UNIT 2: DB2 INTERNALS : OVERVIEW

OVERVIEW

- **DATA STORAGE**
- **THE OPTIMIZER**
- **DB2 CATALOG & DIRECTORY**
- **DATA LOCKING**

Figure 2.1 DB2 Internals

DATA STORAGE

There are two types of DATA STORAGE.

- **Logical storage**
- **Physical storage**

Figure 2.2Data Storage

Notes:

Logical storage is how the user sees the data. Physical storage is how data is actually stored. The total collection of stored data is divided into a number of user databases and system databases. Each database is divided into number of table spaces and index spaces.

A space is a dynamically extendible collection of pages, where a page is a block of physical storage. The pages in a given space are all of the same size 4KB for index spaces and either 4KB or 32KB for table spaces. Each table space contains one or more tables. A stored table is a physical representation of a base table; it consists of a set of stored records, one for each row of the base table.

A given stored table must be wholly contained with in a single table space. Each index space contains exactly one index. A given index must be wholly contained in an index space. A given stored table and all its associated indexes must be wholly contained with in a single database.

DATA STORAGE (Cont….)

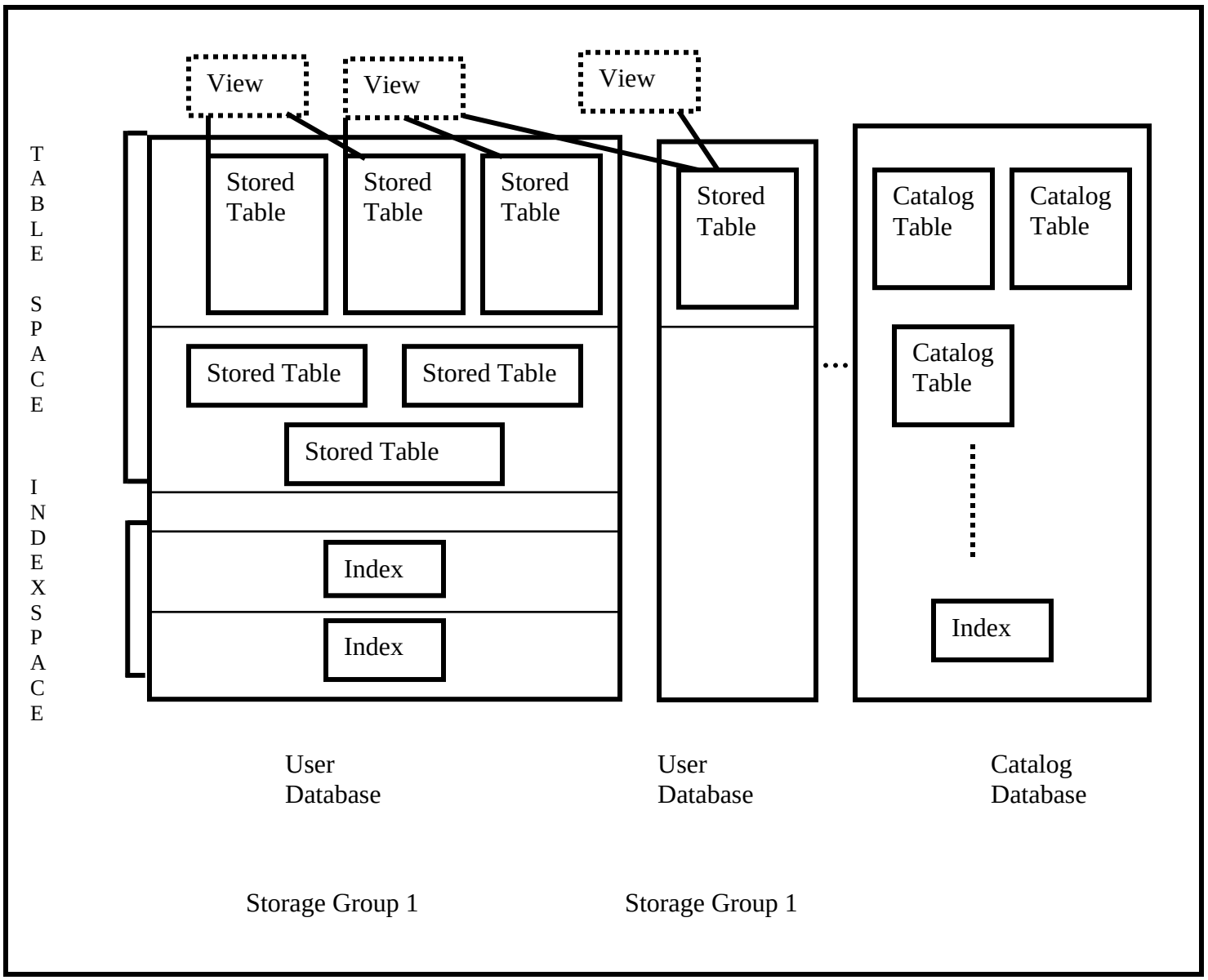

Figure 2.3Data Storage (Cont….)

DATA STORAGE (Cont….)

DATABASES:

A database is a collection of logically related objects. That is a collection of stored tables that are related together in some way, together with their associated indexes and the various spaces that contain those tables and indexes. It thus consists of a set of table spaces and index spaces.

TABLESPACES:

A table space can be considered a logical address space on secondary storage that is used to hold one or more stored tables. As the amount of data in those tables grows, storage will be acquired from the appropriate storage group and added to the table space. One table space can be up to approximately 64 billion bytes in size. The table space is the storage unit for recovery and reorganization purposes, i.e., it is the unit that can be recovered via RECOVER utility or reorganized using the REORG utility. If the table space is very large, then recovery and reorganization can take a very long time.
Table spaces can be of three types

1. **Simple**
2. **Partitioned**
3. **Segmented**

A simple table space can contain more than one stored table, though one is the best option. One reason for having more than one table is that related data can be kept together, making the access faster. Partitioned table spaces are intended for tables that are sufficiently large i.e. operationally difficult to deal with the entire table as a single unit.

A partitioned table space thus contains exactly one table, partitioned in accordance with the value ranges of a partitioning column or column combination.
In the segmented table spaces, the space with in the table space is divided in to segments, where each segment contains logically contiguous ‘n’ pages where ‘n’ must be a multiple of 4 and must range between 4 and 64 and should be the same in all segments. No segment will have records of more than one stored table and if the table grows in size to fill a segment a new segment will be obtained.

Figure 2.4Data Storage (Cont….)

DATA STORAGE (Cont….)

INDEXSPACES:

An index space is to an index like a table space is to a table. The correspondence between indexes and index spaces are always one-to-one, there are no data definition statements to index spaces. Thus there is no CREATE INDEX SPACE but only CREATE INDEX, which will automatically create the index space. Index spaces are always 4 KB in size.

INDEXES:

Indexes in DB2 are based on a structure known as B-Tree. B-Tree is a multilevel, tree structure index with a property that tree is always balanced, which means that all the leaf entries in the structure are equidistant from the root of the tree and this property is maintained as new entries are inserted into or existing entries are deleted from the tree.

STORAGE GROUPS:

Storage groups are named collections of direct access volumes, all of the same device type. Each table space each index space normally has an associated storage group. When storage is needed for the space or partition, it is taken from the specified storage group. Within each storage group, spaces and partitions are stored using VSAM linear datasets. DB2 uses VSAM for such things as direct access space management, dataset cataloging, and physical transfer of pages into and out of memory.

Figure 2.5 data Storage (Cont….)

NORMALIZATION 1-NF

Normalization is the process of steps that will identify for elimination of redundancies in a database design.

- **FIRST NORMAL FORM: Eliminating repeating groups**

Unnormalized EMPL

EMPNO	LASTNME	WORKDEPT	DEPTNAME	SKILL1	SKILL2	SKILLN
000030	KWAN	GRE	OPERATIONS	141		
000250	SMITH	BLU	PLANNING	002	011	067
000270	PEREZ	RED	MARKETING	415	447	
000300	SMITH	BLU	PLANNING	011	032	

Figure 2.6 Normalization 1-NF

Notes:

Eliminating Repeating Groups:

One of the first rules of relational tables is that all the rows must have the same number of columns and each column must mean the same thing in the same sequence in each row.

In the example of EMP table, if it were required to represent the multiple skills of employees, this would be represented in an additional table where each skill for a given employee is represented in a row. If new skill is created and an employee already has the maximum number of skills allocated in the data structure, then the entire file structure and all the programs have to be changed, in an unnormalized file structure. If this table is normalized, then the new skill simply becomes a new row.

NORMALIZATION 1-NF (Cont….)

NORMALIZED TO 1NF-EMPL

EMPNO	LASTNAME	WORKDEPT	DEPTNAME
000030	KWAN	GRE	OPERATIONS
000250	SMITH	BLU	PLANNING
000270	PEREZ	RED	MARKETING
000300	SMITH	BLU	PLANNING

EMPL_SKILL TABLE

EMPNO	SKILL	SKILLDESC
000030	141	RESEARCH
000250	002	BID PREP
000250	011	NEGOTIATION
000250	067	PROD SPEC
000270	415	BENEFITS ANL
000270	447	TESTING
000300	011	NEGOTIATION
000300	032	INV CONTROL

Figure 2.7 Normalization 1-NF (Cont….)

.

NORMALIZATION 2-NF

Second normal form Eliminate the columns that depend only on part of the key

Normalized – 1NF

EMPNO	SKILL	SKILLDESC
000030	141	RESEARCH
000250	002	BID PREP
000250	011	NEGOTIATION
000250	067	PROD SPEC
000270	415	BRNRFITS ANL
000270	447	TESTING
000300	001	NEGOTIATION
000300	032	INV CONTROL

Figure 2.8 Normalization 2-NF

Notes:

Second normal form requires a close look at proposed tables that have multi-column primary keys. The objective is to eliminate all columns from a table design that are dependent on only part of the primary key.

NORMALIZATION 2-NF (Cont….)

Normalized - 2NF

EMPNO	SKILL	DATE CERT
000030	141	
000250	002	
000250	011	
000250	067	
000270	415	
000270	447	
000300	011	
000300	032	

SKILL	SKILLDESC
002	BID PREP
011	NEGOTIATION
032	INV CONTROL
067	PROD SPEC
141	RESEACH
415	BENEFITS ANL
447	TESTING

Figure 2.9 Normalization 2-NF (Cont….)

Notes:

In the example of EMP table in the 1NF the skills of the employees was matched with the employees in a separate table and the skill description. But skill description depends only upon the skill number. The problems that arise due to this are:

1. **The skill description will be repeated every time the skill is repeated.**

2. **If this description for the skill changes, then multiple rows will need to be changed, risking some update anomalies (data inconsistency due to not changing all rows).**

3. **If an employee is the sole source for a given skill and that person leaves the company, that particular skill and skill description is lost forever. (Delete anomaly).**

NORMALIZATION 3-NF

Third normalization: Eliminate columns that don't depend on the key at all.

Unnormalized EMPL

EMPNO	LAST NAME	WORK DEPT	DEPTNAME	MGRNO
000030	KWAN	GRE	OPERATIONS	000080
000250	SMITH	BLU	PLANNING	000010
000270	PEREZ	RED	MARKETING	000020
000300	SMITH	BLU	PLANNING	000010

Normalized to 3NF

EMPNO	LASTNAME	WORKDEPT
000030	KWAN	GRE
000250	SMITH	BLU
000270	PEREZ	RED
000300	SMITH	BLU

DEPTNO	DEPTNAME	MGRNO
BLU	PLANNING	000010
GRE	OPERATIONS	000080
RED	MARKETING	000020

Figure 2.10 Normalization 3-NF

Notes:

Third normal form tables ensure that each column's value depends on the primary key or some other column with the table.

In the example of the EMP table the Department number in the EMP table, that is necessary to reflect the relationship between the employee and the department tables – many employees work for one department and the department name column is a fact about the department number column. The same three problems exist here as the 2NF and can be solved in the same way – i.e. an additional table where the department number is the primary key and department name is dependent upon it.

NORMALIZATION 4-NF

Fourth normalization: Isolate independent multiple relationships, i.e. no table may contain more than 1:n or n:m relationships that aren't directly related.

Figure 2.11 Normalization 4-NF

Notes:

Suppose in the skills table of the 1NF the language that each employee knows is also included along with skills of each employee. So, now there is one-to-many relationship between employee and language in the same table, since an employee can know more than one language, which is a violation of fourth normalization.

A separate table has to be created isolating language from skills UNLESS THERE IS A DIRECT RELATIONSHIP BETWEEN LANGUAGE AND SKILL, i.e. an employee requires certain language for certain skill, they should remain together.

NORMALIZATION 5-NF

Fifth normalization: Isolate related multiple relationships

Figure 2.12 Normalization 4-NF

Notes:

Suppose in the above example the skills and languages are related to each other, there may be some other reason why they should be separated in to yet another table to resolve many-to-many relationship between them. So we should have three tables

1. **Employee-Skill**
2. **Employee-Language**
3. **Skill-Language**

NORMALIZATION RECOMMENDATION

- **Always Normalize to Third Normal Form.**
- **Analyze the table's Views To Evaluate The Benefits of the denormalizing.**

Figure 2.13 Normalization Recommendation

Notes:

Normalization solves many problems like it eliminates redundancies that saves disk space and reduces update and delete anomalies.

It helps in keeping data model flexible, which is a key factor in today's world of rapidly changing business requirements, but it may impact performance depending on user requirements if the data's are separated into many tables, but it is often retrieved, many I/O operations are required than if it is not normalized. These decisions and trade offs between normalizing and denormalizing has to be decided.

OPTIMIZER

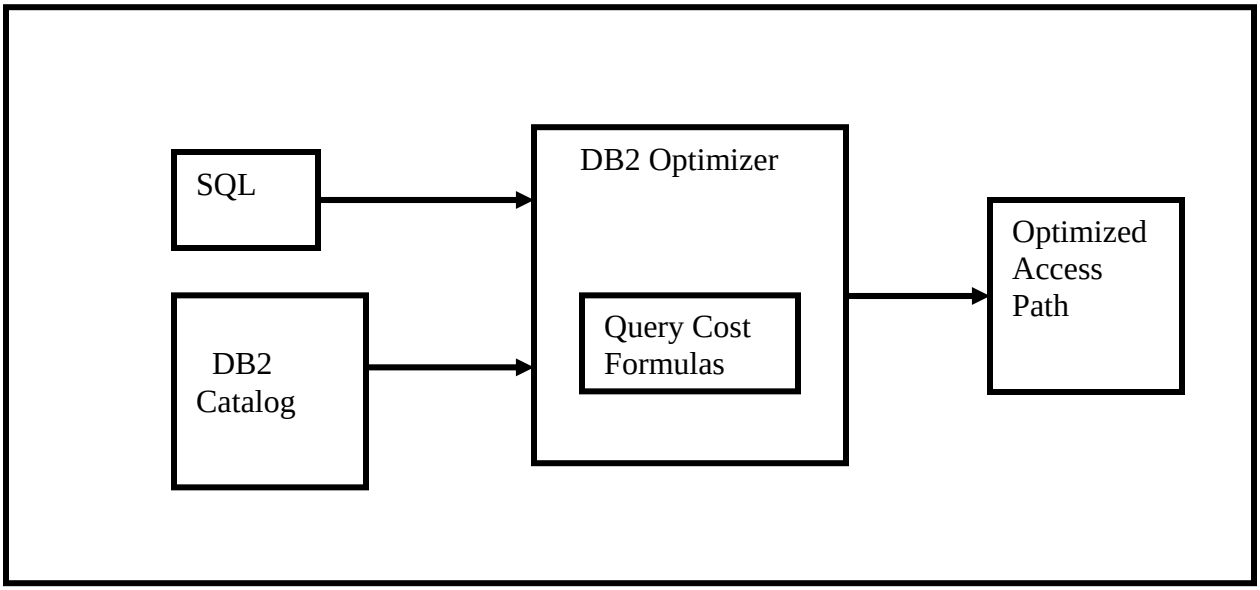

Figure 2.14 Optimizer

Notes:

The Optimizer is the most important part of DB2. It is the brain of DB2. It analyzes the SQL statements and determines the most efficient access path available for satisfying the statement.

In the SQL statements, we only say what we want, the optimizer decides how to get the things that we have asked for. It accomplishes this by parsing the SQL statements to determine which table must be accessed.

It then queries statistics stored in DB2 catalog to determine the best method of accomplishing the tasks necessary to satisfy the SQL statements.

OPTIMIZER (Cont….)

Physical Data Independence:

The concept of optimizing data access in the DBMS is one of the most power full capabilities of DB2. Access to DB2 data is achieved by telling DB2 what you want and not how to get it. DB2 ‘s optimizer accomplishes this task thus giving physical data independence. Regardless of how data is physically stored and manipulated, DB2 and SQL can still access data. This separation of access criteria from physical storage characteristics is called physical data independence. If indexes are removed DB2 can still access data (the performance is reduced). If a column is added to the table being accessed, the data can still be manipulated by DB2 with out changing the program code. This is all possible because the physical access path to DB2 data are not coded by programmers but generated by DB2.

Working of the optimizer:

The optimizer performs complex calculations based on a host of information. Four steps of optimizers are

1. **Receive and verify the SQL statement.**
2. **Analyze the environment and optimize the method of satisfying the SQL statement.**
3. **Create machine-readable instructions to execute the optimized SQL.**
4. **Execute the instructions or store them for future execution.**

The optimizer has many types of strategies for optimizing the SQL. The strategy is a cost based one. This means that the optimizer will always attempt to find an access path that will reduce the overall cost. To accomplish this the DB2 optimizer evaluates four factors for each potential access path. Those factors are

1.CPU cost
2.I/O cost
3.DB2 catalog statistics
4.SQL statements.

Figure 2.15 Optimizer (Cont….)

DB2 CATALOG AND DIRECTORY

- **DB2 has a set of tables that function as a repository for all DB2 objects, the DB2 Catalog and DB2 Directory. The entire DBMS relies on the DB2 Catalog.**

- **The DB2 Catalog is composed of eleven table spaces and 43 tables all in a single database DSNDB06.**

- **Each DB2 Catalog table maintains data about an aspect of DB2 environment. In the respect, DB2 catalog functions as a data dictionary for DB2, supporting and maintaining data about the DB2 environment.**

- **The DB2 catalog records all information required by DB2 about STOGROUPS, data spaces, table spaces, partitions, tables, columns, views, synonyms, aliases, indexes, index keys, foreign keys, relationships, plans, packages, DBRM s, database privileges, plan privileges, system privileges, table privileges, view privileges, use privileges, image copy datasets, REORG executions.**

- **The DB2 catalog is built, maintained and used as a result of the creation of objects defined to the catalog. In other words as the user creates or modifies the DB2 objects, metadata (data about data) is being accumulated in the catalog. The DB2 catalog can be updated using the RUNSTATS utility.**

Figure 2.16 DB2 Catalog and Directory

CATALOG STRUCTURE

- **The DB2 catalog is structured as DB2 tables, but they are not standard tables. Many of the DB2 catalog tables are stored together hierarchically using a special relationship called 'link'.**

- **The DB2 catalog table SYSIBM.SYSLINKS store the information defining the relationships between other catalog tables.**

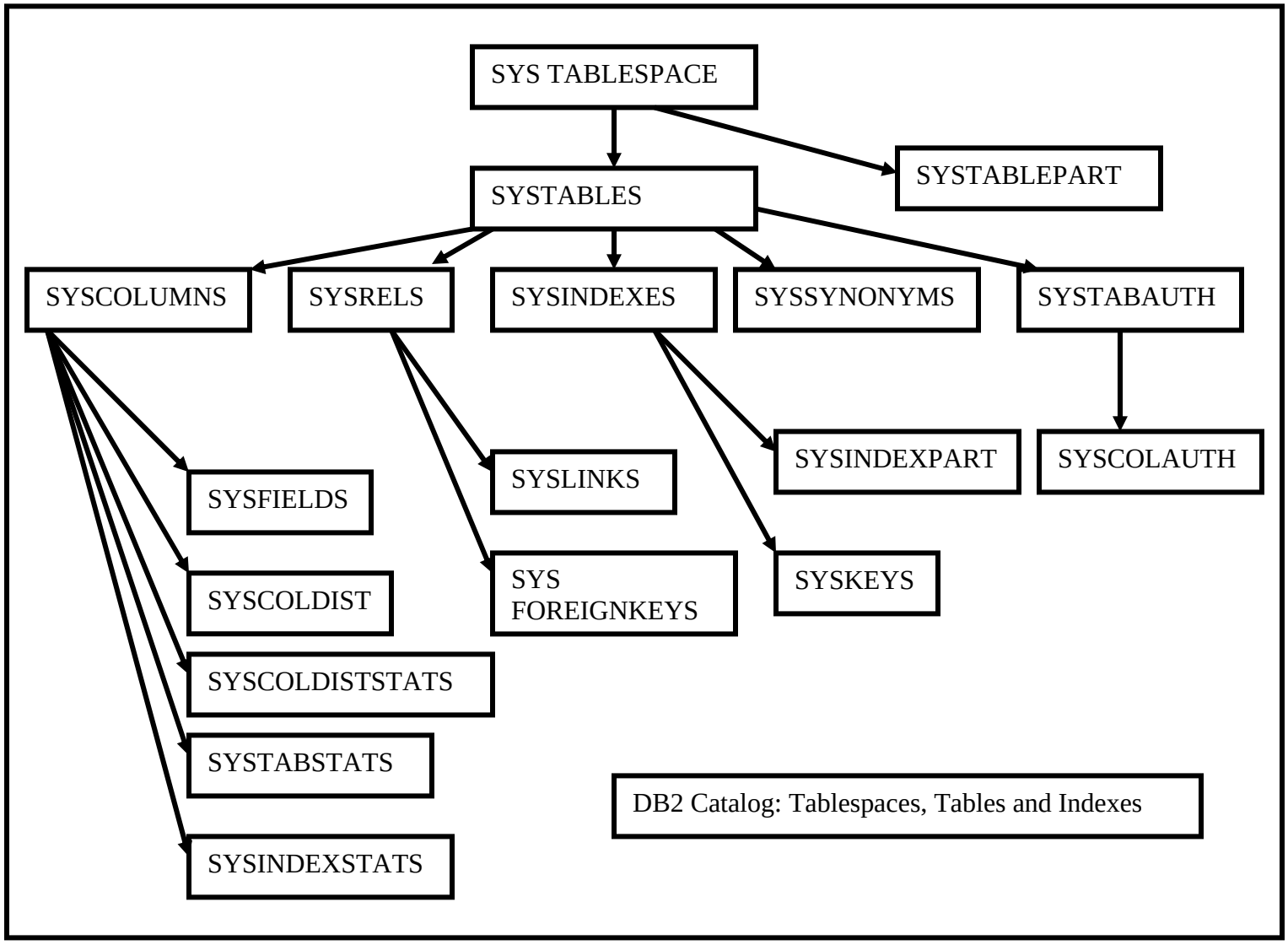

Figure 2.17 Catalog Structure

CATALOG STRUCTURE (Cont….)

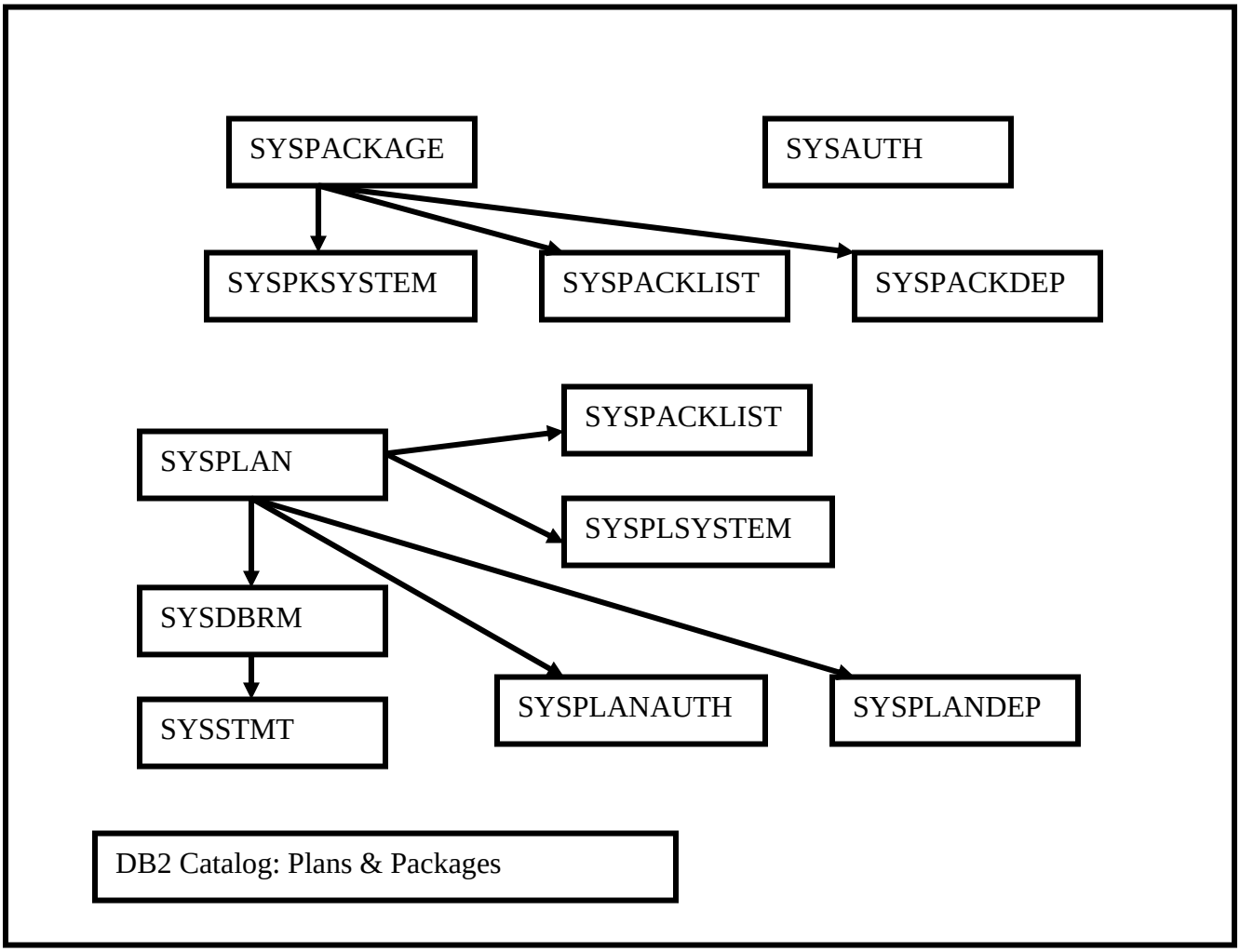

Figure 2.18 Catalog Structure (Cont….)

CATALOG STRUCTURE (Cont….)

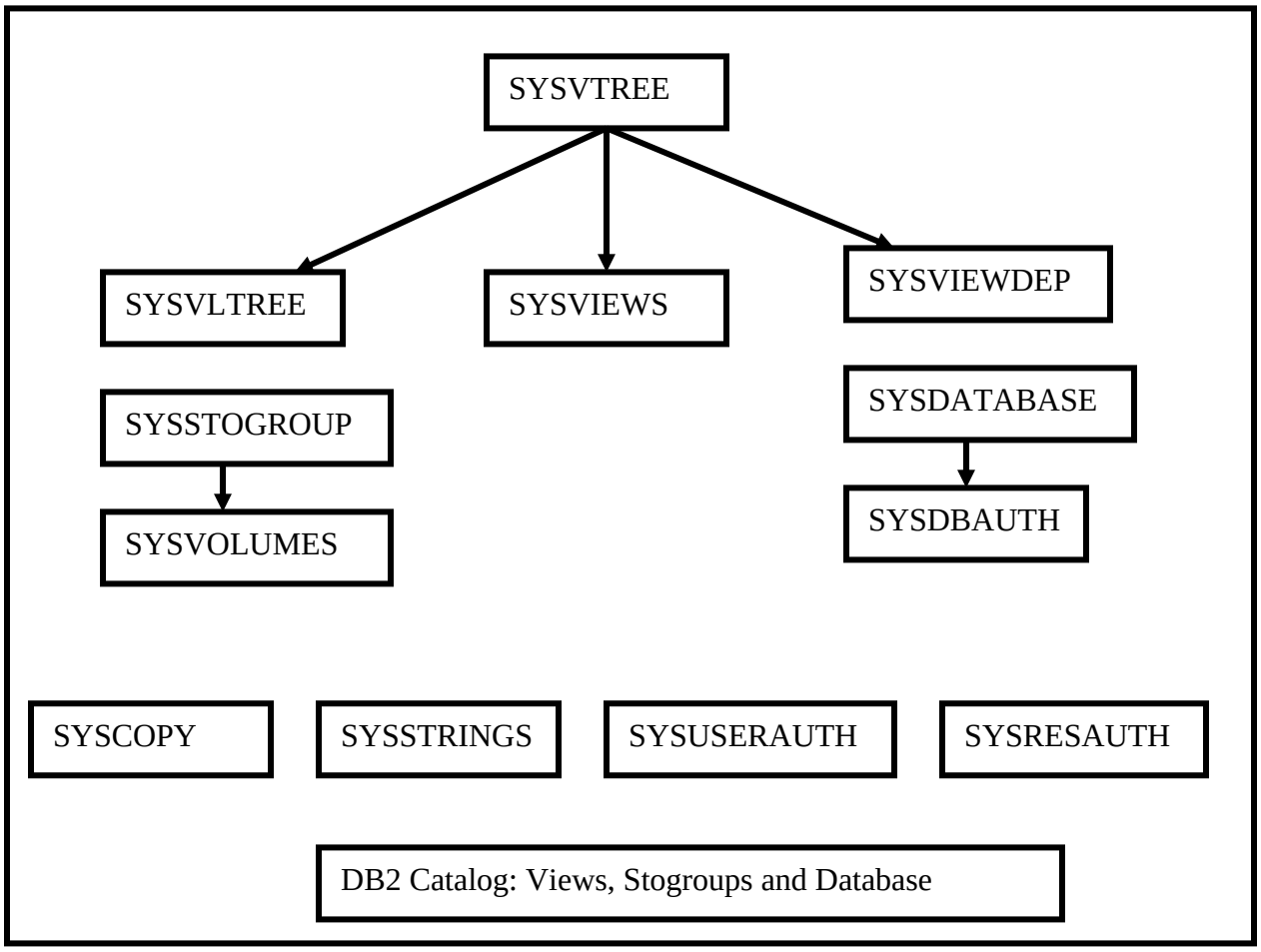

Figure 2.19Catalog Structure (Cont….)

DB2 DIRECTORY

- **DB2 uses a second data directory, other than the catalog, called the DB2 Directory for storing detailed technical information about the aspects of DB2 ‘s operation.**

- **The DB2 directory is for DB2 ‘s internal use only. The SQL statements cannot access the data in the DB2 directory.**

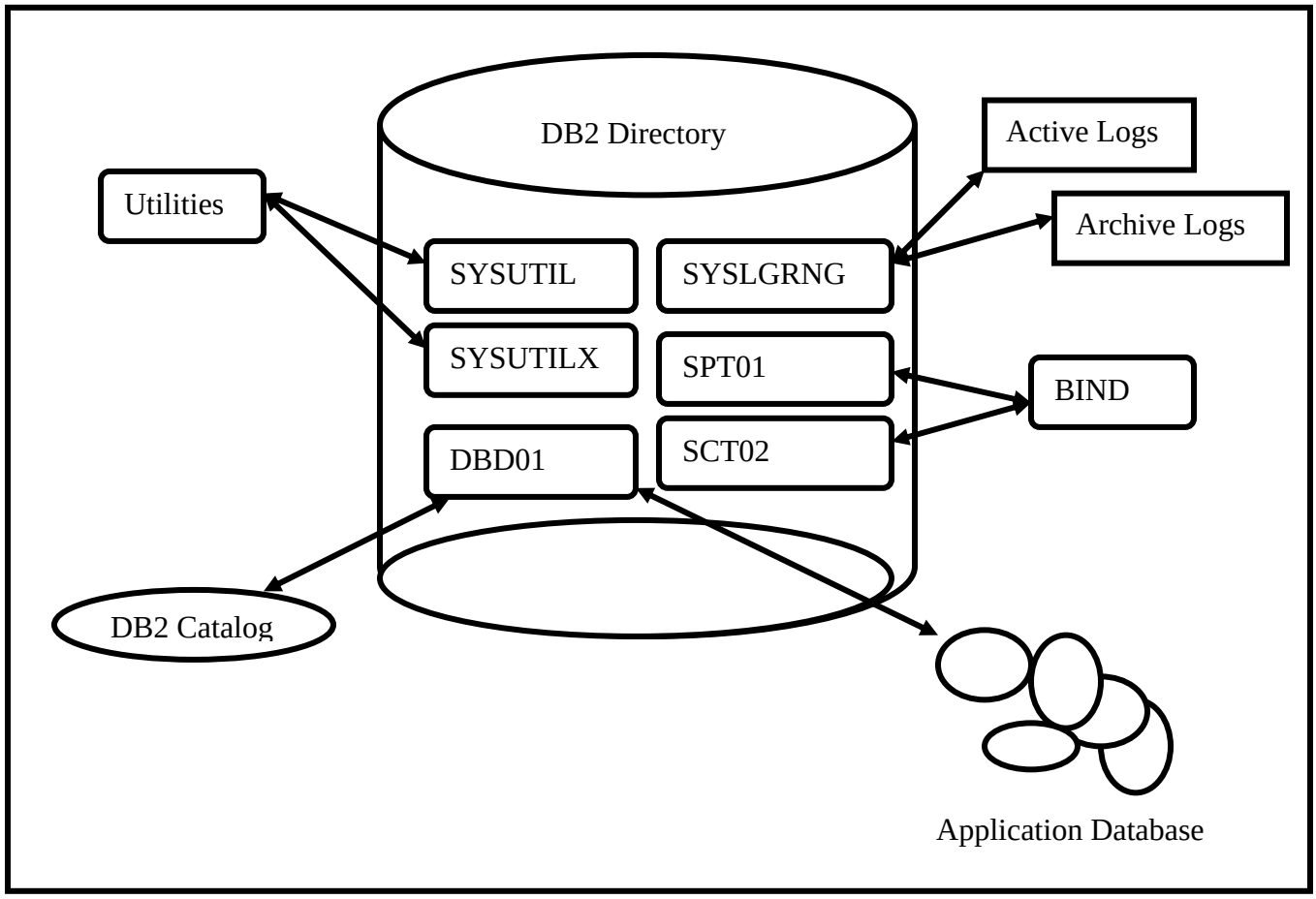

Figure 2.20 DB2 Directory

DATA LOCKING

DB2 guarantees data integrity by using several locking mechanisms.

Why data locking?

When multiple users can access and update the same data at the time, a locking mechanism is required. This mechanism must be capable of differentiating between stable data and uncertain data. Stable data has been successfully committed and is not involved in an update in a current unit of work. Uncertain data is currently involved in an operation and could get modified. If two users try to update the same record at the same time, or if one user deletes a record while some other user is trying to update it, with out proper locking strategy and locking mechanisms, the data integrity can go haywire. To avoid such situations a DBMS uses a locking mechanism.

Figure 2.21 Data Locking

LOCKING MECHANISM

DB2 supports locking at three levels. Table space level, Table level and Page level. Locks can be taken at any level in the locking hierarchy without taking a lock at the lower level. However, locks cannot be taken at a lower level with out compatible higher-level lock also being taken. For example, we can take a table space lock with out taking any other lock, but for taking a page lock we have to first get a table space lock. Different modes of locking are:

- **Locks to enable reading of data.**
- **Locks to enable updating of data.**

Table space/ Table locks			
Lock	Meaning	Access Required	Access Allowed To Others
S	SHARE	Read only	Read only
U	UPDATE	Read with intent to Update	Read only
X	EXCLUSIVE	Update	No access
IS	INTENT SHARE	Read only	Update
IX	INTENT EXCLUSIVE	Update	Update
SIX	SHARE/INTENT EXCLUIVE	Read or Update	Read only
Page Locks			
S	SHARE	Read only	Read only
U	UPDATE	Read with intent to Update	Read only
X	EXCLUSIVE	Update	No access

Figure 2.22 Locking Mechanism

LOCKS VS LATCHES

A true lock is handled by DB2 using IRLM (IMS Resource Lock Manager). DB2 will try to lock pages without going to IRLM, whenever it is possible. This type of lock is referred to as a latch.

True locks are always set in the IRLM. Latches, by contrast are set internally by DB2 with out going to the IRLM. When a latch is taken instead of lock, it is handled by internal DB2 code. So cross memory service calls to IRLM is eliminated. Also a latch requires about one third of instruction as a lock. Therefore, latches are more efficient than locks because they avoid the overhead associated with calling an external address space.

Latches are used when a resource serialization is required for a short time. Prior to DB2 version 3, latches were generally used to lock only DB2 index spaces and internal DB2 resources. When running V3, DB2 uses latching more frequently. This includes data page latches.

Figure 2.23 Locks Vs Latches

DEADLOCKS AND TIMEOUTS

- **Locks can be used to solved problems of concurrency, and ensure data integrity. But unfortunately, locking introduces a problem called deadlock.**

- **Deadlock is a situation in which two or more transactions are in simultaneous wait state, each waiting for one of the others to release a lock before it can proceed.**

- **If a deadlock occurs the system will detect and break it. Breaking a deadlock involves choosing one of the deadlock transactions as the victim and depending on the transaction manager concerned, either rolling it back automatically or requesting it to roll back itself. Either way, the transaction will release its locks and thus allow some other transactions to proceed.**

- **In general, therefore, any operation that requests a lock may be rejected with a negative SQL code indicating the transaction has been selected as the victim of a deadlock situation and has either been rolled back or request to roll back.**

- **A time-out is caused by the unavailability of a given resource. Time outs are caused when a transaction has to wait for a resource, which is held by another transaction. The duration of the waiting time, after which the time-out should occur is determined by the IRLMRWT DSNZPARM parameter**

Figure 2.24 Deadlocks and Timeouts