accelerator and maintaining the excellent performance for analytical queries: *High Performance Storage Saver*
  - Workload Manager integration and better monitoring capabilities
  - Increasing the query off-load scope via new special register `CURRENT QUERY ACCELERATION`

- **High performance SPSS in-database scoring via PACK/UNPACK (rolled back to v10)**

- **Hadoop access via table UDF**
  - UDFs shipped with BigInsights
  - Uses new V11 generic table UDF capability

- **JSON support**
GROUP BY GROUPING SETS

- A grouping sets specification allows multiple grouping clauses to be specified in a single statement.
- DB2 11 introduces support for 3 additional variants of GROUP BY
  - GROUP BY GROUPING SET: fundamental building block for GROUP BY operations
  - GROUP BY ROLLUP: Produces sub-total rows in addition to regular grouped rows.
  - GROUP BY CUBE: Produces row summaries and grand totals

**Example:** Create a result set from the SALES table based on person and date.

```
SELECT WEEK(SALES_DATE) as WEEK, DAYOFWEEK(SALES_DATE) AS DAY,
       SALES_PERSON, SUM(SALES) AS SOLD
FROM SALES
WHERE SALES_DATE > '1999-12-31'
GROUP BY GROUPING SETS
(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON)
```
GROUP BY GROUPING SETS results

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DAY</th>
<th>SALES_PERSON</th>
<th>SOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NULL</td>
<td>GOUNOT</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>NULL</td>
<td>LEE</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>NULL</td>
<td>LUCCHESI</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>NULL</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>NULL</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>NULL</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>NULL</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>NULL</td>
<td>145</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>NULL</td>
<td>8</td>
</tr>
</tbody>
</table>
**GROUP BY ROLLUP**

- An extension to the GROUP BY clause

  Produces a result set that contains “sub-total” rows.

  The sequence of the columns is significant.

GROUP BY ROLLUP (a, b, c) is equal to

GROUP BY (a, b, c) + GROUP BY (a, b) + GROUP BY (a) + grand-total

**Example**

```sql
SELECT WEEK(SALES_DATE) AS WEEK, DAYOFWEEK(SALES_DATE) AS DAY,
       SALES_PERSON, SUM(SALES) AS SOLD
FROM SALES
WHERE SALES_DATE > '1999-12-31'
GROUP BY ROLLUP
       (WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON)
ORDER BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
```
### GROUP BY ROLLUP results

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DAY</th>
<th>SALES_PERSON</th>
<th>SOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>LEE</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>LEE</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>LUCCHESI</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>GOUNOT</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>LUCCHESI</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>LUCCHESI</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>LUCCHESI</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>LUCCHESI</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>GOUNOT</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>LEE</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>LUCCHESI</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>NULL</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>LUCCHESI</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>LUCCHESI</td>
<td>43</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>NULL</td>
<td>145</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>NULL</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>NULL</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>7</td>
<td>LUCCHESI</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>NULL</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>NULL</td>
<td>8</td>
</tr>
<tr>
<td>21</td>
<td>7</td>
<td>NULL</td>
<td>8</td>
</tr>
<tr>
<td>22</td>
<td>7</td>
<td>NULL</td>
<td>153</td>
</tr>
<tr>
<td>23</td>
<td>NULL</td>
<td>NULL</td>
<td>153</td>
</tr>
</tbody>
</table>

© 2013 IBM Corporation
GROUP BY CUBE

- An extension to the GROUP BY clause

Produces a result set that contains ROLLUP aggregation plus cross-tabulation rows.

The sequence of the columns is not significant.

GROUP BY CUBE (a, b, c) is equal to

GROUP BY (a, b, c) +
   GROUP BY (a, b) + GROUP BY (a,c) + GROUP BY (b,c) +
   GROUP BY (a) + GROUP BY (b) + GROUP BY (c) + grand-total

Example

```
SELECT WEEK(SALES_DATE) AS WEEK, DAYOFWEEK(SALES_DATE) AS DAY,
       SALES_PERSON, SUM(SALES) AS SOLD
FROM SALES
WHERE SALES_DATE > '1999-12-31'
GROUP BY CUBE
```
### GROUP BY CUBE results

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DAY</th>
<th>SALES_PERSON</th>
<th>SOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>4</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>4</td>
<td>LEE</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>4</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>4</td>
<td>NULL</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>5</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>5</td>
<td>LEE</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>5</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>5</td>
<td>NULL</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>6</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>LEE</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>6</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>6</td>
<td>NULL</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>7</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>7</td>
<td>LEE</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>7</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>16</td>
<td>13</td>
<td>7</td>
<td>NULL</td>
</tr>
<tr>
<td>17</td>
<td>13</td>
<td>7</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>18</td>
<td>13</td>
<td>7</td>
<td>LEE</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>7</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>20</td>
<td>13</td>
<td>7</td>
<td>NULL</td>
</tr>
<tr>
<td>21</td>
<td>53</td>
<td>7</td>
<td>GOUNOT</td>
</tr>
<tr>
<td>22</td>
<td>53</td>
<td>7</td>
<td>LEE</td>
</tr>
<tr>
<td>23</td>
<td>53</td>
<td>7</td>
<td>LUCCHESSI</td>
</tr>
<tr>
<td>24</td>
<td>53</td>
<td>7</td>
<td>NULL</td>
</tr>
</tbody>
</table>

© 2013 IBM Corporation
### GROUP BY CUBE results contd.

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DAY</th>
<th>SALES</th>
<th>PERSON</th>
<th>WEEK CUBE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>53</td>
<td>NULL</td>
<td>GOUNOT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>53</td>
<td>NULL</td>
<td>LEE</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>53</td>
<td>NULL</td>
<td>LUCCHESSI</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>53</td>
<td>NULL</td>
<td>NULL</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>NULL</td>
<td>4</td>
<td>GOUNOT</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>NULL</td>
<td>4</td>
<td>LEE</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>NULL</td>
<td>4</td>
<td>LUCCHESSI</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>NULL</td>
<td>4</td>
<td>NULL</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>NULL</td>
<td>5</td>
<td>GOUNOT</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>NULL</td>
<td>5</td>
<td>LEE</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>NULL</td>
<td>5</td>
<td>LUCCHESSI</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>NULL</td>
<td>5</td>
<td>NULL</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>NULL</td>
<td>6</td>
<td>GOUNOT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>NULL</td>
<td>6</td>
<td>LEE</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>NULL</td>
<td>6</td>
<td>LUCCHESSI</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>NULL</td>
<td>6</td>
<td>NULL</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>NULL</td>
<td>7</td>
<td>GOUNOT</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>NULL</td>
<td>7</td>
<td>LEE</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>NULL</td>
<td>7</td>
<td>LUCCHESSI</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>NULL</td>
<td>7</td>
<td>NULL</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>NULL</td>
<td>NULL</td>
<td>GOUNOT</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>NULL</td>
<td>NULL</td>
<td>LEE</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>NULL</td>
<td>NULL</td>
<td>LUCCHESSI</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>153</td>
<td></td>
</tr>
</tbody>
</table>

Total 48 records shown
Best Practice Shops

- Keep SQL skills up to date
- Have a culture geared toward proper use of SQL
- Have SQL code review policies and procedures prior to production
- Monitor SQL performance on a regular basis
- Move the access path initial review into the developer's hands using tools – topic for another day
Recommended Reading

Implementing IBM CICS JSON Web Services for Mobile Applications
IBM Redbooks Solution Guide

This IBM Redbooks Solution Guide describes the existing and new aspects of IBM CICS Transaction Server that allow you to move your CICS applications, and business, into the mobile space. This Solution Guide gives you an overview of how to connect mobile devices to IBM CICS Transaction Server to use existing enterprise services that are hosted on CICS, or to develop services that support new lines of business (LOB). We summarize the steps to develop, configure, and deploy an application that connects either directly to CICS Transaction Server or to CICS through IBM z/OS Server.

Figure 1 gives an overview of this solution.
DB2 11 is Great!
This is Why You Should Care!

Sheryl M. Larsen
IBM WW DB2 for z/OS Evangelist
smlarsen@us.ibm.com