DB2 Blue Book

-beginners challenge-

← DB2 Fundamentals →

- Raju -

Alias Venkata Rama Rajesh mallina

Contents page

Contents

1. Introduction

2. DB2 – Data Stuctures

3. DB2 - Data objects management

4. DB2 – Data Integrity

5. SQL – Preprocess

6. DB2 – System objects

7. DB2 – Concurancy control

8 DB2 - SubSystem

9. DB2 – Routines

10. DB2 – Utiltes

11. DB2 – Commands

12. DB2-Security

13. Roles in DB2 world

14. DB2-design

15. Information management

16. DB2-SQL

1. Intoduction

1. What is Database, types of database & its applications?

An well organised & logical collection of data so that it can easily accessed and updated is called database.

Types of Database

Hierarchical database: Records are organized into a tree structure. each parent record have one or more child records. A record is a collection of fields, each field containing only one value.

Network database: In hierarchical each record having one parent record and many children, the network model allows each record to have multiple parent and child records

Relational database: This model organizes data into one or more tables (or "relations") of rows (tuple or record) and columns (attributes or field), with a unique key for each row

Object database: it has information is represented in the form of objects as used in object-oriented programming.

Applications:

- Banking: Customer info, Account number, loans and banking transactions
- Universites: studend information, course and grades
- Sales: customer product purchase information

2. What is database management system(DBMS)

DBMS is a software application which is used to create, query, update, & administration of database. Generally a database is not portable across different DBMS. but different DBMS can interoperate by using standards like Sql, JDBC,...

Ex: MySql, Oracle, Sybase, DB2

3. What is universal database(UDB)?

database which can run any OS like Linux, Unix, Windows..... is called universal database. now db2 also a UDB. So db2 refers as db2 LUW.

Where L = Linux,

U = Unix,

W = Windows.

4. What is db2 & its versions?

Amount of data increases 60% every year and 80% of worlds data processed by mainframes. We required one mainframe favorable database product which is called DB2. Db2 is universal relational database run on mainframes & introduced by IBM

Version	Code Name
3.4	Cobweb
8.1, 8.2	Stinger
9.1	Viper
9.5	Viper 2
9.7	Cobra
9.8	It added features with Only PureScale
10.1	Galileo
10.5	Kepler

These version names applicable to LUW only. for Z/os version name will be like v7 → v8 → v9 → v10 → v11

5. What is Tables

- The data in RDBMS is stored in logical database objects called tables.
- Table is a collection of related data entries and it consists of columns & rows.
- Tables are also called relations
- columns are called fields or attributes
- rows of a table are called records or tuple

ID	NAME	AGE	ADDRESS	SALARY
1	Phani	32	Banglore	2000.00
2	Sudheer	25	Banglore	1500.00
3	Harinath	23	Hyderabad	2000.00
4	Deepak	25	Chennai	6500.00
5	Summanth	27	Chennai	8500.00

- A table that holds persistent user data is a base table.
- A table that stores data temporarily is a global temporary table.

Types of attributes

- Simple attribute – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

- Composite attribute – Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- Derived attribute – Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- Single-value attribute – Single-value attributes contain single value. For example PAN Number.
- Multi-value attribute – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like –

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Database normalization: it is the process of organizing the attributes and tables of a relational database to minimize data redundancy.

ACID Model

These are properties obeys by data resides in RDMS

A → Atomicity

C → Consistency

I → Isolation

D → Durability

Atomicity: do all or none.DBMS to guarantee that either all of the tasks of a transaction are performed or none of them are.

Consistency: Any data written to the database must be valid according to rules defined. If an integrity constraint states that no salaries can exceed 75k, then any transaction violating this rule will be aborted.

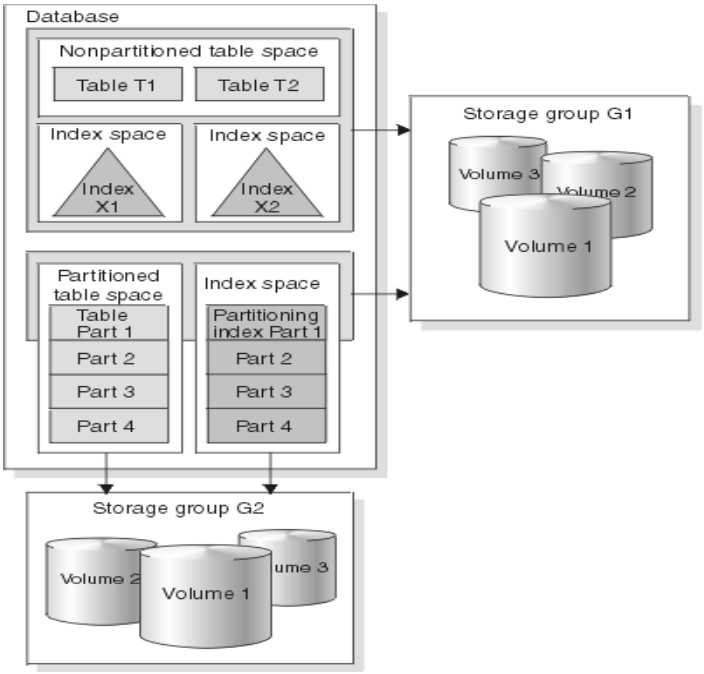
Isolation: if 2 different process accessing modifying same data at same time. We have to provide isolation between that concurrant process.

Durability: A transaction is durable if once it has been successfully completed, all of the changes it made to the system are permanent. There are safeguards that will prevent the loss of information, even in the case of system failure.

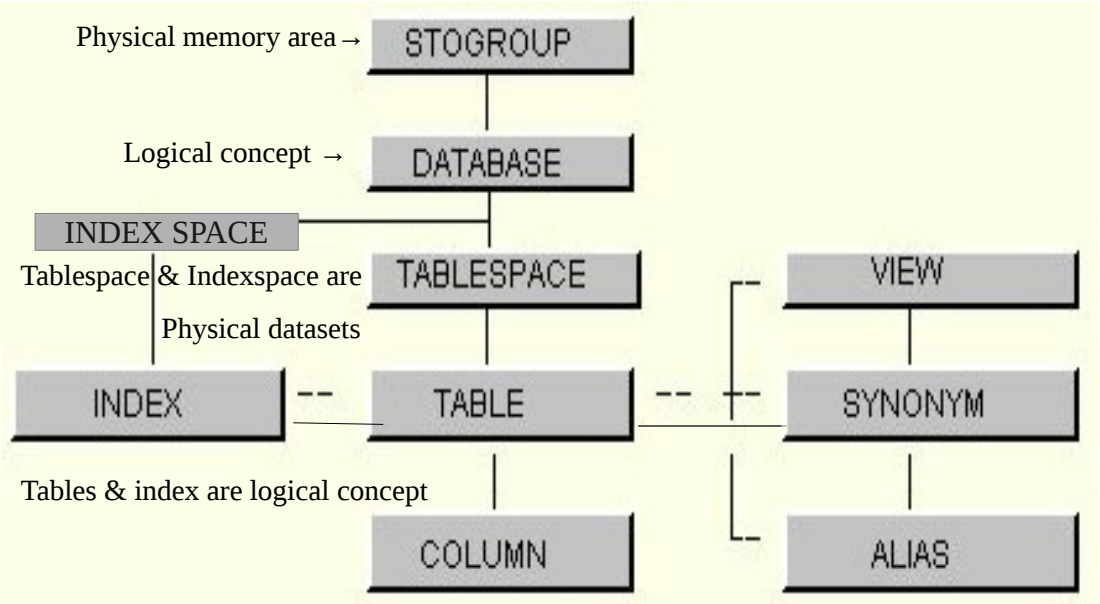
Difference between Oracle & DB2

Feature	Oracle	DB2
Memory Space on disk	100 MB	140 MB/30gb?
Name length of column, index, table & view	Max 30	Max 128
Row length	255000	32677
Mainly suited apps	Distributed apps	Mainframes
Price	$15K (SE),$40K (EE)	$7.5K (SE),$25K (EE)

2. DB2 – Data Structures

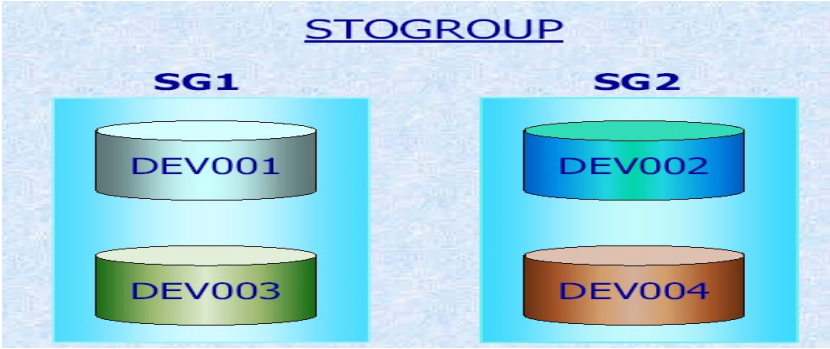


Data structures: These are accessable elements that are required to organise or store data in DB2 systems. Below diagram shows hierarchy.



2.1 Storage groups

- A set of volumes on disk that hold the table space & index space.
- Different Storage groups (STOGROUP) can point to the same volumes.
- A volume assigned to a storage group is not dedicated to it. Non-DB2 data sets can be allocated on the volume
- a volume can be assigned to more than one storage group.



- The default storage group(SYSDEFLT which was created when db2 install) is used when an automatic storage managed table space is created i.e without explicitly specifying the storage group.
- While creating storage group we will give mention HLQ for all VSAM datasets under it through VCAT(volume catlog) name
- Storage group are two types
    1. User managed storage group
    2. System/DFSMS/SMS managed Storage group

User managed storage group: here we specifies volume serial no. under storgroup. We can specify up to 59 volumes in volume clause (All volumes must be same device type However, parts of a single database can be stored in different storage groups.)

```
CREATE STOGROUP SG1
VOLUMES(VOL001,VOL002,....)
VCAT VCATID;
```

System/DFSMS/SMS managed Storage group :When defining DB2 storage groups, use the VOLUMES('*') attribute on the CREATE STOGROUP statement to let SMS (a z/OS component) control the selection of volumes during allocation as per the ACS (Automatic class selection) routines.

```
CREATE STOGROUP SG2
VOLUMES('*')
VCAT VCATID;
```

- The max number of volumes that can be managed for a storage group is 133.
- There are some optional keywords in CREATE STOGROUP syntax which can be managed by SMS ACS routines by itself (In practical there is no need to mention)

```
DATACLAS <dc-name> MGMTCLAS<mc-name> STORCLAS<sc-name>
```

- Identifies the name of the SMS data/management/storage class to associate with the DB2 storage group. The class name must be from 1-8 characters in length. The SMS storage administrator defines the class that can be used.
- Data class is a template of data set attributes (such as record length and format) that you can specify is to be used when creating a new data set.
- Management class represents different categories of management activities. For example, a **MGMTCLAS=MGMT01** could mean all datasets which are backed up everyday, and retained upto 1 month.
- Storage class represents different categories of devices and volumes. For example, a **STORCLAS=CLASSOA** could be representing all volumes on 3380 DASD Device. Performance depend on storageclass (wanted response time)
- These are 3 ACS routines which will run at data set allocation. The stuff in each class used to manage dataset.

- By ALTER STOGROUP we can add or remove allocated volumes for Non-SMS Storage group or we can change name of storage/management/data class
- Removing a volume from a storage group does not affect existing data even though you write remove clause first then add clause second.

```
ALTER STOGROUP SG2
ADD VOLUMES (VOL007, VOL008)
REMOVE VOLUMES(VOL001);
```

Catlog tables for storagegroup

- SYSIBM.SYSSTOGROUP has a row for each storage group
- SYSIBM.SYSVOLUMES has a row for each volume.

```
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT NAME||', '||CREATOR||', '||VCATNAME FROM SYSIBM.SYSSTOGROUP;     00000100
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
SYSDEFLT, SYSPROG, DBDSN9
DSNOPTSG, SYSPROG, DBDSN9
DSN8G110, SYSPROG, DBDSN9
DSN8G11U, SYSPROG, DBDSN9
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
```

```
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT SGNAME||', '||SGCREATOR||', '||VOLID                          00000100
FROM SYSIBM.SYSVOLUMES;                                              00000200
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
SYSDEFLT, SYSPROG, *
DSNOPTSG, SYSPROG, *
DSN8G110, SYSPROG, *
DSN8G11U, SYSPROG, *
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

drop storage group

db2 drop stogorup <stogroup_name>

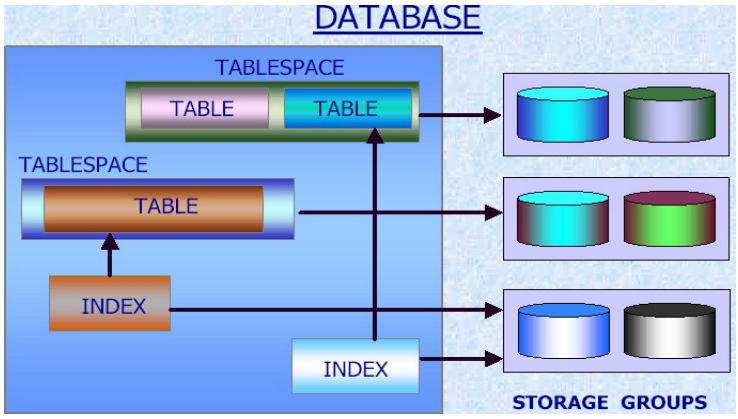
Difference between Data Structures & Storage Structures

1. Data structures are elements that are required to use DB2. You can access & use these elements to organize your data. EX: tables, table spaces, indexes, index spaces, keys, views, and databases.
2. a storage structure are data stuctures which is a set of one or more VSAM datasets that hold DB2 tables or indexes. A storage structure is also called a page set. Ex: Table space, Index space Hash Space

1. What is storage group? Is storage group's volumes can be sharable?
2. Default storage group? When it will create
3. Types of storage group?
4. Max,How many volumes can specify in volume clause of CREATE STOGROUP
5. Maximum how many volumes can be manged under storage group
6. What is VCAT
7. What role ACS play in System manged storage group
8. What ALTER STOGROUP can do?

2.2 Databases

- Database is a logical Collection of table spaces and index spaces
- So in practical Database is a qualifier for physical storage data structures (table space and index space)



- If we create table space without IN clause i.e without specifying database name then default database is DSNDB04.
- Creating Database is nothing but creating qualifier. It is not dedicated to particular storage group, it can use in another storage group also.
- Database is logical in Z/OS but it is physical in LUW.

Create Database

```
>>-CREATE DATABASE--database-name------------------------------>

   .-----------------------------------------.
   V  (1)                                    |
>--------+-----------------------------------+-+--------------><
         +-BUFFERPOOL--bpname----------------+
         +-INDEXBP--bpname-------------------+
         +-AS WORKFILE--+-----------------+-+
         |              '-FOR--member-name-' |
         |           .-SYSDEFLT------.       |
         +-STOGROUP--+-stogroup-name-+-------+
         '-CCSID--+-ASCII---+---------------'
                  +-EBCDIC--+
                  '-UNICODE-'
```

- All the parameters in below example can be Alter.

```
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE DATABASE WWV8310D                                                 00001020
STOGROUP WWV8310S                                                        00002020
BUFFERPOOL BP1                                                           00003020
INDEXBP BP2;                                                             00004020
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
```

Why Database?

We can active or de-active database. So that we control set of all db2 objects under database at a time. In addition to it,We can display status of all objects under database at a time

- Coressponding catlog table: SYSIBM.SYSDATABASE

```
---------+---------+---------+---------+---------+---------+---------+--
SELECT NAME||'   '||CREATOR||'   '||STGROUP||'   '||BPOOL||' '||DBID
FROM SYSIBM.SYSDATABASE;
---------+---------+---------+---------+---------+---------+---------+--

---------+---------+---------+---------+---------+---------+---------+--
DSN00001   SYSIBM   SYSDEFLT   BP0      256
DSN00002   SYSIBM   SYSDEFLT   BP0      257
DSN00003   SYSIBM   SYSDEFLT   BP0      258
DSNDB01   SYSIBM                1
DSNDB04    SYSIBM     SYSDEFLT   BP0      4
DSNDB06    SYSIBM                         6
DSNDB07   SYSPROG   SYSDEFLT   BP0      7
DSNRLST   SYSPROG   SYSDEFLT   BP0      259
DSNRGFDB   SYSPROG   SYSDEFLT   BP0      260
DSNOPTDB   SYSPROG   DSNOPTSG   BP0      261
DSNMDCDB   SYSPROG   SYSDEFLT   BP0      262
DSNADMDB   SYSPROG   SYSDEFLT   BP0      263
DSNATPDB   SYSPROG   SYSDEFLT   BP0      264
```

What Alter Database can do?

```
>>-ALTER DATABASE--database-name-------------------------------->

   .-----------------------------------.
   V  (1)                              |
>----------+-BUFFERPOOL--bpname------+-+----------------------><
           +-INDEXBP--bpname---------+
           +-STOGROUP--stogroup-name-+
           '-CCSID--ccsid-value------'
```

Drop Database

DROP DATABASE <*database-name*>

1. What is default database
2. What is database
4. IS single database can shared by different storage groups
5. how concept of database differ in zos from LUW
6. What is purpose of database
7. What Alter Database can do

2.3 Table Spaces

- Table spaces is a DB2 storage stucture which consist of set of VSAM Linear Data Sets which are divided into equal units, called pages to store tables.
- Creating tablespace means creating VSAM LDS in underlaying.
- Page size may 4 or 8 or 16 or 32 KB. By default 4kb
- Each page contains max 255 records. For LUW it is 127 records

WHY LINEAR VSAM for TS

→ VSAM Datasets has random search, where as Non-Vsam has sequential serach

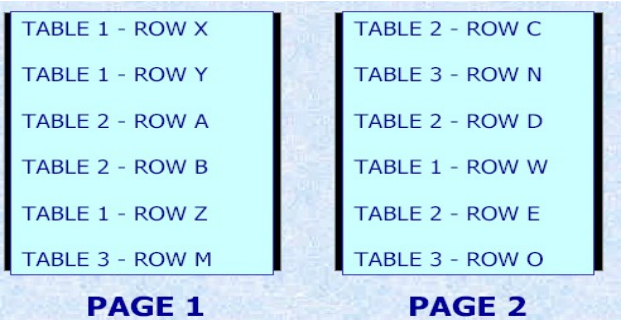
→ Linear VSAM is one and only unstructured VSAM. It does't contain RDF. If it contain RDF it while processing RDF treated as data

Types of Table Spaces

1. Simple Table space
2. Segmented Table space
3. Partition Table space
4. Universal Table space
    - Partition by range (PBR)
    - Partition by growth (PBG)
5. Auxiliary or LOB Table space
6. XML Table space

Simple table space :

- The rows of different tables are not kept separate. rows from multiple tables in it to reside on same DB2 page
- Simple table spaces can no longer be created from DB2 9. However, simple table spaces created in DB2 8. but earlier will continue to be supported.



Advantages

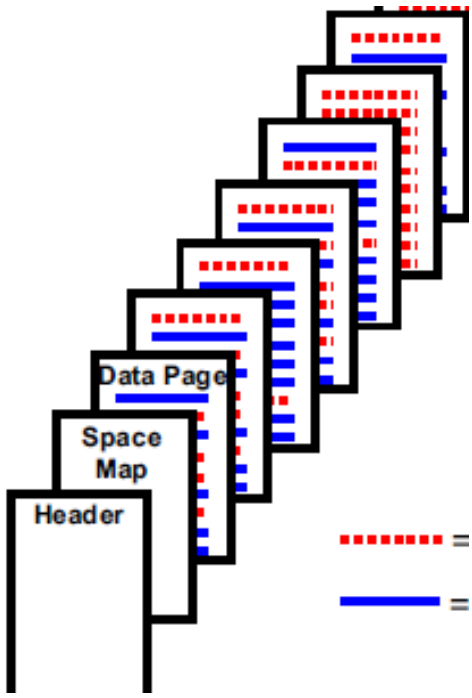
- You can add as many tables you can

Disadvantages

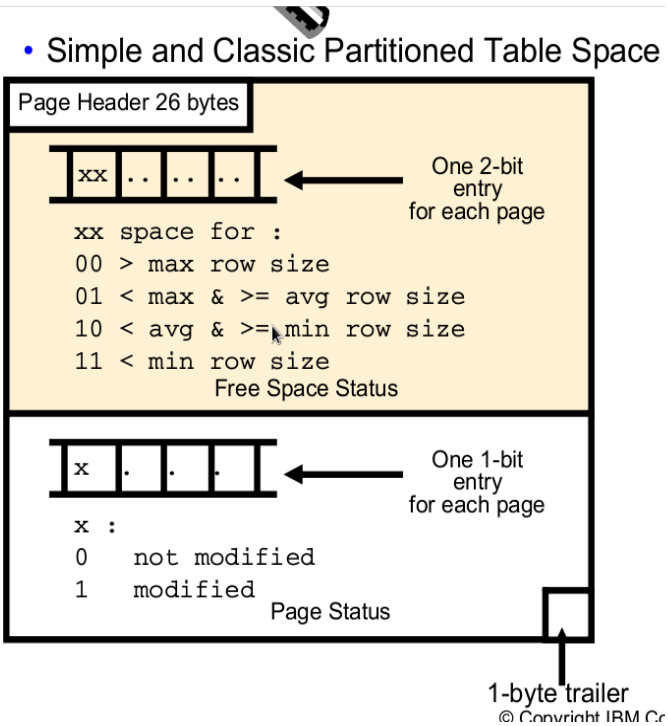
- As rows are mixed in page, it takes more time to scan rows pertaining to a table.
- Even though a table is dropped or rows from a table is deleted, space could not be claimed until and unless REORG is performed on TS
- There is no lock at the table level.

Note:

Header page: it is first page of table space (or partition). it contains PSID (page set ID which is unique for tablespace), DBID(database ID unique for database). If a table is created under it then there is OBID (object id unique for the table)....the information required to recover data.



SPACE MAP: this is the 2nd page.

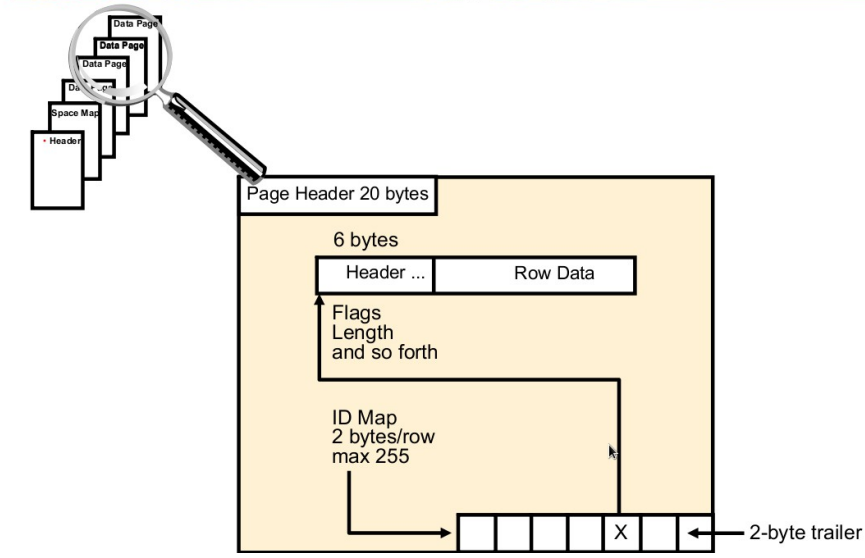


It contains following info

- Free space status:
  - DB2 uses two bits for each page to indicate whether space is available on a particular page for the largest (00), average (01), or smallest row (10), or whether the page is full (11).
  - This status is used during Data insertion & Data Modification in Tablespace.

- Page status:
  - DB2 uses one bit for each page to indicate whether page is modified (1) or not (0)
  - The space map pages also tell the COPY utility which pages have been changed since the last copy. This information is used to determine which pages should be copied when making an incremental copy.
- These statuses are required during reorgnization (Applying Reorg)

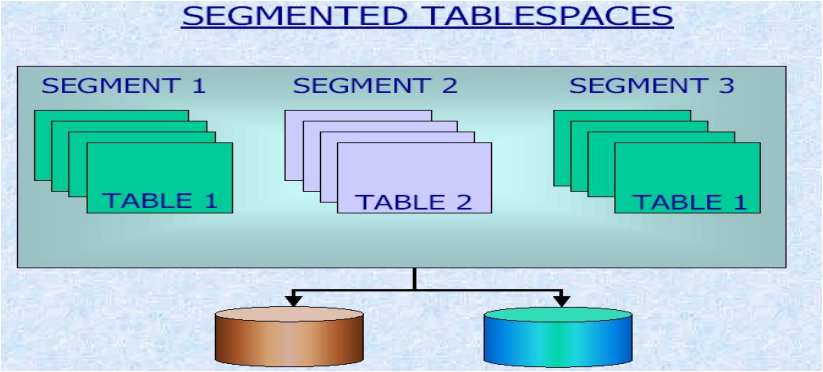
Data Page: Remaining pages holds Data records and they refers as Data pages



- The first data page has page number 2 in the absence of other pages (for example, dictionary pages, see later).
- Data pages have the same layout in all types of table space .
- 6 Byte header for row data which contain OBID, row type, row length...

Segmented table space :

1. Divide the available space into groups of pages generally 4 to 64 pages called segments.
2. Each segment is the same size. A segment contains rows from only one table. As shown in below example one table may occupy more than one segment.
3. SEGSIZE is defined as an integer representing the number of pages to be assigned to a segment. The SEGSIZE of a can be any multiple of 4, from 4 to 64



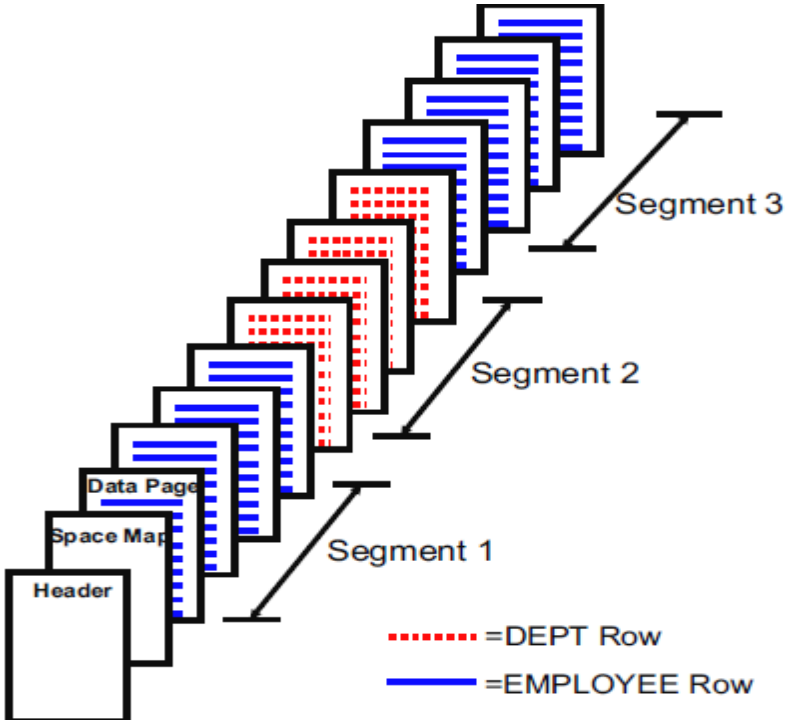

```
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TABLESPACE TSLOCN                                                 00010019
IN WWV8210D                                                              00020025
USING STOGROUP WWV8310S                                                  00030017
PRIQTY 720                                                               00050006
SECQTY 720                                                               00060021
SEGSIZE 64                                                               00061009
LOCKSIZE ANY                                                             00070018
MAXROWS 255;                                                             00080018
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
```

Advantages:

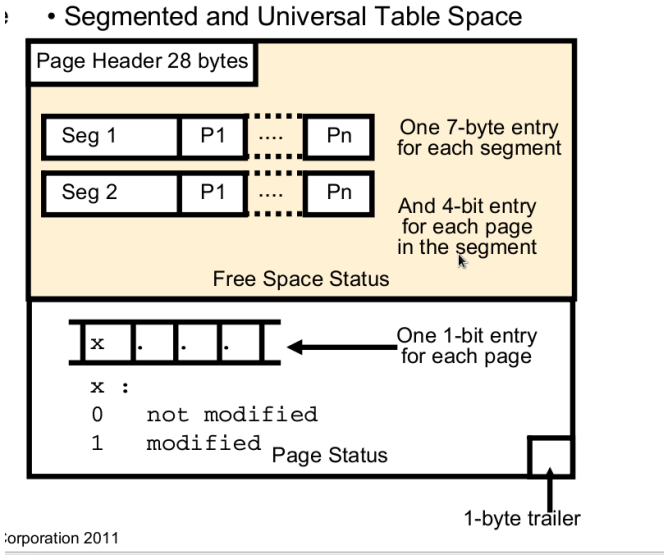
- A REORG is not necessary to reclaim space made by a dropped table. However, a REORG is necessary to reclaim space from a mass delete.
- Provides proper isolation between tables through segments which make flexible to system for data identification
- Table level locking is possible.
- When you take a full image copy, the COPY utility does not copy empty pages , since the space map is used to avoid scanning empty segments .

Disadvantages:

- Maximum memory per table space is 64 GB only (main drawback)
- All segments are same size. Each segment has records of one table only, even records are very few. This inefficient use of memory . (as I observed)



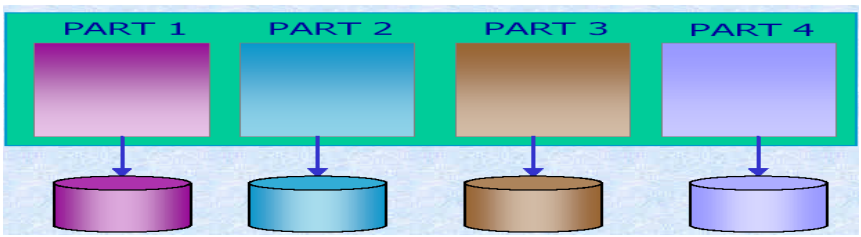
Entire table space contain only one header page and space map page just like simple tablespace. The space map page layout given below.



- One 7-byte entry for each segment & 4-bit entry for each page in the segment as shown in below.
    - 0 - Page is empty and unformatted
    - 1 - Page is emptied by mass delete
    - 2 - Page is empty and formatted
    - 3 - Page has space for longest row
    - 4-10 - Values used for placing variable length rows
    - 11-14 - For future use
    - 15 - Page is full
- When you take a full image copy, the COPY utility does not copy empty pages , since the space map is used to avoid scanning empty segments .

Partitioned table space :

- Stores one table
- Divide the available space into separate units of storage called partitions.
- Each partition contains one VSAM linear data set. And max partitions 4096.




```
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TABLESPACE TSEMPL                                                 00010029
IN WWV8210D                                                              00020025
USING STOGROUP WWV8310S                                                  00030017
PRIQTY 720                                                               00050006
SECQTY 720                                                               00060021
SEGSIZE 0                                                                00061026
PCTFREE 10                                                               00070026
FREEPAGE 5                                                               00071026
NUMPARTS 10                                                              00072026
(PARTITION 1 USING STOGROUP WWV8310S                                     00073028
             PRIQTY 200                                                  00080027
             SECQTY 100                                                  00090026
             ERASE YES                                                   00100026
             COMPRESS YES)                                               00110026
BUFFERPOOL BP2;                                                          00120026
---------+---------+---------+---------+---------+---------+---------+---------+
```

- NUMPARTS: number of partitions in TS
- SEGSIZE: make as Zero
- DSSIZE: Max size of each partition or dataset. In practical ther is no need to DSSIZE. B2 chooses a default DSSIZE governed by page size & NUMPARTS as shown in below tables . (Range 1 GB to 64 GB)

NUMPARTS < 255

NUMPARTS	MAX PARTITION SIZE	MAX TS SIZE
1 to 16	4GB	64GB
17 to 32	2GB	64GB
33 to 64	1GB	64GB
64 t0 254	4GB	1TB (Approx.)

NUMPARTS >= 255

DS SIZE	Page size				Max Memory (Based on DSSIZE)
	4k	8k	16k	32k	
1-4 GB	4096	4096	4096	4096	4TB-16 TB
8 GB	2048	4096	4096	4096	32 TB
16 GB	1024	2048	4096	4096	64 TB
32 GB	512	1024	2048	4096	128 TB
64 GB	256	512	1024	2048	128 TB

- Above table also gives info how many partition can be create if we specify DSSIZE.

Advantages

- Partitions can be processed in parallel.
- Store large table data more than 64 GB.

Disadvantages

- to recover the free space we have to apply REORG.

Universal table space :

- DB2 9 introduced a new type of table space, called the universal table space (UTS), which combines the advantage of segmented and partitioned table spaces.
- Through partitioning feature Tablespace can grow beyond 64 GB. Through Segmenting feature is nothing but mass delete concept. We will get both advantages in Universal tablespace.
- A universal table space can hold one table. Types
  - Partition by Range
  - Partition by growth

Partition by Range

- You can implement range-partitioned universal table spaces by specifying the NUMPARTS keyword, or both keywords SEGSIZE and NUMPARTS on a CREATE TABLESPACE statement.

Note:

- In below example, If SEGSIZE is not mentioned upto DB2 9 versions, it will creates Partition table space.
- From DB2 10, even though if we not mentioned SEGSIZE It will take value of DSNZPARM SEGSIZE and creates PBR. If it is set to 0, a classic partitioned table space is created.

```
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TABLESPACE TSEMPLR                                                 00010030
IN WWV8210D                                                               00020025
USING STOGROUP WWV8310S                                                   00030017
PRIQTY 720                                                                00050006
SECQTY 720                                                                00060021
SEGSIZE 32                                                                00061030
PCTFREE 10                                                                00070026
FREEPAGE 5                                                                00071026
NUMPARTS 10                                                               00072026
(PARTITION 1 USING STOGROUP WWV8310S                                      00073028
             PRIQTY 200                                                   00080027
             SECQTY 100                                                   00090026
             ERASE YES                                                    00100026
             COMPRESS YES)                                                00110026
BUFFERPOOL BP2;                                                           00120026
---------+---------+---------+---------+---------+---------+---------+---------+
```

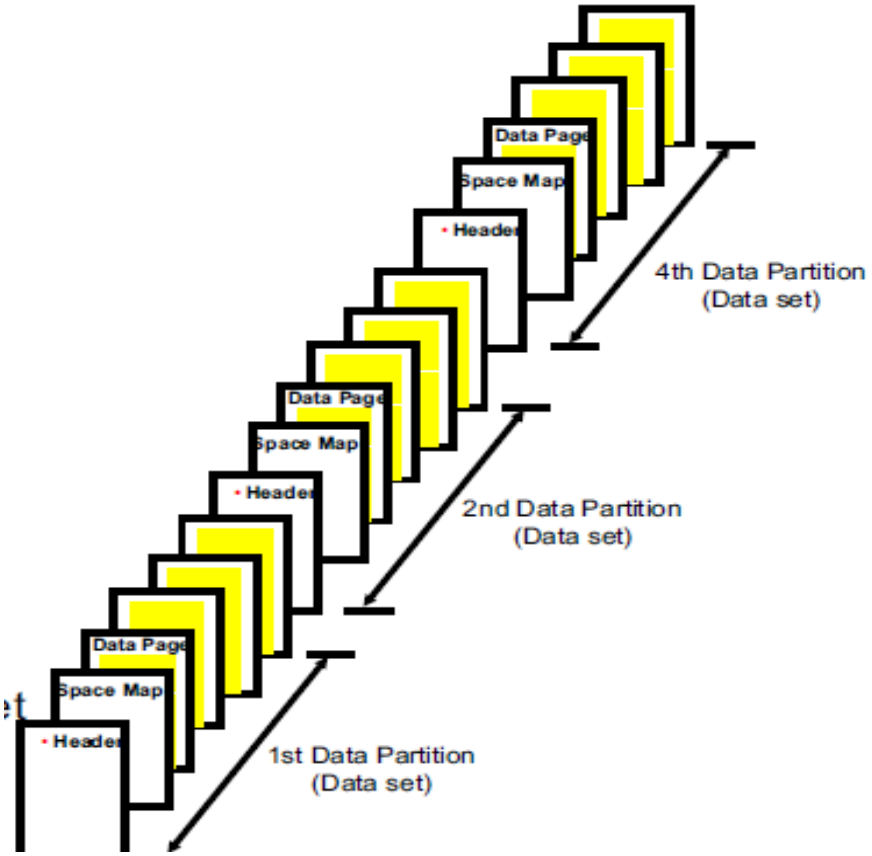
Partition by growth

- No partitioning key is needed in partition by growth.
- If partition is full then automatically 2 partition will be populated.
- Maximum how many partitions can be created is decided by MAXPARTITIONS.
- Created using MAXPARTITIONS clause on CREATE TABLESPACE
- MAXPARTITIONS, DSIZE, PAGESIZE must be compatable.
- The default SEGSIZE value 32 (Ref: IBM Knowlege Center)

```
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TABLESPACE TSEMPLG                                                 00010032
IN WWV8210D                                                               00020025
USING STOGROUP WWV8310S                                                   00030017
PRIQTY 720                                                                00050006
SECQTY 720                                                                00060021
SEGSIZE 32                                                                00061030
PCTFREE 10                                                                00070026
FREEPAGE 5                                                                00071026
MAXPARTITIONS 10                                                          00072031
BUFFERPOOL BP2;                                                           00120026
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
```

Cons:

1. Once created partitions cannot be deleted if not needed anymore (*)
2. Using ALTER TABLESPACE, you can increase, but not decrease the MAXPARTITIONS value.
3. Only NPIs allowed (*)
4. Loading partition is not possible.

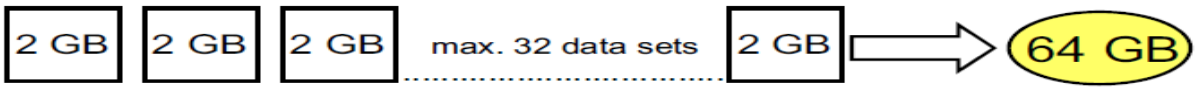


Each Partion of Partition or Universal Tablespace contain Header page, Space map page.

Why 64 GB only

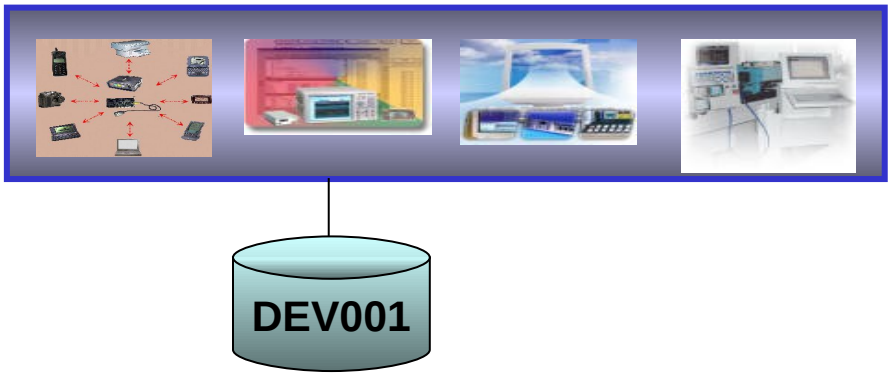
If segmented tablespace is created, then in underlaying One dataset will created which can extends upto 2 GB. If one dataset reaches its Max memory level 2 GB, than one more VSAM will create this process can go on upto 32 times. So max memory for segmented is 2*32=64GB

For of parition of Classic Partition or universal Table space has one Undering dataset which itself can grow upto 64GB



Auxiliary/LOB table space:

Holds very large object data such as graphics, video or very large text strings.



SEGSIZE	MAXPARTITIONS	NUMPARTS	TS
Specified	Not Specified	Not Specified	Segmented
Not Specified	Specified	Not Specified	PBG (ZPARM SEGSIZE)
Not Specified	Not Specified	Specified	Partition
Specified	Specified	Not Specified	PBG
Specified	Not Specified	Specified	PBR
Not Specified	Not Specified	Not Specified	Segmented (ZPARM SEGSIZE)

Some of the clauses used during Table space creation

1. ERASE

- When we perform Delete or Drop operation, ERASE YES, As a security measure , wheather the coressponding data inunderlaying datasets are erased.
- ERASE NO,Does't erase in the data sets.However, the data is still accessible, though not through DB2. This is the default.

2. FREEPAGE integer

- Specifies how often to leave a page of free space in the table space or partition.when it is loaded or reorganized.
- The default is FREEPAGE 0, You must specify an integer in the range 0 to 255.
- If the table space is segmented, FREEPAGE must be less than SEGSIZE other wise FREEPAGE takes value as SEGSIZE-1.
- These free space is used when we apply REORG

3. PCTFREE smallint

- Indicates what percentage of each page to leave as free space when the table is loaded or reorganized.
- smallint in the range from 0 to 99. The default is PCTFREE 5

4. DEFINE

- Specifies when the underlying data sets for the table space are physically created.
- YES, The data sets are created whixh is default. If MAXPARTITIONS is also specified, only the first partition is created when the table space is created
- NO ,The data sets are not created until data is inserted into the table space.
- DB2 uses the SPACE column in catalog table SYSTABLEPART to record the status of the data sets (undefined or allocated).
- Do not specify DEFINE NO for a table space in a work file database. DEFINE NO is not recommended if you intend to use any tools outside of DB2 to manipulate data, such as to load data, DB2 may encounters this inconsistent state.

5. LOGGED or NOT LOGGED

Specifies whether changes that are made to the data in the specified table space are recorded in the log. By default LOGGED

6. TRACKMOD

- YES, DB2 tracks changed pages in the space map pages to improve the performance of incremental image copy, which is default

- NO,DB2 does not track changed pages in the space map pages. It uses the LRSN value in each page to determine whether a page has been changed.

7. CLOSE

- When the limit on the number of open data sets is reached, specifies the priority in which data sets are closed.

- CLOSE YES Eligible for closing before CLOSE NO data sets.
- YES is the default unless the table space is in a work file database.
- DB2 uses CLOSE NO regardless of the value specified for a table space in a work file database

8. MAXROWS integer

- Specifies the maximum number of rows that DB2 will consider placing on each data page. The integer can range from 1 through 255 & 255 is default.
- Do not use MAXROWS for a LOB table space or a table space in a work file database.

9. CCSID encoding-scheme

- Coded Character Set Identifier, Specifies the encoding scheme for tables stored in the table space.
- If you do not specify a CCSID when it is allowed, the default is the encoding scheme of the database in which the table space resides, except for table spaces in DSNDB04.
- Options for CCSID
    1. ASCII,American Standard Code for Information Interchange
    2. EBCDIC , IBM's Extended Binary Coded Decimal Interchange Code
    3. UNICODE
- All data stored within a table space must use the same encoding scheme unless the table space is in a work file database.
- Do not specify CCSID for a LOB table space or a table space in a work file database.
- The encoding scheme for a LOB table space is inherited from the base table space. A table space in a work file database does not have an associated encoding scheme because the table space can contain created and declared temporary tables with a mixture of encoding schemes.

10. MEMBER CLUSTER

- Specifies that data inserted by an insert operation is not clustered by the implicit clustering index (the first index) or the explicit clustering index. Instead, DB2 chooses where to locate the data in the table space based on available space.
- Do not specify MEMBER CLUSTER for segmented table spaces or LOB table space or a table space in a work file database.
- MEMBER CLUSTER can be specified with MAXPARTITIONS.

Catlog tables :

- SYSIBM.SYSTABLESPACE
- SYSIBM.SYSTABLEPART

```
---------+---------+---------+---------+---------+---------+---------+
SELECT NAME||' '||CREATOR||' '||DBNAME||' '||OBID||' '||PSID||' '||
BPOOL||' '||PARTITIONS||' '||SEGSIZE||' '||MAXPARTITIONS||' '||TYPE
FROM SYSIBM.SYSTABLESPACE;
---------+---------+---------+---------+---------+---------+---------+


---------+---------+---------+---------+---------+---------+---------+
DBD01 SYSIBM DSNDB01 16 31 BP0      1 4 1 G
SCT02 SYSIBM DSNDB01 96 95 BP0      0 0 0
SPT01 SYSIBM DSNDB01 128 127 BP32K    1 32 1 G
SYSDBDXA SYSIBM DSNDB01 2 1 BP0      0 0 0 0
SYSLGRNX SYSIBM DSNDB01 192 207 BP0      1 32 1 G
SYSSPUXA SYSIBM DSNDB01 10 9 BP0      0 0 0 0
SYSSPUXB SYSIBM DSNDB01 18 17 BP0      0 0 0 0
SYSUTILX SYSIBM DSNDB01 160 175 BP32K    0 4 0
```

```
---------+---------+---------+---------+---------+---------+---------+-
SELECT PARTITION||' '||TSNAME,DBNAME,PERCACTIVE||' '||PERCDROP
FROM SYSIBM.SYSTABLEPART;
---------+---------+---------+---------+---------+---------+---------+-
                                DBNAME
---------+---------+---------+---------+---------+---------+---------+-
1 DBD01                         DSNDB01                  -1 -1
0 SCT02                         DSNDB01                  -1 -1
1 SPT01                         DSNDB01                  -1 -1
0 SYSDBDXA                      DSNDB01                  -1 -1
1 SYSLGRNX                      DSNDB01                  -1 -1
0 SYSSPUXA                      DSNDB01                  -1 -1
0 SYSSPUXB                      DSNDB01                  -1 -1
0 SYSUTILX                      DSNDB01                  -1 -1
0 SYSALTER                      DSNDB06                  -1 -1
0 SYSCONTX                      DSNDB06                  -1 -1
0 SYSDDF                        DSNDB06                  -1 -1
0 SYSEBCDC                      DSNDB06                  -1 -1
0 SYSGPAUT                      DSNDB06                  -1 -1
```

1 What is table space

2 Types of table space

3 How simple table space will create

4 Merits and Demerits of Simple TS

5 How Segmented table space will create

6 Merits and Demerits of Segmented TS

7 How Partition table space will create

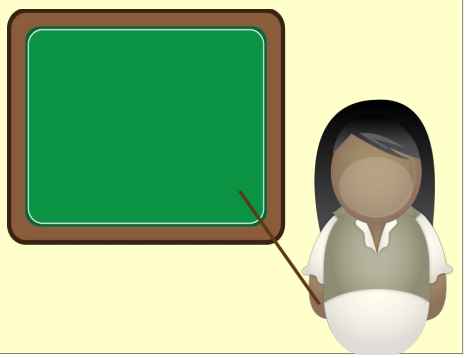
8 Merits and Demerits of Partition TS

9 How Partition by range TS will create

10 Merits and Demerits of PBR TS

11 How Partition by growth TS will create

12 Merits and Demerits of PBG TS



2.4 Tables

- Every table in DB2 must be given a unique name. A table always has two parts, separated by a period . a schema/owner, followed by the table name.
- Optionally, if there are multiple DB2 systems connected together, it may become necessary to qualify the table with a location name as well. When that is the case, the location name comes first.
- The location name must begin with a letter and cannot contain special characters. Acceptable characters are letters A through Z, numbers 0 through 9, and underscores.
- By default location name is LOC1
- Each column in a table must be unique .

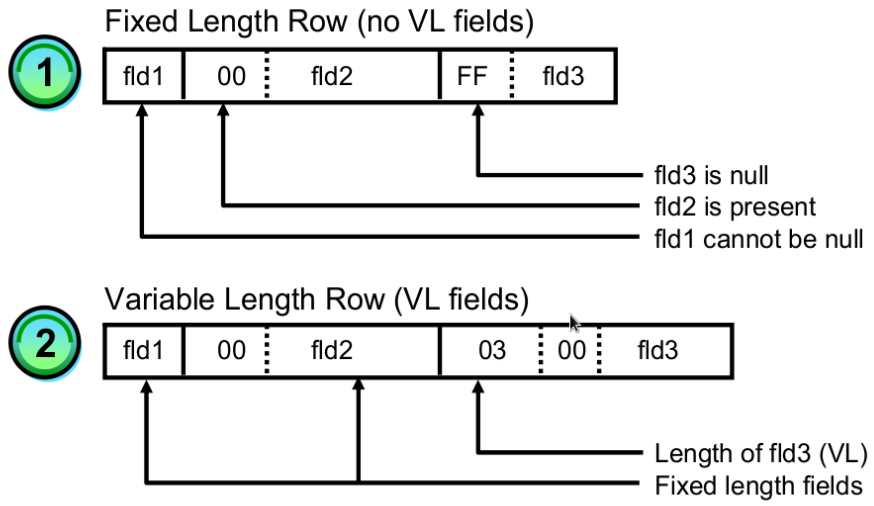
Syntax: CREATE clause
(column description area)
IN clause

```
                        Datatype      Null Attribute    Default Attribute

CREATE TABLE EMP
 (EMPNO                 CHAR(6)       NOT NULL                          ,
  FIRSTNME              VARCHAR(30)   NOT NULL                          ,
  MIDINIT               CHAR(1)                                         ,
  LASTNAME              VARCHAR(30)   NOT NULL                          ,
  WORKDEPT              CHAR(3)                                         ,
  PHONENO               CHAR(4)                                         ,
  HIREDATE              DATE                                            ,
  JOB                   CHAR(8)       WITH DEFAULT 'NEW HIRE'           ,
  EDLEVEL               SMALLINT                                        ,
  SEX                   CHAR(1)       NOT NULL                          ,
  BIRTHDATE             DATE          NOT NULL                          ,
  SALARY                DECIMAL(9,2)                                    ,
  RESUME                CLOB(10M)                                       ,
  INFO                  XML) ;
[ or IN DBX.TSX ;]
[ or IN DATABASE DBX ;]
[ or IN TSX ;]
```

Row Format

- Nullable fields have a flag to indicate whether the field is present (X'00') or null (X'



- ALTER TABLE ... ADD columnname does not update the existing records

we can assign default values for column

- Default can be either:
  - A constant
  - USER (special register)
  - CURRENT SQLID (special register)
  - NULL
  - System defaults

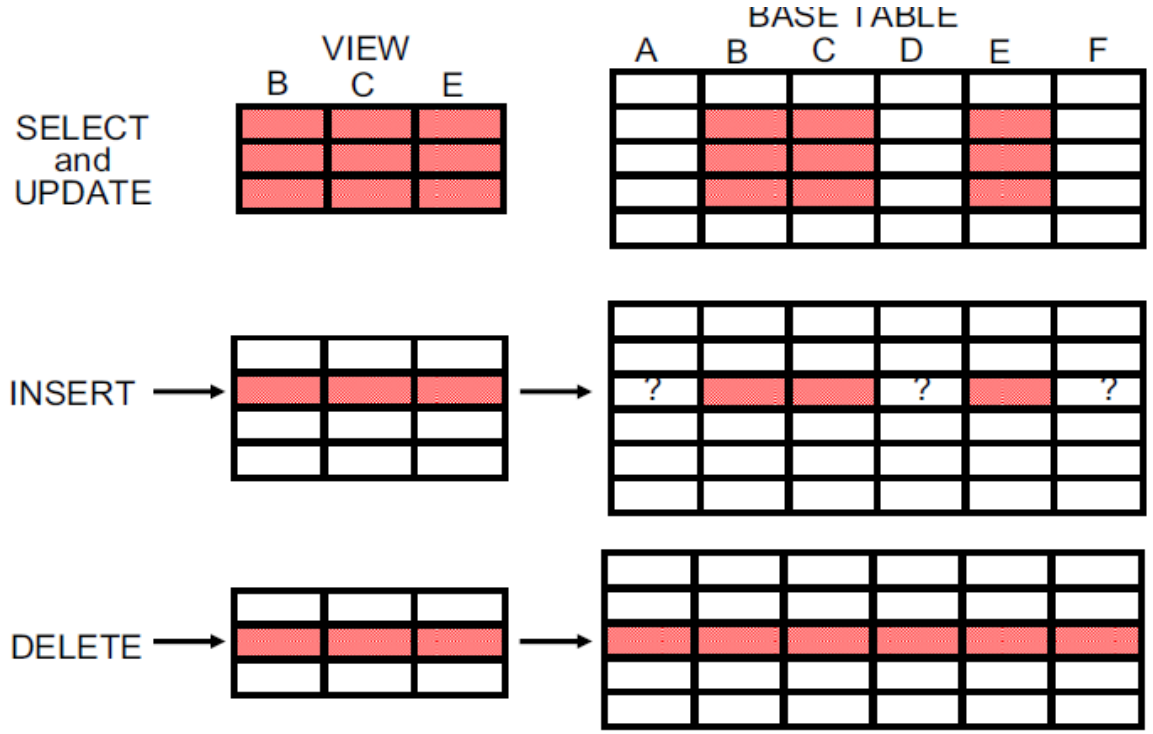
- Examples:

```
WITH DEFAULT 'MY OWN VALUE'
WITH DEFAULT USER
WITH DEFAULT CURRENT SQLID
WITH DEFAULT NULL
WITH DEFAULT
```

2.5 Views-MQT

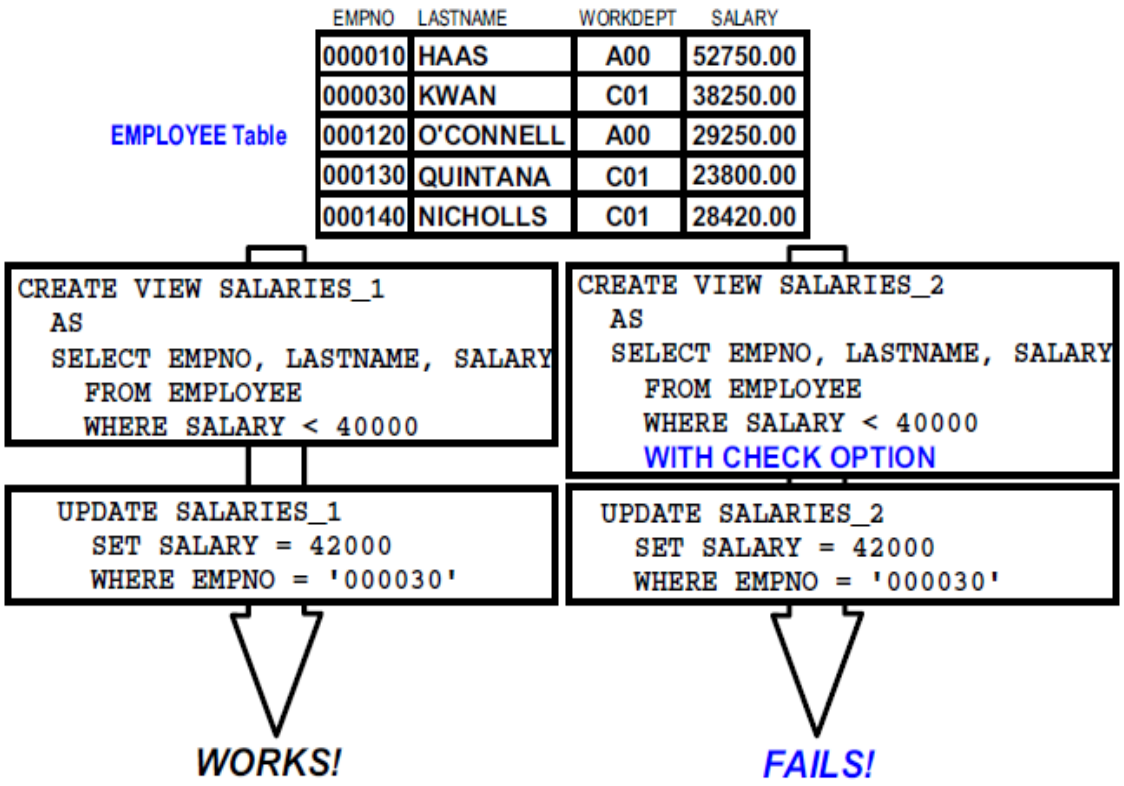
View

- A view is an alternative way of representing data that exists in one or more tables. View provides a way of looking at the data in one or more tables. It is a virtual table.
- Def: View is a virtual table which has subset of rows and columns from one or more tables.
- Restriction: You cannot create an index for a view. In addition, you cannot create any form of a key or a constraint (referential or otherwise) on a view.
- You can create a view on views as long as the view references no more than 15 base tables directly or indirectly (I mean the context when we apply view on view and real table combined).



- UPDATE: The change of values in the view changes the values in the corresponding columns the base table. However, base table columns which are not part of the view are not changed.

- INSERT: Inserting a row into a view is only possible, if all columns of the base table, which are not part of the view, are nullable or have defaults.
- DELETE: The deletion of a row from the view deletes the entire row from the base table even view does't contain all columns of base table.



Disapperaing ROW : We can insert or update data in view even that voilates VIEW defination. Then that row will be in table but diaappear in VIEW

With Local Check option: We can insert or update data in view. It must follow VIEW defination.

Casacade checkoption effect

EMPLOYEE

EMPNO	NAME	DEPT	ROOM	TELEPHONE	SALARY
110	LIEBHERR	A10	1018	4388	7500
220	ABELE	E10	1003	4407	4900
290	OBERHAUS	E11	1012	4112	5500
300	SCHMIDT	E11	1034	4234	4300
310	MUELLER	E11	1022	4419	4100

1.
```
CREATE VIEW V1
AS
SELECT EMPNO, NAME, SALARY
FROM   EMPLOYEE
WHERE  SALARY < 5000
WITH CHECK OPTION;
```

3.
```
CREATE VIEW V2
AS
SELECT EMPNO, NAME, SALARY
FROM   V1
WHERE  SALARY > 4200;
```

V1

EMPNO	NAME	SALARY
220	ABELE	4900
300	SCHMIDT	4300
310	MUELLER	4100

V2

EMPNO	NAME	SALARY
220	ABELE	4900
300	SCHMIDT	4300

2.
```
UPDATE V1
SET SALARY = SALARY + 200;
```

FAILS!

4.
```
UPDATE V2
SET SALARY = SALARY + 200;
```

FAILS!

- In this example, view V1 is created on table EMPLOYEE using WITH CHECK OPTION. Notice the WHERE clause that allows only rows where SALARY < 5000.
- Updating the table through view V1 by increasing the SALARY by 200 would fail because SALARY for EMPNO 220 would go over 5000.
- View V2 is created on view V1 without the use of WITH CHECK OPTION. Notice the WHERE clause that allows only rows where SALARY > 4200 to be retrieved. Updating the table through view V2 by increasing the SALARY by 200 would fail because SALARY for EMPNO 220 would go over 5000.
- Although WITH CHECK OPTION is not used when creating view V2, since view V1 is created using WITH CHECK OPTION, the search condition of view V1 is inherited by view V2.

CATLOG Tables for views

- SYSIBM.SYSVIEWS: view definitions are stored as rows
- SYSIBM.SYSVIEWDEPT: view dependencies are stored.
- SYSIBM.SYSTABLES: Rows having TYPE ‘V’ indicating this row is for the view
- SYSIBM.SYSCOLUMNS: a row is inserted for each column in the view definition.

MQT (Materialized query tables)

A Materialized Query Table (MQT) is a table whose definition is based on the result of a query ,and whose data is in the form of pre-computed results that are taken from one or more tables

```
EX:  CREATE TABLE MQT AS ( SELECT D.DEPTNO, D.DEPTNAME,
                           SUM(E.SALARY) AS TOTAL_SALARY,
                           SUM(E.BONUS) AS TOTAL_BONUS,
                           SUM(E.COMMISSION) AS
TOTAL_COMMISSION
                           FROM EMPLOYEE E, DEPARTMENT D
                           WHERE E.DEPTNO = D.DEPTNO
                           GROUP BY D.DEPTNO, D.DEPTNAME
                           ORDER BY D.DEPTNO)
     MAINTAINED BY SYSTEM
     ENABLE QUERY OPTIMIZATION
     IN DBX.TSX;
```

- Note: you can't unique create index on MQT. You can create index

REFRESH TABLE SQL statement

- All rows are deleted from the MQT.
- The MQTs Subselect is executed to recalculate the data from the tables
- Access to the MQT itself
- The catalog is updated.

DATA INITIALLY DEFERRED

This means the data is not inserted into the materialized query table when it is created. Use the REFRESH TABLE statement to populate the materialized query table, or use INSERT, LOAD...

REFRESH DEFERRED

This means the data in the table is not refreshed on update of the base table (except by use of triggers), but can be refreshed at any time using the REFRESH TABLE .

MAINTAINED BY SYSTEM

Only the REFRESH TABLE SQL statement is allowed on the table. This is the default.

MAINTAINED BY USER

This means the materialized query table is maintained by the user, who can use the LOAD utility, an SQL data change statement, a SELECT from data change statement, or REFRESH TABLE SQL statements on the table.

ENABLE QUERY OPTIMIZATION

This means the materialized query table can be used for query optimization. If the Subselect specified does not satisfy the restrictions for query optimization, an error occurs. This is the default.

DISABLE QUERY OPTIMIZATION

This means the materialized query table cannot be used for query optimization. The table can still be queried directly.

Catlog tables:

The SYSIBM.SYSVIEWS table contains one or more rows for each view, materialized query table, or user-defined SQL function.

Type:(F :SQL function, M :Materialized query table , V :View)

```
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT CREATOR||' '||CHECK||' '||TYPE||' '||ENABLE||' '||ISOLATION      00000100
||' '||NAME FROM SYSIBM.SYSVIEWS;                                       00000200
---------+---------+---------+---------+---------+---------+---------+---------+


---------+---------+---------+---------+---------+---------+---------+---------+
DSN81110 N V N   VDEPT
DSN81110 N V N   VHDEPT
DSN81110 N V N   VEMPLP
DSN81110 N V N   VEMP
DSN81110 N V N   VPROJ
DSN81110 N V N   VACT
DSN81110 N V N   VPROJACT
DSN81110 N V N   VCONA
DSN81110 N V N   VEMPPROJACT
DSN81110 N V N   VDSPTXT
DSN81110 N V N   VOPTVAL
```

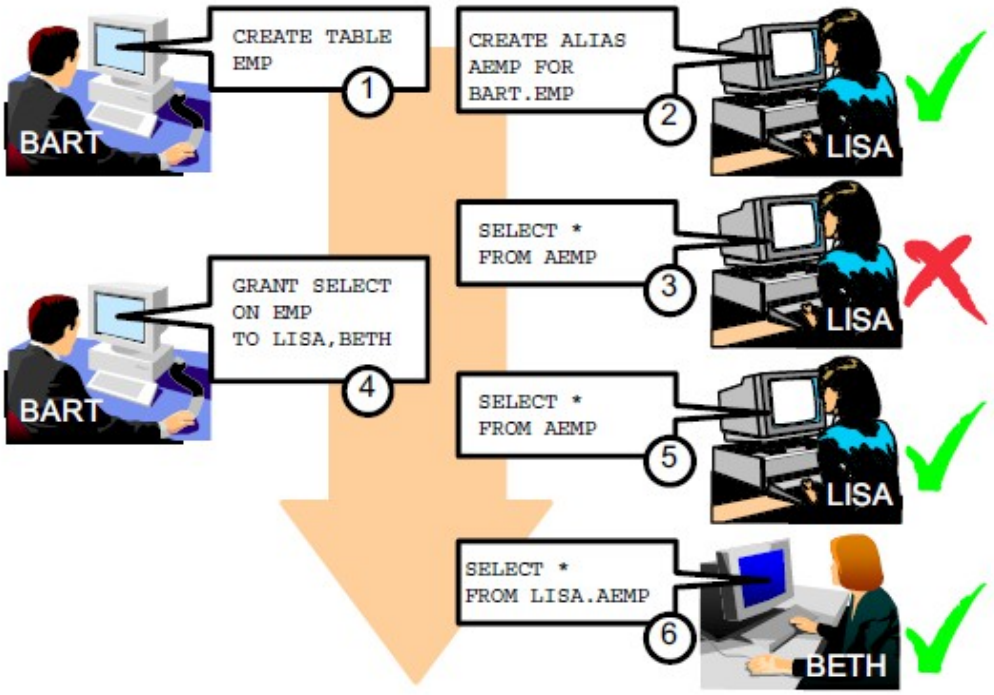
1. what is view
2. can you apply view on view
3. how many tables you can refer with view
4. what happen if you apply DML operations on view
5. View restrictions
6. Row disappearing concept in view
7. Tables related to view in CATLOG
8. Local, Cascade check options.
9. Difference Between view & MQT
10. REFRESH command
11. How MQT will create
12. DATA INITIALLY DEFERRED , REFRESH DEFERRED

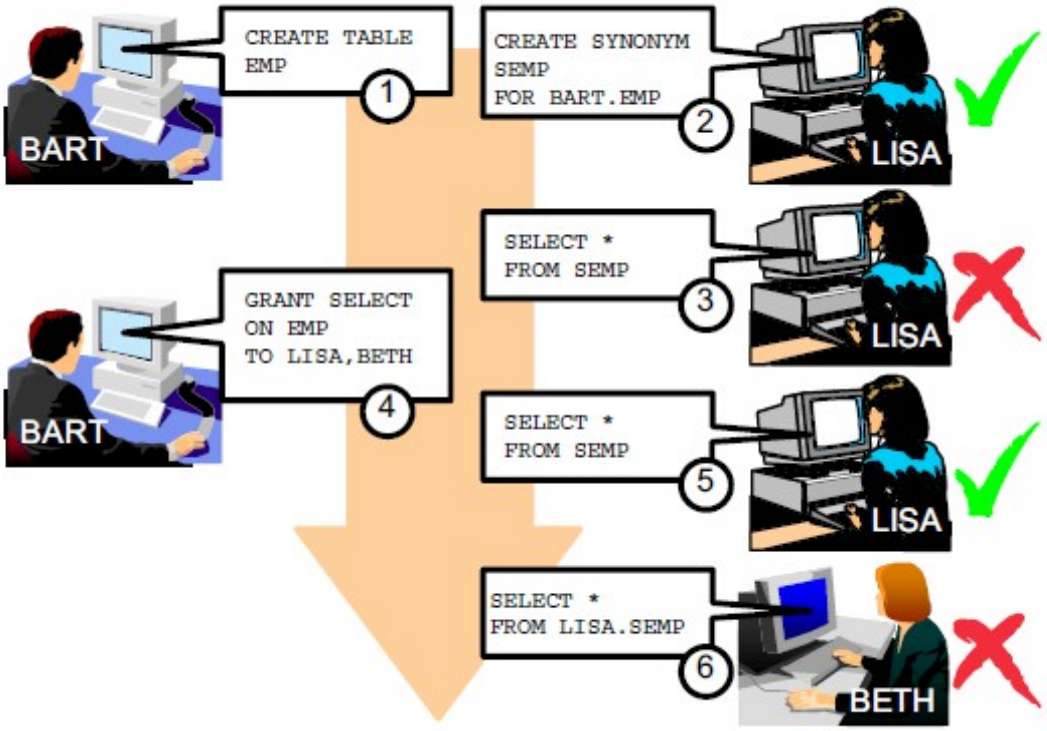
2.6 Alias-Synonym

Alias & Synonym are alternative names for a table or view. The difference between them shown in below.

S.NO	Alias	Synonym
1	Global object which can accessible to anyone	Private object which can be accessible to person who ctreate it
2	Used in Distributed enviorement	Used in Local envioremententonly
3	It can be created even the coresponding physical object is not there	It is created when the coresponding physical object is there
4	When object is droped coressponding physical objects will not droped	When object is droped coressponding physical objects will droped automatically
5	SYSADM authority or CREATE ALIAS Previlages will required.	SYSADM authority will not required
6	SYSIBM.SYSTABLES stores ALIAS catalog information	SYSIBM.SYSSYNONMS stores ALIAS catalog information
7	Quqlifer can be used for ALIAS	Qualifer can not be used for SYNONMN
8	CREATE ALIAS AEMP FOR BART.EMP	CREATE SYNONYM SEMP FOR BART.EMP

Below diagram visualise concept of view ans sysnonm


BART
CREATE TABLE
EMP
1
CREATE ALIAS
AEMP FOR
BART.EMP
2
LISA
SELECT *
FROM AEMP
3
LISA
BART
GRANT SELECT
ON EMP
TO LISA,BETH
4
SELECT *
FROM AEMP
5
LISA
SELECT *
FROM LISA.AEMP
6
BETH


BART
CREATE TABLE
EMP
1
CREATE SYNONYM
SEMP
FOR BART.EMP
2
LISA
SELECT *
FROM SEMP
3
LISA
BART
GRANT SELECT
ON EMP
TO LISA,BETH
4
SELECT *
FROM SEMP
5
LISA
SELECT *
FROM LISA.SEMP
6
BETH

2.7 Schema

- A schema is a collection of named objects classified logically in the database.in a database, you cannot create multiple database objects with same name. To do so, the schema provides a group environment. You can create multiple schemas in a database and you can create multiple database objects with same name, with different schema groups.

2.8 Index & Partitioning

1. What is Index & Indexspace?

- Index is logical data structure, it is ordered set of pointers (called row Id's) which points rows of a table.
- Physically index is stored in seperate Linear VSAM dataset called Indexspace. It will create automatically when index is created.
- There is no explicit CREATE INDEXSPACE statement.
- Index name has a maximum of 128 alphanumeric characters and index space name has a maximum of 8 alphanumeric characters that to made by Db2 based on your index name.

2. How Index Works

- For Explaination consider one example I.e is below query

```
SELECT * FROM Employee
WHERE Employee_Name = 'SUDHEER'
```

Query Execution without Index

- Once we run that query, what exactly goes on behind the scenes to find employees who are named Sudheer will search row by row.
- we can't just stop looking once we find just one row with the name ‘Sudheer’, because there could be other rows with that name. So it will serch upto last row. → This is called Full Table Scan

Full Table scan:

- A full table scan looks through *all* of the rows in a table – one by one – to find the data that a query is looking for.
- Obviously, this can cause very slow SQL execution. So performance is worst if it contain millions of rows.

Query Execution with Index

- When an index is created on column, it records the values & row ID's of that column.
- Entries are added to the index when new data is added to the table.
- When a query is executed & a condition is specified on a column in the WHERE clause is indexed, then if fetches in index first and navigate to data pages of tablespace through its pointer(row id's)
- By using index search time will reduce by using Binary tree algorithem (B-tree) which will explain in Index Architecture.

NON-UNIQUE INDEX	Multiple RIDs
ADAMS,BOBBY	1,11
ADAMS,SANDY	1,4
BROWN,TERRY	1,3
CALDWELL,SIMONE	1,14
FRIDRICH, JOHN	1,5
GAINES,LOIS	1,2
JONES,LARRY	1,7 - 1,12
JONES,LAVERNE	1,9
MITCHELL,JEAN	1,10
MITCHELL,TED	1,6
SMITH,DIANA	1,1
SMITH,GRACE	1,13
WISE,SARA	1,8

3. Types of Index?

There are different types of index based on different factors.

- Based on Uniqueness of column values
  - Unique Index
  - Non-Unique Index.
  - Unique where Not Null index
- Based on key/non-key column where index applied → Refer Keys (ch-3)
  - Primary Index
  - Secondary Index
- Based on builted index architecture → Refer Index Architecture
  - Clustered Index
  - Non-Clustered Index
- Partitiioning → Refer Partitiioning Concepts
  - Non Partitioning index
  - Partitioning (Secondary) index (PI)

    It has some further classification for table level partitioning
- Other Types
  - Expression based Index
  - XML Index

Note: Secondary index may be Non primary key index or Non partitioning index

Non-Unique Index :

- Each entry in index may have multiple Row ID's.
- For example if we create index on Name column then we may get multiple rows with same name as show in abow figure.
- For these kind of index number of rows in table & coressponding index may not match.

```
CREATE INDEX EMPNAME
ON EMP (FIRSTNME, LASTNAME);
```

Unique Index

- The column on which we are applying index should not contain dubplicate values.
- To ensure that a row is unique. For example, a unique index on the employee table ensures that no two employees have the same employee number.
- It may contain Null value but only one Null is permitted to make uniqueness.

```
CREATE UNIQUE INDEX DSNB10.XPHOTO
ON DSNB10.EMP_PHOTO_ATAB (DEPTNO ASC) -- ASC is default
```

UNIQUE WHERE NOT NULL

- Prevents the table from containing two or more rows with the same value, but Multiple null values are allowed.

```
CREATE UNIQUE WHERE NOT NULL INDEX DSNB10.XPHOTO
ON DSNB10.EMP_PHOTO_ATAB (DEPTNO ASC)
```

Primary Index

- A unique index on the primary key of the table.

Secondary Index

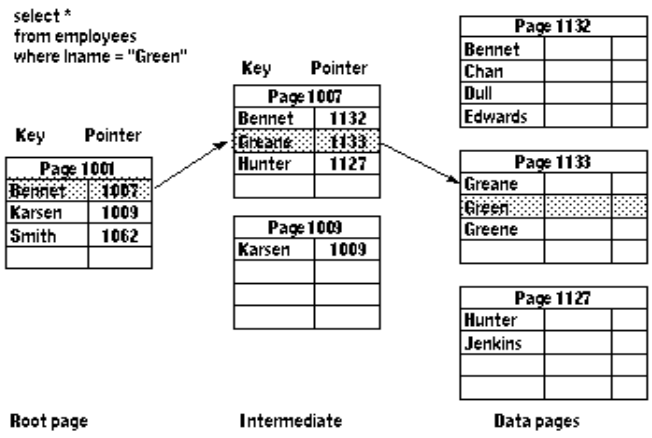
- An index that is not a primary index.

4. Explain Index Architecture?

index architecture

Clustering index

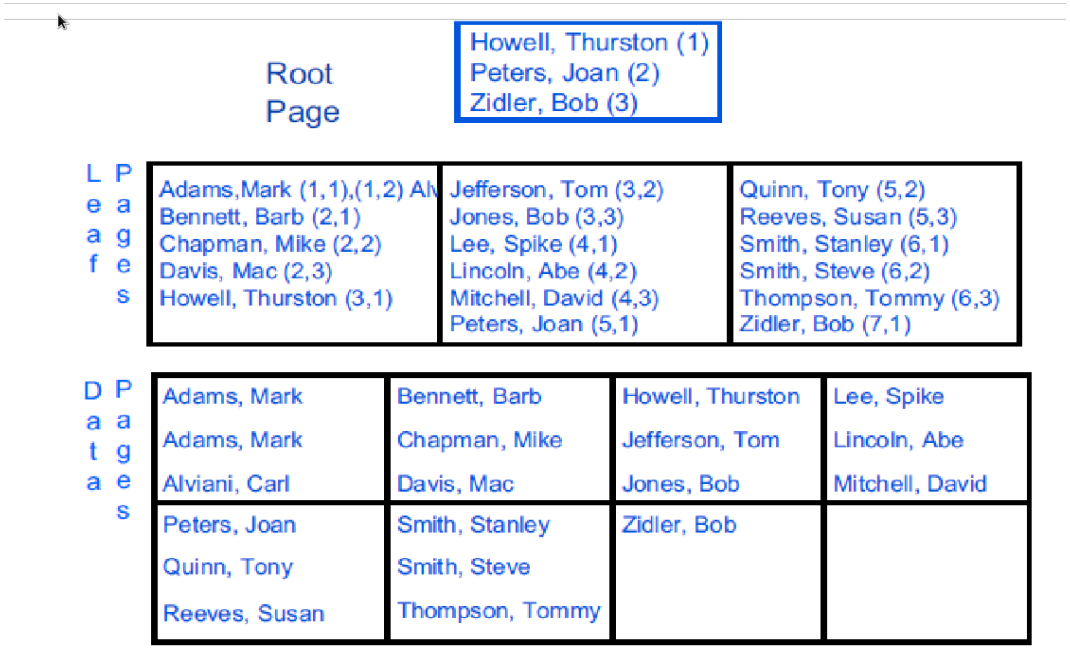
all data pages are divided into several levels if we want fetch one record binary search will hapend as shown in below example for searching green record



On the root level page, "Green" is greater than "Bennet," but less than Karsten, so the pointer for "Bennet" is followed to page 1007. On page 1007, "Green" is greater than "Greane," but less than "Hunter," so the pointer to page 1133 is followed to the leaf level page, where the row is located and returned to the user.

Clustering index

- Keep the data in the data pages in the same order as the corresponding keys in the index pages. This order is why only one clustered index can exist in any table
- Using a clustered index it is easy when groups of data that can be clustered are frequently accessed by some queries.--> advantage
- if we did any UPDATE or INSERT or DELETE operation then sorting will happened to make meaningfull order --> disadvantage
- Imagine if we have two clustered indexes on a single table – which index would determine the order in which the rows will be stored? Since the rows of a table can only be sorted to follow just one index, having more -- than one clustered index is not allowed. -->Table has single clustered index
- In our example, all of the car entries belonging to a single owner would be right next to each other on disk. This is the "clustering", or grouping of similar values, --> this is why it refered as clusterd index

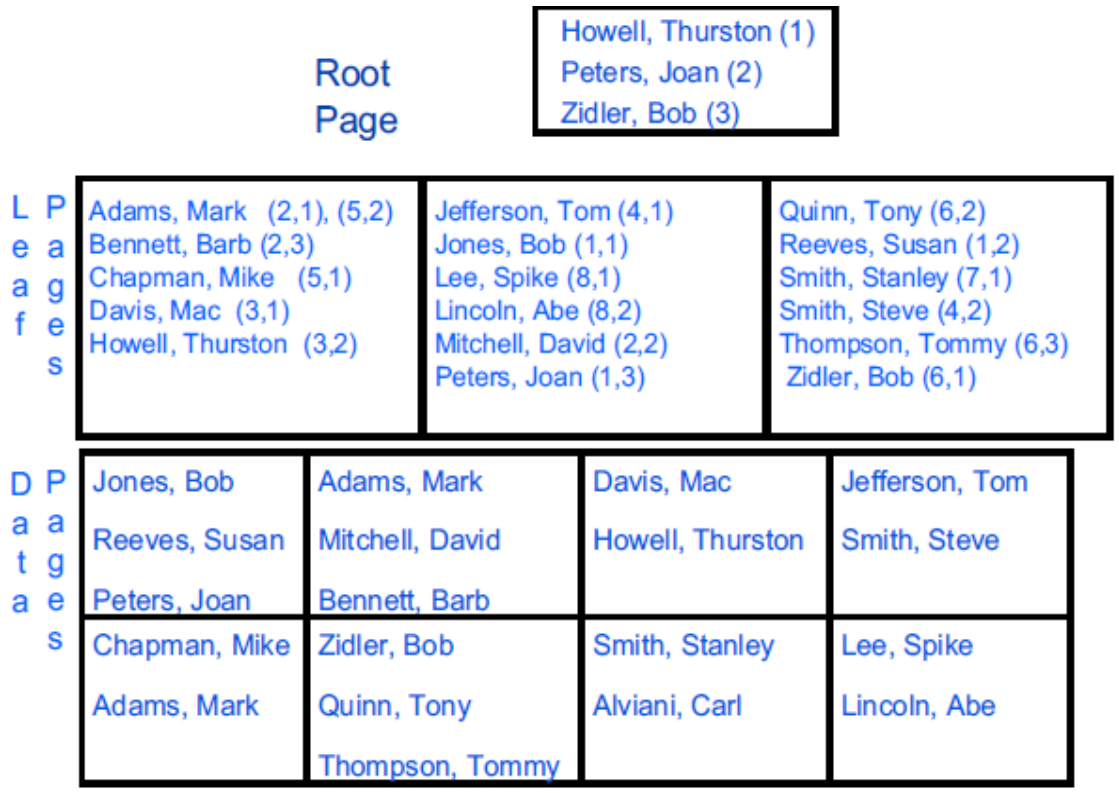


Let's retrieve all the rows for Mark Adams. For the clustered index, here are the I/Os necessary to accomplish this:

1. One I/O to read the index root page
2. One I/O to read the index leaf page containing the key Adams, Mark. There are two RIDs: 1,1 and 1,2
3. One I/O to read the 1st page, where rows 1 and 2 are both Mark Adam

Non-Clustering index

The database manager does not maintain any order of the data when compared to the order of the corresponding keys of a non-clustered index.so,many non-clustered indexes can exist in the table.

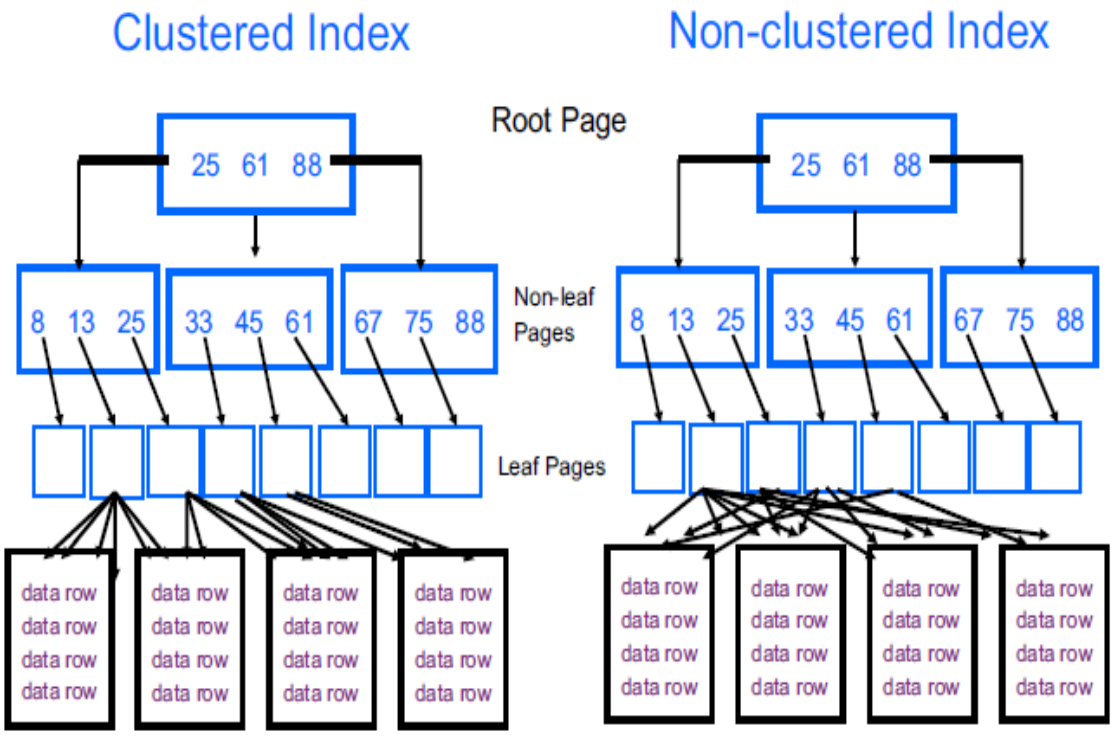


Let's retrieve all the rows for Mark Adams. For the non-clustered index, here are the I/Os necessary to accomplish this: --> total 4 I/Os

1. One I/O to read the index root page
2. One I/O to read the index leaf page containing the key Adams, Mark. There are two RIDs: 2,1 and 5,2
3. One I/O to read the 2nd page, where row 1 is Mark Adams
4. One I/O to read the 5th page, where row 2 is Mark Adams

Note:

- Non-clustered consumes relatively more IO & CPU time then that of Clustered Architecture
- In between Root and Leaf pages there may some non leaf gate also exit as shown in below to reduce search time



Expression based Index

You normally build an index on column values. You can also build an index on expression. DB2 stores the results of the expression in the index.

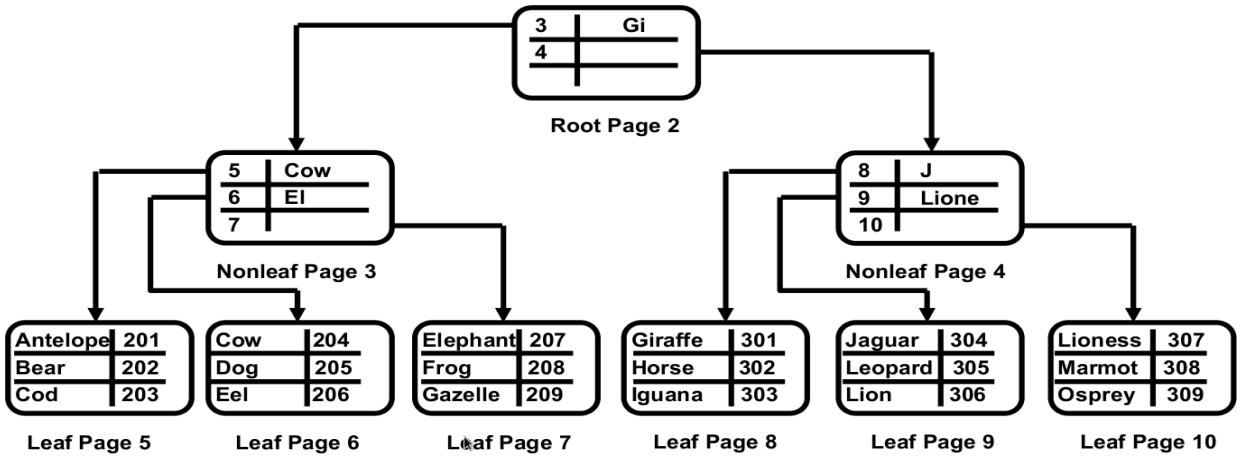
```
CREATE TABLE EMPLOYEE (
            ID          INTEGER       NOT NULL,
            LASTNAME    VARCHAR(20)   NOT NULL,
            FIRSTNAME   VARCHAR(20)   NOT NULL,
            SALARY      DEC(15,2)     NOT NULL,
            BONUS       FLOAT );

CREATE INDEX INDEX1
ON EMPLOYEE (SALARY + BONUS);
SELECT SALARY + BONUS, LASTNAME, FIRSTNAME
FROM EMPLOYEE
WHERE SALARY + BONUS < 51000;
```

Restrictions:

1. Maximum number of expressions in an index key is 64.
2. An index on an expression cannot be a clustering index.
3. DESC / RANDOM cannot be specified for indexes on expressions.
4. You cannot use ALTER to add columns to an index on an expression.
5. Primary key and foreign keys with expressions are not supported.

High-key value: A leaf page has a high-key value. This high-key value is, in fact, the first key on the next leaf page. All keys on a leaf page are less than its high-key value. For example, the high-key for leaf page 9 is “Lioness”.



1. Difference Between clusterd & non-clusterd index?
2. When is using a clustered index an advantage?
3. Why is it called a clustered index?
4. disadvantage to using a clustered index?
5. A table can have multiple non-clustered indexes?
6. A table can have multiple clustered indexes?

Partitioning

Partitioning Types

1. Index-controlled partitioning
2. Table-controlled partitioning.

Index control partitioning

Index-controlled partitioning is possible only with a classic partitioned table space.

Creating an index-controlled partitioned table space is a 3-step process.

Step 1: Create the partitioned table space

```
CREATE TABLESPACE USTS7
IN USDB1
NUMPARTS 4
SEGSIZE 0
```

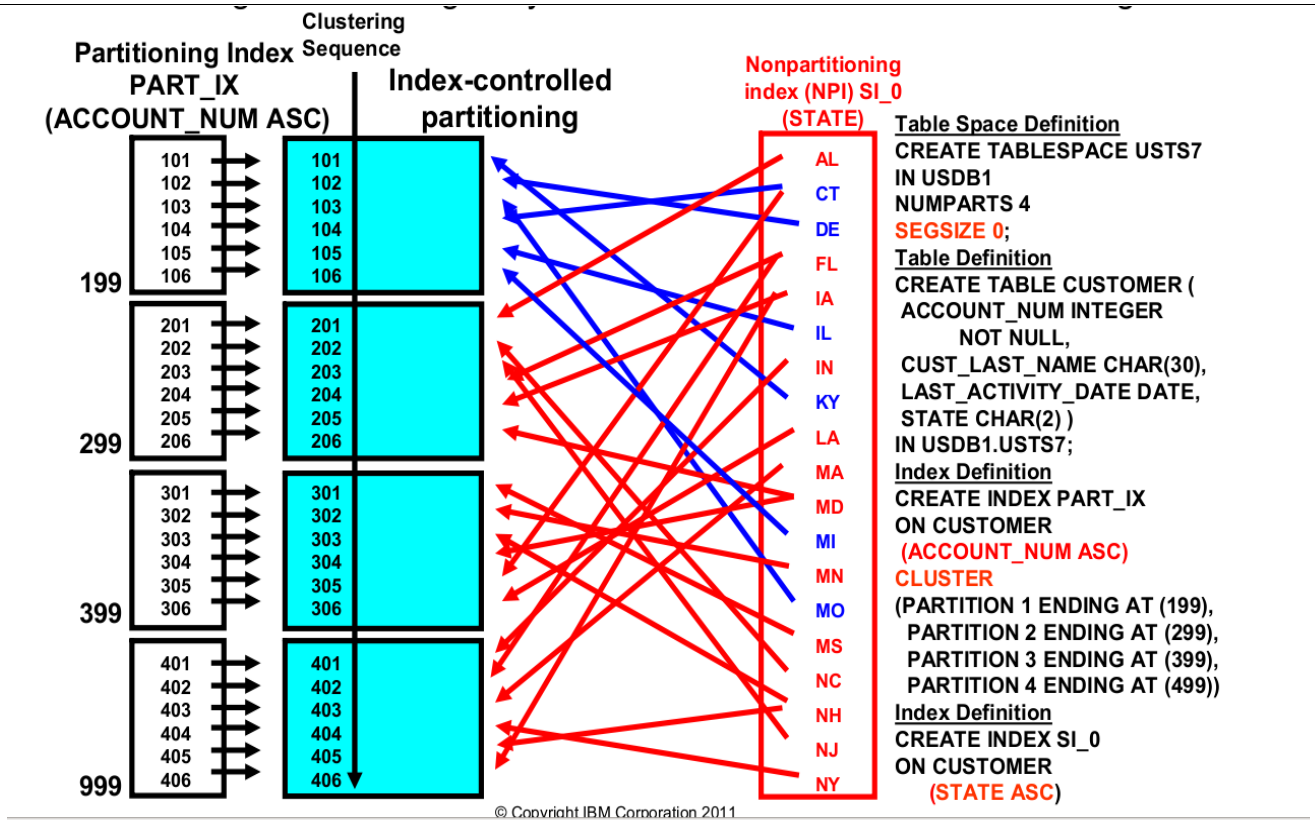
Step 2: Create the table

```
CREATE TABLE CUSTOMER ( ACCOUNT_NUM      INTEGER,
                        CUST_LAST_NAME   CHAR(30),
                        ACTIVITY_DATE    DATE,
                        STATE            CHAR(2) )
                        IN USDB1.USTS7;
```

Step 3: Create the partitioning index

```
CREATE INDEX PART_IX ON CUSTOMER (ACC_NO ASC)
CLUSTER ( PARTITION 1 ENDING AT (199),
          PARTITION 2 ENDING AT (299),
          PARTITION 3 ENDING AT (399),
          PARTITION 4 ENDING AT (499) )
```

Note: This example shows an outline of the syntax. DB2 also accepts the syntax PART n VALUES (value) in place of PARTITION n ENDING AT (value).



Q.The partitioning index in Index control partitioing ?

- In index control partitioning, Partitioning index is a Partitioned & Clustered index Index created with Limit keys on for table belongs to Partitioned table space.
- The partitioning index is also partitioned into the same number of VSAM data sets as is used for the table space partitions.

Q. Only one partitioning index can be created in a table space ?

In the concept of Index level Partitioning. The partioning Index created on the Table must be partitioned and clustered, we can't create more than one clustered intex on table so we can create only one paritioning Index here.

Q. high key for the last partition is mandatory or not?

You have to specify a high key value for the columns in the partitioning index. highest key for the last partition depends on how you define the table space.

- For table spaces you create without LARGE or DSSIZE, the constant you specify after ENDING AT is not enforced.
- For table spaces you create with LARGE or DSSIZE, the constant you specify after ENDING AT is enforced.

Q. Can we ALTER High limit key?

You can ALTER the high limit key values for one or more partitions. The altered partitions then have to be Reorg.

Q. Non-partitioning index (NPI) ? Logical partition ?

- In a non-partitioning index (NPI), the index entries that point to one particular data partition constitute an index logical partition.
- Some utilities, for example, REBUILD INDEX can run at the logical partition level. But NPIs can cause, for example, contention problems.

Q. Problems created by Non-partitioning index (NPI) ?

Table-controlled partitioning

Table-controlled partitioning is possible either with a classic partitioned table space or with a UTS PBR.

Create table

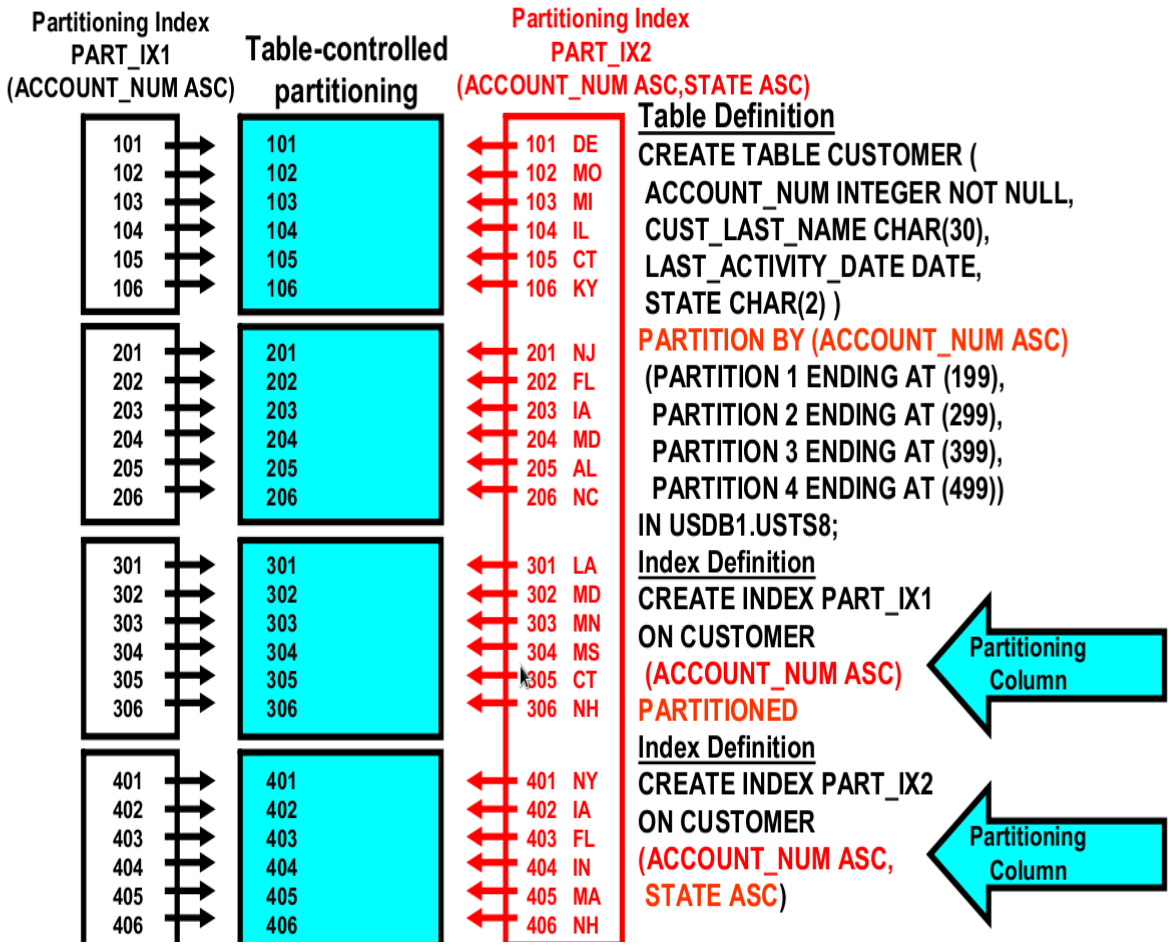
```
CREATE TABLE CUSTOMER ( ACC_NO    INTEGER   NOT NULL,
                        CUST_LAST_NAME        CHAR(30),
                        LAST_ACTIVITY_DATE  DATE,
                        STATE  CHAR(2) )
    PARTITION BY (ACCOUNT_NUM ASC)
    (PARTITION 1 ENDING AT (199),
    PARTITION 2 ENDING AT (299),
    PARTITION 3 ENDING AT (399),
    PARTITION 4 ENDING AT (499))
    IN USDB1.USTS8;
```

Create partitioned index

```
CREATE INDEX PART_IX1 ON CUSTOMER (ACCOUNT_NUM ASC)
PARTITIONED
```

Create non-partitioned index

```
CREATE INDEX PART_IX1 ON CUSTOMER (ACCOUNT_NUM ASC)
```



Partitioning Indexes

Partitioning index may or may not be partitioned ?

- Here partitioning is contolled by table. If we create index on partioning key column then it is a partitioning index (index may be partitioned or non-partitioned)
-

Index PART_IX1 : This is a partitioning index, because its key has the same left-most columns, in the same order, and (in this case only one column, ACCOUNT_NUM ASC) which partitions the table. The partitioning index PART_IX1 is also partitioned because the keyword PARTITIONED is specified in the CREATE INDEX statement.

Index PART_IX2 : This is also a partitioning index, because its key also has the same left-most columns, in the same order (in this case only one column, ACCOUNT_NUM ASC) which partition the table. In fact, it has a superset of the partitioning columns, because column STATE is included. The partitioning index PART_IX2 is not partitioned because the keyword PARTITIONED is not specified in the CREATE INDEX statement.

Table control Partitioning – Index

Partitioning Index

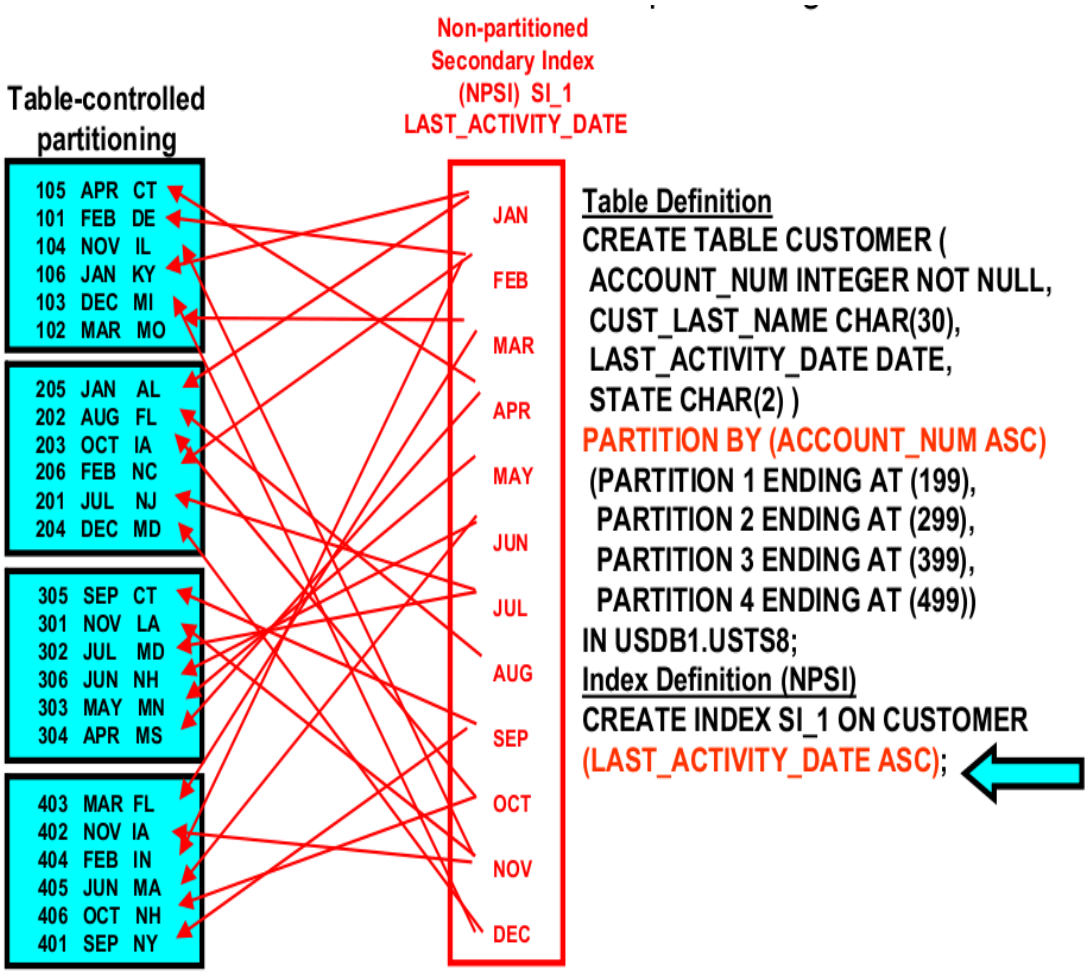
- Partitioned Partitioning Index
- Non-Partitioned Partitioning Index

Non-Partitioning or Secondary Index

- Non-Partitioned Secondary Index (NPSI)
- Data Partitioned Secondary index (DPSI)

→ Where the columns do not coincide with the partitioning columns of the table. That is, a non-partitioning (or secondary) index

Table-controlled partitioning – Non Partitioning(NPSI)



Non-partitioned secondary index (NPSI) :

- Index SI_1 is a secondary index, because its key (LAST_ACTIVITY_DATE ASC) does not have the same left-most columns as those which partition the table (in this case, ACCOUNT_NUM ASC).
- Simply saying partition key column and index column is not same. And Index is not parttioned
- Also index SI_1 lacks the option PARTITIONED so the index is non-partitioned. It is made up of a single data set.

Table-controlled partitioning – Non-Partitioning (DPSI)

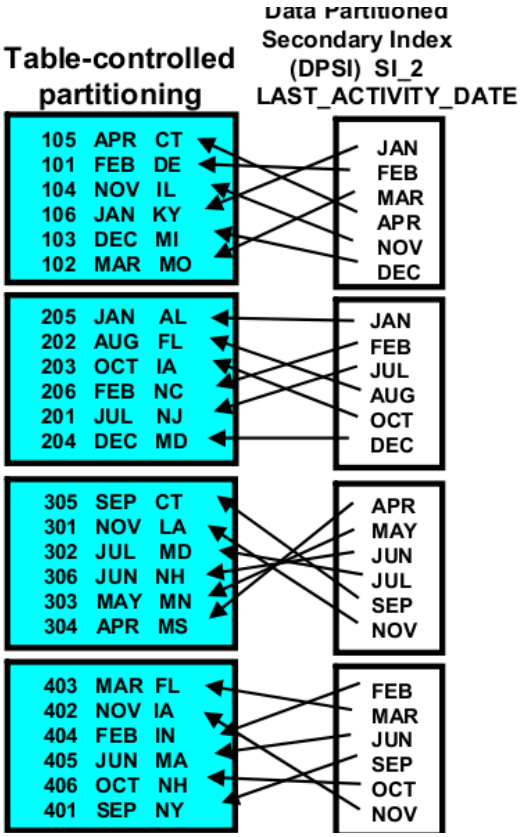


Table Definition

```
CREATE TABLE CUSTOMER (
 ACCOUNT_NUM INTEGER NOT NULL,
 CUST_LAST_NAME CHAR(30),
 LAST_ACTIVITY_DATE DATE,
 STATE CHAR(2) )
PARTITION BY (ACCOUNT_NUM ASC)
 (PARTITION 1 ENDING AT (199),
  PARTITION 2 ENDING AT (299),
  PARTITION 3 ENDING AT (399),
  PARTITION 4 ENDING AT (499))
IN USDB1.USTS8;
```

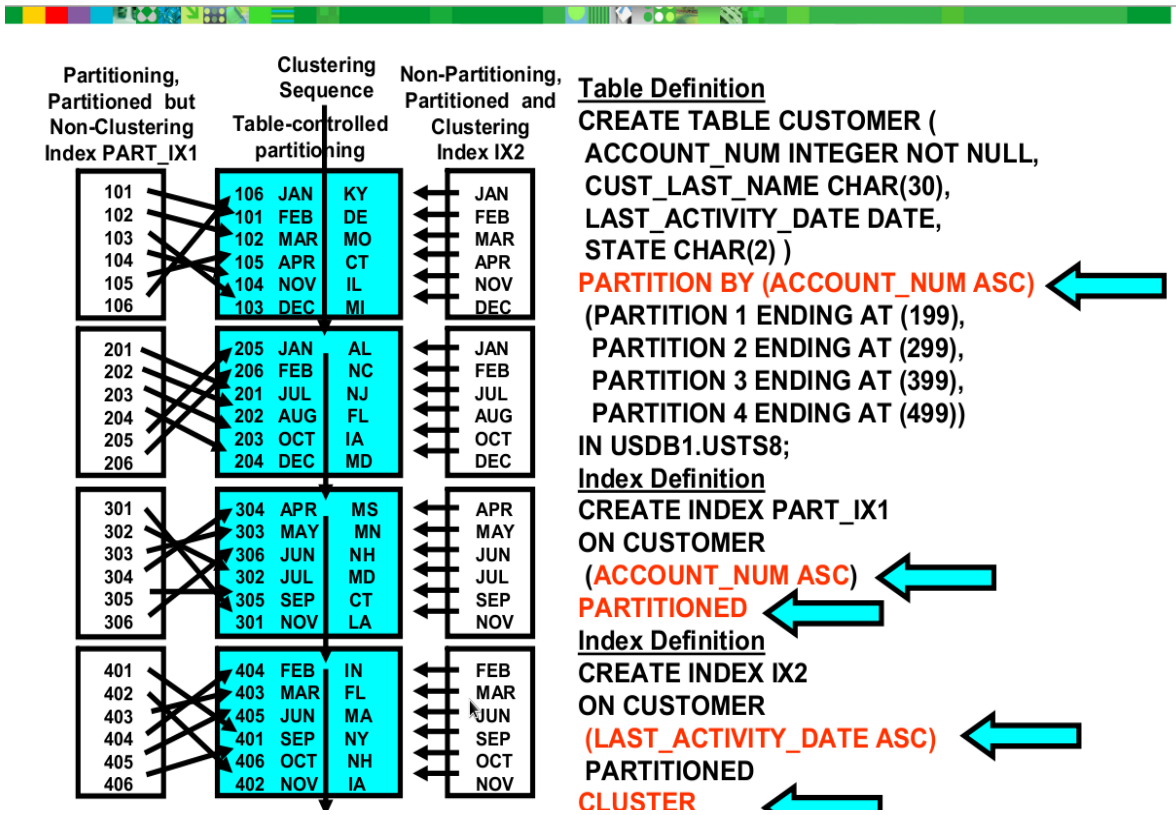
Index Definition (DPSI)

```
CREATE INDEX SI_2 ON CUSTOMER
(LAST_ACTIVITY_DATE ASC)
PARTITIONED;
```

Data Partitioned Secondary index (DPSI)

- Index is partitioned but but Partitioning key column and Index applied column is not same.
- The DPSI SI_2 is made up of multiple physical index partitions. The number of these index partitions equals the number of table space partitions. All RIDs of any one index partition point to the data rows of the corresponding table space partition.
- If the keyword UNIQUE is used when creating a DPSI, the uniqueness is enforced within each partition, not across partitions.

Table-controlled partitioning – partition one way and cluster another way



In above example we can observe

Paritioning + Partitioned + Non-Cluster

Non-Partitioning + Partitioned + Cluster

No explicit CLUSTER index

- If the clustering index is not defined, then the oldest index created on the table is used as the clustering index. earliar concepts obeys this point only.
- In above example PART_IX1 can't be clustered by default, since IX2 is cluster.

No index at all

If there are no indexes defined on the partitioned table, then clustering is done using the partitioning columns specified in the PARTITION BY clause.

NOTE:

- ALTER INDEX <index name> NOT CLUSTER on the current clustering index. Clustering continues to be done according to this index until a new clustering index is explicitly defined.
- ALTER INDEX <index name> CLUSTER on any index that should become the clustering index. Rows added to the table after these two ALTERs are clustered according to the new clustering index, but the existing rows remain in their current location.
- REORG the table space to rearrange the rows in the new clustering index order. All existing rows are rearranged in the new clustering sequence. Any new rows are inserted in the new clustering sequence.

S.NO	Index control partitioning	Table-controlled partitioning
1	It possible only with a classic partitioned table space.	It is possible either with a classic partitioned table space or PBR.
2	Partitioning is controlled by an index called a partitioning index.	Partitioning is controlled by table.
3	A partitioning index is required; a clustering index is required.	A partitioning index is not required; a clustering index is not required.
4	Partitioning index must be partitioned	Partitioning index may or may not be partitioned
5	Only one partitioned index can be created in a table space.	Multiple partitioned indexes can be created in a table space.
6	The high-limit key is not enforced if the table space is non-large.	The high-limit key is always enforced.

http://www.toadworld.com/platforms/ibmdb2/w/wiki/6829.indexing-and-partitioning

5. Advantages/Reasons by using Index?

Optimization

To improve performance. Access to data is often faster with an index compare to without index. i.e gives best acess path.

Uniqueness

Thes advantage will come under Unique index only. For example, a unique index on the employee table ensures that no two employees have the same employee number.

Clustering

Concept of Cluster index reduces CPU time & IO to some more extent then that using non-clustered index.

Partitioning

1. To find access path (optimization) → performance will increase
2. To identify rows uniquly

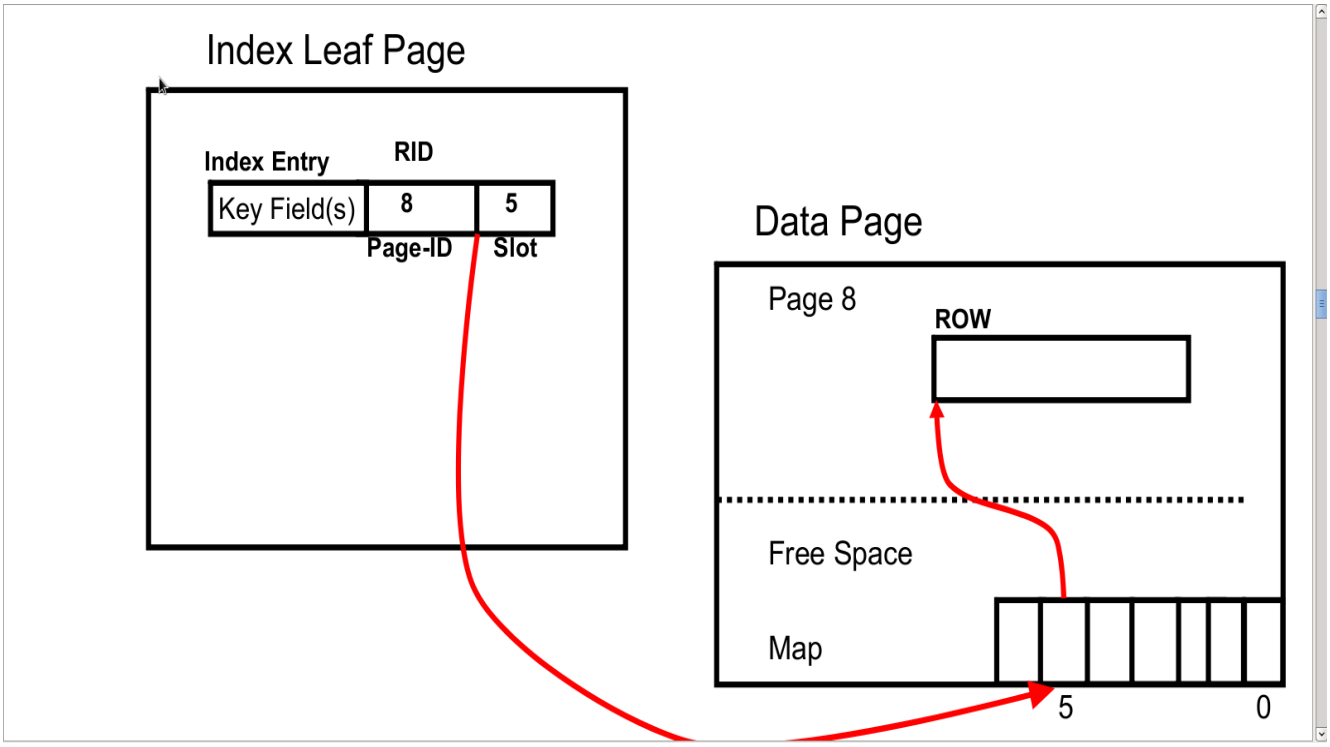
To establish relation between tables

Row ID

It is a datatype or pointer data type which points data in from index.

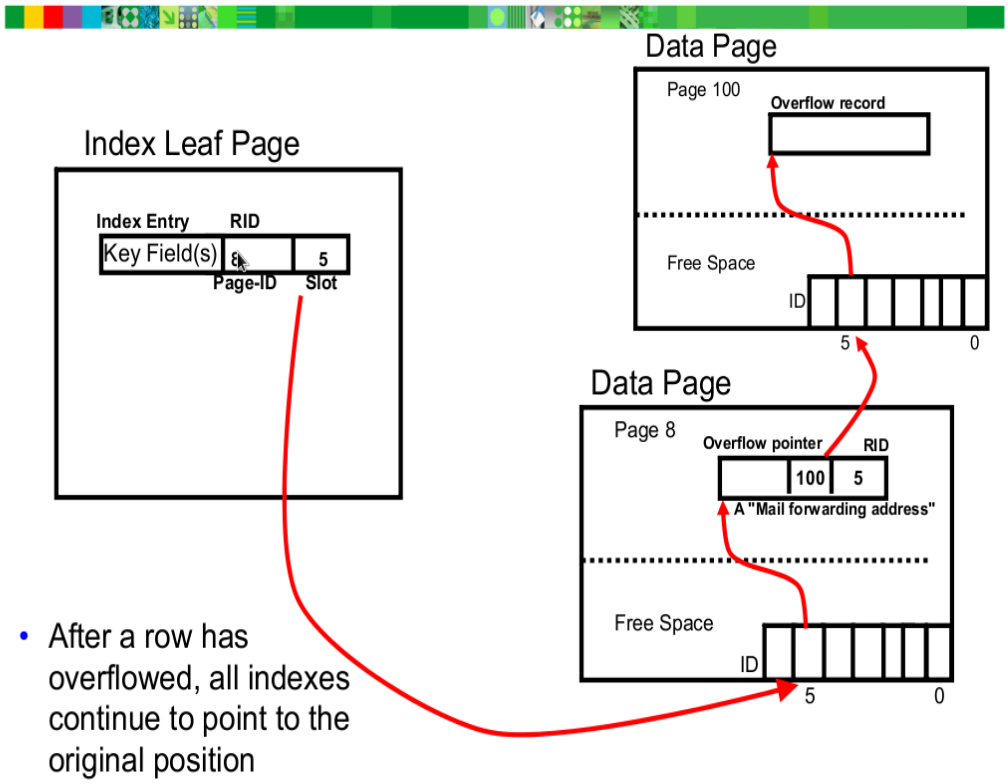
Row ID (Page number, Row number in that page)

Addressing Shemes



1. RIDs are divided into a page number followed by a slot number.
2. A RID is 4 bytes in length for non-large partitioned & nonpartitioned table spaces. The first 3 bytes of a RID refer to a data page, and the 4th byte of the RID points to a slot in the ID map array that contains the offset of the row on the page.
3. In the case of a large partitioned table space, the RIDs in indexes on the table are 5 bytes. The first 4 bytes of the RID refer to a data page, and the 5th byte of the RID points to a slot in the ID map array.

Over flow pointer



1. Rows with VARCHAR columns . An SQL UPDATE may increase the length of a variable length row. it attempt to keep the row on its original page. If there is insufficient free space, DB2 may have to move the expanded row to a new page before that it have to be leave overflow pointer at original datapage called MAIL FORWORDING ADDRESS .
2. Their occurrences can be reported by the RUNSTATS utility (see later) and updated in SYSIBM.SYSTABLEPART in columns NEARINDREF and FARINDREF. it's value is -1 if statistics have not gathered. Not applicable if the table space is a LOB
3. FARINDREF :Number of rows that have been relocated far from their original page.
4. NEARINDREF :Number of rows that have been relocated near their original page.

2.9 Index space & Index Characteristics

Indexspace Attribiutes at Index Creation

```
CREATE INDEX IX4EMPL
       ON EMPLOYEE(LAST_NAME ASC,DEPTNO DESC)
USING STOGROUP USCV831S
PRIQTY 7200
SECQTY 7200
PADDED
FREEPAGE 5
PCTFREE 20
BUFFERPOOL BP2

ALTER INDEX IX4EMPL
PRIQTY 14400
SECQTY 1440
BUFFERPOOL BP3
FREEPAGE 15
PCTFREE 25
NOT PADDED

RENAME INDEX
```

PRIQTY & SECQTY nonpartitioned indexes

- If you specify PRIQTY with a value other than -1, the primary space allocation is at least n kilobytes, where n is the value of integer with the following exceptions:

  - For 8KB page sizes, if integer is less than 24, n is 24.
  - For 16KB page sizes, if integer is less than 48, n is 48.
  - For 32KB page sizes, if integer is less than 96, n is 96.
  - For any page size, if integer is greater than 4194304, n is 4194304.

- If you specify PRIQTY -1 or do not specify PRIQTY, actual primary quantity is determined as follows:
  - if the IXQTY subsystem parameter value is specified and is greater

than 0, actual primary quantity is at least the value of IXQTY.

- If the IXQTY subsystem parameter is not specified or is 0, actual primary quantity is one cylinder.

- For SECQTY db2 has som set of formulaes to determine it

PRIQTY & SECQTY partitioned indexes

- A value of 12 is used for PRIQTY and SECQTY

Index Characteristices updated in Catlog tables

SYSIBM.SYSINDEXES

One row is inserted into the SYSIBM.SYSINDEXES catalog table when a CREATE INDEX statement is executed.

- **SYSIBM.SYSINDEXES**

NAME	INDEXSPACE	TBNAME	CLUSTERING	CLUSTERRATIOF
IX1EMPL	IX1EMPL	EMPLOYEE	Y	+0.1000000000000000E+01
IX2EMPL	IX2EMPL	EMPLOYEE	N	+0.7777777777777777E+00
IX3EMPL	IX3EMPL	EMPLOYEE	N	+0.7777777777777777E+00
IX4EMPL	IX4EMPL	EMPLOYEE	N	+0.7777777777777777E+00

NLEAF	NLEVELS	UNIQUERULE	PADDED	AVGKEYLEN	PGSIZE	BPOOL
1	2	U		6	4096	BP2
1	2	D	N	28	4096	BP2
1	2	N		51	4096	BP2

CLUSTERING: This indicates if the index is a clustering index or not

CLUSTERRATIOF : When multiplied by 100, the value of the column is the percentage of rows that are in clustering order. For example, a value of +0.1000000000000000E+01 indicates 100%.

NLEAF : Number of active leaf pages in the index. The value is -1 if statistics have not been gathered.

NLEVELS : Number of levels in the index tree. The value is -1 if statistics have not been gathered.

UNIQUERULE : This indicates whether the index is unique.

AVGKEYLEN : Average length of keys within the index. The value is -1 if statistics have not been gathered.

- **SYSIBM.SYSINDEXPART**

IXNAME	PQTY	SQTY	PCTFREE	FREEPAGE	SPACEF
IX1EMPL	-1	-1	10	0	+0.7200000000000000E+03
IX2EMPL	-1	-1	10	0	+0.7200000000000000E+03
IX3EMPL	-1	-1	10	0	+0.7200000000000000E+03

- **SYSIBM.SYSKEYS**

IXNAME	COLNAME	COLSEQ	ORDERING
IX1EMPL	EMPNO	1	A
IX2EMPL	LAST_NAME	1	A
IX2EMPL	FIRST_NAME	2	A
IX3EMPL	E_MAIL_ADDR	1	A
IX4EMPL	LAST_NAME	1	A
IX4EMPL	DEPTNO	2	D

3. DB2 - Data objects management

3.1 Naming conventions

→ Any name in DB2 may include the alphabetic characters A - Z, numbers 0 – 9, & the special characters @, #, $, and _ . But it does't start with _

→ The name for a database & Pagesets (table space,index space) can be up to a maximum of 8 characters. The name of other objects on the visual can be up to a max of 128 characters.

Below concept explains namming conventions of underlaying LDS of Tablespace and indexapces

VCAT.DSNDB*x*.*dbname*.*psname*.*y*0001.*znnn*

1. *VCAT* : catalog name or alias (up to 8 chars) mention in VCAT clause
2. DSNDB*x*: **x** :<C (for VSAM clusters) or D (for VSAM data components)>
3. *dbname* : database name
4. *psname* : 8 character name of Table space name or First 8 characters of index space name. If first 8 chars of indexspace name is name then Db2 is smart to scrambling the index name to ensure uniqueness within the database .
5. *y*0001:Instance qualifier for the data set. Define one data set for the table space or indexspace with a value of I for *y* and if you *do not* plan to run the following items:

REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE

CHECK DATA with SHRLEVEL REFERENCE

CHECK INDEX with SHRLEVEL REFERENCE

CHECK LOB with SHRLEVEL REFERENCE

Otherwise Define two data sets if you plan to run the following items:Define one data set with a value of I for *y*, and one with a value of J for *y*.

6. znnn (Data set number)

- The first digit *z* of the data set number is represented by the letter A, B, C, D, or E, which corresponds to the value 0, 1, 2, 3, or 4 as the first digit of the partition number. A001-

A001- A999 for partitions 1 to 999

B000-B999 for partitions 1000 to 1999

C000-C999 for partitions 2000 to 2999

D000-D999 for partitions 3000 to 3999

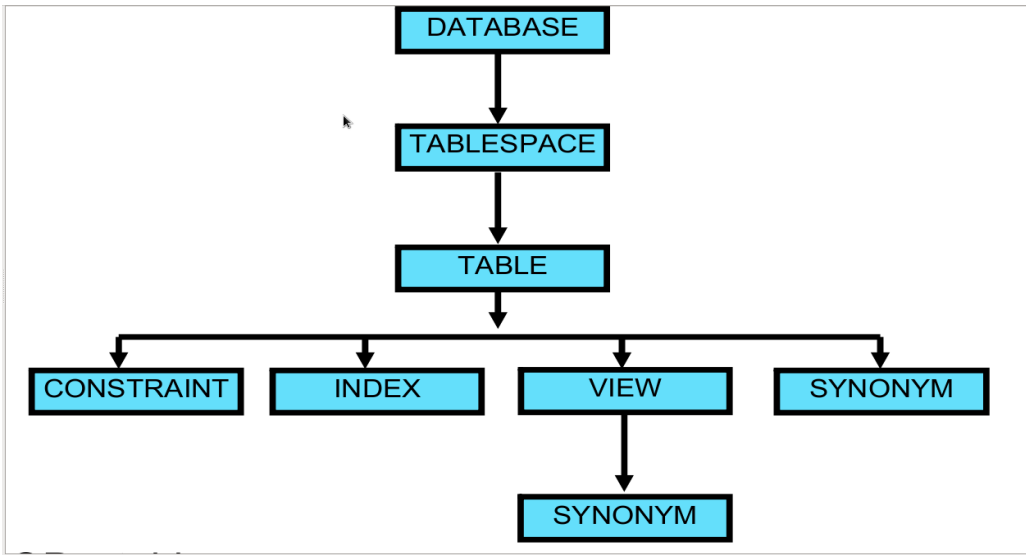
E000-E096 for partitions 4000 to 4096

For segmented table space A001-A032

- For simple or segmented table spaces, the number is 001 (preceded by A) for the first data set. When little space is available, DB2 issues a warning message. If the size of the data set for a simple or a segmented table space approaches the maximum limit, define another data set with the same name as the first data set and the number 002. The next data set will be 003, and so on

3.2 Object dependencies

(I dropped one, but I lost every thing)



If we Drop an object then the dependencies on that object also deleted. For example if we drop the table then following will happened

1. Any dependent objects are dropped such as indexes, views, synonyms, and triggers
2. Any referential constraints are dropped
3. Entries for PROJECT are removed from SYSTABLES and SYSCOLUMNS
4. Any dependent application plans or packages are invalidated
5. Cached dynamic SQL statements that use PROJECT are removed
6. Access path statistics and space statistics for the table are deleted
7. Security privileges for PROJECT are removed from SYSTABAUTH
8. DB2 reclaims the space if PROJECT is in a segmented table space .

RESTRICT ON DROP

You must alter tables that have been created with RESTRICT ON DROP to remove the restriction before you can drop them.

3.3 Automatic Creation of Objects

1. If you do not specify the IN clause in CREATE TABLE statement, DB2 creates an implicit database. name of the database is DSNnnnnn where nnnnn ranges from 00001 to 10000 & creator is SYSIBM.
2. The characteristics of the implicit database depend on the DSNZPARM settings. You can use the ALTER DATABASE statement to change the characteristics.
3. partition-by-growth universal table space is implicitly created if the CREATE TABLE statement does not specify PARTITION BY clause.
4. The implicit partition-by-growth universal table space is created with DSSIZE 4G, SEGSIZE 32, MAXPARTITIONS 256, LOCKSIZE ROW, & LOCKMAX SYSTEM.
5. partition-by- range universal table space is implicitly created if the CREATE TABLE statement specifies PARTITION BY clause.
6. The partition-by-range universal table space is implicitly created with DSSIZE 4G, SEGSIZE 32, MAXPARTITIONS 0, LOCKSIZE ROW, and LOCKMAX SYSTEM.

```
CREATE TABLE TT(C1 INTEGER NOT NULL UNIQUE
                C2 INTEGER NOT NULL UNIQUE);
```

- **SYSIBM.SYSTABLES**

```
SELECT NAME,DBNAME,TSNAME,OWNER
FROM   SYSIBM.SYSTABLES
WHERE  NAME = 'TT' AND
       CREATOR = USER;
---------+-----+---------+---------+-----
NAME     DBNAME      TSNAME     OWNER
---------+-----+---------+---------+-----
TT       DSN00050    TT         TSOUD01
```

- **SYSIBM.SYSINDEXES**

```
SELECT NAME,OWNER,TBNAME,TBCREATOR
FROM   SYSIBM.SYSINDEXES
WHERE  TBNAME = 'TT';
---------+---------+---------+---------+---------+-----
NAME             OWNER          TBNAME         TBCREATOR
---------+---------+---------+---------+---------+-----
TTABCDEF_#_Y6P   TSOUD01        TT             TSOUD01
TTABCDEF_#_VZP   TSOUD01        TT             TSOUD01
```

7. If tablespace is implicitly created the it will take DSNDB04 database.
8. These implicitly created databases are not dropped, even though all the content in it is drooped. It should not resuse also until reaches DSN10000. Once this number is reached, DB2 starts wrapping around.

3.4 Compression in DB2

Compression in DB2 refers 2 concepts

- Data Compression
- Index Compression

Data Compression

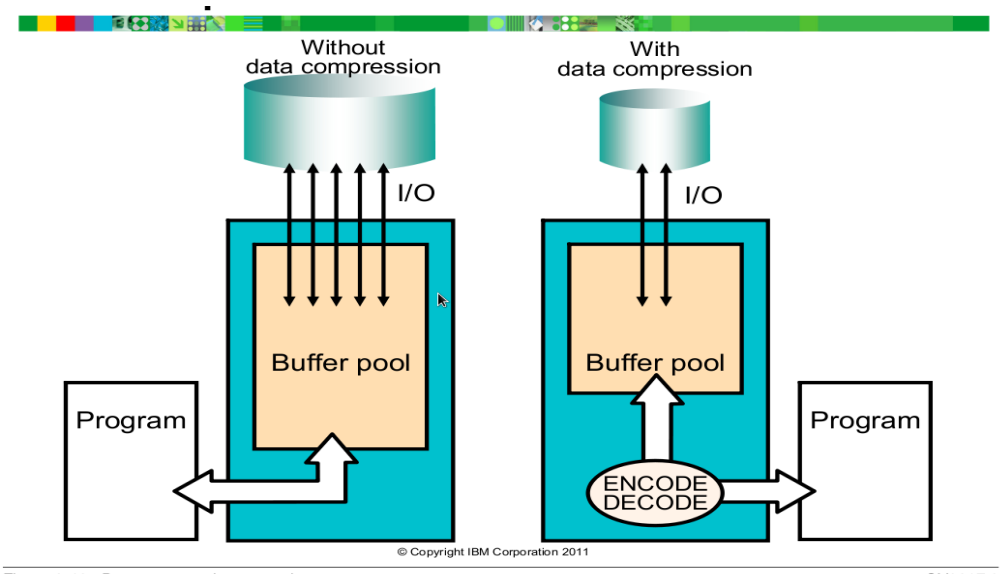
1. What is data compression ?How Db2 nows wheater the data is compressed or not?

   Data compression means compression of data in table space or Partition. A bit in the row header indicates whether a row has been compressed or not.

2. What to compress? What not to compress?

   You use the COMPRESS clause of the CREATE or ALTER TABLESPACE statements. You can't compress data in LOB table spaces and workfile database.

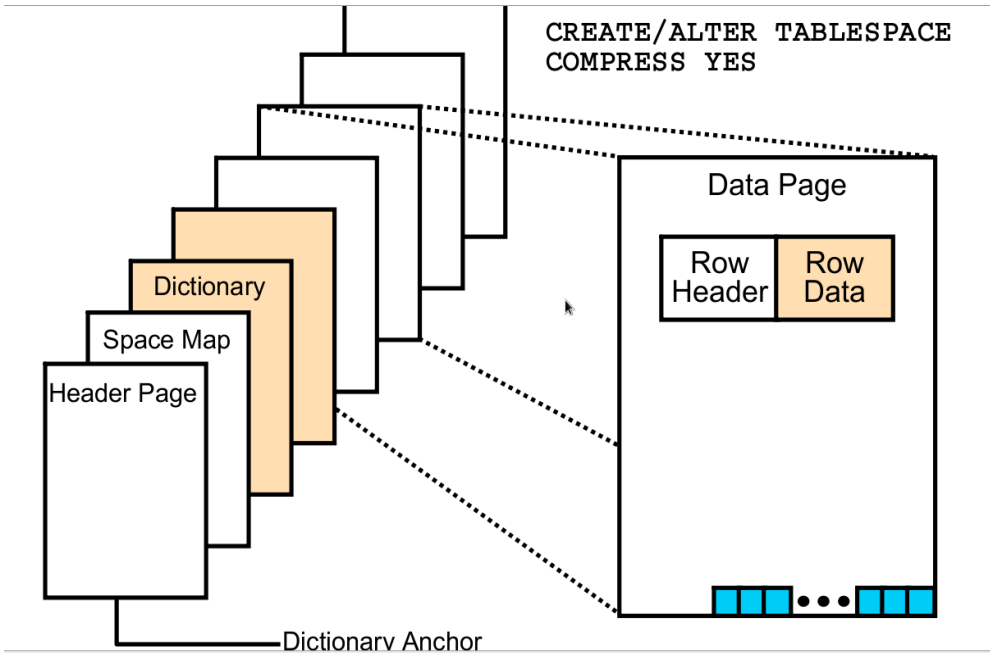
3. Db2 data compression advantages ?
   - Data compression helps to save disk space.
   - Compressing data can also help improve buffer pool performance in following ways
     - Bufferpool can accomdate more data if that data is compressed.
     - It will take less IO to get data from disk if it is compressed.



4. Wheater I have to do Data compression or not ?

   The decision whether to use data compression for a specific table space is a tradeoff between resource savings (disk space, I/O, and buffer pool requirements) and the incurred CPU overhead of the compression or decompression.

5. How to Compress?What is dictionary page?



- The dictionary page is a structure which defines the rules used to encode and decode data.
- Dictionary not built on a table space with COMPRESS YES attribute ,
- Prior to DB2 10, Dictionary page can build during LOAD & REORG utilities are executed. For some customers, REORG or LOAD are not executed frequently .
- From DB2 10 NFM, Dictionary page can build by Insert & merge statements execution

USE DATA COMPRESSION field (IMPTSCMP subsystem parameter)

The IMPTSCMP subsystem parameter determines whether a table space that is created in an implicitly created database is to use data compression.

Acceptable values	NO, YES
Default:	NO
Update:	option 11 on panel DSNTIPB

Index Compression? Different from Data compression?

Compression Results	Data	Index
Level of compression	Row	Page
Compression in DASD	Yes	Yes
Compression in buffer pool and log	Yes	No
dictionary page	Yes	No
Typical compression ratio	10% to 90%	25% to 75%

DSN1COMP Stand-alone utility:

- DSN1COMP stand-alone utility provides a close approximation to DB2 compression results without altering the data.
- Input to the utility can be VSAM data sets containing DB2 table space or partition data, DB2 full image copy data sets, or sequential data sets containing DB2 table space or partition data

3. DB2 Data integrity & constraints

- A key is a column or an ordered collection of columns in the table to identify each record uniquely.

Unique Key:

- A unique constraint is a rule that the values of a key are valid only if they are unique. A unique can contain one unique key.

Primary Key:

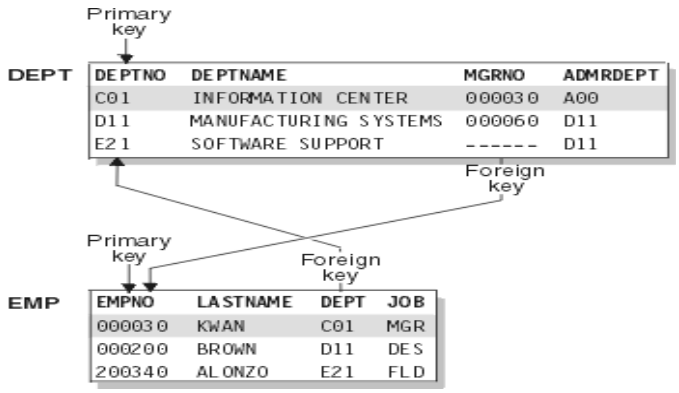
- A primary key is unique key which cannot contain null values. A table may have one and only one primary key

composite key:

- it is an ordered set of two or more columns of the same table.

Parent Key & Foregin key:

- Key in parent table is parent key



A column or set of columns that contains values from some table's unique key and is dependent on them is called foreign key.

consider first table which has 100 deparment numbers. For suppose each

DEPTNO has one of 5 sets of common data(EMPNO, LASTNAME, DEPT, JOB) these 4 items replaced by single MGRNO for each employee. That 5 sets of 4 items build child table with 5 records. Here MGRNO in parent table is foreign key which may not be unique, and which may null also.

In this example we reduce data redundancy which comes under concept called database normalization

Foreign key is non unique & may be nullable.

The constraint foreeign key is refered as referential constraint

<u>Entity integrity</u>: it concerns the concept of a primary key. Entity integrity is an integrity rule which states that every table must have a primary key

<u>Referential integrity:</u> It concerns the concept of a foreign key. The referential integrity rule only be in one of two states. The usual state of affairs is that the foreign-key value refers to a primary key value of some table. Otherwise a foreign-key value can be null.

<u>Domain Integrity:</u> maintaining the correctness of attribute values within conditions.

- data type,

- length or size

- is null value allowed

4th type os integrity is <u>User-defined Integrity</u>

Integrity means maintaing consistant data

<u>DELETE RULES</u>

The table designer chooses one of the following DELETE RULES:

- ON DELETE CASCADE
- ON DELETE SET NULL
- ON DELETE RESTRICT
- ON DELETE NO ACTION

ON DELETE CASCADE

DEPT

DEPTNO	DEPTNAME	MANAGER	DIVISION
BLUE	PLANNING	000020	EASTERN
GREEN	OPERATIONS	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPL

EMPNO	LASTNAME	FIRSTNAME	WORK DEPT	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-88-2794	52750.00
000020	THOMPSON	MICHAEL	BLUE	888-89-4261	31000.00
000030	KWAN	SALLY	GREEN	888-88-9456	33000.00

DEPT

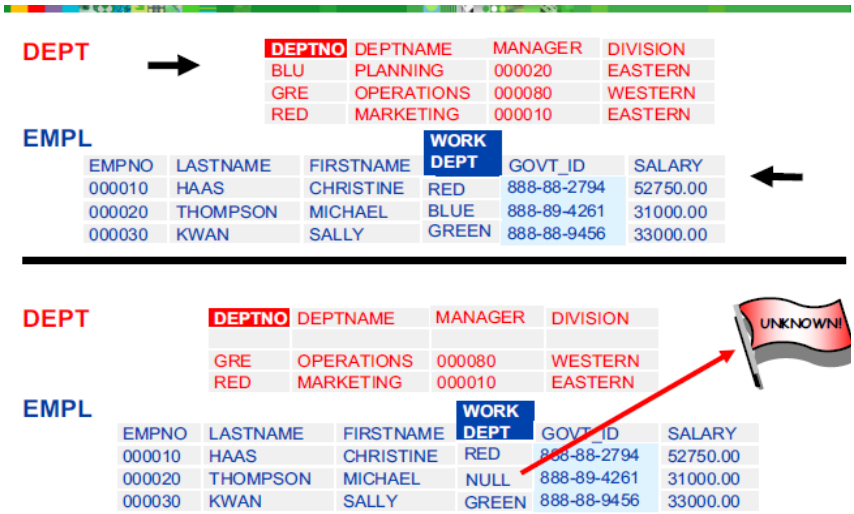
DEPTNO	DEPTNAME	MANAGER	DIVISION
GREEN	OPERATIONS	000080	WESTERN
RED	MARKETING	000010	EASTERN

EMPL

EMPNO	LASTNAME	FIRSTNAME	WORK DEPT	GOVT_ID	SALARY
000010	HAAS	CHRISTINE	RED	888-88-2794	52750.00
000030	KWAN	SALLY	GREEN	888-88-9456	33000.00

In this example, the referenced table is the department table. The foreign key EMPL.DEPTNO was defined with the ON DELETE CASCADE rule. What happens when the BLUE WORKDEPT row is deleted from the department table with ON DELETE CASCADE as the delete rule? All employees in the BLUE DEPTNO are also deleted.Record in Parent table is deleted then all dependent rows in child table are also deleted

ON DELETE SET NULL



In this example, the referenced table is the department table. The foreign key EMPL.WORKDEPT was defined with an ON DELETE SET NULL rule. When the BLUE DEPTNO row is deleted from the department table, DB2 will set the WORKDEPT value for all BLUE employees to NULL.

Record in Parent table is deleted then valus dependent rows in child table will become null

ON DELETE NO ACTION

In this example, the referenced table is the department table. The foreign key

EMPL.WORKDEPT was defined with an ON DELETE RESTRICT rule. Before DB2 will allow the BLUE DEPT row to be deleted from the department table, it checks to ensure that there are no employee rows where WORKDEPT = 'BLUE'. If DB2 finds even one employee row with a value of BLUE in the WORKDEPT column, the delete fails (is restricted).

Records of parent table canlt deleted unless or until depentened tables are present in child table.

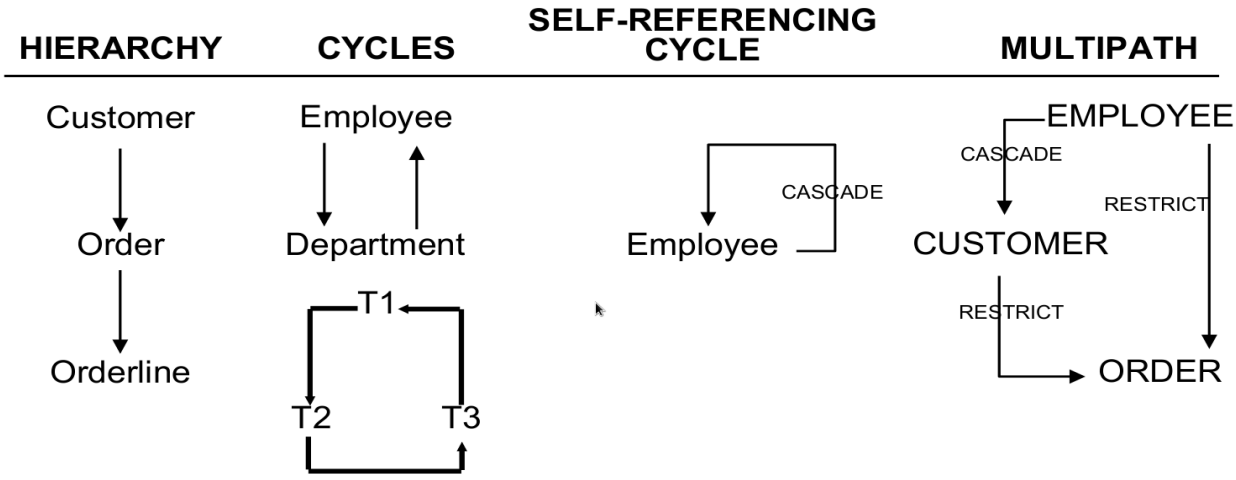
ON DELETE NO ACTION

Same as for RESTRICT except for self-referencing tables.

Referential structures

- Tables linked by Referential Constraints is called Referential structures.

Types of Referential Structure:



Primary key Creation through ALTER

```
ALTER TABLE LOCATION
      ADD CONSTRAINT LOCID
  PRIMARY KEY(LOCID);
```

Foreign key Creation through ALTER

```
ALTER TABLE DEPARTMENT
      ADD CONSTRAINT DD
          FOREIGN KEY (SUPERIOR_DEPTNO)
          REFERENCES DEPARTMENT
          ON DELETE NO ACTION;

ALTER TABLE DEPARTMENT
      ADD CONSTRAINT ED
          FOREIGN KEY (MGRNO)
          REFERENCES EMPLOYEE
          ON DELETE SET NULL;
```

Pk catlog info.

Primary Key and **Unique Key** constraints are recorded in two catalog tables:

1 **SYSIBM.SYSTABCONST**

CONSTNAME	TBCREATOR	TBNAME	TYPE	IXNAME	COLCOUNT
PKDEPT	KIDDJA	DEPT	P	XDEPT	1
UKNI_NUMBER	KIDDJA	EMP	U	XNI_NUMBER	1

2 **SYSIBM.SYSKEYCOLUSE**

CONSTNAME	TBCREATOR	TBNAME	COLNAME	COLNO	COLSEQ
PKDEPT	KIDDJA	DEPT	DEPTNO	1	1

- SYSIBM.SYSTABCONST contains one row for each primary key or unique key

constraint.

- SYSIBM.SYSKEYCOLUSE contains one row for every column in a primary key or unique key constraint from the SYSIBM.SYSTABCONST table.

FK catlog info.

reign Key constraints are recorded in two catalog tables:

1 **SYSIBM.SYSRELS**

TBNAME	CREATOR	RELNAME	REFTBNAME	COLCOUNT	DELETERULE
EMP	KIDDJA	RED	DEPT	1	N

2 **SYSIBM.SYSFOREIGNKEYS**

TBNAME	CREATOR	RELNAME	COLNAME	COLNO	COLSEQ
EMP	KIDDJA	RED	WORKDEPT	6	1

her details of RI definitions are recorded in:

– **SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS and SYSIBM.SYSINDEXES**

© Copyright IBM Corporation 2011

- SYSIBM.SYSRELS contains one row for every foreign key constraint.

- SYSIBM.SYSFOREIGNKEYS contains a row for every column of every foreign key. It includes the name of the constraint for which the column is part of the foreign key.

Table Check Constraints

```
CREATE TABLE EMPLOYEE
(EMPNO       INTEGER  NOT NULL,
 NAME        CHAR(10) NOT NULL,
 DEPT        INTEGER  CHECK (DEPT BETWEEN 1 AND 100),
 JOB         CHAR(10),
 YEARS       INTEGER,
 SALARY      DECIMAL(10,2),
 INVESTP     INTEGER NOT NULL WITH DEFAULT,
 MAXINVEST   INTEGER NOT NULL WITH DEFAULT,
 CHECK    (JOB IN ('MANAGER','SALES','TECHNICAL')),
 CONSTRAINT  NOINVEST CHECK ((YEARS < 5 AND INVESTP = 0)
                             OR (YEARS >= 5)),
 CHECK    (INVESTP <= MAXINVEST))

 WITH RESTRICT ON DROP
 IN USDB1.USTS2;
```

WITH RESTRICT ON DROP:

- Can't drop table or database or table space that contains the table can't be dropped.
- Restrict status is recorded in CLUSTERTYPE column of SYSIBM.SYSTABLES.

Catlog info :SYSIBM.SYSCHECKS describes the check constraints.

SYSIBM.SYSCHECKDEP has a cross-reference of the constraints and the columns they refer to.

SYSIBM.SYSCHECKS

```
---------+---------+---------+---------+---------+---------+---------+--
SELECT TBNAME,OBID,TBOWNER,CHECKNAME, CHECKCONDITION
FROM   SYSIBM.SYSCHECKS
WHERE  TBOWNER    = 'TSOUD01' AND
       TBNAME = 'EMPLOYEE'
---------+---------+---------+---------+---------+---------+---------+--------
TBNAME   OBID  TBOWNER  CHECKNAME CHECKCONDITION
---------+---------+---------+---------+---------+---------+---------+--------
EMPLOYEE    9  TSOUD01  DEPT      DEPT BETWEEN 1 AND 100
EMPLOYEE   12  TSOUD01  INVESTP   INVESTP <= MAXINVEST
EMPLOYEE   10  TSOUD01  JOB       JOB IN ('MANAGER','SALES','TECHNICAL')
EMPLOYEE   11  TSOUD01  NOINVEST  (YEARS < 5 AND INVESTP = 0) OR  (YEARS >= 5)
```

SYSIBM.SYSCHECKDEP

```
---------+---------+---------+---------+---------+-
SELECT TBOWNER,TBNAME,CHECKNAME,COLNAME
FROM   SYSIBM.SYSCHECKDEP
WHERE  TBOWNER    = 'TSOUD01' AND
       TBNAME = 'EMPLOYEE'
---------+---------+---------+---------+---------+-
TBOWNER    TBNAME     CHECKNAME  COLNAME
---------+---------+---------+---------+---------+-
TSOUD01    EMPLOYEE   DEPT       DEPT
TSOUD01    EMPLOYEE   INVESTP    INVESTP
TSOUD01    EMPLOYEE   INVESTP    MAXINVEST
TSOUD01    EMPLOYEE   JOB        JOB
TSOUD01    EMPLOYEE   NOINVEST   INVESTP
TSOUD01    EMPLOYEE   NOINVEST   YEARS
```

© Copyright IBM Corporation 2011

Examples for adding / removing check constraints

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT SALCHECK
CHECK(SALARY > 0);
--
ALTER TABLE EMPLOYEE
DROP CONSTRAINT SALCHECK;
```

```
---------+---------+---------+---------+---------+---------+-----
SELECT TBNAME||','||CHECKNAME||' || '||CHECKCONDITION
FROM SYSIBM.SYSCHECKS;
---------+---------+---------+---------+---------+---------+-----

---------+---------+---------+---------+---------+---------+-----
IPNAMES,DSNIPN_SEC_OUT_C || SECURITY_OUT IN('A','P','R','D','E')
IPNAMES,DSNIPN_USERNAMES_C || USERNAMES IN (' ','O','S')
LUNAMES,DSNLUN_ENCRPSWDS_C || ENCRYPTPSWDS IN('Y','N')
LUNAMES,DSNLUN_GENERIC_C || GENERIC IN('Y','N')
LUNAMES,DSNLUN_MODESEL_C || MODESELECT IN('Y','N')
LUNAMES,DSNLUN_SEC_IN_C || SECURITY_IN IN('A','V')
LUNAMES,DSNLUN_SEC_OUT_C || (SECURITY_OUT IN('A','P','R')) AND
LUNAMES,DSNLUN_USERNAMES_C || USERNAMES IN(' ','B','I','O')
SYSCOLDIST,DSNDIS_CARDF || CARDF = -1 OR CARDF >= 0
```

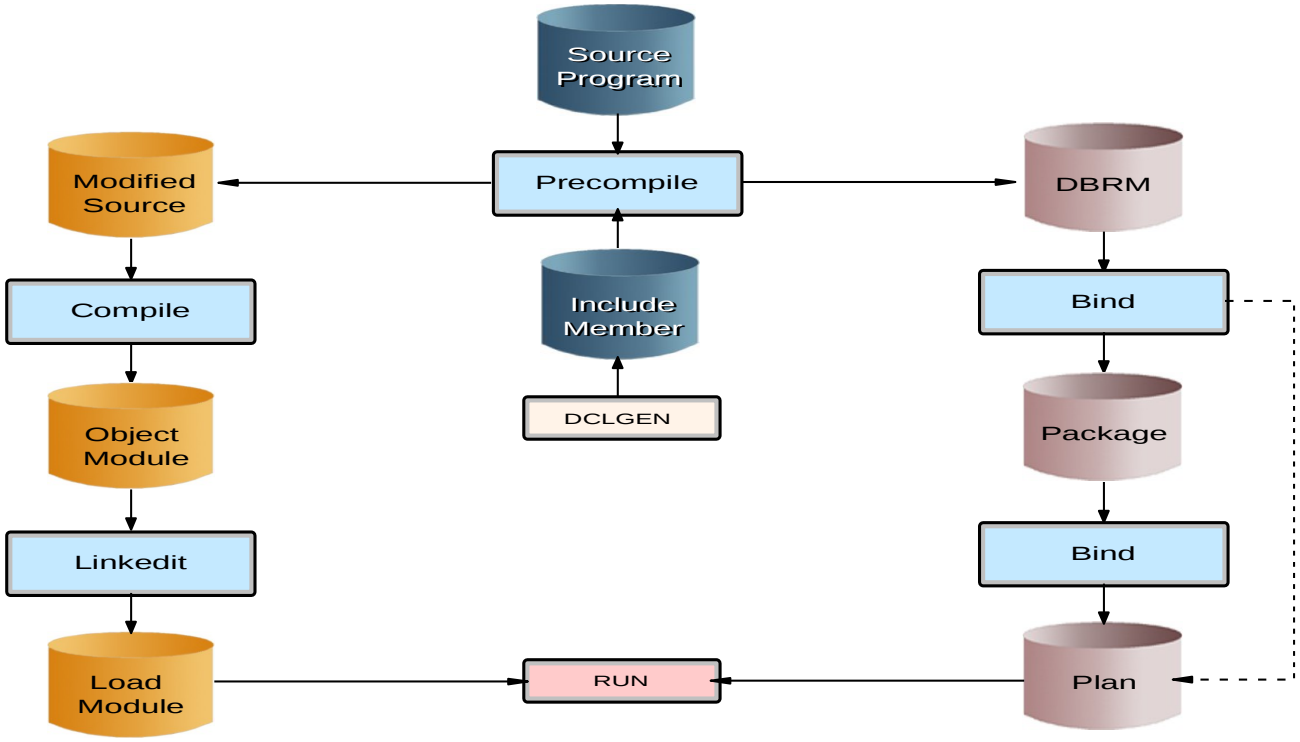
one row for each check constraint for catalog tables created in or after Version 7.

```
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT TBNAME||'      '||CHECKNAME                                    00000100
FROM SYSIBM.SYSCHECKS2;                                               00000200
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
DSN_VIRTUAL_INDEXES      CLUSTERING
DSN_VIRTUAL_INDEXES      CLUSTERRATIOF
DSN_VIRTUAL_INDEXES      COLCOUNT
DSN_VIRTUAL_INDEXES      COLNO1
DSN_VIRTUAL_INDEXES      COLNO10
DSN_VIRTUAL_INDEXES      COLNO11
```

4. SQL – Preprocess

1. Sql – Preprocess or Binding



Source Program: When we write a program that needs data from DB2 tables, we embed SQL in the program by surrounding it with delimiters. The delimiters for COBOL are EXEC SQL and END-EXEC. For example

```
EXEC SQL

    DECLARE EMPCURS CURSOR
      FOR SELECT LNAME, FNAME, PAYRATE, HOURS
      FROM EMPLOYEE WHERE DEPT = :DEPT-NUM
END-EXEC
```

Each EXEC SQL ... END-EXEC block must contain only one SQL statement. If the SQL command is not terminated by an END-EXEC operand, a COBOL compiler error will be generated.

DB2 Precompiler: DB2 precompiler seperate pure cobol code and sql statements

and create DBRM(database request module) with that sql statements.

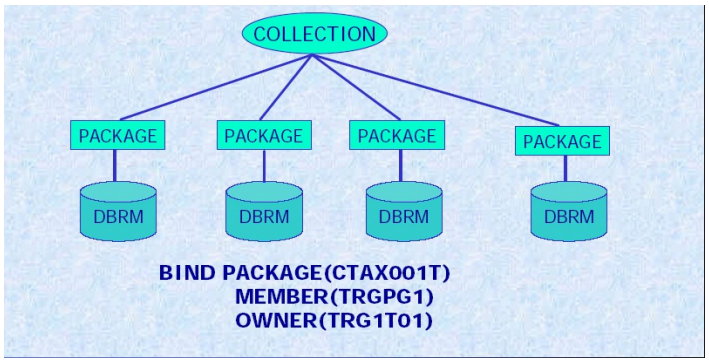
DBRM: a PDS member which contain only raw sql statements. DBRMs will be copied into DB2 catlog for future Rebind process.

Bind: here every sql statement in DBRM is processed and access path is determined which is refered as called package. In this process there are 3 steps will be happend

- Check authorization
- Check syntax
- Optimization (Load module & access path will be generated)

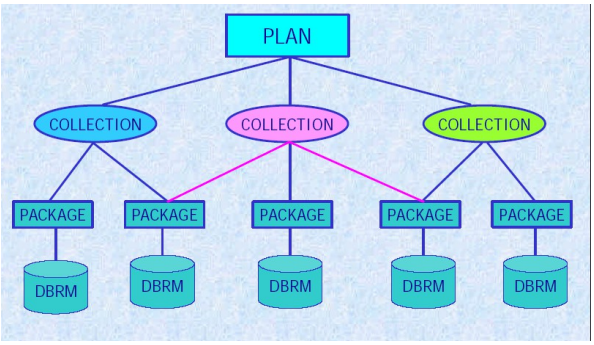
package is written into DB2 directory

Collection: many packages which as common characteristich and which are part of one or more plan are logically grouped through common ID (collection ID) that group is called collection. Maybe one group of PACKAGEs is used by the payroll application. Maybe another group is very popular and called by every program regardless of application area.

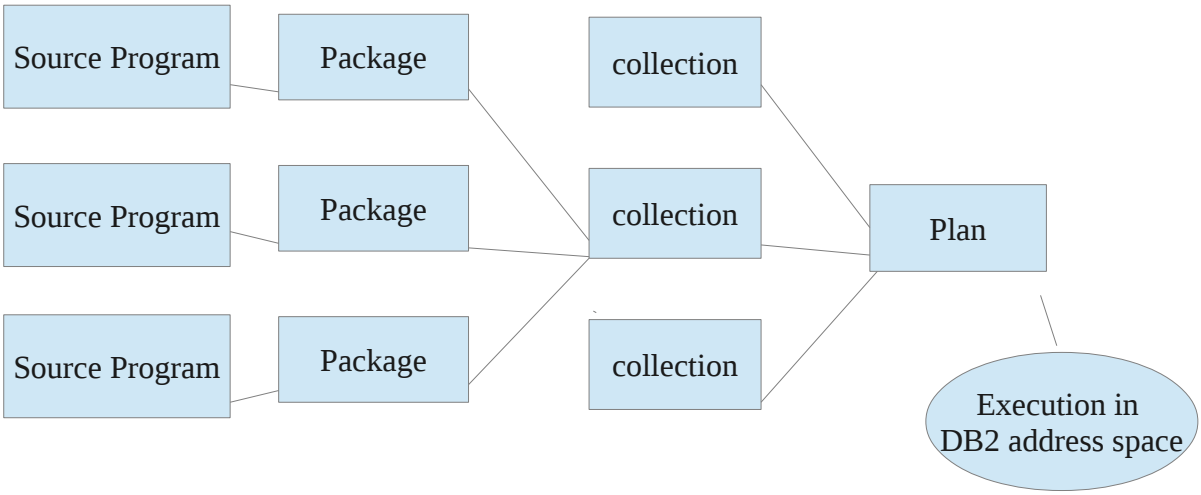


Plan: Set of collection/packages make into plan. List of collections which are ready to search are put into plan by doing bind process second time (BIND PLAN). Plan give info to cobol program how/where to search its twin module in

collections of plan which we gave. If it does it found it generate error 805 SQLCODE. Plan info is places in DB2 catlog tables.



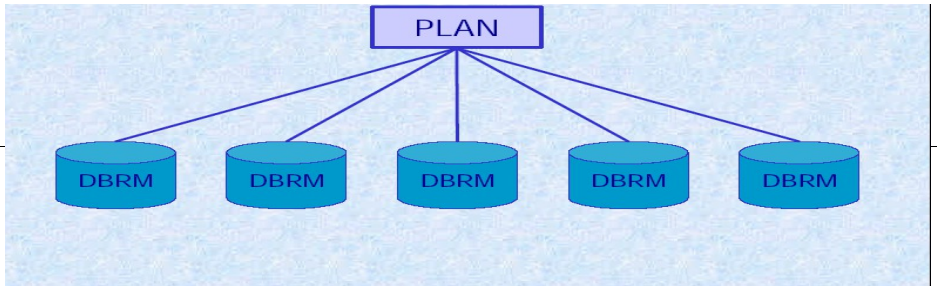
Versions: Sometimes we would like to keep multiple VERSIONs of the same PACKAGE. For example, in a fallback situation, it would be nice to have both the current VERSION and the prior VERSION of a PACKAGE available.



---------- Binding ----------
----------Execution----------

Should I use packages/collections or bind directly to the plan?

Previous version does it contain concept of packages and collections



- When you change one SQL statement, you do not need to bind the entire plan again. You need to bind only the package that is associated with the changed SQL statement.
- Binding packages into collections enables you to add packages to an existing application plan without having to bind the entire plan again.
- Maintaining several versions of a plan without using packages requires a separate plan for each version.
- *You cannot bind or rebind a package or a plan while it is running. However, you can bind a different version of a package that is running & you can add package & and you can update package this prevents database outage.*

2. what is DB2 Optimisor

- The DB2 Optimizer calculates low & risk less access path from all posible path for data retrival. Here low means less nuber of CPU & I/O cycles.
- considers all of this information about each possible path and determines cost in terms of I/O, CPU, RID SORT, and DATA SORT overhead.

3. what is rebinding

Rebinding is the process of recreating a package

4. skeleton cursor table(SKCT),skeleton package (SKPT),cursor table(CT) & cursor table(CT)?

- It is a table of DB2 catlog at which plan stores. Single SKCT may longer than max record length. The entire SKCT section is stored in single record if it fits; if it does not fit, it is stored in more than one record records. In v7 5 tables are given to store plan content

- Similarly the table of DB2 catlog at which package stores is called skeleton package table(SKPT) in v7 8 tables is given to package and stored PROC content

5. what is host varibles

Host variables are the means by which the program and DB2 pass data back and forth. always start with colon':' DCLGEN is a declaration generator used to generate DB2 equivalent COBOL variables called Host variables.

6. What is SQLCA

The SQLCA (SQL Communications Area) is set of control blocks that provides return codes and other important information to the program, after each SQL statement is executed.

Return code	RESULT	Char (5) Sql state
<0	Error	>02nnn
0	sucess	00000
>0 and < 100	warnings	01nnn
100+	Data not found	02nnn

7. What are the Pools and Why are They Important?

There are four pools used by DB2 address space to cache information in memory

as it operates. The longer information can be cached into memory, the better the chance the data can be reused by other processes without having to read it from disk again.

- Buffer pool
- EDM pool
- RID pool
- Sort pool

EDM pool:– Enviormental Descriptor Manager pool

- The EDM pool is used by DB2 to control programs as they execute against DB2. For that it will caches info about access paths of the SQL statements for running programs.

- DB2 also permits dynamic SQL prepare information to be cached in the EDM pool to optimize dynamic SQL.

- The EDM pool also contains database information (DBDs) for the databases being accessed.

- Managing the size and efficiency of the EDM pool is vitally important because if there is no room in the EDM pool, a critical application may not be allowed to run.

- It contain statistics

  - You can view skeleton cursor table (SKCT) storage statistics.

  - You can view skeleton package table (SKPT) storage statistics

  - You can view all the active DB2 threads that have cursor table storage allocated.

- You can view information about the size and percentage of cursor table storage being used for each thread.
- You can view the packages that are using package table (PT) storage for the selected thread.

RID pool: - Record identifier pool

- sort record identifiers

Sort pool:

- used to sort data

CURSORS

- A query that returns a maximum of one row in the results set is known as a singleton SELECT.
- it is possible for the SELECT to return multiple rows in the results set, another programming technique is required, called cursor processing. It is comprised of the following four steps.
  - DECLARE cursor
  - OPEN cursor
  - FETCH the rows from stack
  - CLOSE the cursor
- Each fetch moves the cursor down one row in the stack.
- Scrollable cursors allow us to fetch forwards, backwards, using a relative

number forwards & backwards, or using an absolute number. Scrollable cursors are currently only available in DB2 for z/OS.

Static SQL	Dynamic SQL
sql statements are hard-coded/embeded in a program by the programmer before compile & run	The program must dynamically allocate memory to receive the queries.the program contains some presentation logic to take an SQL statement from the user. (Through intractive sql or using jdbc or odbc functions)
Static sql is compiled and optimized prior to its execution	dynamic is compiled and optimized during execution.
Db2 system aware of binding time	Db2 system does't aware of binding time
here authorization check at every step then complete that step. If he does;t have authorization to bind plan. It will stop their after binding package	it scan authorization all steps before start of binding then start binding process.If he does;t start binding at all

5. System Objects

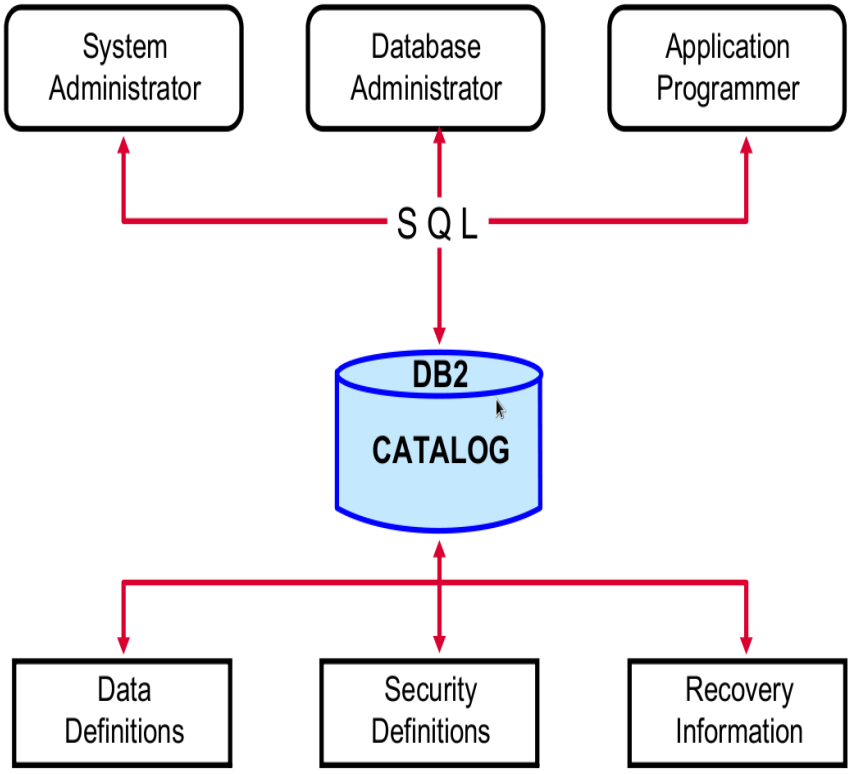
DB2 System Structures or Objects

DB2 has a comprehensive infrastructure that enables it to provide data integrity

- DB2 Directory database (DSNDB01)
- DB2 Catlog database (DSNDB0C)
- DB2 Work file database (DSNDB07)
- DB2 Active & Archive logs
- Bootstap dataset
- Buffer pools

5.1 DB2 Catlog database (DSNDB06)

- The catalog tables has entities for all things. things refers storage groups, tablespaces, tables, indexapces, indexes, columns, keys, procedures, packages, plans, sql statement in package & plans, privileges for user to query.....
- All catalog tables are qualified by SYSIBM.XXXXXX , so Do not use this qualifier for user-defined tables.
- The catalog tables are updated by DB2 during normal operations in response to certain SQL statements, commands, and utilities. below are some of catlog tables in db2



Catlog Tables

1. Data Definations
   - Data structure definations
   - Constraints definations
   - Plan & Package Definations
2. Security definations
   - Authoritization Definations
   - PrevIlages Defination
3. Recovery Info
4. Communication database definations

Data structure definations

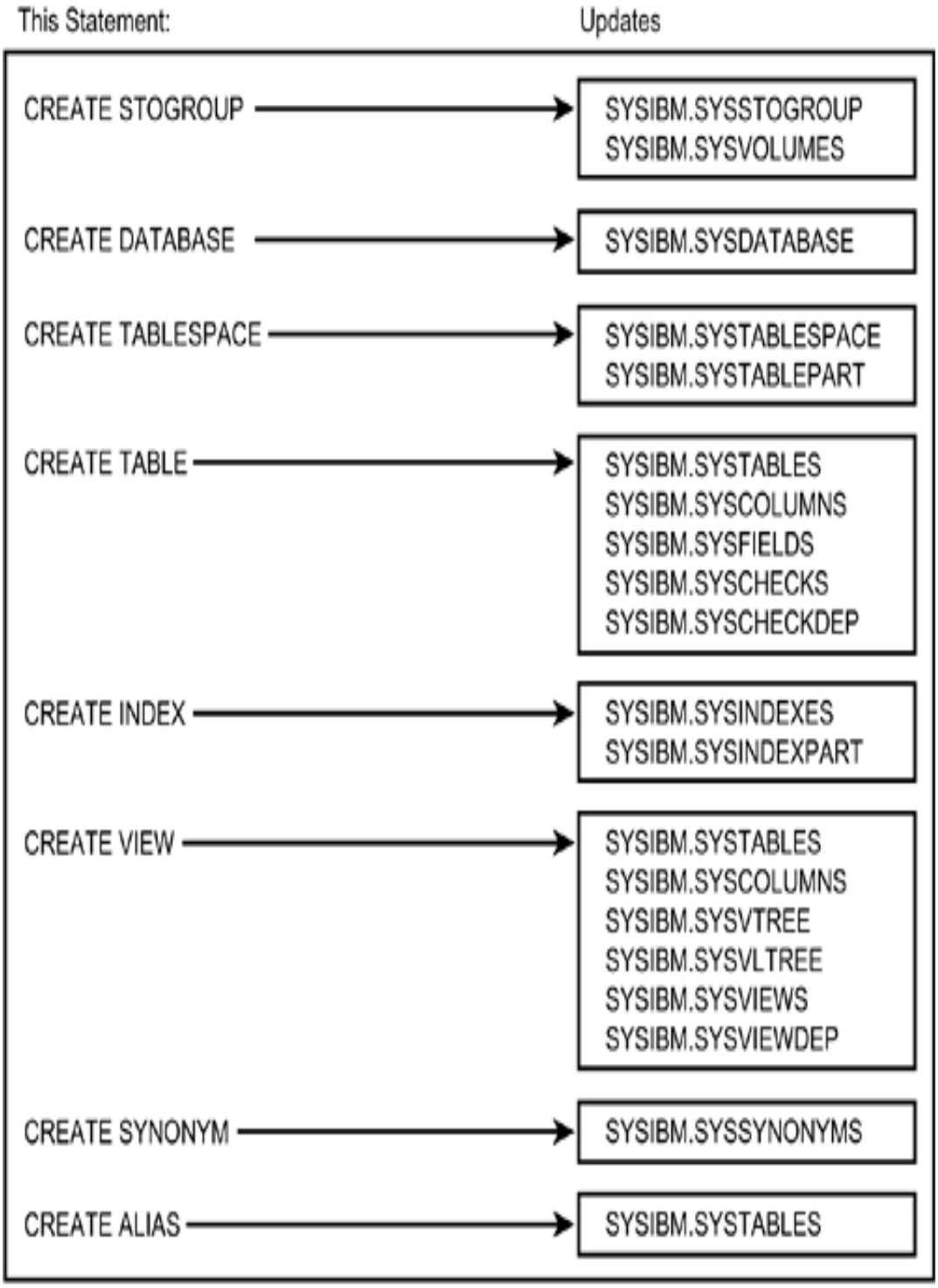


Table	Description
SYSVOLUMES	one row for each volume of each storage group. (Sgname,Sgcreator, Volid (* if SMS manged),....
SYSTOGROUP	contains one row for each storage group. (Name, Dataclas, Mgmtclas, Storclas, CreatedTs, Space, Vcat, Creator.....
SYSDATABASE	contains one row for each database, except for database DSNDB01.
SYSTABLESPACE	contains one row for each table space
SYSTABLEPART	contains one row for each nonpartitioned table space and one row for each partition of a partitioned table space.
SYSINDEXES	one row for every index.
SYSINDEXPART	one row for every index partition
SYSTABLES	one row for every table
SYSCOLUMNS	one row for every column

Constraints definations

Table	purpose
SYSKEYS	One row for one key
SYSFOREIGNKEYS	one row for every column of every foreign key

Plans & packages

Table	purpose
SYSIBM.SYSDBRM	one row for each DBRM of each application plan.
SYSIBM.SYSPACKAGE	One row for one package
SYSIBM.SYSSTMT	Sql statements bound in plan
SYSIBM.SYSPACKSTMT	Sql statements bound in package

5.2 DB2 Directory database (DSNDB01)

- Dirctory is a system database(DSNDB01) which contain set of directory tablespaces, which contain set of directory tables. which holds records of info which needs in DB2 normal operations.

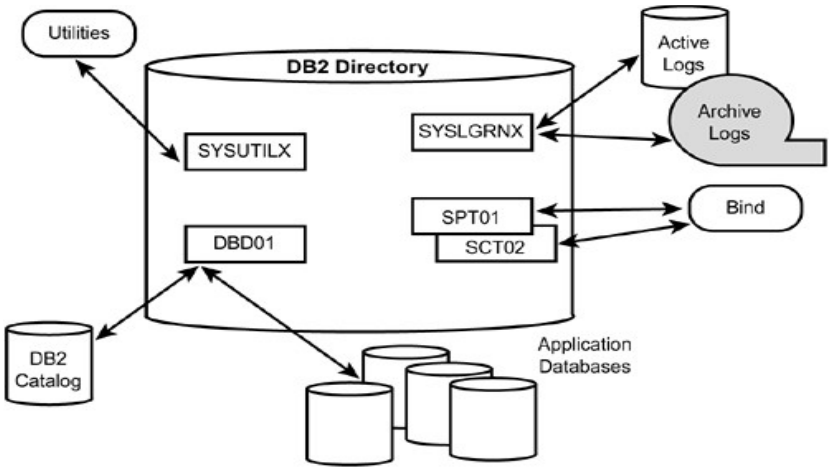


Table space	Content (Refer this table after SQL preprocess)
DB01	database descriptor information (one table)
SPT01	Stores skeleton cursor table information (one table)
SPT02	stores skeleton package table information (one table)
SYSUTILX	utility processing status (one table)
SYSLGRNX	recovery log range information (one table) Tracks the relative byte address (RBA) range of the logs for updates

5.3 DB2 Work file database (DSNDB07)

- it contain global temporary tables for processing SQL statements. Such as sort operations.
- DB2 creates a work file database and some table spaces in it for you at installation time. You can create additional work file table spaces at any time

5.4 Default database(DSNDB04)

- Default database in DB2 subsystem . Users can create DB2 objects in it .

5.5 Buffer pools & its Tunning (v11)

1. What is Bufferpool?

   The areas of virtual storage comes under DBM1 address space which temporarily stores the pages of table spaces or index spaces is called Bufferpools

2. Why Bufferpool?

   - The Purpose of Bufferpool is to reduce CPU & IO cycles (cumsuption)
   - When an application program accesses a row of a table, DB2 places the page that contains that row in a buffer. Then Access rows from that temporary storage which is faster than accessing data on a disk.
   - If the required data is already in a buffer, the application program does not need to wait for it to be retrieved from disk, so the time and cost of retrieving the page is reduced.

3. Relation B/W Table space & Buffer pool
4. What is Default Bufferpool?
5. What is Index Bufferpool?
6. What is Group Bufferpool?

   A special type of buffer pool that is used only in Parallel Sysplex data sharing by different db2 systems is called group buffer pool, which resides in the coupling facility.

7. Bufferpool Maximum Size?

The maximum size of a buffer pool is 1 TB.

8. Can we modify size of bufferpool? How?

Bufferpool Read Activity

9. What is Getpage? How Getpage different from IO request?

- A GETPAGE is a request by DB2 for the page it may be in bufferpool or in disk.
- GETPAGE counter will increment for every GETPAGE request.
- A GETPAGE is NOT equivalent to an I/O request.If DB2 issues a request to look at a particular page, and that page is already in one of the subsystem's buffer pools, then there is no need of IO for that page. In that senario GETPAGE counter is incremented by one and the I/O counter is not incremented.
- Getpages are reduced by writing more efficient SQL. I/O is decreased by increasing the size of your buffer pool.

10. How is Read I/O performed?

DB2 can schedule four different types of read I/O for SQL calls:

- Synchronous I/O of a single page, also called a random read
- Sequential prefetch of up to 32 (V8) or 64 (V9) contiguous 4 KB pages
- Dynamic prefetch of up to 32 contiguous 4 KB pages
- List prefetch of up to 32 contiguous or non-contiguous 4 KB pages

11. What & Why Prefetch?

- When synchronous read is used, just one page is retrieved at a time.
- Prefetch is a mechanism for reading a set of pages, usually 32, into the buffer pool with only one asynchronous I/O operation.
- Prefetch provides for substantial savings in both CPU and I/O costs.
- The maximum number of pages read by a single prefetch operation is determined by the size of the buffer pool that is used for the operation.
- Types of Prefetch
    - Sequential Prefetch
    - Dynamic Prefetch
    - List Prefetch

Sequential Prefetch

- Sequential prefetch of up to 32(V8) or 64(V9) contiguous 4 KB pages
- Sequential prefetch is used only for table space scans.
- DB2 uses Sequential Prefetch when Indexes are not available.
- When DB2 expects that sequential prefetch will probably be used, the value of the PREFETCH column is 'S.' However, DB2 might use dynamic prefetch in some cases if when the value of PREFETCH column is 'S.

Dynamic Prefetch

- Dynamic prefetch of up to 32 contiguous 4 KB pages.
- In dynamic prefetch, DB2 uses a sequential detection algorithm to determine whether data pages are being read sequentially.
- DB2 cannot detect whether the pages can be accessed sequentially (because the catalog statistics might not always be correct).

- When DB2 chooses dynamic prefetch, the value of the PREFETCH column is normally set to 'D.' However, DB2 might also use dynamic prefetch when the value of PREFETCH indicates 'S.'

List Prefetch

- The maximum number of pages that can be retrieved in a single list prefetch is 32 (64 for utilities).
- The value of the PREFETCH column is set to 'L' or 'U'
- DB2 uses list prefetch to read a set of data pages that is determined by a list of record identifiers (RIDs) from an index or from DB2 log.

12. What is Bufferpool Hit ratio?

Buffer pool hit ratio is a measure of how often a page access (a getpage) is satisfied without requiring an I/O operation.

Bufferpool Write Activity

13. Types of Pages in Bufferpool?

14. Bufferpool Thresholds?

Deferred write threshold (DWQT):

- This threshold is a percentage of the buffer pool that might be occupied by unavailable pages(updated pages & in-use pages).
- The default value for this threshold is 30%. You can change that to any value from 0% to 90% by using the DWQT option on the ALTER BUFFERPOOL command.
- DB2 checks this threshold when a page update is completed.
- If Threshold is reached then it sart asynchronous write to decrease the number of unavailable pages to 10% below the threshold. For example, if the threshold is 50%, the number of unavailable buffers is reduced to 40%.

15. When DB2 performs Bufferpool Write activity

Updated pages are queued by data set until they are written when one of the following events occurs:

- Once Checkpoint is taken
- The percentage of updated pages in a buffer pool for a single data set exceeds a preset limit called the Vertical Deferred Write Threshold (VDWQT)
- The percentage of unavailable pages in a buffer pool exceeds a preset limit called the Deferred Write Threshold (DWQT).

16. Number of Pages that DB2 can do write activity by single IO cycle

- The pages in the bufferpool, which are updated with data but are not written onto the disk, are called “Dirty” pages. After the updated data pages in the bufferpool are written on the disk, the bufferpool is ready to take another data.
- One tablespace is associated with one bufferpool. The size of bufferpool and tablespace must be same.
- The size of the bufferpool page is set when you use the “CREATE DATABASE” command. If you do not specify the page size, it will take default page size, which is 4KB. Once the bufferpool is created, it is not possible to modify the page size later

3. Dirty Pages means what ?
4. What is page size & its posibilities
5. When default bufferpool is created? Its Name?
6. Size of Default bufferpool?
7. Group Buffer pool?
8. Maximum size of bufferpool
9. Can we modify size of bufferpool? How?
10. Relation B/W Table space & Buffer pool

5.6 DB2 Active & Archive logs

1. What are /why Logs?
   - Logs are system object which is uses for recovery the data from a system errors or disastrous conditions.
   - Physically they are linear VASM (active logs) & sequential (archive logs) datasets where Data changes and other significant events stored in terms of records called log records

2. What is /Why Log Output Buffer? Its size?
   - Initially DB2 writes logs in real storage area called Log output buffers.
   - So that DB2 can get recent log records from Log output Buffer with out any IO.
   - So it is better to maintain Large Log Output Buffer as must as possible.
   - The OUTPUT BUFFER field sets the OUTBUFF subsystem parameter, which specifies the size of the output buffer

     Acceptable values: 409600 to 409600000, 400K to 400000K, or 1M to 390M
     Default: 4000K

3. What is /Why Log Active dataset? Its size? Its number?
   - Log Output Buffer is a teporary storage for log records, it further moved in to VSAM Linear Datasets called <u>Active logs</u>
   - we can have to maintain 2-93 active log datasets and we can maintain dual active logging (we can't maintain more than dual logging)

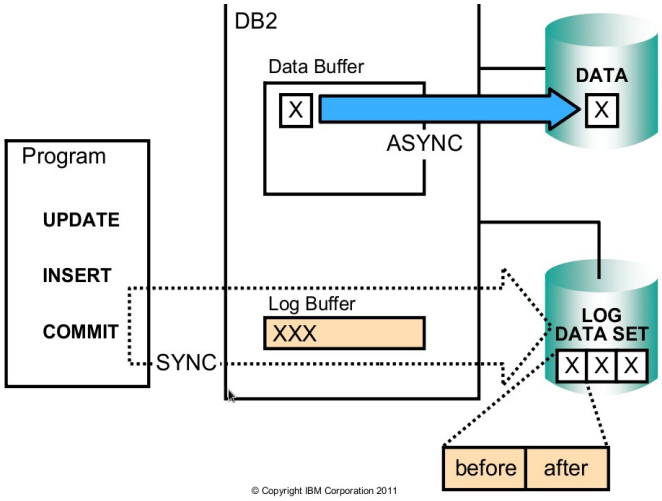
- Max log size: 4GB – 1byte

4. What is /Why Log archive dataset?
   - it maintain certain number of active logs, once one active log is fill it take its backup into disk or tape dataset called archive logs

5. Give Block view for Log Write activity?

Log write activity includes 3 steps as follows

1. Log writes to Output Buffer
2. Output Buffer to Active Log dataset
3. Active logs to Archive log dataset



Log writes are divided into 2 categories: asynchronous & synchronous.

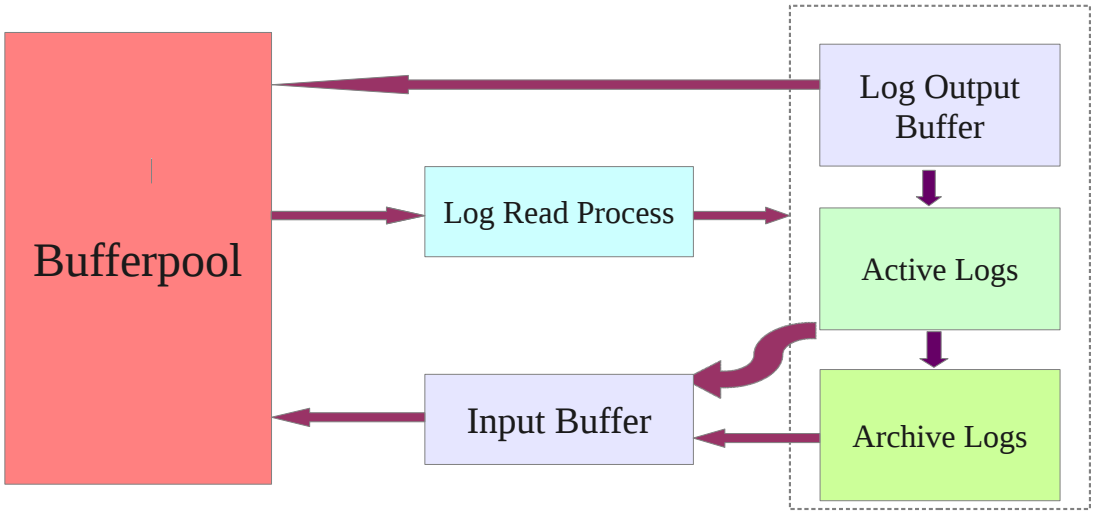
Asynchronous writes	Synchronous writes
This is most common. writes occur when data is changed. Before, and after, image records are usually moved to the log output buffer, and control is returned to the	writes usually occur at commit time when an application has changed data. This write is called 'forcing' the log because the application must wait for DB2 to force the

application.	log buffers to disk before control is returned to the application.
However, if no log buffer is available, the application must wait for one to become available.	If the log data set is not busy, all log buffers are written to disk.If the log data set is busy, the requests are queued until it is freed.

6. Give Block view for Log Read activity?

DB2 requires Log records during Rollback or Recovery. It searches for records in following order

1. Output buffer
2. Active log data set
3. Archive log data set



If the log records are in the output buffer, DB2 reads the records directly from that buffer. If the log records are in the active/archive log, DB2 moves those records into the input buffer.

7. What is Input log Buffer? Its Size? Its Purpose?

- Upon a change request or a system event, These log output buffers are externalized (written to DASD) to **active logs** when buffers are full or when a transaction is committed.

  DB2 writes to the output buffer area whenever an update, commit, rollback, or checkpoint occurs. DB2 reads into the input buffer area during rollback, recovers utility jobs, and when it restarts.
- A commit from one UR (Unit of Recovery) writes ALL log records that haven't yet been written from the output buffer. This includes even records from other URs. Transactions typically trigger this mechanism.
- When the WRITE THRESHOLD in the log output buffer area is reached, a write commences. A batch job that does not commit might trigger this.
- Log Write Ahead Force (LWAF) is a fundamental DB2 protocol: When the Buffer Manager wants to externalize a page to page set DASD, it must first force any log descriptions of the updates to log DASD (if they have not already been written).

2. What is log output Buffer? Why it is required

   Physically it resides in Real storage.

   So, The larger the output buffer, the more likely that a log read request can return the required log records without needing I/O operations.

3. Log Output Buffer size?

4.

5. Explain Log write activity? And its types?

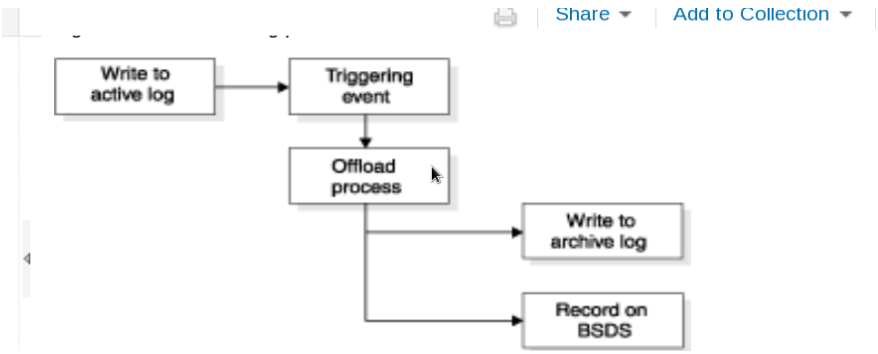
6.

7. When logs are afloaded from Buffer to Active logs

   DB2 also writes the log buffers to an active log data set when they become full, or when the write threshold is reached.

8. When logs are afloaded from Active logs to Archive logs

   The process of copying active logs to archive logs is called offloading.



- When all active logs become full, the DB2 subsystem runs an offload and halts processing until the offload is completed.

- DB2 determines which data set to offload. Using the last log relative byte address (RBA) that was offloaded, as registered in the BSDS.
- DB2 also determines the log RBA at which to end, from the RBA of the last log record in the data set, and registers that RBA in the BSDS.
- Offloading will also occur when we issue ARCHIVE LOG command.

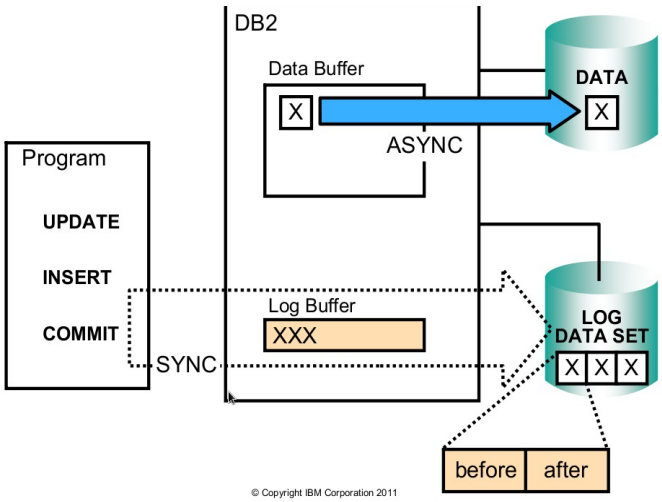
9. Explain RBA, LRSN
10. What is check point
11. What is one-phase commit and two-phase commit

→ Is Log buffer is part of master address space

→

- DB2 records all data changes and other significant events in a as records in on disk dataset (VSAM linear datasets) called active logs
- we can have to maintain 2-93 active log datasets and we can maintain dual active logging (not more than two)
- it maintain certain number of active logs, once one active log is fill it take its backup into disk or tape dataset called archive logs

- Max log size: 4GB – 1byte



Log write operations

a. Log write in Output Buffer

Log writes through output buffer are divided into two categories: asynchronous and synchronous.

Asynchronous writes

- Asynchronous writes are the most common. These asynchronous writes occur when data is updated. Before, and after, image records are usually moved to the log output buffer, and control is returned to the application.

- However, if no log buffer is available, the application must wait for one to become available.

Synchronous writes

- Synchronous writes usually occur at commit time when an application has updated data. This write is called 'forcing' the log because the application must wait for DB2 to force the log buffers to disk before control is returned to the application. If the log data set is not busy, all log buffers are written to disk. If the log data set is busy, the requests are queued until it is freed.

→ The larger the output buffer, the more likely that a log read request can return the required log records without needing I/O operations.

→ The OUTPUT BUFFER field sets the OUTBUFF subsystem parameter, which specifies the size of the output buffer

Acceptable values: 409600 to 409600000, 400K to 400000K, or 1M to 390M

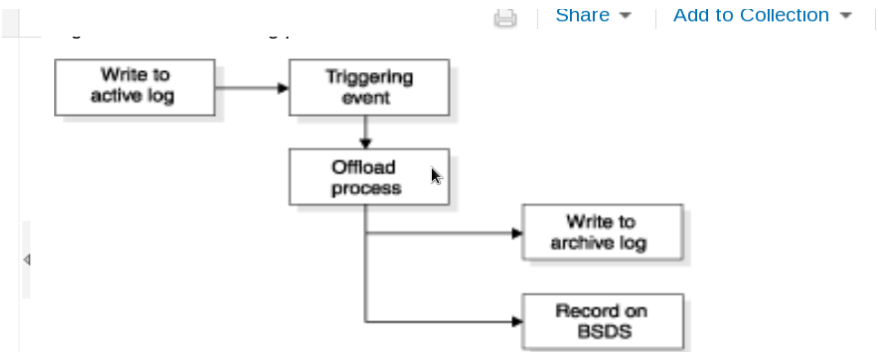
Default: 4000K

b. Output Buffer to Active logs

- DB2 also writes the log buffers to an active log data set when they become full, or when the write threshold is reached.

c. Active logs to Archive logs

The process of copying active logs to archive logs is called offloading.



- When all active logs become full, the DB2 subsystem runs an offload and halts processing until the offload is completed.
- DB2 determines which data set to offload. Using the last log relative byte address (RBA) that was offloaded, as registered in the BSDS.
- DB2 also determines the log RBA at which to end, from the RBA of the last log record in the data set, and registers that RBA in the BSDS.
- Offloading will also occur when we issue ARCHIVE LOG command.

Log Read operations

DB2 requires Log records during Rollback or Recovery. It searches for records in

following order

1. Output buffer
2. Active log data set
3. Archive log data set

- If the log records are in the output buffer, DB2 reads the records directly from that buffer.

If the log records are in the active/archive log, DB2 moves those records into the input buffer

5.7 Bootstap dataset

- The bootstrap data set (BSDS) is a VSAM key-sequenced data set (KSDS).

This contains info that is critical to DB2. Specifically, the BSDS contains:

- An inventory of all active and archive log data sets that are known to DB2.
- A wrap-around inventory of all recent DB2 checkpoint activity.
- The distributed data facility (DDF) communication record, which contains info that is necessary to use DB2 as a distributed server or requester.
- Information about buffer pools.

Because the BSDS is essential to recovery in the event of subsystem failure, during installation DB2 automatically creates two copies of the BSDS and, if space permits, places them on separate volumes.

6. Concurrancy control

6.1 Locks for Concurancy control

- Concurrency is the process of allowing multiple users or applications to use the same data.

- one user or program does not change data while another is in the process of using or changing that data,a situation that can lead to errors or inconsistencies in the data.

- DB2 manages concurrency with several types of locks that restrict access to data while the data is being used.The IRLM (Internal Resource Lock Manager) controls all locks.

Practical Example:

Venkatadhri have a 10000/- checking account balance. he withdraws 2000/- at 10:01:01.001 A.M. At precisely that instant plus a millisecond, before her transaction completes and the new 5000/- balance is entered. His deduction is taken from the still-intact 10,000/-, and a new balance of 8000/- is written to the account, overwriting 10000+5000 =15000/- . The bank lost 2000/-

6.2 DB2 serialization mechanisms

1. Transaction Locking
2. Restricted states
3. Claims & Drains
4. Latching

Transaction Locking

- Heart of concurancy control. There are 3 charateristices of Transaction Locking

1. Lock size
2. Lock mode
3. Lock Duration

Restricted states

TheseRestricted states can be set on DB2 objects by a DB2 utility or a DB2 command. this states influene the database objects(Tablespace, Partion of Tablespace, IndexSpace, Indexspace partition) when utilty is processing on it.

1. UTUT - Serialized for utility access and exclusive control
2. UTRO - Serialized for utility access and available for read-only access
3. UTRW - Serialized for utility access; read and write access allowed
4. UT-Utility processing only

Claims & Drains

Claims and drains enable DB2 utilities and commands to take over access to some objects independently of any transaction locks held on the objects.

Claims and drains apply to the following types of objects:

- Table spaces that are not partitioned.
- Data partitions.
- Indexes that are not partitioned.
- Index partitions.

Note: Transaction Locks: Used to control access by SQL Statement

Drain Locks: Used to control access by Utilities & DB2 commands

Latching

- Latches are internal DB2 locks used to serialize access to shared data structures.
- A true lock is handled by DB2 using the IRLM. However, whenever doing so is practical, DB2 tries to lock pages without going to the IRLM. This type of lock is called a latch .
- Duration of Latch is very less than that of lock. a latch requires about one-third the number of instructions as a lock.So Latches are used when a resource serialization situation is required for a short time and we are using heavily shared resources are used.

6.3 Lock Size

- DB2 for z/OS supports LOCKSIZE ROW, PAGE, TABLE, or TABLESPACE.
- DB2 for Linux, UNIX, and Windows supports ROW or TABLE level locking.
- The same piece of data can be controlled by locks of different sizes.
- Defining LOCKSIZE is a straight-forward task, done as a parameter on a CREATE or ALTER TABLESPACE statement on z/OS. There are some differences in the type of locking supported on the various DB2 platforms.

ANY

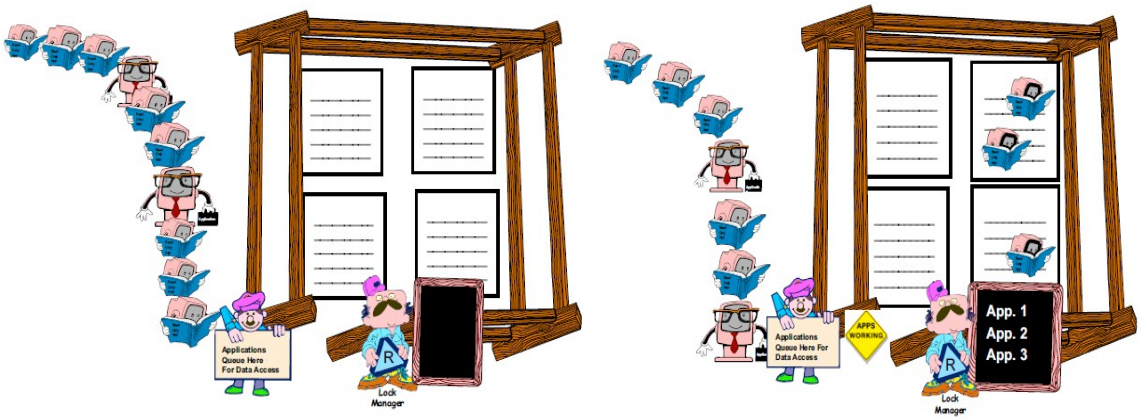
ANY permits DB2 to make the final choice and DB2 favours page locking as a good

compromise between high concurrency and high CPU consumption.

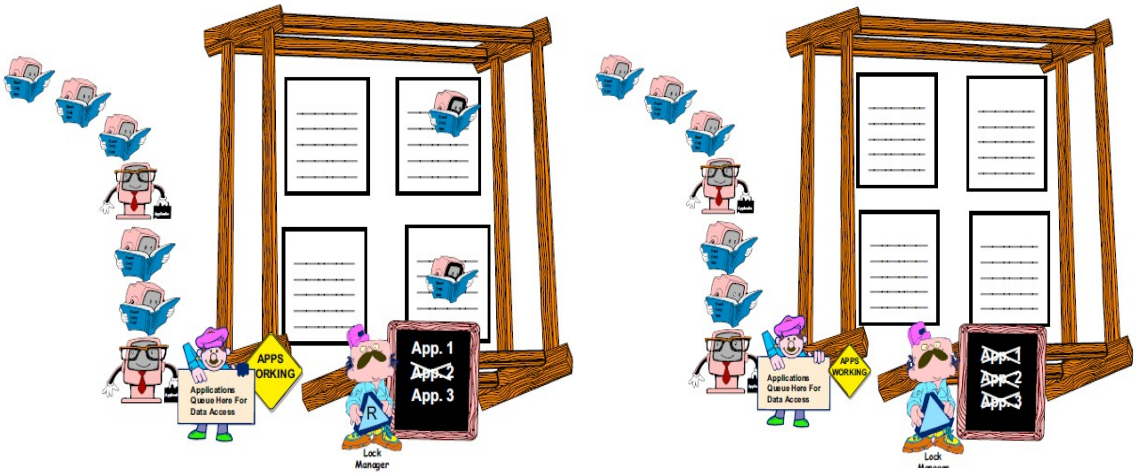
Concurrency – Table level

1. many apps wnat to access table which are in queue

2. the first three applications in the queue wanted to read only, so DB2 allows them access to the table. The fourth application wanted to make changes to the table, so it and the applications behind it in the queue must wait until all the readers finish their work and leave the table.

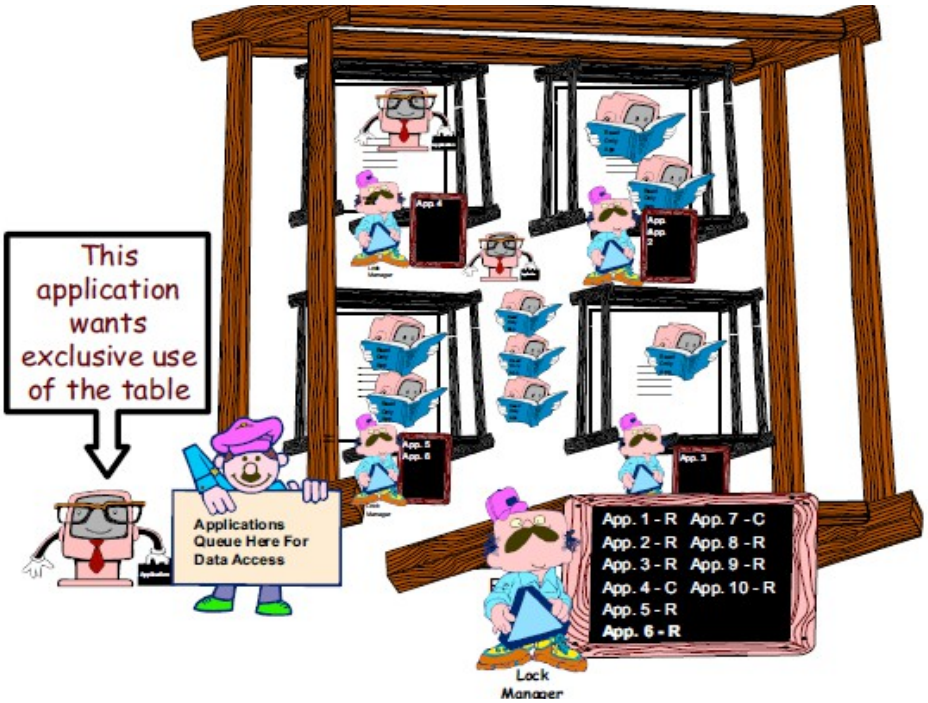


3. As each of the reading applications finishes its work and leaves the table, the Lock Manager crosses it off the blackboard. When the last read application leaves, allows 4th app which want to change data



- this process contionous.....

Concurrency – Page level – z/OS



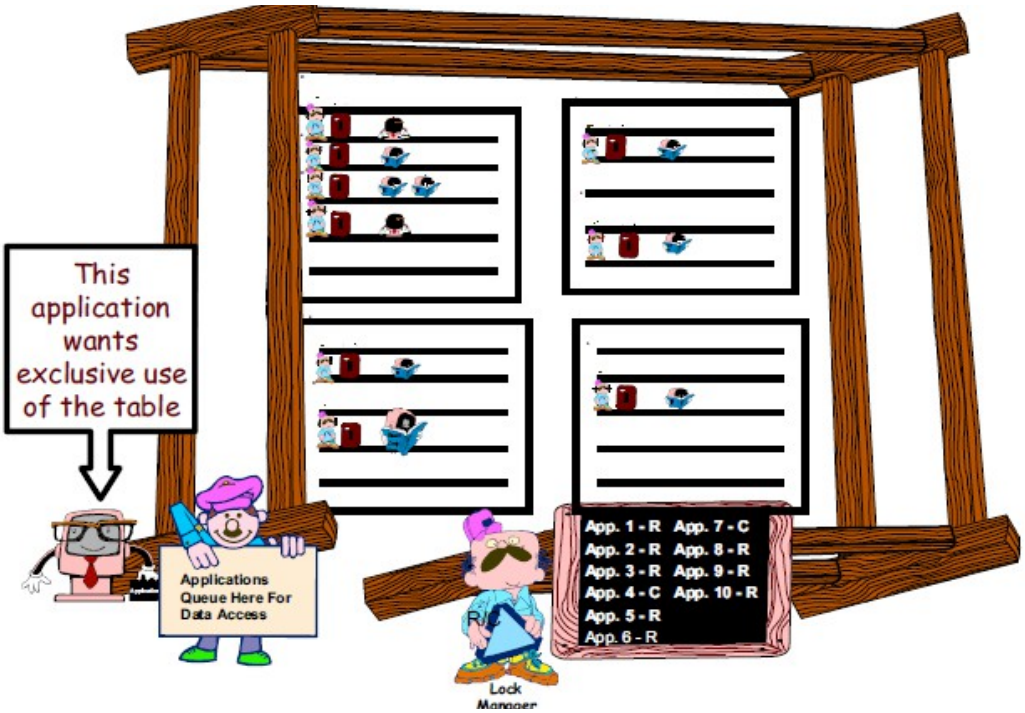
- Suppose that, instead of choosing TABLE locking, the DBA chose page locking. Page locking is like hiring a lock enforcer for the table, along with a lock enforcer for each page.

- Again, all applications queue up for access. The lock enforcer for the table sets the sign for both reading and change activity to occur concurrently within the table, so long as all applications are willing to work together.

- In this example, the first 10 applications in the queue are willing to work concurrently.

- The 11th application is demanding EXCLUSIVE access to the table. The table lock enforcer has documented which applications are working within the table. Inside the table, each page has its own lock enforcer that is ensuring concurrency at the page level.

Concurrency – Row level



- This example assumes that the DBA chose row level locking for this table. When row level locking is in place, it is like hiring a lock enforcer for the table and one lock enforcer for each row.

- The example shows that there is one application waiting outside the table because it demands EXCLUSIVE access to the table; however, within the table, all applications are working because, for this example, there is no conflict at the row level.

- Conflict at the row level would occur if a change application and any other application wanted access to the same row at the same time. When this occurs, the lock enforcer gives access to the first application queued for the row.

6.4 Isolation Levels

This controls the duration of read locks during the execution of the program. Below isolation as per the increasing order of isolation and decresing order of Concurancy.

1. UR - Uncommitted Read

- Specifies that statements can read rows that have been modified by other transactions but not yet committed.
- There are many cases where errors can result from reading uncommited data. You want to update the row just fetched - the row could have changed after it was fetched. Do not use UR when balancing financial accounts
- UR will be used for read-only SELECT statements since it is lock avoidance level

2. CS - Cursor Stability

- Default Isolation level
- Locks and unlocks each row, 1 at a time (never has 2 locks at once
- Isolation level cursor stability (CS) takes a lock for row 1 when it is fetching it, but releases it as soon as it fetches row 2, and so on.
- Guaranteed to only return data which was committed at the time of the read

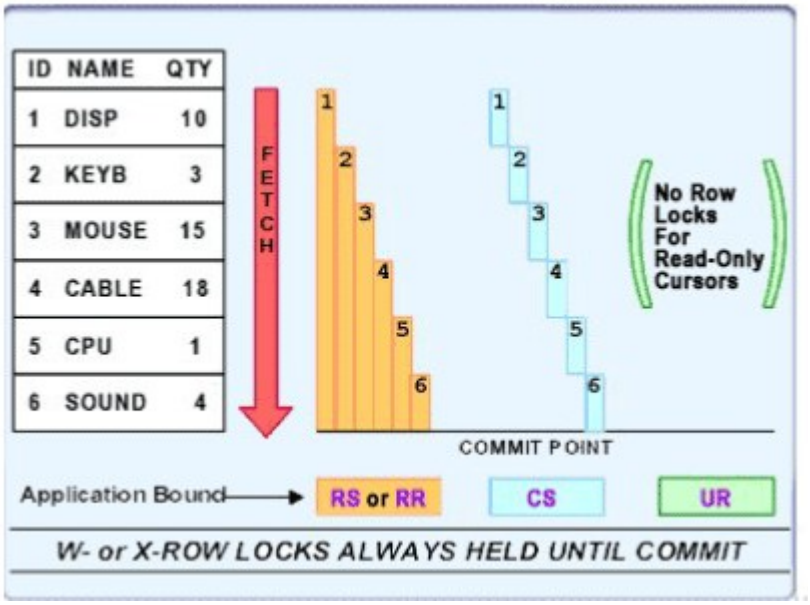
3. RS - Read Stability

- Will keep all qualifying rows locked until the transaction is completed
- although other applications cannot change any of the qualifying rows returned to the original application, they can insert new rows or update rows that did not satisfy the original application's search criteria.

4. RR - Repeatable Read

- Locks the table within a unit of work. An application can retrieve and operate on rows in the table as many times as needed.
- However, the entire table is locked wheather the rows are are qualifying or not, not just the rows that are retrieved. Until the unit of work completes, no other application can update, delete, or insert a row that would affect the table.
- the inner query executes first (so isolation applied first to Inner query)and the result feeds into the outer query (then isolation provides to outer query).

Duration of read locks



RR has highest isolattion and least concurancy

NOTE: WITH clause is used to apply isolation level at query

6.5 Lock Mode

What is LOCK Mode?

The mode of a lock tells what access to the locked object is permitted to the lock owner and to any concurrent processes.

Page or row lock modes

The modes and their effects are listed in the order of increasing control over resources.

Share (S)

- lock owner and concurrent processes can read simultaneously, without changing
- here many concurrent processes may have S or Ulock on a page or row, or might not have any locks.

Update (U)

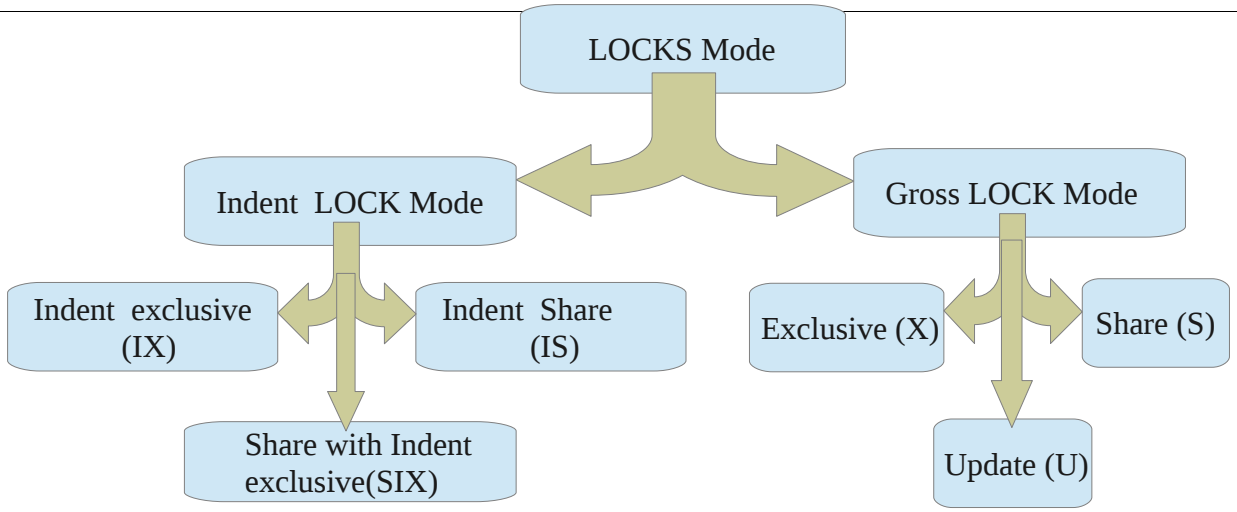
- The lock owner can read, but not change, the locked data. However, the owner can promote the U-lock to an X-lock and then change the data.
- Processes concurrent with the U-lock can acquire S-locks and read the data, but no concurrent process can acquire a U-lock on data.
- U-locks reduce the chance of deadlocks when the lock owner is reading data to determine whether to change it.

Exclusive (X)

- Only the lock owner can read or change the locked page or row.
- A concurrent process can access the data only if the process runs with uncommitted read isolation.

Partition, table space, and table lock modes

The modes and their effects are listed in the order of increasing control over resources.



Intent share (IS)

- lock owner can read data in the table, table space or partition but not change it.
- The lock owner acquires S-locks on pages or rows accessed.
- Concurrent processes may both read and change data.

Intent exclusive (IX)

- lock owner & concurrent processes can read & change data in table, table space or partition.
- The lock owner acquires S locks on pages or rows Read.
- The lock owner acquires X locks on pages or rows changed.
- The lock owner is also eligible to read pages or rows with update acquires U locks on these pages or rows. When a page or row is changed then the U lock is promoted to an X lock.

Share(s)

- The lock owner and any concurrent processes can read, but not change, data in the table, partition, or table space. The lock owner does not need page or row locks on data it reads.

Update (U)

- The lock owner can read, but not change, the locked data. However, the owner can promote the U-lock to an X-lock and then change the data.

- Processes concurrent with the U-lock can acquire S-locks and read the data, but no concurrent process can acquire a U-lock.
- U-locks reduce the chance of deadlocks when the lock owner is reading data to determine whether to change it.

SIX lock (share with intent exclusive)

- The lock owner can read and change data in the table, table space or partition.
- Concurrent processes can read data in table, table space or partition but not change it.
- Only when the lock owner changes data, is the X lock acquired on a page or row.

Exclusive (X)

- Only the lock owner can read or change the locked page or row.
- A concurrent process can access the data only if the process runs with uncommitted read isolation.

MODE Type	Current/owner Process		Concurrent Process	
	Read	Write	Read	Write
IS	Can	X	Can	Can
IX	Can	Can	Can	Can
S	Can	X	Can	X
U	Can	Can	Can	X
SIX	Can	Can	Can (with SI)	X

What is Lock mode compatibility

Lock compatibility controls whether multiple transactions can acquire *locks* on the same resource at the same time.

Page or row lock modes

Held Lock	Requested Lock		
	S	U	X
S	Yes	Yes	No
U	Yes	No	No
X	No	No	No

Partition, table space, and table lock modes

Held Lock	Requested Lock					
	IS	IX	S	U	SIX	X
IS	Yes	Yes	Yes	Yes	Yes	No
IX	Yes	Yes	No	No	No	No
S	Yes	No	Yes	Yes	No	No
U	Yes	No	Yes	No	No	No
SIX	Yes	No	No	No	No	No
X	No	No	No	No	No	No

3 Characteristics of Transaction Loacking

- Lock Size
- Lock Mode
- Lock Duration

Lock Escalation:

1. DB2 provides the application developer with the option to dynamically switch from low

level locking to high level locking based on the number of low level locks held by the application process. This option is called lock escalation and is controlled by the parameter LOCKMAX of CREATE TABLESPACE and ALTER TABLESPACE.

2. The purpose of lock escalation is to reduce the impact of low level locks, both in terms of CPU and storage, at the possible expense of concurrency. When the number of low level locks exceeds a specified threshold, DB2 attempts to promote the high level intent locks, either IS or IX, to S or X, respectively.

3. The LOCKMAX parameter specifies the maximum number of locks that can be taken on this table space by a single process before lock escalation occurs.

4. When lock escalation occurs, DB2 requests a single lock at the next higher level in the lock hierarchy of the table space.

5. The DSNZPARM NUMLKUS corresponds to field LOCKS PER USER on the DB2

6. DSNZPARM NUMLKTS corresponds to the field LOCKS PER TABLE . The acceptable values are 0 to 104857600 and the default is 2000. The value that you specify for this field must be less than the value specified for LOCKS PER USER (NUMLKUS), except when LOCKS PER USER is set to 0.

6.6 Deadlock & Timeouts

What is deadlock?

In concurrent programming, a *deadlock* is a situation in which two or more competing actions are each waiting for the other to finish

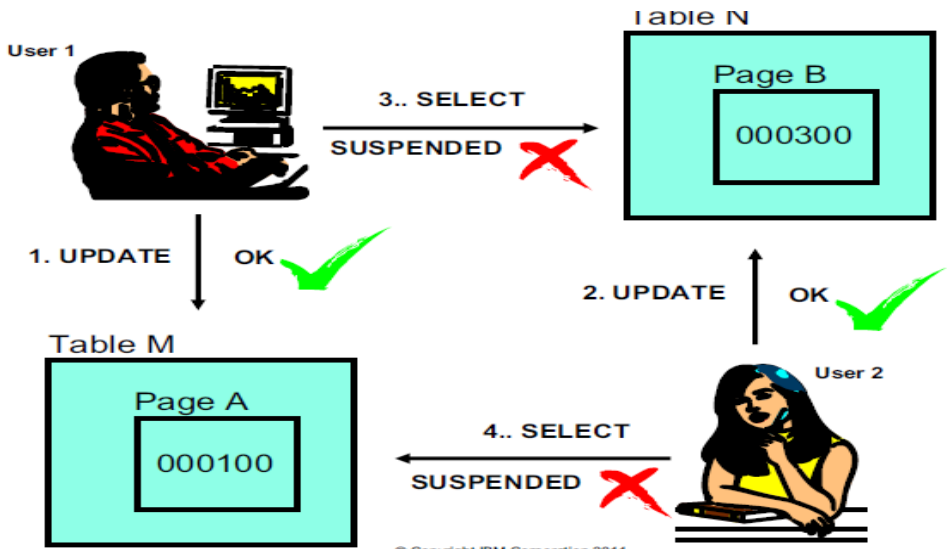
Example:

- User 1 updates table M, and acquires an exclusive lock for page A, which contains record 000100. User 1 then attempts to SELECT from page B of table N, as a part of UOW, while still holding the lock on page A of table M. The job is suspended, because user 2 is holding an exclusive lock on page B.

- User 2 updates table N, and acquires an exclusive lock for page B, which contains

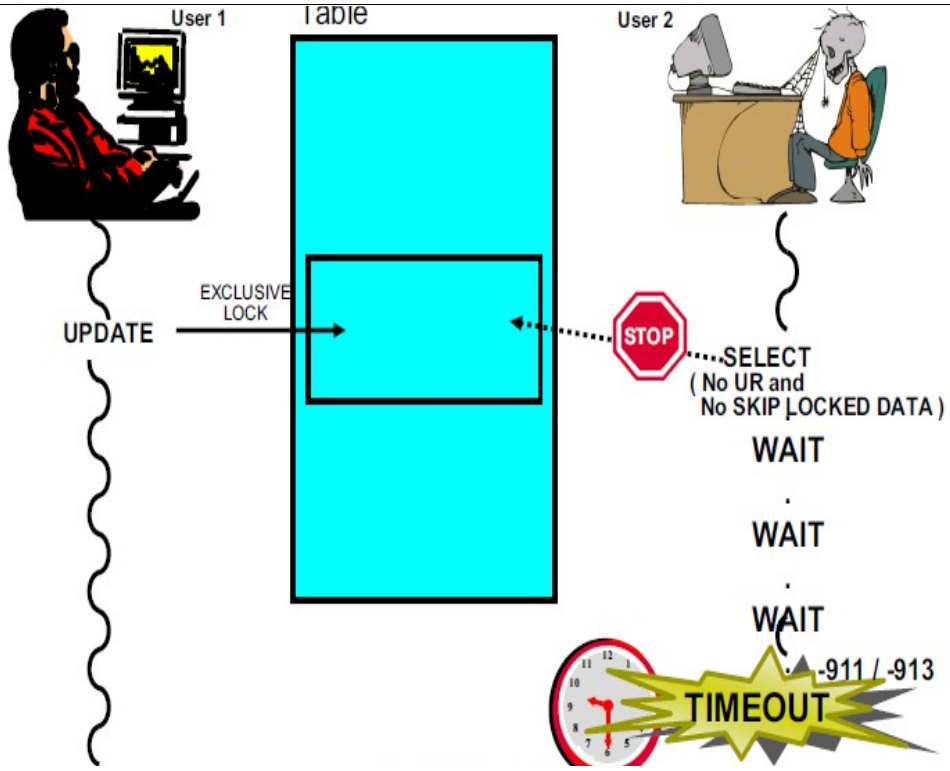
record 000030. User 2 then attempts to SELECT from page A of table, M while still holding the lock on page B of table N. The job is suspended, because user 1 is holding an exclusive lock on page A.

- The situation is a deadlock. After a preset time interval, DB2 can roll back the one of the processes. That frees the locks and allows the remaining processes to continue.



What is timeout

An application process is suspended when it requests a lock that is already held by another application process and cannot be shared. The suspended process temporarily stops running.



- The maximum time requesting process can wait for resource is mentioned in DSNZPARM IRLMRWT – IRLM resource wait timeout
- If timer is expired then we get SQL err (SQLCODE = -911/-913 & SQLSTATE = 57033)

Acceptable values:	1 to 3600
Default:	30

1. Why DB2 need serialization (Concurrancy control)
2. Isolation levels
3. Characteristics of transaction locking
5. What is Lock Escalation
4. What is Deadlocks? How it will happend
5. What is Lock Timeout
6. Default value of IRLMRWT
7. Lock mode compatability
8. Two locking strategies
9. what is LOCKMAX, NUMLKUS, NUMLKTS
10. What are Isolation levels
11. which Isolation has high concurrancy/Isolation
12. How the lock duration measures

7. DB2 - Subsystem

- Sub-system is an sub operating enviormrnt which run on the operating system and which can explicitly started/stopped by the System Administrator/Operator
- DB2 sub-system can defined as set started tasks or set of address spaces that runs on the top of the Operating System.

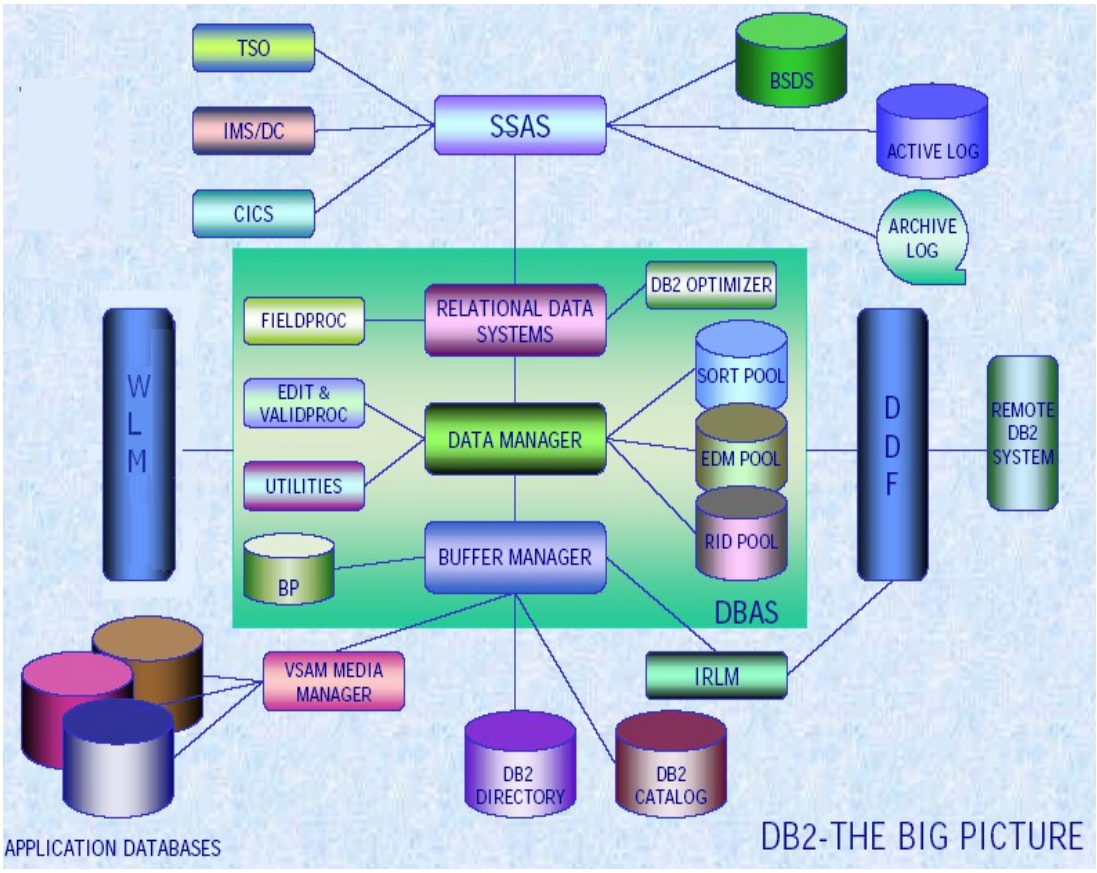
  - DSNMSTR – System Service Address space / Master address space

- DSNDBM1 – Database Address space/ Database Manager

- DSNIRLM – Internal Resource Lock Manager Address space

- DSNDDF – Distributed Data Facility Address Space

- DSNWLM – DB2 WLM Address Space



What database manager/subsystem will do

- Logging – save changes (INSERT, DELETE, UPDATE)
- Locking – lock data when it was using by a process
- Security – provide authentication
- Optimization – reduce CPU cycles as musch as posible
- Recovery – when system failure
- Integrity - everytime

DSNMSTR – System Service Address space / Master address space

1. The Master address space is also called the System Service Address Space.

2. SSAS, or System Service Address Spaces, provides the interface to DB2 with other subsystems, such as CICS, IMS/TM, or TSO.

3. So, The system services address space is address space where DB2 start and as well as stop. so it is mandatoty address space

4. SSAS is address space which support all logging(archive) and recovery in DB2. (Bootstrap dataset has log inventory)

DSNDBM1 – Database Address space/ Database Manager

1. DBM1 contains the core logic of DB2 as a database management system. i.e it contain SQL compiler, Runtime executer, catlog service

2. SQL will execute in DBM1.

3. So, Buffer pools and buffer pool management is also done in DBM1.

4. So it is mandatory address space

DSNIRLM – Internal Resource Lock Manager Address space

- IRLM is not only started task or address space it is also a seperate sub-system. It is address space for Lock manger and concurancy control

- IRLM is shipped with DB2, and each DB2 subsystem must have its own IRLM. You cannot share IRLM between DB2 subsystems

- it is also mandatory sub-system

DSNDDF – Distributed Data Facility Address Space(optional)

Supports distributed data accessing using Distributed Relational Database Architecture (DRDA) Includes the following sub-components:

Data Communications Resource Manager (DCRM)

Distributed Transaction Manager (DTM)

Distributed Relational Data System Manager (DRDS)

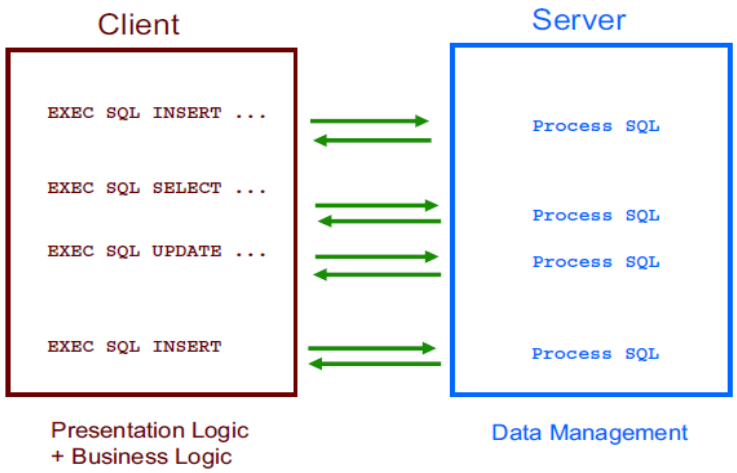
Distributed Data Interchange Services (DDIS)

DSNWLM – DB2 WLM Address Space(optional

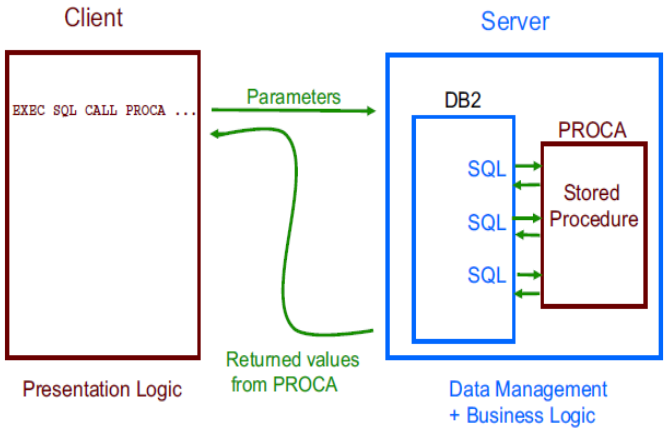
- Prior to DB2 V9, DB2 uses a separated address space called Stord Proc Address Space to execute stored procedures and user-defined functions.
- In V9 and beyond, each stored proc & UDF runs in its own address space spawned by the Work Load Manager (WLM).
- Zero to many address spaces for stored procedures and user-defined functions. It handles in order of priority and isolated from other stored procedures or user-defined functions running in other address spaces

8. db2 Routines

- client has both the presentation logic and business logic. So client send and receive transmissions across the network to the database server very high which have a very noticeable impact upon overall application performance. Due to Network traffic. Client refered as thick client



- Ideally, applications should be designed to minimize message traffic across the network. The business logic is encapsulated at the database server location. The business logic is invoked by the client issuing a CALL to a program resident at the database, called a <u>stored procedure</u>. Client refers as thin client



- her client transfers parameters to server and receives its return values

- Stored procedure = SP/sproc/storp/stopro/storedproc

Syntax:

CREATE PROCEDURE <sp_name>(IN <in_parm dt_type> ,OUT<out_parm, dt_type>)

LANGUAGE SQL

BEGIN

::::::::::::::::::::

END

Difference between Sproc & UDF

- UDFs can be used like any other expression within SQL statements, whereas stored procedures must be invoked using the `CALL` statement.

- Stored procedures may also contain declared variables for processing data and cursors,control statements typically include `IF`, `WHILE`, `LOOP`, `REPEAT`, and `CASE` statements, and more.

- stored procedure is a database object.

- Function must return a value but in Stored Procedure it is optional(Procedure can return zero or n values).

  - Functions can be called from Procedure whereas Procedures cannot be called from Function.

Trigger

- A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server.

- DML triggers execute when a user tries to modify data through any data manipulation language (NSERT, UPDATE or DELETE) event.

Types of trigar

1. BEFORE triggers : They are executed before any SQL operation.
2. AFTER triggers: They are executed after any SQL operation.

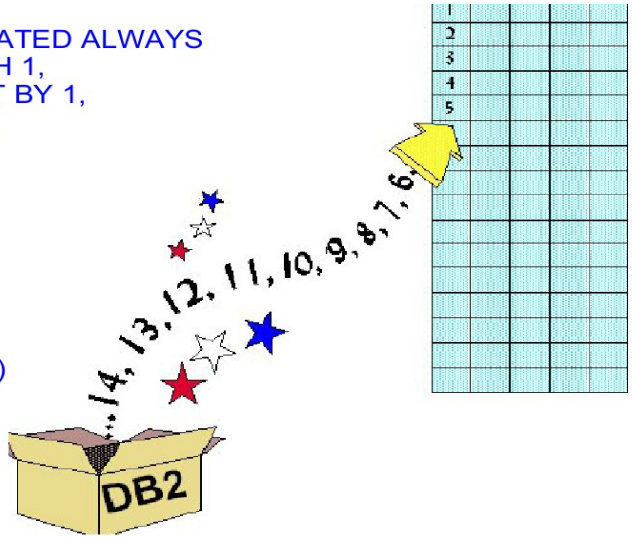
Identity columns

```
CREATE TABLE EMPLOYEE
    (EMPNO    INTEGER GENERATED ALWAYS
       AS IDENTITY (START WITH 1,
                    INCREMENT BY 1,
                    CACHE 20),
     NAME      CHAR(30),
     SALARY    DECIMAL(8,2),
     DEPTNO    SMALLINT);

INSERT INTO EMPLOYEE
    (NAME, SALARY, DEPTNO)
VALUES ('Kate', 100000.00, 50);

VALUES IDENTITY_VAL_LOCAL()
INTO :IVAR;
```

- Can avoid concurrency and performance problems
- Useful for generating unique primary key values



DB2 can automatically generate unique, sequential, and recoverable values in table called Identity columns

SYNTAX: CREATE TABLE ... ... colname datatype ...

GENERATED ALWAYS / BY DEFAULT

AS IDENTITY (START WITH n, INCREMENT BY n,

CACHE n / NO CACHE,

CYCLE / NO CYCLE,

MINVALUE n / NO MINVALUE,

MAXVALUE n / NO MAXVALUE,

ORDER / NO ORDER)

- GENERATED ALWAYS specifies that DB2 always generates values for the column when a row is inserted into the table. DB2 guarantees that all column values are unique. If an INSERT or UPDATE statement is issued in an attempt to provide an explicit value, DB2 returns an error and the statement fails.
- GENERATED BY DEFAULT specifies that DB2 generates a value for the column when

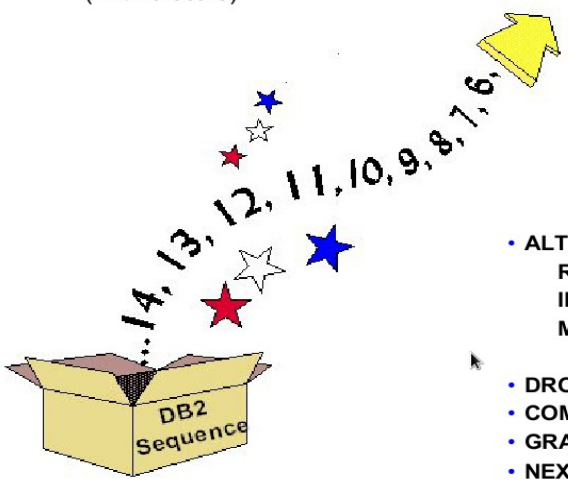
a row is inserted into the table except when a value is provided on the INSERT statement. Values can be changed using an UPDATE statement. DB2 does not guarantee the uniqueness of a generated value among all the column values.

- START WITH & INCREMENT BY, These are numeric data types, can be any +ve or -ve value with a zero scale. If not specified, this keywords defaults to 1.
- CACHE, the number of values that should be cached. We can specify Any value greater than 2 . the default is 20.
- CYCLE allows an identity to wrap around to a new beginning when the minimum or maximum value is reached. Cycling through values a second time can create duplicate values. NO CYCLE causes the identity value to stop being generated when the minimum or maximum is reached. NO CYCLE is the default.
- MAXVALUE specifies the maximum value that can be generated for this identity column.
- MINVALUE specifies the minimum value that can be generated for this identity column.
- NO MINVALUE and NO MAXVALUE are the defaults if either MINVALUE or MAXVALUE is not specified. NO MINVALUE specifies that no minimum end point of the range has been set. The minimum value for an ascending sequence becomes the START WITH value, or 1 if a START WITH value is not specified. For a descending sequence, the default is the lowest value, the largest negative value, for the data type of the column to which the identity is assigned.

Sequences

A stored user-defined object that generates Sequences with certain increment values with ASC / DESC value s

- A stored user-defined object that generates next ASC / DESC value
  - Values generated can be:
    - SMALLINT, INTEGER, DECIMAL (With 0 scale)




```
• CREATE SEQUENCE SEQTEST1
        AS INTEGER
        START WITH 1
        INCREMENT BY 1
        MINVALUE 0
        MAXVALUE 5
        CYCLE
        CACHE 5
        NO ORDER;

• ALTER SEQUENCE SEQTEST1
      RESTART WITH 100
      INCREMENT BY 100
      MAXVALUE 1000000;

• DROP SEQUENCE
• COMMENT ON SEQUENCE
• GRANT / REVOKE ON SEQUENCE
• NEXT VALUE FOR and PREVIOUS VALUE FOR
```

Identity columns	Sequences
Associated with one specific column in one specific table . One to one relationship between identity column and table	Multiple sequences can be used in one table or One sequence can be used in many tables many times.
ALTER TABLE....ALTER COLUMN	ALTER SEQUENCE

9. DB2 – utilites

9.1 Utilites Intoduction

1. What is DB2 Utility?

Db2 utilites are pre-written programes which provide set of services to db2 sub-system. Db2 has utilites to reavh some goals as shown in below.

1. Support core DB2 functionality
2. Reduce CPU consumption, elapsed time
3. Maximize availability
4. Remove constraints and limitations

2. DB2 Utility types ?

Two categories

→ Online Utilites

DB2 online utilities run as standard batch jobs or stored procedures, and they require DB2 to be running.

→ Standalone/ Offline Utilities

The stand-alone utilities run as batch jobs that are independent of DB2. The only way to run these utilities is to use JCL.

Ex: DSN1COPY

Online Utility Catagories

- SETUP/MOVE Utilities
- Maintainance Utilites
- Recovery Utilites

STEPUP or MOVE Utilities

→ LOAD (LUW: IMPORT)

→ UNLOAD (LUW: EXPORT)

→ CHECK DATA

→ CHECK INDEX

MAINTAINANCE Utilites

→ REORG TABLESPACE

→ REORG INDEX

→ REBUILD

→ RUNSTATS

→ REPAIR

RECOVERY Utilities

→ COPY (LUW: BACKUP)

→ RECOVER (LUW: RESTORE)

→ MERGE COPY

→ COPYTOCOPY

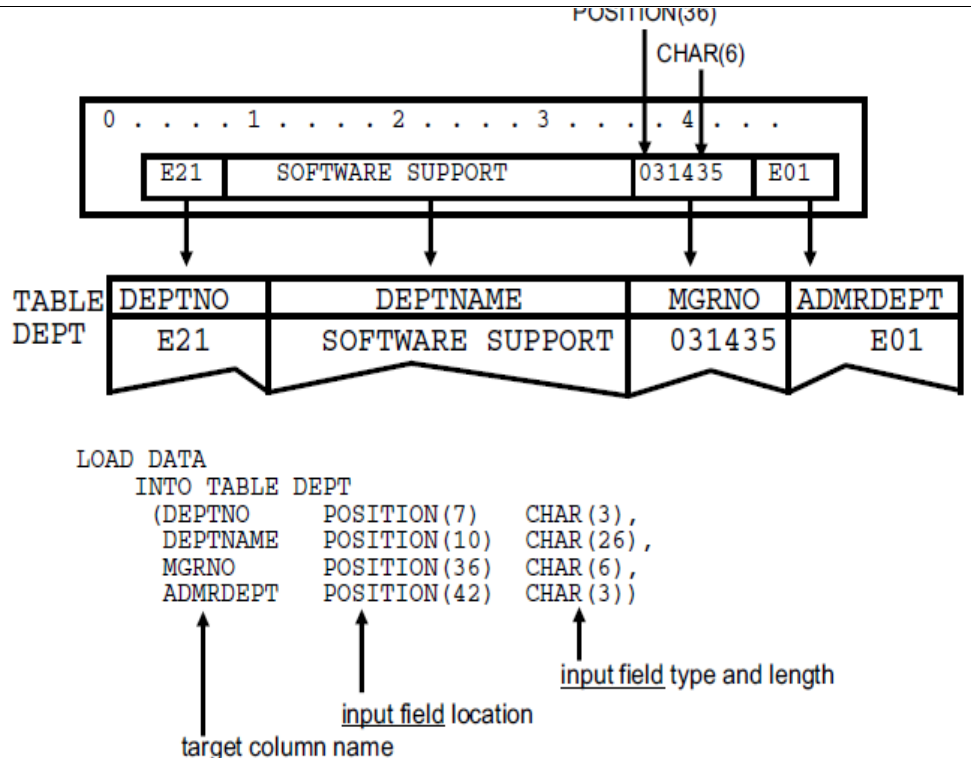
→ QUIESCE

DB2 Utilites Pannel (DB2I 8th option)

- We have to mention Utility name in option 3 (UTILITY)
- In STATEMENT DATA SET (Option 4), you must enter the name of the data set (or PDS member) which contains the control statements for the utility.

9.2 LOAD

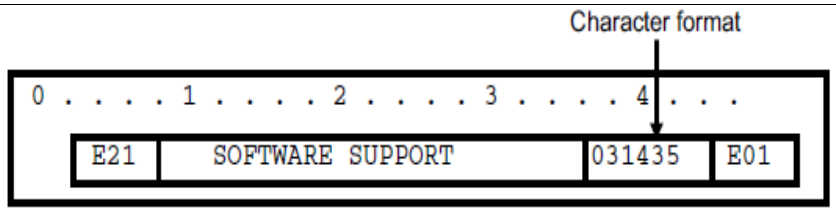
- LOAD utility provides a great deal of flexibility in loading data into a table space from a sequential data set
- Position is a keyword used to mention the positions od column data while loading the *positional input data*.

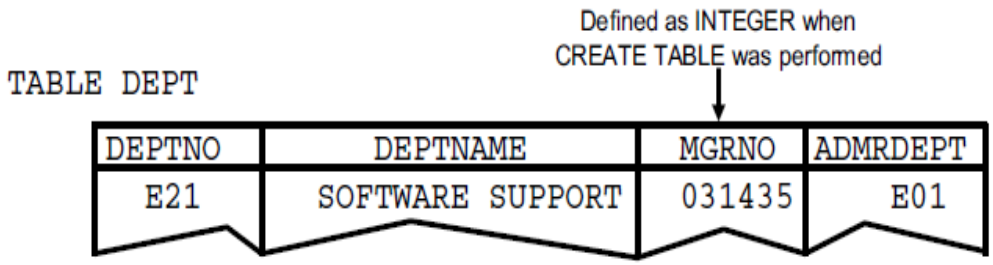


Datatype convention while Loading

- In some cases, the data types of the input record field and the target column do not match. In the example, the INTEGER EXTERNAL keyword tells DB2 that the record field contains numeric data formatted as characters.

- LOAD supports data type conversion among data types as shown in below diagram.






```
LOAD DATA
    INTO TABLE DEPT
     (DEPTNO      POSITION(7)    CHAR(3),
      DEPTNAME    POSITION(10)   CHAR(26),
      MGRNO       POSITION(36)   INTEGER EXTERNAL(6),
      ADMRDEPT    POSITION(42)   CHAR(3))
```

- Numeric Data Conversion

Output Data Types

Input Data Types

	SMALLINT	INTEGER	BIGINT	DECIMAL	FLOAT	DECFLOAT
SMALLINT	D	Y	Y	Y	Y	Y
INTEGER	Y	D	Y	Y	Y	Y
BIGINT	Y	Y	D	Y	Y	Y
DECIMAL	Y	Y	Y	D	Y	Y
FLOAT	Y	Y	Y	Y	D	Y
DECFLOAT	Y	Y	Y	Y	Y	D

D=Defaults used when you do not specify the input data type in a field spec of the INTO TABLE statement

- Character Data Conversion

Output Data Types

Input Data Types

	CHAR	VARCHAR	GRAPHIC	VARGRAPHIC	BINARY	VARBINARY
CHAR	D	Y	Y*	Y*	Y	Y
VARCHAR	Y	D	Y*	Y*	Y	Y
GRAPHIC	Y*	Y*	D	Y	N	N
VARGRAPHIC	Y*	Y*	Y	D	N	N
BINARY	N	N	N	N	D	Y
VARBINARY	N	N	N	N	Y	D

*=Conversion applies when either the input data or the target table is Unicode

- Time Data Conversion

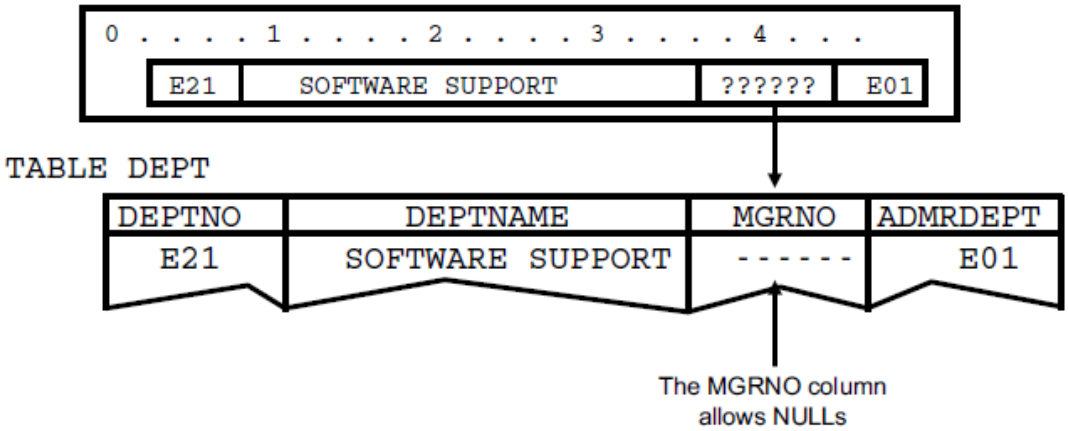
Output Data Types

Input Data Types

	DATE	TIME	TIMESTAMP
DATE EXTERNAL	D	N	N
TIME EXTERNAL	N	D	N
TIMESTAMP EXTERNAL	Y	Y	D

Handling Nulls & Defaults while Loading

- Setting columns to NULLs




```
LOAD DATA
    INTO TABLE DEPT
     (DEPTNO      POSITION(7:9)     CHAR,
      DEPTNAME    POSITION(10:35)   CHAR,
      MGRNO       POSITION(36:41)   CHAR
                     NULLIF (36:41) = '??????',
      ADMRDEPT    POSITION(42:44)   CHAR)
```

In the visual, '??????' in the input data is used to indicate that a value is unknown. The NULLIF (36:41) = '??????' parameter of the MGRNO column causes X'FF' to be loaded in the one-byte null flag prefixing the MGRNO column. This indicates in DB2 that the value is unknown.

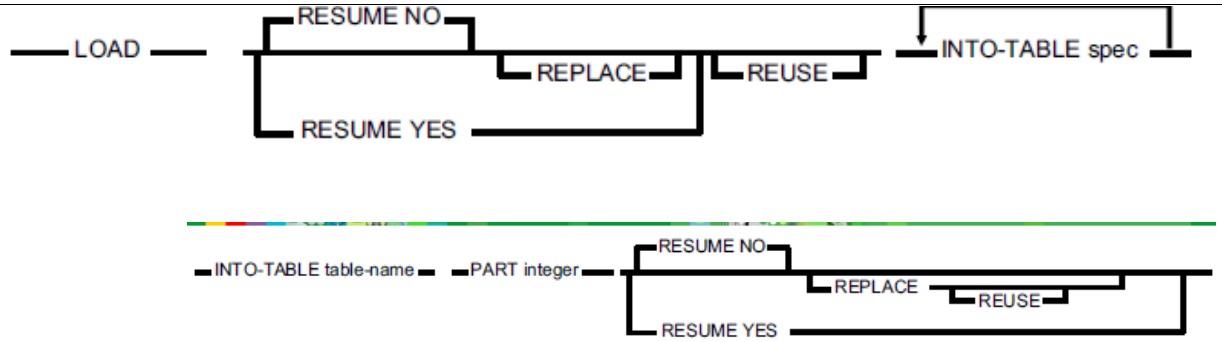
the DEFAULTIF parameter can be used with the LOAD statement as shown below:

LOAD DATA INTO TABLE TAB1

(COL1 POSITION (1:4) INTEGER

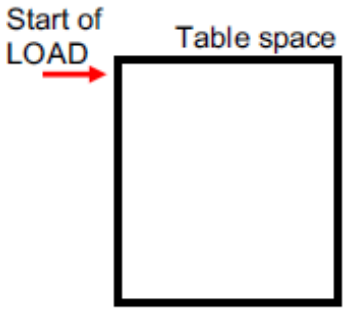
DEFAULTIF (1:4) = -1,

.....



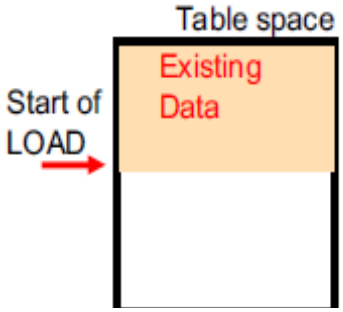
RESUME NO

This is default, for this we required a Emty Tablespace to load data.



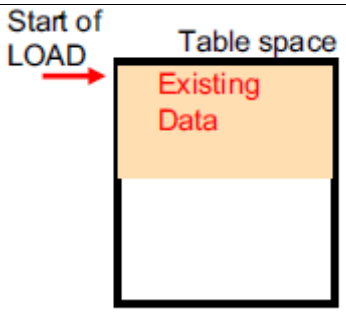
RESUME YES

Loading data will abend to existing data, so we does't need Empty tablespace



REPLACE

DB2 will reset to empty the table space and all its indexes. For DB2-managed data sets, the data sets are deleted and redefined. The data in all tables is lost when you LOAD REPLACE a single table in a multi-table table space.



REPLACE with REUSE

LOAD will reset and reuse all existing DB2-managed data sets. That is, the data sets are not deleted and redefined.

RESUME YES with SHRLEVEL CHANGE

It causes the LOAD utility to add new rows using processing similar to SQL INSERT. It inserts rows according to the clustering index, provided that sufficient free space is available. LOG YES is required.

Other compatable combinations RESUME YES SHRLEVEL NONE

RESUME NO SHRLEVEL NONE

ENFORCE CONSTRAINTS/NO

Allow or suppress enforcement of the table check or referential constraints while LOADing. ENFORCE NO sets the CHECK PENDING state . ENFORCE CONSTRAINTS is default

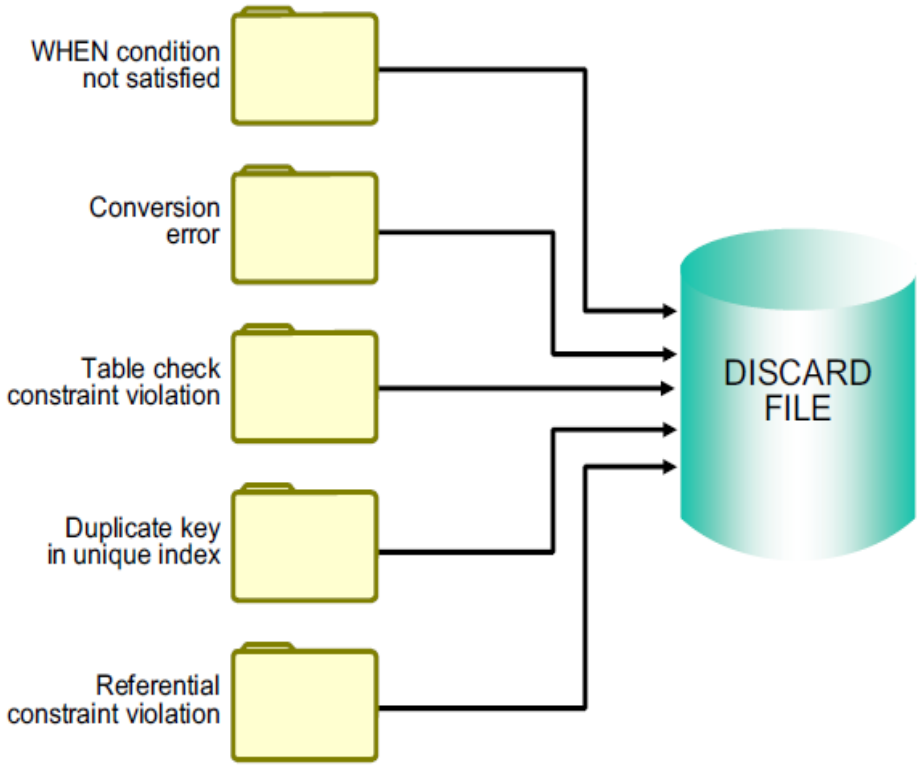
LOG YES/NO /NOCOPYPEND

LOG NO LOADs faster but impact on RECOVERY . So when a table space is loaded with LOG NO. SQL INSERTS, UPDATES, and DELETES cannot be executed, so we have to mention clause for image copty,without copy leads copy-pending state we can avoid with NOCOPYPEND

DISCARDS n

Records that cannot be loaded are written to the discard data set. DISCARDS parameter sets a limit for the number of records to be discarded.

Ex: LOAD DATA DISCARDS 25 -- LOAD stops after 25 errors



LOAD - delimited input

```
LOAD DATA
FORMAT DELIMITED COLDEL ',' CHARDEL '"' DECPT '.'
INTO TABLE DELT
(FILENO      CHAR,
 NAME        VARCHAR,
 DATE1       DATE EXTERNAL,
 TIME1       TIME EXTERNAL)
 "001", "KUMAR",2000-02-16, 00.00.00
 "002", "KASCHTA",2001-04-17, 06.30.00
 "003", "HUTH",2002-06-18, 12.30.59
 "004", "WHITLARK",1991-08-19, 18.59.30
 "005", "BAUMGARTNER",2000-12-20
```

The FORMAT DELIMITED option indicates that the data is in delimited format. The COLDEL option indicates that the column delimiter is a comma (,). The CHARDEL option indicates that the character string delimiter is a double quotation mark ("). The DECPT option indicates that the decimal point character is a period (.).

INDDN-sysrec

INDDN is control card for input dataset. By Default it hold DD name SYSREC where we have to specify input dataset

DISCARDDDN sysdisc

DISCARDDDN is control card for discard dataset. By Default it hold DD name SYSDISC where we have to specify discard dataset

COPYDICTIONARY integer

copy the data with dictionary page from one partition or tablespace to other

integer is partion number.

PRESORTED(YES/NO)

if we mentioned yes, it means that data is already sorted in order.

PARALLEL n

maximum number of subtasks run in utilty execution. DB2 will take optimum value

COPYDDN(ddname) or RECOVERYDDN ddname

suports inline copy for image copy.

FLASHCOPY(NO/YES/CONSISTENT)

enble flashcopy technology. If you mention SHARLEVEL CHANGE copy operation must be on uncommited data then we have to use consistent.

FCCOPYDDN

dd name of flash copy image copy dataset

EBCDIC/ASCII/UNICODE

format of data

STATISTICS

inline statistics here supports RUNSTATS keywords like TABLE, freqval..

CONTINUEIF

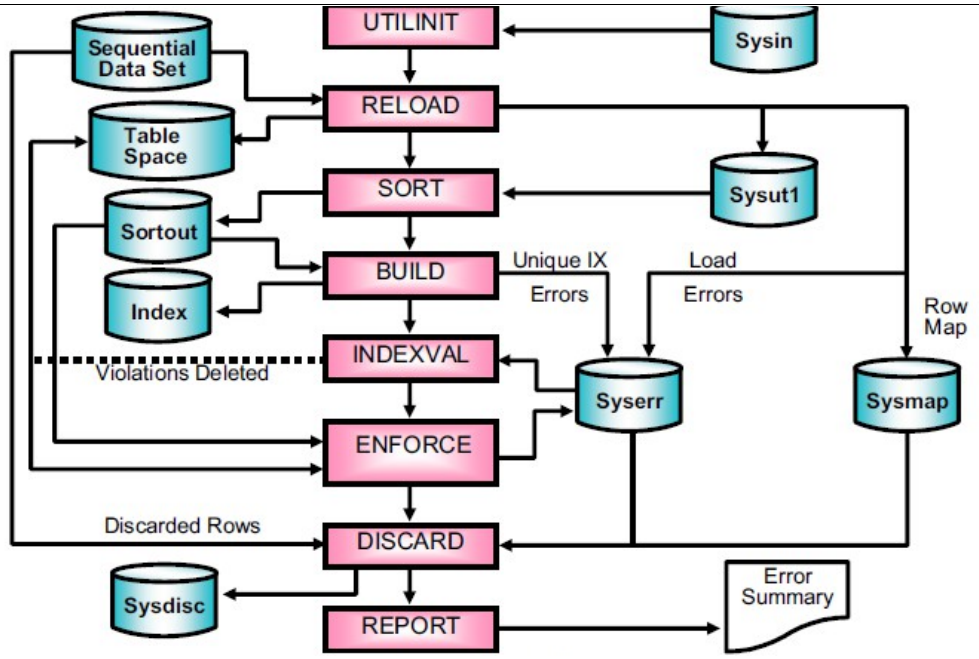
specifies the position where record concatination need to be happend

```
LOAD DATA INDDN INPUT REPLACE LOG NO
COPYDDN(SYSCOPY) RECOVERYDDN(REMCOPY)
CONTINUEIF(79:80)='++'
INTO TABLE ADMIN.DEPT
```

WHEN Clause

The WHEN clause tells the utility which records in the input data set are to be loaded. Input records that do not satisfy any WHEN clause of any INTO TABLE statement are written to the discard data set, if it exists.

Load Phases



1. UTILINT: Utility Initilization process.

2. RELOAD: simpley Loded into tablespace

   → Load Record Template (Record types)

   → Write temporary file records for indexes & foreign keys

   → Once read the sequential i/p dataset & check constraints for each row

   → Load Data into tablespace or partition and create Inline copies if you specify COPYDDN or RECOVERYDDN

3. SORT: Sorts temporary file records before creating indexes or validating referential constraints, if indexes or foreign keys exist.

4. BUILD:Creates indexes from temporary file records

   if indexes are building parallel the sort and build phases are combined as single phase called SORTBLD phase

5. INDEXVAL: check and corrects any index voilations

6. ENFORCE: check and corrects any FK voilations

7. DISCARD: Discard voilated rows

9.3 CHECK DATA

1. What is Check Pending (CHKP)?

The CHECK-pending (CHKP) restrictive status indicates that an object might be in an inconsistent state due to violation of a referential constraints and must be checked. we can see this status through display database command.

2. What causes Check Pending?

- The following utilities set CHECK-pending status on a table space if referential integrity constraints voilations are encountered:
    - LOAD with ENFORCE NO
    - RECOVER to a point in time
    - CHECK LOB
- LOAD REPLACE of parent table in its own table space
- Adding FK or table check constraint to populated table (when voilation occur)

5. What is the purpose of Check Pending?

The purpose is to ensure the integrity of DB2 data.

6. What is permitted or not permitted?

No SELECT, INSERT, UPDATE, or DELETE - But DROP and ALTER allowed

No update of PK when FK table is in CHKP - No COPY or REORG

But CHECK DATA, LOAD REPLACE, and RECOVER allowed

7. What does it apply to?

Check Pending applies to Dependent table spaces of an RI structure

8. How can Check Pending be reset?

Some of the ways Check Pending can be reset are:

• CHECK DATA...DELETE YES.

• CHECK DATA...DELETE NO and no violating rows are found.

• REPAIR.....NOCHECKPEND

9. Using the CHECK DATA utility

The CHECK DATA utility checks table spaces for violations of referential and table check constraints, and reports information about violations that it detects.

1. Always produces a report with details of each violation

```
CHECK DATA TABLESPACE DB1.TS1
```

Invalid rows can optionally be copied to a user-specified exception table.

```
CHECK DATA TABLESPACE DB1.TS1
FOR EXCEPTION  IN UID1.TBL1 USE UID1.E_TBL1
                    IN UID1.TBL2 USE UID1.E_TBL2
DELETE NO
```

Here IN clause has Original Table & USE clause has Exception table. Invalid rows can optionally be deleted from the dependent table after they have been copied to the exception table.

```
CHECK DATA TABLESPACE DB1.TS1
FOR EXCEPTION  IN UID1.TBL1 USE UID1.E_TBL1
                    IN UID1.TBL2 USE UID1.E_TBL2
```

DELETE YES

10. Exception Tables:

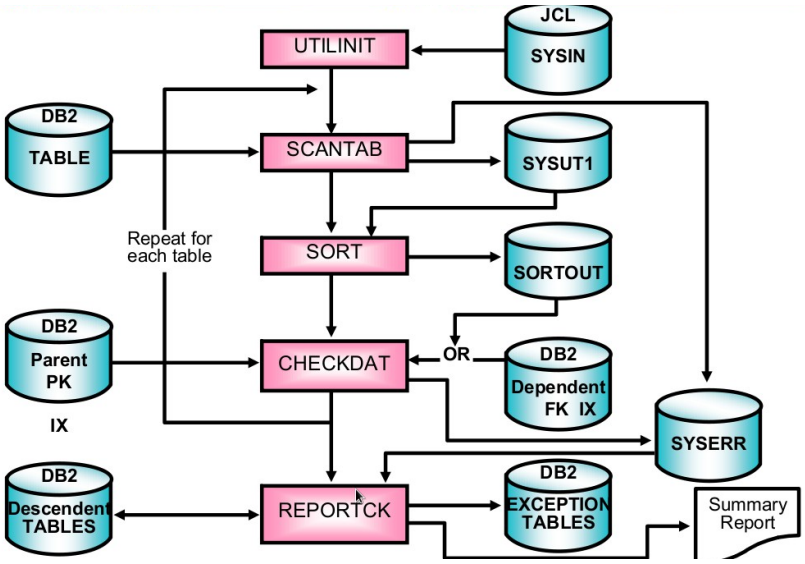
1. Exception table need to create same column names and descriptions when

2. Any other columns of the exception table that are not in the dependent table will be ignored, but must be nullable or NOT NULL WITH DEFAULT.

3. The exception table cannot have more than 750 columns.

4. An exception table can reside in any table space, but should not have a unique index or any other constraint that could cause errors at insert.

11. CHECK DATA Utility phases



1. UTILINIT : Performs initialization

2. CHECKXML : checking XML structure (only forXML table spaces otherwise optional)

3. SCANTAB : Scan the tablespace and associated FK's

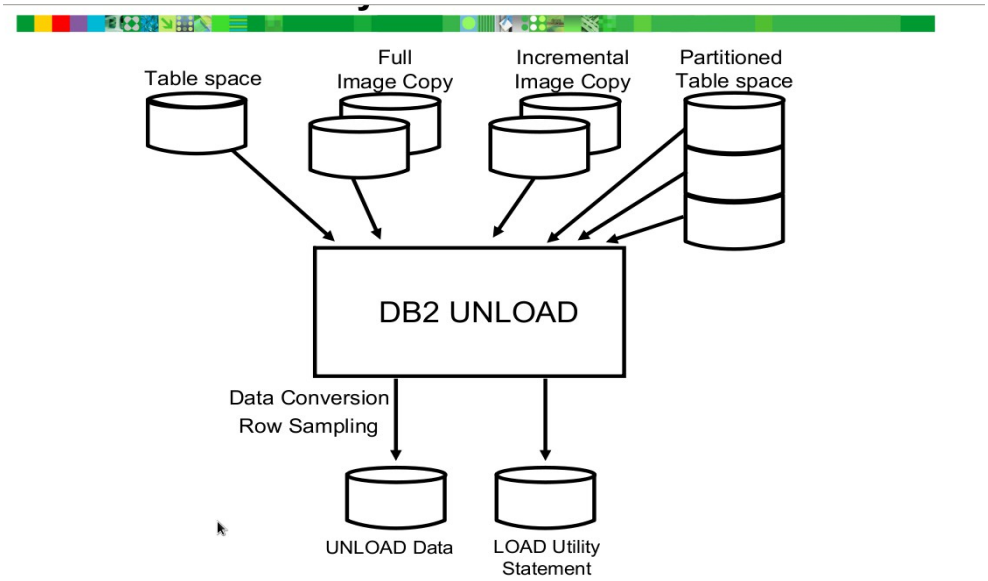
4. SORT: Sorts foreign keys if they are not extracted from the foreign key index

5. CHECKDAT : validations and detect errors

6. REPORTCK : Copies error rows into exception tables, and delete them from source table if DELETE YES is specified

7. UTILTERM : Performs cleanup

9.4 UNLOAD



You can unload into sequential dataset from:

1. An entire table space
2. Multiple table spaces with one run of UNLOAD
3. List of tablespace

4. Specific partitions using parallelism

5. Certain tables only

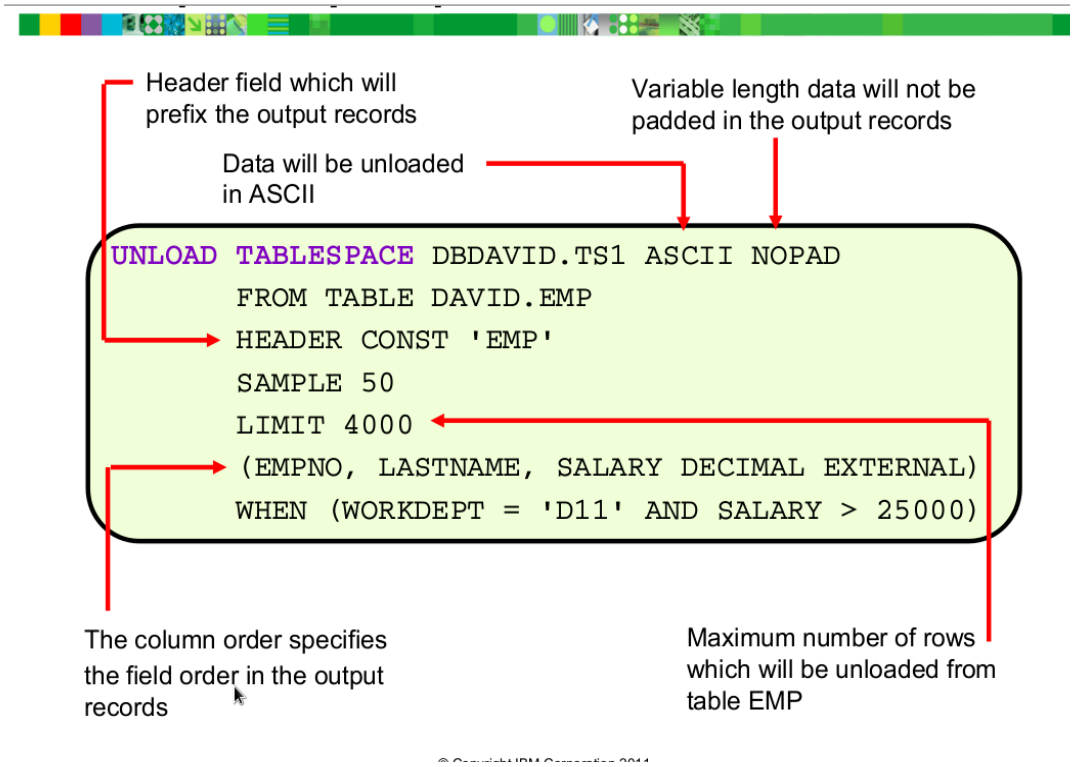
6. Certain rows only

7. Certain columns only

8. Load with data conversion (Datatype or CCSID)

9. apply when clause

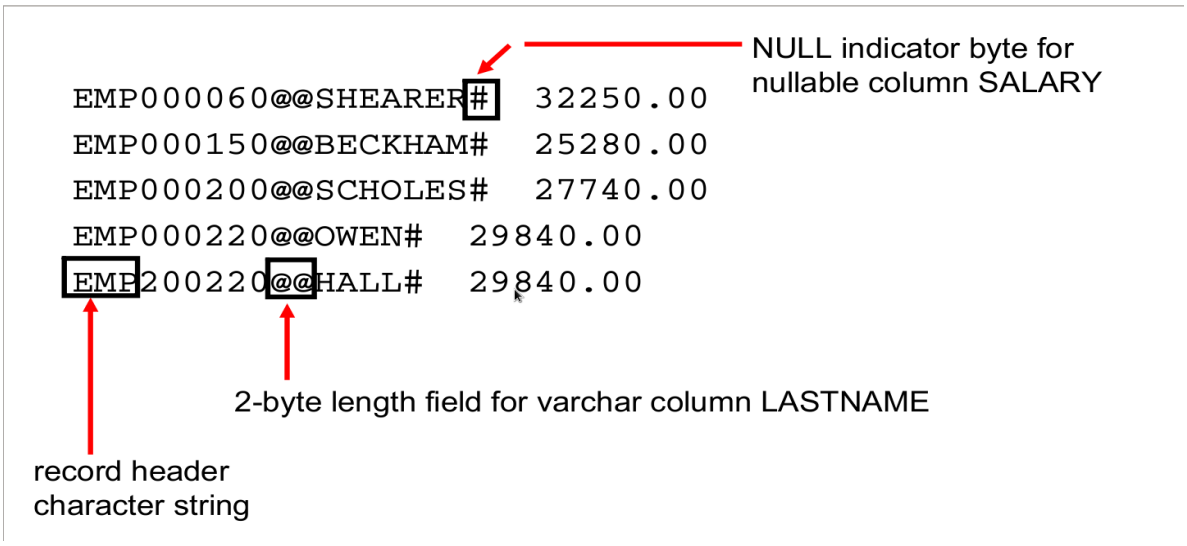
SYNTAX:



- SAMPLE and LIMIT are independent options. If both options are specified, the limit is applied to the sampled rows.

- For example, let's say you specify SAMPLE 50 LIMIT 4000. If there are approximately 10,000 rows in the table, and approximately 5,000 rows qualify the WHEN conditions specified,
- then the UNLOAD utility unloads approximately 2,500 rows from the table, The SAMPLE option indicates that 50% of the rows are to be sampled. but no more than 4,000 rows.

The visual shows an example of the output records unloaded by the UNLOAD utility control statement shown on the previous visual.



UNLOAD - delimited output

```
UNLOAD TABLESPACE DELDB.DELTS
   DELIMITED CHARDEL '#' COLDEL ';'  DECPT '.'
   PUNCHDDN SYSPUNCH
   UNLDDN SYSREC EBCDIC
   FROM TABLE  DELT
    (FILENO      POSITION(*)   CHAR(3),
     NAME        POSITION(*)   VARCHAR(30),
     DATE1       POSITION(*)   DATE EXTERNAL,
     TIME1       POSITION(*)   TIME EXTERNAL)
```

Optional keywords

POSITION(*) keywords and char field lengths optional

Unloaded data looks like:

```
#001#;#KUMAR#;02/16/2000;00:00 AM
#002#;#KASCHTA#;04/17/2001;06:30 AM
#003#;#HUTH#;06/18/2002;12:30 PM
#004#;#WHITLARK#;08/19/1991;06:59 PM
#005#;#BAUMGARTNER#;12/20/2000;12:00 AM
```

No trailing blanks for VARCHAR

Note that field lists are optional for LOAD / UNLOAD, and are primarily used

The UNLOAD utility does not support:

– Index spaces

– Views

– Concurrent image copies

– FlashCopy image copies

– Directory table spaces

Practical Code for positional output into single dataset

```
UNLOAD TABLESPACE WWV8301D.TSPROJ
PUNCHDDN SYSPUNCH
UNLDDN SYSREC
FROM TABLE PROJECT
HEADER CONST 'UNLOADED'
```

Practical Code for delimited output into single dataset

```
UNLOAD TABLESPACE WWV8301D.TSPROJ
DELIMITED
PUNCHDDN SYSPUNCH
UNLDDN SYSREC EBCDIC
FROM TABLE PROJECT
```

- PUNCHDDN SYSPUNCH specifies that the UNLOAD utility is to generate LOAD utility control statements and write them to the data set that is defined by the SYSPUNCH DD statement. By default they are Positional control statements will generate.
- UNLDDN (Keyword) SYSREC (dfault dd name) specifies dataset where data is unloaded.

Practical Code for positional output into multiple datasets

```
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 1
PUNCHDDN SYSPUNCH1
UNLDDN SYSREC1
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 2
PUNCHDDN SYSPUNCH2
UNLDDN SYSREC2
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 3
PUNCHDDN SYSPUNCH3
UNLDDN SYSREC3
```

in above syntax we give SYSPUNCH at each paetition so we will get LOAD utility statements at PARTITION level. If we give single SYSPUNCH we will get single set of load utility statements for entire tablespace

```
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 1
```

```
PUNCHDDN SYSPUNCH
UNLDDN SYSREC1
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 2
PUNCHDDN SYSPUNCH
UNLDDN SYSREC2
UNLOAD TABLESPACE WWV8301D.TSPROJ  PART 3
PUNCHDDN SYSPUNCH
UNLDDN SYSREC3
```

9.5 REORG

REORG Results on Tablespace

After the execution of the REORG utility, following things will happened

1. all rows have been put into their clustering sequence

– SYSTABLESPACESTATS column REORGUNCLUSTINS

– SYSINDEXPART columns NEAROFFPOSF AND FAROFFPOSF

2. Restore free space

3. Eliminate overflow records

– SYSTABLEPART columns NEARINDREF and FARINDREF

– SYSTABLESPACESTATS column REORGFARINDREF

4. Modify PRIQTY, SECQTY (Eliminate extents & over allocations)

When we apply REORG on Tablespace ?

1. Pending restrictive status (REORP): Indicates that a table space partition definition has changed and the affected partitions must be reorganized before the data is accessible.

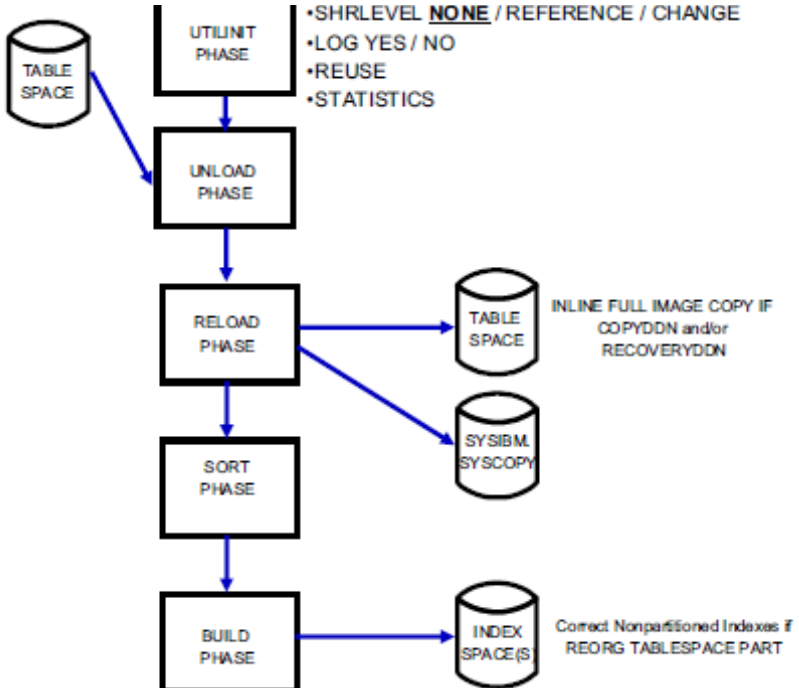
2. Pending advisory status (AREO*): Indicates that a table space, or partition needs to be reorganized for optimal performance.

3. Advisory REORG-pending status (AREOR): Indicates that a table space needs to be reorganized for optimal performance and to apply pending definition changes. This state is applicable only in DB2 10.

SHRLEVEL parameter determines the accessibility of the data during reorganization

- NONE: no SELECT, INSERT, UPDATE or DELETE are allowed
- REFERENCE: data can only be read , for this shadow dataset is requied
- CHANGE: SELECT, INSERT, UPDATE & DELETE can be executed .for this shadow dataset is requied

REORG Phases



UTILINIT : Performs initialization and setup.

UNLOAD : Unloads the table space and sorts data if a clustering index exists and it writes them to a sequential data set.

RELOAD : Reloads data from the sequential data set into the table space and creates full image copies if you specify COPYDDN, RECOVERYDDN

SORT: Sorts index keys. The sorted keys are passed in memory.

BUILD : Builds indexes and updates index statistics.

REORG Results on Indexspace

After the execution of the REORG utility, following things will happened

1. Restore physical ordering of leaf pages

– SYSINDEXSPACESTATS column REORGLEAFFAR

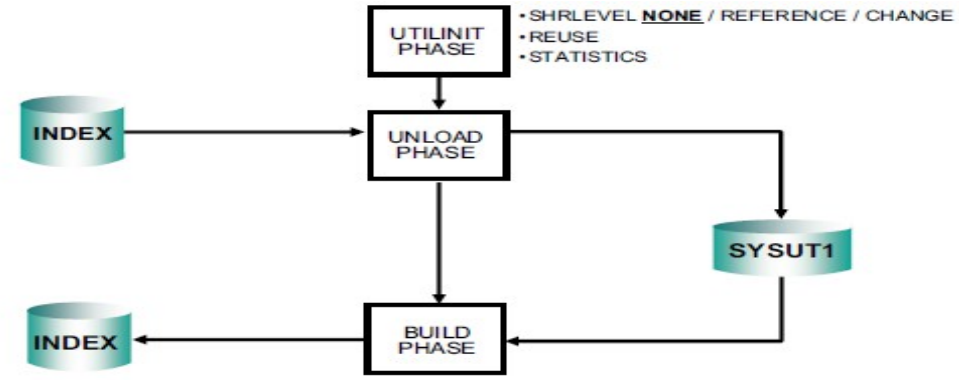
– SYSINDEXPART column LEAFDIST

2. Restore free space

3. Reduce number of levels

– SYSINDEXSPACESTATS column REORGNUMLEVELS

4. Modify PRIQTY, SECQTY (Eliminate extents & over allocations)



Note:

- For partitioned table spaces, if the whole table space is reorganized, or For non-partitioned table spaces, all indexes related to all tables in the table space are "reorganized" when the table space is reorganized, since they are rebuilt as part of the process.

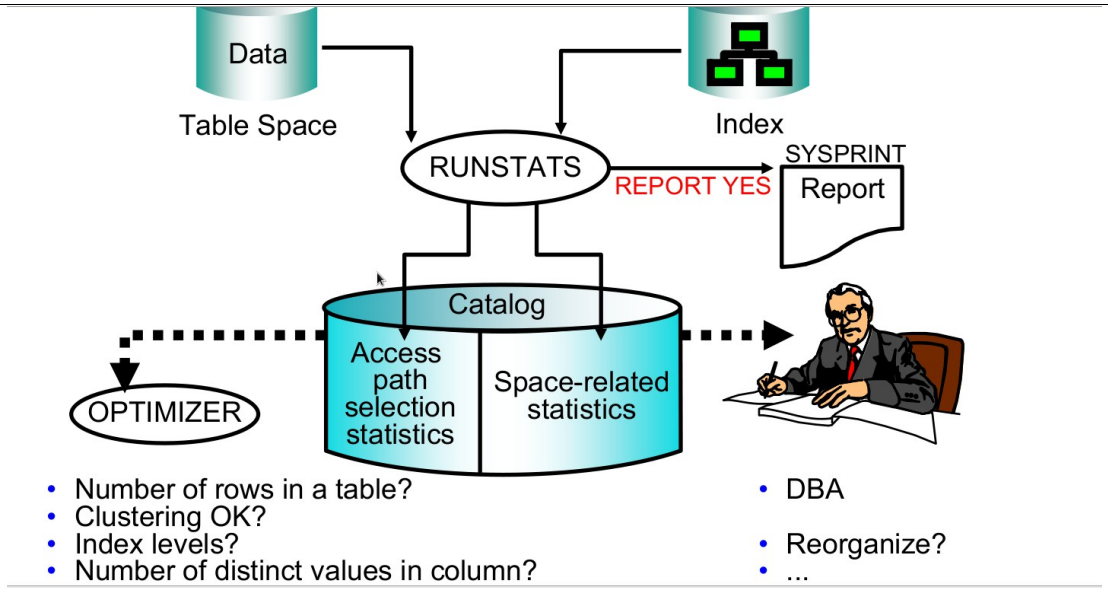
- For partitioned table spaces, if one or more partitions are reorganized, the corresponding partitions of all partitioned indexes related to the table in the table space are "reorganized", since they are rebuilt as part of the process. The nonpartitioned indexes are not rebuilt, and therefore are not reorganized.

REORG TABLESPACE PERSDB.PERSTSP1

```
REORG TABLESPACE PERSDB.PERSTSP2
        PART 3
REORG TABLESPACE PERSDB.PERSTSP2
        PART 3:6
        SHRLEVEL NONE
REORG INDEX DBA01.DEPTNO_INDEX
REORG INDEX DBA01.DEPTNO_INDEX
        PART 3  REUSE
    DISCARD
            REORG TABLESPACE HR.PERS
           DISCARD        FROM TABLE COMPANY.EMPLOYEE
          WHEN(EMPDEPT='1CA' OR EMPDEPT='1CF')
```

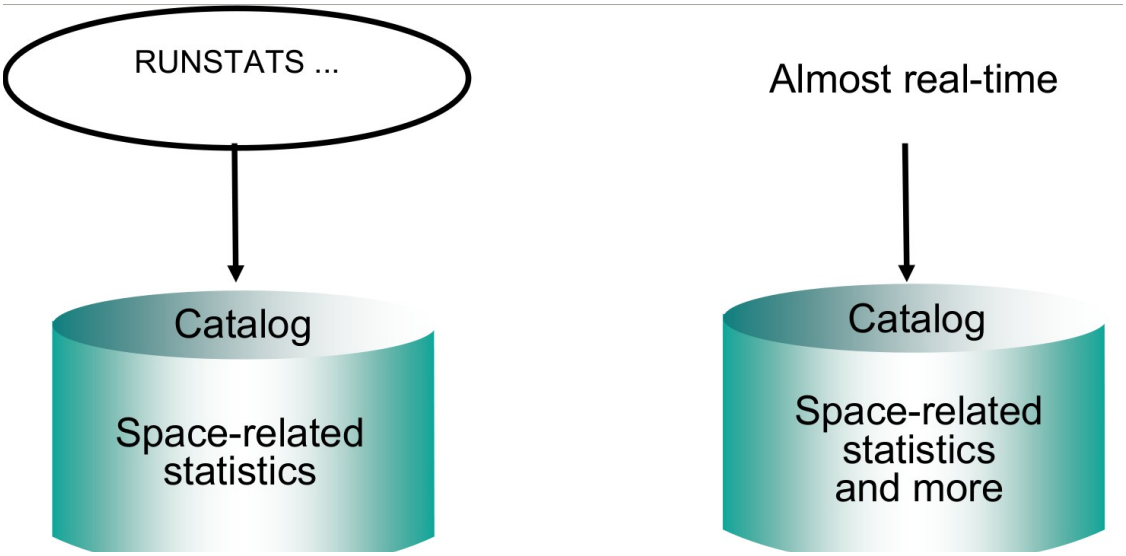
Discard is a keyword used dicard rows meeting conditions in when clause

9.6 RUNSTATS and real-time statistics



updates DB2 catalog statistics with current information on tables & indexes . The statistical information in the catalog provides significant input to the DB2 optimizer, so that an efficient access path is chosen to access the data. space-related statistics are needed to apply REORG

RUNSTATS versus real-time statistics



- For the space-related statistics, there is an alternative, called real-time statistics, which collects the same information as RUNSTATS and some other statistics, such as the

1. number of rows inserted, updated, or deleted since the last execution of REORG, LOAD REPLACE, REBUILD INDEX, RUNSTATS,

2. the number of changed pages since the last execution of COPY.

- DB2 writes or updates these statistics to the mentioned catalog tables every n minutes, n being a system parameter (DSNZPARM STATSINT).
- The default value for STATSINT is 30 minutes.
- But RUNSTATS must apply atleast once after creation to activate counters of related catlog tables.

RUNSTATS Example

RUNSTATS TABLESPACE DB.TS

SHRLEVEL CHANGE

TABLE (ALL) INDEX (ALL) → specifies all tables & indexes

SHRLEVEL CHANGE allows other programs to change the table space or index.

Inline statistics

You can also use inline statistics to collect statistics for database objects, when you run other utilities, such as the LOAD, REBUILD INDEX, REORG INDEX, and REORG TABLESPACES. Inline statistics enabled by using the STATISTICS keyword

Catlog table updated

RUNSTATS+	Results

TABLESPACE DB1.TS01	Only Tablespace and table characteristices will be updated.
TABLE(ALL)	column characteristices of all tables of tablespace will be updated
TABLE(...) COLUMN(.....)	column characteristices of that tables will be updated
INDEX(ALL)	Mentioned Index characteristices is updated
INDEX(.....)	Index characteristices of all tables of tablespace will be updated
SHRLEVEL	SHRLEVEL REFERENCE SHRLEVEL CHANGE → default one
HISTORY ALL ACCESSPATH SPACE NONE	Allows old statistics to be kept in separate catalog tables called SYSIBM.SYS..._HIST options specify wheater all or Accesspath or space related or none statistices will kept
PART n	For partitioned table spaces and indexes, allows the collection of statistics for the specified partition only
REPORT YES/NO	Generate set of messages that report collected statistices
UPDATE ALL ACCESSPATH SPACE NONE	→ All statistices will updated → Acesspath statistices will updated → Space related statistices will updated → nothing will update

COLGROUP(...)	Set of columns treated as group. Runstats collect cardinality on that group (not applicable for indexes and XML ts)
HISTOGRAM NUMQUANTILES n	(n>1) How quantities that utility to collect. It must not exceed distint values of COLUMN or COLUMNGROUP, By default 100
KEYCARD	
FREQVAL	

Table Characteristices

TABLE	Column
SYSIBM.SYSTABLESPACE	
SYSIBM.SYSTABLEPART	
SYSIBM.SYSTABLES	

TABLE CHARACTERISTICES

```
DSNU613I  -DSNA 182 04:51:29.77 DSNUSUTP - SYSTABLEPART CATALOG STATISTICS FOR D
                                CARD              = 114
                                CARDF             = 1.14E+02
                                NEARINDREF        = 0
                                FARINDREF         = 0
                                PERCACTIVE        = 0
                                PERCDROP          = 0
                                PAGESAVE          = 0
                                SPACE             = 720
                                SPACEF            = 7.2E+02
                                PQTY              = -1
                                SQTY              = -1
                                DSNUM             = 1
                                EXTENTS           = 1
```

```
DSNU614I  -DSNA 182 04:51:29.77 DSNUSUTB - SYSTABLES CATALOG STATISTICS FOR TSOU
                                CARD              = 114
                                CARDF             = 1.14E+02
                                NPAGES            = 2
                                NPAGESF           = 2.0E+00
                                PCTPAGES          = 50
                                PCTROWCOMP        = 0
                                AVGROWLEN         = 62
                                SPACEF            = 8.0E+00
DSNU610I  -DSNA 182 04:51:29.77 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR TSOUA16.
DSNU612I  -DSNA 182 04:51:29.77 DSNUSUTS - SYSTABLESPACE CATALOG STATISTICS FOR
                                NACTIVE           = 180
                                NACTIVEF          = 1.8E+02
DSNU610I  -DSNA 182 04:51:29.77 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DB16
```

COLUMN CHARACTERISTICES

```
DSNU615I  -DSNA 182 05:15:18.64 DSNUSUCO - SYSCOLUMNS CATALOG STATISTICS FOR CIT
                                COLCARD           = 55
                                COLCARDF          = 5.5E+01
                                HIGH2KEY          = X'00E3D9C5D5E3D6D5404040404040
                                LOW2KEY           = X'00C1D5D5C1D7D6D3C9E240404040
DSNU615I  -DSNA 182 05:15:18.64 DSNUSUCO - SYSCOLUMNS CATALOG STATISTICS FOR STR
                                COLCARD           = 36
                                COLCARDF          = 3.6E+01
                                HIGH2KEY          = X'00E5C9C140D9C1D4C1C4C1404040
                                LOW2KEY           = X'00C2C5D9D5C1D340D9C44B404040
DSNU615I  -DSNA 182 05:15:18.64 DSNUSUCO - SYSCOLUMNS CATALOG STATISTICS FOR MAN
                                COLCARD           = 35
                                COLCARDF          = 3.5E+01
                                HIGH2KEY          = X'00E8C1D4D6E3D640404040'
                                LOW2KEY           = X'00C2D9D6E6D54040404040'
```

INDEX CHARACTERISTICES

```
DSNU618I  -DSNA 182 05:22:13.73 DSNUSUIP - SYSINDEXPART CATALOG STATISTICS FOR T
                                CARD              = 114
                                CARDF             = 1.14E+02
                                NEAROFFPOS        = 19
                                NEAROFFPOSF       = 1.9E+01
                                FAROFFPOS         = 0
                                FAROFFPOSF        = 0.0E0
                                LEAFDIST          = 66
                                LEAFNEAR          = 2
                                LEAFFAR           = 0
                                SPACE             = 720
                                SPACEF            = 7.2E+02
                                DSNUM             = 1
                                EXTENTS           = 1
                                P_DEL_ENT         = 0
                                PQTY              = -1
                                SQTY              = -1
```

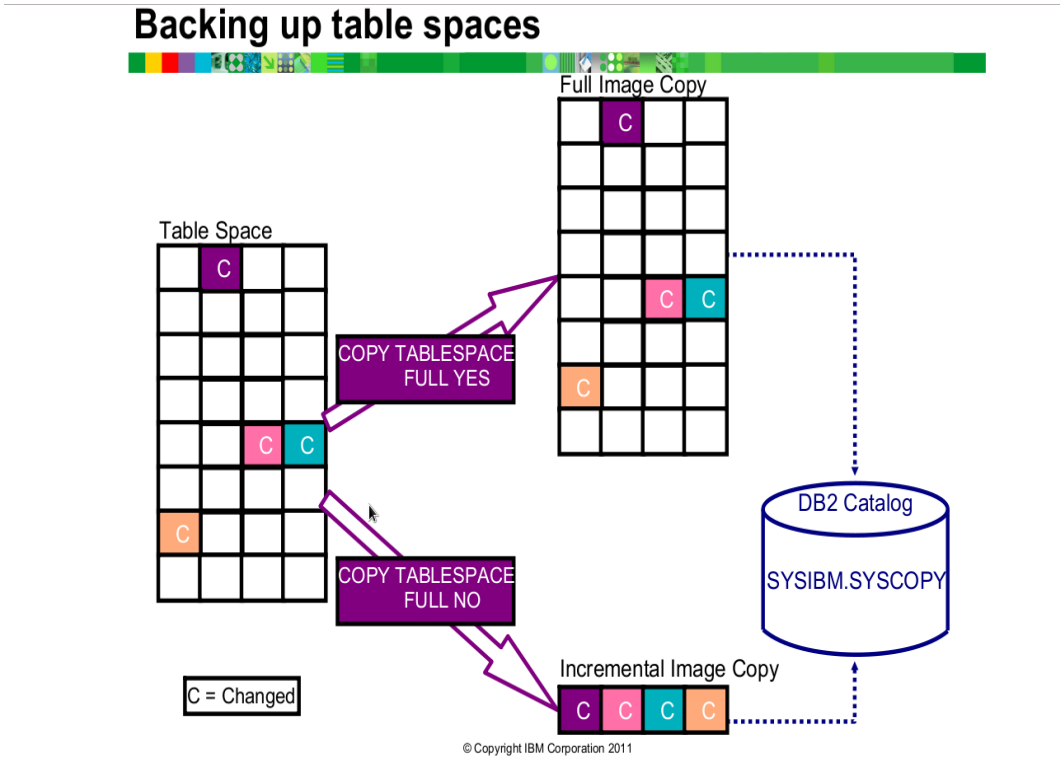
```
DSNU617I  -DSNA 182 05:22:13.74 DSNUSUIX - SYSINDEXES CATALOG STATISTICS FOR TSO
                                CLUSTERED         = Y
                                CLUSTERRATIO      = 100
                                CLUSTERRATIOF     = 1.0E+00
                                FIRSTKEYCARD      = 48
                                FIRSTKEYCARDF     = 4.8E+01
                                FULLKEYCARD       = 89
                                FULLKEYCARDF      = 8.9E+01
                                NLEAF             = 3
                                NLEVELS           = 2
                                SPACEF            = 7.2E+02
                                DATAREPEATFACTORF = 2.0E+00
```

```
DSNU616I  -DSNA 182 05:22:13.75 DSNUSUCD - SYSCOLDIST CATALOG STATISTICS FOR
                                          STATE,CITY
                                CARDINALITY =  6.9E+01
DSNU616I  -DSNA 182 05:22:13.75 DSNUSUCD - SYSCOLDIST CATALOG STATISTICS FOR
                                          STATE,CITY,STREET
                                CARDINALITY =  8.9E+01
DSNU616I  -DSNA 182 05:22:13.75 DSNUSUCD - SYSCOLDIST CATALOG STATISTICS FOR
                                          STATE
                                FREQUENCY           COLVALUE
                                ---------           --------
                                2.2807017543859E-01 X'00D5E8'
                                1.578947368421E-01  X'00C9D3'
                                7.8947368421052E-02 X'00C3C1'
                                3.5087719298245E-02 X'00C9D5'
                                2.6315789473684E-02 X'00D5C3'
                                2.6315789473684E-02 X'00D5D1'
                                1.7543859649122E-02 X'00C3E3'
                                1.7543859649122E-02 X'00C6D3'
```

9.7 COPY, COPYTOCOPY

The COPY is a online utility which creates copies of certain objects. It can creates up to five image copies, four sequential image copies, and one FlashCopy image of any of the following objects.

- Table space
- Table space partition
- Data set of a nonpartitioned table space
- Index space
- Index space partition



The two types of image copies are:

- Full Image copy
- Incremental image copy

Full image copy

A full image copy consists of all pages of the table space, whether changed or not. You specify FULL YES on the COPY utility control statement.

FULL YES is the default.

EX:
```
COPY TABLESPACE DSN8D10A.DSN8S10E
  FULL YES
```
(or)
```
COPY TABLESPACE DSN8D10A.DSN8S10E
```

Incremental image copy

An incremental image copy just contains the pages changed since the previous full or incremental image copy. FULL NO gives incremental copy.

EX:
```
COPY TABLESPACE DSN8D10A.DSN8S10E
  FULL NO
```

SYSIBM.SYSCOPY

- Image copies are sequential data sets.
- The fact that an image copy exists is recorded in SYSIBM.SYSCOPY table in the DB2 Catalog.
- Each image copy has a row in this table. Column ICTYPE specifies the type of image copy taken:
    - A full image copy has an ICTYPE of F.
    - An incremental image copy has an ICTYPE of I.

SYSIBM.SYSCOPY

- Information needed by recovery
- Description for some of the columns given below

Column Name	Description
DBNAME	Name of the database.
TSNAME	Name of the target table space or index space.
DSNUM	Data set number within the table space. For partitioned table spaces, this value corresponds to the partition number for a single partition copy, or 0 for a copy of an entire partitioned table space or index space. It is explained in upcomming page
ICTYPE	F → COPY FULL YES I → COPY FULL NO Q → QUIESCE
START_RBA	For ICTYPE I or F, the starting point for all updates since the image copy was taken
FILESEQNO	Tape file sequence number of the copy.
DEVTYPE	Device type the copy is on.
TIMESTAMP	The date and time when the row was inserted.
JOBNAME	Job Name of the utility
.....	.....

Partitioned table spaces

You can create image copies for individual partitions by specifying the partition number via the DSNUM keyword of the COPY utility control statement. DSNUM integer identifies the partition within the partitioned table space to be copied.

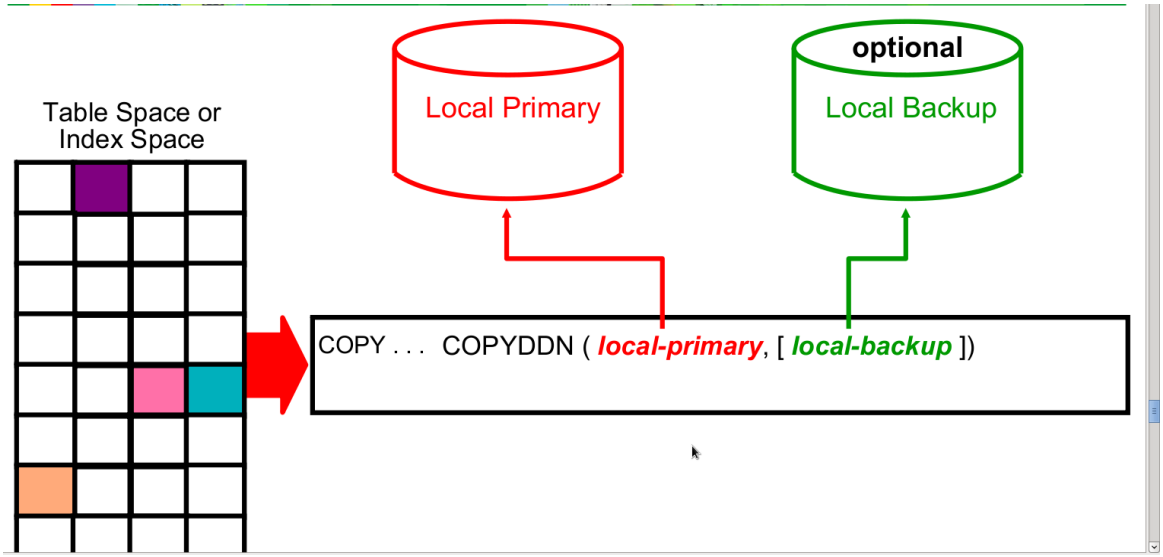
Non-partitioned table spaces

You can create image copies for individual data sets by specifying the data set number via the DSNUM keyword of the COPY utility control statement. DSNUM integer identifies the data set number within the non-partitioned table space to be copied.

COPYDDN

COPYDDN Keyword holds DDNAME or Template of the dataset where data is copied as shown in below JCL .By default COPYDDN is ddname SYSCOPY.

```
//STEP1   EXEC DSNUPROC,UID='IUJMU111.COPYTS',UTPROC='',
//             SYSTEM='DSN'
//SYSCOPY DD   DSN=COPY001F.IFDY01,UNIT=SYSDA,
//            VOL=SER=CPY01I,SPACE=(CYL,(15,1)),
//            DISP=(NEW,CATLG,CATLG)
//SYSIN   DD *
       COPY TABLESPACE DSN8D10A.DSN8S10E
/*
```



- DB2 gives you the capability to take multiple copies , By means of then COPYDDN parameter of the COPY utility control statement.
- If an image copy is not available because it has been destroyed, then DB2 uses the second copy. This avoids lengthy recoveries from the log.
- Here 1st copy refer as Primary copy & optional copy refer as Backup copy.

Example:

Instead of defining the data sets in the JCL, you can use templates. In the following example. In this example, the name of the template is LOCALDDN.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU111.COPYTS',UTPROC='',
//              SYSTEM='DSN'
//SYSIN    DD *
   TEMPLATE LOCALDDN UNIT SYSDA DSN(COPY001F.IFDY01)
                     SPACE(15,1)CYL DISP(NEW,CATLG,CATLG)
   COPY TABLESPACE DSN8D10A.DSN8S10E
   COPYDDN(LOCALDDN)
/*
```

RECOVERYDDN (ddname3,ddname4)

- Specifies a DD name or a template name for the primary (ddname3) and backup (ddname4) copy data sets for the image copy at the recovery site.
- It will use when any disaster occur

FLASHCOPY NO/YES/CONSISTENT

- Specifies whether FlashCopy technology is used to create a copy of the object.
- When FlashCopy is used, a separate VSAM data set is created for each partition or piece of the object.
- If the input data set is less than one cylinder, FlashCopy technology might not be used for copying the objects regardless of the FLASHCOPY settings.
- the COPY utility might not use FlashCopy even though YES is specified when FlashCopy Version 2 disk volumes are not available
- If SHRLEVEL CHANGE is not specified, specifying FLASHCOPY CONSISTENT is the same as specifying FLASHCOPY YES.
- Specifies that when SHRLEVEL CHANGE is also specified,CONSISTENT cannot be specified when copying objects that have been defined with the NOT LOGGED attribute. If CONSISTENT is specified for an object that is defined with the NOT LOGGED attribute, the COPY utility does not make a copy of the object and issues message DSNU076I with return code 8.

FCOPYDDN (template-name)

- Specifies the template to be used to create the FlashCopy image copy data set names. If a value is not specified for FCCOPYDDN,the value specified on the FCCOPYDDN subsystem parameter determines the template used.

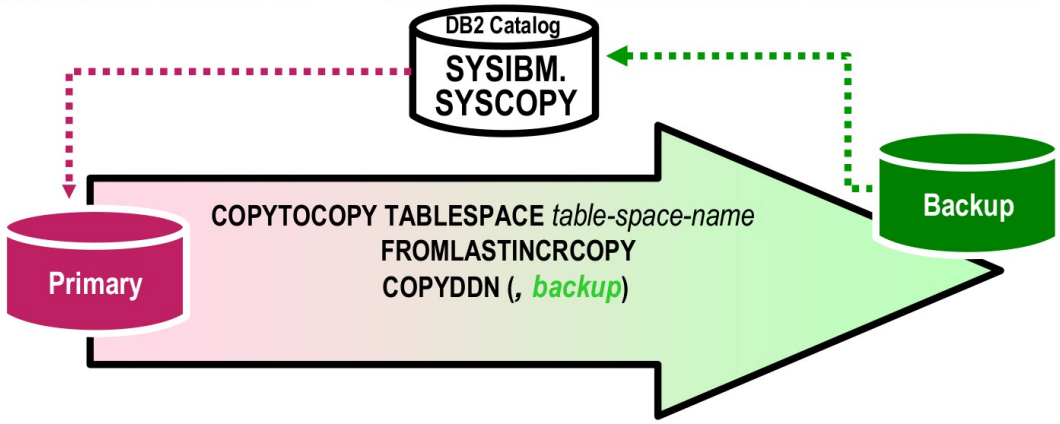
Example:Making full image copies for local site and recovery site

```
OPTIONS EVENT(ITEMERROR,SKIP)
COPY TABLESPACE DSN8D81P.DSN8S81C
COPYDDN(COPY1,COPY2)
RECOVERYDDN(COPY3,COPY4)
PARALLEL(2)
```

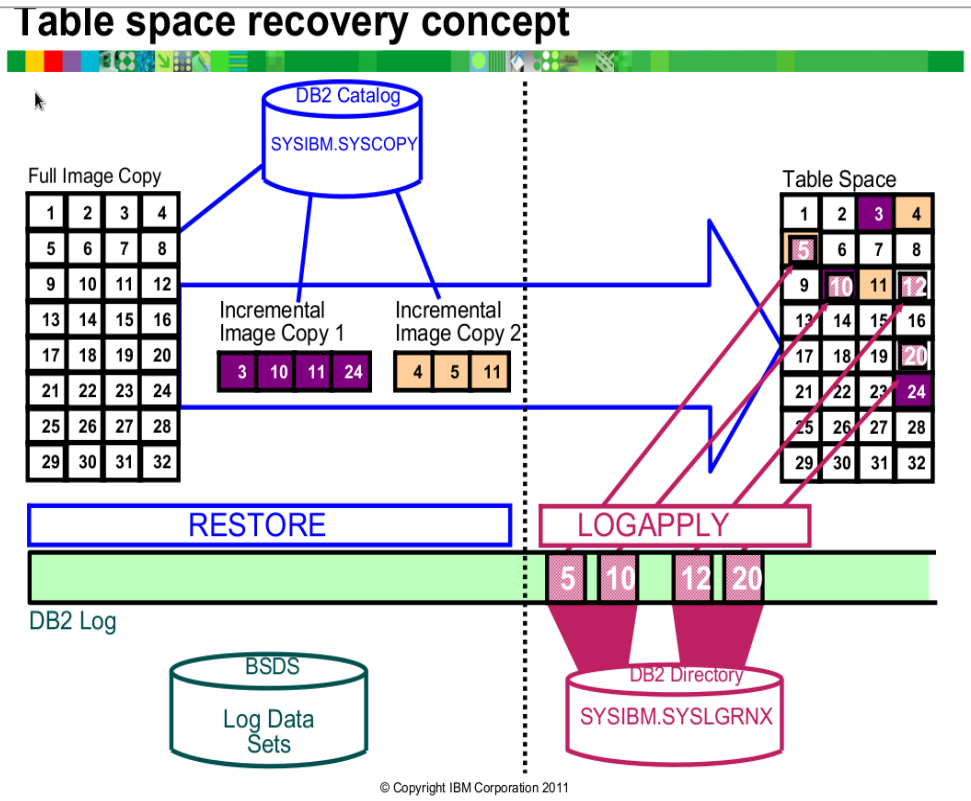
- The PARALLEL option indicates that up to 2 objects are to be processed in parallel.
- The OPTIONS statement at the beginning indicates that if COPY encounters any errors (return code 8) while making the requested copies, DB2 ignores that particular item. COPY skips that item and moves on to the next item. For example, if DB2 encounters an error copying the specified data set to the COPY1 data set, DB2 ignores the error and tries to copy the table space to the COPY2 data set.

COPYTOCOPY utility

You can establish only the primary copy during the utility execution and create the missing backup copy later by means of the COPYTOCOPY utility.



9.8 RECOVER



When recovering a table space, the RECOVER utility performs 2 major phases,

RESTORE phase (MERGE): During the RESTORE phase, it merges the full image copy with its incremental image copies to form the basis for the restored table space in proper order of timestamp. By means of the rows in catlog table SYSIBM.SYSCOPY_

LOGAPPLY phase: During the LOGAPPLY phase, the appropriate log records are read from the DB2 log and applied to the restored table space.

with the help of directory table SYSIBM.SYSLGRNX

incremental copies reduces recovery time

Example 1

```
RECOVER TABLESPACE DB01.TS01
```

Example 2

```
RECOVER TABLESPACE DB01.TS01
        TABLESPACE DB01.TS02
```

Example 3

```
RECOVER TABLESPACE  DB01.TS01 DSNUM 1
        TABLESPACE  DB01.TS01 DSNUM 2
        TABLESPACE  DB01.TS01 DSNUM 3
```

Example 4

```
RECOVER TABLESPACE DB01.TS01 DSNUM ALL
```

Note

- a device is damaged, but data is good,so data is not available we want to move all of the data, as it is at the current time, to a different device. This is called <u>forword or full recovery</u>.
- <u>Point-in-time recovery or partial recovery</u> is used when data has been logically damaged, normally by a DB2 application program with flawed logic. The goal for logically-damaged data is to recover the data to a point-in-time just prior to where the processing that caused the logical damage occurred. --> for this we required ROLLFORWARD utility
- <u>Version recovery</u> is the restoration of a previous version of the database, using an image that was created during a backup operation.

9.9 REBUILD

- The REBUILD INDEX utility allows you to rebuild an index from scratch using the data in the table on which the index is based.
- You can rebuild the entire index, a partition of a partitioned index, or a logical partition of a nonpartitioned index. A logical partition always corresponds to a physical partition of the associated partitioned table space.

Example 1: Rebuild a single index

REBUILD INDEX (DSN8710.XDEPT1)

Example 2: Rebuild all partitions of a partitioning index

REBUILD INDEX (DSN8710.XEMP1)

Example 3: Rebuild multiple partitions of a partitioning index

REBUILD INDEX (DSN8710.XEMP1 PART 2, DSN8710.XEMP1 PART 3)

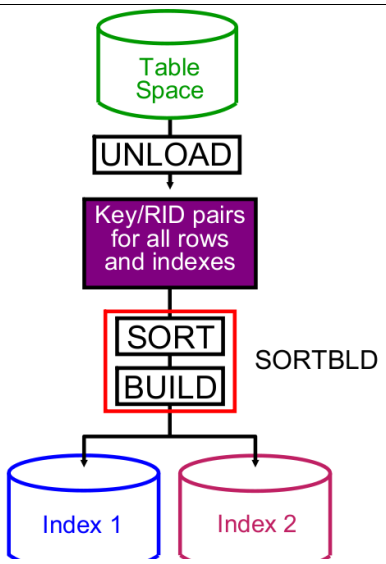
Example 4: Rebuild all indexes on a partitioned table space

REBUILD INDEX (ALL) TABLESPACE DB01.PTS01

Example 5: Rebuild one index partition specifying index space name

REBUILD INDEXSPACE (DB1.IS1 PART 9)

REBUILD Phases



1. UTILINIT :Performs initialization and setup.
2. UNLOAD :Unloads index entries.
3. SORT: Sorts unloaded index entries.
4. BUILD :Builds indexes.
5. SORTBLD :Sorts and builds a table space
6. UTILTERM :Performs cleanup.

Rebuild considerations

REUSE option avoids deletion and redefinition of data sets

If you are rebuilding an index because of media failure:

- Do not use the REUSE option
- Remove defective volume from storage group before rebuilding the index

9.10 DSN1COPY

- It is normal copy of VSAM Linear dataset to Sequential dataset or viceversa. We can copy VSAM to VSAM or sequential to sequential
- To use this Copy as recovery we have tranform OBID

```
//DSN1        EXEC  PGM=DSN1COPY,PARM='FULLCOPY'
//SYSPRINT    DD    SYSOUT=*
//SYSDUMP     DD    SYSOUT=*
//SYSUT1      DD    DSN=....................., DISP=SHR
//SYSUT2      DD    DSN=....................., DISP=OLD
```

→RESET: rest the page number in header page to zero

→ OBIDXLIT: this parameter is used to transulate OBID

- When db2 is ina active then we want to apply DSN1COPY then following steps have to follow

▪ Start the table space as read-only by using START DATABASE.

▪ Run the QUIESCE utility with the WRITE (YES) option to externalize all data pages and index pages.

▪ Run DSN1COPY with DISP=SHR on the data definition (DD) statement.

9.11 QUIESCE

The QUIESCE utility records a quiesce point for a table space, partition, table space set, or list of table spaces and table space sets.

A quiesce point is a consistent point at which data is consistent across these objects. You can later recover a table space to its quiesce point by using the RECOVER utility.

When you request that the QUIESCE utility take a quiesce point, the quiesce point is the current log RBA or log record sequence number (LRSN). then records the quiesce point in the SYSIBM.SYSCOPY catalog table.

```
QUIESCE TABLESPACE DSN8D11A.DSN8S11D
        TABLESPACE DSN8D11A.DSN8S11E
        TABLESPACE DSN8D11A.DSN8S11P
```

Don't RUN

You cannot run QUIESCE on a table space that is in COPY-pending, CHECK-pending, RECOVER-pending, or auxiliary CHECK-pending status. It may generate max return code 8

TERMINATE

If you use TERM UTILITY to terminate QUIESCE when it is active, then QUIESCE releases the drain locks on table spaces. You can restart a QUIESCE utility job, but it starts from the beginning again.

Note:

A QUIESCE flushes all DB2 buffers on to the disk. This gives a correct snapshot of the database which mean a consistant state.

9.12 DIAGNOSE

The DIAGNOSE online utility generates information that is useful in

diagnosing problems.

DIAGNOSE DISPLAY MEPL

First of all, MEPL is an acronym for Module Entry Point List (MEPL).

Termination or restart of DIAGNOSE

You can terminate a DIAGNOSE utility job by using the TERM UTILITY command if you submitted the job or have SYSOPR, SYSCTRL, or SYSADM authority. You can restart a DIAGNOSE utility job, but it starts from the beginning again.

Force a DUMPS

The following control statement forces a dump for any utility abend that occurs during the execution of the specified COPY job. The DIAGNOSE END option ends DIAGNOSE processing.

```
EX:1  DIAGNOSE ALLDUMPS
      COPY TABLESPACE DSNDB06.SYSDDF
      DIAGNOSE END
```

```
EX:2  DIAGNOSE TYPE(66,32)   → Type of diagnose
      REBUILD INDEX (IDOS0302...., IDOS0304..., IPOS0301..)
      DIAGNOSE END
```

9.13 REPAIR

The REPAIR online utility repairs the corrupted data. Corrupted may at datapages or space map pages or dictionary page or index entries. Be extremely

careful when using REPAIR. Improper use can damage the data even further.

Before starting to use REPAIR to change data, ensure that you have a copy (full image copy or DSN1COPY generated copy) of the affected table space to enable fallback.

Do not run REPAIR on encrypted data. REPAIR does not decrypt the data.

REPAIR option descriptions

option	descriptions
OBJECT	Indicates that an object is to be repaired. This keyword is optional.
LOG YES/NO	REPAIR LOG NO can override the LOG YES attribute of a table space. REPAIR LOG YES cannot override the LOG NO attribute of a table space.
SET TABLESPACE	Resets the COPY-pending, RECOVER-pending, CHECK-pending
SET INDEX	Resets RECOVER-pending, CHECK-pending, REBUILD-pending

```
REPAIR OBJECT
SET INDEX (ALL) TABLESPACE DBNI1601.TSNI1601 NORBDPEND
SET INDEX (ALL) TABLESPACE DBNI1601.TSNI1602 NORBDPEND
SET TABLESPACE DBNI1601.TSNI1601 PART 1 NOAUXCHKP
SET TABLESPACE DBNI1601.TSNI1602 PART 4 NOAUXCHKP
```

(we can apply if there is no proper syn b/w catlog and direcory-naseer)

10. DB2 commands

9.1 System Related Commands

No.	Command	Description
1	START DB2	
2	SET LOG	
3	STOP DB2	
4	ARCHIVE LOG	
5	DISPLAY LOG	
6	DISPLAY GROUP	
7	DISPLAY ARCHIVE	
8	RECOVER BSDS	
9	SET ARCHIVE	
10	SET SYSPARM	

9.2 Database Related Commands

No.	Command	Description
1	START DATABASE	
2	DISPLAY DATABASE	
3	STOP	

	DATABASE	

9.3 Buffer pool Related Commands

No.	Command	Description
1	DISPLAY BUFFERPOOL	
2	DISPLAY GROUPBUFFERPOOL	
3	ALTER BUFFERPOOL	
4	ALTER GROUPBUFFERPOOL	

9.2 Utility Related Commands

No.	Command	Description
1	DISPLAY UTILITY	
2	ALTER UTILITY	
3	TERM UTILITY	

9.3 Threads Related Commands

9.4 Procedure Related Commands

9.5 Function Related Commands

9.6 Trace Related Commands

9.7 DDF Related Commands

9.8 DSN Related Commands

RUN PROGRAM (A) PLAN(A)

RUN PROGRAM (A)

if plan name & program name is same then there is no need to mentioned PLAN in RUN command in LUW. In Z/OS we have to mention both

The EXPLAIN statement obtains information about access path selection. If we want to access same data every time then there is no need to calculate access path every time. We can know access path using this command

Display DDF details

System Related DB2 commands

-START DB2

-STOP DB2

-DISPLAY LOG

-DISPLAY ARCHIVE

-SET ARCHIVE

-SET LOG

-ARCHIVE LOG

-DISPLAY GROUP

-RECOVER BSDS

-SET SYSPARM

Database Related :

-START DATABASE

-DISPLAY DATABASE

-STOP DATABASE

Threads Related

Bufferpool related

-DISPLAY BUFFERPOOL
-DISPLAY GROUPBUFFERPOOL
-ALTER BUFFERPOOL
-ALTER GROUPBUFFERPOOL

-DISPLAY THREAD
-CANCEL THREAD
-RECOVER INDOUBT
-RECOVER POSTPONED
-RESET GENERICLU
-RESET INDOUBT

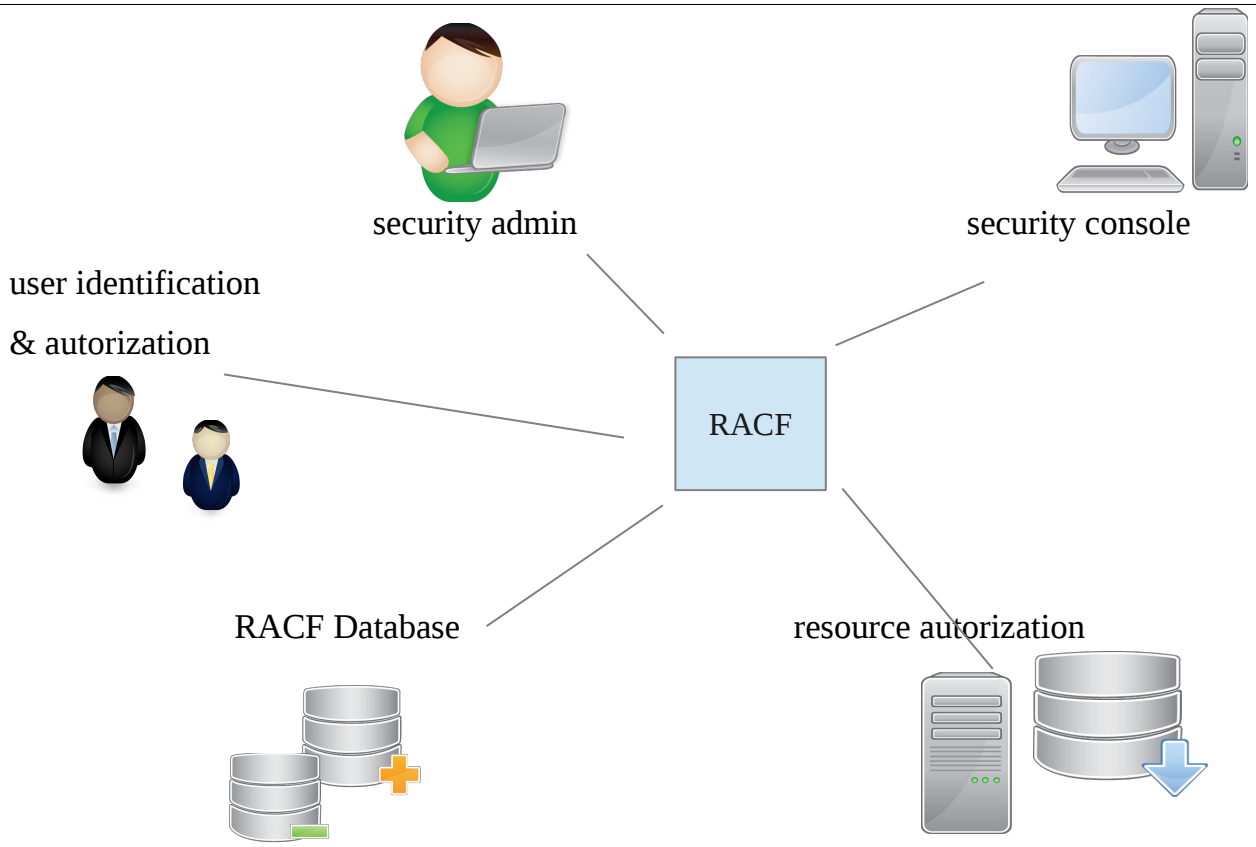
Utility Related

-DISPLAY UTILITY
-ALTER UTILITY
-TERM UTILITY

11. DB2 – Security

10.1 RACF

Resource Allocation Control Facility (RACF) is an optional ESM (External Security Manager) software product included as a part of Z/OS to provide security for operating system



RACF Functions

- validation of user request
- controls access of resources to user as per the restrictions for that users
- security admin & console for configuring & monitoring
- hold users info & their activities in database

10.2 Who is User?

Before a user can access a DB2 object, DB2 goes to the catalog and checks to see if the user has the authority or privilege to do what they are trying to do.

1. Link establishment from different sub-systems with DB2 refered as CONNECTIONS

2. before connection establishment there are two validations will occur.

- RACF
- AUTH EXIT Routines

3. Every process that connects to or signs on to DB2 is represented by a set of one or more DB2 identifiers called Authorization IDs.

4. Authorization IDs can be assigned to a process connecting DB2 by default procedures or user-written exit routines called USER AUTHORIZATION ROUTINES. These are two types.

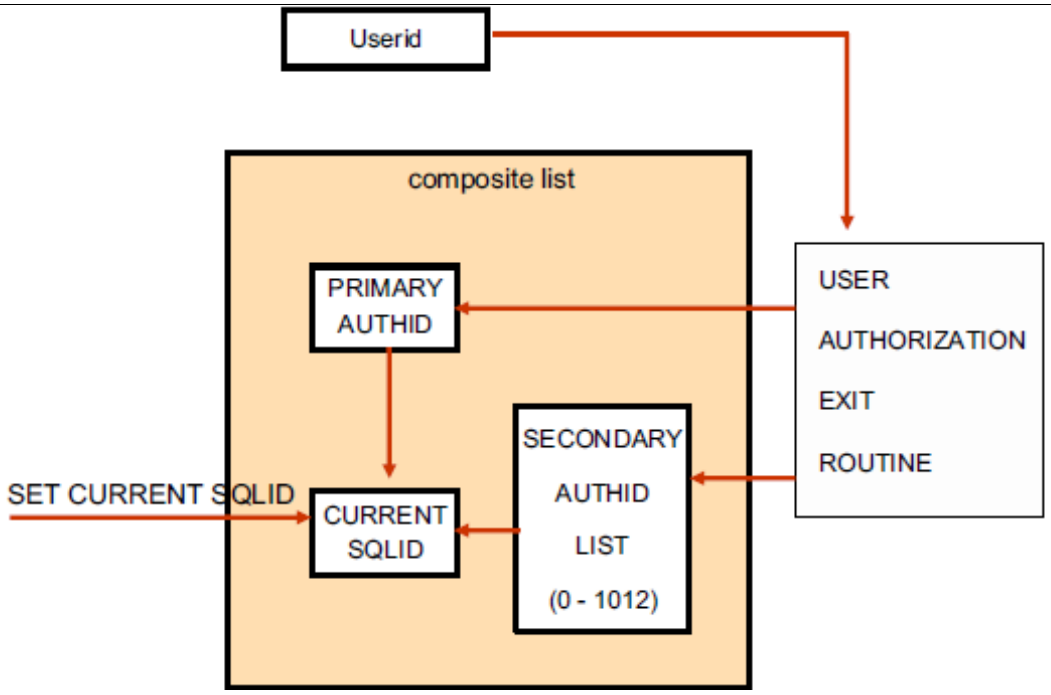
- CONNECTION EXIT ROUTINE
- SIGNON EXIT ROUTINE

5. Every process which connects DB2 is assigned one mandatory ID called the primary authorization ID.

6. In addition with primary authorization ID it can provide 1012 optional group related id's called Secondary authorization ID's

7. If TSO connecting DB2 subsystem then the process is called CONNECTION PROCESSING and routine required for that is CONNECTION EXIT ROUTINE (DSN3@ATH). This routine establishes the TSO log in ID as PRIMARY AUTHID and the initial value of the CURRENT SQLID, and can provide up to 1012 SECONDARY AUTHIDs.

8. Similarly, requests from CICS or IMS transactions go through SIGNON PROCESSING, which drives a SIGNON EXIT ROUTINE(DSN3@SGN).



9. USER special register always refers to your PRIMARY AUTHID. Cannot be changed.

10. CURRENT SQLID special register , this is set to either your primary authid or one of your secondary authids. (Except for SYSADM / SYSCTRL) but initially set to PRIMARY AUTHID.

11. SYSADM can set value using statement SET CURRENT SQLID='.....' before that DSNZPARM SEPARATE_SECURITY is set to NO.

12. If an invalid ID is specified for SET CURRENT SQLID, a -553 return code results and the SQLSTATE is set to 42503.

13. If you issue a CREATE statement dynamically, and if the name of the table, view, index, alias, or synonym you are creating is unqualified, then the owner of the created object is your CURRENT SQLID.

14. Accounding will happened on PRIMARY AUTHID

15. CURRENT SQLID not used by Static SQLand it is default qualifier of created objects.

16. With secondary IDs, individuals have no GRANTs done to their primary authorization ID

10.3 Previlages

DB2 internal security determines whether or not a particular user or application can:

- Define, modify or delete objects in the DB2 catalog
- Execute DB2 programs (plans and packages)
- Run DB2 utilities
- Access tables with SQL
- Access certain columns in a table with SQL
- What kind of SQL can be executed on a DB2 table
- Use specific buffer pools
- Issue DB2 commands

Owner/Implicit tablePrivilages

The individual who CREATE's an object is referred to as the owner, and the authorization ID used at the time of the CREATE becomes the qualifier for the table name. The owner is automatically granted privileges such as:

- ALTERing or DROPping that table, when changes are needed
- Issuing DML statements on the table

- Adding additional objects related to the table(indexes, constraints, etc.)
- Running DB2 utility programs on the table data

Explicit table privileges

- Explicit table privileges are given via a GRANT SQL statement. Privileges can be taken away with the REVOKE SQL statement.

Indirect privileges

- we have authority on package or plan. even we did't have any authority (GRANT) on sql statements includend in plan or package. Since we have authority on plan/package we can execute it – Static SQL
- as a invidual we does't have privilage but the group we joined has privilage with coressponding group ID

DB2: Authorization vs Authentication

Authentication is the process by which a system verifies a user's identity for security purose to enter into (DB2 sub-)system

After a user is authenticated(after entering ino system(db2 sub-system)), the database manager determines if that user is allowed to access DB2 data and resources or not. It is nothing but checking autherization.

10.4 Administrative authorities

Authorities are two types . One category is installation authorities controlled by DSNZPARM settings. Another category is Grantable authorities controlled by use of GRANT and REVOKE statements.

– Installation authorities

1. Install SYSADM

2. Install SYSOPR

3. SECADM

– Grantable authorities

4. SYSADM

5. SYSCTRL

6. SYSOPR

7. DBADM

8. DBCTRL

9. DBMAINT

10. PACKADM

11. System DBADM

12. ACCESSCTRL

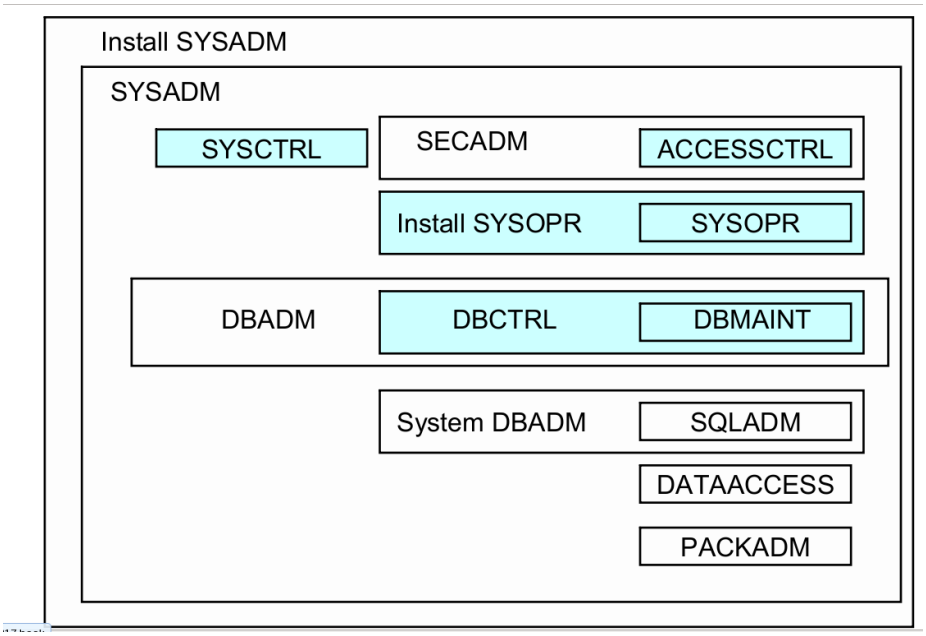
13. DATAACCESS

14. SQLADM

DSNZPARM SEPARATE_SECURITY:

Users with SYSADM authority can manage all security objects, if it make NO, SYSADM and SYSCTRL authorities to remain unchanged. So this concepts are valid. is only valid if the SEPARATE_SECURITY is set to NO.

Acceptable values:	NO, YES
Default:	NO



DATAACCESS

1. SELECT previlages on all catallog table
2. DML previlages on updatable catalog tables except SYSAUDITPOLICIES
3. DML Privileges on all user tables, views, and MQTs
4. Execute Privileges on all plans, packages, and routines
5. Privileges on all JARs & Sequences

PACKADM

Privileges on a collection: CREATEIN

Privileges on all packages in a collection: BIND COPY EXECUTE

SECADM

ALTER CREATE DROP Privileges on security-related objects

ACCESSCTRL

1. SELECT previlages on all catallog table
2. DML previlages on updatable catalog tables except SYSAUDITPOLICIES
3. GRANT & REVOKE privilages

DBADM: Privileges on a database

ALTER DELETE INDEX INSERT

REFERENCES SELECT TRIGGER UPDATE

DBCTRL: Privileges on a database

DROP LOAD RECOVERDB REORG REPAIR

DBMAINT: Privileges on a database

CREATETAB CREATETS DISPLAYDB

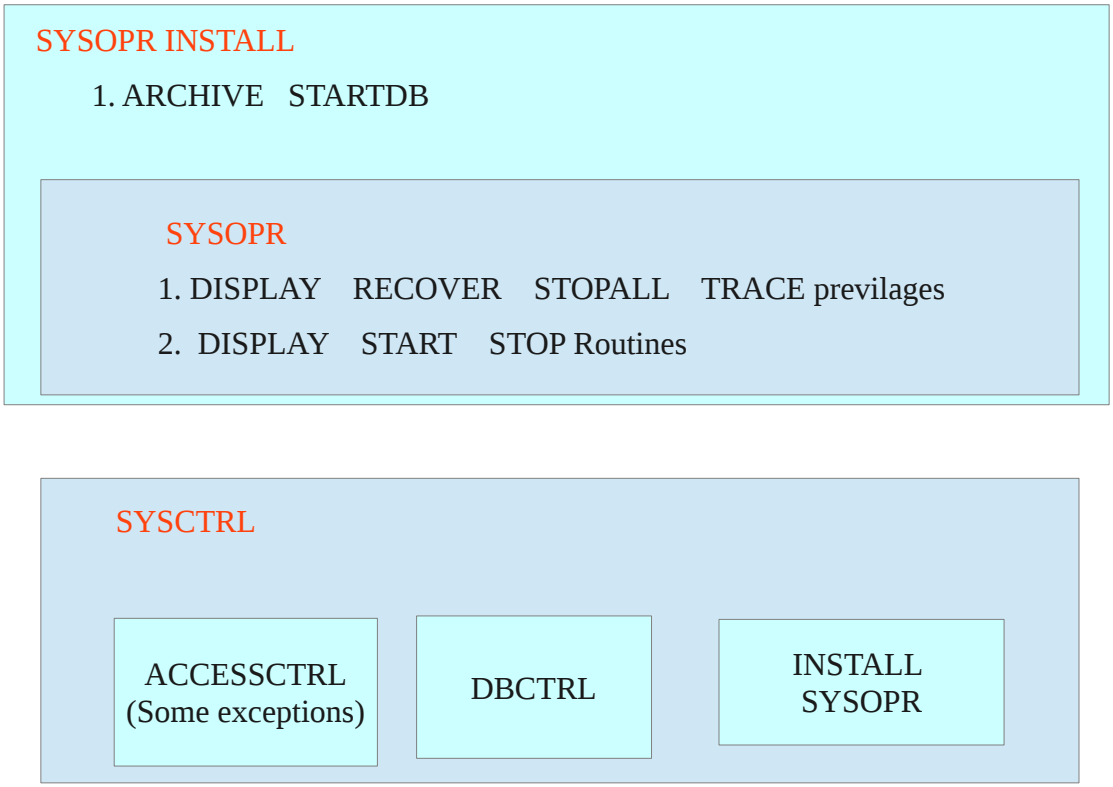
IMAGCOPY STATS STARTDB STOPDB

SYSTEM DBADM

1. Previlages on plans and packages
2. Previlages on all user defined databases
3. Previlages on all schemas & sequences

SQLADM

1. SELECT previlages on all catallog table
2. DML previlages on updatable catalog tables except SYSAUDITPOLICIES
3. EXPLAIN MONITOR1 MONITOR2 (System previlages)



SYSCTRL

- <u>he does't have authority to give authority</u>

- Bind & copy privilages on plan & packages but he can't execute them
- privilages on BUFFERPOOLS, STOGROUP, TABLESPACE
- Privilage on all catlog tables for SELECT
- Privilages for ALTER, TRIGER
- Privilage on updatable catlog table for INSERT, UPDATE, DELETE execept on table called SYSIBM.SYSAUDITPOLICIES

Install SYSADM

- Install SYSADM One or two IDs are assigned this authority when DB2 is installed. They have all the privileges of SYSADM.
- Authority is not recorded in the DB2 catalog. The catalog need not be available to check installation SYSADM authority
- No ID can revoke this authority

Note:

- System control authority (SYSCTRL) is the highest level in System control authority. It provides to perform maintenance and utility operations against the database manager instance and its databases.
- System administration authority (SYSADM) is highest level of administrative authority at the instance level. They Change the database manager configuration file (including specifying the groups having SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority) automatically granted ACCESSCTRL, DATAACCESS, DBADM and SECADM authority on the database.

11. Roles in DB2 world

The members in DB2 family is

- Designer
- System Admin
- Database Admin
- Programmer
- Operator
- End user

Role of Designer

- study business info and entities involved
- pictorial representation of the user's view which is entity relationship digram
- identify differnt senarios like how to implement as simple as possible, will optimistic,where we have to use foreign keys.....

Role of DB2 System Admin

System administrators may have responsibilities over many DB2 systems.

Product installation

- Study the documentation.
- Check all the hardware and software prerequisites.
- Plan DASD space requirements.
- Verify the installation.

Creator for system objects

The system programmer is normally the person responsible for defining certain system-level objects (such as storage groups, databases, etc.)

Once those objects are in place, the database administrator (DBA) is usually the one who will create those DB2 objects that are related to the business applications (tables, indexes, views, triggers, etc.)

Grantor for initial previleges

Once the system-level objects have been created, the system administrator can GRANT[sql] needed privileges to DBAs. Then DBAs can GRANT[sql] privileges on objects to users, for those objects that those users are authorized to access.

Responsible for the overall performance of DB2

- System administrators will typically use software tools like Omegamon for DB2 in order to assist them in measuring overall performance and detecting performance problems within the DB2 environment.

Instance

- Instance is a logical enviorement which DB2 works
- To install DB2 on LUW, the system programmer must first run the install .exe in order to establish what is known as a DB2 instance.
- This Custamization settings is accomplished through a database manager configuration file
- Within an instance of DB2, multiple databases may be configured. In DB2 for Linux, UNIX, and Windows, each database is considered as a separate entity, and each has its own catalog, Logs, set of bufferpools, etc.
- Each database undergoes further customization by means of a database configuration file.
- In DB2 LUW, in order to accomplish joins of tables within separate databases, it is necessary to federate the databases.

Role of DBA

- the database administrator (DBA) is usually the one who will create those DB2 objects that are related to the business applications (tables, indexes, views, triggers, etc.)
- Grant privileges to user to insert or update or change data (INSERT, SELECT, UPDATE, DELETE, LOAD, etc.)
- Maintenance of database objects application point of view , making backups recovering data if needed
- Binding plan & package (BIND)

Role of Programmer

- code programes and stored procedures
- Programmers physically embed SQL statements into host language programs.
- Bind packages & plan
- create test enviorement before going to test enviorement

Role of operator

- Operators normally do not have data access authority
- Operators take some of the maintenance burden from DB2 system administrators and DBAs
- utility and application programs were developed and tested by the DBAs and application programmers. If any problem arises during the execution the operator will contact them

Role of End users

- use developed and tested software for his business application

- if needed update or delete or insert or read data through SQL commands
- but he can do any operation on database iff coresponding previlages grant to him.
- Produce reports using that application

Interactive SQL

Interactive SQL refers to SQL statements that you submit to DB2 by using SPUFI (SQL processor using file input), the command line processor, or by using query tools, such as DB2 QMF(quey managment facility, it is licence).

☒ DB2 for Linux, UNIX, and Windows

– Command Line Processor (CLP)

– IBM Data Studio

– Optim Database Administrator

☒ DB2 for z/OS

– SPUFI

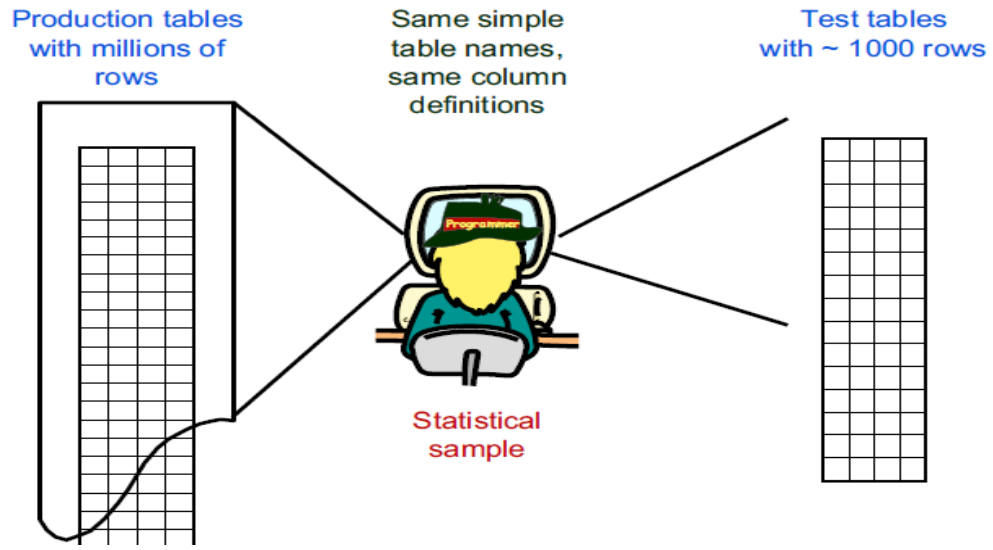
– QMF

Note:

--> STAFF & ORG tables are sample tables comes with QMF which is used to practice Sql statements. This STAFF table is available in many DB2 platforms. This tables are used for IVP (Installation verification process)

--> QMF procedures are set of QMF comands which can enter one command per line (SQL not allowed)

Create Test Envioroment



- Production tables with millions of rows implemented with Test tables with ~ 1000 rows
- but we have to implement with simple table names, same column definitions same indexes and views — everything about the test tables and the production tables should be the same.
- if production tables & row tables are in same database then we have to change schema

This can be accomplished by the following SQL statements:

1. CREATE TABLE new_table_name LIKE old_table_name
2. INSERT INTO new_table_name SELECT * FROM old_table_name WHERE clause

Testing

1. All possible forms of input are verified/validated
2. All logic paths in the code are exercised
3. The most commonly occurring error conditions are handled

Note: Although our test tables have very few rows, and DB2 will process

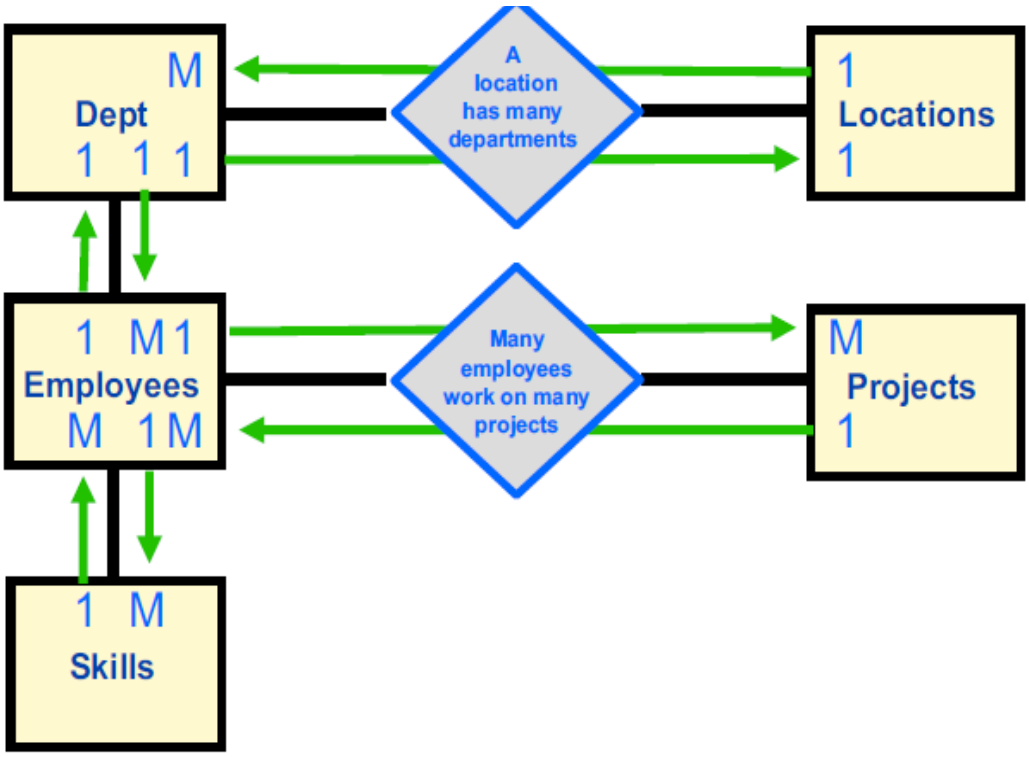
them very quickly when we run queries against the test tables, we want the DB2 Optimizer to make the same access strategy decisions as it would make when running against production data. i.e We want DB2 to think that the test tables contain production quantities of data. In other words, we are replicating the statistics from the production system to test.

13. Database Design

The aim of Database design includes refers as problem statement once poblem statement is given there are 3 views we have to observe

- Conceptual view: here we have make data inventory. (attributes, & their

values or range of values , Entity relationship model,Draw lines between related entities and notate the relationships. Relationships between business entities can be one-to-one, one-to-many, or many-to-many.)



- Logical view: here we have make Process inventory. list all process which will affect our database ex: location is change, new emplyee is added...
- Storage view: list database objects such as integrity rules, tabless, indexes, triggers and views requires and we have to do normalization.

What is a business entity?

A business entity is something fundamental to the business that can be uniquely Identified. The formal representation of business entities and relation ship between them is called business model

1st form:

First Normal form is minimum requirement for every relation. In every attribute at every tuple must contain single value. Below statement shows voilation of this statement --> atomic values

ID	NAME	Phone no	Child
11	Phani	9440499102	Sai
12	Venkatadhri	9492770394	Dileep, sudheer
13	Rajesh	9440499102	Harinath, Sumanth, Deepu
14	Harsha	9492770394	Ahan
15	Praharsha	9492770394	-

Otherwords, there is no repeated columns in a table.

Otherwords, all the rows must have equal attributes

Below statement shows voilation of this statement

ID	NAME	Phone no	Child1	Child2	Child3
11	Phani	9440499102	Sai		
12	Venkatadhri	9492770394	Dileep	sudheer	
13	Rajesh	9440499102	Harinath	Sumanth	Deepu
14	Harsha	9492770394	Ahan		
15	Praharsha	9492770394	-		

Rajesh have 3 children if he hand one more after creation of table. Then what will you do, adding new coloumn for his fourth children???

Praharsha does't had children but lot of space is allocated for his child entities.

There is any alternative way to save this memory???

To eliminate these problems implement 1NF as shown in below

ID	NAME	Phone no
11	Phani	9440499102
12	Venkatadhri	9492770394
13	Rajesh	9440499102
14	Harsha	9492770394
15	Praharsha	9492770394

ID	SEQ	CHILD
11	1	Sai
12	1	Dileep
12	2	sudheer
13	1	Harinath
13	2	Deepu
13	3	summanth
14	1	ahan

Even through now praharsha had one children we can add as a single row to above table.

2nd form

it must be in 1NF, we have to avoid partial dependency. I.e Non key attribiutes must fully depend on primary key, Not on part of primary key. Below table shows voilation of above statement

Emp ID	Project ID	Name	Project	Mobile no
11	0A	HARINATH	SMART GLOVES	9492770394
12	0A	MAYA	SMART GLOVES	9440499102
13	0A	SUMANTH	SMART GLOVES	9492770394
14	0A	PHANI	SMART GLOVES	9440499102
11	0B	HARINATH	E-CYCLOPEDIA	9492770394
12	0B	MAYA	E-CYCLOPEDIA	9440499102
13	0B	SUMANTH	E-CYCLOPEDIA	9492770394
14	0B	PHANI	E-CYCLOPEDIA	9440499102

In the above table Emp ID, Project ID both are repeated values in its columns. So as individual can't provide uniqueness to row, both together provides uniqueness to row this 2 column together refer as Composite key.

Here Name fully & mobile no depend upon EMP ID only not on Project ID. Project name fully depend on ProjectID not on EMPID.

So all the attributes(Name, Project, Mobile no) not fully depend on key. Depend on part of key. This is called partial dependency.Now we will get 3 forms

Emp ID	Name	Mobile no
11	HARINATH	9492770394
12	MAYA	9440499102
13	SUMA	9492770394

	NTH	
14	PHANI	9440499102

Project ID	Project
0A	SMART GLOVES
0B	E-CYCLOPEDIA

EMP ID	Project ID
11	0A
12	0A
13	0A
14	0A
11	0B
....	....

3rd form

it must be in 2NF, we have to avoid transitive dependency.

All attributes depend on key in non transitive manner

Name	PIN	Location	State
Phani	212	Banglore	Karnataka
Venkatadhri	121	Hyderabad	TELANGANA
Rajesh	212	Banglore	Karnataka
Harsha	121	Hyderabad	TELANGANA
Praharsha	121	Hyderabad	TELANGANA

Summanth	111	Chennai	Tamil nadu
Deepu	111	Chennai	Tamil nadu

Name & Location has Direct dependancy. Location & State has Direct dependancy

Name & State has transitive dependancy

Name	PIN

PIN	Location	State

What is denormalization

- process of attempting to optimize the read performance by adding redundant data or by grouping data.
- Storage point of you Normalization is good but user doest need that table, he want his full pledged table. So we denormalization is compulsory while read operation
- if there is no concept of denormalization we have to do several joins operation to make actual full table

14. Information Management

1. What is unit of work or unit of recovery?

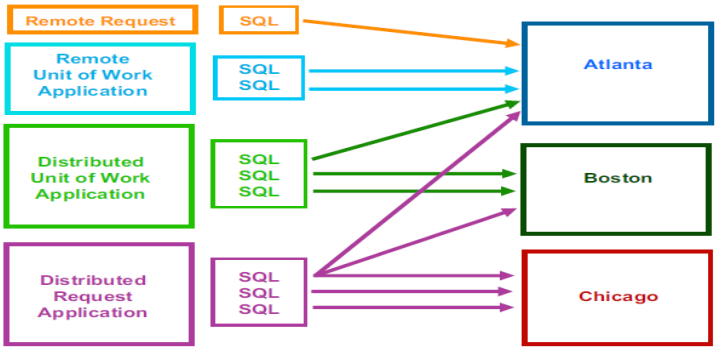
- It is a sequence of SQL operations that is handled in an all or nothing way.

- If all the operations are successful, changes to data are put into effect (RDMS performs COMMIT operation). If the any operations in that are unsuccessful, changes to data are removed (RDMS performs ROLLBACK)
- Logs are use to perform ROLLBACK operation

EX: sql statements include for fund transfer is a unit of work

2. Types of access for distributed data

Type of request	No of sql statements	No of server locations	Servers accesed per sql satatment
Remote Request	1	1	1
Remote Unit of Work	>1	1	1
Distributed Unit of Work	>1	>1	1
Distributed Request	>1	>1	>1



Remote Request : One SQL to one server

Remote unit of work: One unit of work(multiple sql) to one server

Distributed unit of work: Many sqls comes undder unit of work sent to many servers

Distributed request:will let you access data at multiple locations with a single SQL

3. DDF-Distibuted Data Facility ?

- Built-in component of DB2 for z/OS. Provides the connectivity to and from other databases over the network. i.e distributed database connections.
- DDF implements a full DRDA Application Server and Application Requester.
- DDF has developed thread management strategies to handle thousands of connections
- DDF runs as an additional address space in the DB2 subsystem *ssid***DIST**, which can communicate well with *ssid***DBM1**

4. Access methods for Distibuted data

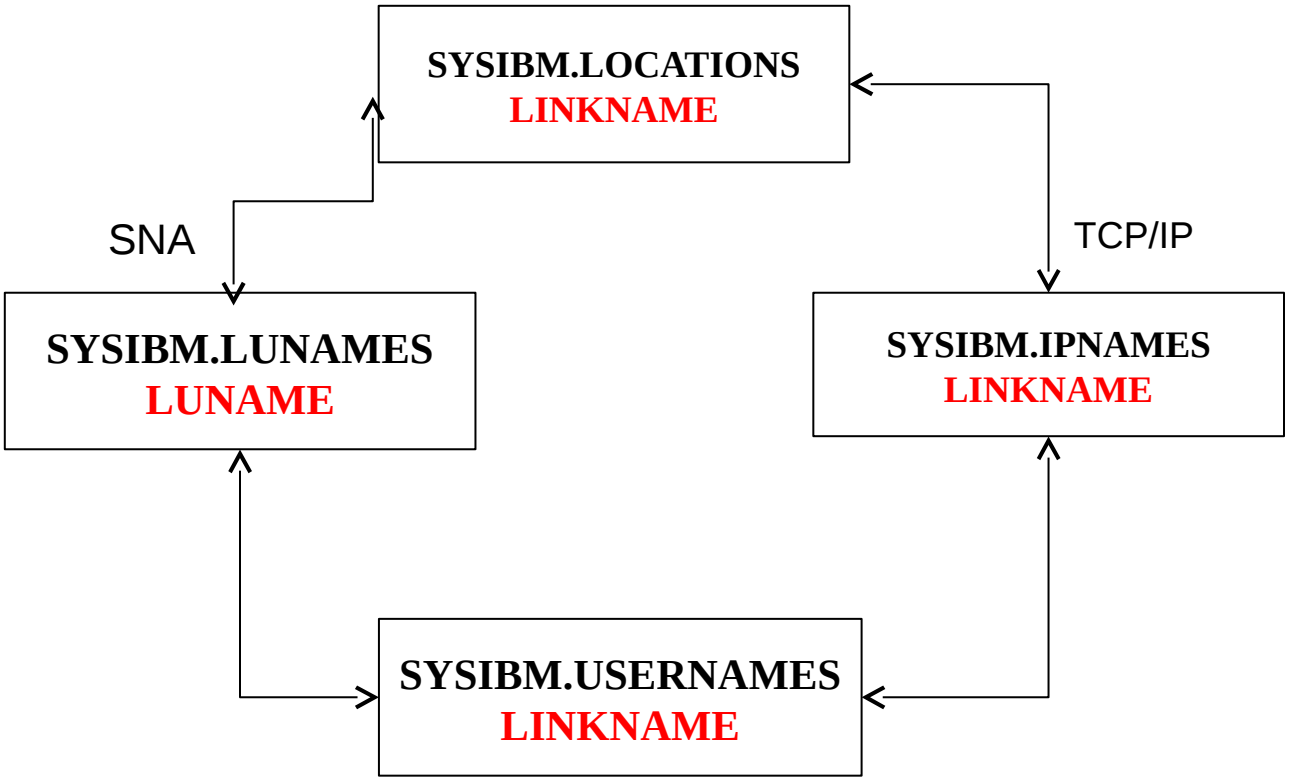
Private Protocal	Distributed Relational Database Architecture
No remote bind will posible. So it suppoet Dyamic Sql	Support both static & dynamic sql
Stored procedures will not support	Stored procedures will support
Support 3 part nSames only	Support both 3 part name & CONNECT Statement
Introduced in v2.2	Intoduced in v2.3 (TCP/IP & SNA support)

S N A : Systems Networking Architecture. Set of protocals developed by IBM.

These protocals+TCP/ip are used to transferdata among network

5. Communication database

1. SYSIBM.LOCATIONS
2. SYSIBM.IPNAMES
3. SYSIBM.LUNAMES
4. SYSIBM.USERNAMES
5. SYSIBM.IPLIST
6. SYSIBM.LULIST



SYSIBM.LOCATIONS

LOCATION	LINKNAME	IBMREQD	PORT	TPN	DBALIAS

- provide location name for the node in the network
- provide link name
- FTP (port number 21), Telnet (port number 23), and DRDA relational database (port number 446).

SYSIBM.USERNAMES

The USERNAMES table is used for outbound ID translation (TCP/IP and SNA) and inbound ID translation and 'come from' checking

SYSIBM.LUNAMES

The IPLIST table allows multiple IP addresses to be specified for a given LOCATION.

6. There are 3 options for DDF in ZPARM (DDF startup option field)

- AUTO : Specifies that this facility is to be automatically initialized and started when the DB2 subsystem is started.
- NO: If you specify NO Signifies that DDF is not to be loaded at DB2 startup and that it cannot be started with a command.
- COMMAND : Specifies that the facility is to be initialized at DB2 startup and is prepared to receive the DSN1 START DDF command.

To Make Sucessfull Distributed Data we requires. . . . . ?

- Management by Right People
- Security (Hardware, Workstation, Network, Data, Auditing)
- Backup & Recovery
- Data synchornization

Auditing:ensure that access of distibuted data can be tracked for security purpose

Some Specific DDF ZPARMs

1. MAXDBAT → max remote database access threads
2. CONDBAT → Max remote connections
3. DDF → startup option
4. CMTSTAT → DDF threads
5. IDTHTOIN → Idle thread timeout
6. TCPKPALV → TCP/IP keepalive
7. POOLINAC → pool thread timeout

15.DB2 – SQL

14.1 SQL Processing

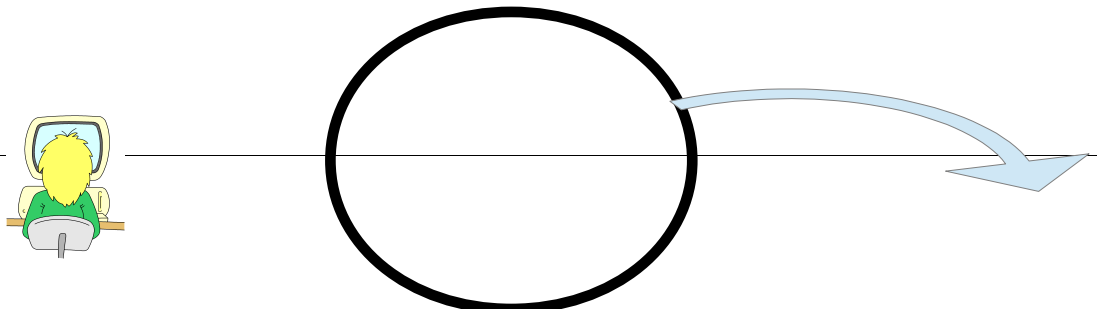
- SQL commands are instructions given to relational database to perform specific task. SQL(Structured Query Language) is a language understood by Relational database

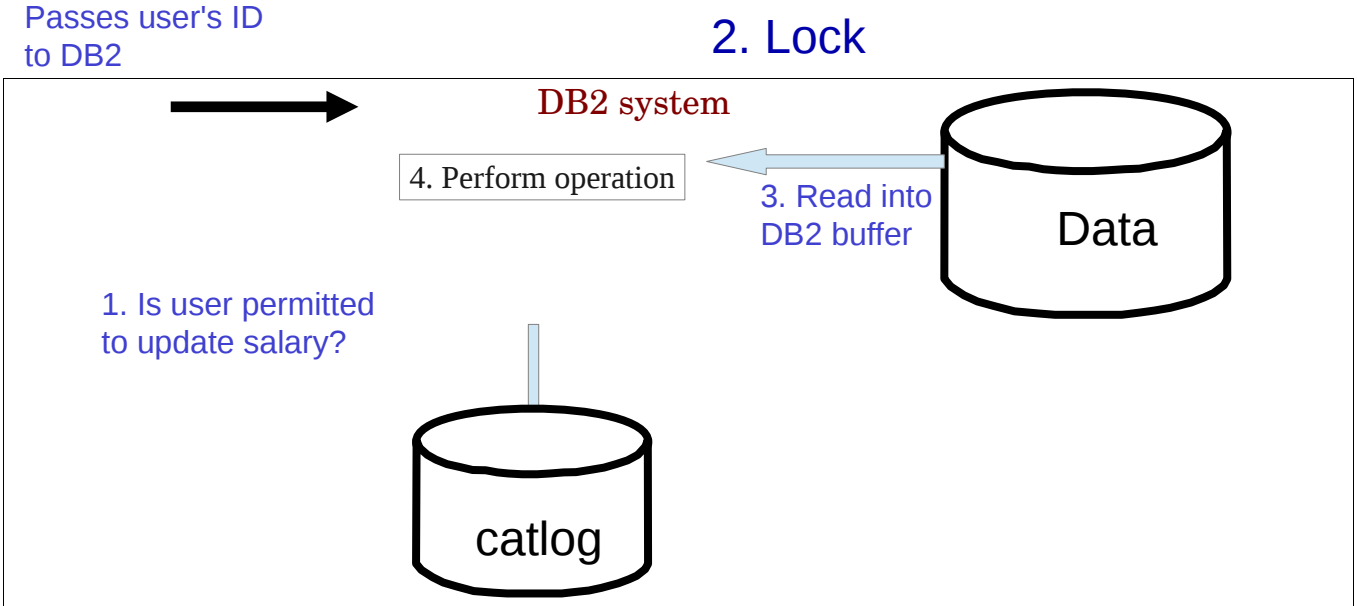
- Sql commands are grouped into 4 major categories

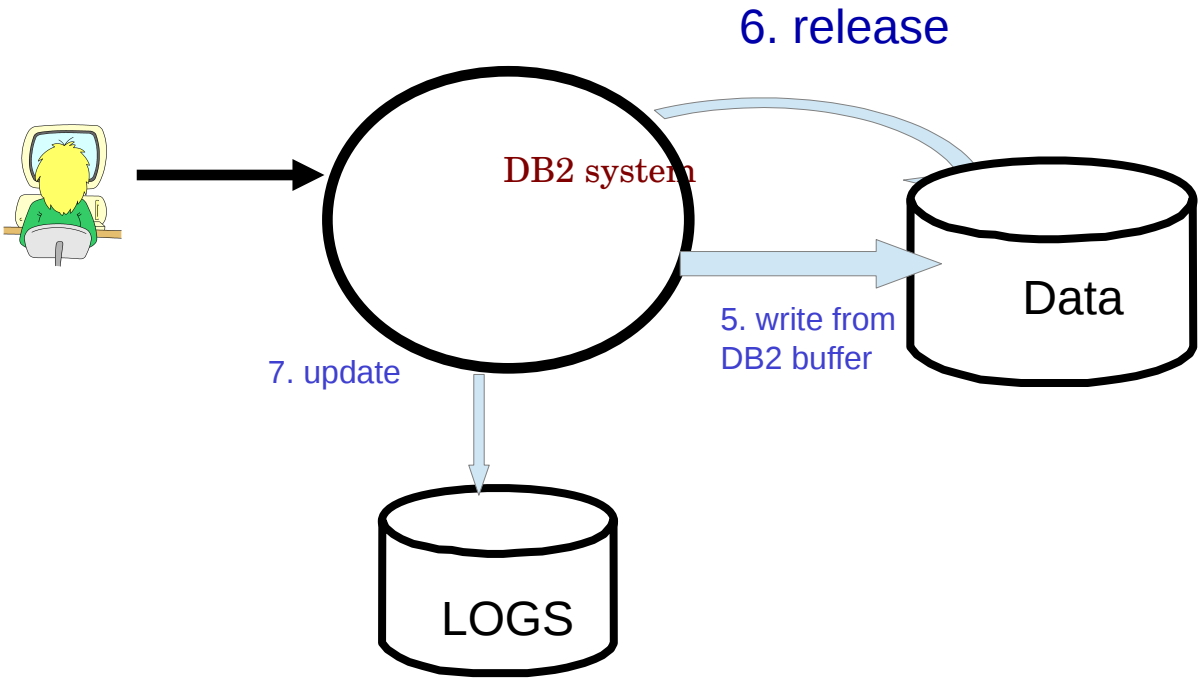
- Data defination language (DDL)
- Data manipulation language (DML)
- Data control language (DCL)
- Transaction control language (TCL)

DB2 handles an SQL *Change* request

consider example user want to change salary in database







14.2 Data types

Every column in every DB2 table has a data type. The data type influences the range of values that the column can have and the set of operators and functions that apply to it. Datatypes are two categories as shown in below.

1. Build in/IBM supplied datatypes

- Timestamp datatypes
- String datatypes
- Numeric datatypes
- ROWID datatype
- XML datatype

2. user defined/ distint datatypes

What are the three characteristics of column?

- Column name
- Data type (may or maynot be with length)
- NULL characteristic (NOT NULL, NOT NULL WITH DEFAULT, nullable)

"nullable" Column can be marked as having an 'unknown value'

Timestamp datatypes

Datatype	Description
DATE (4 bytes)	A date is a three-part value representing a year, month, and day in the range of 0001-01-01 to 9999-12-31.
TIME (3 bytes)	A time is a three-part value representing a time of day in hours, minutes, and seconds, in the range of 00.00.00 to 24.00.00.
TIMESTAMP (10 bytes)	A timestamp is a seven-part value representing a date and time by year, month, day, hour, minute, second, and microsecond, in the range of 0001-01-01-00.00.00.000000000 to 9999-12-31-24.00.00.000000000 with nanosecond precision.

DB2 can convert it to or from any of the formats in the following table.

Format	*Time*	*Date*
International Standards	2003-12-25	13.30.05

Organization		
IBM USA standard	12/25/2003	1:30 PM
IBM European standard	25.12.2003	13.30.05
Japanese Industrial Stadard	2003-12-25	13:30:05

String datatypes

Datatype	*Decription*
CHARACTER(n)	Fixed-length character strings with a length of n bytes. n must be greater than 0 and not greater than 255. The default length is 1.
VARCHAR(n)	Varying-length character strings with a maximum length of n bytes. n must be greater than 0 and less than a number that depends onthe page size of the table space. The maximum length is 32704.
CLOB(n)	(Character Large Objects)Varying-length character strings with a maximum of n characters. n cannot exceed 2 147 483 647. The default length is 1M.
GRAPHIC(n)	Fixed-length graphic strings that contain n double-byte characters. n must be greater than 0 and less than 128. The default length is 1. this is used to represent letters of chinease, japanease, korea...
VARGRAPHIC(n)	Varying-length graphic strings. The maximum length, n, must be greater than 0 and less than a number that depends on the page size of the table space. The maximum length is 16352.
DBCLOB(n)	(Double Byte CLOB) Varying-length strings of double-byte characters with a maximum of n double-byte characters. n cannot exceed 1 073 741 824. default is

	1M
BINARY(n)	Fixed-length or varying-length binary strings with a length of n bytes. n must be greater than 0 and not greater than 255. default length is 1.
VARBINARY (n)	Varying-length binary strings with a length of n bytes. The length of n must be greater than 0 and less than a number that depends on the page size of the table space. The maximum length is 32704.
BLOB(n)	(Binay LOB)Varying-length binary strings with a length of n bytes such as pictures & voice. n cannot exceed 2 147 483 647. The default length is 1M

ROWID Datatype

ROWID data type to uniquely and permanently identify rows in a DB2 subsystem.

XML Datatype

The XML data type is used to define columns of a table that store XML values.

Numeric datatypes (default value=0)

Datatype	*Decription*
SMALLINT	A small integer is binary integer with a precision of 15 bits. The range is -32768 to +32767.
INTEGER or INT	Large integers. A large integer is binary integer with a precision of 31 bits. The range is -2147483648 to +2147483647.
BIGINT	A big integer is a binary integer with a precision of 63 bits. The range of big integers is -9223372036854775808 to +9223372036854775807.

DECIMAL(p,s) or NUMERIC	decimal number, Precision (p) determines how many digits (up to a maximum of 31) to be reserved for storing/expressing the number. Scale (s) must be a non-negative (i.e. either 0 or positive), and indicates how many places to reserve to the right of the decimal point when displaying the result.
DECFLOAT	decimal floating-point number
REAL	single-precision floating-point number of 32 bit
DOUBLE	double-precision floating-point number of 64 bit

Datatype	default
Numeric datatypes	0
CHAR, GRAPHIC	Blanks
Binary	Hexadecimal zeros
Varible Length Strings (VARCHAR, CLOB, VARGRAPHIC, DBCLOB, VARBINARY, or BLOB)	Empty
Time stamp datatypes	Current data& time
ROWID	DB2 generated

Data Definition Language

set of SQL commands which is used to create, alter or drop database objects is called DDL

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database

14.3 DML

The SQL **WHERE** clause is used to specify a condition while doing operation on data from single table or joining with multiple tables. Here operation refers SELECT, UPDATE, DELETE...consider below example

ID	NAME	AGE	ADDRESS	SALARY
1	Deepu	32	Ahmedabad	2000.00
2	Phani	25	Delhi	1500.00
3	Venky	23	Kota	2000.00
4	harsha	25	Mumbai	6500.00
5	Sudheer	27	Bhopal	8500.00
6	Harinath	22	MP	9500.00
7	Summanth	24	Indore	10000.00

Syntax: SELECT column1, column2, columnN
FROM table_name
WHERE [condition]

Example: SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 8000;

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  5 | Sudheer  |  8500.00 |
|  6 | Harinath |  9500.00 |
|  7 | Summanth | 10000.00 |
+----+----------+----------+
```

SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE NAME = 'Sudheer';

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  5 | Sudheer  |  8500.00 |
```

+-------+----------+----------+

CREATE/ ALTER/ DROP Bufferpool

indicates size for page you need to declare for bufferpools (4K,8K,16K,32K)

db2 create bufferpool <bp_name> pagesize <size>

Drop bufferpool

drop bufferpool <bp_name>

Syntax: db2 create table <schema_name>.<table_name> (col_name col_type,...) in <tablespace_name>

EX: db2 create table professional.employee(id int, name varchar(50), joindate date,salary double) in ts1

Creation of user defined datatype

```
CREATE DISTINCT TYPE US_DOLLAR AS DECIMAL (9,2);
CREATE DISTINCT TYPE CANADIAN_DOLLAR AS DECIMAL (9,2);
CREATE TABLE US_SALES(PRODUCT_ITEM_NO     INTEGER,
                      MONTH               INTEGER,
                      YEAR                INTEGER
                 TOT_AMOUNT        US_DOLLAR);
```

CREATE/ ALTER/ DROP INDEX

db2 create unique index <index_name> on <table_name>(<unique_column>) include (<column_names..>)

CREATE/ ALTER/ DROP VIEW

```
db2 create view <view_name> (<col_name>, <col_name1...) as select <cols>..
     from <table_name>
```

Data Manipulation Language

set of SQL commands which is used to manipulate content in database objects is called DML

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records
MERGE	

INSERT clause

INTO clause

VALUES clause

DELETE clause

FROM clause

WHERE clause

UPDATE clause

SET clause

WHERE clause

UPDATE CUSTOMERS

SET ADDRESS = 'Pune', SALARY = 1000.00;

```
+------+--------------+------+----------------+-------------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Deepu    |  32 | Pune        |  1000.00 |
|  2 | Phani    |  25 | Pune      |  1000.00 |
|  3 | Venky    |  23 | Pune      |  1000.00 |
|  4 | harsha   |  25 | Pune      |  1000.00 |
|  5 | Sudheer  |  27 | Pune      |  1000.00 |
|  6 | Harinath |  22 | Pune      |  1000.00 |
|  7 | Summanth |  24 | Pune      |  1000.00 |
+------+--------------+-------+---------------+-------------+
```

UPDATE CUSTOMERS

SET ADDRESS = 'Pune'
WHERE ID = 6;

```
+------+--------------+------+----------------+-------------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Deepu    |  32 | Ahmedabad |  2000.00 |
|  2 | Phani    |  25 | Delhi     |  1500.00 |
|  3 | Venky    |  23 | Kota      |  2000.00 |
|  4 | harsha   |  25 | Mumbai    |  6500.00 |
|  5 | Sudheer  |  27 | Bhopal    |  8500.00 |
|  6 | Harinath |  22 | Pune     |  9500.00 |
```

```
|   7 | Summanth |   24 | Indore     | 10000.00 |
          +------+-------------+--------+---------------+-------------+
```

SELECT NAME, JOB, SALARY + BONUS AS INCOME

FROM EMPLOYEE

WHERE INCOME > 100000 --> syntax error described in below note

ORDER BY INCOME DESC;

Note : Every SELECT statement must have a SELECT and a FROM clause. The remaining clauses are option.

syntactical order

select --> from --> where --> GROUP BY --> orderd by

execution order

from --> where --> GROUP BY --> HAVING --> select --> orderd by

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
-
```
SELECT dept, SUM (salary)
FROM employee
GROUP BY dept
HAVING SUM (salary) > 25000
```

Note : all three of the queries below would execute successfully:

SELECT EMPNO, LASTNAME, BIRTHDATE

FROM EMPLOYEE

WHERE BIRTHDATE = ‘1950-12-18’ ;

SELECT EMPNO, LASTNAME, BIRTHDATE

FROM EMPLOYEE

WHERE BIRTHDATE = ’12/18/1950’ ;

SELECT EMPNO, LASTNAME, BIRTHDATE

FROM EMPLOYEE

WHERE BIRTHDATE = ’18.12.1950’ ;

Data Control Language

set of SQL commands which grant or revoke control on database objects is called DML

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

GRANT privileges ON object TO user;

```
REVOKE privileges ON object FROM user;
```

GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO smithj;

Transaction Control Language

set of SQL commands which are used to manage changes affecting the data is called data manipulation language

Command	Description
COMMIT	Gives a privilege to user
ROLLBACK	Takes back privileges granted from user
SAVEPOINT	

Constraints are the rules enforced on data columns on table.

NOT NULL

It is a rule to prohibit null values from one or more columns within the table.

db2 create table <table_name>(col_name col_type not null,..)

Unique constraints

Using these constraints, you can set values of columns uniquely. Unique Constraint column can only contain one NULL value

db2 create table <tab_name>(<col> <col_type> not null unique, ...)

NOT NULL WITH DEFAULT

- for suppose if you sort or compare null with other we will get UNKNOWN result.
- If a column is defined as NOT NULL, DB2 does not supply a default value. If a column is defined as NOT NULL WITH DEFAULT, DB2 supply a default value.
- if you do not specify NOT NULL, DB2 stores a default value for a column

DEFAULT

In DB2, we can specify a default value for any column.

COUNTRY CHAR(30) DEFAULT ' UNKNOWN',

MILES_TO_LA SMALLINT DEFAULT -1

- Default values can be a constant of the column's data type, the value of a DB2 special register(USER and CURRENT SQLID), or null.

- These two special registers contain the default (USER) and modifiable (CURRENT SQLID) value of the user's current DB2 authorization ID.

14.4 Operators

S.nO	Category	Operators
1	Arithmetic	+, -, *, /
2	Relational	<, >, =, >=, <=
3	Boolean	AND, OR

Example:1

```
SELECT ID, SALARY, COMM,
        SALARY+COMM AS INCOME        -- + operator in derived
```

attribute

```
FROM EMPLOYEE
WHERE (COMM/SALARY)*100 < 2000  --   div, mul, < operators
```

if we don't mention name with 'AS' clause the derived column will display with out column name.

ORDER of Precedence

- Process form left to right.
- inner post parentheses evaluate first, from that upto outermost parentheses.
- AND process before OR.
- multiplication & division before addition & subtraction.

Consider following example

```
SELECT EMPNO, JOB, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST';
AND EDLEVEL=16
OR EDLEVEL=18;
```

here AND will execute before OR which gives inconsistant results, to avoid it

```
SELECT EMPNO, JOB, EDLEVEL
FROM EMPLOYEE
WHERE JOB='ANALYST';
AND (EDLEVEL=16 OR EDLEVEL=18);
```

to avoid multiple OR clauses we can replace with one IN clause which will reduce complexity in terms of coding and execution

```
SELECT EMPNO, JOB, EDLEVEL
FROM EMPLOYEE
WHERE EDLEVEL IN (14,19,20);
```

+------+----------+------+----------+----------+

```
| ID | NAME      | AGE | ADDRESS    | SALARY    |
+----+-----------+-----+------------+-----------+
|  1 | Deepu     |  32 | Ahmedabad |  2000.00 |
|  2 | Phani     |  25 | Delhi      |  1500.00 |
|  3 | Venky     |  23 | Kota       |  2000.00 |
|  4 | harsha    |  25 | Mumbai     |  6500.00 |
|  5 | Sudheer   |  27 | Bhopal     |  8500.00 |
|  6 | Harinath |  22 | MP         |  9500.00 |
|  7 | Summanth |  24 | Indore     | 10000.00 |
          +------+----------------+--------+-----------------+-----------------+
```

SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE SALARY > 2000 AND age < 25;

```
+------+------------+-------------+

| ID | NAME     | SALARY    |
+----+----------+-----------+
|  6 | Harinath|  9500.00 |
|  7 | Sumanth | 10000.00 |
+----+----------+-----------+
```

SELECT ID, NAME, SALARY

FROM CUSTOMERS
WHERE SALARY > 8000 OR age < 25;

```
+------+-------------+--------------+

| ID | NAME      | SALARY    |
+----+-----------+-----------+
|  5 | Sudheer   |  8500.00 |
|  6 | Harinath |  9500.00 |
|  7 | Sumanth   | 10000.00 |
          +------+----------------+---------------+
```

14.5 Functions

Catagories

- Aggregate functions

- Scalar functions
  - Datetime scalar functions
  - Numeric scalar functions
  - String scalar functions
  - Cast scalar functions
- Table functions
- Row functions
- User defined functions

6.1 Aggregate/column functions

Pre defined Functions apply on qulified or entire rows at paricilar column

EX: SUM, AVG, MAX, MIN, COUNT

```
-- CREATE TABLE AHAN(ID CHAR(5),NAME CHAR(10), DT DATE,DE DECIMAL(9,5));
-- INSERT INTO AHAN VALUES('10000','IBM','2015-6-9',112.3);
-- INSERT INTO AHAN VALUES('10001','MICROSOFT','2015-6-10',108.16);
-- INSERT INTO AHAN VALUES('10002','HCL','2015-6-11',80.62);
-- INSERT INTO AHAN (ID,NAME) VALUES ('10003','TECHM');
   SELECT * FROM AHAN;
---------+---------+---------+---------+---------+---------+---------+---------+
ID     NAME        DT                  DE
---------+---------+---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-09    112.30000
10001  MICROSOFT   2015-06-10    108.16000
10002  HCL         2015-06-11     80.62000
10003  TECHM       ----------  -----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

```
  --  SELECT ID,NAME, COALESCE(DT,'0000-00-00')
  -- FROM SUPPLY ;
    SELECT SUM(DE) AS SUM ,
           AVG(DE) AS AVG ,
           MAX(DE) AS MAX ,
           COUNT(*) AS COUNT,
           COUNT(DE) AS COUN
      FROM AHAN;
 ---------+---------+---------+---------+---------+---------+---------+---------+
               SUM                AVG          MAX        COUNT         COUN
 ---------+---------+---------+---------+---------+---------+---------+---------+
         301.08000    100.36000000000    112.30000            4            3
 DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION
```

Ex: SELECT MAX(RATE) – MIN(RATE) --> column functions

INTO :RATE_SPREAD --> host variable

FROM CONTRACTOR

WHERE JOB = 'BrooklynBridge'

Some other aggregate functions

- VARAIANCE for variance
- STDDEV(which is squreroot of variance) for standerd deviation
- COVARIANCE for covariance

6.2 Scalar functions

Scalar functions produce one value for each row within the set of qualified rows. With the most current versions of DB2, there are close to 200 scalar functions.

Numeric Scalar Functions

1. The ROUND(attribute,round value) function is used to round a numeric field to the number of decimals specified.

```
SELECT ROUND(873.726,2), --   873.730
               ROUND(873.726,1),   --   873.700
             ROUND(873.726,0),     --   873.000
             ROUND(873.726,-1),    --   870.000
             ROUND(873.726,-2),    --   900.00
             ROUND(873.726,-3),    --   1000.00
             ROUND(873.726,-4)     --   0
         FROM SYSIBM.SYSDUMMY1;
```

2. DECIMAL(attribute, precision, scale) display attribute with this precision & scale

```
---------+---------+---------+---------+---------+---------+---------+---------+
     SELECT ID, DECIMAL(ROUND(DE*1.108,2),8,3) FROM AHAN;
---------+---------+---------+---------+---------+---------+---------+---------+
ID
---------+---------+---------+---------+---------+---------+---------+---------+
10000     124.430
10001     119.840
10002      89.330
10003  ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

3. ADS() or ABSVAL: it returns absolute value

   SELECT ABS(-24.6);

   Result: 24.6

4. CEIL or CEILING: Round to upper value

   CEIL(32.65)

   Result: 33

5. FLOOR: Round to lower value

   SELECT FLOOR(5.9);

   Result: 5

6. LN: Gives natural algorithm.

7. LOG10: Gives log base 10 value

8. MOD(m,n): returns reminder

9. TRUNCATE(m,n): after period it returns only n digits

10. SIN: Sine of value

11. COS: Cosine of value

12. TAN: it returns tan value

13. POWER(p,q): p power of q

14. SQRT: gives square root of number.

String Scalar Functions

1. SUBSTRING (expression ,start , length)

displays sub-string from “start” position upto “length

```
---------+---------+---------+---------+---------+---------+
     SELECT ID, SUBSTR(NAME,2,4) FROM AHAN;
---------+---------+---------+---------+---------+---------+
ID
---------+---------+---------+---------+---------+---------+
10000  BM
10001  ICRO
10002  CL
10003  ECHM
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

2. UPPER(),converts the value of a field to uppercase.

3. LOWER(),converts the value of a field to lowercase.

These uppercase & lowercase convertion happent at query execution only, neither at table nor at display

```
---------+---------+---------+---------+---------+---------+
     SELECT ID, LOWER(SUBSTR(NAME,2,4)) FROM AHAN;
---------+---------+---------+---------+---------+---------+
ID
---------+---------+---------+---------+---------+---------+
10000  bm
10001  icro
10002  cl
10003  echm
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

5. ASCII() - returns ascii value of left most character of expresion

6. CHARACTER_LENGTH: returns the expression length.

7. CHR(), retuns character for coressponding ascii.

8. LEFT(exp,len): returns left part of expression

9. RIGHT(exp,len): returns right part of expression

10. CHAR(x,y), wher x is cloumn defined with date or time datatype. Where y is timestamp standard

11. CONCAT()

The DB2 CONCAT function will combine two separate expressions to form a single string expression.

```
SELECT CONCAT(MFNAME, MFPARTNUMBER)
FROM CATENTRY;
```

below example with multiple concat functions used to include spaces or comas or other substrings explicitly

```
SELECT CONCAT(CONCAT(MFNAME, ' '), MFPARTNUMBER)
FROM CATENTRY;
```

below syntax simpilifies query by using symbol '|| ||' instead of using concat function

```
SELECT MFNAME || ' ' || MFPARTNUMBER
FROM CATENTRY;
```

```
---------+---------+---------+---------+---------+---------+
     SELECT CHAR(DT,USA)||','||CHAR(DT,ISO)||','||
            CHAR(DT,EUR)||','||CHAR(DT,JIS) AS DATE_FORMATS
            FROM AHAN;
---------+---------+---------+---------+---------+---------+
DATE_FORMATS
---------+---------+---------+---------+---------+---------+
06/09/2015,2015-06-09,09.06.2015,2015-06-09
06/10/2015,2015-06-10,10.06.2015,2015-06-10
06/11/2015,2015-06-11,11.06.2015,2015-06-11
-----------------------------------------
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

```
---------+---------+---------+---------+---------+---------+---------+---------+
     SELECT CHAR(CURRENT TIME,USA)||', '||CHAR(CURRENT TIME,ISO)||', '||
            CHAR(CURRENT TIME,EUR)||', '||CHAR(CURRENT TIME,JIS)
            AS TIME_FORMTS FROM AHAN WHERE ID='10000';
---------+---------+---------+---------+---------+---------+---------+---------+
TIME_FORMTS
---------+---------+---------+---------+---------+---------+---------+---------+
11:03 AM, 11.03.39, 11.03.39, 11:03:39
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

DB2 special registors

These registors have current time and date status provided by OS

CURREENT TIME		CURREENT__TIME
CURRENT DATE	OR	CURRENT__DATE
CURRENT TIMESTAP		CURRENT__TIMESTAP

we can use space or _ in register names

TimeDate Scalar Functions

1. YEAR: Extract year portion from Date or Timestap attribute or days
2. MONTH: Extract month portion from Date or Timestap attribute or days
3. DAY: Extract day portion from Date or Timestap attribute or days
4. HOUR: Extract Hour portion from Time or Timestap attribute
5. MINUTE: Extract Minute portion from Time or Timestap attribute
6. SECOND: Extract Second portion from Time or Timestap attribute

7. MICROSECOND:Extract Microsecond portion from Time or Timestap attribute

8. Date: Extract Date portion from Timestap attribute or days

9. Time: Extract Time portion from or Timestap attribute

10. DAYOFWEEK : returns the day of the week in the range 1-7, 1 refers Sunday.

11. DAYOFYEAR: day of year from date or timestamp or days

12. DAYOFWEEK_ISO :similar to DAYOFWEEK, where 1 represents monday.

13. WEEK: range of return value 1-54. where week starts with Sunday.

14. WEEK_ISO: returns the week of the year in the range 1-53.where week starts with monday.

15. MONTHNAME: returns month name from year or timestamp or days

16. DAYS: convert date into days since 1st jan AD.

```
SELECT ID, SALARY, COMM,
FROM EMPLOYEE
WHERE DAYS(start_Date) – DAYS(CURRENT DATE) <= 2000
```

gives number of days between start_date and current data.

We can implement these kind of quires without using DAYS function also

```
SELECT ID, SALARY, COMM,
FROM EMPLOYEE
        WHERE  start_Date) – CURRENT DATE <= 2000
```

16. COALESCE

This COALESCE function is used to substitute null values in display or at query execution

COALESCE(EXP_1,EXP_2,....,'<Constant >')

we can replace nulls of many expresions or attibutes at a time with above syntax.

```
SELECT ID, COALESCE(DT,'2010-01-01') FROM AHAN;
```

the constant value place must be compatable with coresponding attribute or expression. In above example DT is date datatype we can give 0001-01-01 to 9999-12-31 as constant values, but we can't give 0000-00-00 it will not acceptable by coressponding datatype

Why we required COALESCE

consider following examples

```
SELECT ID, SUM(SALARY) as  SUM,
FROM EMPLOYEE
```

in above example in column 'SALARY' row wise summation will happend if there is any null values it will neglet

```
SELECT ID, SALARY, COMM,  SALARY+COMM AS INCOME
FROM EMPLOYEE
```

in above example column wise summation will happend if there is any null values it null as overall result. To solve this problem we will use following query

```
SELECT ID, SALARY, COMM,  SALARY+ COALESCE(COMM,0) AS INCOME
FROM EMPLOYEE
```

to visualise this concept see below table

S.no	SALARY	COMM	SALARY+COMM
1	1k	2k	3k
2	1k	1k	2k
3	----------	2k	--------
4	2k	1k	3k
5	2k	--------	---------
SUM	6k	6k	

Cast Scalar Functions

- SMALLINT
- INIT
- BIGINT
- DECIMAL
- FLOAT
- VARCHAR
- GRAPHIC

14.6 Joins

Before going to joing concepts we have know below concepts

Given: table EMPL, with columns EMPNO, NAME, DEPTNO .
table DEPT, with columns DEPTNO, DEPTNAME

```
SELECT DEPTNO, DEPTNAME
FROM EMPL, DEPT
WHERE DEPTNO = DEPTNO ; --> SQL Error! Ambiguous column name DEPTNO
```

```
SELECT EMPL.DEPTNO, DEPT.DEPTNAME
FROM EMPL, DEPT
WHERE EMPL.DEPTNO = DEPT.DEPTNO ; --> OK  column names are qualified by  table names
```

```
SELECT E.DEPTNO, D.DEPTNAME
FROM EMPL E, DEPT D
WHERE E.DEPTNO = D.DEPTNO ; --> OK , column names are qualified by the correlation names
```

JOINS

1. INNER JOIN
2. LEFT JOIN or LEFT OUTER JOIN
3. RIGHT JOIN OR RIGHT OUTER JOIN
4. FULL JOIN
5. LEFT EXCLUDING JOIN
6. RIGHT EXCLUDING JOIN
7. OUTER EXCLUDING JOIN
8. SELF JOIN

9. CROSS JOIN

10. NATURAL JOIN

JOINS:

The SQL **Joins** clause is used to retrive records from two or more tables in a database. There are different types of joins to explain them consider following 2 tables.

SUPPLY Table

```
--  SELECT * FROM SUPPLY NATURAL JOIN ORDER ON SUPPLY.ID=ORDER.ID;
--  CREATE TABLE SUPPLY(ID CHAR(5),NAME CHAR(10));
--  INSERT INTO SUPPLY VALUES('10000','IBM');
--  INSERT INTO SUPPLY VALUES('10001','MICROSOFT');
--  INSERT INTO SUPPLY VALUES('10002','HCL');
--  INSERT INTO SUPPLY VALUES('10003','TECHM');
  SELECT * FROM SUPPLY;
---------+---------+---------+---------+---------+---------+---------+
ID     NAME
---------+---------+---------+---------+---------+---------+---------+
10000  IBM
10001  MICROSOFT
10002  HCL
10003  TECHM
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

ORDER Table

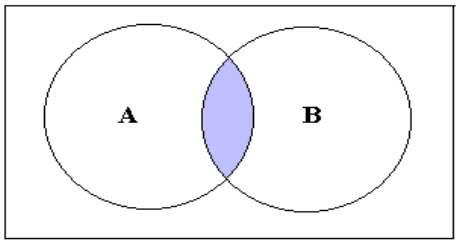
```
--CREATE TABLE ORDER(ORD_ID INT,SUPP_ID CHAR(5),ORD_DATE DATE);
--INSERT INTO ORDER VALUES(50125,'10000','2015-06-12');
--INSERT INTO ORDER VALUES(50126,'10001','2015-06-13');
--INSERT INTO ORDER VALUES(50127,'10004','2015-06-14');
 SELECT * FROM ORDER;
---------+---------+---------+---------+---------+---------+---------+
     ORD_ID  SUPP_ID  ORD_DATE
---------+---------+---------+---------+---------+---------+---------+
      50125  10000    2015-06-12
      50126  10001    2015-06-13
      50127  10004    2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

INNER Join

Below describes shows INNER join

```
  SELECT SUPPLY.ID,NAME, ORD_DATE
  FROM SUPPLY
  INNER JOIN ORDER
  ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

Inner join results common part of two two tables as visualised in below diagram



Here A means Supply table and B means Order table. There is a alternative tradational syntax for INNER join with out using Clause INNER

SELECT SUPPLY.ID, NAME, ORD_DATE

FROM SUPPLY, ORDER

WHERE SUPPLY.ID=ORDER.ID

or

SELECT SUPPLY.ID, NAME, ORD_DATE

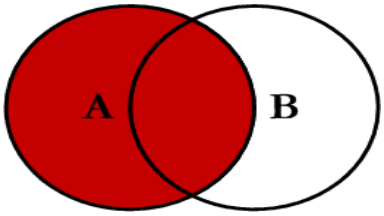
FROM SUPPLY

JOIN ORDER

ON SUPPLY.ID=ORDER.ID

LEFT Join

we will get records A and B Intersection and all records of A



A : Supply table, O : Order table

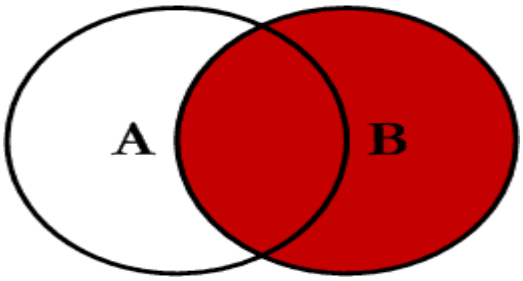
```
  SELECT SUPPLY.ID,NAME, ORD_DATE
  FROM SUPPLY
  LEFT JOIN ORDER
  ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
10002  HCL         ----------
10003  TECHM       ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

Alternative syntax

```
  SELECT SUPPLY.ID,NAME, ORD_DATE
  FROM SUPPLY
  LEFT OUTER JOIN ORDER
  ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
10002  HCL         ----------
10003  TECHM       ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

RIGHT Join

we will get records A and B Intersection and all records of B

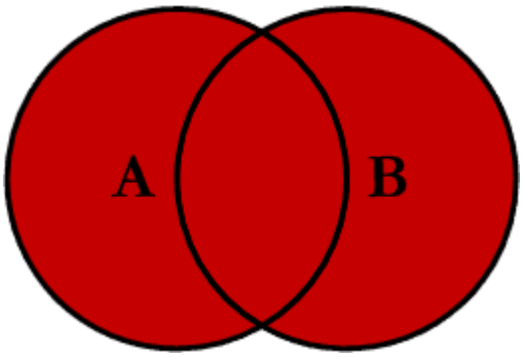


A : Supply table, O : Order table

```
---------+---------+---------+---------+---------+---------+
  SELECT SUPPLY.ID,NAME, ORD_DATE
  FROM SUPPLY
  RIGHT JOIN ORDER
  ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
-----  ----------  2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

FULL Join

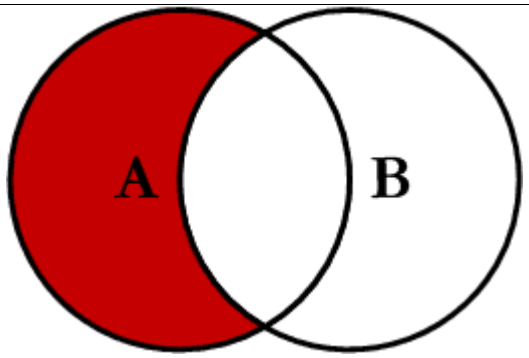
we will get all the records of A and B.




```
---------+---------+---------+---------+---------+---------+
  SELECT SUPPLY.ID,NAME, ORD_DATE
  FROM SUPPLY
  FULL JOIN ORDER
  ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+
ID      NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+
10000   IBM         2015-06-12
10001   MICROSOFT   2015-06-13
10002   HCL         ----------
10003   TECHM       ----------
-----   ----------  2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

LEFT EXCLUDING JOIN

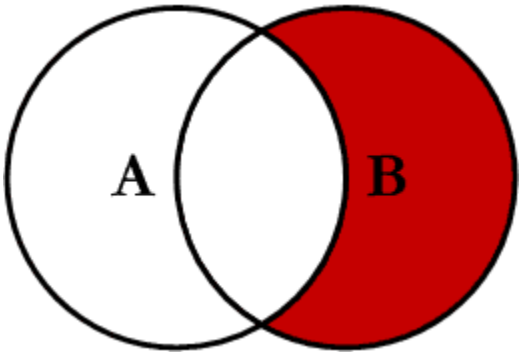
rows only belongs to TABLE A and not a part of B




```
  SELECT * FROM SUPPLY S LEFT JOIN ORDER O
  ON S.ID=O.ID
  WHERE O.ID  IS NULL ;
---------+---------+---------+---------+---------+---------+
ID     NAME            ORD_ID  ID     ORD_DATE
---------+---------+---------+---------+---------+---------+
10002  HCL         -----------  -----  ----------
10003  TECHM       -----------  -----  ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

RIGHT EXCLUDING JOIN

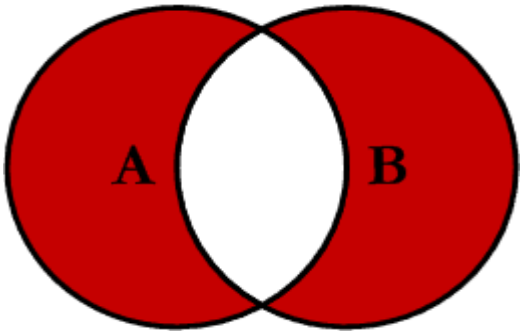
rows only belongs to TABLE B and not a part of A



```
  SELECT * FROM SUPPLY S RIGHT JOIN ORDER O
  ON S.ID=O.ID
  WHERE S.ID  IS NULL ;
---------+---------+---------+---------+---------+---------+
ID     NAME             ORD_ID  ID     ORD_DATE
---------+---------+---------+---------+---------+---------+
-----  ----------        50127  10004  2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

OUTER EXCLUDING join

all the records except intersection



SELECT <*select_list*>
FROM Table_A A

FULL OUTER JOIN Table_B B

ON A.Key = B.Key

WHERE A.Key IS NULL OR B.Key IS NULL

SELF Join

The SQL **SELF JOIN** is used to join a table to itself .

SELECT a.column_name, b.column_name...

FROM table1 a, table1 b

WHERE a.common_field = b.common_field;

```
  SELECT * FROM SUPPLY A INNER  JOIN SUPPLY B
  ON A.ID=B.ID;
---------+---------+---------+---------+---------+---------
ID      NAME        ID      NAME
---------+---------+---------+---------+---------+---------
10000   IBM         10000   IBM
10001   MICROSOFT   10001   MICROSOFT
10002   HCL         10002   HCL
10003   TECHM       10003   TECHM
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------
```

CROSS join

produces the Cartesian product of the tables. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table.

```
  SELECT * FROM SUPPLY CROSS JOIN ORDER;
---------+---------+---------+---------+---------+---------+
ID      NAME              ORD_ID  ID      ORD_DATE
---------+---------+---------+---------+---------+---------+
10000   IBM                50125  10000   2015-06-12
10001   MICROSOFT          50125  10000   2015-06-12
10002   HCL                50125  10000   2015-06-12
10003   TECHM              50125  10000   2015-06-12
10000   IBM                50126  10001   2015-06-13
10001   MICROSOFT          50126  10001   2015-06-13
10002   HCL                50126  10001   2015-06-13
10003   TECHM              50126  10001   2015-06-13
10000   IBM                50127  10004   2015-06-14
10001   MICROSOFT          50127  10004   2015-06-14
10002   HCL                50127  10004   2015-06-14
10003   TECHM              50127  10004   2015-06-14
```

14.7 Unions

UNION-clause

The SQL **UNION** clause/operator is used to combine the results of two or

more SELECT statements without returning any duplicate rows.

It gives rows iff either of condition obeys. In below example we make union between right join and left join which gives full join

```
    SELECT SUPPLY.ID, NAME, ORD_DATE FROM SUPPLY
    LEFT JOIN ORDER
    ON SUPPLY.ID=ORDER.ID
    UNION
    SELECT SUPPLY.ID, NAME, ORD_DATE FROM SUPPLY
    RIGHT JOIN ORDER
    ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
10002  HCL         ----------
10003  TECHM       ----------
-----  ----------  2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
```

UNION ALL -clause

The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows. Remaining as same as UNION

```
   SELECT SUPPLY.ID, NAME, ORD_DATE
   FROM SUPPLY
   LEFT JOIN ORDER
   ON SUPPLY.ID=ORDER.ID
   UNION ALL
   SELECT SUPPLY.ID, NAME, ORD_DATE
   FROM SUPPLY
   RIGHT JOIN ORDER
   ON SUPPLY.ID=ORDER.ID;
```

```
---------+---------+---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
10002  HCL         ----------
10003  TECHM       ----------
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
-----  ----------  2015-06-14
DSNE610I NUMBER OF ROWS DISPLAYED IS 7
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

INTERSECT -clause

returns records iff both conditions are obeyed. In below example intersection of left join and right join gives inner join

```
   SELECT SUPPLY.ID, NAME, ORD_DATE
   FROM SUPPLY
   LEFT JOIN ORDER
   ON SUPPLY.ID=ORDER.ID
   INTERSECT
   SELECT SUPPLY.ID, NAME, ORD_DATE
   FROM SUPPLY
   RIGHT JOIN ORDER
   ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+---------+---------+
10000  IBM         2015-06-12
10001  MICROSOFT   2015-06-13
```

EXCEPT -clause

returns records The SQL **EXCEPT** clause/operator is used to combine two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement. Below example results

Left excluding join.

```
    SELECT SUPPLY.ID, NAME, ORD_DATE FROM SUPPLY
    LEFT JOIN ORDER
    ON SUPPLY.ID=ORDER.ID
    EXCEPT
    SELECT SUPPLY.ID, NAME, ORD_DATE FROM SUPPLY
    RIGHT JOIN ORDER
    ON SUPPLY.ID=ORDER.ID;
---------+---------+---------+---------+---------+---------+---------+---------+
ID     NAME        ORD_DATE
---------+---------+---------+---------+---------+---------+---------+---------+
10002  HCL         ----------
10003  TECHM       ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Rules for Unions:

Many SELECT Statements can be included in this clauses. The set of statements included refered as Stack. All the queries in stack execute one by one.

1. All quires result same number of columns

2. Datatypes of coressponding datatypes must be compatable may not be same or May not be same length or precession, it will largest precession among them

3. Only one order by clause is allowed that to at last query of stack

4. the coloumns included by Order by must be in every query

14.8 . Fetch through substring

```
SELECT ID, SALARY, COMM,  SALARY+COMM AS INCOME
FROM EMPLOYEE
WHERE UPPER(LASTNAME) like '%SON%'
```

in above example dilevers all rows which as substring son at column 'LASTNAME'

there are many ways to fetch substring using symbols % and _ as shown in below.

- %SON% --> substring is 'SON'
- SON% --> start with 'SON'
- %SON --> end with 'SON'
- '_Son%' --> string must contain SON from 2nd position

And there is many posibilites will exist

observe below practical example

```
---------+---------+---------+---------+---------+---------+---------+
     SELECT ID, NAME FROM AHAN WHERE NAME LIKE '__L%';
---------+---------+---------+---------+---------+---------+---------+
ID     NAME
---------+---------+---------+---------+---------+---------+---------+
10002  HCL
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

15. DB2 sysprog fundamentals

15.1 IPL & POR

15.2 Thread management

1. A thread is a structure that describes a connection made by an application and traces its progress in the DB2 subsystem.

2. Types of threads

- Allied threads
- Distributed database access threads

3. Allied threads are threads that are connected to DB2 from TSO, batch, IMS™, CICS®, CAF, or RRSAF.

4. Distributed database access threads (sometimes called DBATs) are threads that are connected through a network to access data at a DB2 server on behalf distributed requesting systems.

- in DB2 CM mode, ENFM(Enable new function mode), NFM(New Function Mode), v11
-

- User-defined default values
- You can specify a particular default value, such as:
- DEFAULT 'N/A'
- pageset is table space + index space.table space or index space
- recover copy is differ to copy

END

Efforts Never Die

Thread Management

1. What is thread?

A thread is a structure that describes a connection made by an application and traces its progress in the DB2 subsystem.

2. Types of threads?

Allied threads :Allied threads are threads that are connected to DB2 from TSO, batch, IMS™, CICS®, CAF, or RRSAF.

Database Access Threads :Distributed database access threads (DBATs) are threads that are connected through a network to access DB2 server data

3. Modes of DBATS?

ACTIVE mode :When the value CMTSTAT subsystem parameter is ACTIVE, a database access thread is always active from initial creation to termination.

INACTIVE mode: a database access thread can be active or pooled. When a database access thread is active, it is processing requests from client connections within units of work. When a database access thread is pooled, it is waiting for the next request from a client to start a new unit of work. database access threads are terminated under any of the following conditions

- After processing 200 units of work.
- After being idle in the pool for the amount of time specified by the value of the IDTHTOIN subsystem parameter.

4. How number of threads are limited?

- Specify a value for the CTHREAD subsystem parameter to control the maximum number of concurrent allied threads that are started at the local DB2 subsystem.

- Specify a value for the MAXDBAT subsystem parameter to control the maximum number of concurrently active database access threads.

- The sum of the values of the CTHREAD and MAXDBAT subsystem parameters is limited to a maximum value of 20000.(DB2 11)

- Specify the value of the CONDBAT subsystem parameter to control the concurrent inbound DDF connections. The maximum acceptable value for CONDBAT subsystem parameter is 150000.

5. Timeout for IDLE active threads?

- IDTHTOIN,When a thread remains idle than the specified limit, DB2 might cancel the thread. However, pooled and in-doubt threads are not canceled.

- A value of 0, the default, means that the server threads cannot be canceled because of an idle thread timeout. You can specify a value from 0 to 9999 seconds.

6. What is trusted context

  - A trusted context is an independent database entity that you can defines connection characteristices and users who can use that connections.
    - Job name
    - IP address
    - Encryption
    - SERVAUTH
  - A trusted connection is a database connection that is established when the connection

attributes match those of a trusted context that is defined at the server. A trusted connection can be established locally or at a remote location.

7.