

ibm.com



e-business

# What's New in DB2 UDB for iSeries V5R2

BP01

ITSO iSeries Technical Forum

Hernando Bedoya



# Redbooks

International Technical Support Organization

The classic IBM logo consisting of the letters 'IBM' in a bold, sans-serif font, with horizontal stripes behind the letters.

F03BP01.PRZ

© 2003 IBM Corporation

# Acknowledgments



Thank to the following people who contributed to this presentation materials during an ITSO residency at the Rochester, MN Center.

- \* Satid Singkorapoom from IBM Thailand
- \* Nicolas Bueso from IBM Brasil
- \* Kent Milligan from PWD in Rochester

# DB2 UDB for iSeries Strategic Initiatives



## Openness - Industry Standard Support

- Accomodate ISVs
- Portability/Compatibility
- Flexibility

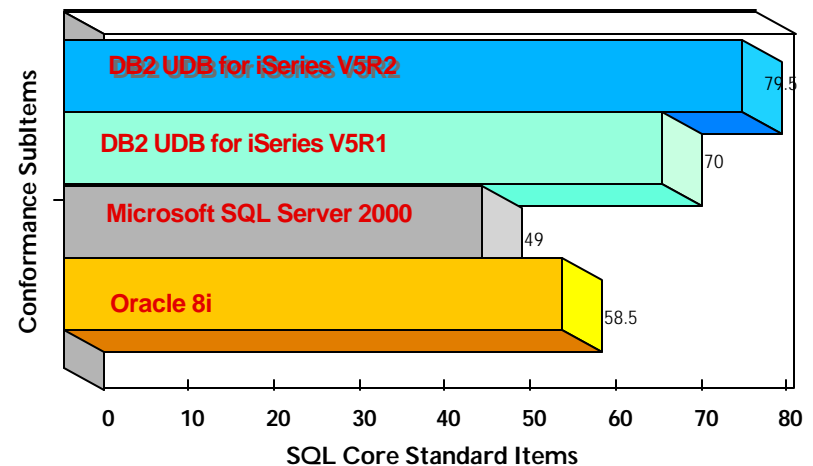
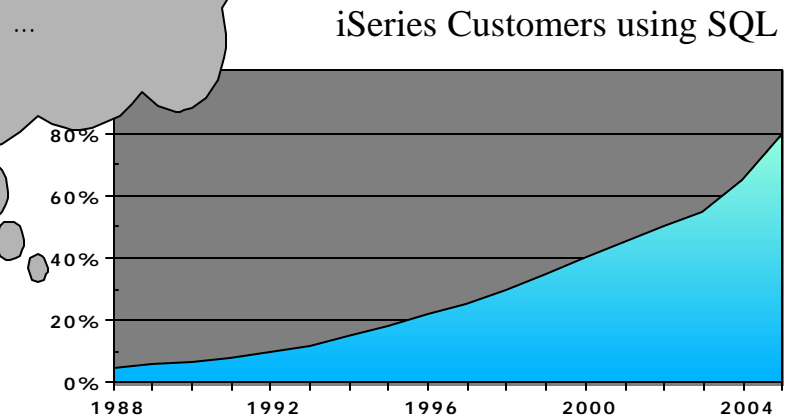
## Continued LEADERSHIP in database technologies

- Consistency across DB2 family
- Shared R & D across IBM Labs

## Continued Leveraging of iSeries Strengths

- Availability
- Scalability
- Usability - Total Cost of Ownership
- Application Flexibility

The industry trend toward **off-the-shelf** software results in a move to SQL



# Notes



SQL is the industry standard for database access and programming. While the heritage of application development on the iSeries has been to use RPG-like native interfaces such as Data Description Specifications (DDS) for defining databases, and using HLL languages such as RPG or COBOL to manipulate the data - ISVs and other application development efforts will be done using SQL.

From a terminology standpoint, there are many SQL-based database access methods, but they are all fundamentally using the SQL (Structured Query Language) constructs. For instance, JDBC, ODBC, DRDA, CLI are all common standards that leverage SQL.

A key BENEFIT of DB2 UDB for iSeries is that you can use SQL or DDS/HLL interfaces interchangeably because you have a SINGLE database management system (DB2 UDB for iSeries). For example, tables created with SQL can be accessed by HLL programs like RPG. Files (Tables) created with DDS can be accessed by SQL programs.

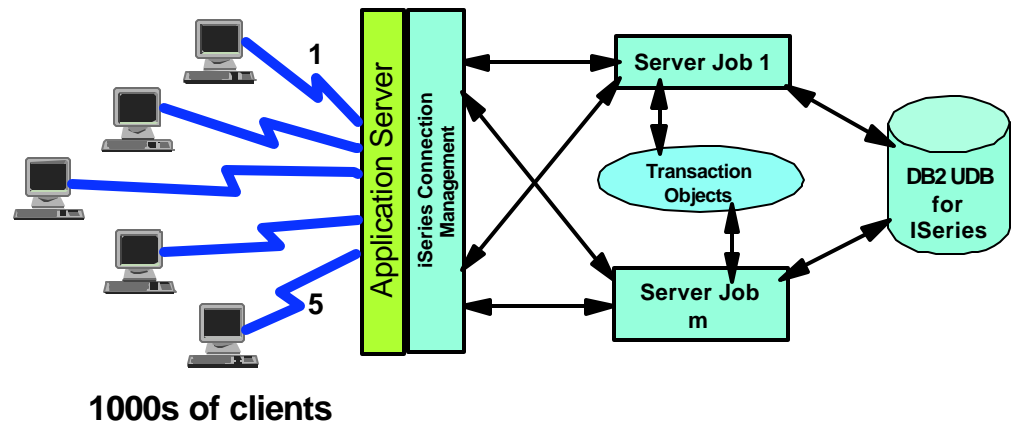
Another benefit of this architecture is you do not have to DIVE FULL SPEED AHEAD into SQL - but you can move at the pace that makes the most sense as you obtain more SQL skills.

# DB2 Re-engineering: Building the Foundation



## Adaptive e-transaction Services

- Extends iSeries robust transaction services to e-business applications
- Further optimizes iSeries performance for Websphere & Java transaction workloads
- Open Standards support:
  - XA & JTA (Java Transaction API)



## Database Technology Enhancements

- Incorporation of the latest query optimization techniques and algorithms
- Object-oriented query optimizer that lays foundation for self-learning query optimizer
- Streamlined data access for SQL interfaces
- Subset of read-only queries supported in V5R2



# Notes



One of the reasons that DB2 UDB for iSeries is such a reliable server is that much of the database engine code has been customer-proven for many, many years. Parts of the database engine even pre-dates the original AS/400 which arrived in 1988. One of the challenges with a mature code base is that it is harder over time to enhance that code to meet new industry standards and interfaces. Thus, a couple of re-engineering projects have been necessary to modernize the engine of DB2 UDB for iSeries. Two of these re-engineering projects have surfaced in V5R2 with the delivery of adaptive e-transaction services and a new query engine.

# DB2 UDB for iSeries: V5R2 Enhancements



## Application Flexibility & Portability

- Enhanced SQL Standards support
  - ▶ IDENTITY column attribute
  - ▶ UNION in a View
- Improved DB2 Family Compatibility
  - ▶ UDTFs & Temporary tables
- Adaptive e-Transaction Services

## Server Consolidation

- DB2 UDB in Linux
- Database Migration Toolkits

## Performance

- Improved EVI Maintenance
- Adaptive e-Transaction Services

## Usability

- iSeries Navigator Enhancements
- DB2 OLAP V8
- SQL Enhancements for Traditional Programmers
  - ▶ SQL Field Reference File
  - ▶ SQL Source Debugger

## Database Availability

- Switchable Disk Clustering with Independent ASP Support
- Journal Standby Mode
- Access Path Protection Advisor

***"Of all the vendors that have embraced the concept of the universal database it is arguably DB2 that is the most comprehensive. In particular, IBM appears to be the most firmly committed to offering an all-embracing product set." - Bloor Research, "Databases - an evaluation and comparison," Jan. 2002***

# Notes



Some of the more interesting V5R2 database enhancements that are covered here are categorized to provide a perspective on the strategy behind these enhancements.



# Agenda



## OS/400 V5R2 Enhancements for

- SQL Enhancements for Traditional Programmers
- SQL Enhancements for Compatibility and Portability
- iSeries Navigator: Database Functions
- Other DB2-related Enhancements

# Notes



This presentation has been divided in four main sections:

- **SQL Enhancements for Traditional Programmers:** In this section we will discuss the SQL enhancements of the database and SQL syntax support of V5R2.
- **SQL Enhancements for Portability and Compatibility:** In this section we will discuss the major SQL enhancements of the database that makes DB2 UDB for iSeries the database the complies the most with the SQL Standard.
- **iSeries Navigator Database enhancements:** Since V4R5, iSeries Navigator has become the main tool for all the database administration tasks in DB2 UDB for iSeries. In V5R2, there are two major enhancements that are covered in this presentation: SQL Assist tool and Enhanced Visual Explain
- **Other DB2-related tools:** We discuss other DB2-related enhancements such as ODBC/JDBC enhancements and SQL source-level debugger.

\*\*\*\*\*

**Note:** Throughout the presentation, we use the following terms :

"Schema" is equivalent to *library or database*.

"Table" is equivalent to *physical file*.

"View" is equivalent to *logical file*.

"Index" is equivalent to *keyed logical file*.

"Column" is equivalent to *field*.

"Row" is equivalent to *record*.



# SQL Enhancements for Traditional Programmers

# Notes



Let's start discussing theSQL enhancements that benefit the traditional iSeries programmers.

# SQL Enhancements for Traditional Programmers



- Field Reference Files and CREATE TABLE AS
- Packaging and Debug improvements
- SQL Procedure Debug improvements
- Library List Interoperability

# Field Reference Files & CREATE TABLE AS



- **CREATE TABLE AS** builds on the **CREATE TABLE LIKE** support introduced in **V5R1**
  - Allows data to be copied into newly created table
    - SQL Interface for CRTDUPOBJ & CPYF support
  - Allows for simulation of DDS Field Reference File support
  - Simplifies summary/work table creation into one step

```
CREATE TABLE SalesByRegion AS  
(SELECT region, SUM(to_sales) FROM sales GROUP BY region)  
WITH DATA
```

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

# Notes



SQL support for field reference files will be the most appealing new function to the traditional iSeries programming community. Field reference files allowed DDS definitions for common fields like city and phone number to be reused instead of duplicating in multiple physical file definitions. SQL's CREATE TABLE statement did not have any equivalent to field reference files, so that made it tough for you to convert physical file definitions over to SQL. CREATE TABLE statement has been enhanced with special syntax in V5R2 to make common field definitions stored in field reference files available to SQL tables. The first SQL statement shown here demonstrates how to use this new syntax. You want to create a customer table reusing the definitions for id, last names (lname), first names (fname) and city from the field reference file, ref\_file. If the city field definition was a 30-byte character field in the reference file, cust\_city will be defined as a 30-byte character field in the customer table. The WITH NO DATA clause specifies that the referenced SELECT statement should not be run to return data.

The second statement shows how this new support on the CREATE TABLE statement can be used to create and populate a work or summary table with a single SQL statement. The nested SELECT statement is processed by DB2 UDB to figure out the column definitions (type and length) that need to be created in the specified table and the WITH DATA clause tells DB2 UDB to run the specified SELECT statement and place the results in the newly created table.

# Packaging & Debug Improvements



- **SQL procedures, triggers, and function can be created without buying the DB2 SQL Development Kit product**
  - V5R1 eliminated the C compiler requirement
  - V5R2 eliminates the SQL C precompiler requirement
    - Easier deployment of SQL Triggers since most of the time they have to be recreated on production systems
  
- **SET OPTION DBGVIEW=\*SOURCE allows SQL procedures, triggers, and functions to be debugged without having to view generated C code**
  - Source debug can only be done within the job that created the SQL object (procedure, trigger, etc) - debuggable view stored in QTEMP
  - iSeries Navigator Generate SQL option makes it very easy to recreate SQL object with \*SOURCE debug view
  - DBGVIEW(\*SOURCE) parameter also added to RUNSQLSTM
  - Lays groundwork for graphical DB2 debugger



# Notes



Traditional iSeries programmers that are already utilizing SQL procedures, functions, and triggers to implement business processes will attest to the debug challenges associated with these SQL objects. When an SQL procedure, function, or a trigger is created, DB2 UDB generates a C-program object that implements the specified logic. iSeries programmers wanting to debug these SQL objects soon found that they had to step thru the C code generated by DB2 UDB which is not a pleasant experience for an RPG or COBOL programmer. To improve the debug process in V5R2, an SQL source-level debugger has been delivered so that the programmer only sees and debugs the SQL statements they coded. The SQL source-level debugger is activated by embedding this SQL statement, SET OPTION DBGVIEW=\*SOURCE within their SQL procedure, trigger, or function definition.

SQL source-level debugging can only be done within the job that created the \*SOURCE debug view since that debug view is stored in QTEMP.

iSeries programmers interested in leveraging SQL procedures, triggers, and functions will no longer have to first convince their boss to buy the DB2 SQL Development Kit product. When SQL stored procedures were first introduced, programmers first had to purchase and install the DB2 SQL Development Kit and the ILE C compiler - the V5R2 implementation has no dependencies on LPPs (licensed-program products).

# SQL Procedure Debug Improvements



The screenshot shows the iSeries System Debugger interface. The top menu bar includes File, Edit, Debug, Breakpoint, Actions, Window, and Help. Below the menu is a toolbar with various icons for debugging. The main window is divided into several panes:

- Programs Pane:** Shows a tree view of loaded programs, including /Qsys.lib/Kmtest.lib/Pp2.pgm.
- Source View (Top):** Displays the source code for procedure PP2. Line 7, `SET Y = ABSVAL ( ( X + 1 ) * Y );`, is highlighted in yellow. A red arrow points to this line from the text '\*SOURCE view'.
- Source View (Bottom):** Shows a detailed view of the highlighted line, with line 145, `17 SQLP_L1.Y = -9;`, highlighted in yellow. A red arrow points to this line.

\*SOURCE view

**V5R1  
PTFs:**  
SI06359  
SI06310  
SI06358

# Notes



You can eval the label or procedure name to get all the values.  
The indicators are always immediately following the variable.

```
EVAL TST
```

```
TST.V1 = 1
```

```
TST.SQLP_I2 = 0
```

```
TST.C1 = SPP:F68CDAC3240011F6
```

```
TST.SQLP_I3 = 0
```

```
EVAL PA99
```

```
PA99.P1 = 1
```

```
PA99.SQLP_I1 = 0
```

```
EVAL TST.V1
```

```
TST.V1 = 1
```

Character data must be eval'd with an \*. It's best to specify the length.

```
EVAL *TST.C1 :S 5
```

```
*TST.C1 :S 5 = "AAAAA"
```

## SQL Procedure Debug Improvements - \*SOURCE tips



- Accessing SQL variables & parameters

Parameters:

```
EVAL P22.PARM1
```

Variables:

```
EVAL SP
```

```
EVAL SP.X
```

```
EVAL *SP.Z :S 5
```

```
CREATE PROCEDURE p22(IN parm1 INTEGER)  
LANGUAGE SQL  
SET OPTION DBGVIEW=*SOURCE  
sp: BEGIN  
DECLARE x,y INT;  
DECLARE z CHAR(5);  
SET x = parm1;  
SET y = -9;  
SET y = absval((x+1)*y);  
SET z = 'ABCDE';  
END;
```

# Library List Interoperability



- **New SET SCHEMA statement for controlling current library via SQL interface (Dynamic SQL only)**
  - Example: SET SCHEMA=proplib
  - Does not effect unqualified procedure, UDF, and UDT references which are controlled by the SET PATH statement
- **New special register, CURRENT SCHEMA, contains the current schema value**
  - VALUES (CURRENT SCHEMA) INTO :hostvar
  - Defaults to \*LIBL for \*SYS naming and USER special register for \*SQL naming
- **SQLID treated as synonym for SCHEMA for DB2 UDB Family compatability, Ex: SET SQLID=proplib**
- **Dynamic SQL executed from an SQL Package (ie, Extended Dynamic) will always use the DYNDFTCOL setting from the package if it has been specified**

# Notes



Traditional programmers will also benefit from being able to control the current library for dynamic SQL statements with the new SET SCHEMA statement.

Static embedded SQL statements (eg, a SELECT embedded in an RPG program) are not affected by the SET SCHEMA statement

Unqualified procedure, function, and UDT references are also not affect since they rely on the SET PATH statement

SET SCHEMA equivalent to calling the QSQCHGDC API that was made available on previous releases - this API sets the default collection for Dynamic SQL requests.

Dynamic SQL statements that are stored in an SQL Package that has been created with a Dynamic Default Collection will continue to use this specified collection (or schema) and ignore any SET SCHEMA requests.



# **SQL Enhancements for Portability & Compatibility**

# V5R2 DB2 UDB Enhancements



- New support for IDENTITY column attribute
- New support for ROWID data type and ROWID function
- New support for UDTF (User-defined Table Function)
- Table Statistics Manager
- Many SQL enhancements



# Notes



Among the most important enhancements to DB2 UDB for iSeries in V5R2 are:

- New support for IDENTITY column attribute. Very useful for automatic value increment generation for a column.
- New support for ROWID data type and ROWID function which identifies a row in a table giving the row a unique value.
- New support for User Defined Table Functions. This is an improvement to the concept of UDFs announced back in V4R4.
- New table statistics manager which will provide an alternative to creating new indexes.
- Many more SQL enhancements
- Other new enhancements will be discussed.

# IDENTITY Column Attribute

New in  
V5R2



## For columns with data types:

- INTEGER, SMALLINT, BIGINT, DECIMAL, or NUMERIC

## For automatic "running number" values

- Can be incremented or decremented with:
  - Starting value and stepping value (+ or -)
  - Minimum and maximum limits
  - Whether to allow external input of values
  - Whether to recycle the value when limit is reached
  - Not guaranteed to be unique, **primary key constraint** or **unique index** should be defined over the column

## Value generated only for newly-inserted rows

## Declared by using CREATE / ALTER TABLE or iSeries Navigator

# Notes



## Identity column attribute

This new feature is used to declare only one column (per table) with a data type of INTEGER, SMALLINT, BIGINT, NUMERIC or DECIMAL to be used as the identity column with automatic value increment (or decrement). This is useful when you want to use that column to contain an automatic running numeric value that serves, for example, as an identity for each row of data.

The running number occurs for every newly inserted row.

You can specify many attributes for the identity column, such as:

- Starting value and its step (positive value for increment or negative for decrement)
- Minimum and maximum values
- Whether to allow external input of the value in this column or not
- Whether to recycle the value when its maximum or minimum limit is reached or not
- An identity column cannot coexist with another column of type ROWID in the same table.

You can also use SQL syntax ALTER TABLE <table name> ALTER COLUMN to:

- Change attributes of the existing identity column
- Add a new column with the identity definition (if none exists yet)
- Drop an existing identity column definition (without dropping the column).

Attention: This syntax does not support declaring an existing eligible column into an identity one (if none exists yet).

# IDENTITY Column Attribute



LIB99.TEST3 Table Properties - I400ws(I400ws) ? X

Column Name	Type	Length	Description
C1	CHARACTER	9	
C2	DECIMAL	5,0	
C3	BIGINT		
X1	SMALLINT		

Browse ...  
New  
Delete

Column | Key Constraints | Indexes | Referential Constraints | Triggers | Check Constraints

The properties below apply to the column definition currently selected above.

Short column name: X1

Heading: X1

Set as identity column

Step value: -1

Restart with value: 9999

Minimum value: 1

Maximum value: 9999

Allow applications to provide a value for the column

Cycle values when the limit is reached



# Notes



iSeries Navigator support for Identity Column declaration

- Right-click the schema object where the table is to be created and select New --> Table from its pop-up menu
- or
- Right-click an existing table object and select Properties from its pop-up menu

\*\*\*\*\*

**IDENTITY\_VAL\_LOCAL** function can be used to retrieve value generated by the database. For example: **VALUES**  
**IDENTITY\_VAL\_LOCAL() INTO :IVAR**

- This function is not affected by the following statements: UPDATE, COMMIT, ROLLBACK, and Non-Identity INSERT
- BEFORE Trigger would obtain the generated value with Trigger transition variable

# ROWID Data Type

New in  
V5R2



Used as a **unique identifier** of each row among tables

- Declaring a unique constraint guarantees uniqueness

## System-generated value

- Generated for existing and new rows
- 40-byte bit-oriented value : not subject to CCSID
- Generated by an algorithm that produces highly unique value
  - Take server's serial number as one of the input
- Example of use: bank account transaction ID, sales order transaction ID, etc.

## Can receive value from **external sources**

- Use ROWID scalar function to convert the external value

## Notes



The ROWID data type can be specified for only one column in a table to hold a unique identity value (40-byte long) for a row of data. This identity value is calculated by a complex formula that takes into account, for example, the machine's serial number, so that even each row of data in tables residing in different machines of the same organization can be uniquely identified with a very low possibility of duplicated ROWID values.

The ROWID data type is not subject to CCSID because it is treated by DB2 UDB as bit-oriented data.

You have no control over the attributes of the ROWID data type. If you need to control certain attributes of the identity value, identity column attribute may be your alternative option. But the identity column attribute is based on a relatively simpler implementation concept than the ROWID type. So, it may not sufficiently serve the purpose of a very highly unique identification of a data row in many situations.

# ROWID Data Type



LIB99.SALES Table Properties - I400ws(I400ws)

Column Name	Type	Length	Description
SALES_DATE	DATE		
REGION	VARCHAR	15	
SALES	INTEGER		
SALES_STAFF	SMALLINT		
<b>STRAN_ID</b>	ROWID	40	

Browse ...  
New  
Delete

select \* from lib99.sales

SALES_DATE	REGION	SALES	SALES_STAFF	STRAN_ID
1995-12-31	Ontario-South	1	99	841F69712273EE9CF1F010F4D9E3F9D4404000000000000000001
1995-12-31	Ontario-South	3	99	841F697122E0987AF1F010F4D9E3F9D4404000000000000000001
1995-12-31	Quebec	1	99	841F697122E0D406F1F010F4D9E3F9D4404000000000000000001
1995-12-31	Manitoba	2	99	841F697122E10F74F1F010F4D9E3F9D4404000000000000000001
1995-12-31	Quebec	1	99	841F697122E14610F1F010F4D9E3F9D4404000000000000000001
1996-03-29	Ontario-South	3	99	841F697122E182B4F1F010F4D9E3F9D4404000000000000000001
1996-03-29	Quebec	1	99	841F697122E1BE6CF1F010F4D9E3F9D4404000000000000000001
1996-03-29	Ontario-South	2	99	841F697122E1F4D2F1F010F4D9E3F9D4404000000000000000001
1996-03-29	Ontario-North	2	99	841F697122E23140F1F010F4D9E3F9D4404000000000000000001
1996-03-29	Quebec	3	99	841F697122E26C8AF1F010F4D9E3F9D4404000000000000000001
1996-03-29	Manitoba	5	99	841F697122E2A2F2F1F010F4D9E3F9D4404000000000000000001
1996-03-29	Ontario-South	3	99	841F697122E2DE3CF1F010F4D9E3F9D4404000000000000000001

Allow applications to provide a value for the column



# Notes



A new column of type ROWID can be added to existing tables and ROWID values will be automatically generated for all existing rows.

# User-defined Table Function

New in  
V5R2



## Return rows of data in a temporary table

- Additional flexibility in tailoring a function for repetitive use

## Invoked by TABLE function

- Example: `SELECT * FROM TABLE ( <udtf> ) AS T1 ;`
  - <udtf> = Its name and parameters
  - "AS T1" is correlation clause representing temp table

## SQL UDTF or External UDTF

- No Sourced UDTF

## Improve performance of DB2 UDB XML Extenders

# Notes



User-defined functions (UDF) has been around since DB2 UDB for OS/400 was introduced in V4R4, but it has only been a scalar function that returns a single value each time it is invoked. As of V5R2, you can create a UDF that can return a set of rows of data, and thus its name of User-defined Table Function (UDTF). Like UDF, UDTF can be an external or SQL UDTF.

Unlike UDF, there is no “sourced” UDTF.

Imagine the advantage a UDTF can bring to you when you wish to standardize a complex customized SQL program that produces a result set into an SQL syntax that everyone who needs its functionality can use it from SQL interface. For example, you can create a UDTF named EMPBYPROJ that contains the following statement:

```
SELECT EMPNO,FIRSTNME,LASTNAME,BIRTHDATE
FROM SAMPLEDB01.EMPLOYEE WHERE EMPNO IN (
  SELECT EMPNO FROM SAMPLEDB01.EMPPROJACT
  WHERE PROJNO = '<project_number> ');
```

Then you can use it with the following standard one-line statement:

```
SELECT *FROM TABLE(EMPBYPROJ('<project_number>')) AS T1;
```

A UDTF is invoked by: `SELECT ... FROM TABLE( <UDTF_name> ( <its_parameters> ) ) AS T1 ;`

Like a UDF, you can create a UDTF by using SQL statement: CREATE FUNCTION or iSeries Navigator. For the latter method, you right-click the schema name in which you want to keep the UDTF definition and select New --> Function --> SQL or External.

# Creating a UDTF (1/2)



New SQL Function in SAMPLEDB90 - I400ws(I400ws) ? X

General | Parameters | SQL Statements

Function: EMPBYPROJ

Description: DISPLAY ALL EMPLOYEES IN A SPECIFIC PROJECT

Data returned to invoking statement

Single value  Table

Column Name	Type	Length	CCSID
EMPNO	CHARACTER	6	
FIRSTNAME	CHARACTER	20	
LASTNAME	CHARACTER	20	
BIRTHDATE	DATE		

Insert

Delete

Table cardinality:

Can run in parallel

Program does not call outside of itself (No External Action)

Same result returned from successive calls with identical input (Deterministic)

Attempt to run in same thread as invoking statement (Not Fenced)

Data access: Reads SQL data

Specific name:

# Notes



On this chart you can see the iSeries Navigator interface for creating User Defined table functions.

Note that in V5R2 the UDF can not only return a scalar function as in V4R4 but it can return a table.

# Creating a UDTF (2/2)



New SQL Function in SAMPLEDB90 - I400ws(I400ws) ? X

General Parameters SQL Statements

Parameter Name	Type	Length	CCSID	Description
PROJNBR	VARCHAR	6		

New SQL Function in SAMPLEDB90 - I400ws(I400ws) ? X

General Parameters SQL Statements

SQL statement examples: /\* SQL Control Statements \*/

Insert

Statements:

```
BEGIN
RETURN
  SELECT EMPNO, FIRSTNME, LASTNAME, BIRTHDATE
  FROM SAMPLEDB99.EMPLOYEE WHERE EMPNO IN (
    SELECT EMPNO FROM SAMPLEDB99.EMPPROJECT
    WHERE PROJNO = PROJNBR );
END
```

# Notes



Note that there are three tabs for creating an UDF.

On the first tab we define that the returning value is a table instead of a scalar function.

On the second tab we define any parameters that the function has an in input.

On the third tab we define the columns which will be the returning values of the table.

# Performance and Self-Management

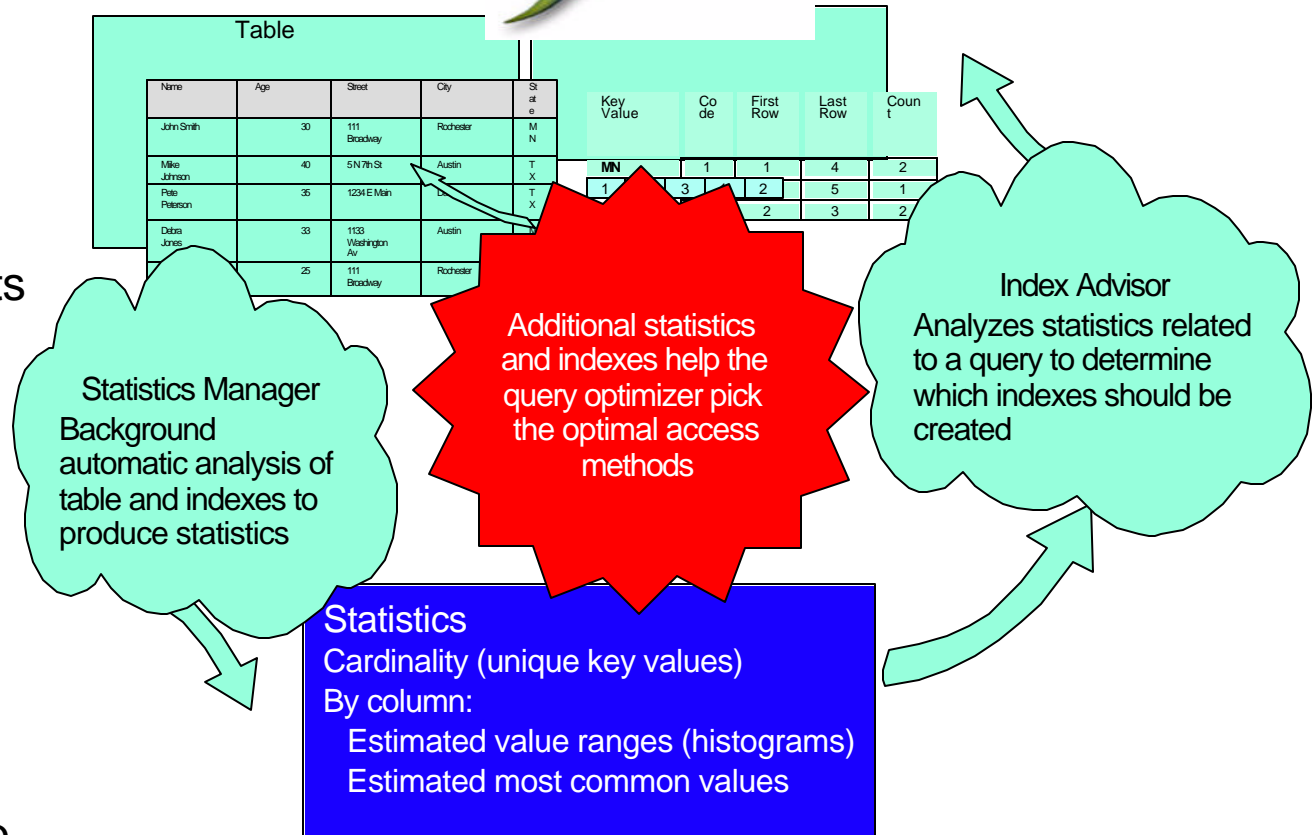


## Index Advisor

- Analyze and suggest recommendations for performance improvements

## Statistics Manager

- Additional statistics to aid query optimization
  - Estimated Value Ranges
  - Estimated Count of Most Common Values
- Stats provide alternative to creating new indexes





# Table Statistics Manager



## Collection of additional table and column statistics data

- Additional statistics to aid query optimization
  - Estimated Value Range in columns
  - Estimated count of Most Common Values in columns
- Providing alternative to creating new indexes
  - Less indexes to manage, less disk space occupied
- Automatic or manual update of statistics

# Notes



Since its creation DB2/400 or DB2 UDB for iSeries has rely on the statistics provided by the logical files or indexes to give the optimizer information on the best way to execute a query request.

As of V5R2 DB2 UDB for iSeries has a additional Table statistics manager which is able to collect statistics on tables and columns. This statistics are useful to aid the optimizer. It is an alternative to creating indexes.

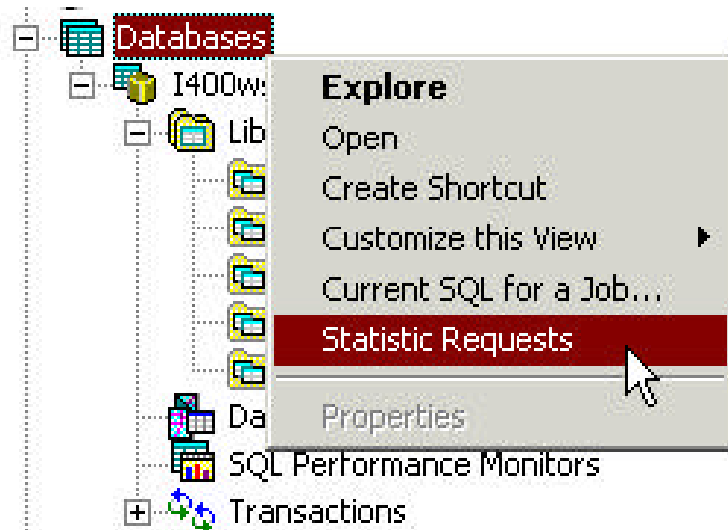
This statistics can be updated on a manual way or it could be automatically.

# Invoking the Table Statistics Manager

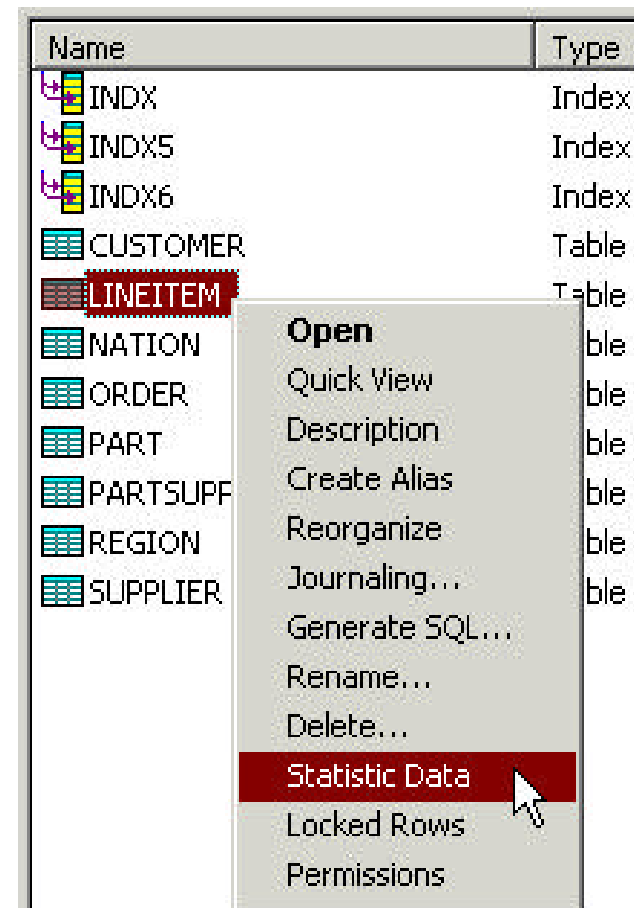


Invoked from three places:

- iSeries Navigator's Databases functional icon right-click pop-up menu



- iSeries Navigator's Table object right-click pop-up menu
- From V5R2 Visual Explain's Statistics and Index Advisor menu option.



# Notes



This chart shows the three different places where the table statistics Manager can be invoked.

# Collecting Statistics

New in  
V5R2



TPCH.PARTSUPP Statistics Data - I400ws(I400ws)

Statistic Name	Column	Type	Length	Stale	Aging	Requester	Translation T:
----------------	--------	------	--------	-------	-------	-----------	----------------

New...  
Update...

New Statistics

Columns available:

Column	Type	Length	Statist
PS_PARTKEY	INTEGER	4	No
PS_SUPPKEY	INTEGER	4	No
PS_AVAILQTY	INTEGER	4	No
PS_SUPPLYCOST	DECIMAL	7	No
PS_COMMENT	VARCHAR	199	No

Add -->  
Remove <--

Columns requested for collection:

Column	Statistic Name	Manual Aging
PS_PARTKEY	System-generated name	<input type="checkbox"/>
PS_SUPPKEY	System-generated name	<input type="checkbox"/>

Estimate Time   Collect Immediately   Collect in Background   Close   Help ?

# Notes



Once you have invoked the Table Statistics manager, you can start to collect statistics data for specific columns as shown on this chart.

# Displaying Statistics

New in  
V5R2



TPCH.LINEITEM Statistics Data - I400ws(I400ws)

Statistic Name	Column	Type	Length	Stale	Aging	Requester	Translation Table
QDBST_907AE00...	L_PARTKEY	INTEGER	4	No	Automatic	DBNAV99	
QDBST_907B0D0...	L_ORDERKEY	INTEGER	4	No	Automatic	DBNAV99	
QDBST_DFFAB40...	L_DISCOUNT	DECIMAL	7	No	Automatic	QSECOFR	
QDBST_A848650...	L_SUPPKEY	INTEGER	4	No	Automatic	DBNAV99	

New...  
Update...  
Details  
Remove...

Statistic Data Details

General | Estimated Value Ranges | Estimated Most Common Values

Value	Count
7434	9002
8310	9002
8396	9002
8705	9002
9455	9002
795	9002
840	9002
1751	9002
2644	9002
3049	9002
4571	9002
4733	9002
5604	9002
5748	9002
...	...

Statistic Data Details

General | Estimated Value Ranges | Estimated Most Common Values

Statistic name: QDBST\_A8486500B1D7183F93860004AC0261B4

Statistic last collected: 4/17/02 4:29:40 PM

Requester: QSECOFR

Statistic created: 3/21/02 1:37:39 PM

Creator: DBNAV99

Estimate of cardinality: 9984

Column nullable: Yes

Number of nulls: 0

Number of rows when collected: 6001215

# Notes



Once we have gathered statistics for certain tables , we can display the statistics by pressing the details radio button. On the chart we are looking the statistics for the column L\_SUPPKEY.



# SQL: Fullselect (UNION) in a VIEW



**UNIONS can now be included in common table expressions and derived tables**



**Example:**

```
CREATE VIEW AS total_sales (  
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' )
```

## Notes



In V5R1, the CREATE VIEW AS only supports subselect which means UNION cannot be specified. As of V5R2, you can use a fullselect statement which means UNION is supported.

V5R2: enhancements for derived tables and common table expressions

You can now use the following syntax in derived tables and common table expressions:

- Fullselect (SELECT with UNION)
- ORDER BY and FETCH FIRST n ROWS ONLY.

An example: `SELECT SUM(COMM) FROM ( SELECT COMM FROM SAMPLEDB01.STAFF WHERE COMM IS NOT NULL ORDER BY COMM DESC FETCH FIRST 10 ROWS ONLY ) AS T1 ;`

# SQL: GLOBAL TEMPORARY TABLE



- Creates a table for the current application process that is not registered in DB2 catalogs and cannot be shared by other processes (global within a process)
- When application process ends (job end or connection end), the temp. table is deleted
- Always created in SESSION schema (QTEMP library)

## Example:

- DECLARE GLOBAL TEMPORARY TABLE tmp\_projects (  
projno INTEGER, name CHAR(30), deadline TIMESTAMP)  
ON COMMIT PRESERVE ROWS
- DECLARE GLOBAL TEMPORARY TABLE summary  
LIKE proplib.sales\_summ ON COMMIT DELETE ROWS

# Notes



## V5R2: DECLARE GLOBAL TEMPORARY TABLE

This syntax creates a temporary table that is accessible to all applications running within the same process (global within a process). The table information does not appear in DB2 system-wide catalogs. Once created, the global temporary table can be manipulated in the same way you do with a normal table. But it cannot be accessed from other application processes and is automatically deleted when you end your application process.

Use this syntax in the same way you would use CREATE TABLE (including LIKE and AS).

# Procedural SQL Syntax (1/2)



**Procedural SQL, used in stored procedures, triggers, and functions, now supports the following:**

- **Nested compound statements**

- CREATE PROCEDURE CheckCategory ...

- BEGIN

- DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

- BEGIN SET PrevSQLState=SQLSTATE; SET ErrorState=ErrorOccurred;

- END;

- ...

- SET ErrorState=CleanState;

- SET MyCategory=SELECT Category FROM orders WHERE...

- ...

- BEGIN

- DECLARE CONTINUE HANDLER FOR SQLEXCEPTION...

- UPDATE inquiries SET inqCount=inqCount+1 WHERE MyCategory=...

- END

- ...

- END

# Notes



In procedural SQL used in Stored Procedures, triggers and functions nested compound statements are supported. Prior to V5R2 nested compound statements were not possible to code in a Stored Procedure, trigger or function.

# Procedural SQL Syntax (2/2)



- Iterate Statement

- ...

```
ins_loop: LOOP
  FETCH c1 INTO v_dept,v_deptname,v_admdept;
  IF at_end =1 THEN LEAVE ins_loop;
  ELSE IF v_dept ='D11 'THEN ITERATE ins_loop;
  END IF ;
  INSERT INTO department VALUES ('NEW ',v_deptname,v_admdept);
END LOOP;
```

- Multiple conditions for a handler

- ...

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, NOT FOUND
BEGIN
  SET PrevSQLState=SQLSTATE;
  SET ErrorState=ErrorOccurred;
END;
```

# Notes



The other two programming structures supported in Procedural SQL are:

- The ITERATE statement
- When you are defining a condition for a Handler, in V5R2 you can define multiple conditions for a handler.



# SQL: Scalar Subselect Enhancement



**Scalar Subselect was first allowed in the select-list in V5R1. V5R2 eliminates restrictions that were part of the initial support**

- Subselect can now be part of an expression and be included anywhere an expression is allowed
- Also now allowed on INSERT VALUES clause

## Example:

```
SELECT deptno, (SELECT MAX(salary) FROM employee WHERE  
workdept = department.deptno) AS maxsalary FROM department  
WHERE location = 'MN'
```

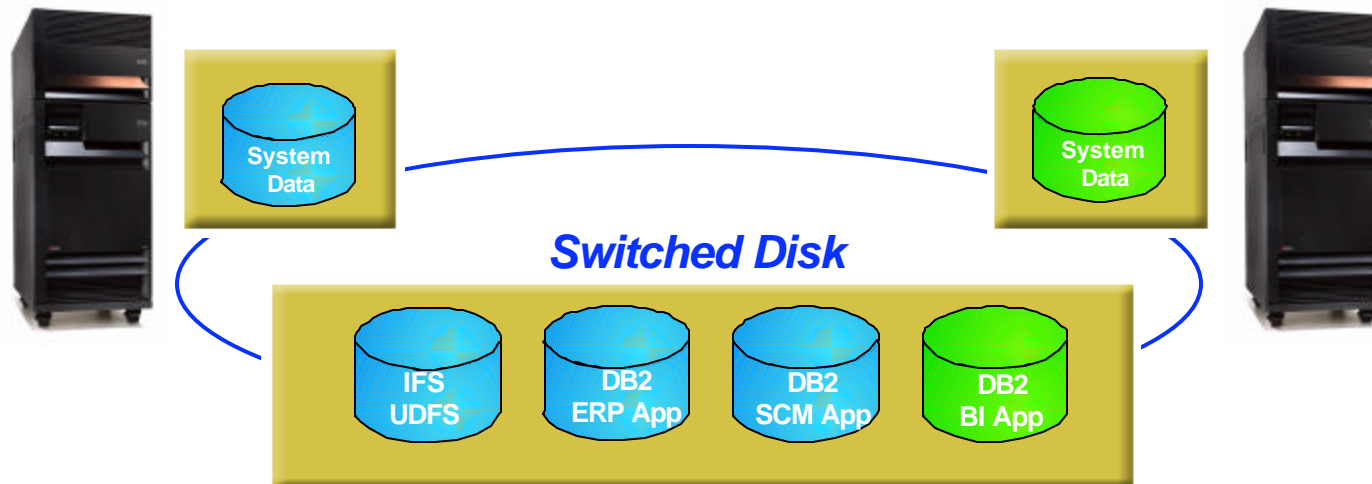
```
INSERT INTO department  
VALUES ( (SELECT MAX(deptno)+1 FROM DEPARTMENT),  
'Corporate Relations', 'MN' )
```

# Notes



Scalar subselect was supported prior to V5R2, but there were some limitations. In V5R2 such limitations have been lifted and now a subselect can now be part of an expression and can be included anywhere an expression is allowed. It can now be also allowed on the INSERT VALUES clause.

## DB2 object support in IASPs (Independent ASPs)



- **Support for switching database and non-database objects between systems**
- **Easy switchover of application database(s) for scheduled maintenance and upgrades**
  - Primary & Secondary system have to be within 250 meters
  - Clustering (IP Takeover) is the switching mechanism

## Multiple Database Names Spaces with IASP



- Library (database) names only have to be unique within an IASP
  - Example: one server could have 3 libraries named MYLIB, each in it's own independent ASP
  - Which "duplicate" library to access is controlled by one of the following:
    - ▶ Job description parameter (INLASPGRP)
    - ▶ SQL CONNECT statement
    - ▶ SETASPGRP CL command
- Provides another level of Server Consolidation by sharing a single application across multiple databases that share the same name
  - NOT recommended if the applications use SQL program, module, or package objects
    - ▶ SQL stores access plans in program & package objects that are dependent on the database location
    - ▶ SQL access plans will be rebuilt each time that an end-user accesses a database that's different than the previous execution with the program - this will cause performance problems
  - SQL application object would need a copy in each IASP to achieve acceptable performance

# Miscellaneous



- Result Set support for stored procedure calls between iSeries servers (ie, web server calls procedure on production server)
- Removal of ORDER BY restriction
  - Previously, columns specified in ORDER BY clause must exist in the SELECT list. As of V5R2, they do not have to.

**SELECT empno, lastname FROM employee ORDER BY birthdate**

- 64K SQL statement length (also delivered via PTF for V5R1)
- Fullselect supported on Table Expressions & Derived Views
  - Allows UNION & FETCH FIRST N ROWS to be specified

# Notes



On this chart there is a list of additional enhancements to DB2 UDB for iSeries. Such as:

- Improved support of result sets for stored procedures
- Previously, columns specified in an ORDER BY clause must exist in the SELECT list. As of V5R2 this is no longer necessary.
- Now an SQL statement can be up to 64K long.
- Fullselect support on Table expressions and derived views.



# **iSeries Navigator: DB functions V5R2 enhancements**

# Notes





iSeries Navigator is the new name for the previous Operations Navigator. This is the preferred graphical tool on the iSeries. From a Database point of view this is the Database administrators administration tool. Now let's look at the iSeries Navigator enhancements related to DB2 UDB for iSeries.



# V5R2 iSeries Navigator: DB functions



## New DB2-related capabilities in iSeries Navigator

- SQL Assist tool
- Managing transactions 
- Enhanced Visual Explain
- Enhanced Database Navigator
- Explain SQL for SQL package, procedure, or function 
- SQL Scripts Center

# Notes



This chart shows a list of all the new DB2 related capabilities in the iSeries Navigator as of V5R2.

- SQL Assist tool
- Managing transactions
- Enhanced Visual Explain
- Enhanced Database Navigator
- Explain for SQL Package, procedure and function
- SQL Script center.

# SQL Assist

New in  
V5R2



## A GUI that helps generate syntax for:

- SELECT, INSERT, UPDATE, DELETE

## Invoked from Run SQL Script Center

- Edit --> Insert Built SQL
- Edit --> Prompt SQL after highlighting an existing statement

## Some useful features

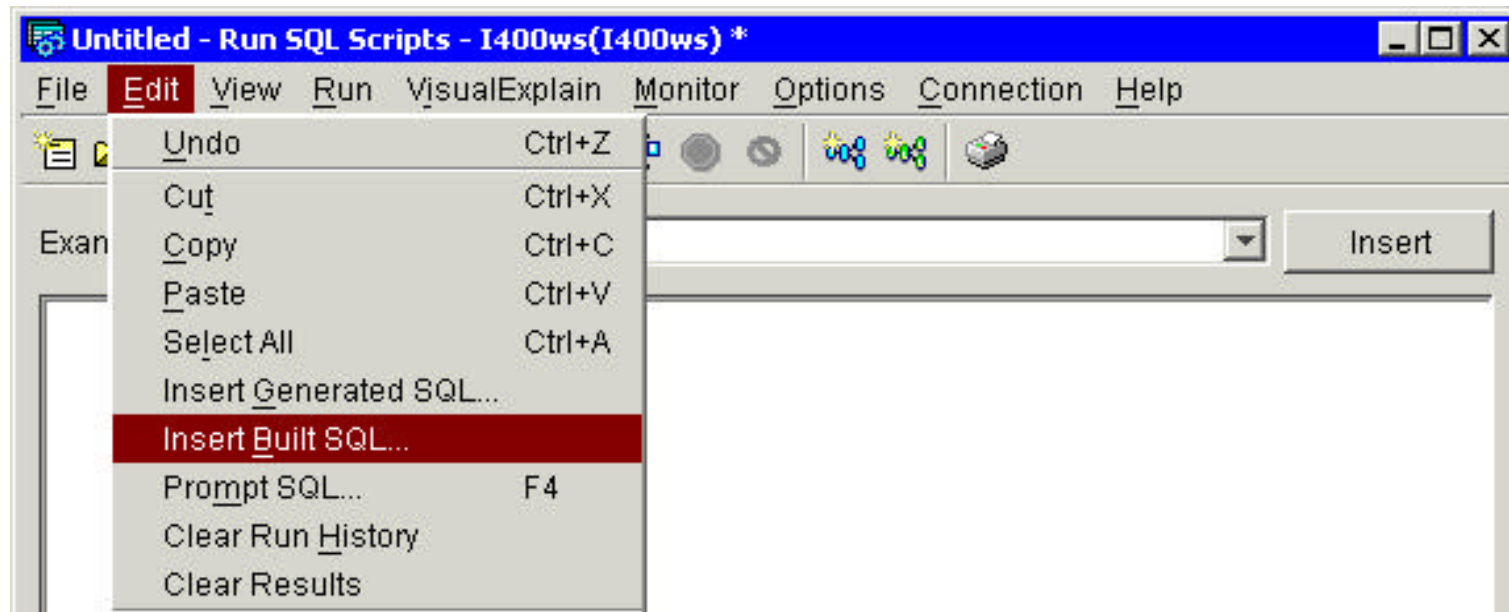
- Expression Builder: lists all supported SQL functions and procedure flow control
- Automatically identify declared primary/foreign keys relationships (from referential constraints)

# Notes



SQL Assist function brings the same ease-of-use feature of 5250 session's CL syntax prompt (F4) to Run SQL Scripts center. It helps you create some basic SQL statements (SELECT, INSERT, UPDATE, and DELETE) by going through step by step prompts in the GUI. It is intended to help those who are not fluent in SQL in building basic statements.

You invoke SQL Assist from Run SQL Scripts center menu item Edit --> Insert Built SQL.. Or you highlight a statement in the work area and press F4 function key or select Edit -> Prompt SQL. You use the latter method when you wish to modify an existing statement.



# SQL Assist



SQL Assist - I400ws(I400ws)

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: /I400WS  
User ID: DBNAV90

Change Connection...

SQL code

Undo Edit   Check   Run

```
SELECT *  
FROM
```

Clear

OK   Cancel   Help   ?

# Notes



This tool is an example of the advantages of the integration between the different DB2 labs. This is a tool which was borrowed from the DB2 for NT and Unix environment.

The main SQL Assist window is composed of three main panels: Outline, Details, and SQL Code

You can use the Change Connection button to switch your connection among multiple database instances that you wish to work with

The screenshot shows the SQL Assist application window titled "SQL Assist - I400ws(I400ws)". The main window is divided into three panels: "Outline", "Details", and "SQL Code". The "Outline" panel shows a tree view of SQL statement properties, including "SELECT statement", "FROM (Source tables)", "SELECT (Result columns)", "WHERE (Row filter)", "GROUP BY (Row groups)", "HAVING (Group filter)", and "ORDER BY (Sort criteria)". The "Details" panel shows the "Statement Type" section with radio buttons for "SELECT" (selected), "INSERT", "UPDATE", and "DELETE". Below this is the "Connection" section, which displays the current database alias as "I400WS" and User ID as "DBNAV90". A "Change Connection..." button is visible at the bottom of the "Connection" section.

The "Database Connection" dialog box is overlaid on the main window. It contains the following fields and controls:

- Database alias: /I400WS
- User ID: DBNAV90
- Password: \*\*\*\*\*
- Connect button
- Change Filter... button
- Driver type: JDBC-ODBC Bridge (dropdown menu)
- Database URL: jdbc:odbc:
- JDBC driver: sun.jdbc.odbc.JdbcOdbcDriver
- OK button
- Cancel button

A red arrow points from the "Change Connection..." button in the main window to the "Database Connection" dialog box.

# SQL Assist : Expression Builder



The screenshot displays the 'Expression Builder - Columns' window. The 'Columns' list on the left includes MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE, SALARY, **BONUS**, and COMM. The 'Operators' list includes +, -, \*, /, and CONCAT. The 'Value' field is empty. The 'Functions' list includes BLOB..., CEILING..., CHAR..., CHR..., CLOB..., **COALESCE...**, CONCAT..., and CORRELATION... The 'Constants' list includes CURRENT DATE, CURRENT TIME, CURRENT TIMESTAMP, CURRENT TIMEZONE, and USER. The 'Expression' field contains 'COALESCE (EMP.BONUS, 0)'. A red arrow points from the 'COALESCE...' function in the list to the 'Function Parameters - COALESCE' dialog box. The dialog box has a title bar 'Function Parameters - COALESCE' and a message 'Select a function parameters format and enter the parameters.' It features a 'Format' dropdown set to 'COALESCE(\*, \*) --> \*', 'Parameter 1 (\*)' set to 'EMP.BONUS', and 'Parameter 2 (\*)' set to '0'. Buttons for 'Clear', 'Undo', 'Redo', 'Help', 'OK', 'Cancel', and 'Help' are visible.

# Notes



The SQL Assist tool has an Expression Builder for making easier the creation of complex expression using a graphical interface.



# SQL Assist: Find Column Value



- Display all distinct values of a column

The screenshot shows the SQL Assist interface with the 'Find Values' dialog box open. The dialog box has a 'Search limit' of 25 and a list of values. The value 50 is highlighted. A red arrow points from the 'Find Values' icon in the 'Value' field to the dialog box. A text box at the bottom explains the icon's function.

SQL Assist - I400ws(I400ws)

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

Predicate

AND  OR

NOT

Column

EMPNO

Operator

IN

Value

Opens a window where you can select from a list of unique values for this column in the current database.

Find Values

Search limit: 25

Values: No limit, 25, 50, 100, 250, 500

The search limit has been reached. Only the first 25 values are displayed.

OK Cancel Search

# Notes

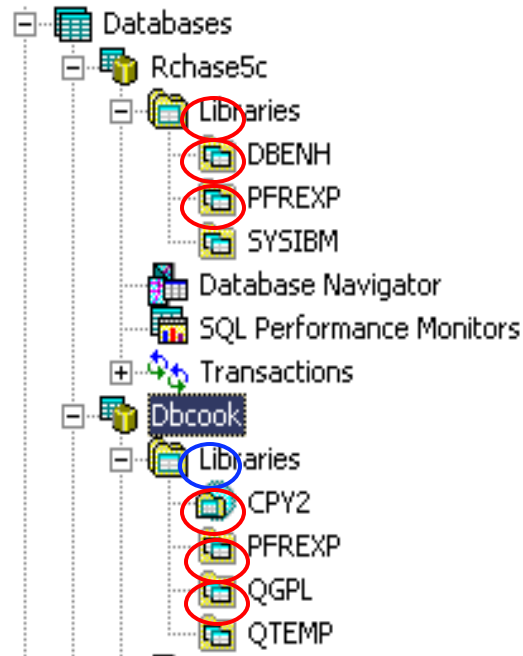




Another of the features of the SQL Assist is the Find Column Value which opens a window where you can select from a list of unique values for the column in the current database.

# Multiple Database Name Spaces with IASPs



**iSeries Navigator view:**



-  Schema / Library in IASP database
-  Schema / Library in system ASP database

**Relational DB Directory (WRKRDBDIRE) View:**

Relational Database	Remote Location
AS20	RCHASX20
DBC00K	LOOPBACK
DBEUOPS	LOOPBACK
RCHASE5C	*LOCAL

- **DDM access of IASP database objects controlled with new RDB parameter**
  - ▶ CRTDDMF ... RMTLOCNAME(\*RDB) RDB(DBCOOK)

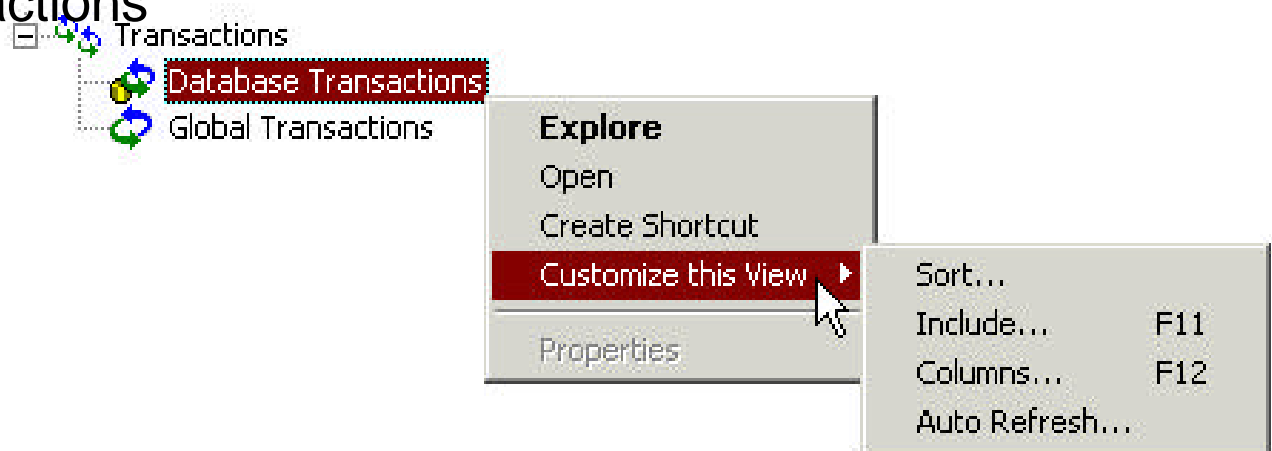
# Managing transactions

New in  
V5R2



## Manage two types of transactions:

- DB2 UDB-managed transactions
- X/Open global transactions



## Can customize view by:

- Sorting
- Include: transaction of interest
- Columns: attributes of interest

# Notes



V5R2 delivers a new support for Adaptive e-Transaction Services. This model complies with X/Open Architecture standard and JTA model that allow multiple transactions per database connection and multiple database connections per transaction. Each thread running in an OS/400 job can now have its own commitment definition and lock control rather than sharing these within the job. So, it is now more necessary for you to be able to monitor and manage transactions within your server. You use Transactions functional icon for such a purpose.

There are two icons here for you to select:

- Database Transactions: displays all transactions managed by DB2 UDB for iSeries
- Global Transactions: displays all transactions associated with X/Open global transaction

You can customize the view of these two types of transactions by right-clicking the functional icon and select Customize this View for a submenu which provides the following features:

- Sort: You can specify a sorting based on various combination of transaction attributes
- Include: You can specify a selection criteria of the transactions of your interest.
  - The available selection criteria for database transactions are: Unit of Work ID, Unit of Work State (All, Reset, Prepare in progress, Prepared, Last agent pending, Commit in progress, Committed, Vote read-only, Rollback required, Rollback in progress), Job Name, Job User, Job Number, Resynchronization in Progress (All, Yes, No)
  - The available selection criteria for global transactions are: Unit of Work ID, Unit of Work State (which contain the same list as database transaction criteria), Global Transaction ID, Branch Qualifier, Branch State (All, Active, Idle, Prepared, Rollback only, Heuristically completed, Non-existent), Lock Scope (All, Transaction, Job), Resynchronization in Progress (All, Yes, No)
- Columns: You can select to view transaction attributes of your interest in the right panel.
  - The available attributes for database transactions are: Unit of Work ID, Unit of Work State, Job Name, Job User, Job Number, Resynchronization in Progress, Commitment Definition
  - The available attributes for global transactions are: Global Transaction ID, Branch Qualifier, Branch State, Lock Scope, Unit of Work ID, Unit of Work State, Resynchronization in Progress

# Managing Transactions



Environment: My Connections | I400ws: Database Transactions | Database: I400ws

Management Central (I400ws)

- My Connections
  - I400ws
    - Basic Operations
    - Work Management
    - Configuration and Service
    - Network
    - Security
    - Users and Groups
    - Databases
      - I400ws
        - Libraries
        - Database Navigator
        - SQL Performance Monitors
        - Transactions**
          - Database Transactions
          - Global Transactions
        - File Systems
        - Backup
        - Application Development
        - AFP Manager

Unit of Work ID	Unit of Work State	Job	User	Number	Resynch...	Commitment Definition
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013292	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013294	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013297	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013300	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013295	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013296	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013298	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013302	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013301	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QSQS...	QU...	013299	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	OSOS...	OU...	013313	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013312	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013304	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013303	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013316	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013315	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013314	No	*DFACTGRP
APPN.RCHASM27.X'2D1...				013317	No	*DFACTGRP
APPN.RCHASM27.X'2D1...	Reset	QQQT...	QSY5	013170	No	*DFACTGRP

Context Menu for Transaction APPN.RCHASM27.X'2D1...:

- Force Commit...
- Force Rollback...
- Cancel Resynchronization
- Jobs
- Resource Status**
- Properties

**Resource Status for Transaction APPN.RCHASM27.X'2D1BCEB93401'.00001 - I400ws**

Record	Object	Journal	SNA Conversation	TCP/IP Connection	Remote File	RDB	API
	File	Library	Member	Commit	Rollback		

# Notes



## Can manage individual transaction on :

- Force commit
- Force rollback
- Cancel resynchronization
- Display associated jobs
- Display resource status
  - Row, DB Object, DB Journal, SNA conversation, TCP/IP connection, Remote file, Remote DB, APIs
- Display other properties



# Visual Explain



## Enhancements in V5R2:



- Print Preview
- Index and Statistics Advisor
  - Can launch windows for Statistics Collection or Index Creation without having to go back to iSeries Navigator main window
- Selectable Graph Detail and Attributes Detail
  - **Basic:** excludes detailed low-level operations
  - **Full:** includes detailed low-level operations



# Notes



In V4R5 , Visual Explain was announced for DB2 UDB for AS/400. Visual Explain provides a graphical representation of the optimizer implementation of a query request. The query request is broken down into individual components with icons representing each unique component. Visual Explain also includes information on the database objects considered and chosen by the query optimizer. Visual Explain's detailed representation of the query implementation makes it easier to understand where the greatest cost is being incurred.

Visual Explain shows the job run environment details and the levels of database parallelism that were used to process the query. It also shows the access plan in diagram form, which allows you to zoom to any part of the diagram for further details. If query performance is an issue, Visual Explain provides information that can help you to determine whether you need to:

Rewrite or alter the SQL statement

Change the query attributes or environment settings

Create new indexes

Best of all, you do not have to run the query to find this information. Visual Explain has a modeling option that allows you to explain the query without running it. That means you could try any of the changes suggested and see how they are likely to work, before you decide whether to implement them.

Visual Explain is an advanced tool to assist you with the task of enhancing query performance, although it does not actually do this task for you. You still need to understand the process of query optimization and the different access plans that can be implemented.

In V5R1, Visual Explain was enhanced to support more complex data access methods. It is now possible to save the explainable statements as a Performance monitor. You can visualize the optimizer messages without having to go to the job log.

In V5R2 there are additional enhancements for this tool such as:

- Print Preview
- Index and Statistics advisor
- Selectable graph detail and attributes detail.

On the following charts you will see this new enhancements.

# Visual Explain: Print Preview



The screenshot displays the Visual Explain application interface. The left pane shows the original query execution plan for the query: `SELECT * FROM PART WHERE P_PARTKEY IN ( SELECT ... )`. The plan consists of a **Table Scan** (20000 rows), an **Index Scan - Key Positioning** (60012 rows), and a **Nested Loop Join** (20000 rows). The **Print Preview...** menu item is highlighted, and a tooltip indicates "Preview printing this graph".

The right pane shows the printed version of the execution plan. It includes all the nodes from the original plan, but adds a **Final Select** node at the bottom, which is represented by a checkered flag icon. The printed plan also shows the row counts for each node: **Table Scan** (20000), **Index Scan - Key Positioning** (60012), **Nested Loop Join** (20000), and **Final Select**.

# Visual Explain: Index Advisor (1/2)



Visual Explain - I400ws(I400ws)

File View Actions Options Help

Statistics and Index Advisor

New in V5R2

Final Select

Nested Loop Join

Table Scan

Index Scan - Key Positioning

Temporary Index

Attribute	Value
<b>Time information (start time, tot...</b>	
Timestamp for Creation of Monit...	2002-04-
Statement Start Timestamp	2002-04-
Statement End Timestamp	2002-04-
Optimization Time, in Milliseconds	13
ODP Open Time, in Milliseconds	250
Total Time, in Microseconds	325728
Statement Open Time, in Micros...	325728
Statement Fetch Time, in Micros...	0
Statement Close Time, in Micros...	0
<b>Information about SQL stateme...</b>	
Statement Number	14

```
SELECT * FROM PART WHERE P_PARTKEY IN ( SELECT L_PARTKEY FROM LINEITEM WHERE L_DISCOUNT = 0)
```

Statement text Optimizer messages

# Visual Explain: Index Advisor (2/2)



Statistics and Index Advisor - I400ws(I400ws)

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:

Create ...

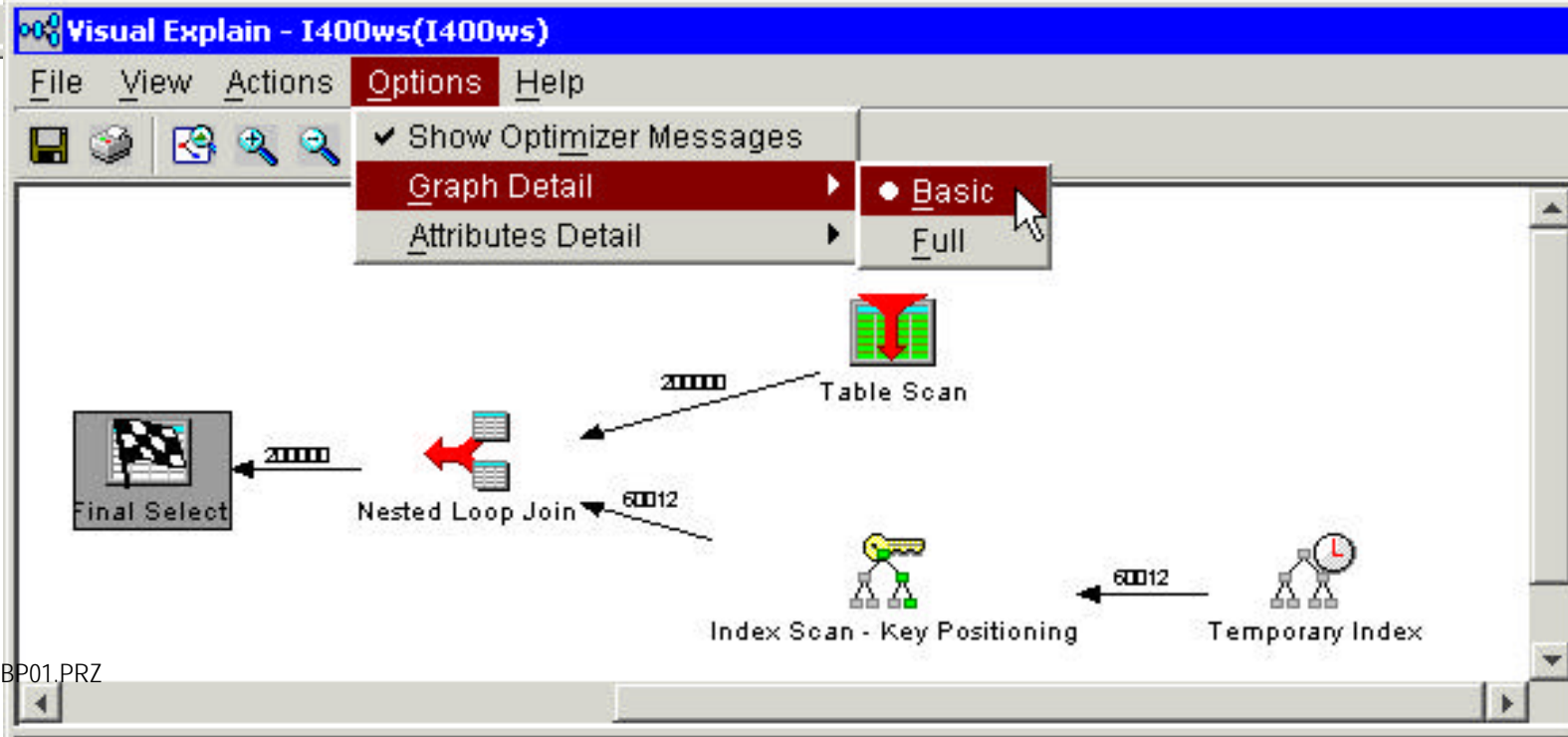
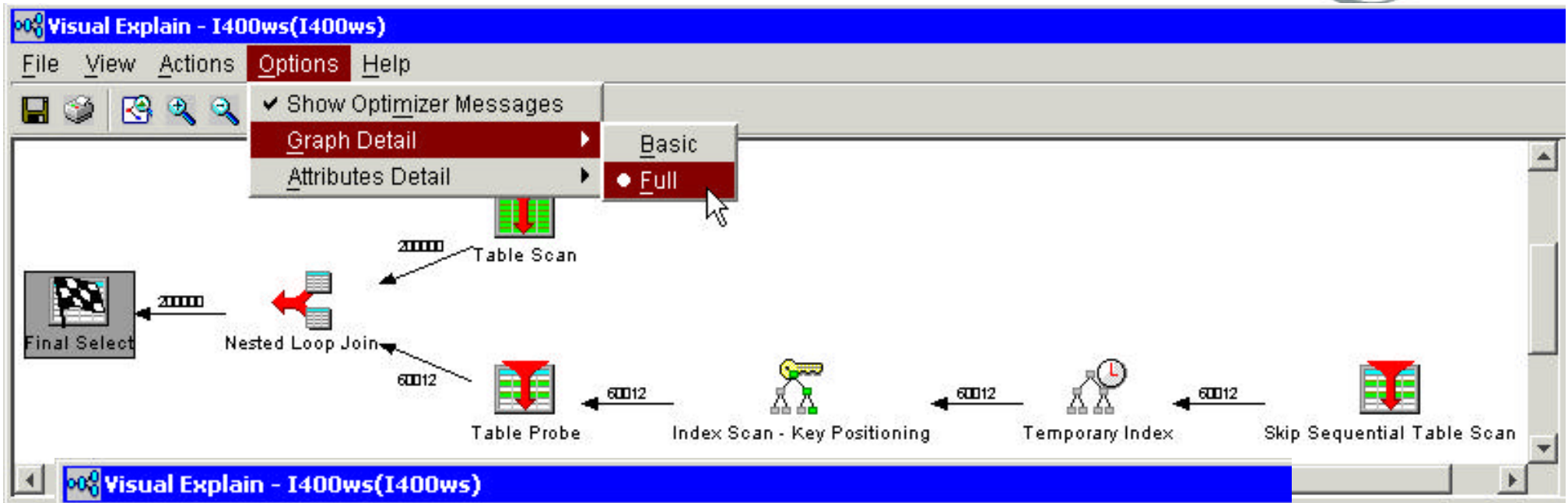
OK Help ?

# Notes



Now using Visual Explain you can see on one window all the indexes advised by the optimizer. You can decide to create this indexes by pressing the Create radio button.

# Visual Explain: Selectable Detail





# DB Nav: Better Map and Trigger Support



The screenshot shows the Database Navigator interface. The title bar reads "Untitled\* - Database Navigator - Asystem". The menu bar includes "File", "View", "Options", "Map", and "Help". The toolbar contains various icons, with a trigger icon (a blue square with a lightning bolt) circled in red. A red arrow points from this icon to a specific node in the database map, which is labeled "TRIGGER" in red text. The map itself is a complex network of nodes representing database objects, connected by lines. On the left side, there is a search panel with "Search for Obj..." and filters for "Name", "Type", and "Library". Below that is a "Library Tree" showing a list of objects including "Tables", "Indexes", "Views", "CL\_SCHED", "DEPARTME", "EMPLOYEE", "EMPPROJA", "EMP\_PHOT", "EMP\_RESU", "IN\_TRAY", and "ORG".

# Notes



The launch of DB2 UDB for iSeries Database Navigator, which is part of Operations Navigator in V5R1M0 Client Access Express, allows database administrators to view a graphical representation of the database that they are trying to administer.

With Database Navigator, you can explore the complex relationships of your database objects using a graphical representation called a map. The relationships that you see on the Database Navigator map are the relationships between:

Tables (for example, Referential Integrity constraints)

Any indexes over the tables

Any constraints, such as primary, foreign, unique, and check

Any views over the tables

Any aliases for the tables, etc.

Note: Database Navigator is not intended to be a data modeling tool like some existing products in the industry.

In V5R2 some enhancements were made to Database Navigator such as:

- Improved graphical tool to visualize the maps created by the tool.
- Better printing facility
- Trigger support was added to the tool.



# Explain SQL Support



1400ws: SAMPLEDB90 Database: 1400ws

Name	Type
DEPT	Alias
EMP	Alias
EMP_ACT	Alias
EMPACT	Alias
PROJ	Alias
<b>EMPBYPROJ ( VARCHAR() )</b>	Function
ACT1	New based on
XACT2	Generate SQL...
XDEPT1	<b>Explain SQL</b>
XDEPT2	Delete...
XDEPT3	Permissions
XEMP1	<b>Properties</b>
XEMP2	
XEMP_PHOTO	Index

**New in V5R2**

Explain SQL for SAMPLEDB90.EMPBYPROJ ( VARCHAR() ) - 1400ws(1400...

```

C1
DFTRDBCOL(*NONE)
DYNDFTCOL(*NO)
SQLPKG(SAMPLEDB90/EMPBYPROJ)
ALWBLK(*ALLREAD)
DLYPRP(*YES)
DYNUSRPRF(*USER)
SRTSEQ(*HEX)
LANGID(ENU)
RDBCNNMTH(*DUW)
TEXT('SQL ROUTINE EMPBYPROJ          ')
STATEMENT TEXT CCSID(37)
SQLPATH('QSYS' 'QSYS2' 'SATID')
*****

DECLARE SQL_TABLE_CURSOR CURSOR FOR SELECT EMPNO , FIRSTNME , ...
  BIRTHDATE FROM SAMPLEDB90 . EMPLOYEE WHERE EMPNO IN ( SELECT E...
  SAMPLEDB90 . EMPPROJECT WHERE PROJNO = : H : H ) FOR READ ONLY ...
SQL4021 Access plan last saved on 04/10/02 at 17:04:18.
SQL4020 Estimated query run time is 1 seconds.
SQL4027 Access plan was saved with DB2 UDB Symmetric Multiprocessing inst...
SQL4017 Host variables implemented as reusable ODP.
SQL4007 Query implementation for join position 1 table 1.
SQL4006 All indexes considered for table 1.
SQL4010 Table scan access for table 1.
SQL4007 Query implementation for join position 2 table 2.
SQL4006 All indexes considered for table 2.
SQL4009 Index created for table 2.
SQL4014 1 join column pair(s) are used for this join position.
SQL4015 From-column 1.EMPNO, to-column 2.EMPNO, join operator MF, join pr...
SQL4011 Index scan-key row positioning used on table 2.
SQL4016 Subselects processed as join query.
OPEN SQL_TABLE_CURSOR
FETCH SQL_TABLE_CURSOR INTO : H : H , : H : H , : H : H , : H : H
CLOSE SQL_TABLE_CURSOR
***** END OF LISTING *****
    
```

# Notes



Right-click an SQL package, procedure or function object and select Explain SQL from the pop-up menu. This produces the same result as running OS/400 command PRTSQLINF (Print SQL Information).



## V5R2 SQL Scripts Center can display **output parameter** values of a called procedure

```
CALL SAMPLEDB90.TOPCOMPAY(20, ?);
```

```
> CALL SAMPLEDB90.TOPCOMPAY(10, ?)
```

```
Output Parameter TOTAL = 9412.55
```

```
Statement ran successfully
```

```
> CALL SAMPLEDB90.TOPCOMPAY(15, ?)
```

```
Output Parameter TOTAL = 11181.70
```

```
Statement ran successfully
```

```
> CALL SAMPLEDB90.TOPCOMPAY(20, ?)
```

```
Output Parameter TOTAL = 11994.05
```

```
Statement ran successfully
```

Messages



# Notes



V5R2: Displaying output parameters of a procedure in Run SQL Scripts

Run SQL Scripts window is a good tool for you to use in testing stored procedures, UDF, or UDTF. Since it uses JDBC to connect to DB2 UDB for iSeries, you can call a stored procedure and receive the returned result without having to do any programming to prepare for the result.

As of V5R2, when you call, from Run SQL Scripts window, a stored procedure that contains output parameters, the returned values of all output parameters will automatically be displayed in the Messages tab for you.



# **Other DB2-related V5R2 Enhancements**

# Notes



Now lets take a look at some of the additional DB2 related V5R2 enhancements.

# V5R2 DB2-related Enhancements



## Other DB2-related enhancements:

- ODBC and JDBC catalog views
- V5R2 ODBC Enhancements
- Enhanced source-level debugger for SQL routines
- Licensed programs requirement for creating SQL routines
- Adaptive e-Transaction Services model



# Notes



On this chart you can see the additional enhancements related to DB2 on V5R2.



# ODBC and JDBC Catalog Views

New in  
V5R2



## ODBC and JDBC Catalog Views

- Satisfy metadata API from client applications that store their data in DB2 UDB for iSeries
- Compatible with those of DB2 UDB for OS/390 and z/OS and DB2 UDB UWO version 8
- Stored in a new schema: SYSIBM for better DB2 UDB portability
- Will be adjusted accordingly if new specifications come out
- Documented in Appendix G of V5R2 SQL Reference manual

# Notes



A new set of catalog views is added in V5R2 to satisfy ODBC and JDBC metadata API requests from client/server applications that use DB2 UDB for iSeries to stored their data. These views are compatible with those on DB2 UDB for OS/390 and z/OS and DB2 UDB UWO Version 8.

They will also be accordingly adjusted when ODBC or JDBC specification enhances or modifies its metadata APIs.

These views exist in a new schema named SYSIBM for better DB2 UDB portability.

You can find more details in appendix G: DB2 UDB for iSeries Catalog Views of the V5R2 SQL Reference.

The screenshot shows a window titled "1400ws: SYSIBM Database: 1400ws". It displays a list of database objects with columns for "Name" and "Type". The objects include tables like SQLTYPEINFO, SYSCHARSETS, and SYSDUMMY1, and views like CATALOG\_NAME, CHARACTER\_SETS, and CHECK\_CONSTRAINTS. A mouse cursor is pointing at the "SYSIBM" part of the window title.

Name	Type
SQLTYPEINFO	Table
SYSCHARSETS	Table
SYSDUMMY1	Table
SYSPRIVILEGES	Table
SYSTABLETYPES	Table
CATALOG_NAME	View
CHARACTER_SETS	View
CHARACTER_SETS_S	View
CHECK_CONSTRAINTS	View
COLUMNS	View
COLUMNS_S	View
INFORMATION_SCHEMA_CA...	View
PARAMETERS	View
PARAMETERS_S	View
REF_CONSTRAINTS	View
REFERENTIAL_CONSTRAINTS	View
ROUTINES	View
ROUTINES_S	View
SQL_LANGUAGES_S	View
SQLCOLUMNS	View
SQLFOREIGNKEYS	View
SQLPRIMARYKEYS	View
SQLPROCEDURECOLS	View
SQLPROCEDURES	View
SQLSPECIALCOLUMNS	View
SQLSTATISTICS	View
SQLTABLES	View

# ODBC Enhancements



## V5R2 ODBC

- Support for 2 GB LOBs (previously only 15 MB)
- 64-bit ODBC Driver for Windows
- ODBC Driver for Linux clients
- Enhanced support for SQLTablePrivileges & SQLColumnPrivileges
- Ability to get back additional descriptor information, such as the base table name for a result set column
- Kerberos authentication
- Improved documentation in iSeries Information Center web page

# Notes



The iSeries ODBC driver for LINUX clients can be downloaded from :  
<http://www.iseries.ibm.com/linux/odbc/>

Further information at : <http://www.iseries.ibm.com/clientaccess/toolkit/>

# Enhanced source debugger for SQL



## Source-level debugger now displays SQL source listing

- Previously, only display C-generated code level
  - SET OPTION DBGVIEW = \*STMT or \*LIST since V4R2
  - Quite a messy view
  - For tracing detailed information by experienced programmers
- New V5R2 option: SET OPTION DBGVIEW = \*SOURCE
  - Used in CREATE FUNCTION, PROCEDURE or TRIGGER
- Run in 5250 session **only**
  1. Run CREATE routine syntax (with RUNSQLSTM)
  2. SQL sources are stored in **QTEMP**/QSQDSRC file (member name = routine name)
  3. STRDBG lib/routine\_name (in the **same** 5250 session as 1.)
- Lay a future groundwork for graphical debugger

New  
in  
V5R2

# Notes



SQL routines are SQL functions, SQL stored procedures, or SQL triggers.

If you are an SQL programmer, you may find that, in many occasions, you wish to perform program source-level debugging on the SQL routines that you create. Since V4R2, you have been able to do this by using the syntax `SET OPTION DBGVIEW = *STMT` or `*LIST` with the `CREATE FUNCTION`, `PROCEDURE`, or `TRIGGER` syntaxes. But the source-level listing you can see in the debugger view is displayed in C-generated codes only, which tends to be quite long and thus difficult to identify the entities of your interest. In V5R2, an enhancement helps you see only SQL source codes listing in the debugger view.

Before you create an SQL routine, add the new syntax: `SET OPTION DBGVIEW = *SOURCE` before the program body of the SQL routine. After the routine is created, you can use the OS/400 command: `STRDBG lib/<routine name>` to invoke the source debugger. You can see SQL code debug view by pressing F15 and select "SQL Output View" option.

This support is not available through iSeries Navigator. You must use 5250 session for this support only. It is also important that you use OS/400 command: `RUNSQLSTM` to execute the SQL scripts that create the routines (with `SET OPTION DBGVIEW = *SOURCE`) and then run `STRDBG` command in that same 5250 session. This is because when the SQL routine is created, its source codes are added as a member (with the same name as the routine's name) into a source file named `QSQDSRC` which is always created in `QTEMP` library. The debugger then uses this member to display the source codes in the debugger view. `QTEMP` library is always destroyed when you log off.

# Requirement for Creating SQL Routines



## SQL procedures, triggers, and functions can be created with less licensed program requirement

- Before V5R1, need both CX1 (ILE C Compiler) and ST1 (SQL Development Kit)
- V5R1: no longer need CX1
- V5R2: no longer need CX1 nor ST1



## Need to install a free-of-charge OS/400 option: System Openness Includes

# Notes



The term SQL routine refers to SQL procedure, trigger, or user-defined function.

Before V5R1, you need two licensed programs to be able to create SQL routines:

57xx-CX1: ILE C Compiler

57xx-ST1: SQL Development Kit

In V5R1, you only need 57xx-ST1 to be able to create SQL routines.

As of V5R2, you do not need any of these licensed programs to create SQL routines.

You still need to install a free-of-charge OS/400 option "System Openness Includes" to create SQL routines.



# Adaptive e-Transaction Services



## Easier porting of new e-business solutions to the iSeries

- Eliminate iSeries specific code to manage transactions and database connections in e-business and distributed applications

## Compliant with XA standard and JTA

- Multiple transactions per database connection
- Multiple DB connections per transaction

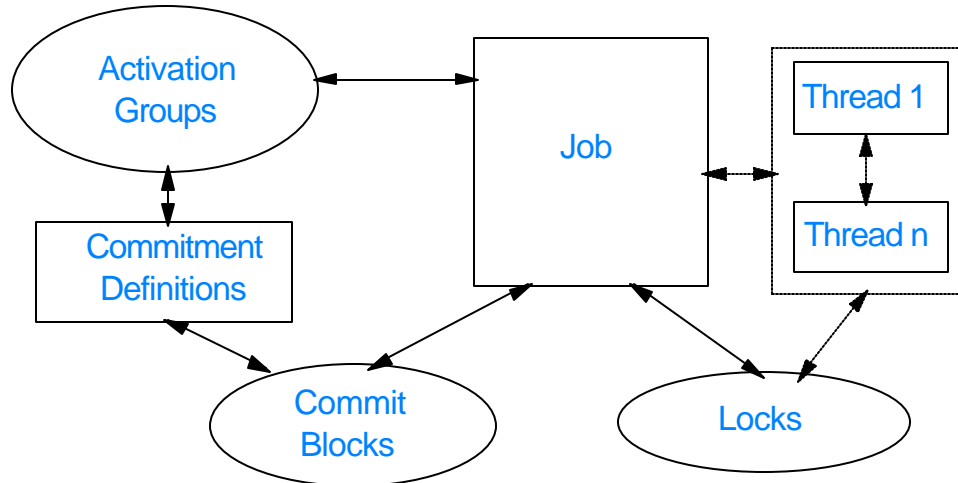
## Increased performance & improved scalability

- Resources shared between clients
- Efficient use of system resources
- No improvement to simple SQL transactions

# Notes Current versus New Model

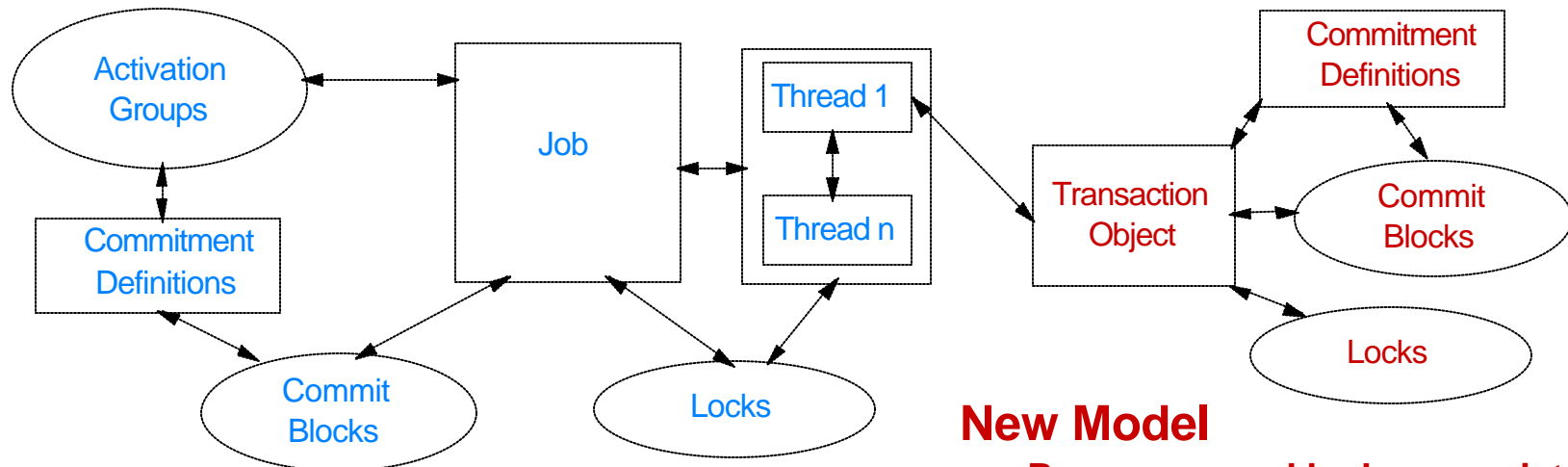


x



## Current Model

- Resources and locks associated with job



## New Model

- Resources and locks associated independent of job

# DB2 UDB for iSeries Strategic Initiatives



## Openness - Industry Standard Support

- Accomodate ISVs
- Portability/Compatibility
- Flexibility

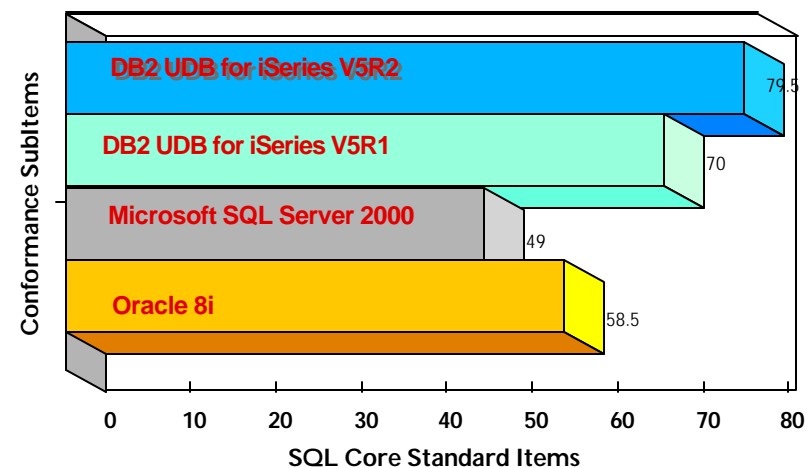
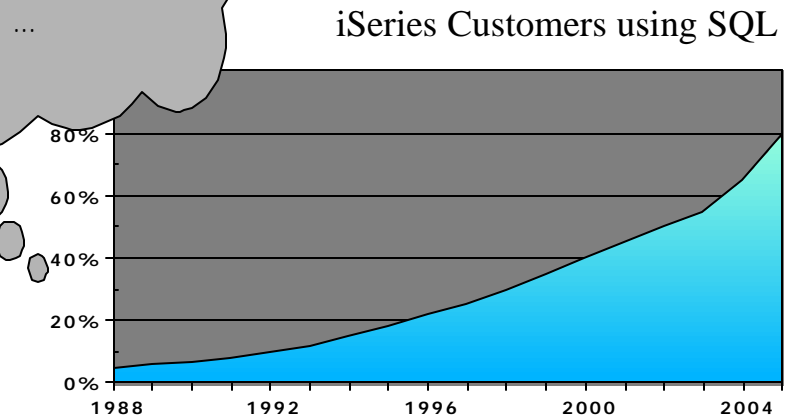
## Continued LEADERSHIP in database technologies

- Consistency across DB2 family
- Shared R & D across IBM Labs

## Continued Leveraging of iSeries Strengths

- Availability
- Scalability
- Usability - Total Cost of Ownership
- Application Flexibility

The industry trend toward **off-the-shelf** software results in a move to SQL



# Additional Information



- **DB2 UDB for iSeries home page** - <http://www.iseries.ibm.com/db2>
- **DB2 UDB Extenders Site:** <http://www.ibm.com/software/data/db2/extenders/>
- **Newsgroups**
  - ▶ USENET: comp.sys.ibm.as400.misc, comp.databases.ibm-db2
  - ▶ iSeries Network (NEWS/400 Magazine) SQL & DB2 Forum - <http://www.iseriesnetwork.com/Forums/main.cfm?CFApp=59>
- **Education Resources - Classroom & Online**
  - ▶ [http://www.iseries.ibm.com/db2/db2educ\\_m.htm](http://www.iseries.ibm.com/db2/db2educ_m.htm)
  - ▶ <http://www.iseries.ibm.com/developer/education/ibo/index.html>
- **DB2 UDB for iSeries Publications**
  - ▶ Online Manuals: <http://www.iseries.ibm.com/db2/books.htm>
  - ▶ Porting Help: <http://www.iseries.ibm.com/developer/db2/porting.html>
  - ▶ DB2 UDB for iSeries Redbooks (<http://ibm.com/redbooks>)
    - Stored Procedures & Triggers on DB2 UDB for iSeries (SG24-6503)
    - DB2 UDB for AS/400 Object Relational Support (SG24-5409)
    - SQL Query Engine Redpiece  
<http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg2456598.html>
    - Integrating XML with DB2 XML Extender and DB2 Text Extender (SG24-6130)

# Upcoming Residencies



## **IS-3101 - SQL Query Enhancements on DB2 UDB for iSeries - Phase 2**

This Rochester residency begins 21 Apr 2003, ends 30 May 2003 (6 weeks), and requires 3 residents.

## **IS-3102 - V5R1 and V5R2 DB2 UDB for iSeries Functionality Update**

This Rochester residency begins 7 July 2003, ends 15 Aug 2003 (6 weeks), and requires 4 residents.

# Trademarks and Disclaimers



© IBM Corporation 1994-2003. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	IBM
AS/400e	IBM (logo)
eServer	iSeries
 eServer	OS/400

Lotus and SmartSuite are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed.

Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.



**Thank for your attention**