Welcome!

- Welcome to the DB2 UDB V8 DBA Certification preparation class.
- This class reviews materials to better prepare you for the DB2 UDB V8 Certification Tests:
  - Exam 700 (DB2 UDB V8 Family Fundamentals) and
  - Exam 701 (DB2 UDB V8 Database Administration for Linux, UNIX, and Windows)
- Students must have the following prerequisites:
  - An understanding of Relational Database Concepts
  - Experience using the SQL language
  - An understanding of DB2 UDB V8.1
- Objectives of the Family Fundamentals and Database Administration exams can be found in
  - www.ibm.com/certify/tests/obj700.shtml
  - www.ibm.com/certify/tests/obj701.shtml

Ready? Let's get started!!
Getting Started with DB2 UDB

DB2 UDB Family Products
DB2 UDB Instances
DB2 UDB Environment
DB2 UDB Tools

IBM Software Group

Theme: Fully Exploit All Resources

- Full exploitation of ALL available resources
  - Within a single query
  - Across separate queries
- Regardless of resource class/amount
  - I/O bandwidth, Main memory, Cache, CPUs, Communication bandwidth
- Regardless of architecture
  - Uni, SMP, NUMA, SMP Cluster, MPP
- With Seamless Scalability
  - Application transparency
  - No migration
DB2 Universal Database

- **DB2 Everyplace**
  - Small footprint database engine on handheld devices (~ 150 KB)
  - A synchronization server to replicate data in both directions

- **DB2 Personal Edition (PE)**
  - A fully functional database for workstations
  - Also a DB2 Satellite Client (can be remotely managed)
  - Cannot act as database server for remote clients

- **DB2 Workgroup Edition**
  - A fully functional database for departmental servers
  - Can be installed on server with up to 4 CPUs

- **DB2 UDB Enterprise Edition (ESE)**
  - Ideal for enterprise environments
  - Includes Host Database Connectivity (DB2 Connect)
  - Informix Integration
  - Audio, Image, and Video Extenders
  - Database Partitioning Feature (DPF)
  - Net Search Extender, Spatial Extender, Data Links Manager
  - Warehouse Manager
  - Intelligent Miner Scoring, Intelligent Miner Modeling, Intelligent Miner Visualization

---

DB2 Host Databases, DB2 Connect, DB2 Clients

- **DB2 for iSeries**

- **DB2 for zSeries**

- **DB2 for VSE & VM**

- **DB2 Connect**
  - Personal Edition - provides PC-based desktops (Linux, Windows NT/98/2000/ME) transparent access to DB2 host databases (single user)
  - Enterprise Edition - acts like a gateway that allows LAN-based systems (Windows NT/2000, AIX, HP-UX, Solaris, Linux, Linux/390) and their desktop applications to access DB2 host databases (multi users)

---

- **Runtime Client**
  - Runtime code to access DB2 servers

- **Administration Client**
  - Includes runtime client
  - Provides tools to administer DB2 servers

- **Application Development Client**
  - Includes runtime client
  - Provides tools and libraries to develop applications that access DB2 servers

---

**Communication Support**

[Diagram of DB2 Communication Support]

- **DB2**
- **DB2 Connect**
- **DRDA Application Requester**
- **DB2 Routine Clients**
- **DB2 Admin Clients**
- **DB2 AD Clients**

---

**DB2 Data Management Software**

---

[Diagram of DB2 Data Management Software]
More about DB2 Editions and Products

- **DB2 Personal Developer’s Edition**
  - A single user database engine, enables a developer to design and build applications
  - Package includes DB2 Personal Edition, DB2 Connect Personal Edition, AD Client, DB2 Extenders
  - Platforms: Linux, Windows NT/2000/XP

- **DB2 Universal Developer’s Edition**
  - Offers a low cost package for a single application developer to design, build, and prototype applications for deployment on any of the DB2 client or server platforms
  - Package includes all client and server DB2 editions as well as DB2 Connect, DB2 Extenders, Warehouse Manager, and Intelligent Miner
  - Platforms: AIX, HP-UX, Linux, Solaris, Windows NT/2000/XP

- **DB2 Warehouse Manager**
  - Performs SQL-based extract, transform, and load (ETL) capabilities to move and transform data
  - Provides distributed agents to transfer data directly between sources and targets, and a metadata management solution

- **DB2 OLAP Server**
  - Delivers analytic applications for fast, intuitive multidimensional analysis, allowing users to ask questions in an intuitive business language, processes multidimensional requests that calculate, consolidate, and retrieve information from a multidimensional database, a relational database

- **Relational Connect**
  - Part of IBM Information Integrator
  - Allows access to non-DB2 data sources, e.g. Oracle, Sybase, MS SQL Server, ODBC data sources

- **DB2 Extenders**
  - AIV Extenders, Spatial Extender, Net Search Extender, XML Extender

---

DB2 Instances

- A DB2 instance is a logical context in which DB2 commands and functions are executed
- An instance manages access to database files
- More than one instance can be defined on a server machine
- Each instance is independent of the others
- An instance is created on install by default
  - `db2inst1` (UNIX)
  - `DB2` (Windows)
- To start an instance
  - `db2start`
  - Starts a remote instance
    - `db2start REMOTE <instance name>`
  - Starts the instance in quiesced mode for administration purposes
    - `db2start ADMIN MODE`
- Terminate all database connections and stop an instance
  - `db2stop force`
Managing DB2 Instances

- To create additional instances:
  - `db2icrt -u <fenced user ID> <instance name>`
    - must specify fenced user ID for UNIX platforms
  - `db2icrt <instance name>`
    - for Windows platforms
- Terminate all database connections and drop an instance:
  - `db2idrop -f <instance name>`
- To list existing instances defined in a server:
  - `db2ilist`
- To update a DB2 instance for access to functions associated with installation or removal of certain product options or fix patches:
  - `db2iupdt <instance name>`
- To migrate an existing instance, for UNIX platforms only:
  - `db2imigr <instance name>`
- To create, drop, update, or migrate an instance, root or administrative access is required.

DB2 Instances and Databases
DB2 Environment

- Configure the DB2 environment via:
  - DB2 registry variables
  - Operating System environment variables
  - DB2 database manager configuration parameters
  - DB2 database configuration parameters

Server machine
OS environment variables

Instance

- db2 registry
  - new features, platform dependent optimizations
  - special optimizations

Database

- database manager configuration file dbm.cfg
- database configuration file db.cfg

DB2 Profile Registry

- Most of the DB2 environment is controlled by entries stored in the DB2 Profile Registry.
- Objective is to centralize control of DB2 environment.
- No need to reboot the system after making changes to DB2 Registry.
- At its various level, controls many aspects of the DB2 UDB environment.
  - DB2 Instance-Level Profile Registry
  - DB2 Global-Level Profile Registry
  - DB2 Instance-Node-Level Profile Registry
  - DB2 Instance Profile Registry

- To list all DB2 registry variables currently set
  - db2set -all
- To display global registry variables
  - db2set -g
- To display instance registry variables
  - db2set -i <instname>
- To list all supported registry variables
  - db2set -lr
- To set a DB2 registry variable
  - db2set <parm>=<value>
    - e.g. db2set DB2COMM=TCPIP
- Must restart the instance for new changes to be in effect.
- DB2 Configuration Assistant (db2ca) can also be used to manage DB2 registry variables.
OS environment variables

- Most environment variables are controlled by the DB2 Profile Registry - those not stored in the DB2 Profile Registry can be referred to as system environment variables, and are set in a location where the platform stores its system variables:
  - Windows - Control Panel -> System -> Environment Variables
  - UNIX - .profile or .kshrc or .cshrc, they are typically incorporate with db2profile (Bourne or Korn shell) or db2cshrc (C shell) found in the instance owner's home directory

- Examples:
  - Windows - set DB2INSTANCE=PROD
  - UNIX - export DB2INSTNCE=PROD

Setting DB2 Variable Values

<table>
<thead>
<tr>
<th>Environment Variables</th>
<th>Platform Specific</th>
<th>Windows NT/2000</th>
<th>UNIX</th>
<th>LINUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance-Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registry Profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global-Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registry Profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Use db2set command to change
- Must be initialized after changing

DB2 Configuration Parameters

- Get/Set/Reset the Database Manager Configuration
  - GET DBM CFG [SHOW DETAIL]
    - SHOW DETAIL displays current and delayed values, this option requires an explicit instance attachment
  - ATTACH TO db2inst1
  - UPDATE DBM CFG USING <param> <value> [ IMMEDIATE | DEFERRED ]
    - With option IMMEDIATE (default behavior), changes to some parameters may take effect immediately while the instance is running.
    - Parameters that cannot be changed online are called configurable online configuration parameters. Those that cannot use the new value immediately requires DB2 to be restarted.
  - RESET DBM CFG

- Get/Set/Reset the Database Configuration
  - GET DB CFG FOR <db> [SHOW DETAIL]
    - Show detail displays current and delayed values, this option requires a database connection
  - CONNECT TO sample
  - UPDATE DB CFG FOR <db> USING <param> <value> [ IMMEDIATE | DEFERRED ]
    - With option IMMEDIATE (default behavior), changes to some parameters may take effect immediately while the database is activated.
    - Parameters that cannot be changed online requires the database to be deactived for new value to be used.
  - RESET DB CFG FOR <db>

- DB2 Control Center can be used to display, update, and reset the DBM and DB configuration parameters.
DB2 Command Line Tools

- **DB2 Command Line Window (Windows Only)**
  - click on the DB2 Command Window icon or from MS-DOS prompt, enter `db2cmd`
  - invoke the interpreter by prefacing commands and SQL with `db2`
    - e.g. `db2 connect to sample`
    - e.g. `db2 "select * from employee"`
    - e.g. `db2 -tvf createtab.db2`
  - to end command line mode and terminate the DB2 backend process (db2bp), enter `db2 terminate`
  - all OS commands can be issued from the DB2 Command Window

- **DB2 Command Line Processor (CLP)**
  - click on the **DB2 Command Line Processor** icon or enter `db2` at a command prompt
  - an interactive input prompt appears:
    - `db2 =>`
  - no need to prefix commands with `db2`
    - e.g. `db2 => connect to sample`
  - to end the interactive mode, enter `quit`
  - to disconnect from the database and terminate the DB2 backend process (db2bp), enter `terminate`
  - to execute OS commands, enter `!<OS command>`

### Command Line Options

- **to list default command options**
  - LIST COMMAND OPTIONS

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Current Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Display SQLCA</td>
<td>OFF</td>
</tr>
<tr>
<td>-c</td>
<td>Auto-Commit</td>
<td>ON</td>
</tr>
<tr>
<td>-w</td>
<td>Display SQLCODE/SQLESTATE</td>
<td>OFF</td>
</tr>
<tr>
<td>-f</td>
<td>Read from input file</td>
<td>OFF</td>
</tr>
<tr>
<td>-l</td>
<td>Log commands in history file</td>
<td>OFF</td>
</tr>
<tr>
<td>-n</td>
<td>Remove new line character</td>
<td>OFF</td>
</tr>
<tr>
<td>-r</td>
<td>Display output</td>
<td>ON</td>
</tr>
<tr>
<td>-p</td>
<td>Display interactive input prompt</td>
<td>ON</td>
</tr>
<tr>
<td>-z</td>
<td>Save output to report file</td>
<td>OFF</td>
</tr>
<tr>
<td>-s</td>
<td>Stop execution on command error</td>
<td>OFF</td>
</tr>
<tr>
<td>-t</td>
<td>Set statement termination character</td>
<td>OFF</td>
</tr>
<tr>
<td>-w</td>
<td>Display FETCH/SELECT warning messages</td>
<td>ON</td>
</tr>
<tr>
<td>-x</td>
<td>Suppress printing of column headings</td>
<td>OFF</td>
</tr>
<tr>
<td>-z</td>
<td>Save all output to output file</td>
<td>OFF</td>
</tr>
</tbody>
</table>

- **To change default command options**
  - e.g. `set db2options=+c` (to turn auto commit off)

- **To change command option for the current session**
  - e.g. to save output to file:
    - UPDATE COMMAND OPTIONS USING `z` ON output.txt
DB2 GUI Tools

- Control Center
  - Central point of administration
- Command Center
  - GUI wrapper for command line processor (CLP)
- Journal
  - Central point for logging DB2 Activities
- License Center
  - Central point of DB2 license management
- Development Center
  - Central point for developing:
    - SQL & Java Stored Procedures
    - SQL & Java User Defined
- Task Center
  - Central point for scheduling database jobs

- Health Center
  - Central point for DB2 Health Monitoring
- Memory Visualizer
- Configuration Assistant
  - Tool to configure the client
  - A light-weight version of the Control Center
- Replication Center
  - A GUI tool to set up and administer a replication environment and to run the Capture and Apply programs

DB2 Administration Server (DAS) and Tools Catalog Database

- A background process to support instance administration
  - db2as on UNIX (default)
  - DB2DAS service on Windows (default)
- Provides support services for DB2 tools such as the Control Center, Configuration Assistant, Replication Center, and Development Center
- Only one DAS per server machine
- Need correct authorization to create a service, e.g. root authority in UNIX
- Some DAS commands:
  - db2admin create
  - db2admin (to obtain the name of the DAS on the machine)
  - db2admin start
  - db2admin stop
  - db2 get admin cfg
  - db2 update admin cfg using <parm> <value>
- The DB2 Tools Catalog must be created to enable task scheduling
- Can be created as an independent database or within an existing database
- Requirements:
  - 32K pagesize tablespaces
  - two 32K pagesize tablespaces are created by default, REGULAR and SYSTEM TEMPORARY tablespaces
Getting Connected to DB2 UDB

Database Connectivity
Instance Attachment vs Database Connection

IBM Software Group

Preparing the Server for DB2 Remote Connections

- **DBA** must ensure server side communications is set up properly

**Step 1:**
- Enable the instance for communication supports, set DB2 registry variable
  - `db2set DB2COMM=TCP/IP,NETBIOS`

**Step 2:**
- Set the protocol information in database manager configuration, for TCPIP:
  - `db2 update dbm cfg using svcename 50000`
  - OR
  - `db2 update dbm cfg using svcename db2icdb2`
  - Update `/etc/services` with the service name and port number
    - `db2icdb2 50000/tcp`

**Step 3:**
- `db2stop` and `db2start` instance after changes to database manager configuration
DB2 Discovery at DB2 Clients

- Search and locate DB2 servers on the network
- Two methods: Known and Search
- Requires DAS running and enabled to be discovered on the DB2 servers
- Can prevent or "hide" a server / database from being discovered

Using Access Profiles

- Assist with automation of client configuration
- Access profiles contain the information that a client needs to catalog databases on a DB2 UDB server
- Two types of Access Profiles:
  - Server Access Profile generated on a UDB server
    - Information about all instances/databases on the server
  - Client Access Profile generated on a client
    - Used to copy the client’s catalog information to a number of other clients

Example of DB2 access profile:

```
[FILE_DESCRIPTION]
APPLICATION=DB2/NT 8.1.0
FILE_CONTENT=DB2 CCA Exported Data Sources
FILE_TYPE=CommonServer
FILE_FORMAT_VERSION=2.0
Platform=5
DB2SYSTEM=CLARALIU
Instance=DB2

[REGISTRY_GLOBAL]
DB2_GRP_LOOKUP=LOCAL
DB2INSTDEF=DB2

[INST>DB2]
instance_name=DB2
NodeType=4
ServerType=DB2NT
Authentication=SERVER
DB2COMM=TCP/IP

[DB>!LOCAL:SAMPLE]
Dir_entry_type=INDIRECT
Drive=C:\DB2
DBName=SAMPLE

[REGISTRY_LOCAL]
DB2ACCOUNTNAME=CLARALIU\db2admin
DB2INSTOWNER=CLARALIU
DB2PORTRANGE=50000:60003
DB2_GRP_LOOKUP=LOCAL
DB2COMM=TCP/IP
```
Manually Configure Client Connectivity via Commands

- **Catalog NODE**
  - Catalog each instance on DB2 UDB Server
  - Specific command for each supported communication protocol
  - Examples:
    - `CATALOG TCPIP NODE mynode REMOTE server1 SERVER 50000`
    - `CATALOG NETBIOS NODE jeremy REMOTE N01FCBE3 ADAPTER 0`

- **Catalog DATABASE**
  - Catalogs each database on DB2 UDB Server and instance
  - Works for local and remote databases
  - Example:
    - `CATALOG DATABASE sample AS mysamp AT NODE mynode`

- **Catalog DCS DATABASE**
  - Database Connection Service (DCS)
  - Catalogs DRDA databases on OS/390, AS/400, or VM,VSE
  - These databases are accessed through an Application Requester (AR), such as DB2 Connect
  - Having a DCS directory entry with a database name matching a database name in the system database directory invokes the AR to forward SQL requests to the host server
  - Example:
    - `CATALOG DCS DATABASE db1 AS dsn_db_1`

Cataloging the remote DAS

- Facilitates remote node (instance) administration
- Required for full remote instance abstraction
- ALWAYS assumes port 523

- `CATALOG ADMIN TCPIP NODE <node alias> REMOTE <hostname>`

- Example:
  - `CATALOG ADMIN TCPIP NODE zeus REMOTE zeus.mycompany.com`
Other Related Commands

- CATALOG APPC NODE ...
  - APPC - Advanced Program to Program Communications protocol
- CATALOG APPN NODE ...
- CATALOG DATABASE ...
- CATALOG DCS DATABASE ...
- CATALOG LDAP DATABASE ...
- CATALOG LDAP NODE ...
- CATALOG LOCAL NODE ...
- CATALOG NAMED PIPE NODE ...
- CATALOG ODBC DATA SOURCE ...
- CATALOG TCPIP NODE ...

- LIST NODE DIRECTORY
  - List all local and remote nodes cataloged
- LIST DATABASE DIRECTORY
  - List all local and remote databases cataloged
- LIST DATABASE DIRECTORY ON c:
  - List local databases created on the C drive

Instance Attachment vs Database Connection

- INSTANCE ATTACHMENT
  - create/drop databases
  - get/update/reset database manager and database configuration file
  - database monitor
  - backup/restore/roll forward database
  - force application
    - Implicit instance attachment:
      - DB2INSTANCE<instance name>
      - Environment Variable
    - Explicit instance attachment:
      - ATTACH TO nodename [USER ... USING...]
      - Nodename is important

- DATABASE CONNECTION
  - DML, DDL, DCL
  - precompile/bind applications
  - load/export/import
    - Implicit:
      - DB2 Profile variable set with db2set command
      - db2set DB2DBDFT=<database name>
    - Explicit:
      - CONNECT TO db-alias [USER ... USING... NEW <new_pwd> CONFIRM <new_pwd>]
Remote Administration - ATTACH

- Attach to a node that has been cataloged in the DB2 node directory
- For example:
  - ATTACH TO db2instx USER db2admin USING passwd
  - RESTORE DATABASE hr FROM /db2/backup REPLACE EXISTING
  - DETACH
- User ID and password are required if the node is a remote node

Connect to a Database

- Connect to a database that has been cataloged in the DB2 database directory
- Example:
  - CONNECT TO sample USER db2admin USING passwd
  - INSERT INTO testtab VALUES (1,'DB2')
  - CONNECT RESET
- User ID and password are required if database is on a remote server

- To find out currently connected database, use one of the following commands:
  - GET CONNECTION STATE
    
    Database Connection State
    Connection state = Connectable and Connected
    Connection mode = SHARE
    Local database alias = SAMPLE
    Database name = SAMPLE
  - VALUES CURRENT SERVER
    
    1
    ------------------
    SAMPLE
    1 record(s) selected.
Some Useful Commands

- **LIST APPLICATIONS [SHOW DETAIL]**
  - Shows all current connections made to any databases defined within an instance

<table>
<thead>
<tr>
<th>Auth Id</th>
<th>Application Name</th>
<th>Appl Handle</th>
<th>Application Id</th>
<th>DB Name</th>
<th># of Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CLARALIU db2bp.exe</td>
<td>*LOCAL.DB2.00E000150926</td>
<td>SAMPLE 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CLARALIU db2bp.exe</td>
<td>*LOCAL.DB2.00F1C0150335</td>
<td>SAMPLE 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DB2ADMIN db2dasstm.exe</td>
<td>*LOCAL.DB2.005A40134734</td>
<td>TOOLSDB 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DB2ADMIN db2dasstm.exe</td>
<td>*LOCAL.DB2.005A40134733</td>
<td>TOOLSDB 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **FORCE APPLICATION ALL**
  - Disconnect all connections made to any databases defined within an instance

- **FORCE APPLICATION (h1 [,h2,...hn])**
  - Selectively disconnect applications by the application handle identifier

- **GET INSTANCE**
  - Retrieve your current instance context

The current database manager instance is: DB2
DB2 Security Overview

- DB2 uses a combination of:
  - External security service
  - Internal access control information
- Authentication
  - Identify the user
  - Check entered user name and password
  - Done by security facility outside of DB2 (Part of the OS, Kerberos and so forth)
- Authorization
  - Users can access only DB2 objects for which they have the appropriate authorization - the required authorities or privileges
  - Check if authenticated user may perform requested operation
  - Done by DB2 facilities
  - Information stored in DB2 catalogs and DBM configuration file

CONNECT TO sample
USER bob using pwd

SELECT * FROM mytable

Authentication Types

- Authentication type determines WHERE the user ID and password are verified
- Authentication types supported in DB2 UDB
  - SERVER (default)
  - SERVER_ENCRYPT
  - KERBEROS
  - KRB_SERVER_ENCRYPT
  - CLIENT

- Two places to specify authentication type: Server and Client
- At Server:
  - One authentication type per instance, applies to all databases under the instance
  - Defined at the instance level in the database manager configuration file
  - Can be updated: UPDATE DBM CFG USING AUTHENTICATION [...]
- At Client:
  - Defined in the CATALOG DATABASE command, therefore authentication type applies to the specified database only
  - Example:
    CATALOG DATABASE sample AT NODE db2server AUTHENTICATION SERVER
Authentication Types

- **AUTHENTICATION=SERVER**
  - Authentication occurs at the server
  - Central management of users
  - User ID and password flow over network
  - Can be encrypted with `AUTHENTICATION = SERVER_ENCRYPT`

- **AUTHENTICATION=KERBEROS**
  - Use conventional cryptography to create a shared secret key
  - This key becomes a user's credential and is used to verify the identity of users when local or network services are requested
  - Eliminates the need to pass the user name and password across the network as clear text
  - Enables the use of a single sign-on to a remote DB2 server

- **AUTHENTICATION=KRB_SERVER_ENCRYPT**
  - Authentication `SERVER_ENCRYPT` and KERBEROS can be used by clients accessing the same DB2 server instance
  - KERBEROS and KRB_SERVER_ENCRYPT only support clients and servers running Windows 2000 or XP or .NET platforms

If authentication type specified at client and server is different:

<table>
<thead>
<tr>
<th>Client Specification</th>
<th>Server Specification</th>
<th>Client/Server Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>KERBEROS</td>
<td>KRB_SERVER_ENCRYPT</td>
<td>KERBEROS</td>
</tr>
<tr>
<td>Any other setting</td>
<td>KRB_SERVER_ENCRYPT</td>
<td>SERVER_ENCRYPT</td>
</tr>
</tbody>
</table>

**Authentication Type - CLIENT**

- Authentication occurs at the client
- If the remote instance has CLIENT authentication, two other parameters determine the final authentication type, `TRUST_ALLCLNTS, TRUST_CLNTAUTH`

- **TRUST_ALLCLNTS** - decides which types of clients to trust
  - **TRUST_ALLCLNTS = YES** => Trust all clients including trusted, non-trusted, and host clients; authentication will take place at client (except one case)
  - **TRUST_ALLCLNTS = NO** => All untrusted clients will be authenticated at the server, must provide user ID and password
  - **TRUST_ALLCLNTS = DRDAONLY** => Only hosts clients are allowed to authenticate at client

- **TRUST_CLNTAUTH** - specifies where authentication will take place when a user ID and password are supplied with a CONNECT statement or ATTACH command
  - **Active when AUTHENTICATION=CLIENT and when userid and password are provided**
  - **TRUST_CLNTAUTH=CLIENT**
    - Authentication done at CLIENT, user ID and password not required
  - **TRUST_CLNTAUTH=SERVER**
    - Authentication done at SERVER if a user ID and password are supplied

<table>
<thead>
<tr>
<th>TRUST_ALLCLNTS</th>
<th>TRUST_CLNTAUTH</th>
<th>Trusted Client Authentication no password</th>
<th>Trusted Client Authentication with password</th>
<th>Untrusted Client Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES (default)</td>
<td>CLIENT (default)</td>
<td>CLIENT</td>
<td>CLIENT</td>
<td>N/A</td>
</tr>
<tr>
<td>YES (default)</td>
<td>SERVER</td>
<td>CLIENT</td>
<td>SERVER</td>
<td>N/A</td>
</tr>
<tr>
<td>NO</td>
<td>CLIENT (default)</td>
<td>CLIENT</td>
<td>CLIENT</td>
<td>SERVER</td>
</tr>
<tr>
<td>NO</td>
<td>SERVER</td>
<td>CLIENT</td>
<td>SERVER</td>
<td>SERVER</td>
</tr>
</tbody>
</table>
DB2 Access Control Authority

- Five authorities in DB2 UDB
  - SYSADM held the most authorities and privileges for the DB2 instance
  - SYSCTRL and SYSMAINT give full privileges for managing the system, but does not allow access to the data
  - DBADM authority gives privileges to perform administrative task on the database and has full data access to the database
  - Load authority gives privileges for running the LOAD utility without the need for SYSADM or DBADM

- System authorities are assigned to user groups defined in operating system or security facility in database manager configuration.

- Maximum length for group name is 8.

---

Privileges

- Database Privileges
  - CONNECT allows a user to access the database
  - BINDADD allows a user to create new packages in the database
  - CREATETAB allows a user to create new tables in the database
  - CREATE NOT_FENCED allows a user to create a user-defined function (UDF) or stored procedure that is "not fenced"
  - IMPLICIT_SCHEMA allows a user to create objects in a schema that does not already exist
  - QUIESCE_CONNECT allows a user to access the database while it is quiesced
  - CREATE_EXTERNAL_ROUTINE allows a user to create external procedures and functions

- Schema Privileges
  - CREATEIN allows the user to create objects within the schema
  - ALTERIN allows the user to alter objects within the schema
  - DROPIN allows the user to drop objects from within the schema
  - To specify a schema owner other than the authorization ID used to connect, execute the CREATE SCHEMA command:
    - CONNECT TO sample USER dbowner ;
    - CREATE SCHEMA dev AUTHORIZATION devusers ;

- Table Space Privilege
  - USE OF TABLESPACE privilege allows users to create tables within the specified table space
  - USE privilege cannot be used with SYSCATSPACE or any system temporary table spaces
Privileges ... continued

Table and Views Privileges
- **CONTROL** provides the user with all privileges for a table or view, as well as having the ability to extend those privileges to others (except CONTROL)
- **ALTER** allows the user to alter a table
- **DELETE** allows the user to delete rows from a table or view
- **INDEX** allows the user to create an index on a table
- **INSERT** allows the user to insert an entry into a table or view
- **REFERENCES** allows the user to create and drop a foreign key
- **SELECT** allows the user to retrieve rows from a table or view, to create a view on a table, and to run the EXPORT utility
- **UPDATE** allows the user to change an entry in a table, a view, or one or more specific columns in a table or view. The user may have this privilege only on specific columns
  - Example: GRANT UPDATE (col1, col2) ON TABLE employee TO user1;
- **ALL PRIVILEGES** option grants all the appropriate privileges, except CONTROL, on the base table, view, or nickname named in the ON clause
  - Example: GRANT ALL PRIVILEGES ON TABLE employee TO user1;

Package Privileges
- **CONTROL** provides the user with the ability to rebind, drop, or execute a package as well as the ability to extend those privileges to others (except CONTROL)
- **BIND** allows the user to rebind an existing package
- **EXECUTE** allows the user to execute a package

Index Privilege
- **CONTROL** allows user to drop the index

Routine Privilege
- **EXECUTE** allows user to invoke a routine
- Example:
  - GRANT EXECUTE ON FUNCTION calc_salary(empno) TO jones;
  - GRANT EXECUTE ON SPECIFIC FUNCTION calc_salary TO jones WITH GRANT OPTION;

Sequence Privilege
- **USAGE** privilege allows user to use NEXTVAL and PREVVAL expressions for the sequence object
- Example:
  - GRANT USAGE ON SEQUENCE org_seq TO PUBLIC;

GRANT / REVOKE database authority
- Database level authorities, DBADM and LOAD, are granted to a user or a group of user by the GRANT statement
- Revoke them with the REVOKE statement
- When DBADM authority is granted, BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA are granted as well
- Users with LOAD authority also requires INSERT privilege to load data in a table
- Example:
  - GRANT DBADM ON DATABASE TO USER john;
  - GRANT LOAD ON DATABASE TO GROUP maintgrp;
  - REVOKE LOAD ON DATABASE FROM maintgrp;
Implicit and Indirect Privileges

- If a user was granted with DBADM
  - Privileges BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA are implicitly granted
- When a database is created
  - DBADM authority with the implicit privileges are granted to the creator
  - The following privileges are all granted to PUBLIC:
    - CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA
    - SELECT on system catalog tables
    - USE privilege on USERPSPACE1
    - BIND and EXECUTE on each successfully bound utility
    - EXECUTE WITH GRANT on all functions in the SYSFUN schema
- When an object (table, index, package) is created
  - CONTROL privilege of object is granted to creator
- If a user is granted with EXECUTE on a package that contains static SQL statements, the user can successfully execute the statements even though this person does not have access to the underlying database objects.

System Catalog Views with Privileges Information

- Most of the information on authorizations is maintained in the catalog views:

<table>
<thead>
<tr>
<th>View Name</th>
<th>Privilege Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCAT.DBAUTH</td>
<td>Database privileges</td>
</tr>
<tr>
<td>SYSCAT.TABAUTH</td>
<td>Table and view privileges</td>
</tr>
<tr>
<td>SYSCAT.COLAUTH</td>
<td>Column privileges</td>
</tr>
<tr>
<td>SYSCAT PACKAGEAUTH</td>
<td>Package privileges</td>
</tr>
<tr>
<td>SYSCAT.INDEXAUTH</td>
<td>Index privileges</td>
</tr>
<tr>
<td>SYSCAT.SCHEMAAUTH</td>
<td>Schema privileges</td>
</tr>
<tr>
<td>SYSCAT.PASSTHRUAUTH</td>
<td>Federated server privileges</td>
</tr>
<tr>
<td>SYSCAT ROUTINEAUTH</td>
<td>Routine privileges (functions, methods, and stored procedures)</td>
</tr>
</tbody>
</table>
Database Objects

Buffer Pools and Table Spaces
Schemas and Catalogs
Data Types
Tables, Identity Columns, Temporary Tables
Views

Indexes
Constraints
Packages
Triggers, Functions, Stored Procedures

IBM Software Group

Buffer Pools, Table Spaces

- **Buffer Pool**
  - Used to buffer data in memory to reduce the number of I/O operations to the physical database
  - Keep often requested data/index pages in memory
  - IBMDEFAULTBP is the default bufferpool created with every database

- **Table space**
  - All database objects are stored within table spaces
  - Two types of table spaces: System Managed Space (SMS) and Database Managed Space (DMS)
  - A table space is composed of one or more containers, data allocated by extents within containers
  - Table spaces can be 4K, 8K, 16K or 32K pages, cannot mix page sizes within a table space
  - Must be associated with a buffer pool with same page size
  - With a simple command: `CREATE DATABASE <dbname>`
  - Three SMS table spaces are created automatically:
    - `SYSCATSPACE` - system catalog tables
    - `USERSPACE1` - default user data
    - `TEMPSPACE1` - temporary data
  - Can change table space storage type and explicitly specify the locations of the containers, e.g.:
    - `CREATE DATABASE sample`  
    - `CATALOG TABLESPACE MANAGED BY SYSTEM USING ('c:\catdir1');`  
    - `USER TABLESPACE MANAGED BY DATABASE USING (FILE 'c:\db2files\usertbsp1 100, FILE 'c:\db2files\usertbsp2 100 )`  
    - `TEMP TABLESPACE MANAGED BY SYSTEM USING ('c:\tempspace') ;`
Schema and Catalogs

- A schema provides a logical classification for a collection of database objects.
- Schema names SYSCAT, SYSIBM, SYSSTAT, SYSFUN are reserved. A complete listing of the catalogs can be found in the SQL Reference.
  - SYSIBM contains base catalogs
  - SYSCAT contains read-only catalog views
  - SYSSTAT contains updateable catalog views about statistics of the database statistics
  - SYSFUN is the schema for system functions (such as SUM, AVG ...)
- If database object does not specify a schema name, table will be qualified with current authorization ID.
- Alternate schema names can be specified using one of the following options
  - SET CURRENT SCHEMA or SET CURRENT SQLID command
  - CREATE ALIAS <aliasname> FOR <tab/view name>
  - CREATE VIEW

Data Types

- Numeric
  - Smallint
  - Integer
  - BigInt
  - Decimal
  - Real
  - Double

- String
  - Character String
    - Single Byte
    - Double Byte
  - Binary String
    - BLOB

- Datetime
  - Date
  - Time
  - Timestamp

Examples:
- CREATE DISTINCT TYPE pound
  AS INTEGER WITH COMPARISONS;
- CREATE DISTINCT TYPE kilogram
  AS INTEGER WITH COMPARISONS;

- CREATE TABLE person
  (f_name VARCHAR(30),
   weight_p pound NOT NULL,
   weight_k kilogram NOT NULL);

- SELECT f_name FROM person
  WHERE weight_p > pound(30);

- SELECT f_name FROM person
  WHERE weight_p > weight_k;

Fails
OK
Data Types ... continued

Selecting the correct data type:

<table>
<thead>
<tr>
<th>Question</th>
<th>DataType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the data fixed in length? Stored in binary format?</td>
<td>CHAR</td>
</tr>
<tr>
<td></td>
<td>CHAR for bit data</td>
</tr>
<tr>
<td>Is the data variable in length? Stored in binary format?</td>
<td>VARCHAR</td>
</tr>
<tr>
<td></td>
<td>VARCHAR for bit data</td>
</tr>
<tr>
<td>Do you need to sort(order) the data?</td>
<td>CHAR, VARCHAR</td>
</tr>
<tr>
<td></td>
<td>NUMERIC</td>
</tr>
<tr>
<td>Is the data to be used in arithmetic operations?</td>
<td>DECIMAL, REAL</td>
</tr>
<tr>
<td></td>
<td>DOUBLE, BIGINT</td>
</tr>
<tr>
<td></td>
<td>INTEGER, SMALLINT</td>
</tr>
<tr>
<td>Does it contain decimal?</td>
<td>DECIMAL, REAL</td>
</tr>
<tr>
<td></td>
<td>DOUBLE</td>
</tr>
<tr>
<td>Does the data have a specific meaning (beyond DB2 base data type)?</td>
<td>UDT</td>
</tr>
</tbody>
</table>

Creating Tables

- You must have SYSADM or DBADM authority or CREATETAB privilege on the db.

    ```sql
    connect to eddb;
    create table artists
    ( artno   SMALLINT NOT NULL
    , name    VARCHAR(50) WITH DEFAULT 'abc'
    , rate    INTEGER WITH DEFAULT 0
    , bio     CLOB(100K) LOGGED
    , picture BLOB(2M) NOT LOGGED COMPAT )
    INDEX IN indtbsp LONG IN longtbsp IN datatbsp;
    ```

- CREATE TABLE ... LIKE command creates table columns to have exact same names and attributes of the specified table, data, constraints, triggers and indexes are not copied. Example: `CREATE TABLE tab1new LIKE tab1;`

- CREATE TABLE ... DEFINITION ONLY command creates table with specified columns through use of query, table not populated

    ```sql
    CREATE TABLE t1new AS (SELECT c1, c8, c10 FROM t1) DEFINITION ONLY;
    ```

- If a specific default value is not specified following the DEFAULT keyword, the system default value of the column data type is used.

- Check the DB2 Command Reference under the ‘ALTER TABLE’ command for a complete list of system default values of each data type, some examples:
  - Numeric - 0
  - CHAR - Blanks
  - VARCHAR - A string of length 0
  - BLOB - A string of length 0
Value Compression

- **NULL and 0-Length Data Value Compression**
  - Available for all tables except global temporary tables.
  - Specifies the VALUE COMPRESSION clause in the CREATE TABLE command so that NULL and 0-length data values are to be stored more efficiently for most data types.
  - Eligible data types: NUMERIC, CHAR, VARCHAR, DBCS, BLOB
  - Not supported data types: DATE, TIME, TIMESTAMP (they are dynamic and are always changing)

- **Example:**
  
  ```sql
  CREATE TABLE comp_t1 ( c1 INTEGER DEFAULT 0 , c2 CHAR(10) DEFAULT NULL ) VALUE COMPRESSION ;
  ```

- **System Default Value Compression**
  - If VALUE COMPRESSION is used, use the optional COMPRESS SYSTEM DEFAULT option to further reduce disk space usage.
  - The default value will not be stored on disk. Data types that support COMPRESS SYSTEM DEFAULT are all numeric, fixed-length character, and fixed-length graphic string data types

- **Example:**
  
  ```sql
  CREATE TABLE comp_t1 ( c1 INTEGER NOT NULL COMPRESS SYSTEM DEFAULT , c2 CHAR(10) COMPRESS SYSTEM DEFAULT ) VALUE COMPRESSION ;
  ```

Identity Columns

- A numeric column in a table which automatically generates a unique numeric value for each row that is inserted. Maximum one identity column per table.
- Values can be generated by DB2 always or by default
  - Generated always - values are always generated by DB2, applications are not allowed to provide an explicit value.

  ```sql
  CREATE TABLE inventory
  ( partno INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 100 INCREMENTED BY 1) , description CHAR(20) );
  INSERT INTO inventory VALUES (DEFAULT,'door');  --->inserts 100,door
  INSERT INTO inventory (description) VALUES ('hinge'); --->inserts 101,hinge
  INSERT INTO inventory VALUES (200,'windor'); --->error
  ```

  - Generated by default - values can be explicitly provided by an application or if no value is given, then DB2 generates one, therefore DB2 cannot guarantee uniqueness.

  ```sql
  CREATE TABLE inventory
  ( partno INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 100 INCREMENTED BY 1), description CHAR(20) );
  INSERT INTO inventory VALUES (DEFAULT,'door');  --->inserts 100,door
  INSERT INTO inventory (description) VALUES ('hinge'); --->inserts 101,hinge
  INSERT INTO inventory VALUES (200,'window'); --->inserts 200,window
  INSERT INTO inventory VALUES (102,'handle');   --->inserts 102,handle
  INSERT INTO inventory VALUES (101,'bolt');     --->error, duplicate
  ```
Declared Temporary Tables

- Created and used by an application and dropped (automatically) when the application terminates
- Can only be accessed by the application that created the table
- Transaction logging is optional
- Automatic cleanup on disconnect
- Performance
  - no system catalog contention
  - no row locking
  - logging optional
  - no authority checking
- Index support
  - any standard index can be created on a temporary table
- Statistics support (RUNSTATS supported against the table)

In-memory, connection based tables.
Requires a user temporary tablespace (none exists by default)

```sql
CREATE USER TEMPORARY TABLESPACE apptemps
MANAGED BY SYSTEM USING ('apptemps');

DECLARE GLOBAL TEMPORARY TABLE t_employees
LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE t_dept
( deptid CHAR(6), deptname CHAR(20) )
ON COMMIT DELETE ROWS NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE t_projects
AS ( fullselect ) DEFINITION ONLY
ON COMMIT PRESERVE ROWS NOT LOGGED
WITH REPLACE IN TABLESPACE apptemps;
```
Creating Views

- Needs to have at least SELECT privilege on the base tables of the view.
- View information kept in SYSCAT.VIEWS, SYSCAT.VIEWDEP, SYSCAT.TABLES
- Views with CHECK OPTION specifies a constraint to every row that is being inserted or updated through the view must conform to the definition of the view.
- A row that does not conform to the definition of the view is a row that does not satisfy the search conditions of the view.
- If a view is defined based on another view or a table with check constraints, it is possible to inherit or not to inherit the search condition, two options available:
  - WITH CASCADED CHECK OPTION (default)
  - WITH LOCAL CHECK OPTION
- Examples:
  - `CREATE VIEW emp_view2 (empno, empname, deptno) AS (SELECT id, name FROM employee WHERE dept = 10) WITH CHECK OPTION;`
  - `CREATE VIEW emp_view3 AS (SELECT empno, empname, deptno FROM emp_view2 WHERE empno > 20) WITH CASCADED CHECK OPTION;`
  - `CREATE VIEW emp_view4 AS (SELECT empno, empname, deptno FROM emp_view3 WHERE name = 'Smith') WITH LOCAL CHECK OPTION;`

Creating Indexes

- Index Characteristics:
  - ascending or descending
  - Unique or non-unique
  - bi-directional (no storage overhead, see notes)
  - compound
  - cluster
  - include columns
- RENAME INDEX xyz TO pdq
  - allows to create new index, remove old, rename new name to old name for consistency
- Examples:
  - `create unique index itemno on albums (itemno) desc`
Type-2 Indexes, Index Recreation

- Version 8 adds support for type-2 indexes to improve concurrency because the use of
  minimal next-key locking, create index on columns that have a length greater than 255
  bytes, allow online table reorg and online table load to be used on the table, and allow
  usage of the new multidimensional clustering facility.
- All new indexes are created as type-2 indexes. If type-1 indexes already exist in a table,
  new index will also be a type-1 index. Type-1 and type 2 indexes cannot coexist on a table.
- All indexes created before Version 8 were type-1 indexes. To convert type-1 indexes to
  type-2 indexes, use the REORG INDEXES command.
  ➤ REORG INDEXES ALL FOR TABLE <tablename> CONVERT
- To find out what type of index exists for a table, use the INSPECT command
  ➤ INSPECT CHECK TABLE NAME <tablename> INDEX NORMAL RESULTS <filename>
- If data in the database is corrupted or damaged, it is possible that the indexes associated
  are also damaged or marked as invalid by DB2. The configuration parameter INDEXREC
  indicates when DB2 will attempt to rebuild invalid indexes.
- There are three possible settings for this parameter:
  ➤ SYSTEM means use the INDEXREC setting specified in the database manager configuration file
  ➤ ACCESS means invalid indexes are rebuilt when the index is first accessed.
  ➤ RESTART means invalid indexes are rebuilt during database restart via the RESTART DATABASE
  command.

Multi-Dimensional Clustering

- Multi-dimensional Clustering
  ➤ Provides range partitioning on multiple dimensions
  ➤ Reduces need for indexing

Prior to MDC
- Clustering in one dimension only
- clustering NOT guaranteed
  (degrades once page free space is exhausted)

With MDC
- Clustering guaranteed !
- Smaller indexes
- Faster query response
- Simple definition syntax
- Fast roll-in & roll-out
Design Advisor

- Design Advisor can help you design and define suitable indexes which can be invoked using the DB2 Control Center or db2advis command.
- The db2advis command:

```
>>>db2advis -- d--database-name-- -------------------------->
+--w--workload-name--
+--s--"statement"--
+--i--filename----+
+--g-------------------->

>>>--a--userid----------'
'--l--disk-limit--'
'--/passwd--'

'--t--max-advise-time--'
'--h--'
'--p--'
'--o--outfile--'
```

- `-l disk-limit` (specifies the maximum space to be used for all recommended indexes)
- `-t max-advise-time` (specifies the maximum allowable time in minutes to complete the operation)
- Example:
  - `db2advis -d prototype -w production -l 53 -t 20`
    - The utility connects to the PROTOTYPE database, and recommends indexes that will not exceed 53MB for queries and workload name is "production", the maximum allowable time for finding a solution is 20 minutes

Database Objects

Constraints

- Referential Integrity or Referential Constraints are established with the
  - Primary Key clause
  - Unique constraint clause
  - Foreign Key clause
  - References clause

- Referential Integrity rules:
  - Insert rules (Rule is implicit when a foreign key is specified)
  - Delete rules (Restrict, Cascade, No Action, Set Null)
  - Update rules (Restrict, No Action)

- Unique Contraints

<table>
<thead>
<tr>
<th>Unique Key</th>
<th>Unique Index</th>
<th>Primary Key</th>
<th>Unique Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>all values of the key are unique</td>
<td>can have multiple unique index in a table</td>
<td>can only have one primary key in a table</td>
<td>created when primary key or unique clause is used</td>
</tr>
<tr>
<td>cannot contain NULL</td>
<td>allow only one NULL value</td>
<td>cannot contain NULL</td>
<td>cannot contain NULL</td>
</tr>
<tr>
<td>it is a type of unique index</td>
<td>if an index already exists, unique index is created</td>
<td>if an index does not already exist, primary key is created</td>
<td></td>
</tr>
<tr>
<td>can have multiple unique constraint in a table but only one can be primary key</td>
<td>can have more than one unique constraint on the same set of columns</td>
<td>cannot contain NULL</td>
<td></td>
</tr>
</tbody>
</table>

CREATE TABLE artists (artno INT, ...............

  PRIMARY KEY (artno)

  FOREIGN KEY (workdept)

  REFERENCES department ON DELETE NO ACTION)

IN DMS01 ;
Constraints ... continued

- **Check Constraint** enforces data integrity at a table level. Once defined every update/insert must conform, otherwise it will fail.

```sql
CREATE TABLE artists
(artno SMALLINT NOT NULL,
 name VARCHAR(50) WITH DEFAULT 'abc',
 classification CHAR(1) NOT NULL,
 bio CLOB(100K) LOGGED,
 picture BLOB(2M) NOT LOGGED COMPACT )
CONSTRAINT classify
CHECK (classification IN ('C','E','P','R'))
```

If some rows do not meet the constraint then it will fail. You can turn off checking, add the data and then add the constraint, but the table will be placed in CHECK PENDING.

- **Informational Constraint**
  - Rules that can be used in query rewrite but are not enforced by DB2.

- **Constraint Options**
  - ENFORCED - constraint is enforced by the database manager during normal operations such as insert, update, or delete
  - NOT ENFORCED - constraint is not enforced by the database manager
  - ENABLE QUERY OPTIMIZATION - constraint can be used for query optimization under appropriate circumstances
  - DISABLE QUERY OPTIMIZATION - constraint cannot be used for query optimization

Packages

- A package is a database object that contains information needed to execute specific SQL statements in a single source file.
- A database application uses one package for every precompiled source file that contains static or dynamic SQL statements.
- Packages are created by running the precompiler against a source file and bind the generated bind files.
- Use the BIND command or the Configuration Assistant to bind packages:
  - BIND <bind_filename>

Triggers, Functions, Stored Procedures

- A trigger defines a set of actions that are activated or triggered by an update operation on a specified base table. Three types of triggers: INSERT, UPDATE, DELETE. A trigger can be fired BEFORE or AFTER an event.
- INSTEAD OF triggers use view as single interface for ALL SQL operations

```sql
CREATE TRIGGER student_v_insert
    INSTEAD OF INSERT ON student_v
    REFERENCING NEW AS N DEFAULTS NULL
    FOR EACH ROW MODE DB2SQL
    BEGIN ATOMIC
        INSERT INTO students VALUES ( n.name, n.studentid ) ;
        INSERT INTO person VALUES ( n.name, n.studentid, n.age, n.enrolldate ) ;
    END
```

- There are three types of functions: Scalar, Column, and Table. They can be written in SQL, C, JAVA, and OLE.
- Stored procedures can be used to perform intermediate processing avoiding transmitting data across network, allow centralized administration and maintenance. They can be written in C, Java, COBOL, OLE, and SQL.

```sql
CREATE PROCEDURE update_salary
    ( IN employee_number CHAR(6), IN rate INTEGER, OUT newsalary INTEGER )
    LANGUAGE SQL BEGIN ... END
```

Usage: CALL update_salary ( 3422, 50, ? );
Basic SQLs

- The SQL Language has been developed around 4 basic verbs used for 4 major tasks:
  SELECT, INSERT, UPDATE, DELETE
- Optional clauses in SELECT statements:
  - SELECT ALL FROM employee FOR READ ONLY;
  - SELECT empno, salary, comm FROM employee FOR UPDATE OF salary, comm;
  - SELECT empno, salary FROM employee FETCH FIRST 10 ROWS ONLY;
  - SELECT empno, salary FROM employee OPTIMIZE FOR 10 ROWS;
  - SELECT DISTINCT (position) FROM employee;
  - SELECT empno, salary FROM employee WHERE NOT EXISTS (SELECT empno FROM emp);
  - SELECT empno, salary FROM employee WHERE department IN ('A00', 'B00', 'C00');
  - VALUES clause can be used alone to return constant values;
  - ORDER BY clause, GROUP BY clause, HAVING clause
- Declare cursor for a SELECT statement can take advantage of row blocking. Row blocking retrieves a block of rows in a single operation, rows stored in cache.
  - DECLARE CURSOR c1 FOR SELECT id, name FROM employee;
- Using Common Table Expression (CTE) like a "temporary view"
  - WITH staff (deptno, headcount) AS (SELECT deptno, count(*) FROM emp)
    SELECT deptno, headcount FROM staff WHERE headcount = 5;
- Set operators: UNION or UNION ALL, EXCEPT or EXCEPT ALL, INTERCEPT or INTERSECT ALL

Advanced SQLs

- Recursive SQL statement is used when an SQL repeatedly uses the resulting set to determine further results.
  ```sql
  WITH path (origin, destiny, distance, stops)
  AS (SELECT f.origin, f.destiny, f.distance
       FROM flights f
       WHERE origin='Sweden'
       UNION ALL
       SELECT p.origin, f.destiny,
       p.distance+f.distance, p.stops+1
       FROM flights f, path p
       WHERE p.destiny=f.origin)
  SELECT origin, destiny, distance, stops FROM path
  ```
- INNER JOINS - Result set consists only of those matched rows that are present in both joined tables.
- LEFT (or RIGHT) OUTER JOINS - Includes rows from the left (or right) table that were missing from the inner join.
- FULL OUTER JOINS - Includes rows from both the left and right tables that were missing from the inner join.
- OLAP functions: Grouping Sets, Group by rollup, Group by cube, Moving function.
- CASE expressions allow an expression to be selected based on the evaluation of one or more conditions.
Materialized Query Tables

- Aggregate Aware Optimization - the SQL compiler determines to use the MQT if a query will run more efficiently against a MQT than the base table or tables.
- Definition based on the result of a query, contains precomputed results.
- Two types of materialized query tables:
  - MAINTAINED BY SYSTEM - If base tables are updated, use the REFRESH option to indicate when the materialized query tables are refreshed.
  - MAINTAINED BY USER - Use custom applications to maintain and load the tables
- To manually refresh the materialized query table when the base table is changed, use the REFRESH TABLE statement.
- Example:
  ```sql
  CREATE TABLE abc ( col1, col2, col3, col4 )
  AS ( SELECT ..... FROM ..... )
  MAINTAINED BY SYSTEM
  ```
- Example:
  ```sql
  REFRESH TABLE abc INCREMENTAL
  ```
  - INCREMENTAL
    - Only refresh the appended portion content
  - NOT INCREMENTAL
    - Specifies a full refresh for the table by recomputing the materialized query table definition

Data Concurrency

Isolation Levels
Locking Mechanisms
Isolation Levels

- **DB2 provides different levels of protection to isolate data.**
- **Uncommitted Read**
  - It is the lowest level of isolation, provides most concurrency. It may return data that is never committed to the database. Non-repeatable reads and phantom reads may occur.
- **Cursor Stability (default)**
  - Lock on the row is held on which the cursor is positioned until the new row is fetched or the unit of work is terminated. If a row is updated, the lock is held until the unit of work is terminated. It cannot return uncommitted data but non-repeatable read and phantom reads may occur.
- **Read Stability**
  - Locks only the rows an application retrieves within a unit of work. No other application can change the rows that are locked by this application. It cannot read uncommitted data but may get 'phantom' rows if application issues the same query more than once within the unit of work.
- **Repeatable Read**
  - It is the highest isolation level and least concurrency. Locks held on all rows processed to build the result. Optimizer may choose to get a TABLE lock. Same query issued by the application more than once in a unit of work will give the same result each time. No other application can update, delete, or insert a row that would affect the result table until the unit of work completes.

Data Concurrency

Locking

- **Locking controlled by isolation level specified.**
- **Database, table spaces, and tables can be explicitly locked.**
  - Database lock - e.g: CONNECT TO dbname IN EXCLUSIVE MODE
  - Table space lock - e.g.: QUIESCE TABLESPACES FOR TABLE tabname INTENT FOR UPDATE
  - Table lock - e.g.: LOCK TABLE tabname IN EXCLUSIVE MODE
- **Database, tables, and rows can be implicitly locked.**
  - Database lock - during full database restore
  - Table lock - through Lock escalation
  - Row lock - through normal data modification
- **By default, DB2 uses row level locking. It can change to table level locking with:**
  - ALTER TABLE <tabname> LOCKSIZE TABLE
- **To engage row level locking is engaged again:**
  - ALTER TABLE <tabname> LOCKSIZE ROW
- **Lock escalation may happen if an application changes many rows in one table and the optimizer thinks it is better to have one lock on the entire table.**
  - Configuration parameters affect lock escalation: MAXLOCKS, LOCKLIST
- **Set the LOCKTIMEOUT parameter to minimize deadlocks and lock waits:**
  - LOCKTIMEOUT default is -1 or infinite wait.
  - Locktimeout error is SQL0911N with subcode “68” sqlstate: 40001
  - DeadLock error is SQL0911N with subcode “2” sqlstate: 40001
Storage Management

DB2 Memory Model

Managing Table Spaces
Performance Considerations

IBM Software Group
Memory for the buffer pool is allocated when the database is activated or when the first application connects to the database.

Command to create buffer pool:

CREATE BUFFERPOOL bpname IMMEDIATE SIZE sz PAGESIZE pgsz
CREATE BUFFERPOOL bpname DEFERRED SIZE sz PAGESIZE pgsz

- IMMEDIATE - the buffer pool will be created immediately. If there is not enough reserved space in the database shared memory, a warning is returned, and the statement is executed DEFERRED.
- DEFERRED - the buffer pool will be created when the database is reactivated.

Use the ALTER BUFFERPOOL command to increase the size of the buffer pool, memory is allocated as soon as the command is committed if the memory is available.

ALTER BUFFERPOOL bpname IMMEDIATE SIZE sz
ALTER BUFFERPOOL bpname DEFERRED SIZE sz

Information stored in SYSIBM.SYSBUFFERPOOLS

Management of Buffer Pools

- Memory for the buffer pool is allocated when the database is activated or when the first application connects to the database.
- Command to create buffer pool:

CREATE BUFFERPOOL bpname IMMEDIATE SIZE sz PAGESIZE pgsz
CREATE BUFFERPOOL bpname DEFERRED SIZE sz PAGESIZE pgsz

- IMMEDIATE - the buffer pool will be created immediately. If there is not enough reserved space in the database shared memory, a warning is returned, and the statement is executed DEFERRED.
- DEFERRED - the buffer pool will be created when the database is reactivated.

Use the ALTER BUFFERPOOL command to increase the size of the buffer pool, memory is allocated as soon as the command is committed if the memory is available.

ALTER BUFFERPOOL bpname IMMEDIATE SIZE sz
ALTER BUFFERPOOL bpname DEFERRED SIZE sz

Information stored in SYSIBM.SYSBUFFERPOOLS

Table Spaces

- Three types of table space: REGULAR, LARGE, TEMPORARY
- REGULAR table space stores all data except for temporary tables.
- LARGE table space stores long or LOB data, it must be a DMS table space.
- TEMPORARY table space, two types:
  - SYSTEM TEMPORARY table space is a work area used by DB2 to perform operations such as sorts or joins. A database must have at least one SYSTEM TEMPORARY table space.
  - USER TEMPORARY table space stores declared global temporary tables.

A table space is made of containers. Containers are allocation of physical space. A container can be a directory, a file, or a raw device.

- An extent consists of multiple pages and is an allocation of space in a container before the next container is used. EXTENTSIZE is defined at the table space, once a table space is created, EXTENTSIZE cannot be changed.
SMS Table Space Characteristics

- Containers are operating system directories, storage can be extended by enlarging the underlying OS file system.
- All table data and indexes share the same table space. Each table in a table space is given its own file name used by all containers.
- New containers can only be added to SMS on a partition that does not yet have any containers. Note on UNIX, file system size may be increased.
- Dynamic file growth. Disk space is allocated on demand.
  - Allocate one page at a time (default), use db2empfa utility to enable multiple page allocation.
- What happens on disk during the following?
  - CREATE TABLESPACE ts1 MANAGED BY SYSTEM USING ('/mydir1', '/mydir2') EXTENTSIZEx 4;
  - CREATE TABLE t1 ( c1 INT ... ) IN ts1;
  - CREATE TABLE t2 ( c2 FLOAT ... ) IN ts1;

DMS Table Space Characteristics

- Containers are files or raw devices. Space is allocated at table space creation time.
- Data striped across containers by extent. Space Map Pages (SMP) is used to keep track of what extents are used and which are free.
- Containers can be added or dropped, data is rebalanced automatically.
- Container size can be extended, reduced, or resized.
- Flexible data placement because table objects (i.e. data, index, long field data) can be placed into different DMS table spaces.
- Data and database objects are located by
  - Object table to locate first extent of the object.
  - Each object has Extent Map Pages (EMPs) to locate other extents of the object.
- What happens on disk during the following?
  - CREATE TABLESPACE ts2 MANAGED BY DATABASE USING
    ( FILE '/myfile' 1024,
    DEVICE '/dev/rhd7' 2048 )
    EXTENTSIZEx 4 PREFETCHSIZEx 8;
  - CREATE TABLE t1 ( c1 INT ... ) IN ts2;
  - CREATE TABLE t2 ( c2 FLOAT ... ) IN ts2;
CREATE DATABASE and TABLESPACE Examples

- CREATE DATABASE hrdb ON C
  USING CODESET codeset
  TERRITORY territory
  COLLATE USING SYSTEM
  CATALOG TABLESPACE
  MANAGED BY SYSTEM USING (’d:/db2/cattbsp’) 
  EXTENTS SIZE 16
  USER TABLESPACE
  MANAGED BY DATABASE
  USING (’d:/db2/user.f1’ 30M
          ,’d:/db2/user.f2’ 30M )
  EXTENTS SIZE 64
  PREFETCH SIZE 128
  TEMPORARY TABLESPACE
  MANAGED BY DATABASE USING
  ( DEVICE ’d:/db2/temp.f1’ 10M )
- CREATE TABLESPACE enterprise
  PAGESIZE 8K
  MANAGED BY SYSTEM
  USING ( ’/database/firstcnt’,
          ’/database/secondcnt’,
          ’/database/thirdcnt’
  )
  EXTENTS SIZE 16K
  PREFETCH SIZE 32
  BUFFERPOOL BP8K;
- CREATE USER TEMPORARY TABLESPACE usertemp
  MANAGED BY DATABASE
  USING ( DEVICE ’/dev/rusrtmp1’ 10M
          ,DEVICE ’/dev/rusrtmp2’ 10M )
  OVERHEAD 24.1 TRANSFER RATE 0.9;
- CREATE LARGE TABLESPACE kitbsp
  PAGESIZE 16K
  MANAGED BY DATABASE
  USING ( DEVICE ’/dev/rdb2lob1’ 1000
          ,DEVICE ’/dev/rdb2lob2’ 1000 )
  BUFFERPOOL BP16K;
  MANAGED BY DATABASE USING
  ( DEVICE ’d:/db2/temp.f1’ 10M )
- CREATE LARGE TABLESPACE lobtbsp
  PAGESIZE 16K
  MANAGED BY DATABASE
  USING ( DEVICE ’/dev/rdb2lob1’ 1000
          ,DEVICE ’/dev/rdb2lob2’ 1000 )
  BUFFERPOOL BP16K;
- EXTEND TABLESPACE <tbspname>
  RESIZE (FILE ’cont1’ 3000)
- REDUCE (ALL CONTAINERS 100)
- ADD (FILE ’cont2’ 2000, FILE ’cont3’ 2000);
- DROP (FILE ’cont1’ 2000);
- CREATE USER TEMPORARY TABLESPACE usertemp
  MANAGED BY DATABASE
  USING ( DEVICE ’/dev/rusrtmp1’ 10M
          ,DEVICE ’/dev/rusrtmp2’ 10M )
  OVERHEAD 24.1 TRANSFER RATE 0.9;
- CREATE TABLESPACE...
  Only attributes PREFETCH SIZE, BUFFERPOOL, OVERHEAD, TRANSFER RATE can be modified.
- Enable or disable Dropped TABLE RECOVERY
- Switch table space ONLINE after it was placed OFFLINE due to bad containers
- For SMS Table Space, container can only be added on a partition that currently has no containers.
- For DMS Table Space, containers can be added, dropped, or resized.
- ALTER TABLESPACE ts0 ADD (FILE ’cont2’ 2000, FILE ’cont3’ 2000);
- ALTER TABLESPACE ts0 DROP (FILE ’cont1’ 2000);
- ALTER TABLESPACE ts0 EXTEND (FILE ’cont0’ 100) RESIZE (FILE ’cont1’ 3000);
- ALTER TABLESPACE ts0 REDUCE (ALL CONTAINERS 100);

Managing Table Spaces

- List tablespaces [show detail]
  Lists states of tablespaces and storage usage.
- List tablespace containers
  Lists all containers for a table space, need to specify table space ID.
- RENAME TABLESPACE <oldname> TO <newname>;
- DROP TABLESPACE <tbspname>
  All tables, indexes, keys (primary & foreign) and constraints are dropped without warning EXCEPT if there is any table that spans on multiple table spaces, the table must be dropped first.
  Table space will not be dropped if there is any table that has the RESTRICT ON DROP attribute.
- ALTER TABLESPACE...
  Only attributes PREFETCH SIZE, BUFFERPOOL, OVERHEAD, TRANSFER RATE can be modified.
  Enable or disable DROPPED TABLE RECOVERY
  Switch table space ONLINE after it was placed OFFLINE due to bad containers
  For SMS Table Space, container can only be added on a partition that currently has no containers.
  For DMS Table Space, containers can be added, dropped, or resized.
Add and Extend Container Size, Stripe Sets

- When new containers are added to a table space or existing containers are extended, a rebalance of the table space data may occur.
- Access to the table space is not restricted during rebalancing (objects can be dropped, created, populated, and queried as usual) but there will be a significant impact on performance.
- The process of rebalancing when adding or extending containers involves moving table space extents from one location to another, and it is done in an attempt to keep data striped within the table space.
- The rebalancer starts at extent 0, moving one extent at a time until the extent holding the high-water mark has been moved. High-water mark is the page number of the highest allocated page in the table space, it can be obtained from the `LIST TABLESPACES SHOW DETAIL` command.

- If space is added above the high-water mark, rebalance will not occur. However adding a container will almost always add space below the high-water mark, meaning rebalance is often necessary.
- To force new containers to be added above the high-water mark, add a new stripe set.
- The existing containers in the existing stripe sets remain untouched, the new containers become part of a new stripe set.
- Example to add a new stripe set to a table space:
  - `ALTER TABLESPACE abc BEGIN NEW STRIPE SET ( FILE 'file1' 30, FILE 'file2' 40 );`
- It is also possible to add containers to existing stripe set. First obtain the valid stripe set to get a table space map by taking a table space snapshot using the snapshot monitor.
- Example to add containers to an existing stripe set:
  - `ALTER TABLESPACE abc ADD TO STRIPE SET 1 ( FILE 'file1' 30, FILE 'file2' 40 );`

I/O Cleaners and Servers

- Separate threads of control for fetching and writing pages to/from hard drives.
- Use of I/O cleaners and server can greatly enhance the performance.
- I/O cleaners are agents that will write dirty pages from buffer pool to disk before the space is required. This enables asynchronous writes.
- I/O servers (also called prefetchers) prefetch pages that might be required from disk to buffer pool. This enables asynchronous reads.
- Database configuration parameters are:
  - `NUM_IOCLEANERS`
    - Set it to be between one and # of physical storage devices used for the database but no more than # of CPUs of the system.
    - Parameters affect IO cleaners to be triggered are
      - `CHNGPGS_THRESH` - % of dirty pages in the buffer pool
      - `SOFTMAX` - influence # of logs needed to do crash recovery
  - `NUM_IOSERVERS`
    - Set it to one or two more than the number of physical devices on which the database resides.
    - The `PREFETCHSZ` table space option specifies how many pages are fetched for each prefetch request.
Maintaining Data

DB2 Export
DB2 Import
DB2 Load
DB2MOVE, DB2LOOK
REORGCHK, REORG, RUNSTATS, and REBIND
INSPECT Command

IBM Software Group

Data Movement Utility File Formats

- Types of data
  - Non-Delimited or Fixed Length ASCII (ASC)
  - Delimited ASCII (DEL)
  - PC version of the Integrated Exchange Format (IXF)
    - PC/IXF is a structured description of a database table
  - Worksheet Format (WSF)
    - For Export and Import only
  - Cursor
    - A cursor declared against an SQL query
    - For LOAD only
Export Utility

- Exports data from database table(s) to file using an SQL SELECT statement
- Format of exported data can be:
  - WSF - Worksheet Format
  - DEL - Delimited ASCII (DEL)
- Include the MESSAGES option to record error, warning, and informational messages received from the export.
- Must have SYSADM or DBADM authority, or CONTROL or SELECT privilege on table(s) being accessed in the export command.

Example:

```
EXPORT TO myfile OF IXF MESSAGES msg SELECT * FROM staff
```

Additional options to customize the export operation specified with MODIFIED BY:

```
EXPORT TO filename OF filetype MODIFIED BY filetype-mod
MESSAGES message-file select-statement
```

Example of file type modifiers:

- chardelx - specify x, a single character string delimiter, default is a double quotation mark ("")
- coldelx - specify x, a single character column delimiter, default is a comma (,)
- codepage=x - specify x, an ASCII character string, the code page of the data in the output file

Example:

```
EXPORT TO myfile.del OF del
MODIFIED BY chardel" coldel!
SELECT * FROM staff
```

Maintaining Data

Exporting Large Objects

- By default, the first 32 KB of LOB data is exported and place in the same file as the rest of the column data. If LOB greater than 32 KB, it is truncated.
- To store multiple LOBs in a single file, use the LOBSINFILE file modifier
- A LOB Location Specifier (LLS) is a string indicating where LOB data can be found within a file.

Example:

```
EXPORT TO empresume.del OF DEL
LOBs TO d:\lob1\ LOBFILE resume MODIFIED BY LOBSINFILE
SELECT * FROM emp_resume ;
```

Result:

File: empresume.del

```
"000130","asci","res.001.0.1313/"
"000130","script","res.001.1313.1817/"
"000140","asci","res.001.3130.1316/"
"000140","script","res.001.4446.1878/"
"000150","asci","res.001.6324.1363/"
"000150","script","res.001.7687.1923/"
"000190","asci","res.001.9610.1292/"
"000190","script","res.001.10902.1852/"
```

Directory: d:\lob1

res.001 - contains 8 LOB data
Import Utility

- Imports data from a file to a database table.
- The format of the input file can be IXF, WSF, DEL, or ASC (Fixed length ASCII).
- Include the MESSAGES option to record error, warning, and informational messages received from the import.
- Must have SYSADM or DBADM authority, or underlying privileges (SELECT, INSERT, CONTROL, or CREATETAB) on the target table.

Example:
```
IMPORT FROM myfile.ixf
  OF IXF MESSAGES msg.out
  INSERT INTO newtab ;
```

- INSERT option adds the imported data to the table, target table must exist.
- INSERT_UPDATE option adds rows of imported data to the target table, or updates existing rows of the target table with matching primary keys. Target table must exist.
- REPLACE options deletes all existing data from the table and inserts the imported data. Target table must exist. Table and index definitions are not changed.
- REPLACE_CREATE option
  - If the target table exists, deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.
  - If the table does not exist, creates the table and index definitions.
  - Can only be used with IXF files because they contain table structure descriptions.
- CREATE INTO option creates the target table and indexes. The table space clause can also be specified. Only PC/IXF input file format can be used.

Maintaining Data

- Indexes updated and referential integrity are checked during import.
- Import is logged. By default, import only commit once at the end of the operation.
- Make sure there is enough primary, secondary logs and log space. Should also use the COMMITCOUNT option to forces intermediate commits and the RESTARTCOUNT option to permit restart of import after failure.
- Use the MODIFIED BY option to customize the import operation. Example of modifiers:
  - compound=x - uses nonatomic compound SQL to insert the data, and x statements will be attempted each time.
  - provide column names to specify which columns data should go.

Example:
```
IMPORT FROM myfile IFX
  MODIFIED BY compound=5 COMMITCOUNT 100 MESSAGES msg.out
  INSERT INTO newtab ;
```

- To import LOB data, use the LOBSINFILE modifier to indicate the path of the LOB import files. Example:
```
IMPORT FROM asfile1 OF ASC
  LOBS FROM /u/db2load/lob1, /u/db2load/lob2
  MODIFIED BY lobsinfile
  INSERT INTO table1
```
Load Utility

- The load process has four phases:
  - **Load phase** loads data into tables, collects index keys and table statistics, records consistency points, and places invalid data rows in dump file and messages in message file.
  - **Build phase** creates indexes based on the keys collected during the load phase.
  - **Delete phase** deletes rows that caused unique key violation and places them in the exception table. Messages are recorded in the message file.
  - **Index Copy phase** only occur when online load is specified with the ALLOW READ ACCESS option. Index data is copied from a system temporary table space to the original table space.
- The format of the input source can be DEL, ASC, PC/IXF, or CURSOR (A cursor declared against an SQL query).
- Example of CURSOR as the input source:
  ```sql
  DECLARE mycurs CURSOR FOR SELECT col1, col2, col3 FROM tab1 ;
  LOAD FROM mycurs OF CURSOR INSERT INTO tab2 ;
  ```
- Include the MESSAGES option to record error, warning, and informational messages received from the load.
- Target must exist prior to load. Target cannot be a system or a temporary table.
- Authority and privilege required:
  - Must have SYSADM, DBADM, or LOAD authority on the database
  - INSERT privilege on the table when the load utility is invoked in INSERT mode
  - INSERT and DELETE privilege on the table when the load utility is invoked in REPLACE mode
  - INSERT privilege on the exception table if such a table is used as part of the load

Load Utility - Example

- Example:
  ```sql
  LOAD FROM emp.ixf OF IXF
  MODIFIED BY dumpfile = /u/db2/load/rejectrow.out
  ROWCOUNT 10000 SAVECOUNT 1000 WARNINGCOUNT 100
  MESSAGES msg.out
  TEMPFILES PATH /tmp/db2load/
  INTO employee
  FOR EXCEPTION except_table
  ALLOW READ ACCESS USE TABLESPACE systemptbsp
  CHECK PENDING CASCADE DEFERRED
  LOCK WITH FORCE ;
  ```
Let's look at each option

- **INSERT mode**
  - adds the loaded data to the table without changing the existing table data.

- **REPLACE mode**
  - deletes all existing data from the table, and inserts the loaded data. Table and index definitions are not changed. If an error occurs, the original data in the table is lost.

- **RESTART mode**
  - restarts a previously interrupted load. It will automatically continue from the last consistency point in the load, build, or delete phase.

- **TERMINATE mode**
  - terminates a load operation and rolls back the operation to the point in time at which it started, even if consistency points were passed. If the load operation being terminated is a load REPLACE, the table will be truncated. If the load operation being terminated is a load INSERT, original records are retained.

- **Load Dump File (dumpfile modifier)**
  - specifies the name and location of a file to store rejected rows that do not comply with the definition of the table.

- **Load Temporary Files (tempfiles path modifier)**
  - specifies the directory to store temp files during load processing. They are used for load crash recovery, load terminate operations, warning and error messages. They are removed when the load operation completes without error.

Let's look at each option ... continued

- **ROWCOUNT n**
  - specifies the number of n physical records in the file to be loaded.

- **SAVECOUNT n**
  - establishes consistency points after every n rows. Messages are generated at time of a save point. n should be sufficiently high enough so that performance will not be impacted.

- **WARNINGCOUNT n**
  - stops the load operation after n warnings.

- **Indexing mode**
  - specifies whether the load utility is to rebuild indexes or to extend them incrementally.
    - **AUTOSELECT** - Load will automatically decide between REBUILD or INCREMENTAL mode.
    - **REBUILD** - All indexes will be rebuilt.
    - **INCREMENTAL** - Indexes will be extended with new data.
    - **DEFERRED** - Load will not attempt index creation, indexes will be marked as needing a refresh. When the database is restarted or first access to such indexes may cause the index to be rebuilt.
Load Exception Table

- Load exception table is specified by the FOR EXCEPTION clause. It is a user-defined table used to store copies of rows that violate unique index rules. Load does not check for constraints or foreign key violations other than violations of uniqueness.
- Load exception table has to have the same definition of the table being loaded. If at least one of the columns is not present in exception table, offending row is discarded. No additional column is allowed in the exception table.
- Two optional additional columns can be added to the end of the table:
  - Timestamp column is a timestamp recording when row inserted.
  - Message column is a CLOB(32K) or larger recording the associated error message.

```
LOAD FROM emp.ixf OF IXF
MESSAGES emp.msg
MODIFIED BY
DUMPFILE=c:\emp.dmp
TEMPFILES PATH d:\tmp
INSERT INTO employee
FOR EXCEPTION empexp
```

Examine the message file emp.msg and the exception table empexp

<table>
<thead>
<tr>
<th>INPUT</th>
<th>not null, numeric column</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-0</td>
<td>-1</td>
</tr>
<tr>
<td>6-0</td>
<td>-1</td>
</tr>
<tr>
<td>3-0</td>
<td>-3</td>
</tr>
<tr>
<td>0-0</td>
<td>-4</td>
</tr>
<tr>
<td>4-0</td>
<td>-x</td>
</tr>
<tr>
<td>5-0</td>
<td>-6</td>
</tr>
<tr>
<td>0-0</td>
<td>-7</td>
</tr>
<tr>
<td>8-0</td>
<td>-8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>empexp</th>
<th>emp.dmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>30-0</td>
<td>4</td>
<td>timestamp</td>
</tr>
<tr>
<td>50-0</td>
<td>7</td>
<td>timestamp</td>
</tr>
</tbody>
</table>

Rows not Loaded; third column does not comply with NOT NULL and NUMERIC requirements

Table Access During Load

- ALLOW NO ACCESS (default behavior)
  - Target table is locked for exclusive access.
  - Table state is set to LOAD IN PROGRESS during the load.
  - It is the only valid option for LOAD REPLACE.
  - If there are constraints on the table, the table state is also set to CHECK PENDING, use the SET INTEGRITY command to validate the data against the constraints.

- ALLOW READ ACCESS
  - Target table is locked in share mode.
  - This option is not supported with REPLACE mode because existing data is truncated at the beginning of the load.
  - Readers may access the non-delta portion of the data while the table is being loaded.
  - If load fails, data that existed in the table prior to the load operation will continue to be available in read only mode after the failure.
  - If there are constraints on the table, the table state is also set to CHECK PENDING.
    - Use the SET INTEGRITY command to validate the new portion of the data (if possible) against the constraints.
  - Specify the USE TABLESPACE <tbsp-name> option
    - If a full index is being rebuilt, a shadow copy of the index is built in the specified table space (system temporary table space only) and copied over to the original table space during the INDEX COPY PHASE.
    - The target table is taken offline when the new indexes are copied to the target table space.
    - If system temporary table space not specified, then the shadow index will be created in the same table space as the index object.
LOAD Option - CHECK PENDING CASCADE

- If there are constraints defined on the table, LOAD puts it into a check pending state. The CHECK PENDING CASCADE option allows the user to specify whether or not to cascade this check pending state all descendents, for example descendent foreign key tables.
- **CHECK PENDING CASCADE IMMEDIATE**
  - Indicates that the check pending state for foreign key constraints is immediately extended to all descendendent foreign key tables.
- **CHECK PENDING CASCADE DEFERRED**
  - Indicates that only the loaded table will be placed in the check pending state.

**SET INTEGRITY Command**

- It is used to turn off integrity checking for one or more tables.
- Integrity checking options: Foreign Key, Check, Materialized Query, DATALINK, Generated Column, Staging
- **SET INTEGRITY command options:**
  - INCREMENTAL (default) - Forces incremental constraint checking on the appended data.
  - IMMEDIATE CHECKED - Takes table out of check pending state and carry out deferred integrity checking.
  - IMMEDIATE UNCHECKED - Takes table out of check pending state without being checked for integrity violations.
- **CONST_CHECKED columns of SYSCAT.TABLES** indicates status of each constraint defined in the table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Checked by SYSTEM</td>
</tr>
<tr>
<td>N</td>
<td>Not checked ( In CHECK PENDING)</td>
</tr>
<tr>
<td>U</td>
<td>Checked by USER (As a result of SET INTEGRITY ...IMMEDIATE UNCHECKED)</td>
</tr>
<tr>
<td>W</td>
<td>Previously checked by USER and some data needs to be verified by SYSTEM ( In CHECK PENDING)</td>
</tr>
<tr>
<td>F</td>
<td>Byte 5 - materialized query table cannot be refreshed incrementally  Byte 7 - content of staging table is incomplete and cannot be used for incremental refresh of the associated materialized query table</td>
</tr>
</tbody>
</table>
MODIFIED BY Clause and Other Options

- Some file modifier examples:
  - `dumpfile` specifies file to store rejected rows.
  - `fastparse` reduces syntax checking on loaded data to enhance performance.
  - `generatedignore, generatedmissing, generatedoverride, identityignore, identitymissing, identityoverride` ignore or override generated value / identity column data, or indicate generated values / identity column data are missing.
  - `indexfreespace, pagefreespace, totalfreespace` leaves specified amount of free space in index pages and data pages.
  - `lobsinfile` specifies the path to the files containing LOB data.

- Other options and considerations:
  - `STATISTICS YES/NO` specifies if statistics is to be gathered during the load. This option is supported in REPLACE mode only.
  - Table space in which the loading table resides is left in BACKUP PENDING state at completion of load regardless if it is successful or not. A backup is required before the table space can be used.
  - `COPY YES/NO` saves a copy of the loaded data in the specified location. Table space will not be in BACKUP PENDING. This allows the table space to be recoverable after load in case of failure. Database must be enabled with forward log recovery to use this option.
  - `NONRECOVERABLE` specifies that the load transaction is to be marked as non-recoverable. Table space is not put in backup pending state and the loaded table is not recoverable if a rollforward is performed on the table space. If load fails, table must be dropped and recreated.

- `LOAD QUERY` command checks the status of a load operation during processing and returns the table state. e.g. `LOAD QUERY TABLE staff TO /u/mydir/staff.tempmsg`

---

IMPORT vs LOAD

<table>
<thead>
<tr>
<th>IMPORT</th>
<th>LOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slower on large amounts of data</td>
<td>Faster on large loads - writes formatted pages</td>
</tr>
<tr>
<td>Creation of tables &amp; indexes with IXF format</td>
<td>Tables and indexes must exist</td>
</tr>
<tr>
<td>WSF supported</td>
<td>WSF not supported</td>
</tr>
<tr>
<td>Import into tables and views (Aliases supported)</td>
<td>Load tables only (Aliases supported)</td>
</tr>
<tr>
<td>No support for importing into materialized query tables</td>
<td>Support for loading into materialized query tables</td>
</tr>
<tr>
<td>All rows logged</td>
<td>Minimal logging performed</td>
</tr>
<tr>
<td>Triggers will be fired</td>
<td>Triggers not supported</td>
</tr>
<tr>
<td>Temporary space used within the database.</td>
<td>Temporary space used outside the database.</td>
</tr>
<tr>
<td>Constraints validated during import</td>
<td>All Unique key is verified during load Other constraints are validated with the SET INTEGRITY command</td>
</tr>
<tr>
<td>If interrupted table is usable with data up to the last commit point.</td>
<td>If interrupted the table is held in LOAD PENDING state. Either restart or restore tables effected.</td>
</tr>
<tr>
<td>Run RUNSTATS after import for Statistics</td>
<td>Statistics can be gathered during Load</td>
</tr>
<tr>
<td>Import into mainframe database via DB2 Connect</td>
<td>Cannot load into mainframe database</td>
</tr>
<tr>
<td>No back-up image required</td>
<td>Backup can be created during load</td>
</tr>
</tbody>
</table>
Database Movement Tool Commands - db2move, db2look

- db2move can be used to move large numbers of tables between DB2 databases.
- It queries the system catalog tables, compiles a list of all user tables, and exports these tables in PC/IXF format.
- **db2move** <dbname> <action: export/import/load> <options>

**Example:**

```bash
db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2
```

Export all tables created by "userid1" or user IDs LIKE "us%rid2", and with the name "tbname1" or table names LIKE "%tbname2".

- db2look is a tool that can:
  - Extract DDL statements to reproduce database objects.
  - Generate UPDATE statements to replicate statistics on the objects.
  - Generate UPDATE statements to update database configuration dbm config parameters and db2set statements to modify registry variables.
  - Extract statistics or report on statistics.

**Example:**

```bash
db2look -d department -u peter -e -o db2look.sql
```

Generate DDL statements for objects created by user peter in database DEPARTMENT

The db2look output is sent to file db2look.sql.

Data Maintenance Utilities

- **RUNSTATS Utility**
  - Updates statistics about the physical characteristics of a table and the associated indexes.
  - Examples:
    ```bash
    RUNSTATS ON TABLE db2user.employee
    WITH DISTRIBUTION
    ON COLUMNS (empid, empname)
    ALLOW READ ACCESS
    ```
  - Collect statistics on the table only, on columns empid and empname with distribution statistics.
  - Other users can have read-only access to the table while statistics are calculated.

- Statistics of an index can also be collected at index creation time
  - **CREATE INDEX item ON stock (itemno) COLLECT DETAILED STATISTICS**

- **REORGCHK Utility**
  - Calculates statistics on the database to determine if tables or indexes, need to be reorganized.
  - Examples:
    ```bash
    REORGCHK CURRENT STATISTICS ON TABLE USER
    ```
  - Checks the tables that are owned by the run time authorization ID

- **REORG Utility** physically reorder tables specified.
  - Offline REORG
    ```bash
    ALLOW READ ACCESS is default, it can also specify no access with ALLOW NO ACCESS.
    ```
  - Online (or In-place) REORG
    ```bash
    Takes longer time to complete
    Requires more log space to recover from unexpected failure
    Can be paused and resumed later.
```
REBIND Utility, FLUSH PACKAGE CACHE Statement

- REBIND utility allows the user to recreate a package stored in the database without the need for a bind file.
  - Must use qualified package name or it will assume the current authorization ID
  - Example: REBIND PACKAGE <package-name>

- Dynamic SQL statements are cached in the package cache. After RUNSTATS, the dynamic SQL statements should be prepared again so that the latest database statistics is used.
- Use the FLUSH PACKAGE CACHE statement to remove all cached dynamic SQL statements currently in the package cache.
- Cached dynamic SQL statements that are not in use are invalidated.
- Any cached dynamic SQL statement currently in use will continue be used until it is no longer needed by its current user; the next new user of the same statement will force an implicit prepare of the statement by DB2, and the new user will execute the new version of the cached dynamic SQL statement.
- Example:
  - FLUSH PACKAGE CACHE

Maintaining Data

INSPECT Command

- INSPECT checks the database for architectural integrity, database pages for page consistency, and the structures of table objects and structures of table spaces validity.
- INSPECT is online and will not lock objects (uses uncommitted read isolation level).
- Objects can be specified with INSPECT are databases, table spaces, and tables.
- The INSPECT check processing will write unformatted data results to a specified file. This result data can be formatted by the DB2INSPF utility for inspection.
- Examples: INSPECT CHECK TABLE NAME staff INDEX NORMAL RESULTS inspStaff.chk

SQL1141N The operation completed with errors or warnings. The details are located in the result file “inspStaff.chk” that needs to be formatted with the db2inspf utility.

db2inspf inspStaff.chk inspStaff.out
- If there is error, the output file will be stored in the same directory as the db2diag.log (i.e. according to DIAGPATH parameter setting).
Database Recovery

Database Logging
Backup and Recovery Utilities

IBM Software Group

Types of Recovery

- Crash / Restart Recovery
  - An automatic database recovery if the database was terminated abnormally (e.g. power failure).
  - Uses the RESTART DATABASE command or the automatic restart enable configuration parameter (autorestart) to protect a database from being left in an inconsistent state.
  - Example:
    - RESTART DATABASE sample

- Version/Image Recovery
  - Uses the BACKUP command in conjunction with the RESTORE command to put the database in a state that was previously saved.
  - Used for nonrecoverable databases or databases for which there are no archived logs.

- Rollforward Recovery
  - Uses the BACKUP command in conjunction with the RESTORE and ROLLFORWARD commands to recover a database or table space to a specified point in time.
  - The logging mechanism used in the database determines whether the database is recoverable or nonrecoverable.
  - Recoverable databases are databases that can be restored by CRASH, VERSION, or ROLLFORWARD recovery techniques.
  - Nonrecoverable databases only support CRASH and VERSION recovery.
**Transaction Log Files**

- Logs keep record of all database objects and data changes. Maximum log space allowed is 256GB. Logs can be written to raw devices, specified by parameter NEWLOGPATH.
- **Log Buffers**
  - All changes are written to log buffers first before being flushed to disk
  - Size of log buffer is defined with the database configuration parameter LOGBUFSZ
  - LOGBUFSZ is allocated from a memory area called the database heap (DBHEAP)

![Diagram of Transaction Log Files]

**Circular and Archive Logging**

- **Circular Logging (default)**
  - Primary log files (logprimary) used to record all changes; reused when changes are committed.
  - Secondary log files (logsecond) allocated when limit of primary log files reached.
  - If both primary and secondary log limit is reached, an error code is returned. If logpath file system has insufficient space, a 'log disk full error' will be raised.
  - Crash and version recovery possible; roll-forward recovery not possible.

- **Archive logging**
  - Used specifically for rollforward recovery. It is enabled with the LOGRETAIIN and/or USEREXIT database configuration parameter.
  - Logs are archived when they are no longer active (i.e. not required for crash recovery). They are not deleted; they are kept online or offline.
  - If both primary and secondary log limit is reached, an error code is returned. If logpath file system has insufficient space, a 'log disk full error' will be raised.
  - Rollforward recovery and online backup are possible.
How are retained logs handled? Manually or USEREXIT

- With log retention, all logs will be kept in the log path unless userexits are enabled or they are moved manually.
- Userexits can be used to archive log files that are full to another path/drive or tape device.
  - They will be marked as "reusable" once they are inactive.
- With USEREXIT ON, a userexit program will be called by the DB2 system controller for every log file as soon as it is full.
- During roll forward, userexit may be called to get the log file if it is not in the current log path. This is only done for full database restore, not table space.
- Sample userexits included with DB2 that can be modified for any installation and are found under ../sqllib/samples/c/db2uext2.

How Userexits Work

For this example assumes 3 primary logs.

Other Logging Features and Considerations

- Block transaction when log directory is full
  - If the log path is full and no new log files can be created to record database changes, the transaction will receive a log disk full error and will be rolled back. To increase database availability, set the database configuration parameter BLK_LOG_DSK_FUL to YES so that DB2 will not report any error because of disk full in the active log path.
  - DB2 will attempt to create the log file again after 5 minutes. This behavior will continue until log is successfully created. Messages are written to the administration notification log.
  - Until the log file is successfully created, any user application that attempts to update table data will not be able to commit transactions. Read-only queries are normally not affected unless they dependent on rows being locked by any update transactions.

- Infinite active logging
  - Allows a single transaction to use an infinite number of log files. This can be enable by setting database configuration parameter logsecond to -1. Userexit is used to archive log files, therefore userexit has to be enabled.

- Log mirroring
  - The MIRRORLOGPATH database configuration parameter allows the database to write an identical second copy of log files to a different path. If an error is encountered on either path, that log path will no longer be used until the DB attempts to access the next log file. If a subsequent failure occurs while writing to the remaining "good" path, the database shuts down.
  - DB2 will not synchronize the log paths but it keeps information about access errors that occur, so that the correct paths are used when log files are archived.

- On demand archive log
  - Use the ARCHIVE LOG command to close and archive an active log for a recoverable database at any time. Example: ARCHIVE LOG FOR DATABASE sample
Backup Utility

- Full offline database backup
  - Exclusive database lock is required, no other database operation can be performed.
  - Example: `BACKUP DATABASE sample TO /db2backup/dir1, /db2backup/dir2 WITH 4 BUFFERS BUFFER 4096 PARALLELISM 2`

- Full online database backup
  - Database must have LOGRETAIN and/or USEREXIT enabled. Other database operations are allowed.
  - Transaction log files are used to capture changes during the online backup.
  - Example: `BACKUP DATABASE sample ONLINE TO /dev/rdir1, /dev/rdir2 ;`
  - Example: `BACKUP DATABASE sample TABLESPACE (userspace1, userspace2) ONLINE TO /db2tbsp/backup1, /db2tbsp/backup2 ;`

- Two kinds of incremental backups are available for both database and table space:
  - Incremental - backup all of the data that has changed since the last full backup.
  - Delta - backup all data that has changed since the last successful full, incremental, or delta backup

Backup files:

Intel

<table>
<thead>
<tr>
<th>Alias</th>
<th>Instance</th>
<th>Year</th>
<th>Day</th>
<th>Minute</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBALIAS.0</td>
<td>DB2INST.0</td>
<td>19960314</td>
<td>13</td>
<td>1259</td>
<td>001</td>
</tr>
</tbody>
</table>

UNIX

<table>
<thead>
<tr>
<th>Alias</th>
<th>Instance</th>
<th>Year</th>
<th>Day</th>
<th>Minute</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBALIAS.0</td>
<td>DB2INST.0</td>
<td>19960314</td>
<td>13</td>
<td>1259</td>
<td>001</td>
</tr>
</tbody>
</table>

Cumulative Backups

Full

Delta Backups

Full

Restore Utility

- `RESTORE` rebuilds an existing or create a new database or table space that has been backed up using the backup utility.
- Example: `RESTORE DATABASE sample FROM /dev/rdir1, /dev/rdir2 TAKEN AT 2001-12-24:10:23:42 INTO sample NEWLOGPATH /db2/sample/newlog REPLACE EXISTING WITHOUT PROMPTING`

- Rollforward a database with the `ROLLFORWARD` command after `RESTORE`. Log files are required and transactions are applied to the database.
- Example: `ROLLFORWARD DATABASE sample TO END OF LOGS AND COMPLETE ;`
- `ROLLFORWARD` DATABASE sample TO timestamp USING LOCAL TIME AND STOP;
- `COMPLETE` or `STOP` is required to take the database out of the roll forward pending state.
- Once the `ROLLFORWARD` command is issued with `COMPLETE` or `STOP` option, it is not possible to roll forward additional changes.

Backup

- CHECK DATABASE SEED
- RETAIN DATABASE DIRECTORIES
- REPLACE TABLE SPACE ENTRIES
- RETAIN RECOVERY HISTORY IF IT IS NOT DAMAGED
- RETAIN AUTHENTICATION
- RETAIN DATABASE
- DELETE TABLE, INDEX, LONG FIELD FILES

NEW

- CREATE NEW DATABASE
- RESTORE AUTHENTICATION
- RESTORE DATABASE CONFIGURATION FILE
- SET DEFAULT LOG PATH OR USE NEWLOGPATH IF SPECIFIED
- RESTORE COMMENTS

EXISTS
Redirected Restore

- **Redirected Restore:**
  - This can be used to redefine table space containers sizes and locations.
  - Backup images record table space containers information.
  - If containers not available during restore, new containers can be specified through the use of the restore utility.
  - Use the REDIRECT option in the RESTORE command and the SET TABLESPACE CONTAINERS command to do redirect restore.

**EXAMPLE**

```sql
RESTORE DATABASE DB2CERT FROM C:\DBBACKUP
   INTO NEWDB REDIRECT WITHOUT ROLLING FORWARD
```

SQL1277N Restore has detected that one or more table space containers are inaccessible, or has set their state to 'storage must be defined'.

DB20000I The RESTORE DATABASE command completed successfully.

```sql
SET TABLESPACE CONTAINERS FOR 0 USING (FILE "d:\newdb\cat.dat" 5000)
SET TABLESPACE CONTAINERS FOR 1 USING (PATH "d:\newdb2")
RESTORE DATABASE DB2CERT CONTINUE
```

Some Restore Considerations

- **Restore a database with incremental backup images, incremental options:**
  - AUTOMATIC - specifies an automatic cumulative restore operation, the restore utility uses the backup history file to backup image needed.
  - ABORT - aborts an in-progress manual cumulative restore.

- **Example:**
  ```sql
  RESTORE DATABASE sample INCREMENTAL AUTOMATIC FROM /db2backup/dir1;
  ROLLFORWARD DATABASE sample TO END OF LOGS AND COMPLETE;
  ```

- **Restoring a backup on different platforms or word sizes:**
  - Database created and backed up in 32-bit Windows can be restored to a 64-bit Windows platform.
  - Database created and backed up in 32-bit Linux can be restored to a 64-bit Linux platform.
  - Database backups created on AIX, HP, and Sun operating systems in 32-bit or 64-bit, may be restored on AIX, HP, and Sun operating systems in either 32-bit or 64-bit.
  - Incremental and delta images cannot be restored when there is a difference in operating systems or word size (32-bit or 64-bit).
Table Space Recovery

- To restore a table space, use the RESTORE DATABASE command with the TABLESPACE option, example: 
  ```sql
  RESTORE DATABASE sample TABLESPACE ( userspace1 ) ONLINE 
  FROM /db2tbsp/backup1, /db2tbsp/backup2 ;
  ```
- After a table space is restored, it is always in rollforward pending state. Table space must be rolled forward to a minimum Point In Time (PIT). Exception: System catalog table space must be rolled forward to the end of logs and in offline mode.
- Minimum PIT ensures the table space and logs agree with what is in the system catalogs. It can be obtained from:
  - LIST TABLESPACES SHOW DETAIL command OR
  - Table space snapshot - GET SNAPSHOT FOR TABLESPACE ON dbname
- After a table space point-in-time rollforward operation completes, the table space is put in backup pending state. Table space must be backed up because all updates made to it between the point in time recovered and the current time have been removed.
- By default, DB2 will process all log files even if they do not contain log records that affect that table space. For rollforward to process only log files that are required, set the registry variable `DB2_COLLECT_TS_REC_INFO = ON`
- This command can be used to find out the next archive log file required:
  ```sql
  ROLLFORWARD DATABASE sample QUERY STATUS USING LOCAL TIME ;
  ```

Summary of Table Space States

- Rollforward pending
  - After a table space restore, the table space is placed in rollforward pending.
  - Rollforward to the end of the logs or to a point in time or specify COMPLETE to get it out of the pending state.
- Rollforward-in-progress
  - A table space is put in this state when a rollforward on that table space is in progress.
- Restore pending
  - If a rollforward operation on that table space encounters an unrecoverable error, in which case the table space must be restored and rolled forward again.
- Backup pending
  - A table space is put in this state after a point-in-time rollforward operation, or after a load operation with the no copy option.
  - The table space must be backed up before it can be used.
  - If it is not backed up, the table space cannot be updated, but read only operations are allowed.
- Drop pending
  - Nothing can be done with the table space except to drop it.
- Offline
  - Table space is offline because the database manager cannot access one or all of its containers.
Set DBM and DB Configuration Parameters to AUTOMATIC

- Quick review of display / set / reset DBM and DB configuration parameters:
  - GET DBM CFG [SHOW DETAIL]
  - UPDATE DBM CFG using <parm> <value>, <parm> <value> ... [IMMEDIATE | DEFERRED]
  - RESET DBM CFG
  - GET DB CFG for <dbname> [SHOW DETAIL]
  - UPDATE DB CFG for <dbname> using <parm> <value>, <parm> <value> ... [IMMEDIATE | DEFERRED]
  - RESET DB CFG FOR <dbname>

- Some DBM and DB configuration parameters supports the value AUTOMATIC.
- If set to automatic, DB2 will automatically adjust the parameter to reflect current resource requirements.

- DBM configuration parameter that supports AUTOMATIC:
  - instance_memory - Amount of memory reserved for the instance

- DB configuration parameter that supports AUTOMATIC:
  - database_memory - Size of Database Shared Memory
  - maxappls - Maximum Number of Active Applications
Tuning Database Manager Configuration Parameters

- **NUMDB**
  - It limits the maximum number of concurrently active databases.
  - Value should reflect reality as closely as possible.
  - Each database takes up storage and an active database uses a new shared memory segment; system resource can be reduced by limiting the number of separate databases on a server.

- **MAXAGENTS**
  - Maximum number of database manager agents available at any given time to accept application requests. This value limits the memory usage of the instance, because each additional agent requires additional memory.

- **NUM_POOLAGENTS**
  - A guideline for how large the agent pool should grow. If more agents are created than this value, they will be destroyed when execution is completed.

- **SHEAPTHRES**
  - Controls the total amount of memory for sorting available across the entire instance for all sorts.
  - Should be roughly 10x SORTHEAP (DB CFG).

Tuning Database Configuration Parameters

- **MAXAPPLS**
  - Specifies the maximum number of concurrent applications (local and remote) allowed to be connected to the database.

- **AVGAPPLS**
  - This parameter is used by the SQL optimizer to help estimate how much buffer pool will be available at run-time for the access plan chosen. The SQL optimizer should know the number of concurrent (complex) queries that will be running in the system so that it can be more conservative in assumptions of buffer pool availability.

- **SORTHEAP**
  - This parameter defines the maximum number of private memory pages to be used for private sorts, or the maximum number of shared memory pages to be used for shared sorts. If the sort is a private sort, then this parameter affects agent private memory. If the sort is a shared sort, then this parameter affects the database shared memory.

- **SHEAPTHRES_SHR** (sorheap threshold for shared sorts)
  - This parameter represents a hard limit on the total amount of database shared memory that can be used for sorting at any one time.
CHNGPGS_THRESH
- Specifies the level (percentage) of changed pages at which asynchronous page cleaners will be started.

NUM_IOCLEANERS
- Specifies the number of asynchronous page cleaners. To avoid I/O wait, set this parameter to a higher value if insert/update/delete activity is heavy.

NUM_IOSERVERS
- Specifies the number of I/O servers for a database. Set this value to be one or two more than the number of physical devices present on the server to maximize I/O parallelism.

LOCKLIST
- One lock list per database, it contains locks held by all connected applications.

MAXLOCKS
- Defines a percentage of the lock list held by an application that must be filled before the database manager performs lock escalation.

MINCOMMIT
- Delay writing of log records to disk until min # commits performed. Increase count if high update activity from many concurrent users.

---

Query Parallelism

- Inter-partition parallelism - Parallel Query
- Intra-partition parallelism - Parallel Transaction
  - DB2 is able to take advantage of multiple processors machine to parallelize non update operations. Enabled by INTRA_PARALLEL parameter. Data parallelism divides data based on the number of processors.
  - Parallelism is enabled at the instance level (DBM CFG):
    - INTRA_PARALLEL = ON
  - To specify the degree of parallelism, several places can be set:
    - Instance level (DBM CFG) - MAX_QUERYDEGREE
    - Database level (DB CFG) - DFT_DEGREE
    - Statement Level - CURRENT DEGREE, DEGREE at bind time, RUNTIME DEGREE
    - Client Level (db2cli.ini) - DB2DEGREE
  - So, which degree of parallelism will be used? It will be the lowest of MAX_QUERYDEGREE, RUNTIME DEGREE, and Degree when the statement is precompiled or executed.
DB2 optimizer is cost based. It uses database statistics to determine how data is being accessed.

- Very important to keep statistics up to date
- Optimizer has a “throttle” to control how much optimization is done during access plan generation.
  - DFT_QUERYOPT of range 0 - 9 is the parameter to control optimization level. Default optimization level is 5, generally good for most queries.
- Consider reducing optimization level for simple statements and/or require short query compilation time. Complex queries should use at least the default query optimization level.
- Optimizer has ability to rewrite SQL statement to improve performance. Use Explain tool to see rewritten SQL statement.

DB2 Explain is a facility to capture detailed information about the access plan chosen by the SQL compiler.

- Timing information called “Timerons” is captured for each processing step. Timerons are unit of measurement to give a rough relative estimate of the resources, CPU and I/O costs.
- Access plan information captured in DB2 explain tables, created automatically or by executing EXPLAIN.DDL script.
- Information can be analyzed through text or GUI based tools (Visual Explain).

Text based explain tools available:
- db2expln - Capture access plan for static SQL statements in packages that are stored in the DB2, need to provide database name, package name, package creator, and section number.
- dynexpln - Capture access plan for dynamic SQL statement.
- db2exfmt - Formats contents of the explain tables.

There are four general methods of populating the Explain tables:

- EXPLAIN statement captures information about the access plan chosen for the supplied explainable statement and places this information into the Explain tables. The SNAPSHOT option indicated additional information are collected for Visual Explain.
  - Example: EXPLAIN ALL WITH SNAPSHOT FOR “SELECT * FROM STAFF”
- CURRENT EXPLAIN Special Register
  - SET CURRENT EXPLAIN MODE [ NO | YES | EXPLAIN ]
    - Gather information on dynamic SQL.
  - SET CURRENT EXPLAIN SNAPSHOT [ NO | YES | EXPLAIN ]
    - Gather explain snapshot data for dynamic SQL.
  - NO - Information is not gathered
  - YES - tables populated during execution of all statements until register set to NO.
  - EXPLAIN - tables populated without execution
- BIND options
  - Gather information during the bind process.
  - BIND package EXPLAIN [ NO | YES | ALL ]
    - EXPLAIN (only explain info)
  - BIND package EXPLSNAP [ NO | YES | ALL ]
    - EXPLSNAP (full explain plus snapshot)
  - NO - Information is not gathered
  - YES - static SQL information gathered
  - ALL - static & dynamic SQL information gathered
Provides cumulative information in the form of resetable counters.

- Monitors set by switches at the instance or application level with:
  - `UPDATE DBM CONFIGURATION` command (instance level)
  - `UPDATE MONITOR SWITCHES` command (application level)

- Can get a snapshot from:
  - CLP using the `GET SNAPSHOT` command
  - SQL table functions
  - Snapshot monitor APIs in a C or C++ application

- Seven monitor switches:

<table>
<thead>
<tr>
<th>Group</th>
<th>Info provided</th>
<th>Monitor Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorts</td>
<td>number of heaps used, overflows, sorts, performance</td>
<td>SORT</td>
</tr>
<tr>
<td>Locks</td>
<td>number of locks held, number of deadlocks</td>
<td>LOCK</td>
</tr>
<tr>
<td>Tables</td>
<td>measure activity on table (rows read, rows written)</td>
<td>TABLE</td>
</tr>
<tr>
<td>Buffer Pools</td>
<td>number reads and writes, time taken</td>
<td>BUFFERPOOL</td>
</tr>
<tr>
<td>UOW</td>
<td>start/end times and completion status</td>
<td>UOW</td>
</tr>
<tr>
<td>SQL</td>
<td>start/stop time statement identification</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>time stamps</td>
<td>TIMESTAMP</td>
</tr>
</tbody>
</table>
Display, Set, Reset Monitor Switches

- To check the status of the monitor switches:
  - `GET MONITOR SWITCHES`
- To activate monitor switches:
  - `UPDATE MONITOR SWITCHES USING`
    - BUFFERPOOL on LOCK on SORT on STATEMENT on TIMESTAMP on TABLE on UOW on
- The switches will remain active until the application (CLP) terminates, or until they are
deactivated with another UPDATE MONITOR SWITCHES command
  - `UPDATE MONITOR SWITCHES USING`
    - BUFFERPOOL off LOCK off SORT off STATEMENT off TIMESTAMP off TABLE off UOW off
- To reset counter of monitor switches:
  - `RESET MONITOR ALL`
- To obtain snapshots using CLP, examples:
  - `GET SNAPSHOT FOR DBM`
  - `GET SNAPSHOT FOR BUFFERPOOLS ON sample`
- To obtain snapshots using SQL queries with snapshot table functions, and receive result
  sets containing monitor data.
  - **Snapshot table functions:**
    - `SNAPSHOT_AGENT, SNAPSHOT_APPL, SNAPSHOT_APPL_INFO, SNAPSHOT_BP,`
      `SNAPSHOT_CONTAINER, SNAPSHOT_DATABASE, SNAPSHOT_DBM, SNAPSHOT_DYN_SQL,`
      `SNAPSHOT_FCM, SNAPSHOT_FCMNODE, SNAPSHOT_LOCK, SNAPSHOT_LOCKWAIT,`
      `SNAPSHOT_QUESCERS, SNAPSHOT_RANGERS, SNAPSHOT_STATEMENT, SNAPSHOT_SUBJECT,`
      `SNAPSHOT_SWITCHES, SNAPSHOT_TABLE, SNAPSHOT_TBREORG, SNAPSHOT_TBS,`
      `SNAPSHOT_TBS_CFG`

Capturing Snapshots Using SQL - Direct Access

- The monitor table functions have two input parameters:
  - Database name
    - If NULL is used, the database currently connected will be used
  - Partition number
    - Enter an integer between 0 and 999 corresponding to partition number wished to monitor
    - To capture a snapshot for the current connected partition, enter a value of -1 or NULL
    - To capture a global snapshot, enter a value of -2
- Example:
  - To capture a snapshot of the ranges for a table space map:
    ```sql
    SELECT * FROM TABLE(SNAPSHOT_RANGES('SAMPLE', -1))
    AS SNAPSHOT_RANGES
    ```
  - To capture a snapshot of dynamic SQL statement cache information:
    ```sql
    SELECT * FROM TABLE(SNAPSHOT_DYN_SQL('SAMPLE', -1))
    AS SNAPSHOT_DYN_SQL
    ```
  - To capture a snapshot of lock information:
    ```sql
    SELECT * FROM TABLE(SNAPSHOT_LOCK('SAMPLE', -1))
    AS SNAPSHOT_LOCK
    ```
  - To capture a snapshot of application information:
    ```sql
    SELECT agent_id, locks_held FROM TABLE(SNAPSHOT_APPL('SAMPLE', -1))
    AS SNAPSHOT_APPL
    ```
Event Monitors

- Event monitors are used to collect information about the database and any connected applications when specified events occur. For example: connections, deadlocks, statements, and transactions.
- Event monitors are database objects, they are created and manipulated using DML.
- There are no switches, simply define event monitors by the type of event wish to monitor.
- Event types:
  - DEADLOCKS - Applications involved, and locks in contention
  - DEADLOCKS WITH DETAILS - Detailed locking information
  - STATEMENTS - Statement start/stop time, CPU used, text of dynamic SQL
  - TRANSACTIONS - UOW work start/stop time, previous UOW time
  - CONNECTIONS - All application level counters.
  - DATABASE - All database level counters.
  - BUFFERPOOLS - Counters for buffer pool, prefetchers, page cleaners and direct I/O
  - TABLESPACES - Counters for buffer pool, prefetchers, page cleaners and direct I/O
  - TABLES - Rows read or written for each table

- Event Monitor definitions are stored in catalogs SYSCAT.EVENTMONITORS and SYSCAT.EVENTS
- To get status of an event monitor, “SELECT monitor FROM SYSCAT.EVENTMONITORS table” or use the SQL Function EVENT_MON_STATE (event-monitor-name)
- To activate or deactivate an event monitor:
  - SET EVENT MONITOR event-monitor-name STATE = 0|1
- To drop an event monitor:
  - DROP EVENT MONITOR event-monitor-name

Creating a Table Event Monitor

- A table event monitor streams event records to SQL tables, it makes capture, parse, and management of event monitoring data more easily.
- Example #1:
  CREATE EVENT MONITOR stmtmon FOR STATEMENTS WITH DETAILS WRITE TO TABLE
  The STATEMENTS event type collect data from the event_connheader, event_stmt, and event_subsection logical data groups, following event tables are created in the creator's schema:
    - user1.connheader_stmtmon
    - user1.stmt_stmtmon
    - user1.subsection_stmtmon
    - user1.control_stmtmon
  The user1.control_stmtmon table is created for every write-to-table event monitor
  The control table contains event monitor metadata containing event_start, event_db_header, and event_overflow data
- Example #2:
  CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS WRITE TO TABLE CONN,
  DLCONN (EXCLUDES(agent_id, lock_wait_start_time)),
  DLLOCK (INCLUDES(lock_mode, table_name))
  All the monitor elements for CONN are captured
  For DLCONN, all monitor elements except agent_id and lock_wait_start_time are captured
  For DLLOCK, lock_mode, table_name are the only monitor elements captured
Creating a File or Pipe Event Monitor

- Information collected by event monitors can be stored in files or named pipes

File Event Monitor
- Specify the directory where event files are to be stored.
- Streams event records to numbered files with the extension “evt” (e.g. 00000000.evt, 00000001.evt, and 00000002.evt)
- Example #1:
  
  ```sql
  CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
  WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8 BLOCKED AUTOSTART
  ```

- Example #2:
  
  ```sql
  CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
  WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8 NONBLOCKED
  MAXFILES NONE MAXFILESIZE NONE MANUALSTART
  ```

Pipe Event Monitor
- Streams event records directly from the event monitor to a named pipe.
- If the event monitor is unable to write data to the pipe (for instance, if it is full), monitor data will be lost.
- Example:
  
  ```sql
  CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
  WRITE TO PIPE '/home/riihi/dlevents' MANUALSTART
  ```

Examine Event Monitor Output

- How do I examine event monitor output?
  - Query the event monitor tables
  - Use db2evmon productivity tool
  - Use db2eva from Control Center

- Examples:
  
  ```sql
  db2evmon db <dbname> evm <evmon-name> path <evmon-path>
  db2eva db <dbname> evm <evmon-name>
  ```
Monitoring Health of DB2 Systems

- Two new features introduced in DB2 V8 to monitor the health of DB2 systems:
  - Health Monitor and Health Center
- The management by exception capability generates alert to potential system health issues.
- The Health Monitor is a server-side tool that constantly monitors the health of the instance, even without user interaction.
- An alert is raised if a defined threshold has been exceeded (e.g. the available log space is not sufficient) or an object is in an abnormal state (e.g. an instance is down).
- When an alert is raised two things can occur:
  - Alert notifications can be sent by e-mail or to a pager address
  - Preconfigured actions can be taken (e.g. a script or a task)
- The Health Monitor checks the state of your system against the health-indicator thresholds to determine when to issue an alert.
- Use the Health Center, commands, or APIs to customize the threshold settings of the health indicators, and define who should be notified and what script or task should be run if an alert is issued.

Health Alerts and Reactions

- **A. Set up “health” monitoring**
  - Determine the database system parameters that need to be monitored.
  - Determine which database objects will be monitored.
  - Identify / create snapshot or event monitors needed to collect data for the health of the database system.
  - Determine threshold values of the monitored parameters that indicate “health” problems.
  - Implement notification mechanisms and start the monitors.
- **B. React to a “health” issue**
  - Identify an issue from the performance or event monitor data.
  - Find more details and review them.
  - Determine one or more alternative actions to fix the problem (use help and manuals to do this), and how other parameters will be affected.
  - Write scripts (if necessary) to execute the actions or invoke the needed tools.
  - Verify that the problem has been resolved (by running script or checking the affected objects).
**DB2 Administration Notify Logs**

- DB2 provides diagnosis information at the point failure to the administration notification log.
- On UNIX platforms, the administration notification log is a text file called instance.nfy.
- On Windows, all administration notification messages are written to the Event Log.
- The DBM configuration parameter `notifylevel` specifies the level of information to be recorded.
- There are 5 levels of information possible:
  - 0 -- No administration notification messages captured (not recommended)
  - 1 -- Fatal or unrecoverable errors
  - 2 -- Immediate action required
  - 3 -- Important information, no immediate action required (default)
  - 4 -- Informational messages
- There is also a log file called `db2diag.log` that captures diagnostic information and is intended for DB2 customer support.
- Use the DBM configuration parameter `diaglevel` to specify the level of information to be captured.

---

**We are done!**

- Congratulations, we've covered all 125 pages of materials.
- What's next?
  - Review the class materials again
  - Review the exam objectives and study topics you don’t know well
  - Take the online assessment exams (see objective page for the link)
  - Install DB2 V8 and get some hands-on experience, it helps you to remember the concepts more easily
  - We can't cover everything here, use the DB2 manuals to help you clarify things
  - Study ...
  - Hands-on ...
  - Study ...
  - Hands-on ...
  - Take and pass the exams

**Good Luck !!!**