V5R2 DB2 UDB for iSeries
Agenda

iSeries Strategic Initiatives

Transaction Management

Multiple Database Support

SQL Enhancements - keeping up with Industry Standards

iSeries Navigator Interface

DB2 DataPropagator for iSeries V8.1

Summary

Appendix

- Commentary on DB2 UDB Version 8.1
- Resources for additional information
Notes: Agenda

This foil lists the major categories of iSeries database-related new functions and enhanced functions available with V5R2 as of August 2002 general availability.

From a general marketing viewpoint, we have sequenced the topics for Transaction Management and Multiple Databases on the same system before the set of SQL industry-standards enhancements, though V5R2 contains an impressive set of enhancements in this area.
DB2 UDB for iSeries Strategic Initiatives

Without disrupting your current applications

Openness - Industry Standard Support

- Accommodate 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
Notes: DB2 UDB for iSeries Strategic Initiatives

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.

Instead of focusing solely on line item enhancements, it is important to step back and understand from a strategy standpoint why these line items are in place. While the AS/400 has always had a tremendous relational database management system built into the OS, many applications were developed using fairly proprietary interfaces vs. the open Structured Query Language (SQL) methods of accessing and managing databases. But in today's world, ISV solutions are being written for portability and multiple platform support purposes. To accomplish this, they choose application development environments such as WebSphere, Java, or Domino, and they choose SQL as the common, portable method of communicating to the database.

IBM has several research labs that continue to produce new advances in database technology. The beauty of the IBM DB2 UDB family is that we can all share from this, and DB2 UDB for iSeries will continue to leverage the technology leadership position of IBM and maintain close (BUT PROBABLY NEVER 100%) compatibility with the other DB2 family members.

DB2 UDB for iSeries initiatives include (1) ensuring OPENNESS through the support of industry standards such as SQL, and (2) DB2 Family Compatibility. Additionally, DB2 UDB for iSeries will continue to leverage and build on the strengths of the eServer iSeries. Those strengths include reliability/availability, total cost of ownership, scalability, and usability.

The following pages of this presentation demonstrate these strategic initiatives.

Note: This presentation includes updates on the DB2 DataPropagator for iSeries V8.1, product. It does not include updates on other iSeries database and business intelligence applications that are available. See the V5R2 Marketing announcements presentations available under IBM System Sales and business partner PartnerInfo website via servers -> iSeries -> iSeries sales kit. The end of this presentation includes these URLs. The Database marketing presentation does contain updates on these other iSeries database-related products.
Transaction Management: Adaptive e-transaction Services
**Application Evolution**

**Historical**
- IBM directly influenced the development of application servers on AS/400
- Management of resources controlled by OS/400
- Business logic in RPG and COBOL
- Database access dependent on vendor-specific APIs
- XA and CORBA transaction specifications difficult to implement

**Current**
- Application servers being developed using third party transaction managers
- Management of resources controlled by application
- Business logic wrapped in transactional objects
- JDBC API is independent of vendor-specific APIs
- JTA simplifies transaction management implementation
Notes: Application Evolution

Application development on iSeries is moving more and more towards an industry-defined transaction oriented paradigm. In contrast to the popularly called "interactive transactions" in the context of an iSeries metric, which refers only to an almost mechanical event of hitting a key on a keyboard, the term transaction used here is more complex. It is used here to define a logical unit of work (usually associated with, that is part of, a business transaction) which either completely succeeds or has no effect whatsoever. It allows work being performed in many different processes, at possibly different sites, to be treated as a single unit of work.

IBM's DB2 Universal Database (UDB) has supported "two-phase commit" for a distributed set of database operations for years with the COMMIT and ROLLBACK operations, first via SNA DRDA protocols and then with V5R1 also via TCP/IP-based protocols. The transaction processors, such as those provided with IBM's WebSphere Application Server, WebLogic, and Tuxedo can perform these corresponding functions "outside of the programs" which means the COMMIT and ROLLBACK operations do not need to be included in the application's programming statements.

The two-phase commit support for accessing multiple remote databases uses the term Distributed Unit of Work (DUW) to represent the currently active transaction. A Remote Unit of Work (RUW) is the corresponding term representing a one-phase commit transaction - all files/tables processed within a transaction reside on a single system.

The industry-defined transaction oriented paradigm is a TCP/IP-based extension of these capabilities. Because the work being performed (within the bounds of such a transaction) can occur on many different platforms and involve many different databases from different vendors, a standard has been developed to allow a manager process to coordinate and control the behavior of the databases. X/Open is a standards body that developed the Distributed Transaction Processing Model and the XA interface to solve the heterogeneous databases reality. X/Open applications run in a distributed transaction processing (DTP) environment. In an abstract model, an X/Open application calls on resource managers (RMs) to provide a variety of services. For example, a database resource manager provides access to data in a database. Resource managers interact with a transaction manager (TM), which controls all transactions for the application. The X/Open DTP Model defines the communications between an application, a Transaction Manager (TM) and one or more Resource Managers (RM). A RM is nothing more than an interface that provides access to shared resources and that supports the X/Open XA interface. The most common RM is a database (i.e. Oracle, DB2), however a print spooler can be a RM if it were to support commitment control through the X/Open XA interface. The X/Open XA interface is a specification that describes the protocol for transaction coordination, commitment, and recovery between a TM and one or more RMs.

The X/Open Distributed Transaction Processing Model consists of:

- An application program that specifies actions which constitute a transaction.
- Resource Managers (i.e. RDBMS and Tuxedo /Q) which provide access to shared resources.
- A Transaction Manager that assigns identifiers to transactions, monitors their progress, and takes responsibility for transaction completion and for failure recovery.
Notes: Application Evolution -2

The Java Transaction API (JTA) has support for complex transactions. It also provides support for decoupling transactions from Connection objects. A Connection object represents a connection to a data source in Java Database Connectivity (JDBC). It is through Connection objects that Statement objects are created for processing SQL statements against the database. An application program can have multiple connections at one time. These Connection objects can all connect to the same database or connect to different databases. As JDBC is modeled after the Object Database Connectivity (ODBC) and the X/Open Call Level Interface (CLI) specifications, JTA is modeled after the X/Open Extended Architecture (XA) specification. JTA and JDBC work together to decouple transactions from Connection objects.

By decoupling transactions from Connection objects, this allows you to have a single connection work on multiple transactions concurrently. Conversely, it allows you to have multiple Connections work on a single transaction. For V5R2 we use the term *Adaptive e-transaction Services* to distinguish this support from previous release implementation.
Current Application / Transaction Support

1000s of clients

- 1 to 1 relationship between client transaction and system resources
  - One connection associated with each commit definition
- Resources not reusable until client completes transaction
Notes: Current Application / Transaction Support

The current infrastructure in the iSeries to provide access to shared resources, such as DB2, is performed via host server jobs - QSQSRVR job names. These jobs are called whenever a request is brokered against the shared resource. This implies that the transaction boundaries are known only within the instance of the server job that will perform the entire access to the shared resource, even if this would require the job to access another platform. Each transaction is linked with the atomicity of the server job in which it is running.

Although this model requires little or no work to set up and has given proof of stability, it nevertheless has some implications on transaction management, compatibility and scalability. The major drawbacks being that there is a one-to-one relation between a client transaction and system resources and that resources, locked up by one transaction, cannot be made available before the transaction has terminated.

In our example we have five application server jobs each doing a "remote" connection with one of five QSQSRVR jobs. Each QSQSRVR job is responsible for all functions associated with a single connection - commit cycle transaction.

In the next foils you see how transaction management can be done starting with V5R2 of OS/400.
New Transaction Service Implementation

- Multiple connections per transaction in SQL server job or
  Multiple transactions per connection in SQL server job
- Implemented using industry standard transaction specifications (JTA, XA, CORBA, JTS)
- WebSphere Application Server, Tuxedo provide transaction management

1000s of clients
When JTA and JDBC are used together, there are a series of steps between them to accomplish transactional work. Support for XA is provided through the XADataSource class. This class contains support for setting up connection pooling exactly the same way as its ConnectionPoolDataSource superclass. With an XADataSource instance, you can retrieve an XAConnection object. The XAConnection object serves as a container for both the JDBC Connection object and an XAResource object. The XAResource object is designed to handle XA transactional support. XAResource handles transactions through objects called transaction IDs (XIDs).

The XID is an interface that you must implement. It represents a Java mapping of the XID structure of the X/Open transaction identifier. This object contains three parts:

- A global transaction's format ID
- A global transaction ID
- A branch qualifier

The JTA API is designed to decouple transactions from JDBC connections. This API allows you to have either multiple connections work on a single transaction or a single connection work on multiple transactions concurrently. This is called multiplexing and many complex tasks can be performed that cannot be accomplished with JDBC alone - and take less system resource to perform a function.

A key implementation object is a new internal transaction object that maintains the transaction status, object and row/record locks, and other "commitment definition" information necessary to maintain transaction integrity. A single transaction object can be processed by multiple threads or a single thread can process a different transaction object that is passed to it. Under OS/400 the V5R2 transaction object is managed across connections and one or more QSQRVR jobs/threads.

Use of the new transaction model is optional. It can be enabled through the use of specific XA APIs by an application program or it can be enabled by the application transaction manager - on behalf of the application. When enabled by the transaction manager, there is no change to the application program as the resource management implementation can be independent of the application programming.

Transaction managers known to support this implementation include WebSphere Application Server Advanced Edition 40. or later and Tuxedo.

For information on using the XA APIs on iSeries refer to iSeries Information Center and use the search word "xa."
New Transaction Service Objectives

Compatible with other DB2 UDB offerings
- Equivalent XA and JTA support
- Savepoint transactions

Compliant with XA standard and JTA
- Multiple transactions per DB connection
- Multiple DB connections per transaction

Eliminate iSeries specific code to manage transactions and DB connections

Attract new applications and ISVs

Increased performance

Improved scalability
- Resources shared between clients
- Efficient use of system resources
New Transaction Service Outlook

Improved throughput (scalability): 5 to 20% improvement, depending on:

- Complexity of the transaction management reduced
- Increased number of clients per system performance rating

Positions iSeries as an even stronger member of eServer and e-business

- Compatible support with other UDB families with equivalent XA and JTA support
- Savepoint transactions
- Compliant with XA standard and JTA

WebSphere applications most likely to see the benefits

- Local transactions, complex transactions, large numbers of clients
- No benefit to remote transactions

Customers using transaction managers (WebSphere, WebLogic, Tuxedo) will want the capabilities being provided - change to source program not required
Notes: New Transaction Service Outlook

At the time this presentation was published, the application program can use the XA APIs or, a transaction manager can implement this for iSeries applications. Current plans call for:

- iSeries specific code to be removed from WebSphere Application Server Advanced Edition V5.0
- Tuxedo will be able to take advantage of support by using new switch that indicate to Tuxedo, which transaction interfaces to use.
Setting the new Data Source

Current Data Source specification: DB2StdXADatasource

Adaptive e-transaction services Data Source specification: UDBXADatasource
Notes: Setting the new Data Source

DB2StdXADatasource
This implementation class performs the old transaction model by default. A custom property was added to the V5R2 version of this class, so that it can be switched to use the NewTransaction Service transaction model. The custom property xaModel=standard can be used to accomplish this using this class. By default, xaModel=original.

UDBXADatasource
This implementation class will always use the NTS transaction model. xaModel custom property is ignored. The implementation classes that start with the prefix "DB2Std" are not being enhanced beyond V5R1. They are being replaced by implementation classes that start with the prefix "UDB". Users should be encouraged to move to these new classes when it becomes appropriate to do so.
Multiple DB (Namespace) Support
**Multiple Database Support**

Provides another level of server consolidation by sharing a single application across multiple databases that share the same name

Implemented through Independent Disk Pool (IASP) support

Can be used in a switched disk cluster for improved availability

Considerations:

- Library and object names can duplicated in each uniquely named IASP-database

- Application can access "SYSBAS objects" and selected IASP.database.library objects
  - New job description parameter (INLASPGRP)
  - New OS/400 SETASPGRP command
  - SQL CONNECT statement

- Performance overhead at each switch to a different IASP - just like accessing a remote db
  - Vary on: Minimize number of database objects in SYSBAS versus each IASP
  - "Open file:"
    - Each SQL program or SQL package access plan is unique, including database location
    - Each switch to a different database may require access plan to be rebuilt
Notes: Multiple Database Support

This presentation contains much information on Independent ASP support relative to how an IASP is used to support multiple databases on the same system.

For more details about overall implementation and managing of IASPs, including the clustering and high availability considerations, please see the Technical Overview Availability presentation. For example, the new IASP support provides improved application protection from multiple disk unit failures within an IASP on a single system. However a more completely protected environment (for example automatic switch over to another system) typically requires usage of cluster middleware software, provided by companies such as DataMirror, Lakeview Technology, and Vision Solutions. For information on these software offerings, refer to http://www.ibm.com/eserver/iseries/ha.

From a database viewpoint, V5R2 allows you to create multiple individual databases on a single system. This new feature enables operations such as the consolidation of several business entities within a single system, allowing them to maintain the database object naming and contexts as they exist on the original platforms. The enables you to split up within a single system (or partition), the access to specific tables or specific applications. It also allows consolidation of applications coming from other DB2 platforms - even to integrate them within the existing iSeries applications.

By definition, V5R2 uses the term *SYSBAS to refer to the namespace that collectively addresses all objects in the system storage pool ("system ASP" or ASP 1) and any optionally created user ASPs - now called a base disk pool in V5R2. Each Independent ASP will hold its own name space of objects, though we focus primarily on database objects in this topic. When an IASP is varied on ("made available" in iSeries Navigator) the IASP catalog of database information is rebuilt so as to also make all tables in the catalog of the schema (collections) residing in *SYSBAS also accessible.

The V5R2 support of library objects (QSYS.LIB objects) within an IASP greatly expands the capability introduced with V5R1 to "switch disks" to another system to increase the availability of the application environment for activities such as software or hardware that might require quiescing of the system.

Setting up to use multiple databases requires careful consideration of specific "realities" that may or may not apply to your operating environment. The primary considerations are listed on this foil.
Notes: Multiple Database Support -2

A library name as well as object names within that library can be the same on each of the configured IASPs. However, a library name or object name in an IASP cannot be duplicated within *SYSBAS. This has to do with important performance and security implementation.

From a database viewpoint the database associated with an IASP must have a unique name. It is the objects within that database that can have the same names.

With proper authorization, any job or thread within a job can access all objects located within *SYSBAS as well as the "current ASP group" (if one has been assigned to the job/thread).

Note: As described in the Availability presentation the system uses the term ASP Group to refer to a primary IASP and any optional associated secondary IASP. Each ASP Group must have a primary IASP; Optionally an ASP Group can also contain one or more secondary IASPs. A typical use of a secondary IASP is one used for journal receivers - this corresponds to use of the System ASP and a user (now called basic) ASP that would be used for journal receivers of files/tables in the system ASP.

There are several facilities available for an application to identify which, if any, IASP should be used to access an object:

- New V5R2 Job Description Parameter - Initial ASP Group (INLASPGRP)
- V5R2 Set ASP Group (SETASPGRP) command that specifies the ASP Group name
- SQL CONNECT statement: When performing database functions the already available SQL CONNECT statement can specify a database name. That database name must be already defined as either a local or remote database with the Add Relational Database Directory Entry (ADDRDBDIRE) command. For multiple local databases, the local system database (*SYSBAS) entry must specify "LOCAL" in its ADDRDBDIRE and, for an IASP database, the ADDRDBDIRE must specify LOOPBACK for its remote location parameter. This is instead of a remote IP address or host name.
- Distributed Data Management File (with a new *RDB parameter value).

Applications or 5250 work station jobs that do not use SQL CONNECT can take advantage of the other interfaces.
Notes: Multiple Database Support -3

There are performance considerations when accessing an object in an IASP. Depending upon your operating environment these may or may not impact you.

First, the vary on of an IASP takes time to correlate and rebuild the necessary name space information. If you have a large number of database objects in *SYSBAS and your IASP, the longer the vary on will take. This "catalog information" is also maintained synchronously and dynamically at run time when table changes are made. Therefore we recommend you minimize the number of database objects you have in *SYSBAS.

It is important to note, that associated with an application program accessing a database through SQL there is an access plan which serves as a performance aid in accessing the database data. Access plans can be stored in programs, SQL packages, and internal job or system storage depending on the environment. SQL packages are only used for DRDA and Extended Dynamic applications.

The access plan was built knowing the location of database information for the SQL operations you have in the program or package. Changing to a different IASP could require a rebuild of this access plan. Therefore you need to either carefully control the frequency and timing of switching to a different IASP - before performance critical application processing, or consider having a duplicated "application object," that contains the SQL program or package in each IASP. The duplicate SQL program or package would more likely have an access plan that does not require rebuilding when a job or thread initiates the connection to the database.

For programs that are generated from a Create SQL xxx (RPG, COBOL, and so forth) program, the associated SQL package is generated at program creation time, if the CRTSQLxxx command specifies an RDB. An SQL package can also be created and selected by an application at run time. This would typically be the case when the iSeries server is accessed by a client using the ODBC interface. An ODBC application gets an SQL package by specifying the Extended Dynamic option on the data source.

Alternatively your program could use the SQL Call Level Interface (CLI) to perform SQL processes against tables/files. The CLI interface is used by several OS/400 system functions, including Management Central, and typically by Java applications performing SQL functions. When using the CLI interface, the system implicitly creates the information that corresponds to the SQL package/access plan the first time the using application is run.
Notes: Multiple Database Support -4

You need to either carefully control the frequency and timing of switching to a different IASP - before performance critical application processing, or consider having a duplicated "application object," such as your SQL program and/or SQL package in each IASP. The duplicate SQL program/package would more likely have an access plan that does not require rebuilding.

Note the performance impact would be worse for a program with Static SQL when frequent rebuilds of the access plan are required. Dynamic SQL (which includes all CLI applications) uses internal job storage and/or system-wide statement cache available for each IASP name space. Because it is geared to real time updates, performance impact should be minimal for applications using this interface. Performance impact for Extended Dynamic should be between Static SQL and Dynamic SQL.
Multiple Databases: Additional local system considerations

To achieve the fastest performance, make sure that the database to which you perform an SQL CONNECT corresponds with your current library name space.

- If connected database does not match current name space, then a remote database connection will be used
- Current library name space can be set with SETASPGGRP command or job description INLASPGGRP parameter

Temporary objects always reside in the system ASP (SYSBAS)

No implicit access path sharing between databases indexes in the SYSBAS and IASP

Catalog synchronization performed at vary on of the IASP

SMAPP information stored per ASP

Other OS/400 functions and IASPs
Notes: Multiple Databases: Local system considerations

If you are considering having multiple databases on a single system, planning is required - not only if you are planning for clustering but also in a single system image of the implementation.

As previously described you need to consider using the new for V5R2 SETASPGRP command or the new Job Description command INLASPGRP to set up the "current name space" to include the appropriate IASP.

When running query functions against a database, frequently temporary objects, such as indexes, are created. By default, these will always reside in the system ASP - with the exception of the objects used for System Managed Access Path Protection. It is therefore essential that the system ASP has enough resources to allow for as much parallelism as possible while creating or using these temporary objects; carving out some disk units out of an existing ASP may not always be sufficient to allow for the existence of these objects or to ensure that performance remains at the same level as is the case if the database is not split up.

As previously described, since a synchronization of the catalog of the IASP database is performed at the vary on of the IASP, this operation requires some time to complete. Since all information in the system ASP database catalog will be imported into the catalog of the IASP database, we recommend to keep the system ASP database as limited as possible, if frequent IASP vary on and vary off sequences are to be expected and if the throughput of these actions should be as limited as possible.

The jobs that perform this synchronization run in subsystem QSYSWRK and can be identified by a prefix that identifies the IASP number. For example, the prefix QDBc033 represents the IASP number 33.

Several, but not all, V5R2 OS/400 commands support new ASP number and ASP device name parameters with the appropriate values. Examples include Create Library, Make Directory, Restore Library/Directory/object commands. For other situations such as when doing a save function, you may find either the new SETASPGRP command or the new job description command parameter works best for you.

The V5R2 Memo to Users describes this new support along with identifying the commands with new ASP Group parameters.

The following foils give some addition views of this multiple databases support.
Multiple Database Name Spaces

**iSeries Navigator view:**

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

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

- Relational Database Location
  - AS20 RCHASX20
  - DBCOOK LOOPBACK
  - DBEUOPS LOOPBACK
  - RCHASE5C *LOCAL

- Schema / Library in IASP database
- Schema / Library in system ASP database
Notes: Multiple Database Name Spaces

The iSeries Navigator Database view shows the configured databases on the system whose corresponding IASP has been varied on (made available in iSeries Navigator terminology). On the right of this foil, we used the Work with Relational Database Entries (WRKRDBDIRE) command to show the primary (default) database name - Rchase5C and the two additional local databases - Dbcook and Dbeuops - each with the LOOPBACK values for Remote Location. Rchase5c is contained in *SYSBAS. Dbcook is actually stored in IASP named DBITSC.

Note: The IASP name and associated database name default to the same name. It is not a requirement.

The database DBEUOPS is not shown in the Operations Navigator window because its associated IASP is not varied on.

You can view the database name to IASP name mapping in the following ways:
- iSeries Navigator -> Database -> Database name -> Properties. The associated IASP must be varied on ("made available" term with iSeries Navigator).
- 5250 command line: WRKCFGSTS *DEV *ASP -> For an ASP listed chose Work with descriptions -> Display Device Description
- 5250 command line: WRKRDBDIRE -> Display one of the entries showing LOOPBACK for remote location. Note, the IASP-associated database file name is automatically placed as an entry into the Relational Database Directory when it is first varied on after its creation. If the IASP is currently varied off, you can see the RDB entry, but cannot see what its associated IASP name is. Once that IASP has been varied on, the RDB directory entry shows the IASP name.

Notes:
- iSeries Navigator uses two different icons to indicate a library is in *SYSBAS or in an IASP. In this example you can see the library CPY2 icon has a blue circle behind the "warehouse icon." This indicates the library is within the IASP. Library PFREXP is in *SYSBAS and, as such it can be listed under either database Rchase5c or Dbcook.
- V5R2 Information Center for the Database topic has additional operational details for using database within IASPs.
Notes: Multiple Database Name Spaces -2

Distributed Data Management (DDM) supports accessing tables residing in a database defined on an IASP. The CRT/CHGDDMF interface now supports the *RDB value on the Remote Location Name (RMTLOCNAME) parameter of those commands and allow you to specify the name of a remote database on the Remote database (RDB) parameter. The Remote database parameter must match an entry as registered on the Relational Database directory table. This allows to carry over into a consolidated server applications that were running previously on separate systems and made (extensive) use of DDM files to access common tables on another system.
Multiple Database Support - Setup Example

*SYSBAS
- OS/400, under profiles
- Licensed Programs
- Work Mgt & User Definitions

User profiles / Job Descriptions

Common Application Code

IASPs

ASPEUR
Library/Schema: DBLIB

ASPASIA
Library/Schema: DBLIB

ASPUSA
Library/Schema: DBLIB
Notes: Multiple Database Support - Setup Example

This chart explains how to set up a consolidated environment on a single, not partitioned LPAR, where a common set of application code is used to access individual files with the same names, but of which the data has to be kept separately for business reasons.

Three IASPs are defined on this system, of which each will contain the data libraries of each entity in the company. Each IASP will therefore contain the same libraries with the same tables, views and indexes, but with different data - Europe, Asia, and USA. For purposes of simplifying change management, the application programs are stored only once in the system ASP.

The user profiles and their attributes are migrated from each system, and are associated with a job description that directs them to the IASP device where their data resides. This is done by specifying the name of the IASP device on the Initial ASP Group (INLASPGRP) parameter of the Job Description associated with each user profile. This way, a user that signs on to the system will have access to the common application code, residing in the system ASP, and to his data libraries, residing in the IASP.

If this system is running an application that needs to be enabled for support of different cultural values, it can still be achieved in this way, as long as the individual set up of the cultural objects follows the design principles as detailed in the International Application Development (SC41-5603) manual. In this case, multiple instances of the different cultural elements can be stored in the IASPs in their libraries, much as is the case on a system without multiple database support.

The advantage of such a consolidation, which can maintain its naming schema, is simple:

- No need to change the naming of all data containers, data objects or data attributes to fit in a consolidated IT service delivery environment; no secondary changes to the security setup either.
- Transparency for change and problem management including help desk support organizations.
Multiple Database Support - Setup Details

User Profile

- Initial Program
- Job Description

Initial Program:
- Identifies DB libraries to be worked with, uses SETASPGRP

Job Description, per IASP user(s):
- INLASPGRP
- Library List = *SYSVAL

System Value QUSRLIBL:
- PGMGLAPAR
- PGMOE
- PGMPROD
- PGMMISC
- ...

Code Libraries in SYSBAS:
- PGMGLAPAR
- PGMOE
- PGMPROD
- PGMMISC
- ...

...DB Libraries in SYSBAS:
- MASTER

Libraries per IASP DB:
- DBGLIB
- ...

IASP Definition

- IASPEUR
- IASPASA
- IASPUSA
Notes: Multiple Database Support - Setup Details

This figure shows more details of using multiple IASPs on a single system. This example is intended only as what we believe to be an implementation adhering to the principle of separate placing of processes and data.

The base assumption is to keep as much as possible the code libraries (programs, modules, environment descriptions) separated in the system ASP, while the data libraries are contained in separate IASPs, as required by the business entities. This is as discussed earlier in the notes for the foil: Multiple Database Support - Details.

The system value QUSRLIBL contains the names of the common code libraries and is used in the job descriptions associated with the different user profiles; each user profile has an initial program that sets up the data libraries the user will work with.

In a multilingual environment, you can add the libraries, containing the MRI (Machine Readable Information) or any other culturally dependent objects, in the system ASP also. These might also reside in the IASP.

In this example, we are showing the use of the SETASGRP command and the job description Initial ASP Group parameter. The SQL CONNECT or DDM file may also be used.

We show the SYSBAS library MASTER as a reminder the application can access objects in either the SYSASP or the "currently active" ASP Group.
A look at switched IASPs from a database view
DB2 object support in IASP (Independent ASPs)

- Support for switching database and non-database objects between systems

- Easy switchover of application database(s) for scheduled maintenance and upgrades
  - Primary and Secondary system have to be within 250 meters
  - Clustering (IP Takeover) is the switching mechanism
Detailed View - Switchable IASP

Cluster

Active
1.3.22.114

Inactive
1.3.22.114

Application & Data & Device CRGs

Application & Data & Device CRGs

Varied On

Varied off

IASP
Notes: Detailed View - Switchable IASP

If High Availability is the reason for using switchable ASPs in a clustered environment, you might want to take into consideration that a switchable ASP by no means provides a copy of your database. This means that if the tower, in which the IASP is housed, is lost, the data will be lost also.

A way to avoid data loss is to use a replication tool that copies the data as it changes to another system; one could also use remote journaling or a replication, such as provided by the HA business partners, to be able to reconstruct the data on a third system.

Remember also that the switchable IASP fabric allows only distances of up to 250 meter between a system unit and a tower. In case of a disaster, this will definitely prove to be an inefficient way of protecting a system and its applications.

Remember also that only a subsetted list of OS/400 objects is supported in an IASP. See the detailed Availability presentation or refer to the V5R2 Memo to Users.

The advantages of this setup is that the switching of a tower from a failing system to another node in a cluster domain are obvious:

- Faster switchover of the application, since the data is available in a very limited time span
- Full support of clustering features, including IP address takeover.

However, since only the original operational data and no copy of it is kept in this example, it is recommended to have a copy of this data either under the form of a business copy (off-line) and its changes (mostly represented by an off-line copy of the journals and journal receivers) or as a mirror or replica of the operational data. This setup allows to recover from site loss situations, since there is virtually no limit to the distance between the clustered domain and the D/R site, except for the limitations imposed by the available network bandwidth.
SQL Industry Standard Enhancements
Overview - 1 of 2

- ROWID data type and ROWID scalar function
- IDENTITY column attribute
- CREATE TABLE AS (subselect)
- DECLARE GLOBAL TEMPORARY tables
- User-defined Table Functions
- COMMIT ON RETURN procedures
- UNION in views
- Scalar subselect enhancements
- READ ONLY and READ WRITE in SET TRANSACTION
- ITERATE and nested Compound statements in SQL procedures, SQL functions, and SQL triggers
- Fullselect in derived tables and common table expressions
- Parameter markers in labeled durations
- Savepoints
Overview - 2 of 2

- SET SCHEMA and SET SQLID
- HOLD LOCATOR
- ORDER BY expression not required in the select-list
- ORDER BY and FETCH FIRST n ROWS ONLY in derived tables and common table expressions
- Length of SQL statements increased to 64K
- Length of delimited column name identifiers increased
- SUBSTRING enhancements
- VARCHAR concatenation enhancement
- Debug of original source statements in SQL procedures, SQL functions, and SQL triggers
- Multiple relational databases on iSeries
- Standard and ODBC and JDBC catalog views
- C derived variables
Notes: SQL Enhancements

Starting with this and the following foils we bring you a list of SQL-based enhancements that continue to demonstrate the iSeries commitment to provide functions and ease of use interfaces based upon Industry Standards, compatibility with the DB2 UDB family, and "traditional" iSeries programmers SQL requirements.

The following foils expand on many of the enhancements listed on these 2 foils. (We have already discussed multiple databases on the same system or partition.)
**ROWID**

**Data type:**

- Assigns a unique identifier to a row
- Allows direct access to a row
- Example:
  ```sql
  CREATE TABLE EMPLOYER
  (SERIAL ROWID GENERATED ALWAYS,
   NAME CHAR (30),
   ADDRESS CHAR (50))
  ```

**Scalar Function**

- Casts a character string or binary string to a row ID
- Example:
  ```sql
  SELECT NAME
  FROM EMPLOYER
  WHERE SERIAL=ROWID(X'842FEEE6F9DA8000F1F010F3F9F4E3D4404000000000000000001')
  ```
Notes: ROWID

ROWID Data type:
Using ROWID is a way to have the system assign a unique value to a single column in a table. 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. ROWID is similar to identity columns, but rather than being an attribute of a numeric column, it is a separate data type. A column or a host variable can have a row ID data type. A ROWID column enables queries to be written that navigate directly to a row in the table. Each value in a ROWID column must be unique. The database manager maintains the values permanently, even across table reorganizations. When a row is inserted into the table, the database manager generates a value for the ROWID column unless one is supplied. If a value is supplied, it must be a valid row ID value that was previously generated by either DB2 UDB for OS/390 and z/OS or DB2 UDB for iSeries.

The internal representation of a row ID value is transparent to the user. The value is never subject to CCSID conversion because it is considered to contain BIT data. The length attribute of a ROWID column is 40. The ROWID data type is not subject to CCSID because it is treated by DB2 UDB as bit-oriented data.

ROWID Scalar Function:
The ROWID function casts a character string or binary string to a row ID. The string-expression must not be a CLOB. Although the string can contain any value, it is recommended that it contain a ROWID value that was previously generated by DB2 UDB for OS/390 and z/OS or DB2 UDB for iSeries to ensure a valid ROWID value is returned. For example, the function can be used to convert a ROWID that value that was cast to CHAR value back to a ROWID value.

If the actual length of string-expression is less than 40, the result is not padded. If the actual length of string-expression is greater than 40, the result is truncated. If non-blank characters are truncated, a warning is returned. The length attribute of the result is 40. The actual length of the result is the length of string-expression.

The result of the function is a row ID. If the argument can be null, the result can be null; if the argument is null, the result is the null value.

GENERATED may be specified for a ROWID column:
- ALWAYS is the recommended value and specifies that the database manager will always generate a value for the column when a row is inserted into the table.
- BY DEFAULT specifies that the database manager will generate a value for the column when a row is inserted only if a value is not specified for the column. If a value is specified, the database manager uses that value. For a ROWID column, the database manager uses a specified value, but it must be a valid unique row ID value that was previously generated by DB2 UDB for OS/390 and z/OS or DB2 UDB for iSeries.
iSeries Navigator interface for ROWID

New in V5R2

Generated
Identity Column Attribute

Identifies a column (SMALLINT, INTEGER, BIGINT, DECIMAL) which will be automatically updated by the system on an INSERT based on user specifications

- Can be used for "special" key values such as surrogate keys or an order number
- Unique value not guaranteed: primary or unique key must be defined
- Supported only on SQL tables (identity column value is generated for either SQL or non-SQL ("native database") interface

Example: CREATE TABLE employee
          (empno SMALLINT GENERATED ALWAYS AS IDENTITY
           (START WITH 500 INCREMENT BY 1 CYCLE), id SMALLINT,
           name CHAR(30))

Customizable IDENTITY attributes:

- START WITH, INCREMENT BY: control the initial value and increment value
- MAXVALUE, MINVALUE: can be used to allow a limited range of generated values
- CYCLE, NO CYCLE: NOCYCLE (default) causes DB2 error when maximum value reached; CYCLE enables DB2 to start reusing values beginning at START WITH value
- CACHE, NOCACHE: CACHE enables DB2 to reserve the next set (or cache) of IDENTITY column values in main storage. Default of CACHE 20 is recommended.
- ORDER, NOORDER currently have no effect on IDENTITY column
Notes: Identity Column Attribute

The AS IDENTITY attribute specifies that the column is an identity column for the table. The intended function is to have the DB2 automatically set a key value, that could be, for example, the next order number values. A table can have only one identity column. AS IDENTITY can be specified only if the data type for the column is an exact numeric type with a scale of zero (SMALLINT, INTEGER, BIGINT, DECIMAL or NUMERIC with a scale of zero, or a distinct type based on one of these types). An identity column is implicitly NOT NULL.

You can compare this function with a "technical key" that you would create using DB2 Warehouse Manager to uniquely identify a row in a table: every time that a new row is added to a table with an identity column, the identity column value in the new row is incremented (or decremented) by the system; however, there is no implicit within this function to force uniqueness of the contents: if uniqueness is required, the user must add a UNIQUE or PRIMARY KEY constraint or add a UNIQUE index.

`START WITH numeric-constant` specifies the first value that is generated for the identity column. The value can be any positive or negative value that could be assigned to the column without non-zero digits existing to the right of the decimal point. If a value is not explicitly specified when the identity column is defined, the default is the MINVALUE for an ascending sequence and the MAXVALUE for a descending sequence. This value is not necessarily the value that a sequence would cycle to after reaching the maximum or minimum value of the sequence. The `START WITH` clause can be used to start a sequence outside the range that is used for cycles. The range used for cycles is defined by MINVALUE and MAXVALUE.

`INCREMENT BY numeric-constant` specifies the interval between consecutive values of the identity column. The value can be any positive or negative value that is not 0, does not exceed the value of a large integer constant, and could be assigned to the column without any non-zero digits existing to the right of the decimal point. The default is 1. If the value is positive, the sequence of values for the identity column ascends. If the value is negative, the sequence of values descends.

`MINVALUE numeric-constant` specifies the numeric constant that is the minimum value that is generated for this identity column. This value can be any positive or negative value that could be assigned to this column, but the value must be less than the maximum value. If a value is not explicitly specified when the identity column is defined, this is the START WITH value, or 1 if START WITH was not specified, for an ascending sequence; or the minimum value of the data type (and precision, if DECIMAL) for a descending sequence.

`MAXVALUE numeric-constant` specifies the numeric constant that is the maximum value that is generated for this identity column. This value can be any positive or negative value that could be assigned to this column, but the value must be greater than the minimum value. If a value is not explicitly specified when the identity column is defined, this is the maximum value of the data type (and precision, if DECIMAL) for an ascending sequence; or the START WITH value, or -1 if START WITH was not specified, for a descending sequence.

`CYCLE` or `NO CYCLE` specifies whether this identity column should continue to generate values after reaching either the maximum or minimum value of the sequence. CYCLE specifies that values continue to be generated for this column after the maximum or minimum value has been reached. If this option is used, after an ascending sequence reaches the maximum value of the sequence, it generates its minimum value. After a descending sequence reaches its minimum value of the sequence, it generates its maximum value. The maximum and minimum values for the column determine the range that is used for cycling. When CYCLE is in effect, duplicate values can be generated by the database manager for an identity column. If a unique constraint or unique index exists on the identity column, and a non-unique value is generated for it, an error occurs. NO CYCLE specifies that values will not be generated for the identity column once the maximum or minimum value for the sequence has been reached. This is the default.
**ORDER or NO ORDER** currently have no effect on the IDENTITY column.

**CACHE or NO CACHE** specifies whether to keep some preallocated values in memory. Preallocating and storing values in the cache improves the performance of inserting rows into a table. **CACHE integer** specifies the number of values of the identity column sequence that the database manager preallocates and keeps in memory. The minimum value that can be specified is 2, and the maximum is the largest value that can be represented as an integer. The default is 20. During a system failure, all cached identity column values that are yet to be assigned are lost, and thus, will never be used. Therefore, the value specified for CACHE also represents the maximum number of values for the identity column that could be lost during a system failure. NO CACHE specifies that values for the identity column are not preallocated.

In the example the column `empno` is defined with starting value of 500, incremented by 1 for every new row inserted, and will recycle when the maximum value is reached. In this example, the maximum value for the identity column is the maximum value for the data type. Because the data type is defined as SMALLINT, the range of values that can be assigned to ORDERNO is from 500 to 32767. When this column value reaches 32767, it will restart at 500 again. If 500 is still assigned to a column, and a unique key is specified on the identity column, then a duplicate key error is returned. The next insert will attempt to use 501. If you do not have a unique key specified for the identity column, 500 is used again, regardless of how many times it appears in the table.

For a larger range of values, you could specify the column to be an INTEGER or even a BIGINT. If you want you can have the value of the identity column decrease, by specifying a negative value for the INCREMENT option. You could also specify the exact range of numbers by using MINVALUE and MAXVALUE.
Identity Column Attribute - additional column support

IDENTITY_VAL_LOCAL can be used to return the last generated IDENTITY value

Overriding GENERATED ALWAYS columns on INSERT and UPDATE

- Normally any value specified on for a GENERATED ALWAYS column is ignored. Syntax is supported on UPDATE and INSERT to force the database to accept the value instead of generating one

- Examples:
  
  INSERT INTO employee
  OVERRIDE SYSTEM VALUE VALUES( ...)

  INSERT INTO employee
  OVERRIDE USER VALUE VALUES( ...)

  UPDATE employee
  OVERRIDE SYSTEM VALUE SET ...

  UPDATE employee
  OVERRIDE USER VALUE SET ...
IDENTITY_VAL_LOCAL is a non-deterministic function that returns the most recently assigned value for an identity column. The function has no input parameters. The result is a DECIMAL(31,0) regardless of the actual data type of the identity column that the result value corresponds to. The value returned is the value that was assigned to the identity column of the table identified in the most recent INSERT statement for a table containing an identity column. The INSERT statement has to be issued at the same level; that is, the value has to be available locally within the level at which it was assigned until replaced by the next assigned value. A new level is initiated when a trigger, function, or stored procedure is invoked. A trigger condition is at the same level as the associated triggered action.

The assigned value can be a value supplied by the user (if the identity column is defined as GENERATED BY DEFAULT) or an identity value that was generated by the database manager. The result can be null. The result is null if an INSERT statement has not been issued for a table containing an identity column at the current processing level. This includes invoking the function in a before or after insert trigger.

The result of the IDENTITY_VAL_LOCAL function is not affected by the following statements:
- An INSERT statement for a table which does not contain an identity column
- An UPDATE statement
- A COMMIT statement
- A ROLLBACK statement

The following explains the behavior of the function when it is invoked in various situations:
- Invoking the function within the VALUES clause of an INSERT statement
  Expressions in an INSERT statement are evaluated before values are assigned to the target columns of the INSERT statement. Thus, when you invoke IDENTITY_VAL_LOCAL in an INSERT statement, the value that is used is the most recently assigned value for an identity column from a previous INSERT statement. The function returns the null value if no such INSERT statement had been executed within the same level as the invocation of the IDENTITY_VAL_LOCAL function.
- Invoking the function following a failed INSERT statement
  The function returns an unpredictable result when it is invoked after the unsuccessful execution of an INSERT statement for a table with an identity column. The value might be the value that would have been returned from the function had it been invoked before the failed INSERT or the value that would have been assigned had the INSERT succeeded. The actual value returned depends on the point of failure and is therefore unpredictable.
- Invoking the function within the SELECT statement of a cursor
  Because the results of the IDENTITY_VAL_LOCAL function are not deterministic, the result of an invocation of the IDENTITY_VAL_LOCAL function from within the SELECT statement of a cursor can vary for each FETCH statement.
Invoking the function within the trigger condition of an insert trigger
The result of invoking the IDENTITY_VAL_LOCAL function from within the condition of an insert trigger is the null value.

Invoking the function within a triggered action of an insert trigger
Multiple before or after insert triggers can exist for a table. In such cases, each trigger is processed separately, and identity values generated by SQL statements issued within a triggered action are not available to other triggered actions using the IDENTITY_VAL_LOCAL function. This is the case even though the multiple triggered actions are conceptually defined at the same level. Do not use the IDENTITY_VAL_LOCAL function in the triggered action of a before insert trigger. The result of invoking the IDENTITY_VAL_LOCAL function from within the triggered action of a before insert trigger is the null value. The value for the identity column of the table for which the trigger is defined cannot be obtained by invoking the IDENTITY_VAL_LOCAL function within the triggered action of a before insert trigger. However, the value for the identity column can be obtained in the triggered action by referencing the trigger transition variable for the identity column. The result of invoking the IDENTITY_VAL_LOCAL function in the triggered action of an after insert trigger is the value assigned to an identity column of the table identified in the most recent INSERT statement invoked in the same triggered action for a table containing an identity column. If an INSERT statement for a table containing an identity column was not executed within the same triggered action before invoking the IDENTITY_VAL_LOCAL function, then the function returns a null value.

Invoking the function following an INSERT with triggered actions
The result of invoking the function after an INSERT that activates triggers is the value actually assigned to the identity column (that is, the value that would be returned on a subsequent SELECT statement). This value is not necessarily the value provided in the INSERT statement or a value generated by the database manager. The assigned value could be a value that was specified in a SET transition variable statement within the triggered action of a before insert trigger for a trigger transition variable associated with the identity column.
## iSeries Navigator for IDENTITY Column

### DBENHELOYEE Table Properties - Rchasm25(Rchase5c)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPNO</td>
<td>INTEGER</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>ADDRESS</td>
<td>CHARACTER</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

### New in V5R2

- Set as identity column
  - Step value: 1
  - Restart with value: 1
  - Minimum value: 1
  - Maximum value: 2147483647

- Allow applications to provide a value for the column
- Cycle values when the limit is reached

---

[ibm.com/eserver/series]
Notes: iSeries Navigator for IDENTITY Column

This foil illustrates the iSeries Navigator - Database interface to specifying the IDENTITY column attributes.
Field Reference File and CREATE TABLE AS

CREATE TABLE AS builds on the V5R1 CREATE TABLE LIKE support

- Allows data to be copied into a newly created table - an "SQL interface" to OS/400 Create Duplicate Object (CRTDUPOBJ) and Copy File (CPYF) functions, with additional functions
- Simplifies summary/work table creation into a single step
- Simulates OS/400 DDS Field Reference File support

Examples

- 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: Field Reference File and CREATE TABLE AS**

This foil shows how to use V5R2 CREATE TABLE AS to easily create a new table and copy in selected column data as is from another table - an SQL syntax way to perform the OS/400 Create Duplicate Object and Copy File. It also enables an "SQL interface" to simulate the OS/400 database support of an OS/400 "field reference file" (that has been used on OS/400 for years). That is, define a physical file that merely contains the field (column) definitions - no data. Use this as a reference file for specific field/column definitions when defining another table/file "based on" this reference file.

CREATE TABLE AS creates a table from the result of a SELECT statement. All of the expressions that can be used in a SELECT statement can be used in a CREATE TABLE AS statement. You can also include all of the data from the table or tables that you are selecting from.

If you compare this with existing functions, such as the Create Duplicate Object (CRTDUPOBJ) or Copy File (CPYF) commands, you will notice that this tool is far more powerful than the existing ones, since it allows join, projection, select and ordering functions as part of the subselect.

In the example shown in **green** - the CREATE TABLE SalesByRegion AS creates a table and with the SELECT statement copies selected column data from the sales table according to the GROUP BY... WITH DATA.

In the example shown in **red** - the CREATE TABLE customer AS creates a table WITH NO DATA that includes the columns from the specified reference file (ref_file) columns "inheriting" the column attributes from ref_file. In this example the column in table customer known as "id" has the attributes of the column "cust_id" in the reference file/table.

**FOR COLUMN** system-column-name provides an OS/400 name for the column. Do not use the same name for more than one column of the table or for a column-name of the table. If the system-column-name is not specified, and the column-name is not a valid system-column-name, a system column name is generated.

The **SELECT-statement** specifies that the columns of the table are to have the same name and description as the columns that would appear in the derived result table of the select-statement if the select-statement were to be executed. The use of AS select-statement is an implicit definition of n columns for the table, where n is the number of columns that would result from the select-statement. 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
- Nullability
- Column heading and text
Notes: Field Reference File and CREATE TABLE AS -2

The following attributes are not included (the default value and identity attributes may be included by using the copy-options):
- Default value
- Identity attributes

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 implicitly defined columns of the table inherit the names of the columns from the result table of the select-statement. Therefore, a column name must be specified in the select-statement or in the column name list for all result columns. For result columns that are derived from expressions, constants, and functions, the select-statement must include the AS column-name clause immediately after the result column or a name must be specified in the column list preceding the select-statement. The select-statement must not refer to host variables or include parameter markers (question marks).

**WITH DATA** specifies that the select-statement is executed. After the table is created, the result table rows of the select-statement are automatically inserted into the table.

**WITH NO DATA** or **DEFINITION ONLY** specifies that the select-statement is not executed. Therefore, there is no result table with a set of rows with which to automatically populate the table.
Notes: Field Reference File and CREATE TABLE AS -3

Copy Options:

INCLUDING IDENTITY COLUMN ATTRIBUTES specifies that the table inherits the identity attribute, if any, of the columns resulting from select-statement, table-name or view-name. In general, the identity attribute is copied if the element of the corresponding column in the table, view, or select-statement is the name of a table column or the name of a view column that directly or indirectly maps to the name of a base table column with the identity attribute. If the INCLUDING IDENTITY COLUMN ATTRIBUTES clause is specified with the AS select-statement clause, the columns of the new table do not inherit the identity attribute in the following cases:

- The select list of the select-statement includes multiple instances of an identity column name (that is, selecting the same column more than once).
- The select list of the select-statement includes multiple identity columns (that is, it involves a join).
- The identity column is included in an expression in the select list.
- The select-statement includes a set operation (union).

If INCLUDING IDENTITY is not specified, the table will not have an identity column.

EXCLUDING IDENTITY COLUMN ATTRIBUTES specifies that the table does not inherit the identity attribute, if any, of the columns resulting from the select-statement, table-name, or view-name.

INCLUDING COLUMN DEFAULTS specifies that the table inherits the default values of the columns resulting from the select-statement, table-name, or view-name. A default value is the value assigned to a column when a value is not specified on an INSERT. Do not specify INCLUDING COLUMN DEFAULTS, if you specify USING TYPE DEFAULTS. If INCLUDING COLUMN DEFAULTS is not specified, the default values are not inherited.

EXCLUDING COLUMN DEFAULTS specifies that the table does not inherit the default values of the columns resulting from the select-statement, table-name, or view-name.

USING TYPE DEFAULTS specifies that the default values for the table depend on the data type of the columns that result from the select-statement, table-name, or view-name. If the column is nullable, then the default value is the null value.

Note: Using interactive SQL to perform a CREATE TABLE with a subselect does not support prompting.
DECLARE GLOBAL TEMPORARY TABLE

Allows creation of a temporary table in the default SESSION schema

Table is not in the catalog

Cannot be shared by other jobs, only by threads or processes within the same job environment

Example:

- DECLARE GLOBAL TEMPORARY TABLE session.currency
  (cur_id char(3),
  current_rate decimal(9,2))
ON COMMIT PRESERVE ROWS
Notes: DECLARE GLOBAL TEMPORARY TABLE

The DECLARE GLOBAL TEMPORARY TABLE statement defines a declared temporary table for the current application process. The declared temporary table description does not appear in the system catalog. It is not persistent and cannot be shared with other sessions. Each session that defines a declared global temporary table of the same name has its own unique description of the temporary table. When the application process ends, the temporary table is dropped.

Additional syntax:

**WITH REPLACE** specifies that, in the case that a declared global temporary table already exists with the specified name, the existing table is replaced with the temporary table defined by this statement (and all rows of the existing table are deleted). When WITH REPLACE is not specified, then the name specified must not identify a declared global temporary table that already exists in the current session.

**ON COMMIT/ON ROLLBACK** specifies the action taken on the global temporary table when a COMMIT or ROLLBACK operation is performed. The ON COMMIT or ON ROLLBACK clause does not apply if the declared global temporary table is opened under isolation level No Commit (NC) or if a COMMIT HOLD operation is performed.

- **DELETE ROWS**: all rows of the table will be deleted if no WITH HOLD cursor is open on the table. This is the default.
- **PRESERVE ROWS**: rows of the table will be preserved

**NOT LOGGED**: changes to the table are not logged, including creation of the table. When a ROLLBACK (or ROLLBACK TO SAVEPOINT) operation is performed and the table was changed in the unit of work (or savepoint), the changes are not rolled back. If the table was created in the unit of work (or savepoint), then that table will be dropped. If the table was dropped in the unit of work (or savepoint) then the table will be restored, but with no rows.

Any program in an application process can reference a declared global temporary table, and any of those references is a reference to that same instance of that table. If a DECLARE GLOBAL TEMPORARY statement is specified within a compound statement of an SQL function, SQL procedure, or trigger; the scope of the declared temporary table is the application process and not the compound statement. If a declared global temporary table was declared at a remote server, the reference to that table must use the same connection that was used to declare it and that connection must not have been terminated after the table was declared. When the connection to the database server at which the table was declared terminates, it is dropped, and its instantiated rows are destroyed.

The table will be placed by default in the SESSION schema; internally, it is stored in the QTEMP library of the job; the joblog will also reflect the creation of the table in QTEMP.
User-defined Table Functions

Allows a user to create a function (external or SQL) that returns a table

- Provides compatibility with other DB2 products
- Provides ability for the user and 3rd parties to write their own table functions
- Provides function overloading and polymorphism

Example:

```sql
CREATE FUNCTION READ_Customer ( )
RETURNS TABLE
  (CSNRA DECIMAL(9,0), CSNMA CHAR(30), ADNRA CHAR(3), NAMEA CHAR(30))
LANGUAGE SQL
SPECIFIC READ_CUSTOM
NOT DETERMINISTIC
READS SQL DATA
CALLED ON NULL INPUT
NO EXTERNAL ACTION
DISALLOW PARALLEL
NOT FENCED
CARDINALITY 160
RETURN SELECT CSNR, CSNM, ADNR, NAME FROM CUSTOM
```
Notes: User-defined Table Functions

This CREATE FUNCTION (External Table) statement creates a function (external or SQL) at the current server. The function returns a result table. A table function may be used in the FROM clause of a SELECT, and returns a table to the SELECT by returning one row at a time.

Polymorphism refers to the ability to have the function perform differently based on the arguments passed into it. It is very tightly related to overloading which is having several functions with the same name each performing different functions. You can create several function with the same name and depending on the type or number of parameters passed, the database determines which function to invoke.

General remark for creating functions:
In V5R1 the parameter FENCED or NOT FENCED on the CREATE FUNCTION command was allowed for compatibility with other products and not used by DB2 UDB for iSeries; by default a function ran always FENCED.

Starting with V5R2, when creating a User Defined Function (UDF), you need to consider whether to make the UDF an Unfenced UDF. By default, UDFs are still created as Fenced UDFs. Fenced indicates that the database should run the UDF in a separate thread. For complex UDFs, this separation is meaningful as it will avoid potential problems such as generating unique SQL cursor names. Not having to be concerned about resource conflicts is one reason to stick with the default and create the UDF as a fenced UDF. A UDF created with the 'NOT FENCED' option indicates to the database that the user is requesting that the UDF can run within the same thread that initiated the UDF. Unfenced is a suggestion to the database, which could still decide to run the UDF in the same manner as a Fenced UDF.
iSeries Navigator for User-defined Table Functions
Notes: iSeries Navigator for User-defined Table Functions

This foil illustrates the iSeries Navigator interface to the user-defined table function support.
CREATE PROCEDURE

Accepts OLD SAVEPOINT/NEW SAVEPOINT clause

Accepts COMMIT ON RETURN

iSeries Navigator interface
Notes: Create Procedure

CREATE PROCEDURE supports the OLD SAVEPOINT LEVEL or NEW SAVEPOINT LEVEL and COMMIT ON RETURN. This foil shows an iSeries Navigator interface for these parameters.

OLD SAVEPOINT, NEW SAVEPOINT specifies whether a new savepoint level is to be created upon entry to the procedure.

- **OLD SAVEPOINT LEVEL:**
  A new savepoint level is not created. Any SAVEPOINT statements issued within the procedure with OLD SAVEPOINT LEVEL implicitly or explicitly specified on the SAVEPOINT statement are created at the same savepoint level as the caller of the procedure. This is the default.

- **NEW SAVEPOINT LEVEL:**
  A new savepoint level is created on entry to the procedure. Any savepoints set within the procedure are created at a savepoint level that is nested deeper than the level at which this procedure was invoked. Therefore, the name of any new savepoint set within the procedure will not conflict with any existing savepoints set in higher savepoint levels (such as the savepoint level of the calling program) with the same name.

COMMIT ON RETURN specifies whether the database manager commits the transaction immediately on return from the procedure.

- **NO:**
  The database manager does not issue a commit when the procedure returns. NO is the default.

- **YES:**
  The database manager issues a commit if the procedure returns successfully. If the procedure returns with an error, a commit is not issued. The commit operation includes the work that is performed by the calling application process and the procedure. If the procedure returns result sets, the cursors that are associated with the result sets must have been defined as WITH HOLD to be usable after the commit. If the external program was created with ACTGRP(*NEW) and the job commitment definition is not used, the work that is performed in the procedure will be committed or rolled back as a result of the activation group ending.

The following keywords are synonyms supported for compatibility to prior releases. These keywords are non-standard and should not be used:

- The keywords VARIANT and NOT VARIANT can be used as synonyms for NOT DETERMINISTIC and DETERMINISTIC.
- The keywords NULL CALL and NOT NULL CALL can be used as synonyms for CALLED ON NULL INPUT and RETURNS NULL ON NULL INPUT.
- The keywords SIMPLE CALL can be used as a synonym for GENERAL.
- The value DB2GENRL may be used as a synonym for DB2GENERAL.
- DYNAMIC RESULT SET, RESULT SETS, and RESULT SET may be used as synonyms for DYNAMIC RESULT SETS.
Debug and Packaging Improvements

SET OPTION DBGVIEW=*SOURCE allows SQL procedures, triggers, and functions to be debugged without having to view the generated C code

- DBGVIEW(*SOURCE) can be used on ILE programs with embedded SQL code
- Lays the ground work for the DB2 graphical debugger

SQL procedures, triggers, and functions can be created without purchasing and installing DB2 Query Manager and SQL Development Kit, 5722-ST1

- V5R1: C compiler not required for SQL application development
- V5R2: Eliminates need for 5722-ST1
- Enables easier deployment of SQL triggers - previously a trigger had to be recreated on the production system

New SET SCHEMA statement controls current library via SQL interface (Dynamic SQL only)

- Example: SET SCHEMA = PRODLIB (similar to CL command CHGLIBL CURLIB(PRODLIB)
- SET SQLID treated as a synonym for DB2 compatibility
Notes: Debug and Packaging Improvements

By specifying SET OPTION DBGVIEW = *SOURCE in your Create SQL Procedure, Create SQL Function, or Create Trigger statement, you can debug the generated program or module at the SQL statement level. You can also specify DBGVIEW(*SOURCE) as a parameter on a RUNSQLSTM command and it will apply to all routines within the RUNSQLSTM.

The source view will be created by the system from your original routine body into QTEMP/QSQDSRC. The source view is not saved with the program or service program. It will be broken into lines that correspond to places you can stop in debug. The text, including parameter and variable names, will be folded to uppercase.

When you want to create a stored procedure, trigger or function you no longer need to have the SQL Developer Kit (5722-ST1) installed on your system. As of V5R1, the requirement to have the C compiler (5763-CX1) installed had already been dropped.
Availability, Recovery Enhancements: Savepoints

Faster transaction recovery with database Savepoints

- Instead of starting over from the beginning of related transactions, start from a known "interim" step

ROLLBACK TO <savepoint> rolls back changes only to the specified savepoint instead of all changes made by the transaction

- RELEASE SAVEPOINT statement deletes a savepoint
- COMMIT or ROLLBACK

Savepoints in a distributed transaction are scoped to the current connection
Notes: Availability, Recovery enhancements: Savepoints

The Savepoint support enables an application to setup savepoints as sub-transactions of a larger transaction - or as part of a set of related transactions. This enables an application to be able to "restart" at a known "interim" step. This capability becomes more important as applications become more interdependent and use more than one system.

The SAVEPOINT statement sets a savepoint within a unit of work to identify a point in time within the unit of work to which relational database changes can be rolled back. The SAVEPOINT statement includes control of cursor positioning with in the tables in the unit of work, as well as control of any "object locks" associated with the "sub-transaction."

Syntax:

```
SAVEPOINT savepoint-name
 UNIQUE
 ON ROLLBACK RETAIN CURSORS
 ON ROLLBACK RETAIN LOCKS
```

**UNIQUE** specifies that the application program cannot reuse the savepoint name within the unit of work. An error occurs if a savepoint with the same name as savepoint-name already exists within the unit of work. Omitting UNIQUE indicates that the application can reuse the savepoint name within the unit of work. If savepoint-name identifies a savepoint that already exists within the unit of work and the savepoint was not created with the UNIQUE option, the existing savepoint is destroyed and a new savepoint is created. Destroying a savepoint to reuse its name for another savepoint is not the same as releasing the savepoint. Reusing a savepoint name destroys only one savepoint. Releasing a savepoint with the RELEASE SAVEPOINT statement releases the savepoint and all savepoints that have been subsequently set.

**ON ROLLBACK RETAIN CURSORS** specifies that cursors that are opened after the savepoint is set are not closed upon rollback to the savepoint. If the savepoint contains DDL on which a cursor is dependent, the cursor is closed. Attempts to use such a cursor after a ROLLBACK TO SAVEPOINT results in an error. Otherwise, the cursor is not affected by the ROLLBACK TO SAVEPOINT (it remains open and positioned). Although these cursors remain open after rollback to the savepoint, they might not be usable. For example, if rolling back to the savepoint causes the insertion of a row on which the cursor is positioned to be rolled back, using the cursor to update or delete the row results in an error.

**ON ROLLBACK RETAIN LOCKS** specifies that any locks that are acquired after the savepoint is set are not released on rollback to the savepoint.
Notes: Availability, Recovery enhancements: Savepoints -2

The RELEASE SAVEPOINT statement releases the identified savepoint and any subsequently established savepoints within a unit of work.

Syntax:

.-TO-.  

>>RELEASE--+++++++SAVEPOINT--savepoint-name------------------><

If the named savepoint does not exist, an error occurs. The named savepoint and all the savepoints that were subsequently established in the unit of work are released. After a savepoint is released, it is no longer maintained and rollback to the savepoint is no longer possible. The name of the savepoint that was released can be reused in another SAVEPOINT statement, regardless of whether the UNIQUE keyword was specified on an earlier SAVEPOINT statement specifying this same savepoint name.

A RELEASE SAVEPOINT statement is not allowed if commitment control is not active for the OS/400 activation group.
V5R2 SQL Call Level Interface Enhancements

New Functions:
- SQLTablePrivileges
- SQLColumnPrivileges
- SQLNextResult

SQLColAttributes returns additional column attributes in a result set:
- SQL_COLUMN_Auto_Increment
- SQL_COLUMN_Searchable
- SQL_COLUMN_Updatable
- SQL_COLUMN_Base_Column
- SQL_COLUMN_Base_Table
- SQL_COLUMN_Base_Schema
- SQL_COLUMN_Label

Over 100 new constants added to improve porting activities

New data type constant: SQL_C_SLONG

Support for externally-managed XA transactions

SQLGetInfo enhancements:
- SQL_String_Functions
- SQL_Numeric_Functions
- SQL_Aggregate_Functions
- SQL_Convert_Functions
- SQL_TimeDate_Functions
- SQL_SQL_Conformance
- SQL_SQL92_Predicates
- SQL_SQL92_Value_Expressions
- SQL_ODBC_API_Conformance
- SQL_ODBC_SQL_Conformance
- SQL_Qualifier_Usage
- SQL_Qualifier_Location
- SQL_Owner_Usage
- SQL_Correlation_Name
- SQL_Identifier_Case
- SQL_TXN_Capable
- SQL_Default_TXN_Isolation
- SQL_Group_By
- SQL_NoN_Nullable_Columns

UTF-8 via CLI only
The SQL Call Level Interface (CLI) gets many specific enhancements in V5R2, including support for the externally managed XA transactions ("adaptive e-transaction services").

SQLGetInfo() returns general information, (including supported data conversions) about the DBMS that the application is currently connected to.

See Information Center on the Call Level Interface functions. Search with **+DB2 +UDB +CLI**. You should find the overview topic on the second window of search results.
Miscellaneous

Removal of initial restrictions of scalar subselect:

- SELECT deptno, (t1.c1 +
  (SELECT t2.c1 FROM t2 WHERE t1.c2 = t2.c2 AND t2.c3 >0)) AS sel1
  FROM t1
  WHERE t1.c3 >0
  ORDER BY sel1

UNION in Views:

- CREATE VIEW v1
  AS SELECT * FROM t1
  UNION
  SELECT * FROM t2

READ ONLY and READ WRITE in SET TRANSACTION

ORDER BY expression not required in the select-list

HOLD LOCATOR:

- LOB locator variable retains its association with a value beyond a unit of work

EVI Maintenance improvement (less system resource)
Notes: Miscellaneous

Scalar Subselect is allowed anywhere an expression is allowed:
- CASE WHEN and result, ex: CASE (SS1) WHEN (SS2) THEN (SS3) END
- CAST expression, ex: CAST((SS1) as INTEGER)
- BASIC predicate, either operand, ex: (SS1) = (SS2)
- Left operand of a Quantified predicate, ex: (SS1) = ANY (subselect)
- BETWEEN predicate - any operand
- left operand of an IN predicate, ex: (SS1) IN (subselect)
- IN set of values, ex: T1.C1 IN ( (SS1), (SS2),...)
- LIKE predicate - either operand and escape expression
- NULL predicate, ex: (SS1) IS NULL
- Column Function, ex: AVG(SS1)
- Scalar Function, ex: SUBSTR((SS1), 1,10)
- Labeled-duration, ex: (SS1) YEARS
- UDF parameter
- UDTF parameter
- As an operand in an expression, ex: (SS1) + 10
- Join ON clause

Any ORDER BY clause must appear after the last subselect that is part of the UNION.
Enhanced Catalog Support

New Catalog SYSIBM
Notes: Enhanced Catalog Support

The database manager maintains a set of tables containing information about the data in each relational database. These tables are collectively known as the catalog. The catalog tables contain information about tables, user-defined functions, distinct types, parameters, procedures, packages, views, indexes, aliases, constraints, triggers, and languages supported by DB2 UDB for iSeries. The catalog also contains information about all relational databases that are accessible from this system.

There are three classes of catalog views:

- **iSeries catalog tables and views**
  The iSeries catalog tables and views are modeled after the ANSI and ISO catalog views, but are not identical to the ANSI and ISO catalog views. These tables and views are compatible with prior releases of DB2 UDB for iSeries. These tables and views exist in schemas QSYS and QSYS2.
  The catalog tables and views contain information about all tables, parameters, procedures, functions, distinct types, packages, views, indexes, triggers, and constraints in the entire relational database. When an SQL schema is created, an additional set of these views (except SYSPARMS, SYSPROCS, SYSFUNCS, SYSROUTINES, SYSROUTINEDEP, and SYSTYPES) are created into the schema that only contain information about tables, packages, views, indexes, and constraints in that schema.

- **ODBC and JDBC catalog views**
  The ODBC and JDBC catalog views are designed to satisfy ODBC and JDBC metadata API requests. For example, SQLColumns. These views are compatible with views on DB2 UDB for OS/390 and z/OS and DB2 UDB UWO Version 8. These views will be modified as ODBC or JDBC enhances or modifies their metadata APIs. These views exist in schema SYSIBM.

- **ANS and ISO catalog views**
  The ANS and ISO catalog views are designed to comply with the ANS and ISO SQL standard (the Information Schema catalog views). These views are compatible with views on DB2 UDB UWO Version 8. These views will be modified as the ANS and ISO standard is enhanced or modified. These views exist in schema QSYS2 and SYSIBM.

There are several columns in these views that are reserved for future standard enhancements.
JDBC and V5R2

- Removal of 'FOR UPDATE' restriction
- Change in data truncation
- Get and modify columns and parameters by name
- Retrieve auto-generated keys
- Improved performance when running SQL insert statements in a batch
- Enhanced support for ResultSet.getRow()
- Improved support for using mixed cases in column names
- Specify holdability for Statements, CallableStatements, and PreparedStatements
- Enhanced transaction isolation support
Notes: JDBC and V5R2

This foil summarizes the JDBC enhancements available using V5R2. Refer to the JDBC articles in V5R2 Information Center for more details. iSeries has supported the JDBC driver for several releases. The JDBC driver is included in the iSeries Toolbox for Java which supports access to the iSeries database using APIs that are similar to ODBC, though the scope of function may not be exactly the same at the driver level for JDBC and ODBC.

JDBC or ODBC drivers communicate with the same server job on the iSeries (QZDASOINIT for non SSL and QZDASSINIT for SSL). ODBC and JDBC drivers have been enhanced over the last several releases. This foil discusses the V5R2 JDBC driver enhancements.

Removal of the 'FOR UPDATE' restriction
You no longer need to specify FOR UPDATE on your SELECT statements in order to guarantee an updatable cursor. When connecting to V5R1 and later versions of OS/400, Toolbox for Java honors whatever concurrency you pass in when you create statements. The default continues to be a read-only cursor if you do not specify a concurrency.

Data truncation throws exceptions only when truncated character data is written to the database
Data truncation rules for Toolbox for Java now are the same as those for the IBM Developer Kit for Java JDBC driver. For more information, see IBM Toolbox for Java JDBC properties in V5R2 Information Center.

Get and modify columns and parameters by name
New methods allow you to get and update information by column name in ResultSet and to get and set information by parameter name in CallableStatement. For example, in ResultSet, where you previously used the following:

\[
\text{ResultSet } rs = \text{statement.executeQuery( SELECT * FROM MYCOLLECTION/MYTABLE );}
\]
\[
\text{rs.getString(1);} 
\]
You can now use:

\[
\text{ResultSet } rs = \text{statement.executeQuery( SELECT * FROM MYCOLLECTION/MYTABLE );}
\]
\[
\text{rs.getString('STUDENTS');} 
\]

Be aware that accessing parameters by their index results in better performance than accessing them by their name. You can also specify parameter names to set in CallableStatement. Where you might have used the following in CallableStatement:

\[
\text{CallableStatement } cs = \text{connection.prepareCall( CALL MYPGM (?) );}
\]
\[
\text{cs.setString( 1 );} 
\]
You can now use:

```java
CallableStatement cs = connection.prepareCall("CALL MYPGM (?)");
cs.setString("PARAM_1");
```

To use these new methods, you need JDBC 3.0 or later and the Java 2 Platform, version 1.4 (either the Standard or the Enterprise Edition).

**Retrieve auto-generated keys**

The `getGeneratedKeys()` method on `AS400JDBCStatement` retrieves any auto-generated keys created as a result of executing that `Statement` object. When the `Statement` object does not generate any keys, an empty `ResultSet` object is returned. Currently the server supports returning only one auto-generated key (the key for the last inserted row). The following example shows how you might insert a value into a table then get the auto-generated key:

```java
Statement s = statement.executeQuery("INSERT INTO MYSCHOOL/MYSTUDENTS (FIRSTNAME) VALUES ('JOHN')");
ResultSet rs = s.getGeneratedKeys();
// Currently the iSeries server supports returning only one auto-generated
// key -- the key for the last inserted row.
rs.next();
String autoGeneratedKey = rs.getString(1); // Use the auto-generated key, for example, as the primary key in another table
```

To retrieve auto-generated keys, you need JDBC 3.0 or later, and the Java 2 Platform, version 1.4 (either the Standard or the Enterprise Edition). Retrieving auto-generated keys also requires connecting to a OS/400 V5R2 or later.

**Improved performance when running SQL insert statements in a batch**

Performance of running SQL insert statements in a batch has been improved. Run SQL statements in a batch by using the different `addBatch()` methods available in `AS400JDBCStatement`, `AS400JDBCPreparedStatement`, and `AS400JDBCCallableStatement`. Enhanced batch support affects only insert requests. For example, using batch support to process several inserts involves only one pass to the server. However, using batch support to process an insert, and update, and a delete sends each request individually.

To use batch support, you need JDBC 2.0 or later and the Java 2 Platform, version 1.2 (either the Standard or the Enterprise Edition).
**Notes: JDBC and V5R2 -3**

**Enhanced support for ResultSet.getRow()**
Previously, the IBM Toolbox for Java JDBC driver was limited in its support for the `getRow()` method in ResultSet. Specifically, using `ResultSet.last()`, `ResultSet.afterLast()`, and `ResultSet.absolute()` with a negative value made the current row number not available. The previous restrictions are lifted, making this method fully functional.

**Using mixed case in column names**
Toolbox for Java methods must match either column names provided by the user or column names provided by the application with the names that are on the database table. In either case, when a column name is not enclosed in quotes, Toolbox for Java changes the name to uppercase characters before matching it against the names on the server. When the column name is enclosed in quotes, it must exactly match the name on the server or Toolbox for Java throws an exception.

**Specify holdability in created Statements, CallableStatements, and PreparedStatements**
New methods in AS400JDBCConnection allow you to specify the holdability for Statements, CallableStatements, and PreparedStatements that you create. Holdability determines whether cursors are held open or closed when committing the transaction. You can now have a statement that has a different holdability than its connection object. Also, connection objects can have multiple open statement objects, each with a different specified holdability. Calling commit causes each statement to be handled according to the holdability specified for that statement.

Holdability is derived in the following order of precedence:

1. Holdability specified on statement creation by using the Connection class methods `createStatement()`, `prepareCall()`, or `prepareStatement()`.
2. Holdability specified by using `Connection.setHoldability(int)`.
3. Holdability specified by the Toolbox for Java JDBC cursor hold property (when methods in 1. or 2. are not used)

To use these methods, you need JDBC 3.0 or later, and the Java 2 Platform, version 1.4 (either the Standard or the Enterprise Edition). Also, servers running a V5R1 or earlier version of OS/400 are able to use only the holdability specified by the JDBC cursor hold property.

**Enhanced transaction isolation support**
The IBM Toolbox for Java JDBC driver now features support for switching to a transaction isolation level of *NONE after a connection is made. Prior to V5R2, the Toolbox for Java JDBC driver threw an exception when switching to *NONE after making a connection.
**ODBC and V5R2**

- Ability to send Structured Query Language (SQL) statements that are 64K bytes long to the DB2 UDB database (the previous limit was 32K bytes)
- Ability to make use of the DB2 UDB database type of ROWID
- Ability to get back additional descriptor information, such as the base table name for a result set column
- Ability to access multiple databases on the same iSeries server
- Ability to retrieve meaningful information from the SQLTablePrivileges and SQLColumnPrivileges APIs
- Ability to use Kerberos support for authenticating a user to an iSeries server
- Ability to retrieve, regardless of the iSeries server version, more information in the result sets for the catalog
- Ability to retrieve, regardless of the iSeries server version, more information in the result sets for the catalog APIs. The driver now queries the iSeries catalog tables directly to provide the result set for the catalog APIs.
- ODBC to iSeries Linux drivers for Intel and Power4 based workstations
Notes: ODBC and V5R2

See V5R2 Information Center for additional ODBC details.

See also the detailed iSeries Access presentation for more information on the Linux ODBC drivers for iSeries.
iSeries Navigator and Database
**iSeries Navigator and Database V5R2 topics**

SQL Identity Column, User-defined Table function

SQL Assist, including expression builder

Manage new for V5R2 "database table statistics"

SQL Explain

- "Print SQL Information"
- Enhanced Index Advisor

Run SQL Scripts - OUT parameters displayed

Database Navigator enhancements

Database Transaction Management

Additional on-line help information
Notes: iSeries Navigator and Database V5R2 topics

This foil lists the V5R2 iSeries Navigator -> Database functions.

A few foils earlier discussed Identity Column, User-defined Table function support, which included examples of the iSeries Navigator Database interface to these functions.

When using Run SQL Scripts and selecting to run a stored procedure, you can now see any OUT parameters. The OUT parameters show via the Messages tab. The OUT parameter name is used as a label to the parameter value shown.

The following foils provide additional examples on the other topics listed on this foil.
Run SQL Script - the Scripting Center: Assist

GUI Interface to construct SQL statements: SQL Assist

- Select
- Insert
- Update
- Delete
Notes: Run SQL Script - the Scripting Center: Assist

Prior to V5R2, you selected the Run SQL Scripts menu option by right clicking the Database folder. In V5R2, even with only a single local database, under the iSeries Navigator -> Database folder, you right click on the database name to get the corresponding context menu that includes the Run SQL Scripts function. In this example we show two databases - the standard local one Rchase5C and Dbcook, which is in a varied on IASP.

Run SQL Scripts now includes SQL Assist functions which bring 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 to building basic and more complex SQL statements.

You invoke SQL Assist from Run SQL Scripts center menu item Edit --> Insert Built SQL, or you can 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.

The next two foils illustrate usage of the iSeries Navigator assistance in building an SQL statement - more powerful than the 5250 STRSQL interface.
In this example you see the first assistance window after taking the Insert Build SQL menu option (or SQL Prompt menu option if you have already entered an SQL SELECT statement) with the SELECT statement preselected. Many options are available to assist you.

For example, clicking the folder FROM (Source tables) will prompt you with the libraries currently shown on the left iSeries Navigator pane under Database. Selecting the library brings up the window in which you can select table/physical file. Then select the table name and then select SELECT (Result columns) to be able to choose which columns you want to include in your query.

See the next foil.
Notes: SQL Assist - Table and Columns

In this example we first selected FROM (Source tables), then expanded the libraries currently listed under the database folder, and selected library PFREXP and table CSTMST (steps not shown as indicated with the "dashed arrow").

In the window shown we have selected four of the columns in table CSTMST. In the SQL code area you can see the current SELECT statement built so far.

Note: You can check SELECT DISTINCT to eliminate duplicate rows, undo any editing done so far, check syntax on the statement so far, and run the statement.

In the next foil we show an example of using SQL Assist for the WHERE (Row filter) "statement properties" and the assist for the Expression function supported for WHERE - the Expression Builder function under iSeries Navigator -> Database -> Run SQL Scripts..
Notes: Expression Builder

To get the windows shown in this foil (Expression Builder - Columns), we did the following in a window not shown:

- Chose WHERE (Row filter)
- In the details pane, selected the equal (=) operator, and chose Expression from a Values list box

Here you can see the ability to enter or chose, Operators, Case, Values, Constants.

We chose the COUNT function which brought up the lower right window where we specified to do a count on the column CSTMST.CSTNBR.

After clicking OK, the SQL statement shows the syntax for performing count on the CSTNBR column.

This new SQL Assist function has many more ways to help you generate a syntactically correct SQL statement that we cannot show in this presentation.
Statistics Manager

OS/400 keeps information about the usage of tables and indexes; counters are reset only at IPL time.

Database statistics information provides information to the OS/400 the Query Optimizer.*

New system value *Database file statistics collection*(QDBFSTCCOL) allows you to specify how the collection task will run (**NONE - **USER - **SYSTEM - **ALL)

Management of statistical collection can be done via iSeries Navigator.

* The Query Optimizer makes minimal use of the new statistics data at V5R2 general availability.
Notes: Statistics Manager

Statistical information and other factors can be used by the query optimizer to determine the best access plan for a query. To be of value, this statistical information must be accurate and complete. Since the query optimizer bases its choice of access plan on the statistical information found in the table, it is important that this information be current. On many platforms, statistics collection is a manual process that is the responsibility of the database administrator. With iSeries servers, the database statistics collection process is handled automatically, and it is rarely necessary for the statistics to be manually updated, even though it is possible to manage statistics manually.

In this release, the database statistics function of iSeries Navigator gives you the ability to manage statistical information for a table.

With the Database file statistics collection (QDBFSTCCOL) system value you specify the types of statistics collection requests that are allowed to be processed in the background by the database statistics system job, QDBFSTCCOL. Statistical collections which are requested by either a user or automatically by the database manager to be processed in the foreground are not affected by this system value. When this system value is changed to a more restrictive value, background statistic collections in progress in the statistics system job, which are not allowed at the new value, will be ended. Their processing will be restarted when the system value is changed back to a value which would allow their processing. A change to this system value takes effect immediately.

The shipped value is *ALL. However, you must specifically identify the tables for which you want the system to collect statistics.

Note: The Query Optimizer makes minimal use of the new statistics data at V5R2 general availability. Current plans are for more intensive use of this information to become available in future database enhancements. This support is described here because the statistics support is included in V5R2 iSeries Navigator on-line help information.

If you decide to collect statistics manually, and you set the statistics in iSeries Navigator to be maintained manually, not allowing the system to perform automatic updates, or if you want to speed up the automatic update process, then statistics should be updated when:

- A table is loaded or reorganized.
- A significant number of rows have been inserted, updated, or deleted.
- A new column has been added to the table.
- The Statistics Advisor in Visual Explain recommends that statistics should be created or updated.

This option to reset statistics comes in very helpful when loading tables onto a DB2 UDB for iSeries using an SQL interface that uses a access plan, as contained in a SQL package or an SQL program or procedure; initially, when the table is empty, the Query Optimizer may have chosen a table scan, while after a number of updates, an appropriate index might prove more effective.
Statistics Manager - Invocation

Statistics Request

Statistical Data
Notes: Statistics Manager - Invocation

Statistics Request
The upper left window is used to view collection status and control the types of statistics collection through iSeries Navigator - corresponding to the system value QDBFSTCCOL.

Statistical Data
By right clicking a table/file you can selectively specify which columns to collect data for - as shown in the next foil.
### Statistic Data - Interface

![Image of the interface showing statistic data](image)

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Column</th>
<th>Type</th>
<th>Length</th>
<th>Stale</th>
<th>Aging</th>
<th>Requester</th>
<th>Transl</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDBST_AC39E800...</td>
<td>ADNR</td>
<td>CHAR</td>
<td>3</td>
<td>No</td>
<td>Automatic</td>
<td>ITSCID18</td>
<td></td>
</tr>
<tr>
<td>QDBST_AC3A2000...</td>
<td>CSNM</td>
<td>CHAR</td>
<td>30</td>
<td>Yes</td>
<td>Automatic</td>
<td>ITSCID18</td>
<td></td>
</tr>
<tr>
<td>QDBST_AC3A3800...</td>
<td>CSNR</td>
<td>DECIMAL</td>
<td>5</td>
<td>Yes</td>
<td>Automatic</td>
<td>ITSCID18</td>
<td></td>
</tr>
</tbody>
</table>
Notes: Statistic Data - Interface

From the Statistics Data dialog you can view statistical data for the columns in the selected table. The data presented here can be used to decide whether to collect more statistics, delete existing statistics for a column, or to view more detailed column statistics data.

This list contains the statistics that were created manually or automatically. Columns without statistics do not appear in the list.

- **Statistic Name** provides an identifying name for the statistic. This name is assigned when a new statistic is created.
- **Column** provides the name of the column over which this statistic is collected.
- **Type** provides information on the type of column (for example, INTEGER or CHARACTER, or other column types)
- **Length** specifies the number of characters, number of digits, or file size of the value that the function returns.
- **Stale** provides a way to determine if the column's statistics are credible. A Yes in this column would indicate that the statistics exist, but that they should be updated.
- **Aging** indicates whether the system is responsible for maintaining the statistics for a column or if the user is in control of the statistics collection. Manual indicates that the user decides when to update statistics for a column. Automatic indicates that the system determines when statistics are updated for a column.
- **Requester** provides the name of the requester (user profile) that requested statistics to be created or updated. System indicates that the system requested the create or update, while a specific name indicates the user profile of the person who requested the create or update.
- **Translation table** provides the name of the translation table if one was used for the character columns of the statistics collection. Otherwise, the translation table name will be empty.

The Block Collection button controls enabling or disabling the automatic collection of statistics for the entire table. When block is selected, you can still manually generate statistics for the table.
**Statistics data: column example**

![Statistics data example](image)

- **Statistic Name**: QDBST_3258C000...
- **Column**: ADRES2
- **Type**: CHAR
- **Length**: 40
- **Stale**: Yes
- **Aging**: Automatic
- **Requester**: COOK

**Statistic Data Details**

- **Statistic name**: QDBST_3258C00083481849B7620004AC0177CB
- **Statistic last collected**: 7/24/02 12:16:40 PM
- **Requester**: COOK
- **Statistic created**: 7/24/02 12:16:40 PM
- **Creator**: COOK
- **Estimate of cardinality**: 8640
- **Column nullable**: No
- **Number of nulls**: 0
- **Number of rows when collected**: 10000
- **Current rows in table**: 31010
- **Inserts, updates and deletes when collected**: 39422799
- **Current inserts, updates and deletes**: 39855901
- **Translation**: No translation
- **Translation table**: 
- **Total space used for all statistics**: 20.0 KB

**Age statistics manually**
Notes: Statistics data: column example

Although the statistical manager data is generally not used by the Query Optimizer in V5R2, a person with database expertise may find the information recorded useful, for example in understanding the frequency of updates, inserts, and deletes.

In this example we show the "general statistics" for column ADRS2.

In the General details you can see the estimate of unique column values (estimate of cardinality), the difference between the number of rows, inserts, updates, and deletes since the statistics were gathered and the current moment in time.

See the on-line help text for additional descriptive information.
Visual Explain Enhancements

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

Explain SQL for a trigger while viewing properties of a table

Explain a program from IFS view:

- Table, program, procedure, SQL package, or function
Notes: Visual Explain Enhancements

You can now do a print preview of the Visual Explain graphical output so you can adjust the view before printing it.

New with V5R2 you can ask for "basic" graph details and attributes or have advanced ("full") details and attributes shown.

Prior to V5R1 Visual Explain had "index advised' functions. In V5R2 the index advisor interface is improved and associated with the Statistics Advisor which has modest function with V5R2. Following foils give a view of the V5R1 Index Advisor function.

When viewing a table or a program, procedure, SQL package, or function object from the iSeries Navigator IFS view you can select to perform an Explain function.. See the following foils for some Explain examples.
**SQL Explain - SQL Package Support**

Equivalent of PRTSQLINF command

```
C1
CLOSQLCSR("ENDACTGRP")
RDB("LOCAL")
DATFMT("ISO")
TIMFMT("ISO")
DFTRDBCOL("NONE")
DYNDFTCOL("NO")
SQLPKG(DBENHIREAD_00001)
ALWBLK("ALLREAD")
DLYPRP("YES")
DYNUSRPRF("USER")
SRTSEQ("HEX")
LANGID(ENU)
RDBCNNMTH("DMW")
TEXT("SQL ROUTINE ABC")
STATEMENT TEXT CCSID(37)
SQLPATH("QSYS"||"QSYS2"||"ITSCID18")
DECLARE SQL_TABLE_CURSOR CURSOR FOR SELECT CSNR,CSNM,ADM
.CUSTOM FOR READ ONLY
SQL5065 Access plan not found.
OPEN SQL_TABLE_CURSOR
CLOSE SQL_TABLE_CURSOR

***** END OF LISTING *****
```
While using the iSeries Navigator view of all IFS objects, right-click a program, an SQL package, procedure, or function object and select Explain from the context menu. This produces output similar to the output of the OS/400 command PRTSQLINF (Print SQL Information). This information can assist an experienced SQL programmer to understand if the current "access plan" would perform satisfactorily.

New in V5R2 you can use this Explain function for a Trigger while viewing a table's properties.
Visual Explain - Index Advisor

```
SELECT * FROM PART WHERE P_PARTKEY IN ( SELECT L_PARTKEY FROM LINEITEM WHERE L_DISCOUNT = 0 )
```
Notes: Visual Explain - Index Advisor

This example shows a sample Visual Explain graphic. In V5R1 you could see an "advised index set of columns" by:
Selecting a specific icon in the Visual Explain window - such as the circled temporary index on this foil. Then select the menu Actions -> Create Index
Selecting the menu View -> Highlight Index Advised and create an index based upon that advice

In V5R2 you can select the Statistics and Advisor icon as shown. This brings up the window on the next foil.
Visual Explain - Index Advisor -2
Notes: Visual Explain - Index Advisor -2

The new Statistics and Index Advisor window is shown in the upper right background. In this example, of all the possible indexes that could be created the one recommended by the system is shown. In this upper right window we see LINEITEM table name columns recommended.

By selecting Create on that first window you get the lower left window which shows the recommend key columns already selected by default for you. You enter the new index name and library, as well as select the Index type and also select the number of distinct values supported.
Database Navigator

Show database object relationships

- Improved Map View
- Triggers can be viewed
- Print Map
Notes: Database Navigator

Database Navigator was introduced with V5R1 to show the relationships between related database objects - tables, views, indexes, and so forth. Triggers were not shown.

The V5R1 map viewer is completely replaced with the V5R2 map viewer. It is much easier to see all the related objects in a complex database "network of objects" on a single page.

The V5R2 print map function has the capability to print portions of the map in a manner such that you can print sections of the map and order each printed output in a right to left and/or top to bottom sequence. In a complex database map this makes viewing the map in hardcopy form much more useful.
Database Transaction Management

Database Transactions
- DB2 UDB - managed transactions

Global Transaction
- X/Open global transaction
Notes: Database Transaction Management

As previously described V5R2 delivers 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 the Transactions folder icon to do this.

There are two folders for you to select:
- **Database Transactions**: displays all transactions managed by DB2 UDB for iSeries
  - 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
- **Global Transactions**: displays all transactions associated with X/Open global transaction
  - 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

Functions available for a Database or Global Transaction (Distributed Unit of Work (DUW) in DRDA terminology) include:
- Force commit
- Force rollback
- Cancel resynchronization
- Display associated jobs
- Display resource status, including: Row, DB Object, DB Journal, SNA conversation, TCP/IP connection, Remote file, Remote DB, APIs

In this foil we show all the "active" Unit of Work IDs" that represent Database Transactions using commitment control. In this example you see many of the units of work associated with the pre-started QSQSRVR jobs (used by iSeries functions, including WebSphere Application Server applications and Management Central functions) and the ODBC/JDBC QZDASOINIT jobs. We also show a single-threaded 5250 job using commitment control (QPADEV0007/AS0310/011260). For that job we show Resource Status which indicates the current number of Commit and Rollback operations for files/tables CSTFIL and ITMFIL.
You can see the naming convention for Unit of Work ID ("transaction id") and the Unit of Work State.

In the iSeries Navigator for the Unit of Work Id Properties ton-line Help, the acronym LUW is used to represent a unit of work.

A commitment definition is established at each location that is part of the transaction program network. For each commitment definition, the system keeps track of the start of its current Logical Unit of Work (LUW) and previous LUW. The system uses the state to decide whether to commit or rollback if an LUW is interrupted. If multiple locations are participating in an LUW, the states of the LUWs at each location may be compared to determine the correct action (commit or rollback). This process of communicating between locations to determine the correct action is called resynchronization.

All of our LUWs show a state value of Reset. Reset means the period from the commitment boundary until a program issues a request to commit or rollback. Resynchronization is not needed. If interrupted, pending changes are rolled back by the system.

And, of course, on-line help is available for you to understand the various values for Unit of Work State.

At the time this presentation was created we did not have an example showing a global transaction - a distributed unit of work (DUW). See the Transaction Management foils at the beginning of this presentation for an overview of transactions and units of work.
iSeries Navigator Database: on-line help example

- Multiple databases
- Database switchover
- Database transactions
- .......

Work with multiple databases

With iSeries servers, you can work with multiple databases. The iSeries server provides a system database (often referred to as SYSBAS) and the ability to work with one or more user databases. User databases are implemented on the iSeries server through the use of independent disk pools, which are set up in the Disk Management function of iSeries Navigator. Once an independent disk pool is set up, it appears as another database under the Databases function of iSeries Navigator.

When you expand an iSeries server in iSeries Navigator and then expand Databases, a list of databases appears that you can work with. To establish a connection to a database, simply expand the database that you want to work with.

For detailed information on creating and working with user databases, see the following topics.

- Create a new database
- Delete a database
- Make a database available
- Make a database unavailable
DB2 DataPropagator for iSeries
V8.1
Usability Improvements

New Replication Center replaces Control Center and DataJoiner Replication Administration Tool

More control over capture process per registration and over recapturing data from replicas

Reorganization of Change Data and Unit-Of-Work tables
Notes: Usability Improvements

This section details significant enhancements in DataPropagator for iSeries V8.1. There are several other iSeries database products, especially those developed by Independent Software Vendors (ISVs) that have enhancements in the V5R2 general availability time frame. We suggest you visit the following website (listed at the end of this presentation):

http://www.ibm.com/servers/eserver/iseries/db2/

New User Interface

Now you can set up and maintain your replication environment, and operate the Capture and Apply programs, by using a much improved Graphic User Interface - the new Replication Center. It replaces the DJRA and the Control Center and it has the look and feel of the other DB2 centers. It also has a launch pad that organizes the basic functions needed to set up a DB2 replication environment. However, if you prefer to use the native commands to set up and maintain your environment (such as the Add DataPropagator Registration (ADDDPRREG) command), or to operate the programs (such as the Start DataPropagator Apply (STRDPRAPY) command), they are still available.

More control

When you register a table to capture only selected source table columns, you can now specify whether you want the Capture program to always produce a CD row or only when the registered columns change. In earlier versions, you could control what was captured for all registered tables only, not for individual registrations. When you register a source table, you can specify if you want changes recaptured from some tables but not others. By default, changes are not recaptured from replica tables and forwarded to other replica tables, and changes to master tables in update-anywhere replication are recaptured and sent to replica tables.

Reorganization of Change Data and Unit-Of-Work tables

When Capture pruning deletes unneeded UoW and CD rows, the disk space used by these deleted rows are not recovered automatically. A new keyword, RGZCTLTBL, is added to the End DataPropagator Capture (ENDDPRCAP) command to specify that the UoW table and all the active CD tables are reorganized to recover the space occupied by the deleted rows. This is only allowed during the controlled end. It takes longer to end capture if reorganize is requested. This is a significant usability improvement (previously you had to issue Reorganize Physical File Member (PGZPFM) commands to recover the disk space).
Performance Improvements

Fewer joins between Replication tables

Optimal memory allocation decision during Capture process

Fewer updates for subscription sets with multiple members

Apply program optimizes processing if it has only one subscription set
Notes: Performance Improvement

Fewer joins between Replication tables
In V8.1, the Apply program does not need to join the CD and UoW table to populate user-copy target tables under most circumstances. This time-consuming join can now be avoided due to a significant change in the Capture design. V8.1 Capture holds all the CD rows in memory until it processes the application's commit request. If the application does a rollback, the CD rows held in memory are discarded. In other words, the Capture program now writes only the committed transactions into the CD tables. Notice that if the Capture program cannot hold all the data for a transaction in memory, it resorts to the use of spill files. Also the CD and UoW tables do not have to be joined for pruning under most circumstances.

Optimal memory allocation decision during Capture process
To avoid inefficient memory usage and the ensuing use of spill files if memory limits are about to be exceeded, Capture keeps a history about the way applications update a source table. This history information allows Capture to make optimal decisions in its memory allocation algorithm and to minimize its use of spill files.

Fewer updates for subscription sets with multiple members
Compared to previous versions of DataPropagator, the V8.1 Apply program makes fewer updates to control tables for subscription sets with multiple members. The improvement is particularly noticeable where the Apply program runs frequently and has sets with a larger number of members.

Apply program optimizes processing if it has only one subscription set
In V8.1, you can start the Apply program so that it will cache and reuse information about a single subscription set. Using the new parameter (OPTSNGSET) when starting the Apply program improves CPU utilization and throughput rates.
New Functions

- Multiple sets of Capture Control tables
- Automated monitoring
- On-Demand monitoring
- Additional historical data in Control table
- Long table names and column names
- More frequent commits by the Apply program
- Referential Integrity for more types of Target tables
- Option for replicating changes to Target-Key columns
- Signals to control the Capture program
- More ways to set operational parameters for the Capture program
- Rule enforcement
- Migration Utility
Notes: New Functions

Multiple sets of Capture Control tables
In earlier versions, a user could create only one set of Capture control tables in the ASN library. In V8.1, you can create additional sets of Capture control tables. One reason to use an additional set of Capture control tables is to have multiple Capture environments in a single system. For example, the accounting department can have one Capture environment and the marketing department can have its own, both environments run independently of each other.

Automated monitoring
The new Replication Alert Monitor runs continuously and monitors the Capture and Apply programs for you. You define thresholds for criteria that you want to monitor, and specify people who should be contacted automatically via e-mail when those thresholds are met or exceeded. You can use the Replication Center to configure and operate the Replication Alert Monitor.

On-Demand monitoring
You can query the status of the Capture, Apply, and Monitor programs using the Replication Center.

Additional historical data in Control table
DataPropagator provides additional historical data in the Capture monitor (IBMSNAP_CAPMON) table, which describes replication activities.

Long table names and column names
DataPropagator now supports source table and target table names up to 128 characters, and column names up to 30 characters.

More frequent commits by the Apply program
In many situations, if you have user-copy, point-in-time, or replica target tables in a subscription set, you can specify that you want the Apply program to commit its work after it processes a specified number of transactions. To do so, you must run the Apply program in transaction mode.

Referential Integrity for more types of Target tables
In many situations, you can have referential integrity on user-copy and point-in-time target tables by starting the Apply program so that it commits its work in transaction mode.

Option for replicating changes to Target-Key columns
In V7.1, you could ensure that changes to key columns were replicated properly to your target tables by registering your source table to capture updates as delete/insert pairs. This option works fine except that more CD rows are generated as a result. In V8.1, you have a new option to address this problem. When you define a subscription set member, you can specify whether the Apply program should use the before-image values or the after-image values when it builds the WHERE clause using the primary key columns in its predicates. The use of the before-image value allows you to avoid the conversion of an update to an insert. If you choose to use this option instead, your registration should specify the before-image columns as well.
**Signals to control the Capture program**

The Capture program is now controlled by signals written to the signal table (IBMSNAP_SIGNAL). The signal table provides a way to communicate with the Capture program through a trigger on the signal table. Capture uses the signals to determine when to start capturing changes for a particular table (the CAPSTART signal), when to terminate (the STOP signal), and whether it must perform update-anywhere replication (the UPDANY signal).

**More ways to set operational parameters for the Capture program**

You can use the shipped defaults to operate the Capture program or you can create new defaults using the Change DataPropagator Capture Attribute (CHGDPRCAPA) command. You can also supply new operational parameters for the Capture program on the Start DataPropagator Capture (STRDPRCAP) command that are different from the default. When the Capture program is running, you can change the operational parameters by using the Override DataPropagator Capture Attribute (OVRDPRCAPA) command. The effect of this command lasts until you end the session or until you issue another OVRDPRCAPA command.

**Rule enforcement**

The source tables for the members of the same target set must use the same journal (this rule was not enforced in earlier versions). You will not be allowed to add a new member to an existing set if the source table of that member does not use the same journal as all the source tables of the other existing members.

**Migration Utility**

The V8.1 product provides a replication migration utility to allow you to migrate from earlier versions of DataPropagator (supported versions are 5769-DP3 and 5769-DP2). It also provides fall-back support as long as you never start either V8.1 Capture or Apply.
Database Migration Tools
Migration Tools - to DB2 UDB for iSeries

Conversion to DB2 UDB for iSeries

- Custom Technology Center Services, Rochester PartnerWorld assistance recommended
  - Oracle to DB2 Migration
  - SQLServer to DB2 Migration
  - Contacts:
    - Rochester PartnerWorld: Tom Grogan - tgrogan@us.ibm.com
    - Rochester Custom Technology Center: Mark Even - even@us.ibm.com

*IBM plans subject to change without notice
Notes: Migration Tools - to DB2 UDB for iSeries

The V5R2 enhancements make it easier to migrate databases from Oracle and SQLServer to iSeries. An Oracle to DB2 migration toolkit which can convert most (not ALL) of the oracle database elements to DB2 UDB for iSeries is planned for availability later this year.

The Rochester Custom Technology Center should be involved in any migration effort and can leverage the toolkits where necessary.

Porting guides for SQLServer migrations are also available through the DB2 UDB for iSeries home page.
Summary
**V5R2: Re-engineering the DB foundation**

### Adaptive e-transaction Services
- Extends iSeries robust transaction services to broader range of e-business application environments
- Further optimizes iSeries WebSphere, Java transaction workload performance
- Open standards support: XA, JTA (Java Transaction API)

### Database Technology Enhancements
- Incorporation of the latest optimization techniques, algorithms
- Updated object-oriented query optimizer that lays foundation for self-learning query optimizer
- Basis for streamlined data access for SQL interfaces in future releases
- Continued integration of industry-standard functions
Notes: V5R2: Re-engineering the DB foundation

At the beginning of this presentation we highlight the iSeries DB2 UDB strategic initiatives, focusing on industry-standard openness, leadership in database technologies, and continuing to bring new database functions and interfaces to iSeries without impacting existing application implementations.

This foil is used to summarize the V5R2 DB2 UDB for iSeries content that demonstrates delivering on these initiatives.
Appendix
V5R2: iSeries and DB2 Version 8.1

Announced July 2002 for NT, Unix, Linux
- IBM's autonomic computing initiative theme - self managing database
  - http://searchdatabase.techtarget.com/originalContent/0,289142,sid13_gci840473,00.html

DB2 UDB for iSeries is not V7.2 or V8.1 DB2 UDB
- iSeries has a broad range of SQL standard functions
- iSeries has many self-managing, self-monitoring, self-healing software facilities
  - Examples:
    - System performance metric and monitor - Database Capability, V5R2 File monitor
    - Query Optimizer determines when to update the current access plan, build a temporary index, ....

See DB2 UDB for iSeries Frequently Asked Questions document available at:
- IBM System Sales: w3.ibm.com/sales/systems/ibmsm.nsf
- PartnerInfo: www.ibm.com/partnerworld/sales/system
During July 2002, several press releases provided information on IBM's DB2 UDB V8.1 announcement - a new version of DB2 UDB for NT, Unix, and Linux.

The website http://searchdatabase.techtarget.com/originalContent/0,289142,sid13_gci840473,00.html contains commentary from analysts.

A major theme of DB2 UDB V8.1 focused on IBM's autonomic computing initiative, a self-managing, self-monitoring, self-healing software database. iSeries support functions are made available on OS/400 version and release boundaries rather than DB2 UDB version/release schedules - used by DB2 UDB for Unix/NT/Linux. The websites listed under System Sales and PartnerInfo can link you to a document that compares DB2 UDB functions on several operating systems, including OS/400.

OS/400 V5R1 and V5R2 as described in this presentation have many industry-standard and iSeries-unique functions - as well as several self-managing, self-monitoring, self-healing facilities.

For example, even before the announcement of DB2 UDB V8.1, the iSeries Query Optimizer has had the ability to adjust its existing access plan for processing table data based on the extent of changes to the related tables since the access plan was last rebuilt. Also any necessary catalog cross-reference information is updated as needed by the system.

On the monitoring side, before OS/400 V5R2, the System Monitor support under Management Central can be used to monitor and take actions based on thresholds on several metrics including database capability (CPU resource being consumed for database operations). Also, introduced in V5R2 is the Management Central File Monitor, which can be used to detect changes to or the existence of a table/file and the system History log.

The detailed Performance presentation describes the V5R2 File monitor support. The system, job, and message monitors have been available prior to V5R2 and are discussed in the iSeries Information Center.

See also IBM's autonomic computing initiative presentation in this series of V5R2 Technical Overview presentations.
Notes: V5R2: iSeries and DB2 Version 8.1 -2

This foil refers you to a website that can be used to discuss iSeries database support, including positioning DB2 UDB for iSeries as a self-managing database under IBM's autonomic computing initiative. The specific document is titled DB2 UDB for iSeries Frequently Asked Questions.

This document is available via System Sales and PartnerInfo.

Other DB2 UDB for iSeries related material on System Sales and PartnerInfo include:
- DB2 UDB for iSeries Overview (white paper)
- DB2 UDB for iSeries Frequently Asked Questions
- DB2 UDB for iSeries V5R2 Trifold
- DB2 UDB for iSeries: Business Intelligence Concepts (white paper)
- DB2 UDB for iSeries V5R2 (presentation)
- New and Enhanced Utilities for DB2 UDB for iSeries
iSeries Database information sources

- Information Center: http://www.ibm.com/eserver/iseries/infocenter

Newsgroups
- USENET: comp.sys.ibm.as400.misc, comp.databases.ibm-db2
iSeries Database information sources -2

- **Education Resources - Classroom & Online**
  - http://www.iseries.ibm.com/db2/db2educ_m.htm

- **DB2 UDB for iSeries Publications**
  - Online Manuals: http://www.iseries.ibm.com/db2/books.htm
  - 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/400 Developer's Guide* by Paul Conte & Mike Cravitz
  - *iSeries and AS/400 SQL at Work* by Howard Arner
PartnerWorld education courses  - database


- New: Improved Transaction Processing
- New: V5R2 Enhancements
Notes: PartnerWorld education courses - database

The IBM PartnerWorld website has many very useful IBE (Internet Based Education) presentations and labs for several iSeries subjects including Application Development, Building e-business applications, Domino, Porting to iSeries, Java, and Tools.

This foil shows the first page of the Business Intelligence - DB2 for Universal DataBase topics. New items becoming available during July and August 2002 include:

V5R2 Enhancements: This includes details on standards-based SQL functions and performance updates
Improved Transaction Processing: This includes the new with V5R2 adaptive e-transaction services and using Microsoft Transaction Services with DB2 UDB for iSeries.

These PartnerWorld presentations and labs provide more detailed descriptions and usage tips than what is covered in this technical overview presentation.
# Trademarks and Disclaimers

8 IBM Corporation 1994-2002. 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:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>BRMS</td>
</tr>
<tr>
<td>ADSTAR</td>
<td>On-Demand System</td>
</tr>
<tr>
<td>Advanced Function Printing</td>
<td>ClusterProven</td>
</tr>
<tr>
<td>AFP</td>
<td>CODE/400</td>
</tr>
<tr>
<td>AIX</td>
<td>DataGuide</td>
</tr>
<tr>
<td>AnyNet</td>
<td>DB2</td>
</tr>
<tr>
<td>Application Development</td>
<td>DB2 Exenders</td>
</tr>
<tr>
<td>APPN</td>
<td>DB2 Universal</td>
</tr>
<tr>
<td>AS/400</td>
<td>DB2 Universal</td>
</tr>
<tr>
<td>AS/400e</td>
<td>e-business logo</td>
</tr>
<tr>
<td>AT</td>
<td>e(logo) Server</td>
</tr>
<tr>
<td>BriQuery</td>
<td>Enterprise Storage Server</td>
</tr>
<tr>
<td>Host Integration Series</td>
<td>Host on Demand</td>
</tr>
<tr>
<td>Host Publisher</td>
<td>HTTP Server for AS/400</td>
</tr>
<tr>
<td>JustMail</td>
<td>IBM</td>
</tr>
<tr>
<td>Payment Manager</td>
<td>Net.Data</td>
</tr>
<tr>
<td>Payment Server</td>
<td>NetView</td>
</tr>
<tr>
<td>Stylized</td>
<td>Netfinity</td>
</tr>
<tr>
<td>SystemView</td>
<td>NetView</td>
</tr>
<tr>
<td>VisualAge for Java</td>
<td>Netfinity</td>
</tr>
<tr>
<td>VisualAge for RPG</td>
<td>NetView</td>
</tr>
<tr>
<td>WebSphere</td>
<td>NetView</td>
</tr>
<tr>
<td>WebSphere Advanced Edition</td>
<td>NetView</td>
</tr>
<tr>
<td>WebSphere Commerce Suite</td>
<td>NetView</td>
</tr>
<tr>
<td>WebSphere Development Tools</td>
<td>NetView</td>
</tr>
<tr>
<td>for AS/400</td>
<td>NetView</td>
</tr>
<tr>
<td>AS/390</td>
<td>OfficeVision</td>
</tr>
<tr>
<td>Workpad</td>
<td>Operating System/400</td>
</tr>
<tr>
<td>xSeries</td>
<td>Operating System/400</td>
</tr>
<tr>
<td>SmoothStart</td>
<td>iSeries</td>
</tr>
<tr>
<td>iSeries Service</td>
<td>OS/400</td>
</tr>
<tr>
<td>OS/390</td>
<td>OfficeVision</td>
</tr>
<tr>
<td>S/390</td>
<td>OfficeVision</td>
</tr>
<tr>
<td>WebSphere Advanced Edition</td>
<td>OfficeVision</td>
</tr>
<tr>
<td>WebSphere Standard Edition</td>
<td>OfficeVision</td>
</tr>
</tbody>
</table>

cc:Mail, Domino.Doc, Freelance, LearningSpace, Lotus, Lotus Domino, Lotus Notes, iNotes, QuickPlace, Sametime, 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.

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.

IBM's VisualAge products and services are not associated with or sponsored by Visual Edge Software, Ltd.

Linux is a registered trademark of Linus Torvalds.

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.