

DB2 UDB for iSeries and Business Intelligence

2001 Announcements

ITSO Technical Overview

May 2001

IBM @server. For the next generation of e-business.

Agenda

Operations Navigator-based enhancements

General database enhancements

Business Intelligence Application Reference Summary

Operations Navigator

IBM @server. For the next generation of e-business.

New DB2-related capabilities in Operations Navigator include:

- Database Navigator, including basic visualization of database models and relationships
- Generate SQL
- Visual Explain enhanced for complex query types that were not supported in V4R5 version and usability
- Support for defining SQL Triggers & Java User-Defined Functions
- Hotlinking of datalink column values

Most of the changes in the GUI interface for database are documented in the *Operations Navigator* presentation. We will add in this presentation more detail on selected functions, such as the Database Navigator and the SQL Generation.

Note: In order to use the "terminology of the marketplace," this presentation will emphasize the use of the terms such as table, view, rows and columns and now "schema" (similar to old-fashioned term like "collection"), which is actually an OS/400 library with tables, views, and special tables that contain, catalog information.

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

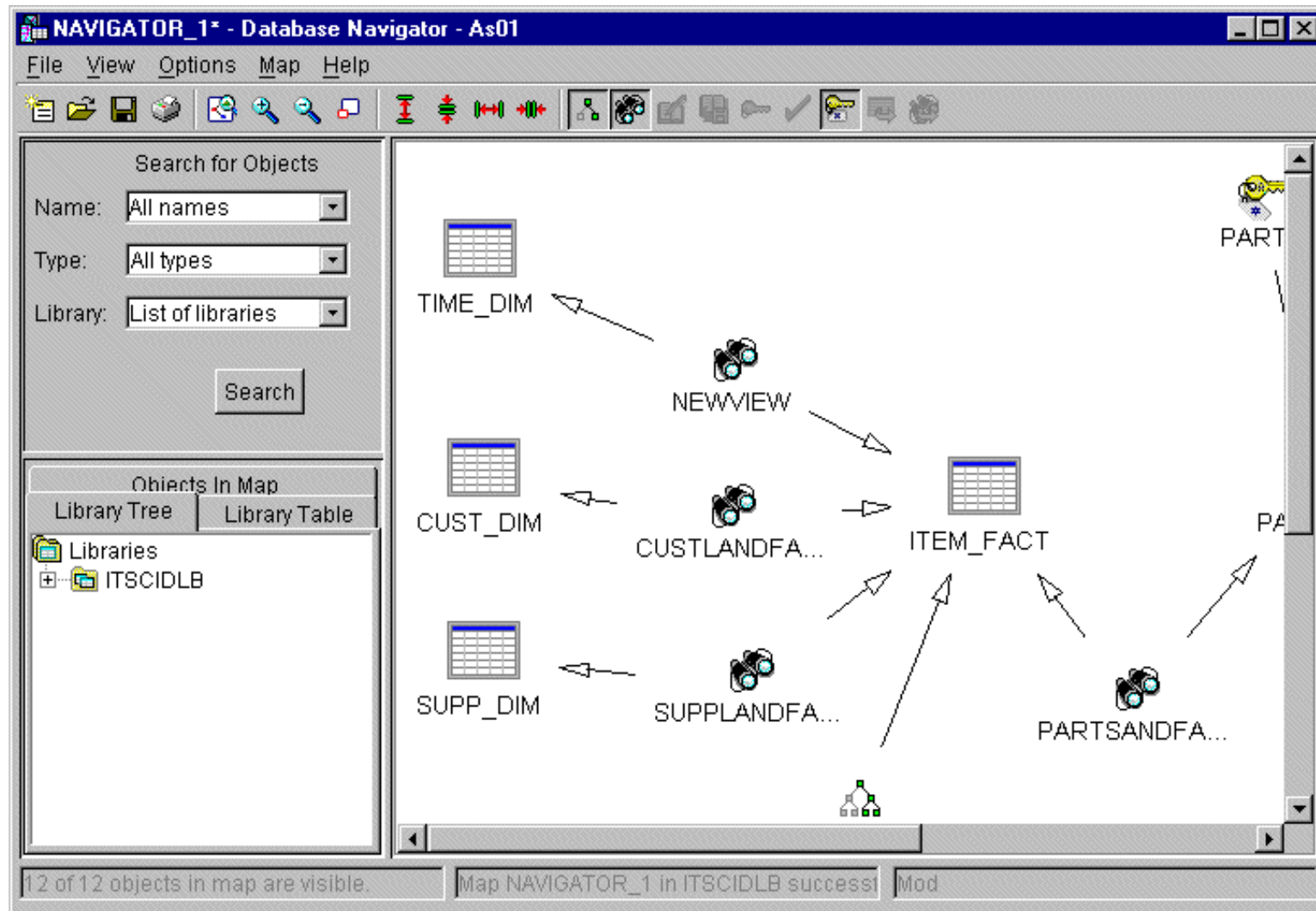
Resulting picture is a Database Navigator Map (DNM)

The information to construct the DB Navigator Map is retrieved from the database catalogs/system cross-reference files. The map therefore represents a snapshot of the data at a particular point in time, so you should remember that relationships and statistics will become out of date over time. A user can refresh as required the maps which he has built.

The following capabilities are provided:

- Generate a map of a set of tables and the relationships between them.
- Manipulate the map to show items of interest to the user, without changing objects on the iSeries. This includes:
 - Adding tables and views that exist on the iSeries but not originally included in that instance of the Database Navigator Map.
 - Removing any of the objects from the map.
 - Changing object placement.
 - Zooming.
- Make changes to the objects shown in the map, which results in changes on the iSeries. The user can perform operations currently provided by Operations Navigator against the objects in the map. The user can also create new SQL objects which are then displayed in the map.
- Generate the SQL for all objects in the map.
- The user can view the create SQL statements for any given object, for selected objects, or all objects. They can also run and save these statements from an SQL scripts window.
- Print maps.
- Save maps and view them again later.

Database Navigator - Interface



The primary workspace for Database Navigator is a window that is divided into areas that allow you to find the objects to include in a map, show and hide items in a map, view the map, and check on the status of changes pending for a map. The following provide a description of the main areas of the Database Navigator window.

Locator Pane: The Locator Pane, on the left side of the Database Navigator window, is used to find the objects that you want to include in your new map, or to locate objects that are part of an open map. The upper Locator Pane is a search facility that can be used to specify the Name, Type, and Library of the objects that you want to include in the map. The results of the search are displayed in the lower Locator Pane under the Library Tree and Library Table tabs. When the results are displayed under these tabs, you can add objects to the map by right-clicking on an object and selecting Add to Map or double-clicking on the object name. Then, when the map is created, you can see a list of the objects in the map by clicking on the Objects In Map tab.

Map Pane: The Map Pane, on the right side of the Database Navigator window, graphically displays the database objects and their relationships. In the Map Pane you can:

- Add tables and views that exist on the system, but that were not originally included in the current instance of the map
- Remove objects from the map
- Change object placement
- Zoom in or out on an object
- Make changes to objects in the map
- Generate the SQL for all objects in the map

Object Status Bar: The Object Status Bar, located on the bottom left of the Database Navigator window, displays the number of visible and eligible objects in the map.

Action Status Bar: The Action Status Bar, located on the bottom center of the Database Navigator window, provides a clear description of what has taken place in the map, and whether modifications are pending.

Modification Status Bar: The Modification Status Bar indicates whether a modification has been made or is pending.

Reverse engineering of DDL for DB2 objects

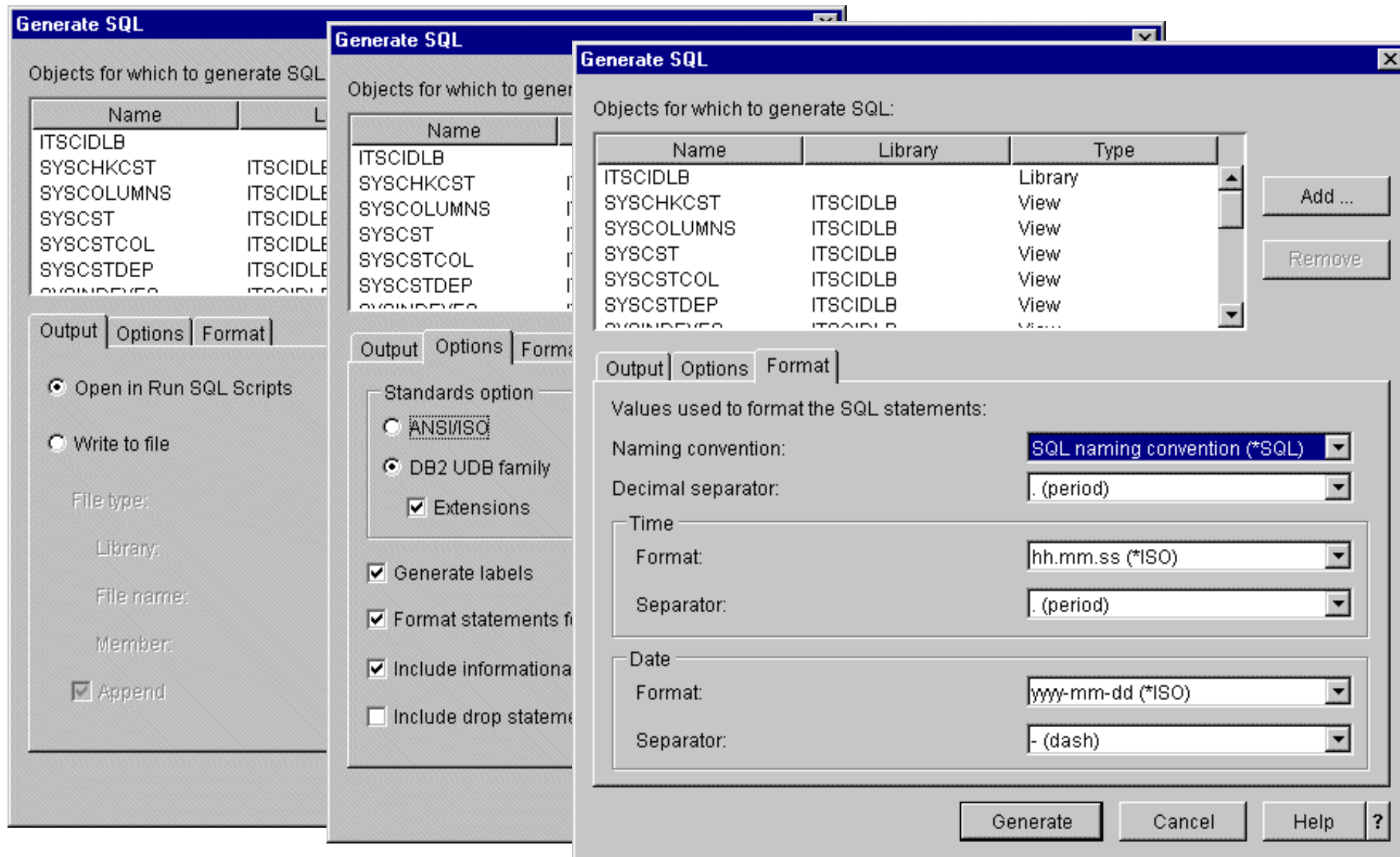
- Useful in converting object definitions from DDS to SQL
 - Not all DDS features can be converted, tool will convert as much as possible and generate warnings for unconvertible options
- Useful in generating SQL script for creating databases

SQL for one or multiple objects:

- Aliases
- Distinct Types
- Functions and Procedures
- Indexes
- Schemas (collections) and libraries
- Tables
- Views

Resulting script can be edited

Generate SQL For Multiple Objects



To bring up the **Generate SQL** panel, right click on a library or any of its objects in **Database** definitions in Operations Navigator. When you click OK, all of the objects that are listed in the **Objects for which to generate SQL** are processed. You can edit this list by selecting **Add** to add an object to the list or **Remove** to remove an object from a list.

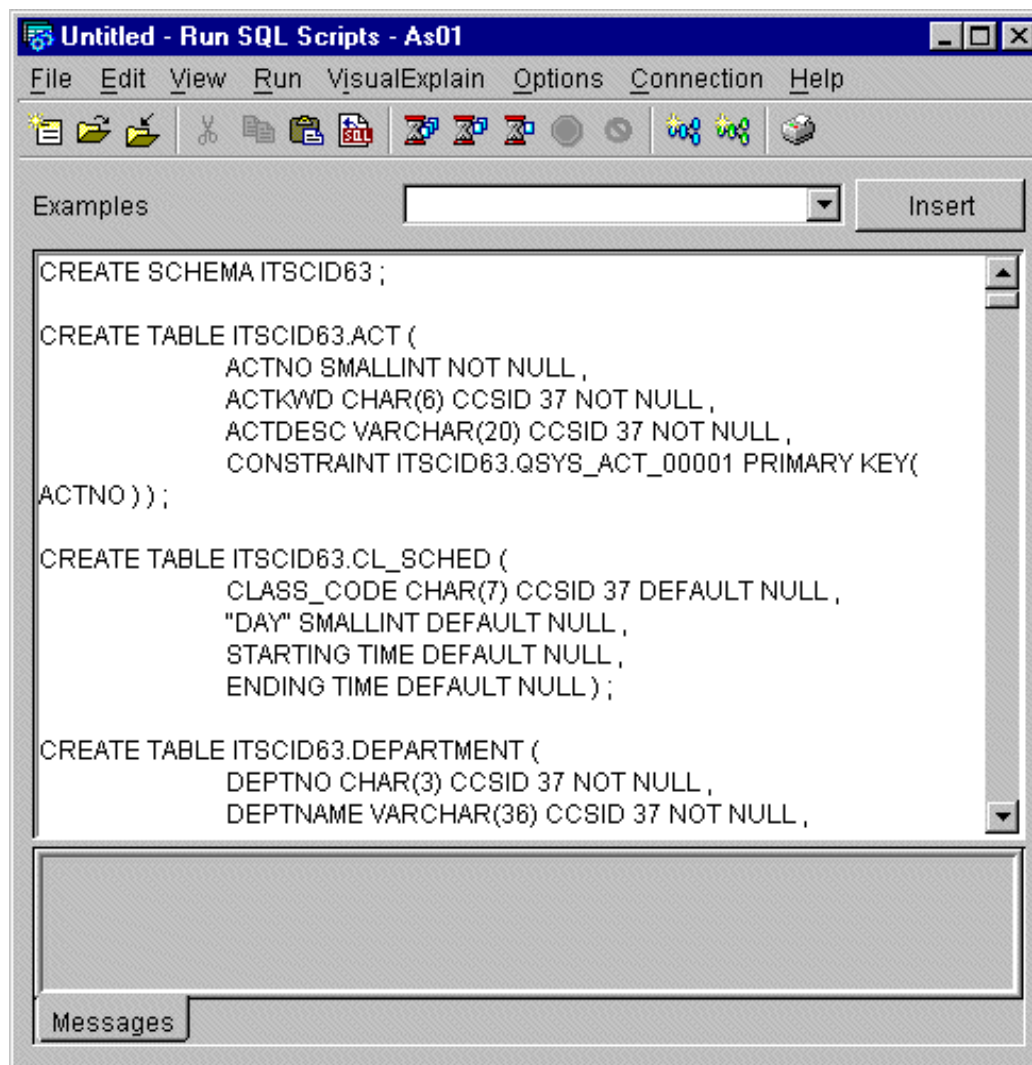
The **Output** tab lets you choose to open the generated SQL in **Run SQL Scripts** or write the output to a file. Additionally, you can choose to write to an OS/400 database source file or to a PC file. Opening the generated SQL in **Run SQL Scripts** allows you to edit and run the SQL immediately. Writing the output to a file gives you the option to save the generated SQL and to run the script later.

The default is to open in **Run SQL Scripts**.

The **Format** tab displays the format values used for your generated SQL whenever time, dates, and literal constants are generated. Selections you make for the Standards option on the Options dialog can cause these values to become invalid.

On the **Options** tab, you can choose different options for the generated SQL.

Generated SQL Output

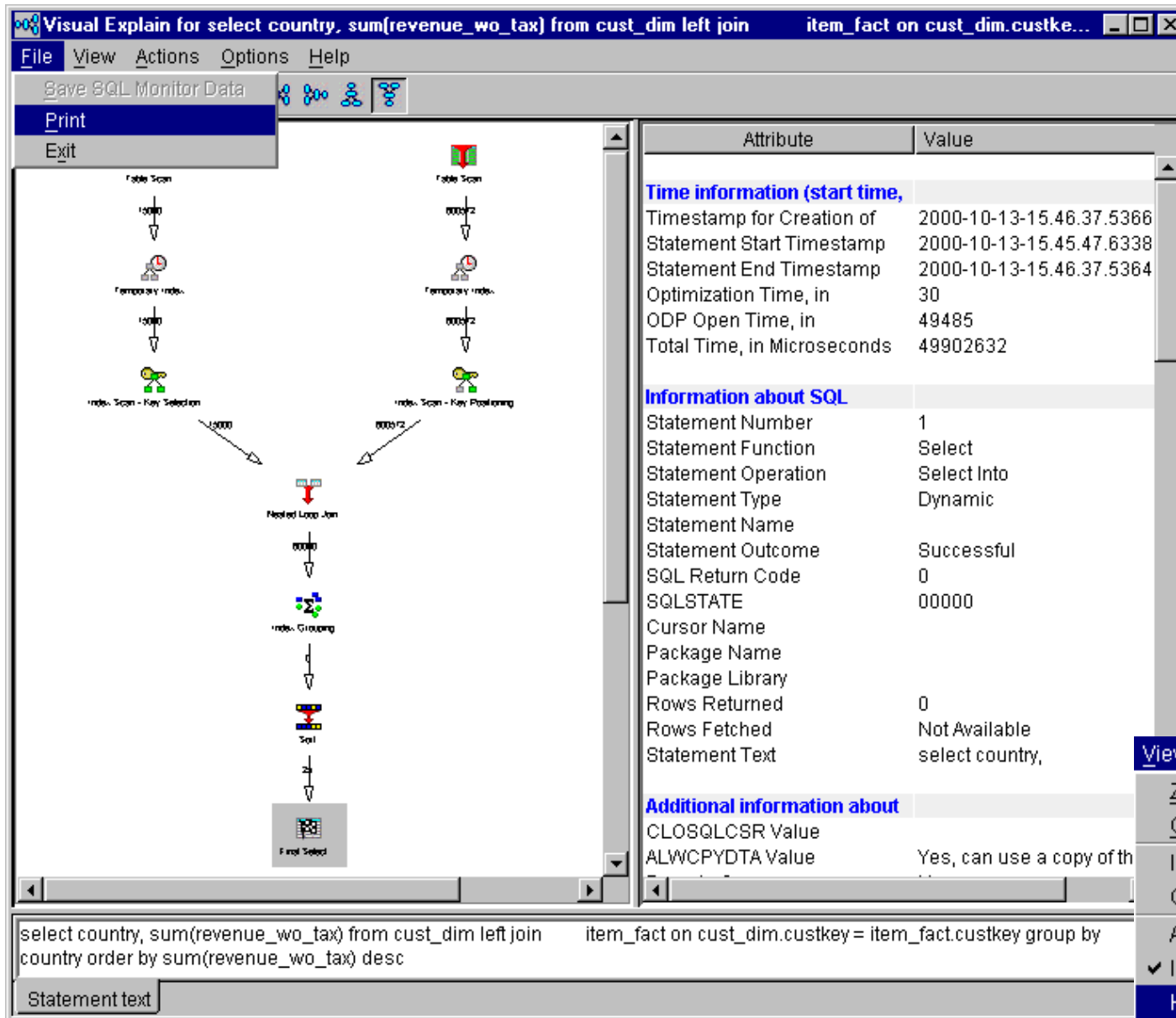


```
CREATE SCHEMA ITSCID63 ;

CREATE TABLE ITSCID63.ACT (
    ACTNO SMALLINT NOT NULL ,
    ACTKWD CHAR(6) CCSID 37 NOT NULL ,
    ACTDESC VARCHAR(20) CCSID 37 NOT NULL ,
    CONSTRAINT ITSCID63.QSYS_ACT_00001 PRIMARY KEY(
ACTNO ));

CREATE TABLE ITSCID63.CL_SCHED (
    CLASS_CODE CHAR(7) CCSID 37 DEFAULT NULL ,
    "DAY" SMALLINT DEFAULT NULL ,
    STARTING TIME DEFAULT NULL ,
    ENDING TIME DEFAULT NULL );

CREATE TABLE ITSCID63.DEPARTMENT (
    DEPTNO CHAR(3) CCSID 37 NOT NULL ,
    DEPTNAME VARCHAR(36) CCSID 37 NOT NULL ,
```



Visual Explain for select country, sum(revenue_wo_tax) from cust_dim left join item_fact on cust_dim.custke...

File View Actions Options Help

Save SQL Monitor Data

Print

Exit

Table Scan

Table Scan

Temporary Index

Temporary Index

Index Scan - Key Selection

Index Scan - Key Positioning

Nested Loop Join

Index - Grouping

Sort

Final Select

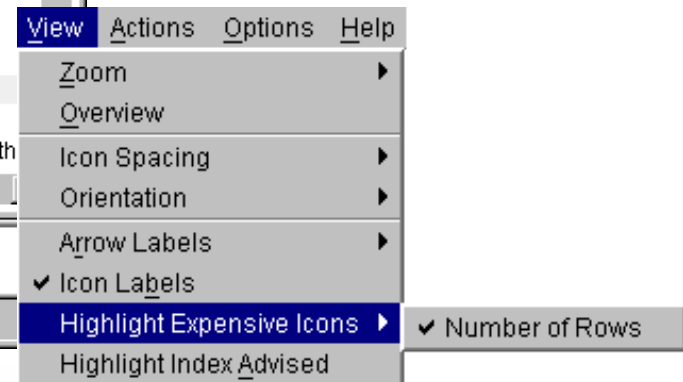
Attribute	Value
Time information (start time, ...)	
Timestamp for Creation of	2000-10-13-15.46.37.5366
Statement Start Timestamp	2000-10-13-15.45.47.6338
Statement End Timestamp	2000-10-13-15.46.37.5364
Optimization Time, in	30
ODP Open Time, in	49485
Total Time, in Microseconds	49902632
Information about SQL	
Statement Number	1
Statement Function	Select
Statement Operation	Select Into
Statement Type	Dynamic
Statement Name	
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	
Package Name	
Package Library	
Rows Returned	0
Rows Fetched	Not Available
Statement Text	select country,
Additional information about	
CLOSQLCSR Value	
ALWCOPYDTA Value	Yes, can use a copy of th

select country, sum(revenue_wo_tax) from cust_dim left join item_fact on cust_dim.custkey = item_fact.custkey group by country order by sum(revenue_wo_tax) desc

Statement text

Enhancements:

- Visualize SQL
- Print
- Save Image Preferences
- Highlight Expensive icons and Index Advised
- View Optimizer messages
- Save VE Performance Monitor Data



View Actions Options Help

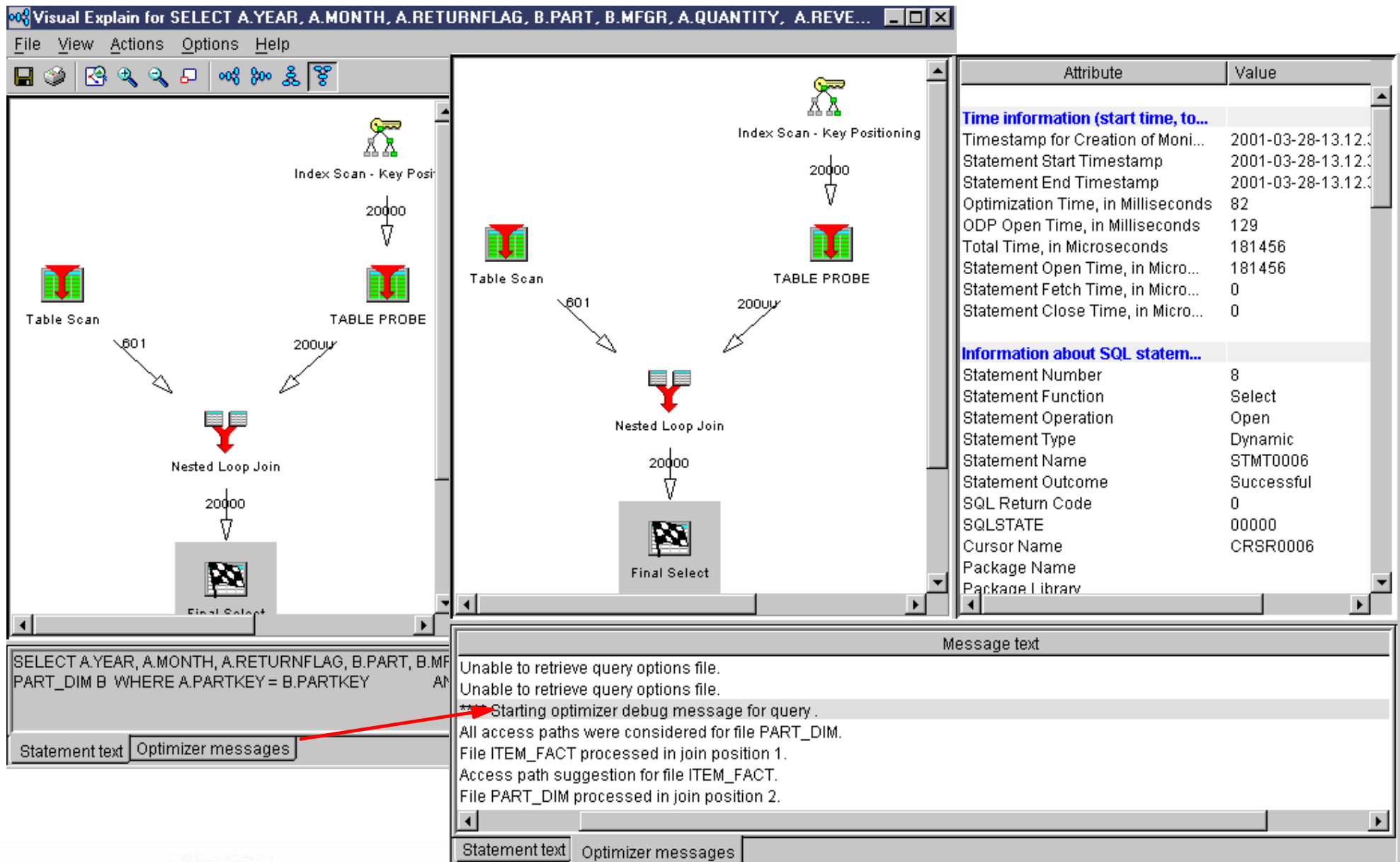
- Zoom
- Overview
- Icon Spacing
- Orientation
- Arrow Labels
- Icon Labels
- Highlight Expensive Icons**
- Highlight Index Advised

Number of Rows

Visual Explain has been enhanced to:

- Include the SQL statement that is being analyzed in same or a new frame
- Allow printing the query implementation graph
- Remember preferences across sessions
- Give a clearer presentation of query attributes and values
- Highlight the most expensive (time and resource consuming) steps
- Highlight advised indexes
- Button to show Optimizer message details (no need to select Include Debug messages in joblog Option)
- Save SQL Performance Monitor data used by Visual Explain

V5R1 Visual Explain, Optimizer Messages



The screenshot displays the Visual Explain interface for a SQL query. The main window shows a query execution plan with the following components:

- Index Scan - Key Positioning**: Feeds into **TABLE PROBE** (20000 rows).
- Table Scan**: Feeds into **Nested Loop Join** (801 rows).
- TABLE PROBE**: Feeds into **Nested Loop Join** (2000U rows).
- Nested Loop Join**: Feeds into **Final Select** (20000 rows).

The right-hand pane shows a table of performance metrics:

Attribute	Value
Time information (start time, to...	
Timestamp for Creation of Moni...	2001-03-28-13.12.3
Statement Start Timestamp	2001-03-28-13.12.3
Statement End Timestamp	2001-03-28-13.12.3
Optimization Time, in Milliseconds	82
ODP Open Time, in Milliseconds	129
Total Time, in Microseconds	181456
Statement Open Time, in Micro...	181456
Statement Fetch Time, in Micro...	0
Statement Close Time, in Micro...	0
Information about SQL statem...	
Statement Number	8
Statement Function	Select
Statement Operation	Open
Statement Type	Dynamic
Statement Name	STMT0006
Statement Outcome	Successful
SQL Return Code	0
SQLSTATE	00000
Cursor Name	CRSR0006
Package Name	
Package Library	

The bottom pane shows the SQL statement and optimizer messages:

```

SELECT A.YEAR, A.MONTH, A.RETURNFLAG, B.PART, B.MFGR, A.QUANTITY, A.REVE...
PART_DIM B WHERE A.PARTKEY = B.PARTKEY
    
```

Optimizer messages:

- Unable to retrieve query options file.
- Unable to retrieve query options file.
- *** Starting optimizer debug message for query.
- All access paths were considered for file PART_DIM.
- File ITEM_FACT processed in join position 1.
- Access path suggestion for file ITEM_FACT.
- File PART_DIM processed in join position 2.

OS/400 Run SQL Scripts Visual Explain was introduced in V4R5. Visual Explain is a graphical depiction of the decisions made by the OS/400 query optimizer support. The output can be used to help analyze performance of the SQL function. In many cases you needed to also run an SQL Performance Monitor or specify to include Optimizer messages in the job log.

You then had to analyze the job log messages in a separate window or look at various SQL Performance Monitor reports to determine, for example if an index should be created that may improve performance.

In V5R1 the Visual Explain interface makes it easier to identify something like a new index recommendation by providing a real time "Optimizer messages" button. Click that button and the lower "message area" is expanded as if you were looking at only the query-related messages, including Optimizer recommendation as shown on the right window of this foil.

You can click on an Optimizer message and get the complete help text as shown on the following foil.

Notes:

- The V5R1 Operations Navigator presentation has some additional pages on database functions.
- V4R5 Visual Explain had some limitation on the complexity level of the SQL statement being explained. V5R1 Visual Explain has most restrictions removed.
- With V5R1, when exiting Visual Explain, you are given the chance to store the SQL performance information collected by Visual Explain into an SQL Performance Monitor without having to define and run an SQL Performance Monitor during a separate session.

Optimizer Message and Explain Attributes

Entire Job

ID	Message text
	Unable to retrieve query options file.
	**** Starting optimizer debug message for query .
	All access paths were considered for file PART_DIM.
	File ITEM_FACT processed in join position 1.
	Access path suggestion for file ITEM_FACT.

Cause : To improve performance the query optimizer is suggesting a permanent access path be built with the key fields it is recommending. The access path will access records from member ITEM_FACT of file ITEM_FACT in library ONLABZZ.

In the list of key fields that follow, the primary key field is the first field. The remaining fields are secondary key fields. The query optimizer is able to process the query whether those key fields should be used. Therefore it is important that the query optimizer is able to process the query with any remaining secondary key fields. The query optimizer is able to process the query with any remaining secondary key fields that are followed by the secondary key fields.

YEAR, MONTH, RETURNFLAG.
If file ITEM_FACT in library ONLABZZ.

Recovery : If this query is re-executed, the query optimizer will choose not to use the index.
For more information, refer to the SQL Reference topic in the Info Center.
File PART_DIM processed in job

Specific Icon

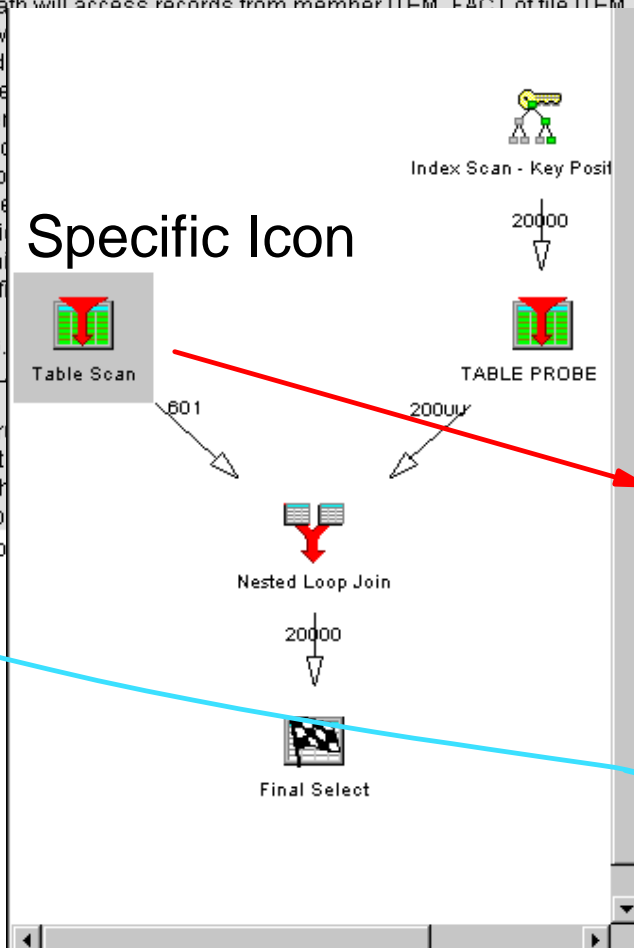


Table name, base table name, i...	
Name of Table Being Queried	ITEM_FACT
Library of Table Being Queried	ONLABZZ
Member of Table Being Queried	ITEM_FACT
Estimated processing time and ...	
Estimated Processing Time	14
Estimated Cumulative Time	14
Total Rows in Table	600572
Table Size	160972800
Estimated rows selected and q...	
Estimated Rows Selected	601
Join Position	1
Table Data Space Number	1
Number of Tables Joined	2
I/O or CPU Bound	CPU Bound
Information about the table sca...	
Data Space Selection	Yes
Derived Selection Performed	No
Skip Sequential Selection	No
Table Scan Reason Code	No indexes exist
Index advised information	
Creation of an Index is Advised	Yes
Number of Primary Key Columns	3
List of Key Columns for Advised...	YEAR, MONTH, RETURNFLAG
Columns for table selection, ke...	
Columns for Data Space Selection	YEAR MONTH RETURNFLAG

This foil shows examples of two different Visual Explain "sets of detail information:"

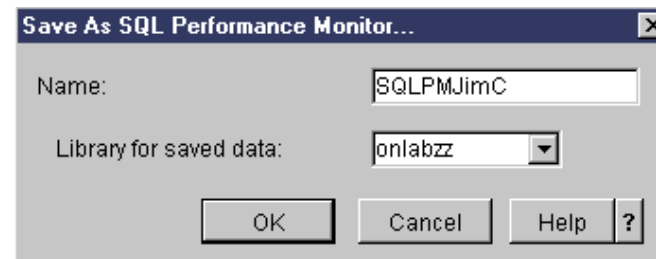
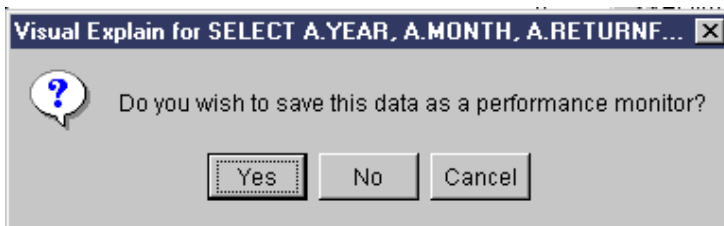
- In the left window we have double clicked the specific "Access path suggestion for file ITEM_FACT" message from the Optimizer messages button list of messages shown on the previous page.
The information shown under Optimizer messages applies to the complete query function.
- In the right window we have clicked over the left Table Scan icon within a Visual Explain window, as indicated by the grayed rectangle. That changed the Attributes shown on the right pane of the Visual Explain window. One of the attributes shown is the **Index advised information**.
The blue arrow shows how the Optimizer message detail text relates to the Index advised information.
The Attributes shown apply only to the Visual Explain icon selected.

These two sets of information can be shown in one V5R1 Visual Explain session.

Saving Visual Explain monitor data

Visual Explain uses SQL performance monitor within the Run SQL Scripts job

- File menu option
- Close message window



Notes: Saving Visual Explain monitor data

When closing the Visual Explain window you can choose to save the SQL Performance Monitor data implicitly collected by Visual Explain.

This monitor data can be analyzed later under Operations Navigator - Database SQL Performance Monitors.

DB2 UDB XML Extender

DB2 UDB Text Extender

IBM @server. For the next generation of e-business.

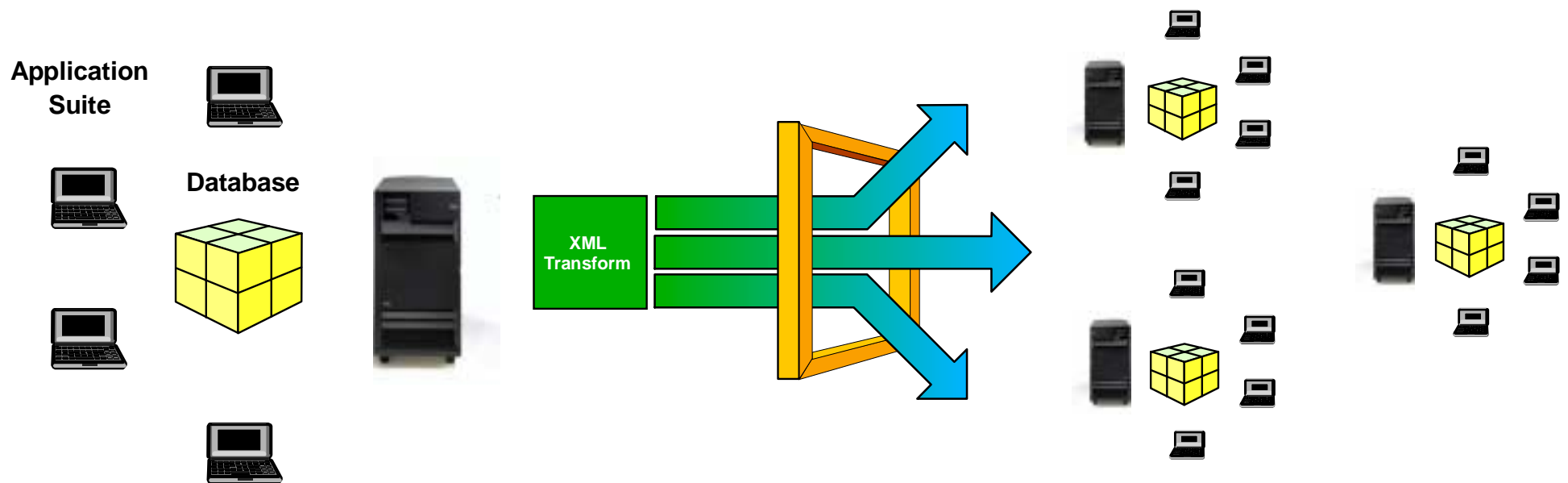
XML, Data and Data Presentation

eXtensible Markup Language:

- Replaces existing proprietary standards
- Allows to interchange data and intent between applications

Content Management is key in B2B application deployment

XML as "natural" extension for data management and interchange



IBM  server. For the next generation of e-business.

Users today have the opportunity to choose the application that suits best the requirements for specific tasks within a number of processes. However, because both processes and tasks need share data, there is a continuous challenge for each component of replicating, transforming, exporting, or saving their data to or from a format that can be imported into another. This can be a critical problem since transformation processes do not necessarily handle cleansed data, or impose consistency of that data.

Applications mostly use open interfaces to handle data and present it in the form in which it is needed for the application. This conversion process is often specific for a specific application requirement; however applications change quickly and tend to become obsolete. In most cases, the presentation of the content is linked to the application and tends to be oriented so that reuse is not always possible, since the level of abstraction is confined to the boundaries of the application. If there were another method that separated data from presentation, this method could be used as a practical form of interchange between applications.

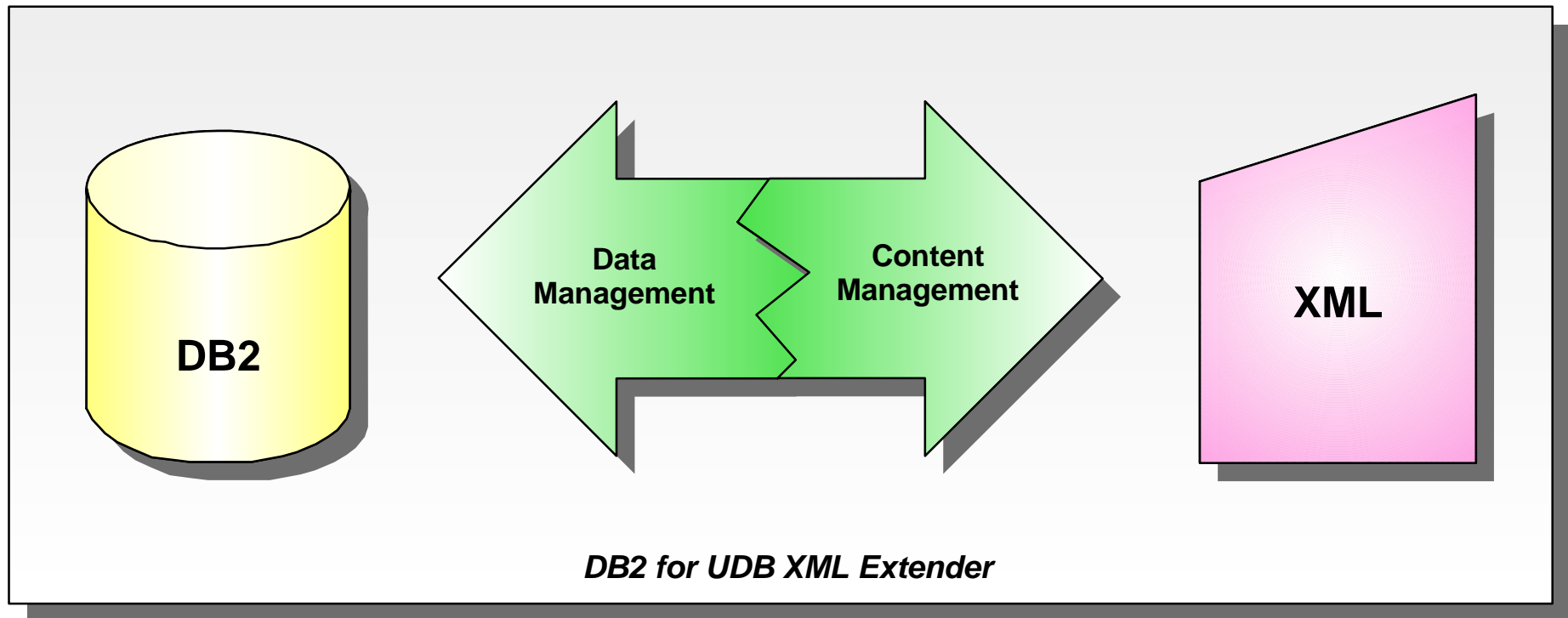
XML has emerged to address this problem. XML is an acronym for eXtensible Markup Language. It is extensible in that the language itself is a metalanguage that allows you to create your own language depending on the needs of your enterprise. You use XML to capture not only the data for your particular application, but also the data structure. XML is not the only interchange format. However, XML has emerged as the accepted standard for data interchange. By adhering to this standard, applications can finally share data without needing to transform data using proprietary formats.

XML therefore answers to the demands of having a standardized representation of data. However, when building applications that use enterprise data, there are many more requirements which need to be answered, for which a data management system is the appropriate tool. XML is therefore presented as an extension to the existing DB2 UDB for iSeries, allowing you to take advantage of the power of DB2 in many XML applications.

Integrating the Power of DB2 and XML

XML delivers base structure for Content Management

DB2 provides base structure for Data Management



By incorporating the XML information and meta-information directly into the database, you can more directly (and more quickly) obtain the XML results that your other applications need for their particular purpose. This is where the XML Extender can assist you. With the XML Extender, you can take advantage of the power of DB2 in many XML applications.

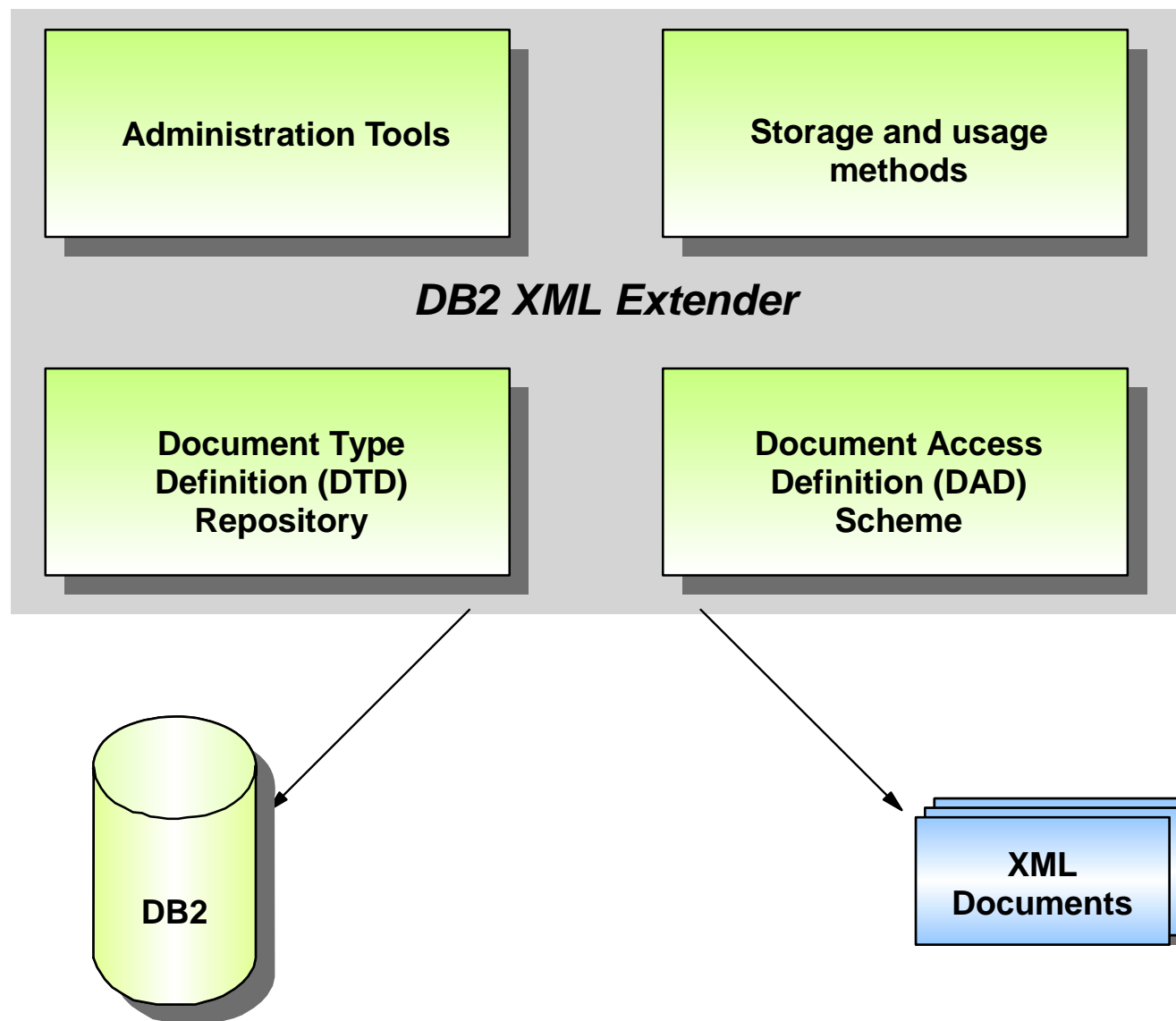
With the content of your structured XML documents in a DB2 database, you can combine structured XML information with your traditional relational data. Based on the application, you can choose whether to store entire XML documents in DB2 as a nontraditional user-defined data type, or you can map the XML content as traditional data in relational tables. For nontraditional XML data types, the XML Extender adds the power to search rich data types of XML element or attribute values, in addition to the structural text search that the DB2 UDB Text Extender provides.

With the XML Extender, your application can:

- Store entire XML documents as *column data* in an application table or externally as a local file, while extracting desired XML element or attribute values into side tables for search. Using the XML column method, you can:
 - Perform fast search on XML elements or attributes of SQL general data types that have been extracted into side tables and indexed.
 - Update the content of an XML element or the value of an XML attribute.
 - Extract XML elements or attributes dynamically using SQL queries.
 - Validate XML documents during insertion and update.
 - Perform structural-text search with the Text Extender.
- Compose or decompose contents of XML documents with one or more relational tables, using the XML *collection storage and access* method.

For a more detailed description of both storage methods, see below.

XML Integration in DB2



Full-text retrieval using SQL queries

Access to both text stored in UDB DB2 and in text stored in files

Uses indexes to search:

- Using linguistic processing
- With a precise index
- With a combination of a linguistic and precise index
- Using an Ngram index

Text Extender adds the power of full-text retrieval to SQL queries in documents embedded in your DB2 tables with a size of up to 2 GB. This feature provides users and application programmers a fast, versatile, and intelligent method of searching through such text documents. Text Extender's strength lies in its ability to search through many thousands of large text documents at high speed, finding not only what you directly ask for, but also word variations and synonyms.

You are not restricted to searching only in text documents stored in DB2 databases, you can also search in text documents stored in files, since Text Extender can access any kind of text document, including word-processing documents in their original native form, and offers a rich set of retrieval capabilities including word, phrase, wild card, and proximity searching using Boolean logic.

At the heart of Text Extender is IBM's high-performance linguistic search technology. It allows your applications to access and retrieve text documents in a variety of ways.

Your applications can:

- Search for documents that contain specific text, synonyms of a word or phrase, or sought-for words in proximity, such as in the same sentence or paragraph.
- Do wild card searches, using front, middle, and end masking, for word and character masking.
- Search for documents of various languages in various document formats.
- Make a "fuzzy" search for words having a similar spelling as the search term. This is useful for finding words even when they are misspelled.
- Make a free-text search in which the search argument is expressed in natural language.
- Search for the names of people, places, or organizations.
- Search for words that sound like the search term.

DB2 Extenders Implementation

Implemented via a New Licensed Program:

- DB2 UDB Extenders for AS/400 , 5722-DE1
- Contains options for both:
 - Text Extenders (option 1)
 - XML Extenders (option 2)

Contains Extender commands and command interface

SQL Triggers

System Triggers

IBM @server. For the next generation of e-business.

V5R1 Trigger Enhancements

Significant additions beyond V4R5 support

V5R1 brings new trigger capabilities for DB2 UDB for iSeries

- SQL Triggers
- Column-level Triggers
 - Not available with System triggers (not written in SQL syntax)
- More than 1 Trigger per database event
 - Triggers for the same event, fired in the order created
- Up to 300 trigger definitions per table allowed

SQL trigger support (that is, CREATE TRIGGER) offers a more standard and portable solution

DB2 UDB for AS/400 SQL Trigger implementation is a superset of the DB2 UDB v7.1 SQL Triggers

- Adds procedural logic (IF test, WHILE loops, etc.) to the DB2 UDB trigger implementation

SQL Triggers provide additional capabilities over external triggers

- Column-level granularity
- Statement and Row Level triggers

DB2 UDB for AS/400 SQL Triggers require the DB2 UDB SQL Development Kit to be installed on development system

What is a trigger?

A trigger is a set of actions that are run automatically when a specified change operation is performed on a specified table. The change operation can be an SQL INSERT, UPDATE, or DELETE statement, or an insert, update, or delete high level language statement in an application program. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

Triggers were announced back in V3R1 and they could be coded in any HLL (such as RPG, COBOL and C). In V5R1 we are announcing the support of SQL Triggers.

SQL trigger support (i.e., CREATE TRIGGER) offers a more standard and portable solution.

DB2 UDB for AS/400 SQL Trigger implementation is a superset of the DB2 UDB v7.1 SQL Triggers

- Adds procedural logic (IF test, WHILE loops, etc.) to the DB2 UDB trigger implementation

SQL Triggers provide additional capabilities over system triggers, including:

- Column-level granularity
- Statement and Row Level triggers

There is a new capability announced for system triggers and it is the new AFTER READ event trigger. There are some performance considerations that must be followed before you decide to use this type of triggers.

Prior to V5R1, the maximum number of triggers per table was 6:

- Before Delete trigger
- Before Insert trigger
- Before Update trigger
- After Delete trigger
- After Insert trigger
- After Update trigger

This created a problem when two different applications required to fire a trigger on the same event.

In V5R1 this limit has been raised to 300 and the triggers are fired in creation timestamp order. Since, multiple triggers can be specified for a combination of table, event, or activation time. The order in which the triggers are activated is the same as the order in which they were created. Thus, the most recently created trigger will be the last trigger activated.

Once a trigger is associated with a table, the trigger support calls the trigger program whenever a change operation is initiated against the table, or any logical file or view created over the table. SQL triggers and system triggers can be defined for the same table.

Good news!!

To create V5R1 SQL triggers, DB2 UDB SQL Development Kit (ST1) needs to be installed. ILE C compiler (CX2) is no longer needed. For run-time support of SQL triggers, only OS/400 is needed. This is also true for Stored Procedures.

- Triggering Event/Time: BEFORE/AFTER DELETE, INSERT, UPDATE
 - . UPDATE allows for column-level: `AFTER UPDATE of empid, salary ON emptbl`
- Granularity: FOR EACH STATEMENT versus FOR EACH ROW
 - If statements operates on table, but modifies zero rows - row level triggers would not be called, but statement level trigger would be called
 - Statement level triggers not supported on BEFORE triggers
 - Statement & Row-level trigger equivalent for non-SQL interfaces
- Correlation variables & Transition tables - similar to trigger buffer
 - OLD and NEW correlation variables - values of triggering row before and after the trigger
 - `REFERENCING OLD AS oldrow REFERENCING NEW AS newrow`
 - `... NEW.salary > OLD.salary +100000 ...`
 - OLD & NEW transition tables - temporary table containing all of the affected rows before and after the trigger execution
 - `REFERENCING OLD_TABLE AS otbl`
 - `... (SELECT COUNT(*) FROM otbl) ...`

There are a number of criteria that are defined when creating a trigger which are used to determine when a trigger should be activated:

- The **subject table** defines the table for which the trigger is defined.
- The **trigger event** defines a specific SQL operation that modifies the subject table. The operation could be delete, insert, or update.
- The trigger **activation time** defines whether the trigger should be activated before or after the trigger event is performed on the subject table.

A table can be associated with six types of triggers:

- Before delete trigger
- Before insert trigger
- Before update trigger: The update trigger supports defining a trigger at a column level.
- After delete trigger
- After insert trigger
- After update trigger

The **trigger granularity** defines whether the actions of the trigger will be performed once for the statement or once for each of the rows in the set of affected rows.

SQL Correlation Variables:

The triggered action may refer to the values in the set of affected rows. This concept is very similar to the concept of the trigger buffer used in the system triggers. In SQL triggers this is supported through the use of transition variables or correlation variables. Transition variables use the names of the columns in the subject table qualified by a specified name that identifies whether the reference is to the old value (prior to the update) or the new value (after the update). The new value can also be changed using the SET transition-variable statement in before update or insert triggers.

SQL Transition Tables:

An SQL trigger may need to refer to all of the affected rows for an SQL insert, update, or delete operation. This is true, for example, if the trigger needs to apply aggregate functions, such as MIN or MAX, to a specific column of the affected rows. The OLD_TABLE and NEW_TABLE transition tables can be used for this purpose.

In this example:

- Transition table variable - if update statement changes 5 rows, the transition table would contain 5 rows even if the underlying table contains 50 rows.

- Trigger condition - WHEN similar to WHERE clause, controls execution of trigger body
 - WHEN (newrow.salary <> oldrow.salary)
 - WHEN (newrow.salary > SELECT max(salary) FROM emp WHERE jobcode=22)
- Trigger Mode
 - MODE DB2ROW
 - ▶ Trigger fired after each row operation
 - ▶ Only allowed on Row-level triggers
 - ▶ Not available in other DB2 UDB implementations
 - MODE DB2SQL
 - ▶ Trigger fires after all row operations. If specified on a Row-level trigger, then trigger called N times after all of the row operations completed
 - ▶ Only allowed on After triggers
 - ▶ Not as efficient as DB2ROW since each row is processed twice
- Trigger body (BEGIN...END)

WHEN condition:

The **WHEN** condition can be used in an SQL trigger to specify a condition. If the condition evaluates to true, then the SQL statements in the SQL trigger routine body are executed. If the condition evaluates to false, the SQL statements in the SQL trigger routine body are not executed, and control is returned to the database system.

There are two trigger modes:

DB2SQL : Triggers are activated after all of the row operations have occurred.

DB2ROW : Triggers are activated on each row operation. This mode is valid for both the **BEFORE** and **AFTER** activation time.

In the body of the trigger which starts with the **BEGIN** clause and it ends with the **END** clause you can code the following SQL statements:

- * SQL procedure statement
- * SQL control statements
- * assignment-statement
- * call-statement
- * case-statement
- * compound-statement
- * if-statement
- * for-statement
- * get-diagnostics-statement
- * goto-statement
- * leave-statement
- * loop-statement
- * repeat-statement
- * resignal-statement
- * return-statement
- * signal-statement
- * while-statement

In this example:

- The transition tables are implemented as temporary tables in QTEMP. The temporary files contain a copy of the rows that were either inserted, updated or deleted.

- Some of the statement types available for the Trigger body:
 - DECLARE local variables
 - SET local variables
 - IF, CASE
 - WHILE, FOR, REPEAT, LOOP
 - DECLARE CONDITION
 - DECLARE HANDLER
 - SIGNAL, RESIGNAL
 - GET DIAGNOSTICS
 - ▶ Provides access to SQLCA-like information
 - CALL
 - ▶ Call external procedures to access HLL programs or OS/400 APIs
 - Normal DDL & DML (CREATE, INSERT, DELETE, DROP, etc.)

In the body of the trigger, which starts with the BEGIN clause and it ends with the END clause, you can code the following SQL statements:

- * SQL procedure statement
- * SQL control statements
- * assignment-statement
- * call-statement
- * case-statement
- * compound-statement
- * if-statement
- * for-statement
- * get-diagnostics-statement
- * goto-statement
- * leave-statement
- * loop-statement
- * repeat-statement
- * resignal-statement
- * return-statement
- * signal-statement
- * while-statement

SQL Triggers - Example

```
CREATE TRIGGER big_spenders
AFTER INSERT ON expenses
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2ROW
WHEN (n.totalamount > 10000)
BEGIN

    DECLARE empname CHAR(30);

    SET empname = (SELECT Iname FROM employee
                   WHERE empid=n.empno);

    INSERT INTO travel_audit
        VALUES(n.empno, empname, n.deptno, n.totalamount, n.enddate);

END
```

Notes: Trigger Example

On this example the trigger name is big_spenders:

```
CREATE TRIGGER big_spenders
```

It is an after trigger defined for the table expenses:

```
AFTER INSERT ON expenses
```

It will be executed when totalamount>10,000

```
WHEN(n.totalamount > 10000)
```

The logic of the trigger begins with the clause: BEGIN

The trigger writes into an audit file called travel_audit the information concerning the employee that its expenses are over 10,000. -->

```
WHEN (n.totalamount > 10000)
```

```
BEGIN
```

```
DECLARE emplname CHAR(30);
```

```
SET emplname = (SELECT Iname FROM employee  
WHERE empid=n.empno);
```

```
INSERT INTO travel_audit
```

```
VALUES(n.empno, emplname, n.deptno, n.totalamount, n.enddate);
```

```
END
```

- Additional Considerations

- All of the transition & correlation variables are not available to each Trigger type
 - ▶ Statement Triggers can only use the transition table variables
 - ▶ Update Triggers can use both old and new transition & correlation variables
 - ▶ Insert Triggers can only access new transition & correlation variables
 - ▶ Delete Triggers can only access old transition & correlation variables
- SET statement can be used by BEFORE Triggers to alter the new data
 - ▶ Generate missing values
 - ▶ Data cleansing
 - ▶ One of the **FEW** ways to change data in a Before trigger
 - New correlation variables can also be changed with SELECT INTO and as an output variable (OUT/INOUT) of a stored procedure

An SQL trigger is created as a program(*PGM) object using the CRTSQLCI CL command. The program is created in the collection specified by the trigger name qualifier. The specified trigger is registered in the SYSTRIGGERS, SYSTRIGDEP, SYSTRIGCOL, and SYSTRGUPD SQL

Note: Even if a Before Trigger calls a stored procedure, the SQL in the stored procedure will not be allowed to change the data (Native I/O interface would be allowed to change data).

SQL Trigger Examples

```
CREATE TRIGGER auditspending  
BEFORE UPDATE ON expenses  
REFERENCING NEW AS n  
FOR EACH ROW MODE DB2ROW  
WHEN (n.totalamount > 10000)  
    INSERT INTO travel_audit  
        VALUES(n.empno, n.deptno, n.totalamount, n.enddate);
```

```
CREATE TRIGGER empsal  
BEFORE UPDATE OF salary ON emp  
REFERENCING NEW AS n OLD AS o  
FOR EACH ROW MODE DB2ROW  
WHEN (n.salary > 1.5 * o.salary)  
    SET n.salary = 1.5 * o.salary;
```

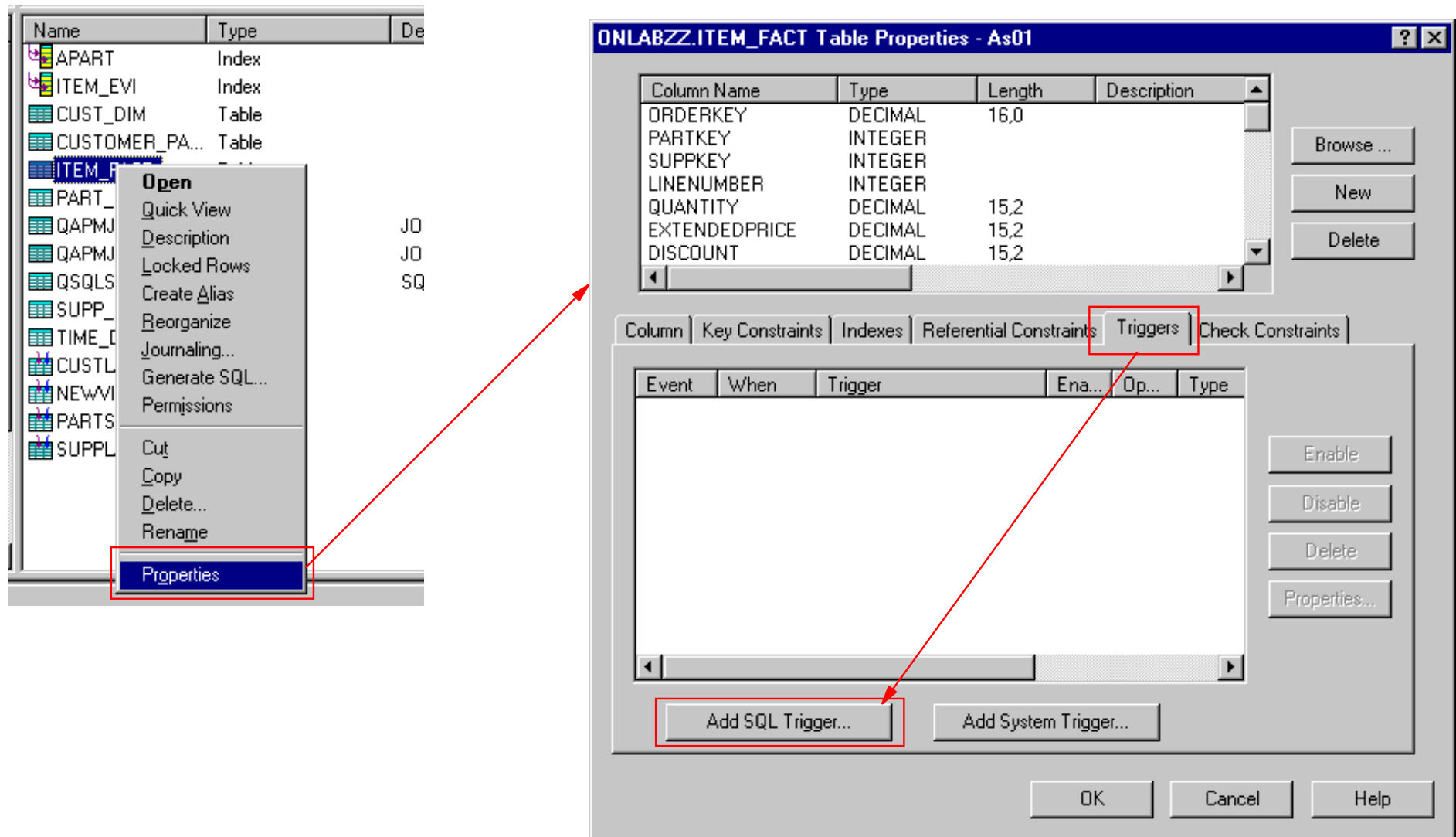
SQL Triggers and Operations Navigator

IBM @server. For the next generation of e-business.

V5R1 Operations Navigator - Database support adds much of the new SQL Trigger support.

The next foils give an overview of this new Trigger support.

Add SQL Trigger Interface



The screenshot illustrates the process of adding a SQL trigger to a table in the IBM iSeries database interface. On the left, a table browser shows a list of tables and indexes. The 'ITEM_F' table is selected, and a context menu is open with the 'Properties' option highlighted. A red arrow points from this 'Properties' option to the 'ONLABZZ.ITEM_FACT Table Properties - As01' dialog box on the right.

The 'ONLABZZ.ITEM_FACT Table Properties - As01' dialog box displays the table's structure in a table format:

Column Name	Type	Length	Description
ORDERKEY	DECIMAL	16,0	
PARTKEY	INTEGER		
SUPPKEY	INTEGER		
LINENUMBER	INTEGER		
QUANTITY	DECIMAL	15,2	
EXTENDEDPRICE	DECIMAL	15,2	
DISCOUNT	DECIMAL	15,2	

Below the table structure, the 'Triggers' tab is selected. It contains a table with columns: Event, When, Trigger, Ena..., Op..., and Type. The 'Add SQL Trigger...' button is highlighted with a red box. A red arrow points from this button to the 'Triggers' tab. On the right side of the dialog, there are buttons for 'Enable', 'Disable', 'Delete', and 'Properties...'. At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

Notes: Add SQL Trigger Interface

An SQL trigger can be created by either specifying the CREATE TRIGGER SQL statement or by using Operations Navigator. The statements in the routine-body of the SQL trigger are transformed by SQL into a program. An SQL trigger is created as a program(*PGM) object using the CRTSQLCI CL command. The program is created in the collection specified by the trigger name qualifier. The specified trigger is registered in the SYSTRIGGERS, SYSTRIGDEP, SYSTRIGCOL, and SYSTRGUPD SQL Catalogs. In this overview, we will have a closer look at the Operations Navigator interface.

In the *Table Properties* panel, a new dialogue appears when the *Triggers* tab is selected. If there are triggers defined for this table they will be displayed in the list box in the order of Event, When, Row, Statement, and Creation date and time order. The creation date and time order is the actual firing order of the triggers within the Event, When, Row, and Statement and context. Statement level triggers are not allowed in the Before context, but will be in the After context if defined. Note that grouping by Event will be done when clicking on the Event column name, and grouping by When will be done by clicking on the When column name.

Column	Description	Possible Values	Show by Default
Event	Operation that causes the trigger program to run.	Insert, Delete, or Update	Yes
When	When is the trigger to execute in relation to the event that is causing it.	Before or After	Yes
Trigger	Name of the trigger		Yes
Library	Library where the trigger resides		Yes
Operative	If the trigger is not operative, the table it is defined on will not be able to be opened.	Yes or No	Yes
Type	The type of trigger	Program, SQL Row, or SQL Statement	Yes

Add SQL Trigger Interface

Add SQL Trigger for Table ONLABZZ.ITEM_FACT - As01

General | Timing | SQL Statements

Trigger:

Library:

Description:

Event:

- Insert
- Delete
- Update
- Update of selected columns

Available columns:

- ORDERKEY
- PARTKEY
- SUPPKEY
- LINENUMBER
- EXTENDEDPRICE
- DISCOUNT
- TAX
- RETURNFLAG

Selected columns:

- QUANTITY

Buttons: Add -->, Add All -->>, Remove <--, Remove All <<--

Buttons: OK, Cancel, Help, ?

The *General* tab of the Create SQL Trigger panel will ask you to identify:

- The **trigger name**, which is a required entry. It has to be unique within the library and has a maximum length of 128 characters.
- When you select an **event**, you can select *Insert, Delete, Update or Update of selected columns*. If you choose *Update of selected columns*, you also need to select the columns that cause the trigger to run. Select the column name from the list of available columns and click *Add* to add to add a column or select the column name from the selected columns and click *Remove* if you want to remove it. If you are adding most of the columns, select **Add all** and then remove the ones you want. To start over, select **Remove all** and then begin again.

Add SQL Trigger Interface

Add SQL Trigger for Table ONLABZZ.ITEM_FACT - As01

General | Timing | SQL Statements

When to run:

- Before event
- After event

Run trigger:

- For each row
 - Correlation name for old row: Not specified
 - Correlation name for new row: Not specified
 - Mode:
 - DB2ROW
 - DB2SQL
 - Temporary name for old table: Not specified
 - Temporary name for new table: Not specified
- Once for the statement
 - Temporary name for old table: Not specified
 - Temporary name for new table: Not specified

OK Cancel Help ?

The *Iming* tab of the Create SQL Trigger panel will ask you to identify:

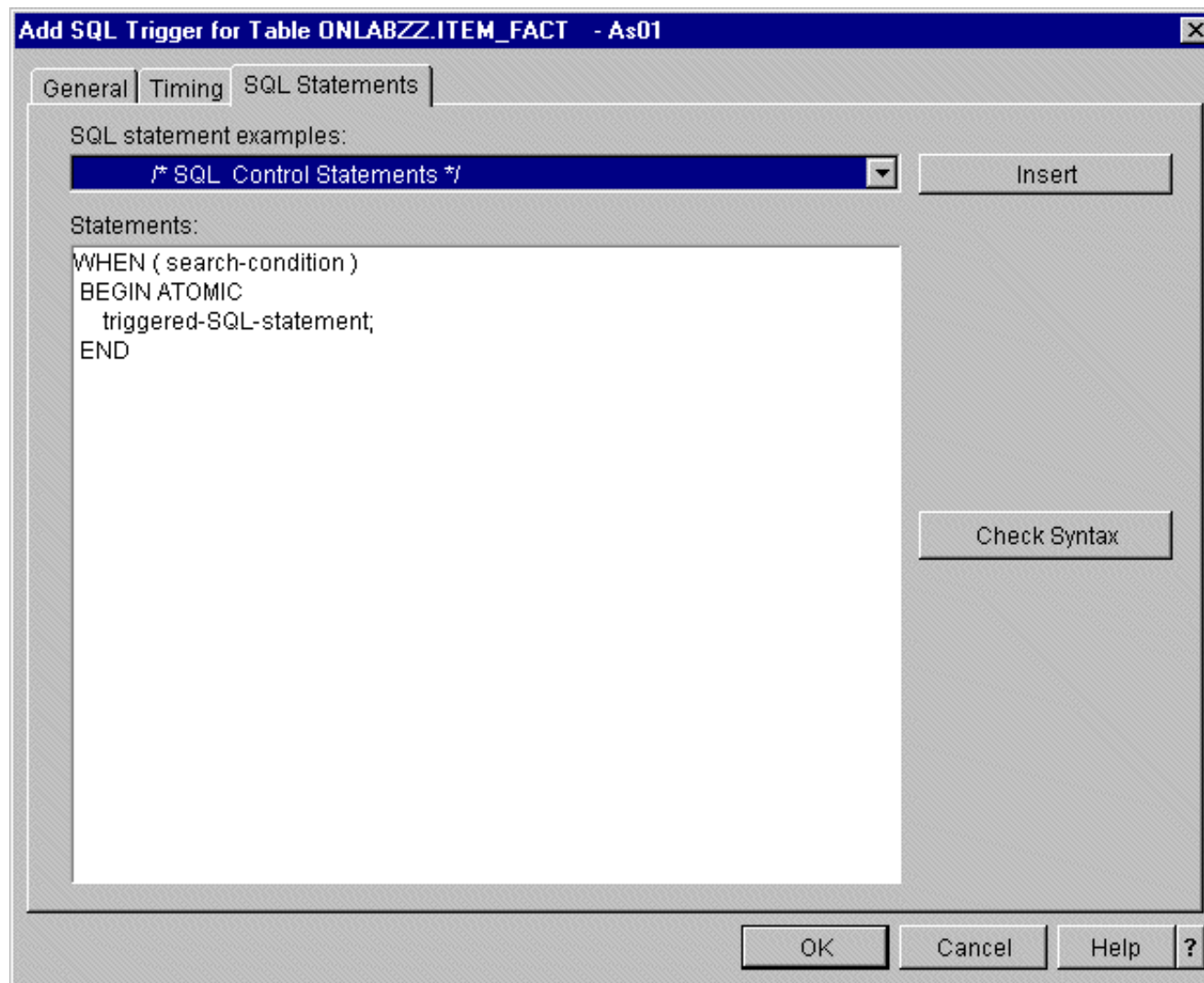
- When to run (Before or After).
- The Run trigger info requires a selection of:
 - FOR EACH ROW, specifying that the database manager executes the triggered-action for each row of the subject table that the triggering operation modifies. If the triggering operation does not modify any rows, the triggered-action is not executed.
 - FOR EACH STATEMENT, specifying that the database manager executes the triggered-action only once for the triggering operation. An UPDATE or DELETE FOR EACH STATEMENT trigger is activated even when no rows are affected by the triggering UPDATE or DELETE statement.
 - OLD ROW AS correlation-name specifies a correlation name that identifies the values in the row prior to the triggering SQL operation. NEW ROW AS correlation-name specifies a correlation name which identifies the values in the row as modified by the triggering SQL operation and any SET statement in a BEFORE trigger that has already executed.
 - OLD TABLE AS specifies the name of a temporary table that identifies the values in the complete set of affected rows prior to the triggering SQL operation. The OLD TABLE includes the rows that were affected by the trigger if the current activation of the trigger was caused by statements in the SQL-trigger-body of a trigger.
 - NEW TABLE AS specifies the name of a temporary table that identifies the state of the complete set of affected rows as modified by the triggering SQL operation and by any SET statement in a BEFORE trigger that has already been executed.
 - The MODE parameter specifies when the triggers are being activated: MODE DB2SQL triggers are activated after all of the row operations have occurred, while MODE DB2ROW triggers are activated on each row operation.

The table on the next page shows when one can specify which attribute.

Notes: Correlation - Summary

Granularity	MODE	Activation Time	Triggering Operation	Correlation Variables Allowed	Transition Tables Allowed
FOR EACH ROW	DB2ROW	BEFORE	INSERT	NEW	NONE
			UPDATE	OLD, NEW	
			DELETE	OLD	
		AFTER	INSERT	NEW	
			UPDATE	OLD, NEW	
			DELETE	OLD	
	DB2SQL	AFTER	INSERT	NEW	NEW_TABLE
			UPDATE	OLD, NEW	OLD_TABLE NEW_TABLE
			DELETE	OLD	OLD_TABLE
FOR EACH STATEMENT	DB2SQL	AFTER	INSERT	NONE	NEW_TABLE
			UPDATE		OLD_TABLE NEW_TABLE
			DELETE		OLD_TABLE

Add SQL Trigger Interface



The SQL Statements page contains the code for the SQL program that you are defining as a trigger. You can use the SQL statement examples and fill in the necessary information to make coding SQL easier. If you are adding a trigger to an existing table, you can check for syntax errors by clicking *Check Syntax* once you have the statement defined. A message is displayed for the first error detected, if any. To check for additional errors, click *Check Syntax* after the first error is fixed. This button is disabled when you are adding a trigger to a new table.

After an SQL trigger has been created, the SQL statements cannot be changed. You will have to delete and recreate the trigger to change the SQL.

The SQL-routine-body, or the executable part of the trigger that is transformed by the database manager into a program. When an SQL trigger is created, SQL creates a temporary source file (QTEMP/QSQLSRC) that will contain C source code with embedded SQL statements; an SQL trigger is created as a program (*PGM) object using the CRTSQLCI and CRTPGM commands.

Remark: To create SQL triggers, you need the 5722-ST1 DB2 Query Mgr and SQL Development Kit. There are no runtime requirements, besides OS/400.

Allows for *READ event

Up to 300 trigger definitions per table allowed

Add Physical File Trigger (ADDPFTRG) command now allows specifying the trigger name (TRG parameter)

*Caution. implementation of a read operation trigger could cause significant performance as currently blocking of rows is not possible and one or more temporary tables are created

A trigger for a read event can now be specified via the system defined triggers. As with SQL Triggers, a maximum of 300 triggers per physical file can be specified. For documentation purposes, the triggers can be named with a description.

Add a System Trigger

Add System Triggers for Table ONLABZZ.ITEM_FACT - As80

General Events

Program name:

Library:

Description:

Allow repeated changes to a row

Is the trigger program thread-safe?

Unknown
 Yes
 No

In a multi-threaded job:

Let system value QMLTTHDACN determine whether to run the trigger program
 Do not run the trigger program
 Run the trigger program and send a diagnostic message
 Run the trigger program

Add System Triggers for Table ONLABZZ.ITEM_FACT - As80

General Events

	Trigger Name	Trigger Library
<input type="checkbox"/> Insert before	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
<input type="checkbox"/> Insert after	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
<input type="checkbox"/> Update before	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
	<input checked="" type="radio"/> Always update	<input type="radio"/> Only update when changed
<input type="checkbox"/> Update after	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
	<input checked="" type="radio"/> Always update	<input type="radio"/> Only update when changed
<input type="checkbox"/> Delete before	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
<input type="checkbox"/> Delete after	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>
<input type="checkbox"/> Read after	<input type="text" value="System-generated"/>	<input type="text" value="ONLABZZ"/>

OK Cancel Help ?

Notes: Add a System Trigger

This foil shows the modifications to the system trigger definitions as they are externalized via the Operations Navigator interface. On the **General** page, you can specify the description, while the **Event** page allows to specify the READ event definition.

General DB2 Enhancements

IBM @server. For the next generation of e-business.

CREATE TABLE

Support for LIKE predicate

Allows to inherit the implicit column definitions of another table

```
CREATE TABLE Accounting.Invoices_detail  
    LIKE Accounting.Invoices_header
```

Specifies that the columns defined in the specified table or view are included in this table. The table-name or view-name specified in a LIKE clause must identify the table or view that already exists at the server.

The use of LIKE is an implicit definition of n columns, where n is the number of columns in the identified table or view. The implicit definition includes the following attributes of the n columns (if applicable to the data type):

- Column name (and system column name)
- Data type, length, precision, and scale
- CCSID
- Column text

If the LIKE clause is specified immediately following the table-name and not enclosed in parenthesis, the following column attributes are also included, otherwise they are not included:

- Default value, if a table-name is specified (view-name is not specified)
- Nullability

If the specified table or view is a non-SQL created physical file or logical file, any non-SQL attributes are removed. For example, the date and time format will be changed to ISO.

The implicit definition does not include any other optional attributes of the identified table or view. For example, the new table does not automatically include a primary key or foreign key from a table. The new table has these and other optional attributes only if the optional clauses are explicitly specified.

The 5250 interface for SQL and the GUI interface from Operations Navigator do not support prompting for the CREATE TABLE LIKE command.

Support for:

- Expressions
- Escape characters

Performance enhancements:

- Can generate reusable ODP
- Use an index for double byte patterns

Example:

```
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 || '%'
```

In the LIKE predicate, you can now specify an ESCAPE pattern, if the pattern-expression needs to include either the underscore or the percent character. In this case, the escape-expression is used to specify a character to precede either the underscore or percent character in the pattern. The following rules need to be observed:

- The escape-expression must be a string of length 1.
- The pattern-expression must not contain the escape character except when followed by the escape character, percent, or underscore. For example, if '+' is the escape character, any occurrences of '+' other than '++', '+_', or '+%' in the pattern-expression is an error.
- The escape-expression can be a parameter marker.

Example

```
SELECT *  
FROM TABLEY  
WHERE C1 LIKE 'AAAA+%BBB%' ESCAPE '+'
```

'+' is the escape character and indicates that the search is for a string that starts with 'AAAA%BBB'. The '+%' is interpreted as a single occurrence of '%' in the pattern.

The instructions using the LIKE predicate will now generate a reusable ODP. This will enhance performance of the SELECT statements being executed. If using double byte encoding, an index can be used also.

JOIN Functionality

Support for RIGHT OUTER JOIN

Allow OR, LIKE, IS NULL, BETWEEN in OUTER JOIN

A right outer join will return all the rows that an inner join returns plus one row for each of the other rows in the second table that did not have a match in the first table. Example:

```
SELECT EMPNO, LASTNAME, PROJNO
       FROM CORPDATA.PROJECT RIGHT OUTER JOIN CORPDATA.EMPLOYEE ON EMPNO = RESPEMP
       WHERE LASTNAME > 'S'
```

Join condition can now contain basic predicates, BETWEEN predicates, IN (list form) predicates, and LIKE predicates. Both AND and OR can be used in the join-condition.

Examples of SQL JOIN syntax enhancements:

```
SELECT * FROM T1 LEFT OUTER JOIN T2
       ON T1.F1 =T2.F1 OR T1.F2 = T2.F2
SELECT * FROM T1 LEFT OUTER JOIN T2
       ON T1.F1 LIKE 'ABC%' AND T2.F1 LIKE 'ABC%'
```

New Scalar Functions:

- TIMESTAMPDIFF
- PI
- SPACE
- GRAPHIC
- MIDNIGHT_SECONDS
- JULIAN_DAY
- DAYOFWEEK_ISO
- WEEK_ISO

The following support for scalar functions has been introduced:

- The **TIMESTAMPDIFF** function returns an estimated number of intervals of the type defined by the first argument, based on the difference between two timestamps. The difference can be expressed in fractions of a seconds, seconds, minutes, hours, days, weeks, months, quarters or years. The following assumptions may be used in estimating the difference: 365 days in a year, 30 days in a month, 24 hours in a day, 60 minutes in an hour and 60 seconds in a minute. These assumptions are used when converting the information in the second argument, which is a timestamp duration, to the interval type specified in the first argument. The returned estimate may vary by a number of days. For example, if the number of days (interval 16) is requested for a difference in timestamps for '1997-03-01-00.00.00' and '1997-02-01-00.00.00', the result is 30. This is because the difference between the timestamps is 1 month so the assumption of 30 days in a month applies.
- **PI**: Returns the value of PI 3.141592653589793. There are no arguments. The result of the function is double-precision floating-point and cannot be null.
- The **SPACE** function returns a character string that consists of the number of SBCS blanks that the argument specifies.
- The **GRAPHIC** function returns a graphic string representation of a string expression. The result of the function is a fixed-length graphic string.
- The **MIDNIGHT_SECONDS** function returns an integer value in the range 0 to 86 400 representing the number of seconds between midnight and the time value specified in the argument.
- The **JULIAN_DAY** function returns an integer value representing a number of days from January 1, 4712 B.C. (the start of the Julian date calendar) to the date specified in the argument.
- The **DAYOFWEEK_ISO** function returns an integer between 1 and 7 that represents the day of the week, where 1 is Monday and 7 is Sunday.
- The **WEEK_ISO** function returns an integer between 1 and 53 which represents the week of the year. The week starts with Monday. Week 1 is the first week of the year to contain a Thursday, which is equivalent to the first week containing January 4. Thus, it is possible to have up to 3 days at the beginning of the year appear as the last week of the previous year or to have up to 3 days at the end of a year appear as the first week of the next year.

FETCH FIRST N ROWS ONLY

Allows to specify number of rows to be retrieved using SELECT

Limits transport of large amounts of data in Client/Server applications

If **N** is omitted, only first row is returned

Journaling Related Enhancements

IBM  server iSeries

Journal Minimal Data

Apply Journal DDL

IBM  server. For the next generation of e-business.

Notes: Journaling Related Enhancements

IBM  server iSeries

OS/400 has several journaling enhancements such as the ones listed here. See the Availability presentation for more information.

IBM  server. For the next generation of e-business.

2 Gigabyte LOBs (old limit 15 MB)

Greater than 80 character embedded SQL statements for C & C++ precompilers

Result Sets for Java Stored Procedures

Java User-Defined Functions

ODBC Version 3.5 Support

- Improved transaction integrity with Microsoft Transaction Server
- Unicode supported

The maximum size of large objects stored in a column is increased from 15 MB to 2 GB and the maximum total size for all large objects for a table row is increased from 1.5 MB to 3.5 GB. The size of a single non-distributed table is also increased to 1 TB. In addition, DB2 UDB for iSeries supports the ability to optionally minimize the size of journal entries.

For B2B between iSeries and Microsoft Windows clients, the ODBC driver for DB2 UDB for iSeries is enhanced with ODBC 3.5 support and support for Microsoft Transaction Server (MTS). MTS support enables DB2 UDB for iSeries to participate in transactions involving two-phase commit coordinated through MTS. ODBC 3.5 support also delivers support for Unicode.

DRDA Result Set Support

DRDA Distributed Unit of Work (2-Phase commit) over IP

RUNSQLSTM now part of OS/400 and ILE C compiler eliminated for SQL Procedures, Functions, & Triggers

System-supplied stored procedure for creating sample **database**

- (CALL QSYS/CREATE_SQL_SAMPLE(**schema-name**))

DRDA Result Set allows OS/400 to serve multiple DRDA clients with UDB DB2. This new support allows a client, probably another UDB2 system, to use Stored Procedures that return result sets when connecting via DRDA. Prior to V5R1, in order to use result sets, you had to use the Client Access ODBC driver. Now other applications using DRDA drivers can use result sets when call stored procedures, providing the driver on the client also has this support.

Support that allows the AS/400 to act as a client (DRDA Requester) is not being added in V5R1, but will be added in a future release.

Create Schema is a system-supplied procedure for creating sample database

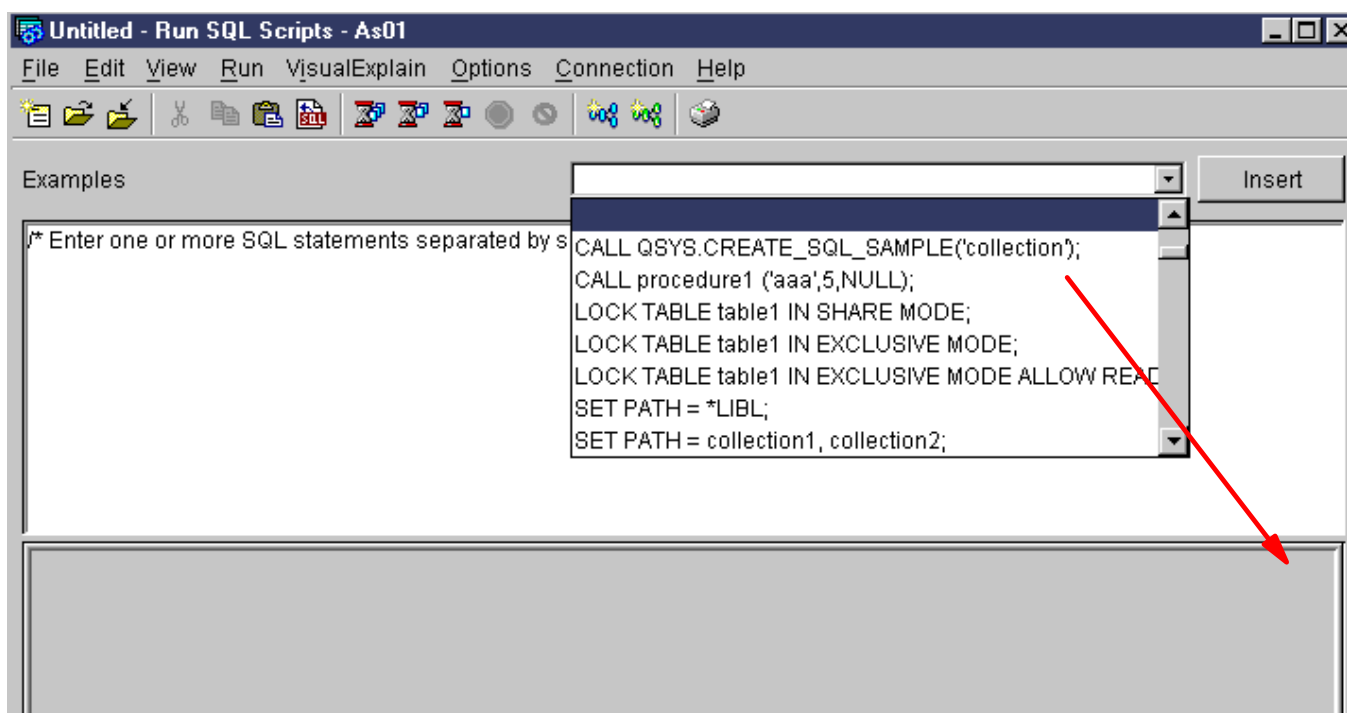
With V5R1 stored procedure Create_SQL_Sample is provided, similarly to that facility on non-AS/400 database products. If the user calls this procedure, many DB/SQL objects are created and stored in the schema (OS/400 library/collection name) named on the CALL.

The industry now calls an SQL collection a schema.

The set of objects generated include, tables, views, and indexes. This set of objects can be used under Operations Navigator to explore and "test out" new database capabilities included in this release.

Create Schema Example

System-supplied stored procedure for creating sample **schema**
(collection/database)



Notes: Create Schema Example

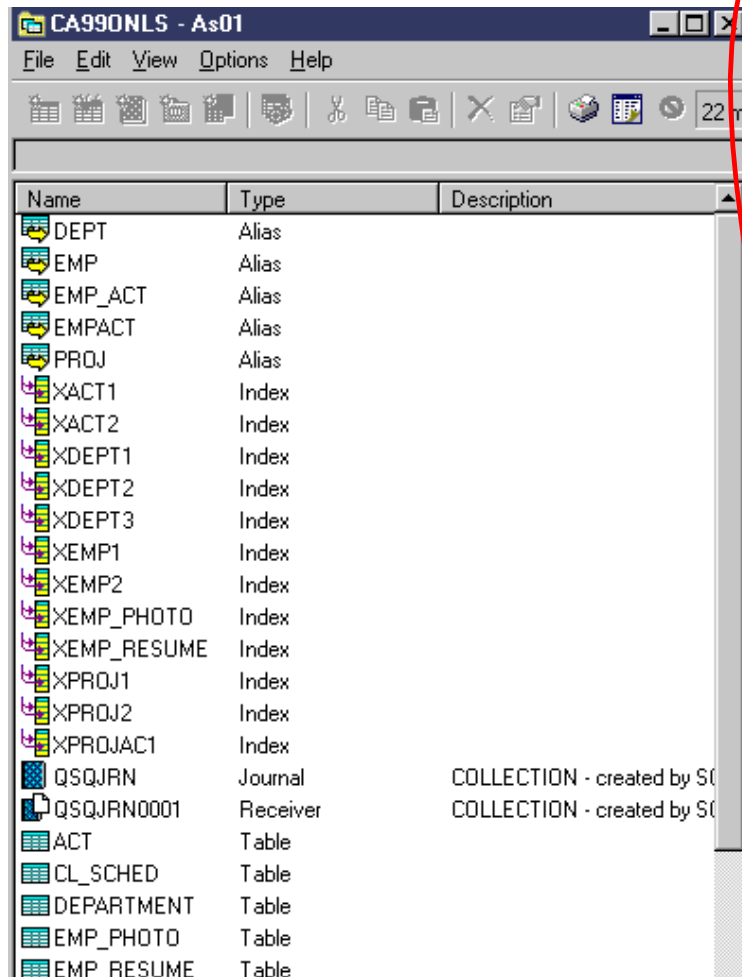
In this example to create a sample schema, complete with tables views and indexes, we use the Operations Navigator Run SQL Scripts (Script Center). We select the list of provided SQL statements from the list window to the left of the Insert button. Scrolling down the list we find the CALL QSYS.CREATE_SQL_SAMPLE statement and click the Insert button. This places the SQL statement into the Run SQL Scripts window for statement.

This example shows we have overtyped the collection name with "CA99ONLS." We then clicked the "Run Selected icon to run this statement. The next foil shows some of the schema objects created by this procedure.



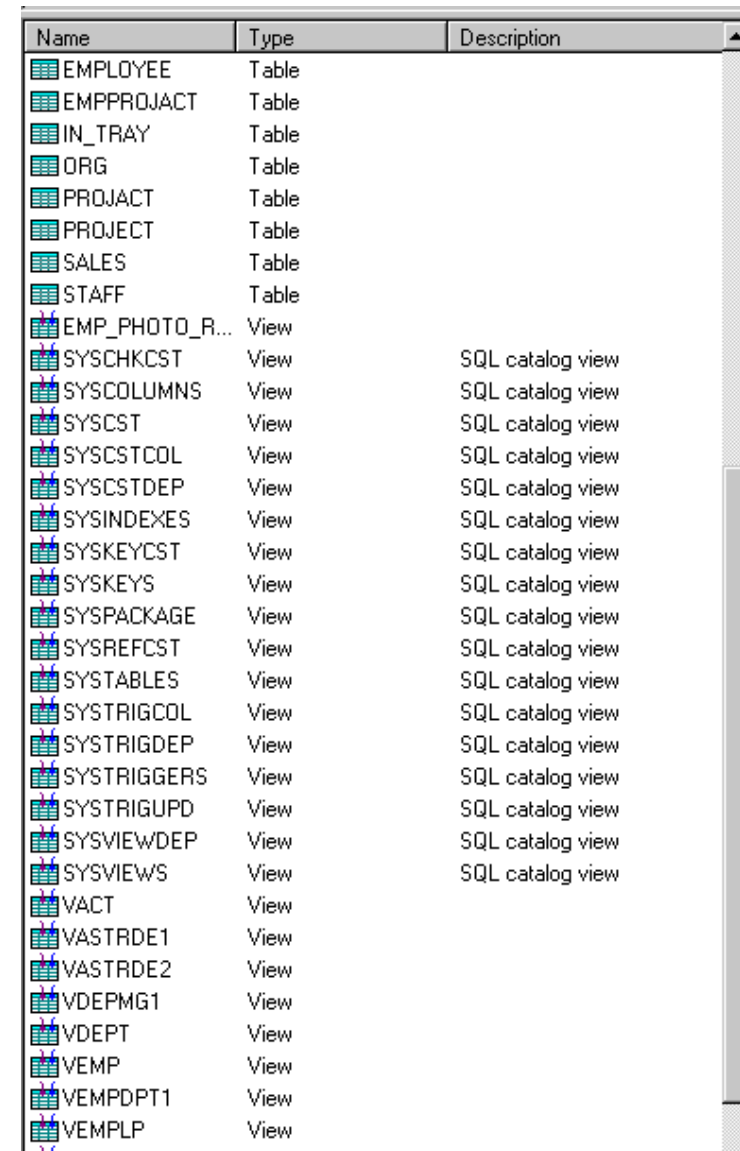
Create Schema Example - Objects Created

Select "CA99ONLS" under Database-Libraries



CA99ONLS - As01

Name	Type	Description
DEPT	Alias	
EMP	Alias	
EMP_ACT	Alias	
EMPACT	Alias	
PROJ	Alias	
XACT1	Index	
XACT2	Index	
XDEPT1	Index	
XDEPT2	Index	
XDEPT3	Index	
XEMP1	Index	
XEMP2	Index	
XEMP_PHOTO	Index	
XEMP_RESUME	Index	
XPROJ1	Index	
XPROJ2	Index	
XPROJAC1	Index	
QSQRN	Journal	COLLECTION - created by SC
QSQRN0001	Receiver	COLLECTION - created by SC
ACT	Table	
CL_SCHED	Table	
DEPARTMENT	Table	
EMP_PHOTO	Table	
EMP RESUME	Table	



Name	Type	Description
EMPLOYEE	Table	
EMPPROJECT	Table	
IN_TRAY	Table	
ORG	Table	
PROJECT	Table	
SALES	Table	
STAFF	Table	
EMP_PHOTO_R...	View	
SYCHKCST	View	SQL catalog view
SYSCOLUMNS	View	SQL catalog view
SYSCST	View	SQL catalog view
SYSCSTCOL	View	SQL catalog view
SYSCSTDEP	View	SQL catalog view
SYSINDEXES	View	SQL catalog view
SYSKEYCST	View	SQL catalog view
SYSKEYS	View	SQL catalog view
SYSPACKAGE	View	SQL catalog view
SYSREFCST	View	SQL catalog view
SYSTABLES	View	SQL catalog view
SYSTRIGCOL	View	SQL catalog view
SYSTRIGDEP	View	SQL catalog view
SYSTRIGGERS	View	SQL catalog view
SYSTRIGUPD	View	SQL catalog view
SYSVIEWDEP	View	SQL catalog view
SYSVIEWS	View	SQL catalog view
VACT	View	
VASTRDE1	View	
VASTRDE2	View	
VDEPMG1	View	
VDEPT	View	
VEMP	View	
VEMPDPT1	View	
VEMPLP	View	

In this example we under Database-Libraries., we added schema "CA99ONLS" and opened it. This foils shows most, but not all of the 69 objects created. Objects types include:

- Alias
- Index
- Journal and Journal receiver
- Table
- View
- SQL Catalog views

These objects can be used to try out and learn most of the Operations Navigator Database functions.

Note: Schemas created with this support can be deleted with the DROP SCHEMA statement.

Business Applications Update

IBM @server. For the next generation of e-business.

Notes: Business Applications Update

IBM  server iSeries

The following pages are excerpted from the April 2001 iSeries announcement marketing presentation - Business Intelligence iSeries.

The V4R5 ITSO Technical Overview Business Intelligence presentation has additional BI-related information on software providers.

IBM  server. For the next generation of e-business.

IBM Business Intelligence Software Solutions

IBM @server. For the next generation of e-business.

DB2 OLAP Server for iSeries, Version 7.1

IBM  server iSeries

Updates DB2 OLAP Server on iSeries with the latest OLAP functions

- Same functionality as Hyperion Essbase 6.0 Patch 1 and ShowCase STRATEGY 4.0
- Choice of data storage options (MDSM or RSM) by application

Advanced attribute analysis

- Economical storage usage
- Built-in functions
- Richer analysis

Enhanced performance and scalability

- Larger outlines

Enhanced analytics

- New Query Designer
- Linear regression, exponential smoothing, allocations, ...

Enterprise manageability

- Parallel user login
- Parallel application start/stop

IBM  server. For the next generation of e-business.

DB2 OLAP Server for iSeries Builder

- Graphical ETML tool to define and populate datamarts from iSeries or DB2 data sources
 - Based on ShowCase Warehouse Builder 4.0
 - Move and transform transaction data
 - Full DB2 UDB for iSeries SQL support for data cleansing via pre- or post- SQL, or user programs
 - Execute load/calc rules to build DB2 OLAP Server "cubes"
- Additional Builder data source options:
 - DB2 OLAP Server for iSeries Builder, DB2 NT Source
 - DB2 OLAP Server for iSeries Builder, DB2 AIX/MVS Source

Previously available as a Tools Bundle, now as separately orderable features:

- DB2 OLAP Server for iSeries Currency Conversion
 - Data conversion via exchange rates
- DB2 OLAP Server for iSeries Extended Spreadsheet Toolkit
 - Macros & VBAs to enable building customized Microsoft Excel and Lotus 1-2-3 applications
- DB2 OLAP Server for iSeries Application Programming Interfaces (APIs)

DB2 OLAP Server for iSeries Partitioning Option

- Integrate multiple cubes, centrally administer, provide replication and share metadata

DB2 OLAP Server for iSeries, Developer Edition V7.1

Full function DB2 OLAP Server

- Packaged and priced for developers of DB2 OLAP Server applications
- Includes entitlement for 1 install and 1 Developer User

Additional Concurrent User entitlements may be purchased

Developer Edition Optional Tools (may be purchased separately):

- Currency Conversion
- Extended Spreadsheet Toolkit
- APIs

Developer Edition Partitioning Option

IBM server. For the next generation of e-business.

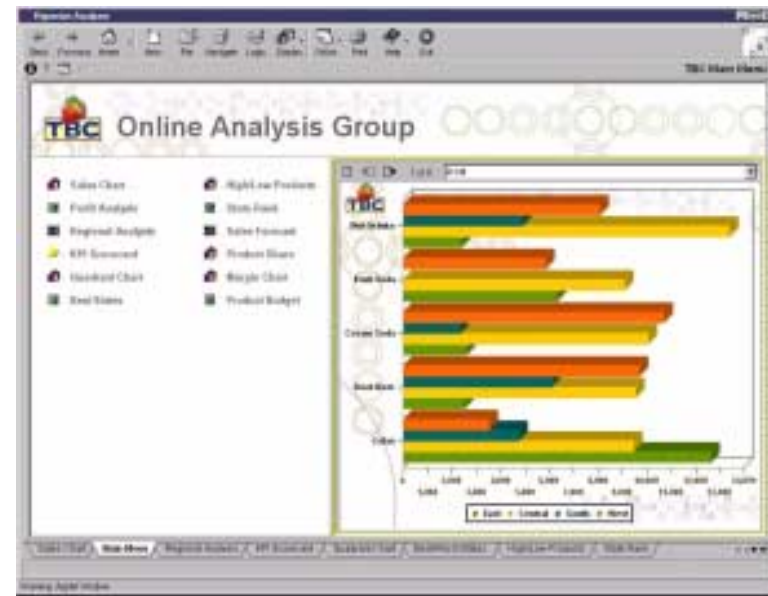
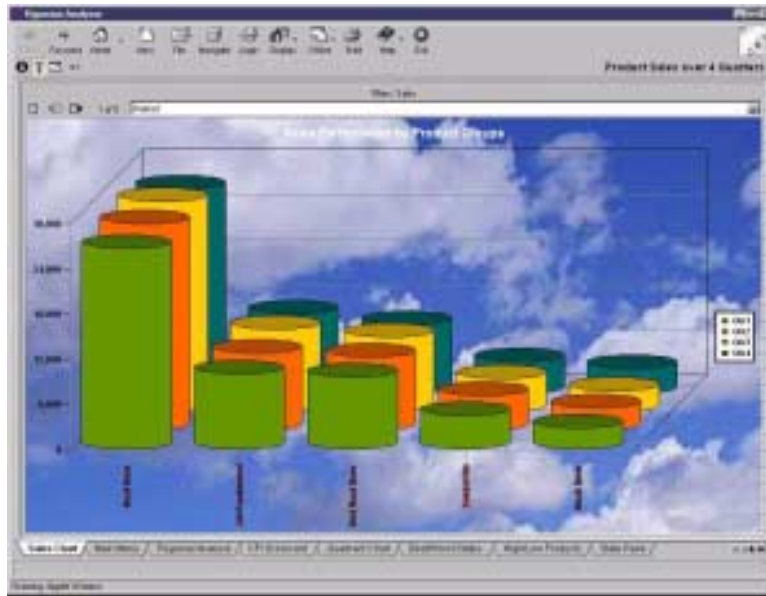
IBM Analyzer - New for DB2 OLAP Server V7.1

Front end presentation and analysis tool for DB2 OLAP Server

- Based on Hyperion Analyzer 5.0.1

Usable out of the box

- Intuitive cube navigation and Direct manipulation
 - Delivers intuitive, easy-to-use Web- and Windows-based interactive analysis
- Charts, spreadsheets, pinboards, forms
 - Provides highly colorful and graphic front-end to deliver OLAP data



IBM  **server.** For the next generation of e-business.

Enterprise deployable to a wide range of users

- JavaJ , HTML, Windows clients

Extendable using open, mainstream development tools

- e.g., Java applications run on Windows and Web... build once and run in both environments

Leverages the power of IBM DB2 OLAP Server

- e.g., Attribute dimensions, write back, LRO, etc.

Ideal for a broad range of applications including

- Sales analysis, Product profitability, Key performance, etc.

Relational Drill-Through Reports

National Language Versions: Japanese, German, French

No product key specification during install

Products remarketed from ShowCase Corporation

- ShowCase Analyzer Server & Analyzer for the Web
- ShowCase Warehouse Manager
- ShowCase Report Writer
- ShowCase Enterprise Reporting
- ShowCase Deployment Accelerators
 - Financial Deployment Accelerators
 - ▶ J.D. Edwards
 - ▶ Infinium Software
 - Sales Analysis Deployment Accelerators
 - ▶ J.D. Edwards

ShowCase fulfills orders with version 4.0 products

- Compatible with DB2 OLAP Server for iSeries V7.1

See U.S. Announcement Letter #200-281, August 15, 2000

IBM  server. For the next generation of e-business.

DB2 Warehouse Manager for iSeries, Version 7.1

IBM  server iSeries

An integrated warehouse management infrastructure

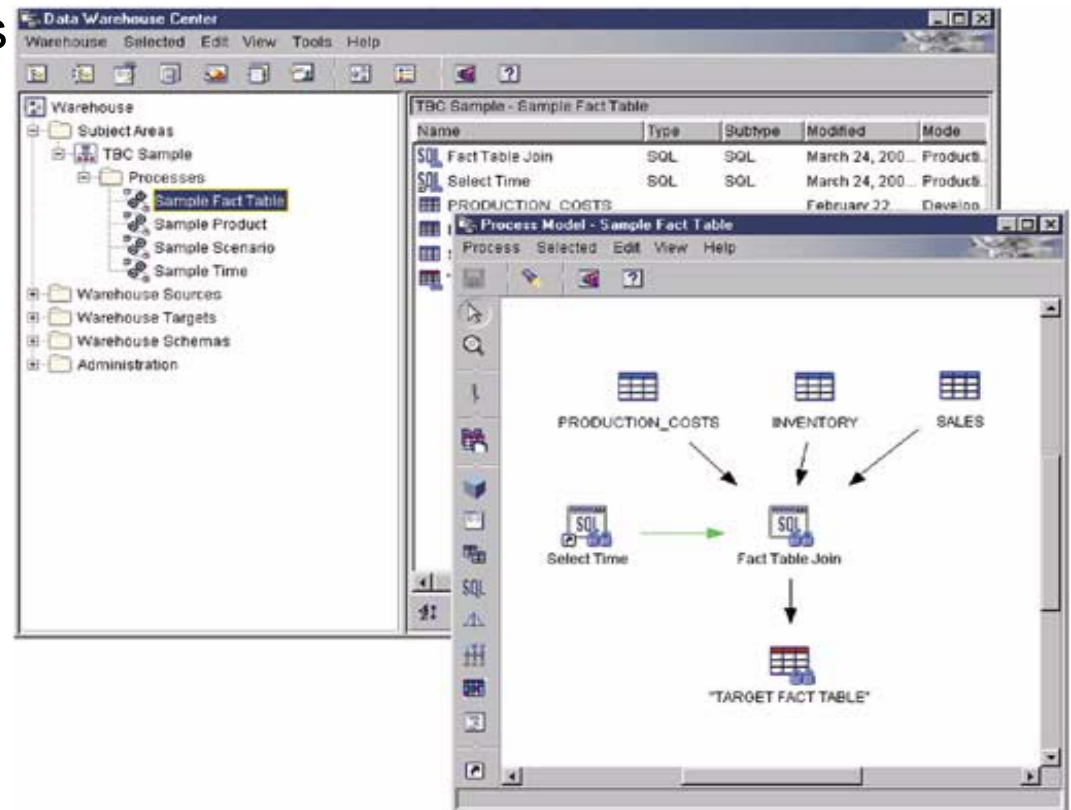
- for building, managing, governing, and accessing DB2 warehouses
- that is extendible with partner tools
- and manages technical and business metadata.

Manage data volume

- Point to point data movement
- Distributed transformation logic on iSeries
- Tracking hot spots and archival candidates

Manage large scale deployment

- Web-based query
- Information catalog
- Query governing



IBM  server. For the next generation of e-business.

DB2 Warehouse Manager for iSeries, Version 7.1

IBM  server iSeries

Provides access to IBM and non-IBM data sources

Simplifies and speeds warehouse prototyping, development and deployment

Empowers the data center to govern queries, analyze costs, manage resources, and track usage

Helps find, understand, and access information

Satisfies common reporting needs of enterprises of any size

Accommodates a myriad of data warehousing tools and techniques

IBM  server. For the next generation of e-business.

DB2 Warehouse Manager 7.1

Data Warehouse Center for iSeries Enablement

- ▶ Warehouse Agent for iSeries
- ▶ Transformer enablement

Information Catalog

DB2 UDB EE V7.1
(Limited Use Lic.)

QMF for Windows



**General Availability:
September 2000**

Languages

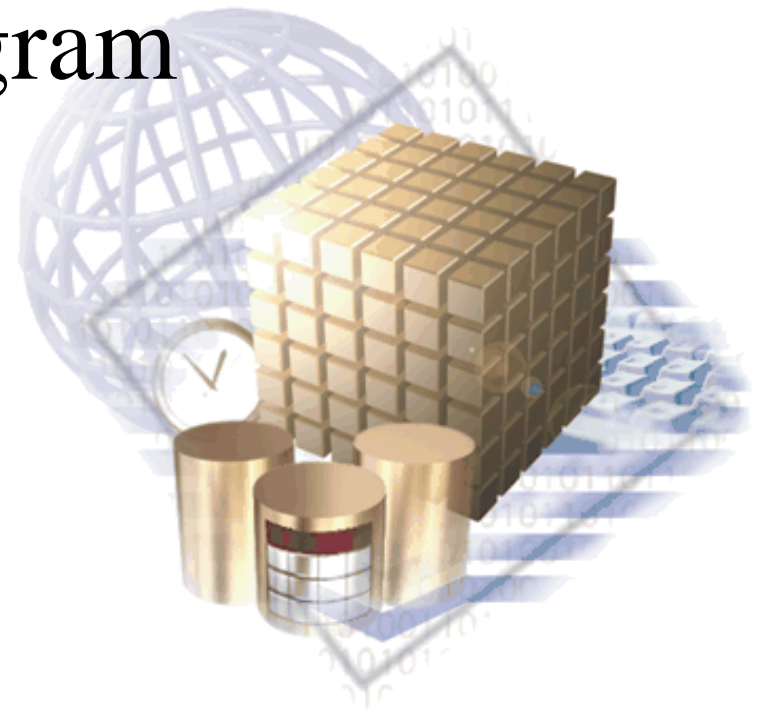
English, French, Italian, Spanish,
German, Brazilian Portuguese,
Japanese, Korean, S. Chinese, T. Chinese,
Swedish, Danish, Finnish, and Norwegian

QMF for Windows



Available Feb 2000
Available as a separate
product today

DB2 OLAP Server for iSeries V7.1 Try & Buy Program



IBM @server. For the next generation of e-business.

DB2 OLAP Server for iSeries V7.1 Try, Buy Program

Trial Edition contains same power and function as the Standard Edition of IBM DB2 OLAP Server for iSeries Version 7. 1

- Same product prereqs apply as the *Standard Edition* of DB2 OLAP Server for iSeries, V7.1

Expires after 180 days from installation

Trial software is for demonstration use only; not for production use

Special Website available specific to the Trial Program:

- Access additional information regarding the Trial Program, Local/Geography Contact information, and additional "free-ware," such as a free try and buy version IBM Analyzer, etc.

www-4.ibm.com/software/data/db2/tryolap.html

Softcopy documentation provided on Trial CD

Trial CDs delivered with new iSeries

- Geo DM focal points have an inventory to cover other opportunities

IBM @server. For the next generation of e-business.

Manufacturing

- Silvon Stratum
 - Sales Analysis
 - Category Management
 - Margin Analysis
- amis Manufacturing Works
- Lawson Analytic Extensions
- Vanguard Solutions - GPS Series
- interBiz ClearView
- American Software Intelliprise
 - Supplier Performance
 - Channel Performance
 - Inventory Performance
- Analytical Solutions E*telligence
- InfoManager Planner
- SAP BW

Finance

- HNC Financial Solutions ProfitVision
- amis Banking Works
- InfoManager Banking Solution
- EZMart for Profitability (IBM)

Retail/Distribution

- JDA Retail IDEAS
- Analytical Solutions E*telligence
- Silvon Stratum
 - Sales Analysis, Category Management, Margin Analysis
- DSS Agent for Retailers
- Retail Operations Analysis
- IBM RetailMart 3D
- InfoManager CRM
- InfoManager Planner
- SVI "The Eye"
- Dimensional Insight

Insurance

- InfoManager Insurance Solution
- amis Insurance Works
- NGS/Wheatley IQ Server for Ins.

Other/Cross Industry

- Hospitality & Gaming Analysis
- NGS Hospital Information Systems
- A3 Profit Center - Budget/Forecasting
- Walker - Budget/Forecasting

Trademarks & Disclaimers

© Copyright International Business Machines Corporation 2001

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

AIX	Application Development	AS/400
AS/400e	DB2	Domino
IBM	OfficeVision	OS/400
Integrated Language Environment	Net.Commerce	Net.Data
PowerPC	PowerPC AS	SanFrancisco
Host on Demand	Screen Publisher	Host Publisher
PCOM	WebSphere Commerce Suite	Payment Manager
WebSphere	WebSphere Standard Edition	WebSphere Advanced Edition
MQSeries	MQSeries Integrator	Host Integration Series
WebSphere Development Tools for AS/400	VisualAge for Java	VisualAge for RPG
CODE/400	DB2 UDB for AS/400	HTTP Server for AS/400
iSeries		

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

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

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

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

PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both and is used by IBM Corporation under license.

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

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

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

Other company, product and 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.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

IBM  server. For the next generation of e-business.