

iSeries. mySeries.

DB2 UDB : Advanced Functionality Sampler

*Alison Butterill
IBM Canada*



common
Belgium

COMMON Belgium Congress
17 November 2004

© Copyright IBM Corporation, 2004. All Rights Reserved.
This publication may refer to products that are not currently
available in your country. IBM makes no commitment to make
available any products referred to herein.

iSeries. mySeries.

Acknowledgment and Disclaimer

Acknowledgment:

Many people contributed to this presentation. In particular (but in no particular order) thanks goes to:

Skip Marchesani - Lexel Custom Systems, New Jersey
Kent Milligan, IBM Corporation, Rochester, Minnesota

Disclaimer:

The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

Reproduction:

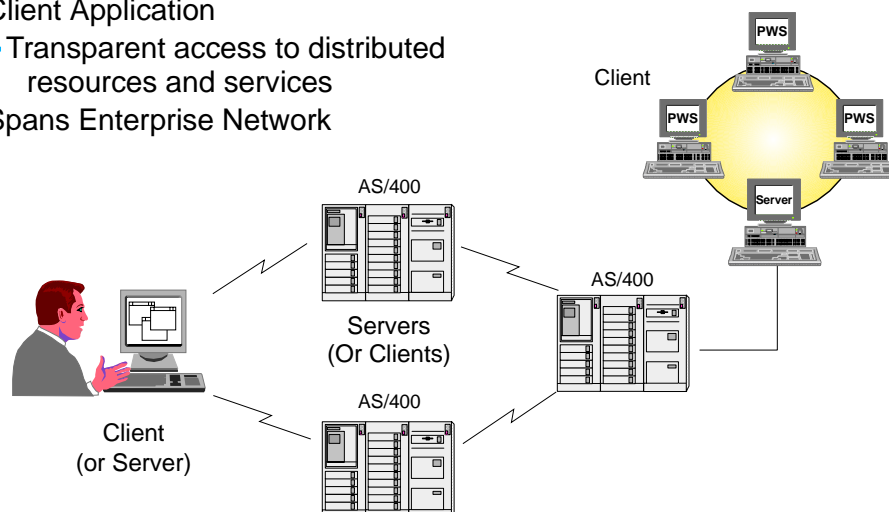
The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

Client/Server Applications

- Client Application
 - Transparent access to distributed resources and services
- Spans Enterprise Network



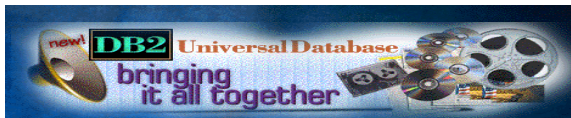
- **Optimize location of applications, data and other system resources**

Database Server Capability

- Relational Database Function
- Stability
- Ease of Use
- Two Interfaces - Native and SQL
- Ease of Management
- Integration with iSeries Architecture
 - Journaling and commitment control
 - System managed access path protection
 - Database integrity
 - Security
- Most widely used multi-user RDB in the world

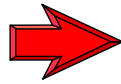
DB2 UDB for iSeries: Best Relational Database in the Marketplace

DB2 Universal Database for OS/400



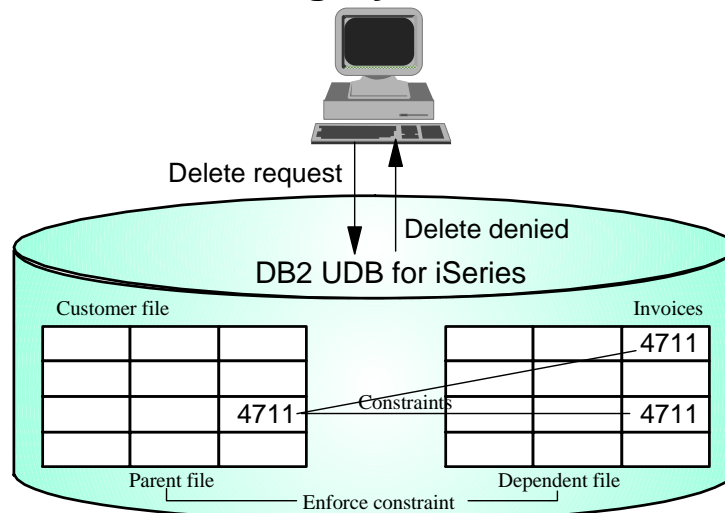
- ▶ **Universal Access** → ODBC, DRDA, SQL, Net.Data, Stored Procs
SQLJ, DDM TCP/IP support
- ▶ **Universal Application** → 30,000 OLTP Apps, 5,000 C/S Apps
Web Based Warehousing Tools
Full JAVA Application/Server Support
- ▶ **Universal Extensibility** → LOB, UDF, UDT, DataLinks
Object Relational Support
XML Extenders
- ▶ **Universal Scalability** → 10,000 Fold Growth, 24 way SMP Support
Clustering Support (32-24 way SMP Nodes)
EVI, Group By Performance, SubQuery
- ▶ **Universal Reliability** → 99.9+% Reliability
Continuous Availability Offerings
- ▶ **Universal Management** → Performance Mgt Tools, Visual Warehouse
Integrated w/Systems Mgt Tools
Operations Navigator, DBA Essentials
Logical Partitioning within a System

Agenda



- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

What is Referential Integrity?

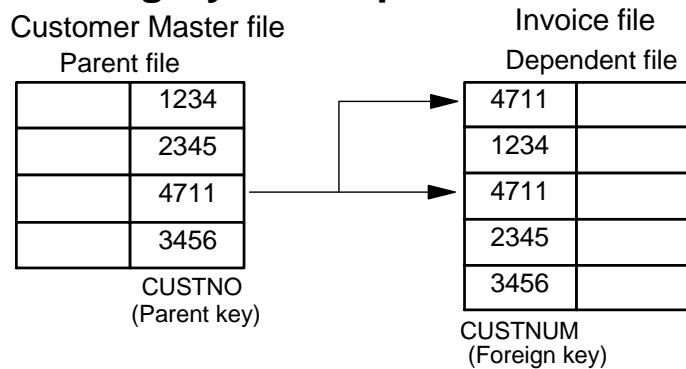


- A capability provided by the database management system to ensure
 - logical consistency of data values between files
 - validity of data relationships
 - robust enforcement of integrity constraints

RI Concepts

- Parent and Dependent files
- Unique key and Primary key constraint
- Parent key and Foreign key
- Referential constraint
- Referential integrity
- Referential constraint rules

Referential Integrity - Example



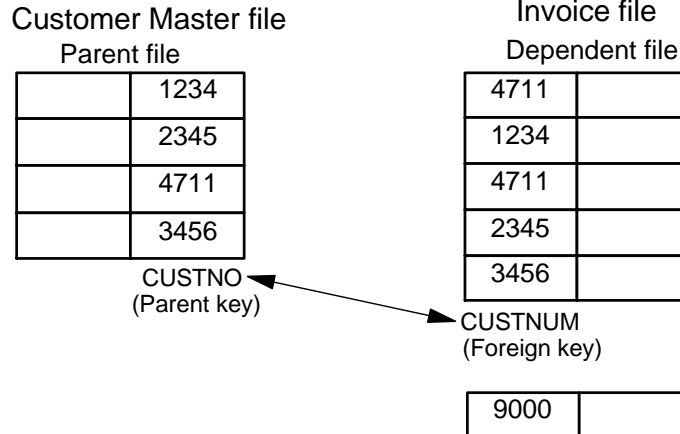
CL --> ADDPFCST FILE(master)
 TYPE(*PRIKEY)
 CST(name)
 KEY(custno)

SQL --> ALTER TABLE master
 ADD CONSTRAINT name
 PRIMARY KEY (custno)

ADDPFCST FILE(invoice)
 TYPE(*REFCST)
 CST(name2)
 KEY(custnum)
 PRNFILE(master) PRNKEY(custno)
 UPDRULE(*RESTRICT)
 DLTRULE(*CASCADE)

ALTER TABLE invoice
 ADD CONSTRAINT name2
 FOREIGN KEY (custnum)
 REFERENCES master (custno)
 ON DELETE CASCADE
 ON UPDATE RESTRICT

Referential Integrity - Example

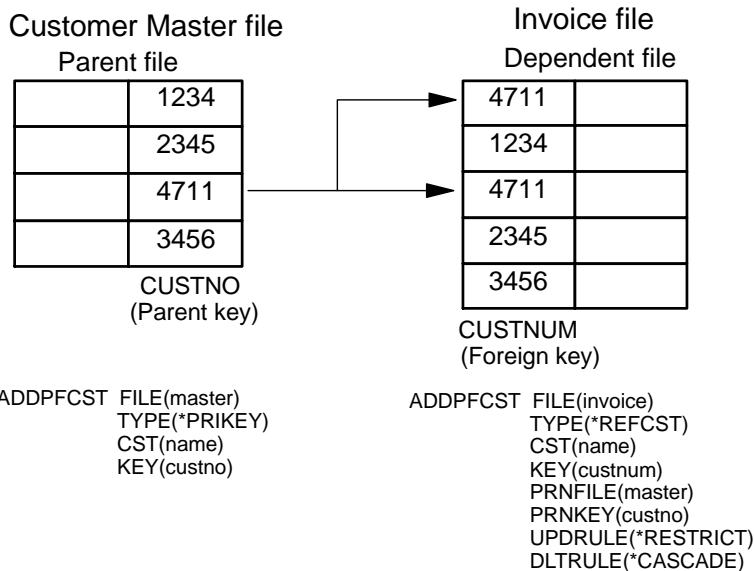


- Try to add this new row to the Invoice File
- What happens?

Referential Constraint Rules

- Insert rule:
 - no explicit rules, but insert operations into Dependent files are checked
- Delete rule:
 - RESTRICT
 - CASCADE
 - SET NULL
 - SET DEFAULT
 - NO ACTION
- Update rule:
 - restrict updates to a Parent key
 - RESTRICT
 - NO ACTION

Referential Integrity - Example



- Deleting customer 4711 will delete both related invoices
- Updating 4711 key value is prevented

Referential Integrity - Requirements

- Basic requirement is:
 - Parent key and foreign key must have matching field attributes
- Parent and Dependent files must be
 - single member physical files
 - externally described
- On Parent file you can create
 - one primary key
 - multiple unique and parent keys
- Journal requirements
 - Restrict rule: journal not required
 - Any other rule:
 - Parent and Dependent files must be journaled to the same journal
 - Implicit commitment control is started

General Implementation Considerations

- Referential constraints not allowed for
 - System files
 - Source files
 - Program described files
- Maximum of one member for parent or dependent file
- Referential constraints cannot span ASPs
- Maximum of 300 referential constraint relationships per file (parent or dependent)

Access Paths and Referential Integrity

- DB2 for iSeries needs access paths or indexes for constraint enforcement
- At creation time, the system will:
 - share an existing access path with matching attributes
 - create a new access path, if sharing is not possible
- A Constraint access path is:
 - comparable to a logical file or SQL index
 - part of the file: saved/restored with the object
 - considered by the OS/400 Query Optimizer

Creating Referential Integrity Constraints

- Native interface:
 - Parent and dependent files can be created with SQL or DDS
 - Define constraints through a new Add Physical File Constraint (ADDPFCST) command
- SQL interface
 - Create Parent and Dependent files through CREATE TABLE
 - CREATE TABLE allows constraints definition
 - ALTER TABLE can be used to add constraints to existing tables
- Operations Navigator

Verifying Constraints

- DB2 UDB for iSeries implements "Declarative RI"
- Constraints will be verified:
 - at creation time
 - when the state goes to established/enabled
 - after a file has been restored

For Large Files, this may take some time!

- Constraints will be enforced
 - whenever an I/O operation occurs
 - delete from Parent file
 - insert into Dependent file
 - update on either file

Constraint States

	Enabled	Disabled
Defined	Will remain enabled when moved to established	Will remain disabled when moved to established
Established	Referential integrity constraints enforced	Referential integrity constraints not enforced

- Defined state:
 - constraint definition exists but is not enforced
 - a constraint goes to defined state:
 - it is defined over files with no member
- Established state:
 - constraint definitions exist and both files have a member
- Enabled
 - integrity checking will occur
- Disabled
 - integrity checking will not happen

Monitoring Exceptions in Applications

- Applications should handle new error messages and status codes
- System messages:
 - Notify: CPF502D, CPF502E, CPF523B
 - Escape: CPF523B, CPF523C
- OPM programs: Mapped to existing error codes
- ILE programs
 - ILE RPG
 - use indicators at first
 - new status code: 01222 and 01022
 - ILE COBOL
 - file status 9R
 - ILE C
 - mapped to existing error numbers
 - SQL
 - SQLCODEs -530, -531, -532
 - SQLSTATEs 23001, 23503, 23504

What is Check Pending Status?

- Condition where some foreign key values do not match any parent key
- A check pending condition may occur:
 - when adding referential constraints to existing files
 - after abnormal system failures
 - after restoring files at different data levels
 - after applying/removing journal changes
- To get out of a Check Pending condition:
 - disable the constraint
 - remove the check pending condition by
 - inserting the required parent records
 - removing/changing the dependent records
 - enable the constraint
- WRKPFCSST - Work with Physical File Constraints →

Constraint Management Commands

```

Display Report
Width .....:      142
Column .....:       1
Control .....
Line   ....+....1....+....2....+....3....+....4....+ .....
          ORDER_NUMBER    CUSTOMER_NUMBER    ORDER_DATE
000001  02020             12312             02/03/1994
000002  02021             12312             04/13/1994
000003  02022             12312             04/25/1994
*****  * * * * * * * * * * END OF DATA * * * * * * * * * *
    
```

- Display Check Pending Constraint - DSPCPCST
 - displays dependent records with no matching parent key
 - constraint must be in disabled state
- Change Physical File Constraint - CHGPFCST
 - sets to enabled/disabled state

iSeries Navigator - RI Constraint Management (V5R3)

Edit Check Pending Constraint ISNAVCPTST.CHK1CST - As400c(Lp01ut5)

The following rows in table ISNAVCPTST.CHK1 do not meet the check condition. You may update or delete the following rows:

DEPARTMENT_ID	MANAGER_ID	DESCRIPTION
99	-1	<NULL>
99	-2	e
99	-2	b
99	-2	a
99	-2	a
99	-2	a
99	-2	a
99	-2	a
99	-2	a
99	-2	a
99	-2	<NULL>
99	-2	a
99	-3	a
99	-3	a
99	-3	a
99	-3	a
99	-3	a
99	-3	a
99	-3	a

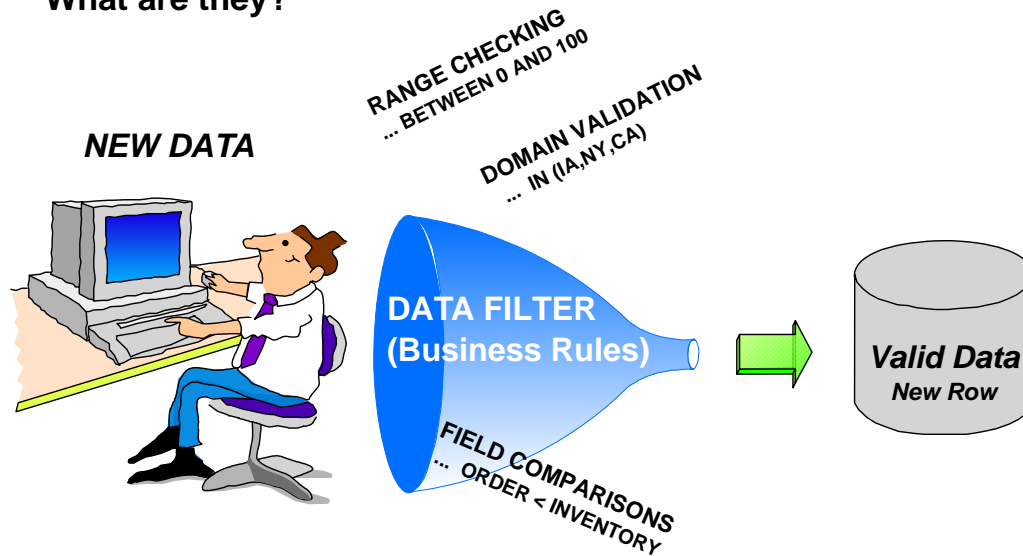
Rows found: 22 Mode: Changes under commitment control

Commit Rollback

Agenda

- Introduction
- Referential Integrity
- ➔ ● Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

Check Constraints What are they?



Check Constraints

- Allow data validation and business rule processing into DB2 UDB for iSeries
- Check constraint is associated with a table
 - values assigned to a column
- Contains a predicate known as a **check condition** enforced against every row in the table
- If the check condition detects invalid values, then associated insert/update is NOT allowed

Defining Check Constraints

Native CL Interface:

```
ADDPFCST FILE(Employee) TYPE(*CHKCST)
CST(SalaryChk)
CHKCST(Salary<40000 AND Bonus<=Salary)
```

SQL Interface (Create Table or Alter Table):

```
ALTER TABLE Employee
ADD CONSTRAINT SalaryChk
CHECK(Salary<40000 AND Bonus<=Salary)
```

DB2 UDB for iSeries locks the table exclusively when adding and verifying a Check Constraint - verification time will NOT be measured in seconds for very large tables

- 4 million row table took 5 minutes to verify

Design Considerations

- Check Condition can contain any expressions or functions allowed on a SQL WHERE clause with the following exceptions:
 - Cannot reference columns in a different table
 - Cannot reference other rows in the table, meaning the following column functions are not allowed:
SUM, AVERAGE, MIN, MAX, & COUNT
 - Subqueries are not allowed
- Check Condition clause can reference more than one column in the source table

Design Considerations

- Only single member files supported
- A table has a limit of 300 constraints per file (includes RI, Check, Unique and Primary Key constraints)
- Constraint name has to be unique across all constraint types that exist in the table's library
- Database does NOT prevent conflicting constraints from being defined
 - CHECK Constraint#1
ORDER_STATUS = 'Open'
 - CHECK Constraint#2
ORDER_STATUS <> 'Open'

Application Integration

- Check Constraints will be enforced on:
 - Insert into table with constraints
 - Update on table with constraints
 - Delete operation on a Parent Table whose RI Constraint rule is Set Default or Set Null
- New System Message: **CPF502F**
 - **Message:** Check constraint violation on member TEST1.
- ILE Programs
 - ILE RPG: Use indicators, maps to status code 1022
 - ILE COBOL: File Status 9W
 - ILE C: Mapped to existing error number
- OPM Programs: Mapped to existing error codes
- SQL
 - SQLCODE -545
 - SQLSTATE 23513

Tips and Techniques

- Start by examining existing application code - isolate code doing "Check Constraint" activity - identify for eventual replacement
 - Migrate "checks" into ILE Procedures that can be bound and reused by several application programs
 - Migrate "checks" into Trigger programs
- Start verifying and cleaning up your data now with queries
 - Good place to use Predictive Query Governor (CHGQRYA)
- Multiple Check Constraints versus One Big Check Constraint
 - When more than one Check Constraint, system implicitly 'AND's the result of each Constraint during enforcement
 - Multiple Check Constraints allow better identification of the problem when system detects constraint violations during I/O operations
 - One Big Check Constraint performs slightly better since implicit 'AND' processing is eliminated

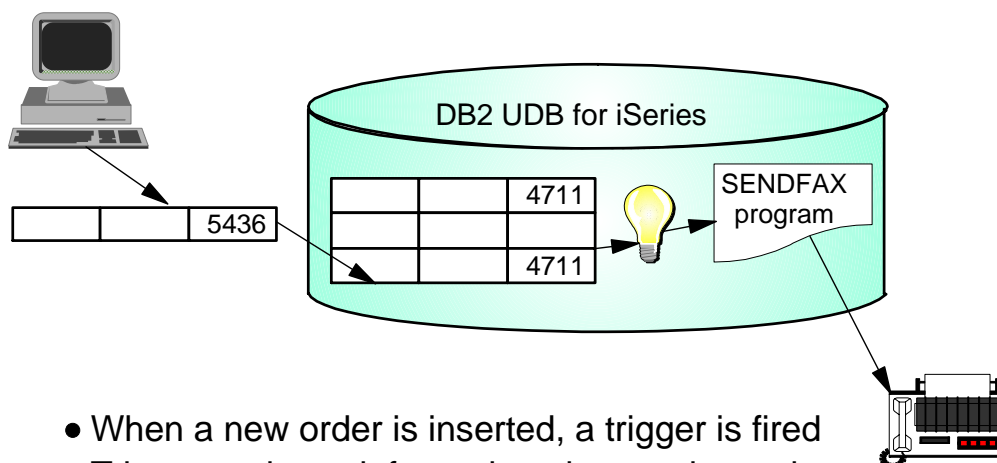
Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

Triggers: Introduction

- Triggers are user-written programs
 - associated with a physical file
 - activated by DB2 UDB for iSeries before or after a database change
 - independent from applications
 - can be developed with any supported compiler
- When do you need triggers?
 - to consistently enforce complex business rules
 - implement special application requirements
 - to monitor critical files
 - validate data
 - in a client-server environment

Triggers: An Example



- When a new order is inserted, a trigger is fired
- Trigger retrieves information about order and customer
- A confirmation fax is automatically sent

OS/400 Support for Triggers

- V4R5 and Prior Releases of OS/400
 - External triggers only (aka Native or System triggers)
 - Two commands
 - ADDPFTRG
 - RMVPFTRG
 - External triggers can be added or removed from a file using the Database function in Operations Navigator
 - A file can have a total of 6 triggers
 - No triggers on catalog files or tables
- V5R1 - Two Types of Triggers
 - External Triggers (see above)
 - First available with DB2/400 in V3R1
 - Originally referred to as Native Triggers
 - SQL Triggers
 - New in V5R1

V5R1 Trigger Enhancements

- SQL Triggers
 - Column level
 - Row level
 - Statement level
- More than 1 trigger per database event
 - maximum 300 per physical file or table
 - triggers for same event, fired in the order created
 - identified or qualified by Trigger name
- CHGPFTRG command
 - disable an active or enabled trigger
 - enable an inactive or disabled trigger
- 'Read only' Trigger
 - use carefully

V5R1 Trigger Enhancements...

- Read Triggers
 - ▶ Read Trigger fires EVERY time a record is read from a file opened for Read only
 - Assumes Read Trigger is defined for the file

This is NOT good news!

- ▶ *READ trigger event
 - Only available with External Triggers
 - NOT available with SQL Triggers

- Use of Read Triggers could have a **SEVERE detrimental effect on your job and/or career!**

System Triggers

Concepts
Definition
Activation
Feedback

Trigger Concepts

- Four main components
 1. base file
 2. trigger event
 - insert
 - delete
 - update
 3. trigger time
 - before
 - after
 4. trigger program
- Triggers have record level scope
- Will be activated whenever the event occurs
- Exception: Update triggers
 - *ALWAYS
 - *CHANGE

Defining Triggers

- Add or remove triggers by using new CL commands
 - ADDPFTRG
 - RMVPFTRG

Add Physical File Trigger (ADDPFTRG)

Type choices, press Enter.

Physical file	FILE	myfile
Library		mylib
Trigger time	TRGTIME	*BEFORE
Trigger event	TRGEVENT	*UPDATE
Program	PGM	mypgm
Library		*LIBL
Replace trigger	RPLTRG	*NO
Trigger update condition	TRGUPDCND	*ALWAYS

- Trigger information will be stored in the file description
- Implications on recompiling, restoring, deleting and renaming trigger programs

Activating Triggers

- Triggers become part of the job that activates them

Opt	Request Level	Program or Procedure	Library	Statement	Instruction
		QCMD	QSYS		0351
		QUICMENU	QSYS		00C1
	1	QUIMNDRV	QSYS		0455
	2	QUIMGFLW	QSYS		0483
	3	QUICMD	QSYS		03E4
		QUOCP	QPDA		0541
		QUOMAIN	QPDA		OFDD
	4	QUOCMD	QSYS		0176
		T4249CINS	ORDENTLIB	136	00D9
		QSROUTE	QSYS		02F0
		QSQINS	QSYS		01C0
		QDBPUT	QSYS		0193
		T4249RADT	ORDENTLIB	.GET	021D

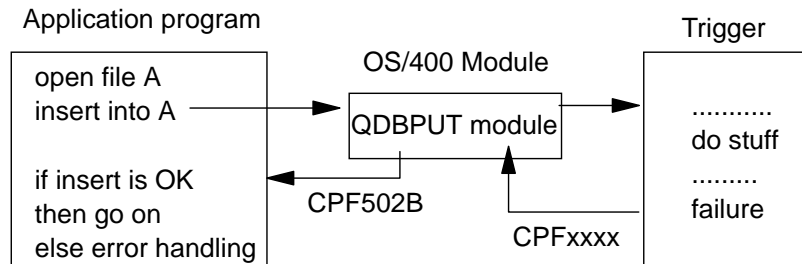
- Triggers and applications will share:
 - the library list
 - the QTEMP library

Trigger Buffer Fields

Seq	Data Type	Len	Field Description
1	Char	10	Physical file name
2	Char	10	Physical file Library
3	Char	10	Physical file member name
4	Char	1	Trigger event
5	Char	1	Trigger time
6	Char	1	Commit lock level
7	Char	3	Reserved
8	Binary	4	CCSID of data
9	Binary	4	Relative record number
10	Char	4	Reserved
11	Binary	4	Original record offset
12	Binary	4	Original record length
13	Binary	4	Original record null byte map offset
14	Binary	4	Original record null byte map length
15	Binary	4	New record offset
16	Binary	4	New record length
17	Binary	4	New record null byte map offset
18	Binary	4	New record null byte map length
19	Char	16	Reserved
20	Char	Var	Original Record
21	Char	Var	Original record null byte map
22	Char	Var	New record
23	Char	Var	New record null byte map

Triggers Feedback

- What happens if a trigger ends abnormally??
 - The unhandled exception will 'percolate'

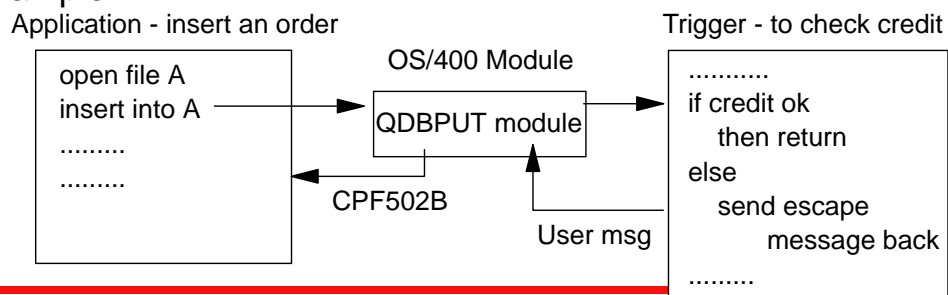


- The originating change will fail

Triggers and Application Programs

- How can a trigger notify a 'logical failure' to the application?
- Triggers cannot pass parameters back
- An escape message must be sent back so I/O will fail
- Use the QMHSNDPM API
- The originating change will fail
- Application will get the CPF502B message
- Escape message is logged in the joblog

Example:



Triggers and Object Management

- When the base file is saved, trigger information is saved as well
- Trigger programs are NOT saved with the file
 - create triggers in the same library as their files
- If trigger program is renamed, moved or deleted, file description information is not updated
 - remove and add the trigger again
- The Copy Library command will update trigger information in the file description
- Security - triggers can access objects that users may not be authorized to use
 - create those triggers with `USRPRF(*OWNER)`

Guidelines for Trigger Programs

Operations that are NOT allowed

- Destructive data changes are not allowed if `ALWREPCHG(*NO)` is specified
 - CANNOT modify the record that fired the trigger, from within the trigger itself
 - 1. application updates existing record
 - 2. UPDATE trigger fires
 - 3. Trigger tries to update same record
 - conflicting operations NOT allowed on the same record
 - 1. application inserts new record
 - 2. INSERT trigger fires
 - 3. Trigger tries to delete new record
 - Contents of trigger buffer would be inconsistent with actual database record
- Commitment control operations
- SQL Connection statements

V5R1 SQL Triggers

- Component Definitions
- Examples

Trigger Components...

- SQL Trigger Components - V5R1
 - Base file or table
 - Trigger name
 - Trigger event
 - Trigger time
 - Trigger granularity
 - Transition variables
 - Transition tables
 - Trigger mode
 - Triggered action

SQL Trigger Components

- Base file or table
 - Physical file or table which the trigger is added to
- Trigger name
 - Provides unique trigger identification within a library
- Trigger event
 - The condition that causes the trigger to fire
 - **Insert** of a new row
 - **Update** of existing
 - ▶ Row
 - ▶ Column (Column level triggers)
 - **Delete** of an existing record
- 1. Trigger time
 - When trigger program is to be run
 - Before or after the trigger event

SQL Trigger Components...

- 5. Trigger Granularity
 - ▶ Column level triggers
 - Extension of UPDATE trigger event
 - Columns listed as part of UPDATE trigger event
 - UPDATE OF column_name_1, column_name_2, ...
 - Only update of a listed column causes trigger to fire
 - If no columns listed, update to any column causes trigger to fire
 - ▶ Row level triggers
 - FOR EACH ROW
 - Triggered action executed for each row satisfying trigger condition
 - If trigger condition never satisfied, triggered action never executed
 - ▶ Statement level triggers
 - FOR EACH STATEMENT
 - Triggered action executed only once for the event causing the trigger to fire regardless of the number of rows processed
 - If trigger condition never satisfied, triggered action executed once at end of statement processing
 - Not valid with Before triggers or Trigger Mode of DB2ROW

SQL Trigger Components...

- 6. Transition Variables
 - ▶ aka Correlation Variables
 - ▶ Provides function similar to before and after images in trigger buffer for external triggers
 - ▶ Qualification of column names for the single row image before and/or after the trigger event has completed
 - OLD ROW - Before image of row
 - NEW ROW - After image of row
 - REFERENCING OLD ROW AS oldrow
REFERENCING NEW ROW AS newrow
 - ...newrow.salary > oldrow.salary + 10000...
 - ▶ Not valid with Statement level triggers

SQL Trigger Components...

- 7. Transition Tables
 - ▶ Provides function similar to before and after images in trigger buffer for external triggers
 - ▶ A single SQL statement can process multiple rows
 - ▶ Temporary tables that contain the image of all affected rows before and/or after the trigger event completes
 - OLD TABLE - Before image of all affected rows
 - NEW TABLE - After image of all affected rows
 - REFERENCING OLD TABLE AS oldtbl
 - ...(SELECT COUNT(*) FROM oldtbl)...
 - ▶ Not valid with Before triggers or Trigger Mode of DB2ROW

SQL Trigger Components...

- 8. Trigger Mode
 - ▶ MODE DB2ROW
 - Trigger fires after each row operation
 - Only valid with Row level triggers
 - Exclusive function of DB2 UDB for iSeries
 - Not available in other DB2 UDB implementations
 - ▶ MODE DB2SQL
 - Trigger fires after all row operations are complete
 - If specified on a row level trigger, triggered action executed N times after all row operations completed
 - N = number of rows processed
 - Not as efficient as DB2ROW since each row is processed twice
 - Only valid with After triggers

SQL Trigger Components...

- 9. Triggered Action
 - ▶ Analogous to trigger program in external triggers
 - ▶ Three parts
 - SET OPTION
 - Specifies the options that will be used to create the trigger
 - WHEN
 - Search condition or execution criteria for Trigger Body
 - Specifies when the SQL statements in Trigger Body will be executed
 - SQL Trigger Body
 - Single SQL statement
 - Multiple SQL statements delineated with BEGIN and END

SQL Trigger Examples

Row Level Trigger with Simple Trigger Body


```
CREATE TRIGGER audit_spending
  AFTER UPDATE ON expenses
  REFERENCING NEW ROW AS nw
  FOR EACH ROW MODE DB2ROW
  WHEN (nw.total_amount > 10000)
  INSERT INTO travel_audit
    VALUES(nw.empno, nw.deptno, nw.total_amount,
           nw.end_date);
```

SQL Trigger Examples ...

Row Level Trigger with Complex Trigger Body

```
CREATE TRIGGER big_spenders
  AFTER INSERT ON expenses
  REFERENCING NEW ROW AS n
  FOR EACH ROW
  MODE DB2ROW
  WHEN (n.totalamount > 10000)
  BEGIN
    DECLARE emplname CHAR(30);
    SET emplname = (SELECT lname FROM employee
                   WHERE empid = n.empno);
    INSERT INTO travel_audit
      VALUES(n.empno, emplname, n.deptno, n.totalamount, n.enddate);
  END
```

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
-  • Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

Column Level Security

What is it?

NAME	SALARY	DEPARTMENT
JOAN	20,000	SALES
FRED	18,000	SALES
LINDA	15,000	ACCOUNTING

Manager - needs Update Access



Employee - NO Update Access



Column Level Security

What is it?

- Provides easier, more flexible way to control access to columns in database
- Update & References authority can now be controlled at the column level instead of being limited to the table level (Read authority control not yet supported)
- System security functions used to restrict users from updating certain columns in a table
- Integrated security controls much more of a factor now that iSeries data is no longer protected behind green-screen applications

Column Level Security

SQL statements required:

```
CREATE TABLE TestTable  
  (Name CHAR(30), Salary DEC(8,2),  
   Department CHAR (10))
```

```
GRANT SELECT,UPDATE(Name, Department) ON TABLE  
  Payroll TO Clerk
```

```
GRANT SELECT, UPDATE(Name, Department, Salary) ON  
  TABLE Payroll TO Manager
```

DSPOBJAUT Test Table

- To show Column Level Authorities

Operations Navigator

No support currently for CL interface:

- send cards and letters to IBM Rochester

Current Methods

- Application Program Code
 - Program used menus to prevent direct access of database objects
- Logical Files & SQL Views
 - Created to "hide" restricted columns from the end user
- Stored Procedures for ODBC access
 - Stored Procedure program used adopted authority to control authority below the table level

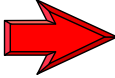
Application Integration

- Column Level Security enforcement primarily during Update operation on the file
 - Verify user is authorized to update the columns changed in the Update operation
(update only rejected when the column(s) being restricted is actually updated to a new value)
- **No new enforcement during open of the file**
 - Implementation gives the user some object authority when column level authority is granted so normal object level authority processing occurs

Application Integration

- New System Message: **CPD5035**
 - **Message:** Not authorized to field COL1 of file TEST1
- SQL Code
 - Violations map to existing SQLCODE (-551) SQLSTATE (42501) for database authority violations
- ILE Programs & OPM Programs
 - Violations map to existing error code for database authority violations
- New system messages when system is unable to enforce column level authorities due to unexpected problems with user profiles, database file object, or DB2 UDB for AS/400
 - CPF328F - Unsuccessful authority checking
 - CPD5030 - User Profile Destroyed
 - CPD5032 - User Profile Damage

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
-  ● Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary

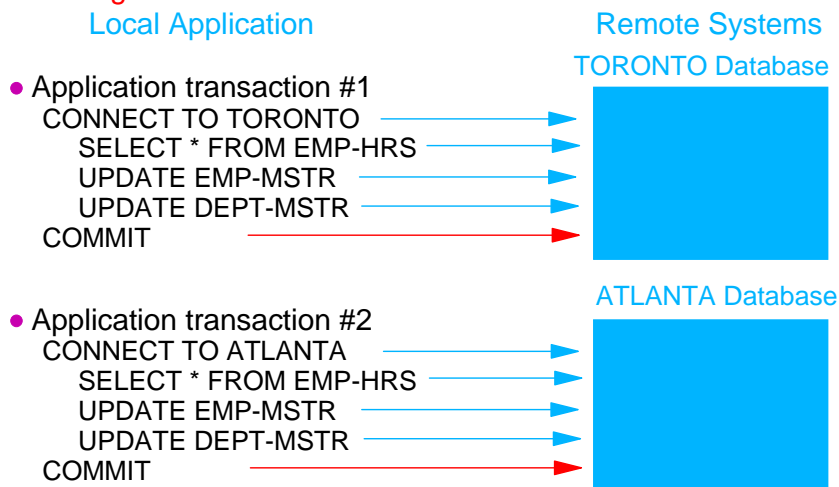
Two Phase Commit: Introduction

- Update data from multiple systems
 - within a single unit of work
 - all DB changes either committed or rolled back
 - connections maintained to multiple systems
- Supported in DRDA, DDM, CICS/400
- APPC or TCP/IP

Two-Phase COMMIT...

Applications using multiple relational databases

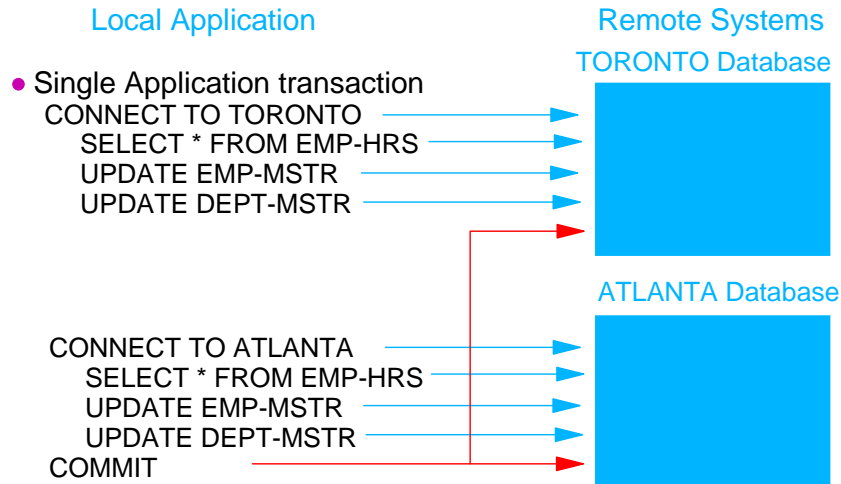
Single Phase COMMIT - Remote Unit of Work



Two-Phase COMMIT...

Applications using multiple relational databases

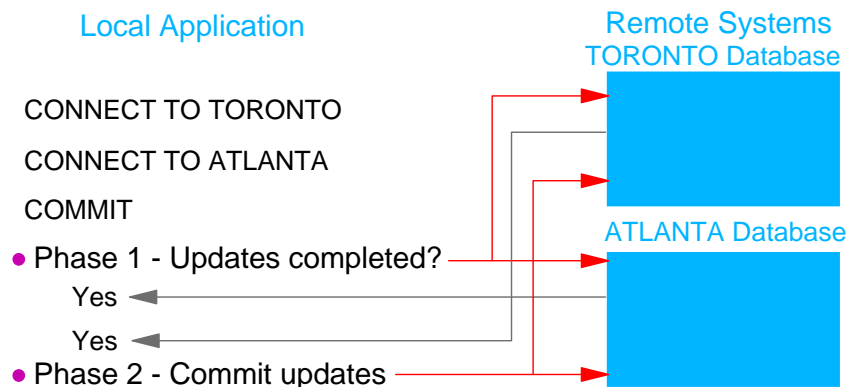
Two Phase Commit - Distributed Unit of Work; Application directed



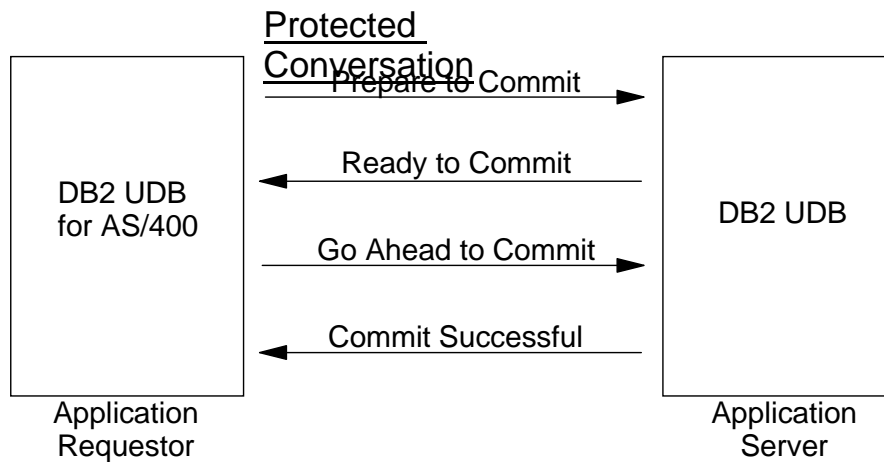
Two-Phase Commit - Concepts and Terminology

- The two phases of Two-Phase Commit
 - Phase 1: Are databases ready to commit? Have all database updates been completed?
 - Phase 2: Commit database updates for application transaction

Two Phase Commit - Distributed Unit of Work; Application Directed



Technical Description



- Two phase commit uses protected conversations
- Sync Point Manager in DB2 UDB for iSeries
 - Coordinates, carries out commit/rollback operations

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary



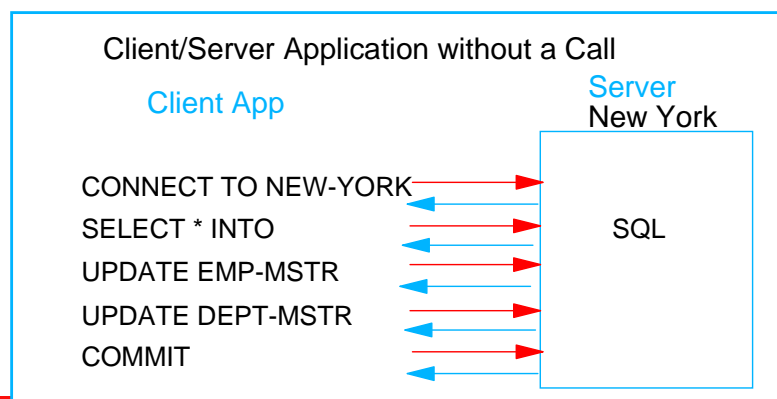
Stored Procedures... (SP)

What is a Stored Procedure?

- A program stored on an iSeries
 - Languages supported
 - SQL
 - Command Language (CL)
 - RPG
 - COBOL
 - C and C++
 - PL/1
 - JAVA
 - May contain embedded SQL statements
- Can be local or remote
- Supported via new CALL statement in SQL

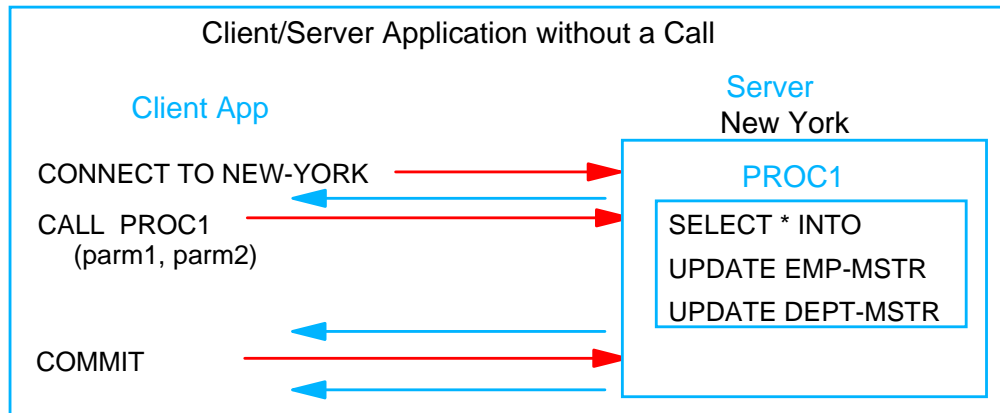
Stored Procedures...

- Improved Client/Server (DRDA) Performance
 - Without CALL of stored procedure every executable SQL statement using DRDA causes a flow from client application to server
 - Each SQL statement is passed to server to perform SQL function to perform SQL function



Stored Procedures...

- Improved Client/Server (DRDA) Performance
 - CALL of stored procedure results in a significant reduction in conversation between client and server
 - SQL statements are embedded in (compiled) program on server resulting in more efficient performance of SQL



Stored Procedures

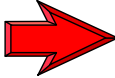
Using SQL for Stored Procedures

- Stored Procedures initially supported in V3R1
 - No SQL procedure language
 - Embedded SQL in a HLL programs like C, COBOL, RPG allowed.
- V4R2 - Stored Procedure can be written purely in SQL
 - First DB2 Family member to deliver this support
- SQL is a programming language

```

CREATE PROCEDURE PROC1 (IN Emp# CHAR(4),IN NwLvl INT)
LANGUAGE SQL Proc1_Src:
BEGIN
  DECLARE CurLvl INT;
  SELECT edlevel FROM empTbl INTO CurLvl
  WHERE empno=Emp#;
  IF NwLvl > CurLvl THEN
    UPDATE empTbl SET edlevel=NwLvl,
    salary=salary + (salary*0.05) WHERE empno=Emp#;
  END IF;
END
  
```

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
-  • SQL Enhancements
- Miscellaneous Enhancements
- Summary

Basic Programming Constructs

Program Control Operations within SQL

- CASE
- IF
- FOR
- LOOP
- REPEAT
- Declaring Variables
- Error Handling

```
IF rating=1
  THEN SET price = price * 0.95;
  ELSEIF rating = 2
  THEN SET price = price * 0.90;
  ELSE SET price = price * 0.80;
END IF;
```

```
FOR loopvar AS
  loopcursor CURSOR FOR
  SELECT firstname, middinit, lastname FROM emptbl
DO
  SET fullname=lastname||', ' || firstname||' ' || middinit;
  INSERT INTO namestbl VALUES( fullname );
END FOR;
```

New SQL Statements

Conditional Logic, Declaration of Variables, Etc

- BEGIN and END
- DECLARE (local variables)
- SET (local variables)
- Comments
- CASE (two forms), END CASE
- IF, THEN, ELSE, END IF
- FOR, END FOR, LOOP, END LOOP
- LEAVE (loop or block)
- REPEAT, END REPEAT
- WHILE, END WHILE
- GET DIAGNOSTICS (SQLCA-like information)
- CALL - SQL procedure, External Procedures like HLL
- Normal DDL and DML

FETCH FIRST n ROWS ONLY Clause

- FETCH FIRST n ROWS only clause added for SELECT statements
 - Enables Top N queries
 - Allows limiting the size of the result set - particularly useful for networked ODBC/JDBC clients
- After the Nth row has been fetched, end-of-file (SQLCODE=100) is returned to the user/application
- To guarantee which FIRST n ROWS are returned - an ORDER BY clause must be specified.
 - If ORDER BY is not specified the FIRST n ROWS will be unpredictable

FETCH FIRST n ROWS ONLY Clause

Examples

Query to return top 20 selling products:

```
SELECT product_item, COUNT(*)  
FROM orders  
GROUP BY product_item  
ORDER BY 2 DESC  
FETCH FIRST 20 ROWS ONLY
```

Query to return the top selling product:

```
SELECT product_item, COUNT(*)  
FROM orders  
GROUP BY product_item  
ORDER BY 2 DESC  
FETCH FIRST ROW ONLY
```

LIKE Predicate enhancements

- Several enhancements have been made to improve the functionality and performance of the iSeries LIKE predicate
 - Expressions on LIKE and Escape characters
 - LIKE predicate no longer causes non-reusable ODP
 - Optimizer can now choose an index for LIKE predicate processing for Double-byte patterns (GRAPHIC/UCS-2)
- Expression Example (find all employees living on same street as John Doe)

```
SELECT b.name  
FROM employee a, employee b, street_tbl c  
WHERE a.name = 'John Doe' AND b.name <> 'John Doe' AND  
a.address LIKE '%||c.street||%' AND  
b.address LIKE '%||c.street||%'
```

JOIN Enhancements

- V5R1 adds new join types for DB2 UDB for iSeries for more portable applications
 - RIGHT OUTER JOIN
 - RIGHT JOIN
- And removal of minor LEFT OUTER JOIN and INNER JOIN restrictions
 - ▶ Allow ORs in the ON clause
 - ▶ Allow the IS NULL, LIKE and BETWEEN predicate in the ON clause

- Examples:

SELECT t1.col1, t2.col2 FROM t1 RIGHT OUTER JOIN t2 ON t1.c1 = t2.c1

SELECT * FROM t1 LEFT JOIN t2 ON t1.c1 > t2.c2 OR t1.c2 > t2.c2

SELECT * FROM t1 LEFT JOIN t2 ON t1.c1 LIKE 'A%'

JOIN Enhancements

Employee Table - EMP

NBR	NAM	CLS	SEX	DPT	SAL
20	Heikki	2	M	901	600
10	Ed	5	M	911	700
50	Marcela	3	F	911	750
40	Mike	4	M	977	650
30	John	5	M	977	320
60	Frank	2	M	990	650

Department Table - DEP

DPT	DNM
901	Accounts
977	Manufact
911	Sales
990	Spares

Left Outer Join	All from left
Left Inner Join (Inner Join)	Only records from left with matching record in right
Right Outer Join	All from right
Right Inner Join	Only records from right with matching record in left

CREATE TABLE LIKE

- CREATE TABLE LIKE clause allows reuse of column definitions from other table or tables
 - Only column definitions are used (no propagation of keys, constraints, triggers, etc)
 - Offers functionality similar to field reference files and CRTDUPOBJ (without data)
- Examples:

```
CREATE TABLE mycustomer LIKE customer_table
```

```
CREATE TABLE table2 (LIKE table1, colx INTEGER)
```
- CREATE TABLE AS extends CREATE TABLE LIKE support

```
CREATE TABLE customer AS
(SELECT id cust_id, lname cust_lastname, fname
cust_firstname, city cust_city FROM ref_file)
WITH NO DATA
```

SQL UNION

- UNIONS can be included in expressions and derived tables

```
CREATE VIEW total_sales AS
SELECT COUNT(*), SUM(total_sale) FROM Sales1999
WHERE product_id ='XYZ'
```

```
UNION
SELECT COUNT(*), SUM(total_sale) FROM Sales2000
WHERE product_id ='XYZ'
```

```
UNION
SELECT COUNT(*), SUM(total_sale) FROM Sales2001
WHERE product_id ='XYZ'
```

New functions for Column Encryption/Decryption (V5R3)

- **Encrypt & Decrypt SQL scalar functions**
 - Requires the IBM Cryptographic Access Provider 128-bit product
 - **Column Requirements:**
 - **Data Type Requirements:** BINARY/VARBINARY, CHAR/VARCHAR FOR BIT DATA, BLOB, and DDS CHAR/VARCHAR with CCSID(65535)
 - **Length Requirements:**
 - Extra 8 bytes & total length must be rounded to 8-byte boundary
(replace 8 with 16, if BLOB or double-byte CCSID)
 - 32-byte hint can optionally be stored with encrypted value
 - Example: 6-byte employee id with no hint needs to be stored in a VARBINARY(16)

```
CREATE TABLE emp(
  id VARCHAR(16) FOR BIT DATA,
  name VARCHAR(50))
```

```
SET ENCRYPTION PASSWORD = 'protect'
```

```
INSERT INTO emp VALUES(ENCRYPT('112233'), 'BOB SANDERS' )
```

```
SELECT DECRYPT_CHAR(id), name FROM emp
```

Sequence Object (V5R3)

- Another DB2 construct that supports the automatic generation of column values
 - Viewed as a superset of V5R2 identity columns
 - Generated values easily shared across tables
 - Can create constant sequence to be used as Global DB2 variables
- Can be changed (altered) with the ALTER SEQUENCE statement

- **Example:**

```
CREATE SEQUENCE order_seq
  START WITH 1 INCREMENT BY 1 NO MAX VALUE
```

```
INSERT INTO orders(ordnum,custnum)
  VALUES (NEXT VALUE FOR order_seq, 123)
```

```
VALUES NEXT VALUE FOR order_seq INTO :hostvar
```

```
UPDATE orders SET ordnum = :hostvar
  WHERE custnum = 123
```

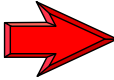
Other SQL Enhancements (V5R3)

- SELECT and INSERT statements
 - blocked INSERT
INSERT INTO table1 VALUES (11,'TESTING'), (2,'ADMINSTRATION')
 - EXCEPT and INTERSECT
(SELECT cusnum FROM orders2003) EXCEPT DISTINCT
(SELECT cusnum FROM orders2004)
(return all rows that are in t1, but not t2)
- Enhancements to Stored Procedures
 - scrollable Stored Procedure result sets
 - result set consumer control
- Richer SQL function set
 - REPLACE, EXTRACT, INSERT, REPEAT, DAYNAME, MONTHNAME, RIGHT, TIMESTAMP_ISO
SELECT REPLACE('ABCXYZ','ABC','123') FROM t1
(returns '123XYZ')

Miscellaneous SQL Enhancements

- New scalar functions:
TIMESTAMPDIFF, PI, SPACE, GRAPHIC,
MIDNIGHT_SECONDS, JULIAN_DAY, DAYOFWEEK_ISO, and
WEEK_ISO
- SQL-based interface for creating debuggable versions of SQL
procedures, functions, and triggers
SET OPTION DBGVIEW = *STMT
- Greater than 80 character embedded SQL statements for C &
C++ precompilers
- Packaging changes:
 - RUNSQLSTM now part of OS/400
 - ILE C compiler not required for SQL Procedures, Functions, &
Triggers
- System-supplied stored procedure for creating updated sample
database
 - CREATE_SQL_SAMPLE procedure created at install time
 - Creating sample database:
CALL QSYS/CREATE_SQL_SAMPLE(schema-name)

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
-  • Miscellaneous Enhancements
- Summary

Predictive Query Governor

- Query Governor added in V3R1
- Limits amount of time for query execution
 - Message issued when time exceeded
 - CPA4259: Estimated query processing time &1 exceeds limit &2
- Decision is made prior to query starting
 - lower impact on system resources
- Change Query Attributes (CHGQRYA) command or via Operations Navigator
 - done on job level
- Query optimizer estimates query processing time
 - compares with time limit from CHGQRYA command
 - makes processing decision

Predictive Query Governor

- Tip for estimating time limit:
 - CHGQRYA - set time limit to 0
 - STRDBG
 - run Query
 - Query will not execute
 - Low level messages will record optimizer estimate of time required to process
 - access plan creation
 - Then set time limit to reasonable value

Changing Fields in a Physical File

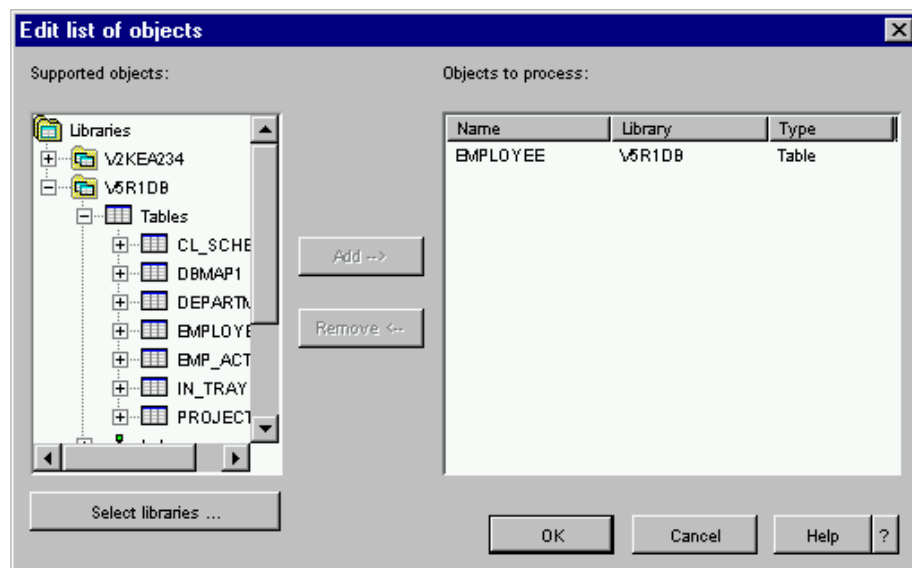
Available since V3R6

- Allows alterations or changes to be made to the definition of a physical file at the data field level without requiring the user to delete and then recreate the physical file or any associated logical files.
- Supported thru native CL and SQL interfaces
 - New parameters on CHGPF
 - New clauses for ALTER TABLE
 - Functional differences between CL and SQL
- Generically known as Alter Table

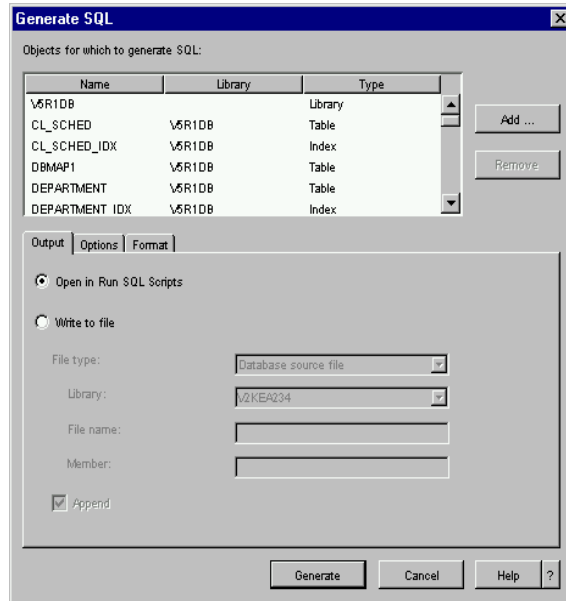
Generate SQL - Operations Navigator

- Reverse engineering of DDL for DB2 objects
 - Useful in converting object definitions from DDS to SQL
 - ▶ Not all DDS features can be converted, tool will convert as much as possible and generate warnings for unconvertable options (eg, EDTCDE)
 - Useful in generating SQL script for creating databases
- Object types supported:
 - Tables and physical files (including triggers and constraints)
 - Aliases
 - Functions & Procedures
 - Indexes, Views, & Logical files
 - Schemas (collections) and libraries
 - Distinct Types
- Can generate SQL for one or multiple objects
- Resulting script can be edited
- API interface - QSQGNDDL

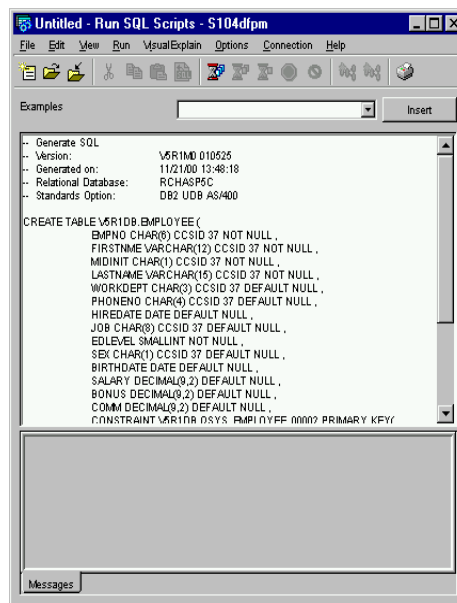
Generate SQL for Multiple Objects



Generate SQL for Multiple Objects



Generated SQL Output



Usability: Visual Explain Index Advisor

Statistics and Index Advisor - 1400ws(1400ws)

Statistics Advisor | **Index Advisor**

The following indexes are being recommended for creation:

Create	Table Name	Library	Index Type	Columns
<input checked="" type="checkbox"/>	LINEITEM	TPCH	Binary Radix	L_PARTKEY ASCEND L_DISCOUNT ASCEND

New Index on Table

Index : Library:

Click to select which columns of the table make up the key. The key position will appear in the left column. To deselect, click it again.

	D...	Column Name	Type	Len...	Description
1	A...	L_PARTKEY	INTEGER		
		L_SUPPKEY	INTEGER		
		L_LINENUMBER	INTEGER		
		L_QUANTITY	DECIMAL	12,2	
		L_EXTENDEDPRICE	DECIMAL	12,2	
2	A...	L_DISCOUNT	DECIMAL	12,2	
		L_TAX	DECIMAL	12,2	

Index type:

Not unique

Encoded vector (not unique)

Unique

Unique where not null

Number of distinct values:

© 2004 IBM Corporation

iSeries. mySeries.

Usability: SQL Assist

SQL Assist - As25(Rchase5c)

Outline

- SQL statement properties
 - SELECT statement
 - FROM (Source tables)
 - SELECT (Result columns)
 - WHERE (Row filter)
 - GROUP BY (Row groups)
 - HAVING (Group filter)
 - ORDER BY (Sort criteria)

Details

Statement Type

SELECT Queries the data in one or more tables

INSERT Inserts new rows in a table

UPDATE Updates existing rows in a table

DELETE Removes rows from a table

Connection

Database alias: /As25

User ID: COOK

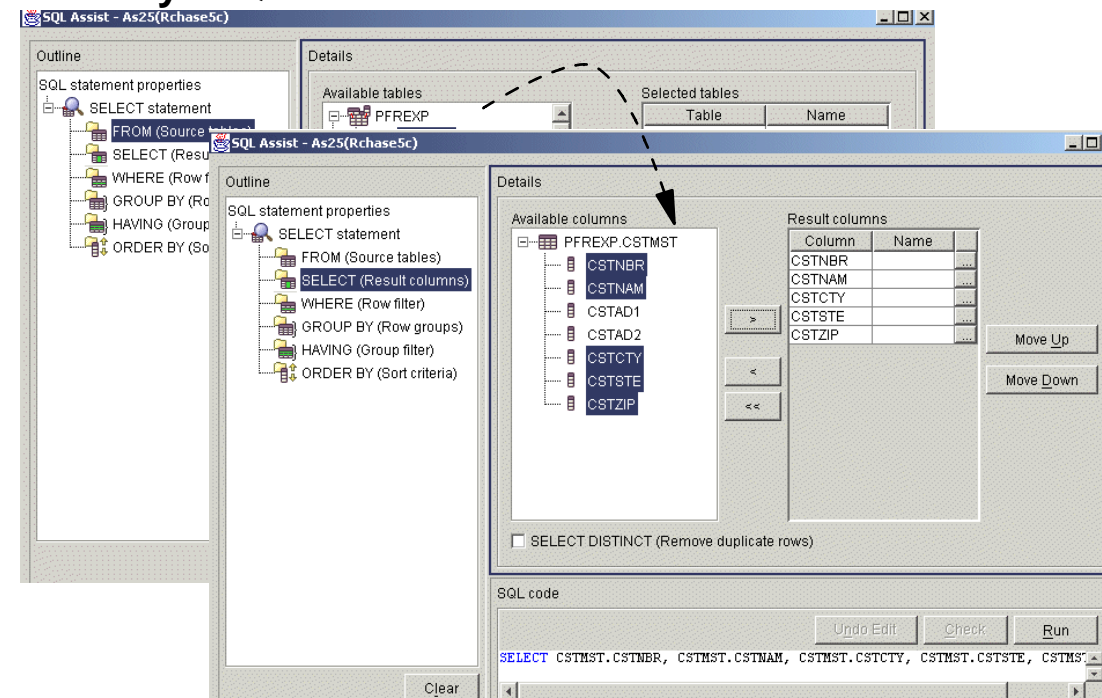
SQL code

SELECT *
FROM

© 2004 IBM Corporation

iSeries. mySeries.

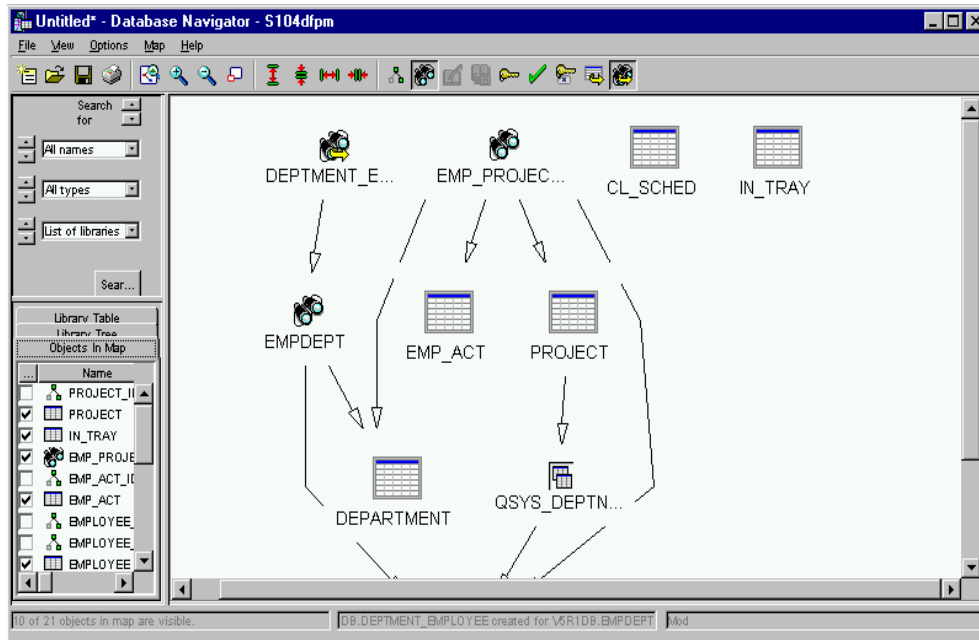
Usability: SQL Assist



Database Navigator

- Provides a graphical view of a database and its relations
- Gives a pictorial representation of a schema in order to:
 - Understand an existing complex database schema
 - Create a new schema
 - Manage objects and relationships in the schema
- Resulting picture is a Database Navigator Map (DNM)

Database Navigator Map



© 2004 IBM Corporation

iSeries. mySeries.

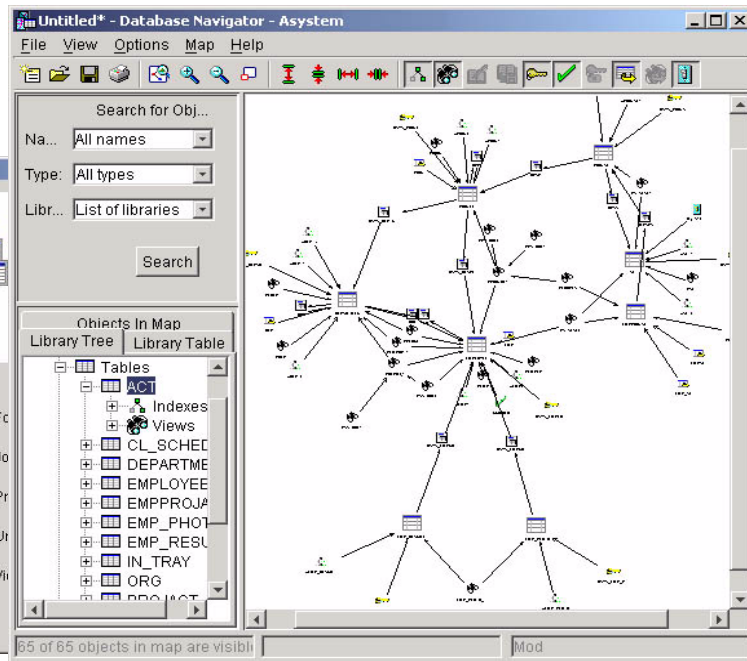
Usability: Enhanced Database Navigator

- Trigger Recognition
- Enhanced Graphics
- Improved Printing

Finding Relations

Related Objects:

8	Tables	11	Fc
5	Aliases	Omitted	Jo
1	Check constraints	7	Pr
12	Indexes	0	Ui
Omitted	Journals	20	Vu
1	Triggers		



© 2004 IBM Corporation

iSeries. mySeries.

Usability: Graphical EXPLAIN/PRTSQLINF

The screenshot shows the iSeries Navigator application. The left pane displays a tree view of the system structure, including 'My Connections', 'Asystem', and 'Databases'. The central pane shows a list of objects, with 'DNAME (SMALLINT)' selected. A context menu is open over this object, with 'Explain SQL' highlighted. The right pane displays the 'EXPLAIN SQL for CORPDATA.DNAME (SMALLINT) - Asystem' window, showing the following output:

```

C1
A 2722S81 V5R2M0 020719 Print SQL information Service program CORPDATA/DNAME 04/05/02 14:38:28 Page 1
A Object name.....CORPDATA/DNAME
A Object type.....SRVPGM
A CRTSOLCI
A OBJ(CORPDATA/DNAME)
A SRCFILE(QTEMP/QSQLSRC)
A SRCMBR(DNAME)
F COMMIT(NONE)
OPTION(*SQL *PERIOD *NOCNULROD)
TOTRLS(*PRV)
ALWCPYDTA(*OPTIMIZE)
CLOSOLCSR(*ENDACTGRP)
RDB(*LOCAL)
DATFMT(*ISO)
TIMFMT(*SO)
DFTRBCOL(*NONE)
DYNDFCOL(*NO)
SQLPKG(CORPDATA/DNAME)
ALWBLK(*ALLREAD)
DLYPRP(*YES)
DYNUSRPRF(*USER)
SRTSEQ(*HEX)
LANGID(ENUS)
RDBCNMTH(*DUW)
TEXT(SQL ROUTINE DNAME )
STATEMENT TEXT CCSI(37)
SQLPATH(QSYS/QSYS2/QSTATSR)
SELECT DEPTNAME INTO : H : H FROM CORPDATA. ORG WHERE DEPTNUMB = : H : H
SQL4021 Access plan last saved on 04/05/02 at 14:34:14.
SQL4020 Estimated query run time is 1 seconds.
SQL4020 Query attributes overrides from query options file QAQINI in library QUSRSYS.
SQL4027 Access plan was saved with DB2 UDB Symmetric Multiprocessing installed on the system.
SQL4017 Host variables implemented as reusable ODP.
SQL4010 Table scan access for table 1.
SET : H : H = : H : H
***** END OF LISTING *****

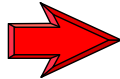
```

System Change Journal Management

- System automatically manages the journaled environment
 - attachment of new journal receiver(s) to the journal
 - optionally delete old journal receiver(s)
- Two new parameters for CRTJRN and CHGJRN commands
 - MNGRCV
 - *USER
 - *SYSTEM
 - *SAME (CHGJRN only)
 - DLTRCV
 - *NO
 - *YES
 - *SAME (CHGJRN only)
- To simulate today's environment - MNGRCV(*USER) DLTRCV(*NO)

Agenda

- Introduction
- Referential Integrity
- Check Constraints
- Database Triggers
 - System Triggers
 - SQL Triggers
- Column Level Security
- Two Phase Commit and DRDA Level 2
- Stored Procedures
- SQL Enhancements
- Miscellaneous Enhancements
- Summary



Summary

- New functions of DB2 UDB for iSeries
- Very powerful
 - use where required
 - use them carefully
- Explore possible design concerns prior to implementation

Summary...

Bibliography

- SC41-4721 CL Programming
- SC41-4611 DB2 UDB for OS/400: SQL Programming
- SC41-4612 DB2 UDB for OS/400: SQL Reference
- SC41-3212 DB2 UDB for OS/400: Query MANAGER User's Guide
- S246-0100 DB2/400: The New AS/400 Database
 - Skip Marchesani - Midrange Computing
- Database Design and Programming for DB2/400
 - Paul Conte - Duke Communications Press
- SQL/400
 - Paul Conte and Mike Cravitz - Duke Communications Press
- Informational APAR II09006
- IBM publications on the Internet
 - <http://as400bks.rochester.ibm.com>
 - V3 and V4 publications
- DB2 UDB for iSeries Web Page:
 - <http://www.as400.ibm.com/db2>
- Redbooks
 - SG24-4249-01 DB2/400 Advanced Database Functions
 - SG24-5184 DB2/400 Mastering Data Warehousing Functions