



DB Fundamentals

People matter, results count.

Introduction to Database



- A database is a computerized record keeping system.
- As such, the purpose of a database is to store a collection of logically related data
- Databases are used whenever a large amount of data is needed to be stored and retrieved quickly and easily
- The database not only stores data, but also a description of the data.

Why do we need Database?

➤ Databases are used whenever a large amount of data is needed to be stored and retrieved quickly and easily

▪ Example 1

- You and your project partner are editing the same
- You both save it at the same time.
- Whose changes survive?



=



Is filesystem and database are same?

1) Yours

2) Partner's

3) Both

4) Neither

Example 2

- You're updating a file.
- The power goes out.
- Which of your changes survive?

1) All

2) None

3) All Since Last Save

4) ???

Client-Server Systems

Database functionality can be divided into:

- **Back-end:** manages access structures, query evaluation and optimization, concurrency control and recovery.
- **Front-end:** consists of tools such as forms, report-writers, and graphical user interface facilities.

The interface between the front-end and the back-end is through SQL or through an application program interface.

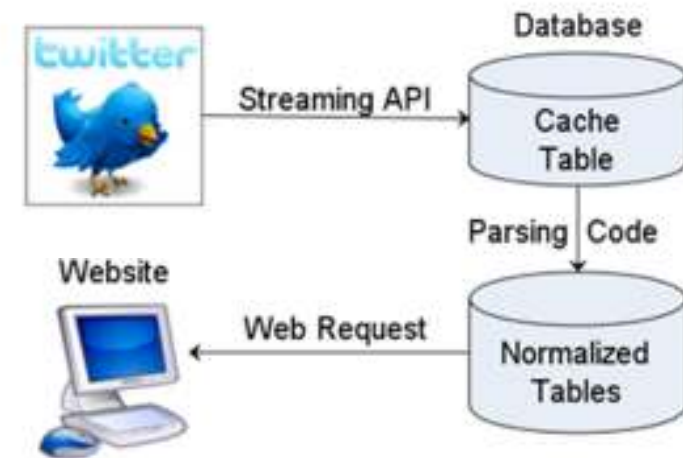
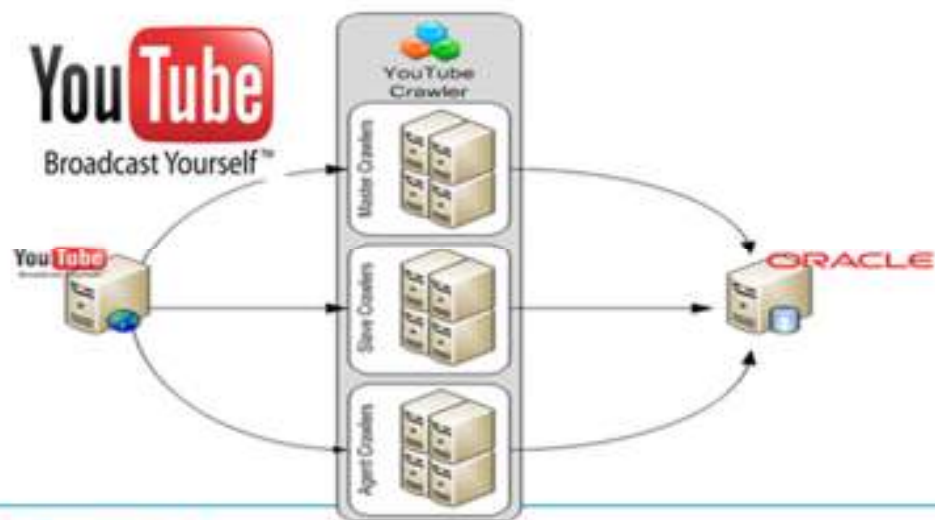
FRONT END



BACK END



Few examples of the usage of databases:



Introduction to Database (Cont...)

History:

A database is a term used to describe an ordered set of information stored on a computer. This ordered set of data or data set is often structured using a data modeling solution in such a way as to make the retrieval of that data more efficient. Depending on the type of applications using the database, the database structure can be modified to allow for efficient changes to that data

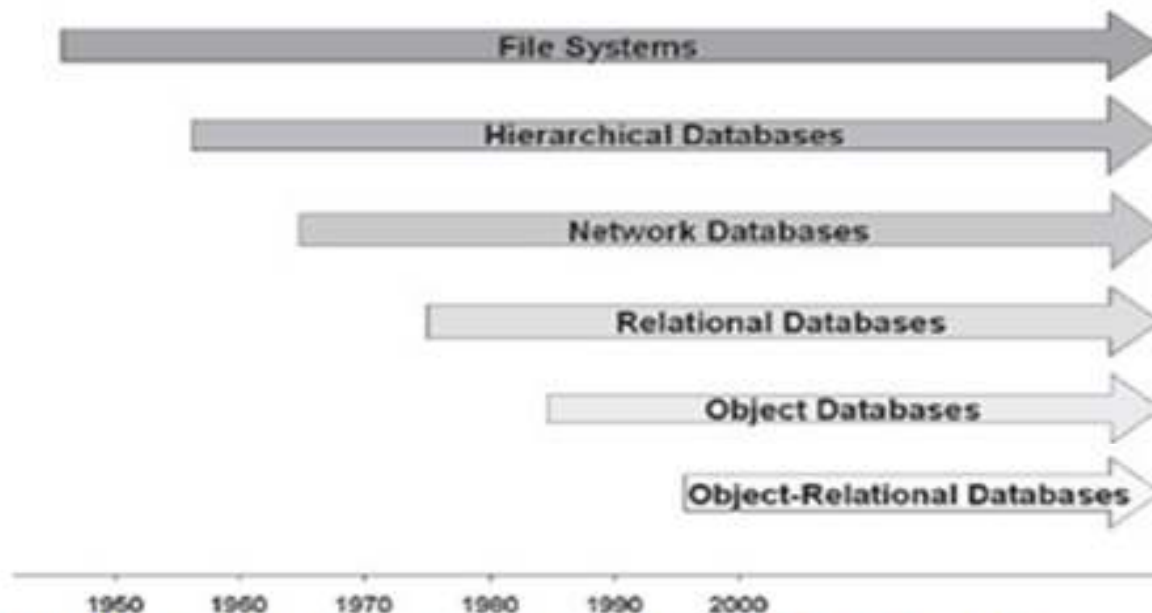


Figure 4.1: The History and Evolution of Data Modeling

Introduction to Database (Cont...)

Database Models:

A database model is a theory or specification describing how a database is structured and used.

- Flat Model
- Hierarchical model
- Network model
- Relational model
- Object-oriented model

Introduction to Database (Cont...)

Flat File

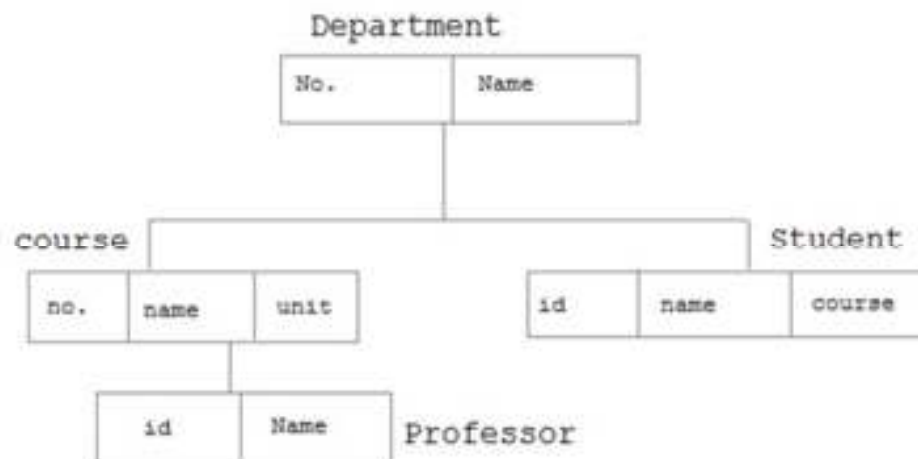
Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack, seal

Not easy to

- Search
- Retrieve
- Insert
- Delete

Hierarchical Model:

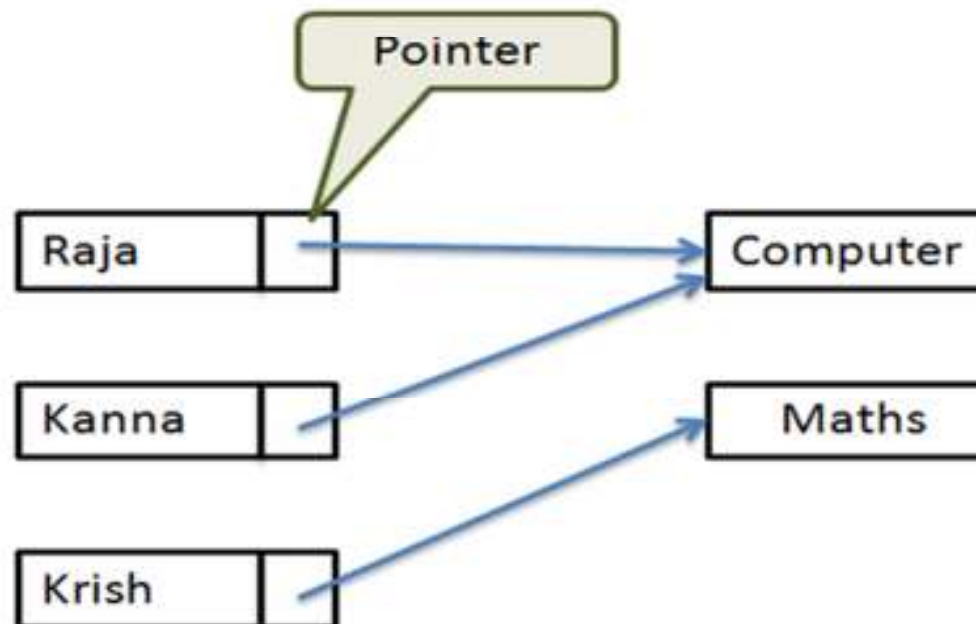


Introduction to Database (Cont...)

Network Model:

- Data are represented as collection of records and their relationship are represented as links (pointer)

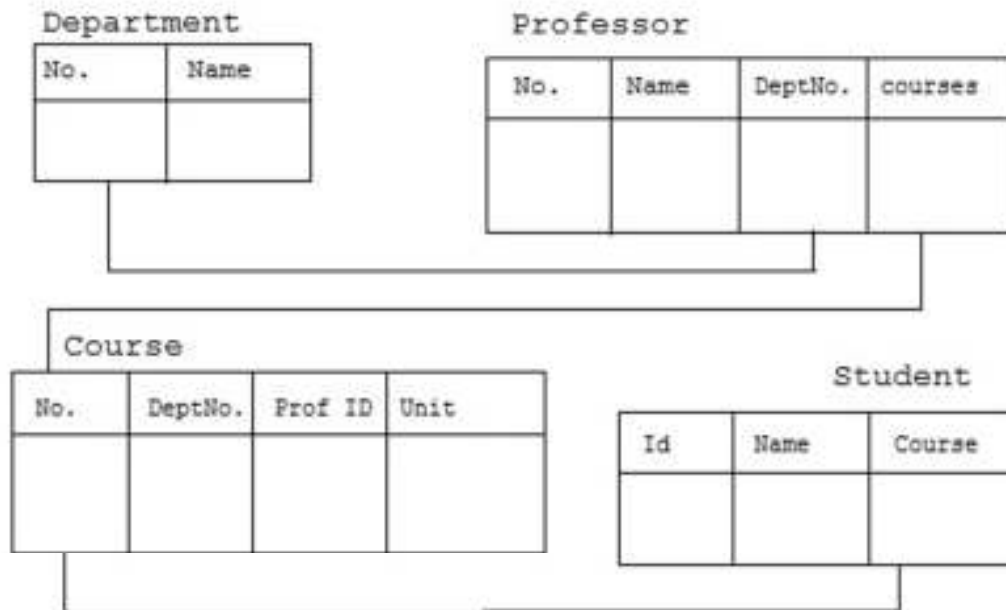
➤ Pointers create the complexity
➤ More space for pointer value



Introduction to Database

Relational Model:

- It is a data model in which both data and their relationships are represented by means of table



➤ Reusability of structure is not possible
➤ Alteration & Maintenance of structure is difficult

Introduction to Database (Cont...)

Object Relational Model:

- In this model data are organized as an object

Object 1: Maintenance Report

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

Object 1 Instance

01-12-01
24
1-95
2.5
6.0
6.0

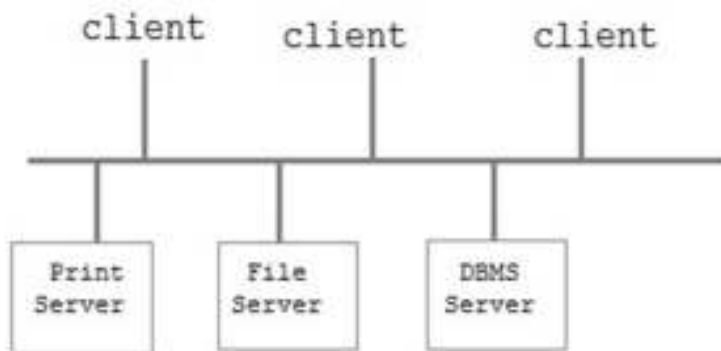
Object 2: Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	

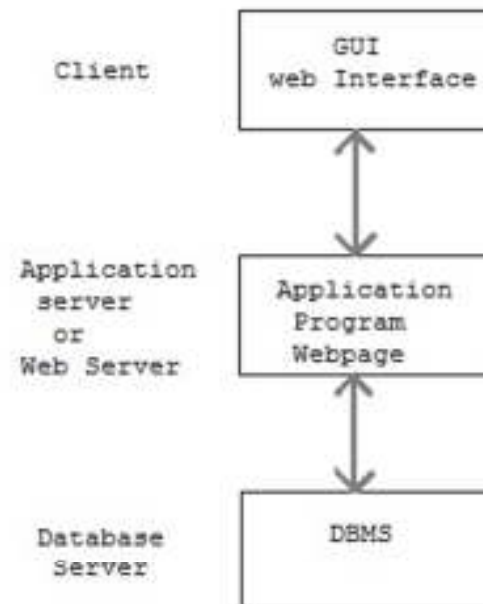
- Reusability of structure is possible
- supports to define interface for the application

Database Architecture

Two-tier Client / Server Architecture



Three-tier Client / Server Architecture



Introduction to Database (Cont...)

Database Concepts:

File-Based Approach:

- Each program defines and manages its own data
- Limitation
 - Separation and isolation of data
 - Duplication of data
 - Data dependence
 - Incompatibility of files
 - Fixed queries/proliferation of application program

Database Approach:

- A shared collection of logically related data, designed to meet the information needs of an organization

Introduction to Database (Cont...)

File Based Approach:



Managing data is very difficult

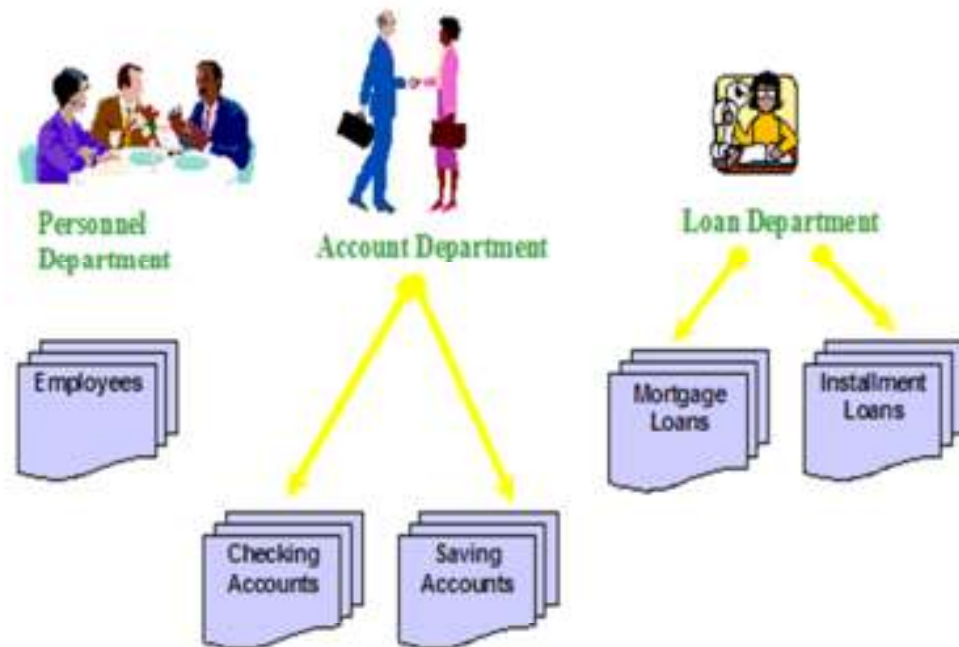
- The file-based approach refers to the process of using application programs to manage user data.
- This approach is an extension of the manual filing system.
- A decentralized approach was taken, whereby each application, or department in a company would store its own information.

Introduction to Database (Cont...)

Disadvantages in using file based approach:

The file based approach has a few limitations, such as :

- 1) Integrity Problems
- 2) Concurrent access anomalies
- 3) Data Isolation
- 4) Unanticipated queries
- 5) Data redundancy
- 6) Data inconsistency
- 7) Duplication of data



Introduction to Database (Cont...)

Database Concepts:

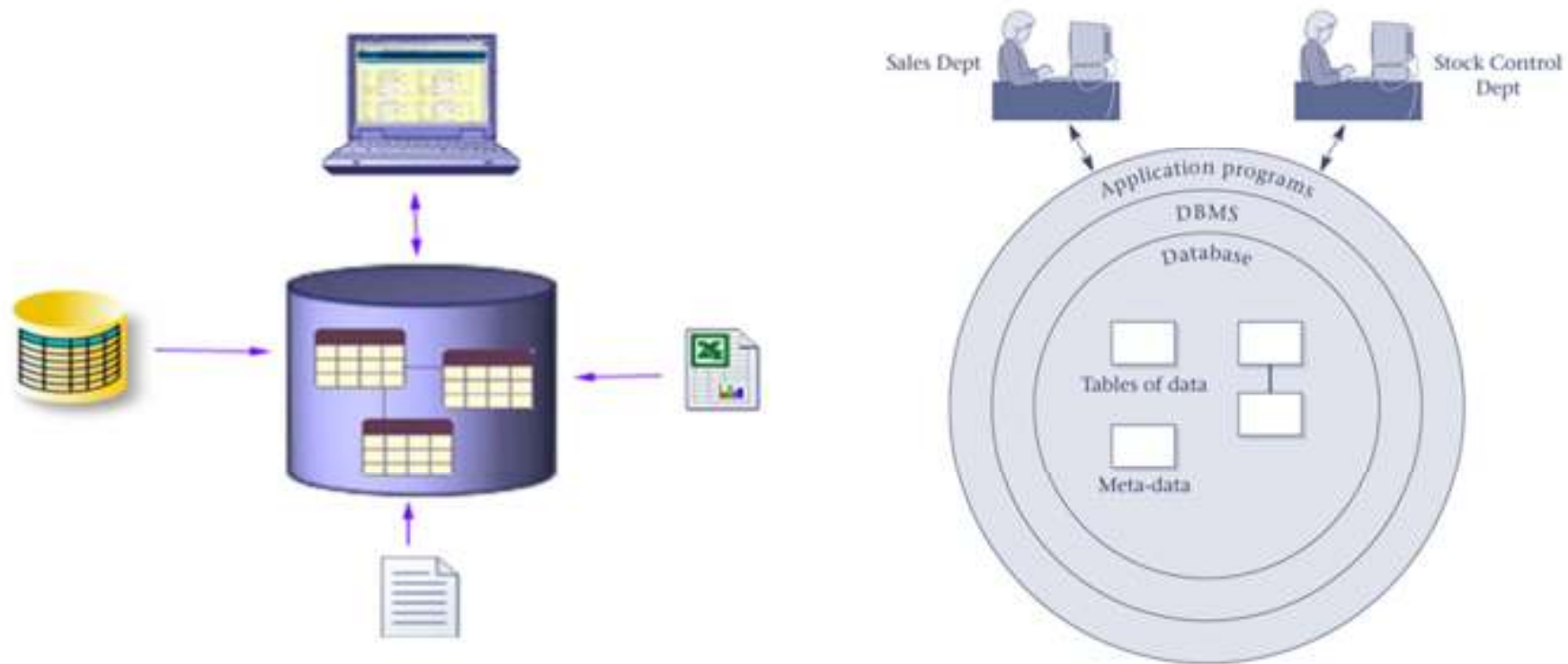
Database Approach:

- A shared collection of logically related data, designed to meet the information needs of an organization



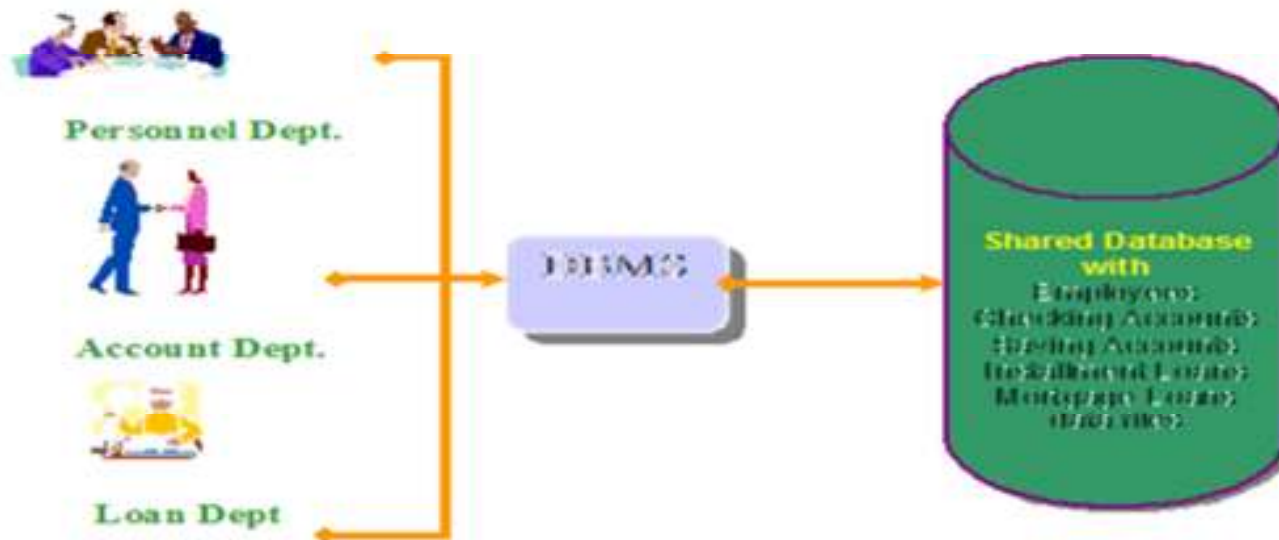
Database Management System

- A software system that enables users to define, create, and maintain the database and provides controlled access to the same database



Database Management System(Cont...)

DBMS – Advantages:

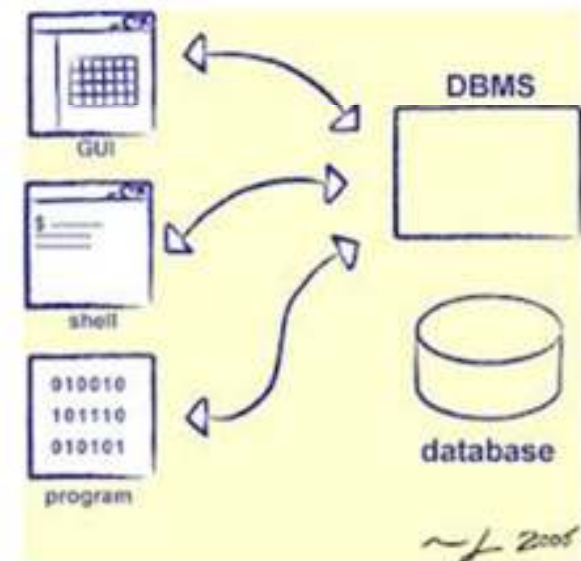


- Control of data redundancy
- Data consistency
- More information from the same amount of data
- Sharing of data
- Improved security

Database Management System(Cont ...)

Functions of DBMS:

1. Data storage, retrieval, and update
2. A user-accessible catalogue – (Ex. Tables, DB Objects)
3. Transaction support
4. Concurrency control services
5. Recovery services
6. Authorization services
7. Support for data communication
8. Integrity services
9. Services to promote data independence
10. Utility services



Database Management System(Cont...)

- **ACID Properties:**



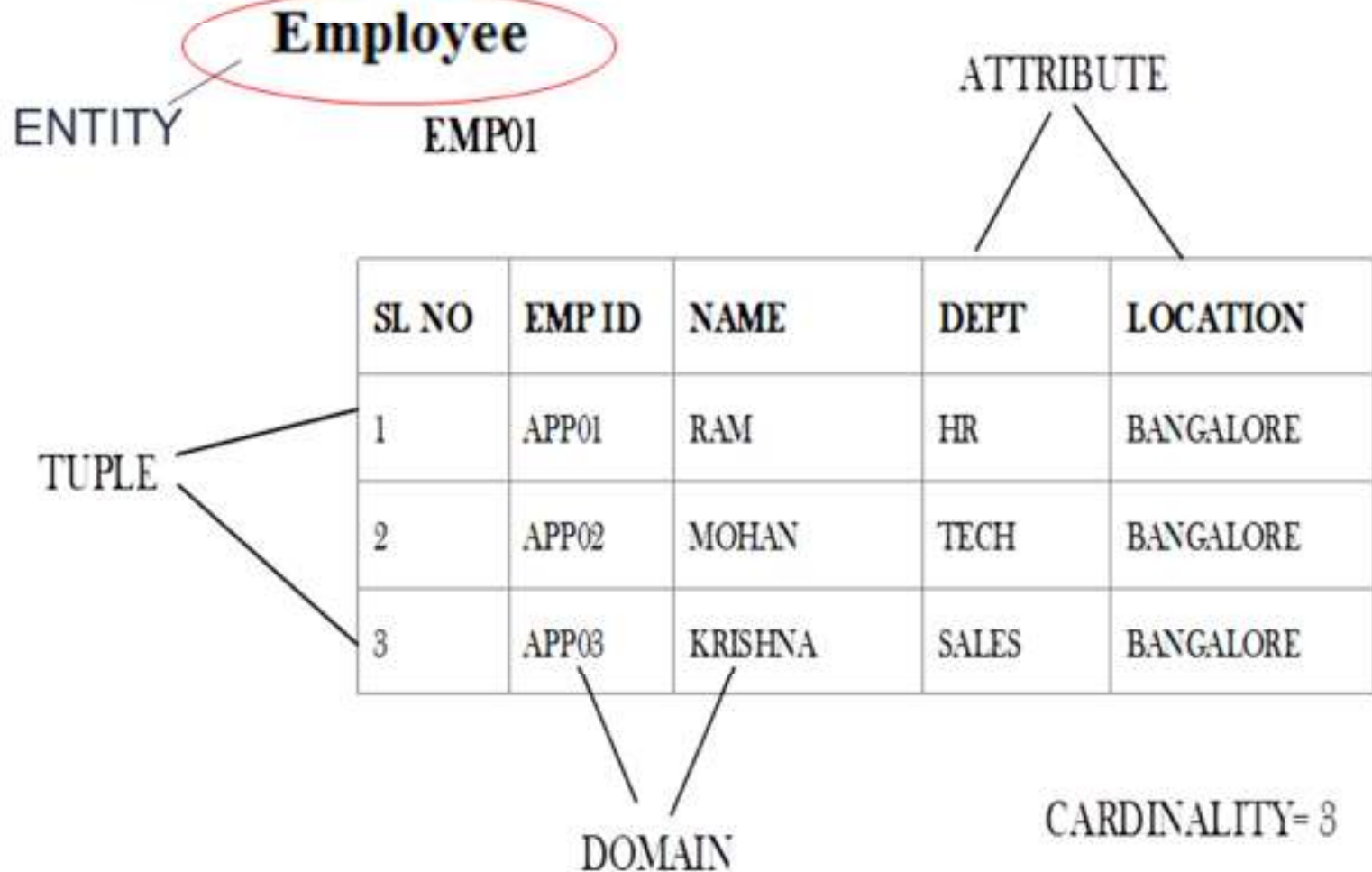
Relational Database Management System

Terminology:

- Relation
- Entity
- Attribute
- Domain
- Tuple
- Degree
- Cardinality
- Relational database

RDBMS (Cont...)

Terminology Explanation:



RDBMS

What is RDBMS?



- RDBMS is based on relational model, in which data is represented in the form of relations, with enforced relationships between the tables.
Example: Oracle, SQL
- RDBMS is the one which fulfills the E.F Codd rules.
- In RDBMS, normalization process will be present to check the database table consistency
- RDBMS helps in recovery of the database in case of loss of database due to system failure or any other reason



The SQL component in an RDBMS provides the means to make changes to database records.

RDBMS

RDBMS	DBMS
In RDBMS data is stored in rows and columns in a table. you can maintain a relation between the tables.	In DBMS data is stored in rows and columns in a table. you cannot maintain a relation between the tables
RDBMS follow the CODD's rule	DBMS does not follow the CODD's rule
RDBMS is used to store large amount of data and highly secured	DBMS is used to store small amount of data where the security is not a major concern
RDBMS follows relational data model	DBMS follows network, hierarchical or other data model.
eg: oracle, mysql, IBM DB2	eg: sysbase, foxpro



RDBMS Concepts

Table:

The data in RDBMS is stored in database objects called tables. The table is a collection of related data entries and it consists of columns and rows. Remember, a table is the most common and simplest form of data storage in a relational database

Example of a CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	20	DELHI	6500.00
2	Khilan	22	CHENNAI	8500.00
3	Kaushik	26	COIMBATORE	10000.00

RDBMS Concepts(Cont...)

Field:

- Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.
- A field is a column in a table that is designed to maintain specific information about every record in the table.

Record or Row:

A record, also called a row of data, is each individual entry that exists in a table. For example there are 7 records in the above CUSTOMERS table. Following is a single row of data or record in the CUSTOMERS table:

3	Kaushik	26	Coimbatore	10000.00
---	---------	----	------------	----------

RDBMS Concepts(Cont...)

Column:

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

For example, a column in the CUSTOMERS table is ADDRESS which represents location description and would consist of the following:

ADDRESS
DELHI
CHENNAI
COIMBATORE

RDBMS Concepts(Cont...)

NULL value:

A NULL value in a table is a value in a field that appears to be blank which means A field with a NULL value is a field with no value.

It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.

RDBMS (Cont...)

RDBMS Concepts – Entity - Relationship Modelling:

- **Concepts of the E-R Modelling:**
 - **Entity Types**
An object or concept that is identified by the enterprise as having an independent existence
 - **Attributes**
A property of an entity or a relationship type
 - **Relationship Types**
A meaningful association among entity types

RDBMS (Cont...)

SQL Constraints:

- Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.
- Constraints could be column level or table level

Commonly used constraints available in SQL:

NOT NULL Constraint: Ensures that a column cannot have NULL value.

DEFAULT Constraint : Provides a default value for a column when none is specified.

UNIQUE Constraint: Ensures that all values in a column are different.

PRIMARY Key: Uniquely identified each rows/records in a database table.

FOREIGN Key: Uniquely identified a rows/records in any another database table.

CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.

INDEX: Use to create and retrieve data from the database very quickly.

RDBMS (Cont...)

Data Integrity:

The following categories of the data integrity exist with each RDBMS:

Entity Integrity : There are no duplicate rows in a table.

Domain Integrity : Enforces valid entries for a given column by restricting the type, the format, or the range of values.

Referential integrity : Rows cannot be deleted, which are used by other records.

User-Defined Integrity : Enforces some specific business rules that do not fall into entity, domain, or referential integrity.

RDBMS (Cont...)

EF Codd Rules:

- **Structural Rule**
 - Rule 1 : Information Representation
 - Rule 6 : View Updating
- **Integrity Rule**
 - Rule 3 : Systematic Treatment of Null Values
 - Rule 10 : Integrity Independence
- **Data Manipulation Rule**
 - Rule 2 : Guaranteed Access
 - Rule 4 : Dynamic online catalog
 - Rule 5 : Comprehensive Data Sublanguage
 - Rule 7 : High-Level Insert, update, delete
- **Data Independence Rule**
 - Rule 8 : Physical data independence
 - Rule 9 : Logical data independence
 - Rule 11 : Distribution independence
- **Foundational rules:**
 - Rule 0 : Foundational rule
 - Rule 12 : Non subversion rule

RDBMS – EF Codd Rules(Cont...)

Foundational rules:

Rule 0 : Foundational rule

Rule 12 : Non subversion rule

Structural rules:

Rule 1 : Information representation

All data should be presented to the user in table form.

Rule 6 : View updating

Data can be presented to the user in different logical combinations, called views. Each view should support the same full range of data manipulation that direct-access to a table has available. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

RDBMS – EF Codd Rules(Cont...)

Integrity rules:

Rule 3 : Systematic treatment of null values

A field should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero. Of course, this can't apply to primary keys. In addition, most database implementations support the concept of a non- null field constraint that prevents null values in a specific table column.

Rule 10 : Integrity independence

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all databases do preserve two constraints through SQL.

- No component of a primary key can have a null value. (see rule 3)
- If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

RDBMS – EF Codd Rules(Cont...)

Data Manipulation Rules:

Rule2 : Guaranteed access

All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key, and column name..

Rule 4 : Dynamic online catalog based on the relational model

A relational database must provide access to its structure through the same tools that are used to access the data. This is usually accomplished by storing the structure definition within special system tables.

RDBMS – EF Codd Rules(Cont...)

Data Manipulation Rules:

Rule5 : Comprehensive data sublanguage

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity, and database transaction control. All commercial relational databases use forms of the standard SQL (Structured Query Language) as their supported comprehensive language.

Rule7 : High-level insert, update, delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

RDBMS – EF Codd Rules(Cont...)

Data Independence Rules:

Rule8 : Physical data independence

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accesses it.

Rule 9 : Logical data independence

How a user views data should not change when the logical structure (tables structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the user view of the data and the actual structure of the underlying tables.

Rule11 : Distribution independence

A user should be totally unaware of whether or not the database is distributed (whether parts of the database exist in multiple locations). A variety of reasons make this rule difficult to implement; I will spend time addressing these reasons when we discuss distributed databases.

Dependencies

- Functional dependencies – direct dependency
- Transitive dependencies – indirect dependency
- Partial dependencies – direct and indirect dependency

Determinant

- Attribute on the LHS is known as the determinant
EmpNum is a determinant of EmpEmail
- Functional Dependency
 $\text{EmpNum} \rightarrow \text{EmpEmail}$

Functional Dependencies

- We say an attribute B, has a functional dependency on another attribute A, if for any two records, which have the same value for A, then the values for B in these two records must be the same.

- We illustrate this as:

$$A \rightarrow B$$

- Example

- Suppose we keep track of employee email addresses, and we only track one email address for each employee.
- Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number
employee number \rightarrow email address

Functional Dependencies

EmpNum	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

- If EmpNum is the primary key then the Functional Dependencies are,

EmpNum \rightarrow EmpEmail

EmpNum \rightarrow EmpFname

EmpNum \rightarrow EmpLname

Transitive dependency

- Consider attributes A, B and C where,

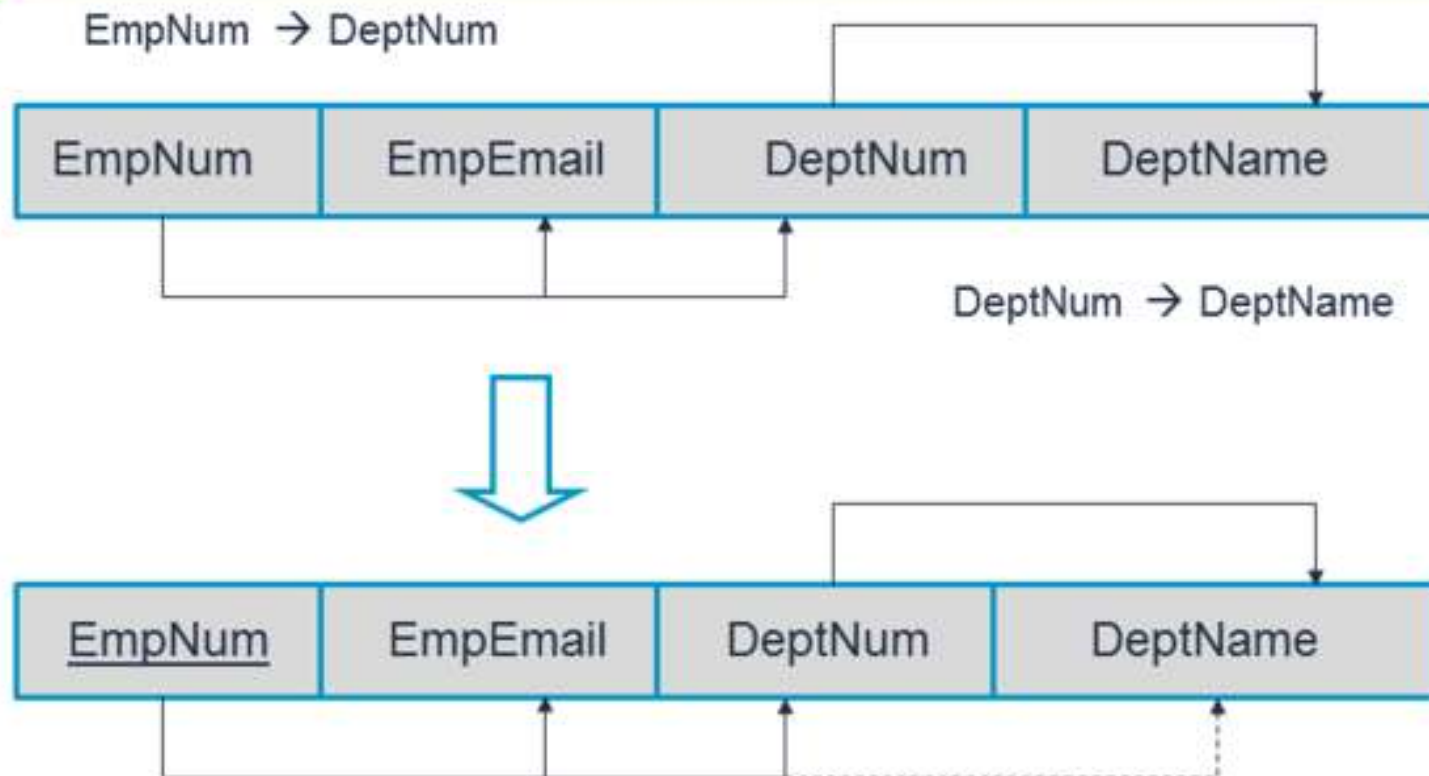
$$A \rightarrow B \text{ and } B \rightarrow C$$

- Functional dependencies are transitive, which means that we also have the functional dependency

$$A \rightarrow C$$

- We say that C is transitively dependent on A through B

Transitive dependency

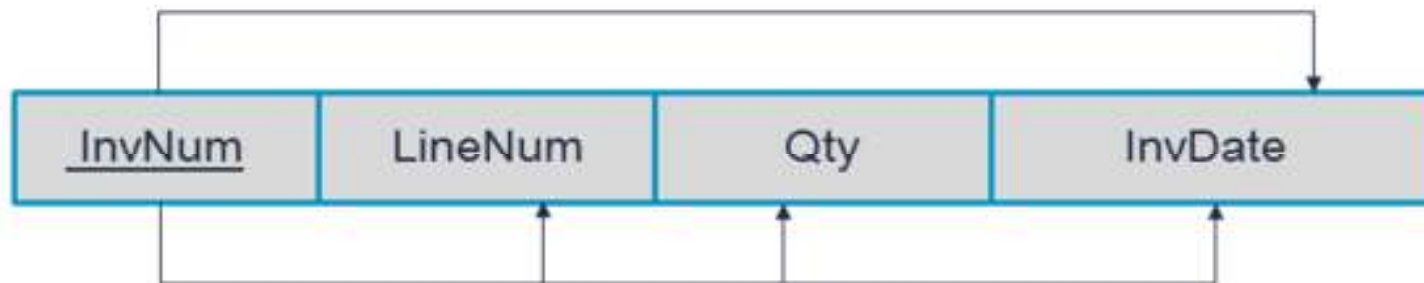


- DeptName is transitively dependent on EmpNum via DeptNum

$\text{EmpNum} \rightarrow \text{DeptName}$

Partial dependency

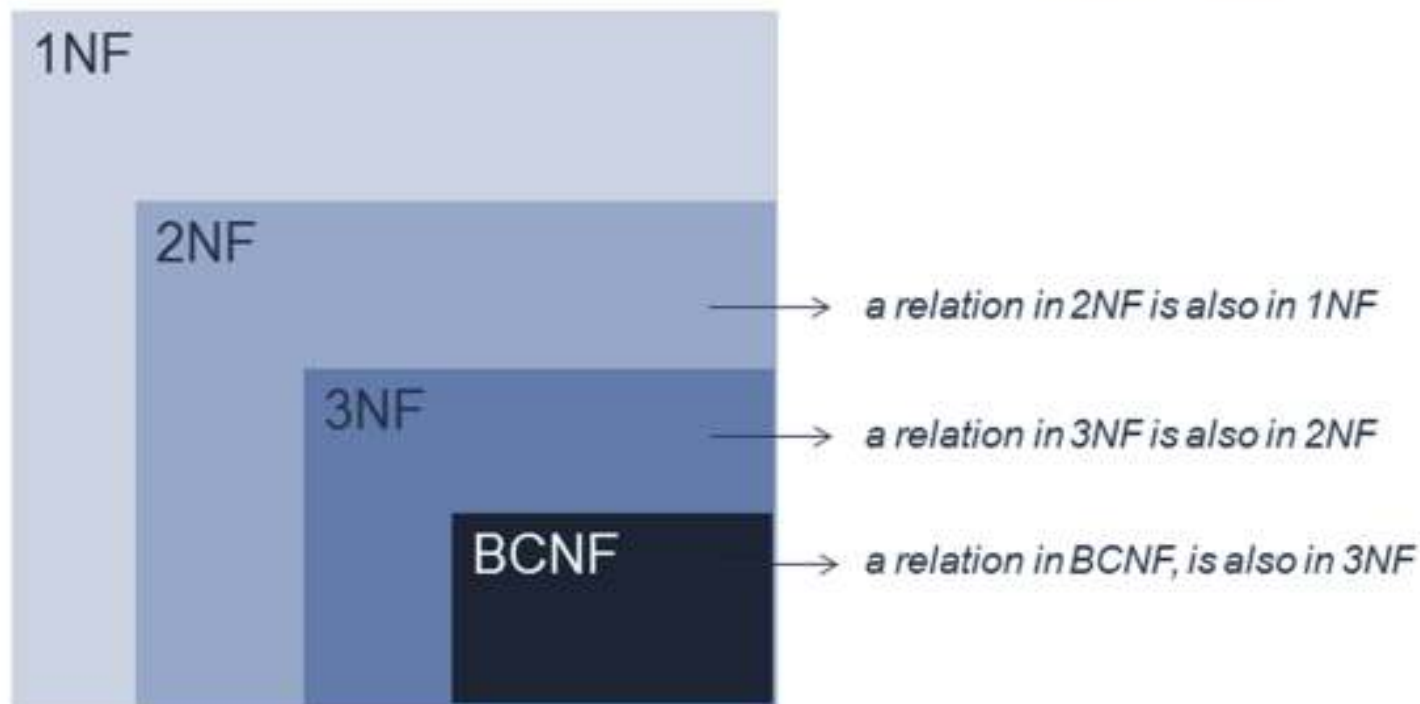
- A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



- Candidate keys: {InvNum, LineNum}
- InvDate is partially dependent on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

Normalization

- Normalization is a method for organizing data elements in a database into tables.
- Normalization is used for mainly two purpose,
 - Eliminating redundant(useless) data.
 - Ensuring data dependencies make sense i.e. data is logically stored.



Normalization Avoids

- **Duplication of Data**

The same data is listed in multiple lines of the database.

- **Insert Anomaly**

A record about an entity cannot be inserted into the table without first inserting information about another entity. Cannot enter a customer without a sales order.

- **Delete Anomaly**

A record cannot be deleted without deleting a record about a related entity. Cannot delete a sales order without deleting all of the customer's information.

- **Update Anomaly**

Cannot update information without changing information in many places. To update customer information, it must be updated for each sales order the customer has placed.

First Normal Form

- Data should be represented in table format
- Intersection of row and column should contain only one value
- No two rows of data must contain repeating group of information

ID	Name	Subject	Location
1	Adam	Biology	Noida
1	Adam	Maths	Noida
2	Adam	Maths	Noida
3	Alex	Maths	Pune
4	Stuart	Physics	Bhopal

Second Normal Form

- Remove Partial Dependencies.

- **Functional Dependency**

The value of one attribute in a table is determined entirely by the value of another.

- **Partial Dependency**

A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).

- Create separate table with the functionally dependent data and the part of the key on which it depends. Tables created at this step will usually contain descriptions of resources.

Second Normal Form

New Student Table following 2NF will be :

ID	Name	Location
1	Adam	Noida
2	Adam	Noida
3	Alex	Pune
4	Stephen	Bhopal

New Subject Table introduced for 2NF will be :

Name	Subject
Adam	Biology
Adam	Maths
Alex	Maths
Stephen	Physics

Third Normal Form

- Remove transitive dependencies.

- **Transitive Dependency**

A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.

- Create a separate table containing the attribute and the fields that are functionally dependent on it. Tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

Third Normal Form

New Student ,Subject & Linker Table following 3NF will be :

ID	S_Id
1	1
1	2
2	2
3	3
4	3

ID	Name	Location
1	Adam	Noida
2	Adam	Noida
3	Alex	Pune
4	Stuart	Bhopal

S_Id	Subject
1	Biology
2	Maths
3	Physics

Boyce-Codd Normal Form - BCNF

- A relational schema R is considered to be in BCNF if, for every one of its dependencies $X \rightarrow Y$, one of the following conditions holds true:
 - $X \rightarrow Y$ is a trivial functional dependency (i.e., Y is a subset of X)
 - X is a super key for schema R
- Informally the Boyce-Codd normal form is expressed as

"Each attribute must represent a fact about the key, the whole key, and nothing but the key."

Boyce-Codd Normal Form - BCNF

- Let's take a look at this table, with some typical data. The table is not in BCNF.

Author	Nationality	Book title	Genre	Pages
William Shakespeare	English	The Comedy of Errors	Comedy	100
Markus Winand	Austrian	SQL Performance Explained	Textbook	200
Jeffrey Ullman	American	A First Course in Database Systems	Textbook	500
Jennifer Widom	American	A First Course in Database Systems	Textbook	500

Boyce-Codd Normal Form - BCNF

- The nontrivial functional dependencies in the table are:
 - author \rightarrow nationality
 - book title \rightarrow genre, number of pages
- We can easily see that the only key is the set {author, book title}
- The same data can be stored in a BCNF schema. However, this time we would need three tables.

Author Table

Author	Nationality
William Shakespeare	English
Markus Winand	Austrian
Jeffrey Ullman	American
Jennifer Widom	American

Boyce-Codd Normal Form - BCNF

Author Book Table

Author	Book title
William Shakespeare	The Comedy of Errors
Markus Winand	SQL Performance Explained
Jeffrey Ullman	A First Course in Database Systems
Jennifer Widom	A First Course in Database Systems

Books Table

Book title	Genre	Number of pages
The Comedy of Errors	Comedy	100
SQL Performance Explained	Textbook	200
A First Course in Database Systems	Textbook	500

Boyce-Codd Normal Form - BCNF

- The functional dependencies for this schema are the same as before:
 - $\text{author} \rightarrow \text{nationality}$
 - $\text{book title} \rightarrow \text{genre, number of pages}$
- The key of the first table is {author}.
- The key of the second table is {book title}
- The key of the third table is {author, book title}.
- There are no functional dependencies violating the BCNF rules, so the schema is in Boyce-Codd normal form.

Structured Query Language

Structured Query Language, is a database computer language designed for managing data in relational database management systems (RDBMS)

Data Definition Language (DDL):

- Create
- Rename
- Truncate
- Alter
- Drop

Data Manipulation Language (DML):

- Insert
- Delete
- Update

Data Control Language (DCL):

- Grant
- Revoke

Transaction Control Language (TCL):

- Commit
- Rollback

SQL Datatypes

Data types	Description
Char(size)	Fixed length character data
Varchar2(size)	Variable length character data
Number Number(p , s) Number(p)	Variable length numeric data
Date	Date and time values
LOB(size)	Sound files , video clips storage

SQL(Cont...)

Data Definition Language:

Creating Tables:

```
mysql> CREATE TABLE dept  
      > (deptno NUMBER(2),  
      >  dname VARCHAR(14),  
      >  loc VARCHAR(13));  
Query OK, 0 rows affected (0.02 sec)
```

- Confirm table creation.

```
mysql> DESCRIBE dept
```

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR(14)
LOC		VARCHAR(13)

SQL- DDL (Cont...)

The ALTER TABLE Statement:

- Use the ALTER TABLE statement to:
 - Modify an existing column
 - Add a new column

You can change a column's data type, size,.

```
mysql> ALTER TABLE dept MODIFY COLUMN dname VARCHAR(15);  
Table altered.
```


SQL- DDL (Cont...)

Adding a Column:

```
mysql> ALTER TABLE dept ADD COLUMN job VARCHAR (9);  
Query OK, 0 rows affected (0.02 sec)
```

The new column becomes the last column.

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
-----	-----	-----	-----	---
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	

4 rows in set (0.00 sec)

SQL- DDL (Cont...)

Dropping a Table:

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- You cannot roll back this statement.

```
mysql> DROP TABLE dept;  
Query OK, 0 rows affected (0.02 sec)
```

SQL- DDL (Cont...)

Changing the Name of an Object:

To change the name of a table execute the RENAME statement.

```
mysql> RENAME table dept TO department;  
Query OK, 0 rows affected (0.02 sec)
```

SQL- DDL (Cont...)

Truncating a Table:

- The TRUNCATE TABLE statement:
 - Removes all rows from a table.
 - Releases the storage space used by that table.
- Cannot roll back row removal when using TRUNCATE.
- Alternatively, remove rows by using the DELETE statement.

```
mysql> TRUNCATE TABLE department;  
Query OK, 0 rows affected (0.02 sec)
```

SQL- Data Manipulation Language

Inserting New Rows:

- Insert a new row into a table containing values for each column.
- List values in the default order of the columns in the table.
- Enclose character and date values within single quotation marks.

```
mysql> INSERT INTO dept(deptno,dname,loc)  
      > VALUES (50,'DEVELOPMENT', 'DETROIT');
```

```
Query OK, 1 row affected (0.00 sec)
```


SQL- DML (Cont...)

Updating Rows in a Table:

- Specific row or rows are modified when you specify the WHERE clause.
- All rows in the table are modified if you omit the WHERE clause.

```
mysql> UPDATE emp  
      > SET deptno = 20  
      > WHERE empno = 7782;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> UPDATE employee  
      > SET deptno = 20;  
Query OK, 14 rows affected (0.00 sec)
```

SQL- DML (Cont...)

Deleting Rows from a Table:

As an example

- Specific row or rows are deleted when you specify the WHERE clause.
- All rows in the table are deleted if you omit the WHERE clause.

```
mysql> DELETE FROM department  
      > WHERE dname = 'DEVELOPMENT';  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> DELETE FROM department;  
Query OK, 4 rows affected (0.00 sec)
```

SQL - Data Control Language

As an example

Grant query privileges on the EMP table by user A.

```
mysql> GRANT select  
      > ON emp  
      > TO sue, rich;  
Query OK, 0 rows affected (0.00 sec)
```

- Grant privileges to update specific columns to users and roles by user A.

```
mysql> GRANT update (dname, loc)  
      > ON emp.dept  
      > TO scott, manager;  
Query OK, 0 rows affected (0.00 sec)
```

SQL – DCL (Cont...)

Revoking Privileges from the user:

As an example

User A can revoke the SELECT and INSERT privileges given to user Scott on the DEPT table.

```
mysql> REVOKE select, insert
      2  ON emp
      3  FROM rich;
Query OK, 0 rows affected (0.00 sec)
```

SQL-Transaction Control Language

Definitions :

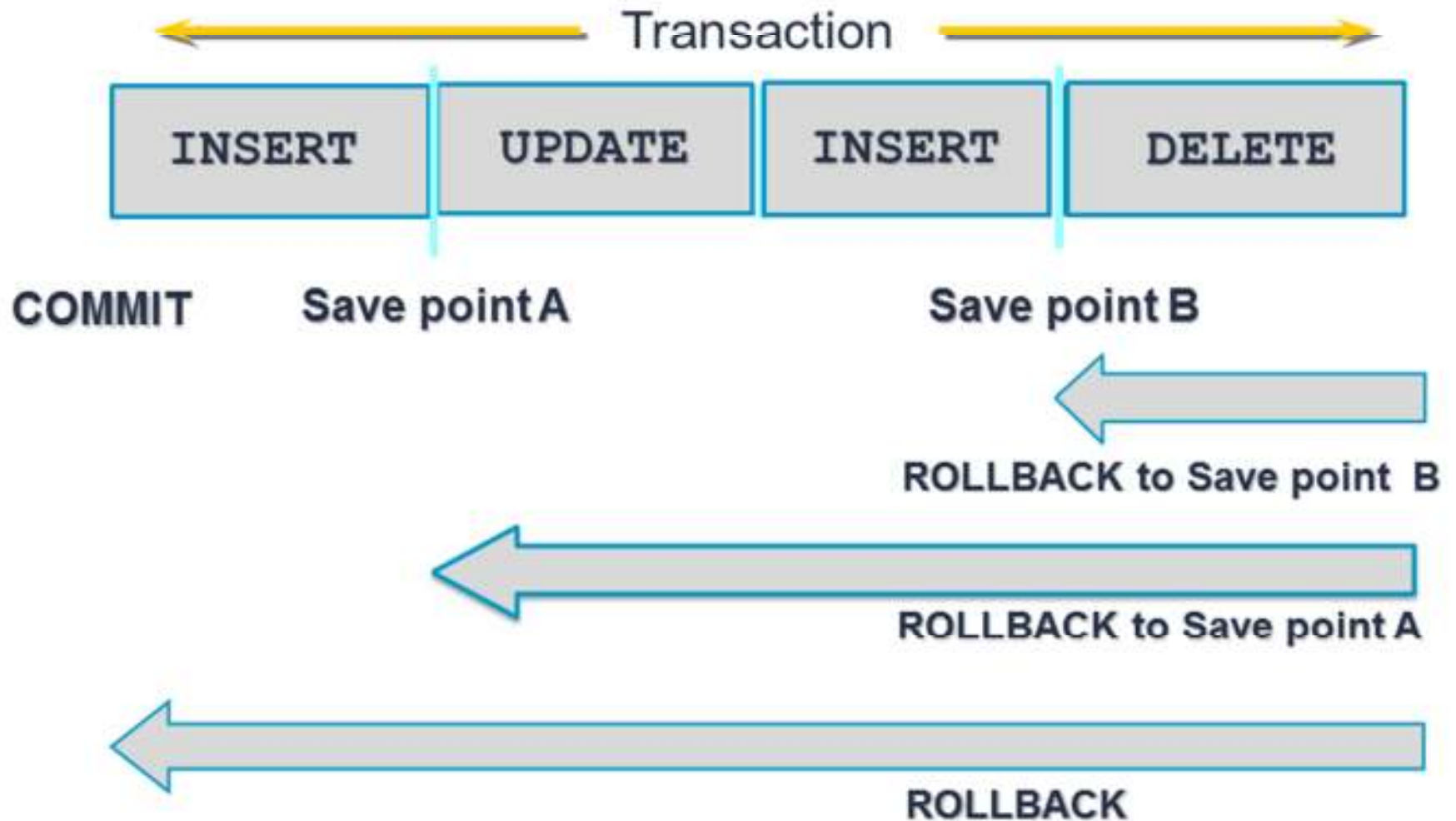
- Commit
- Rollback
- Savepoint

Advantages :

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

SQL-TCL(Cont...)

Controlling Transactions:



SQL-TCL(Cont...)

Committing Data:

Make the changes.

```
mysql> UPDATE emp  
      > SET    deptno = 10  
      > WHERE empno = 7782;  
Query OK, 1 row affected (0.00 sec)
```

Commit the changes.

```
mysql> COMMIT;  
Query OK, 0 rows affected (0.00 sec)
```

SQL-TCL(Cont...)

State of the Data After ROLLBACK:

- Discard all pending changes by using the ROLLBACK statement
 - Data changes are undone.
 - Previous state of the data is restored.
 - Locks on the affected rows are released.

```
mysql> DELETE FROM employee;  
Query OK, 14 rows affected (0.00 sec)  
mysql> ROLLBACK;  
Query OK, 1 row affected (0.00 sec)
```

SQL-TCL(Cont...)

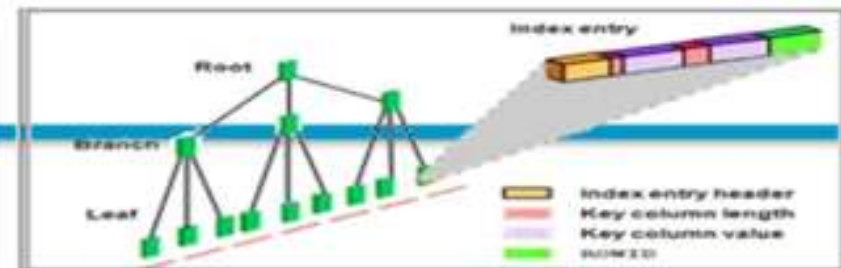
Rolling Back Changes to a Marker:

- Create a marker within a current transaction by using the **SAVEPOINT** statement.
- Roll back to that marker by using the **ROLLBACK TO SAVEPOINT** statement.

```
mysql> UPDATE...  
mysql> SAVEPOINT update_done;  
Query OK, 0 rows affected (0.00 sec)  
mysql> INSERT...  
mysql> ROLLBACK TO update_done;  
Query OK, 0 rows affected (0.00 sec)
```

SQL-TCL

An index helps speed up retrieval.



One of the main reasons for the success of relational databases is that they have efficient indexing systems.

An index typically consists of the search key (one or more attributes) and the row ID(s) of record(s) which match this key

An index on a file speeds up selections on the **search key fields** specified for that index

Any subset of the fields of a relation can be the search key for an index on the relation

A **primary index** allows efficient searching on the primary key of a relation

A **secondary index** allows efficient searching on other attributes which are often used in queries

SQL

CREATE INDEX "INDEX_NAME" ON "TABLE_NAME" (COLUMN_NAME)

- Let's assume that we have the following table,

- **TABLE***Customer*
(First_Name char(50),
Last_Name char(50),
Address char(50),
City char(50),
Country char(25),
Birth_Date date)

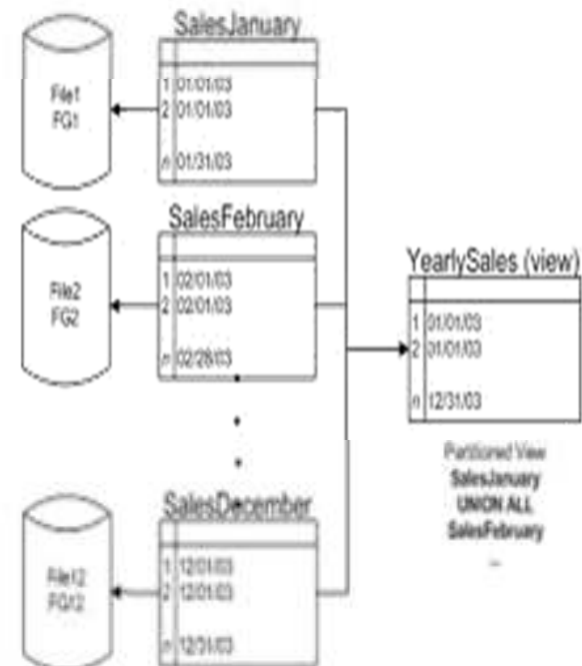
and we want to create an index on the column Last_Name, we would type in,

- **CREATE INDEX IDX_CUSTOMER_LAST_NAME**
on CUSTOMER (Last_Name)
- If we want to create an index on both City and Country, we would type in,
- **CREATE INDEX IDX_CUSTOMER_LOCATION**
on CUSTOMER (City, Country)

SQL

SQL Views

- A VIEW is a virtual table, through which a selective portion of the data from one or more tables can be seen. Views do not contain data of their own. They are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database.



SQL

✓ The Syntax to create a sql view is

- `CREATE VIEW view_name`
`AS`
`SELECT column_list`
`FROM table_name [WHERE condition];`
- **view_name** is the name of the VIEW.
- The SELECT statement is used to define the columns and rows that you want to display in the view.
- **For Example:** to create a view on the product table the sql query would be like
- `CREATE VIEW view_product`
`AS`
`SELECT product_id, product_name`
`FROM product;`

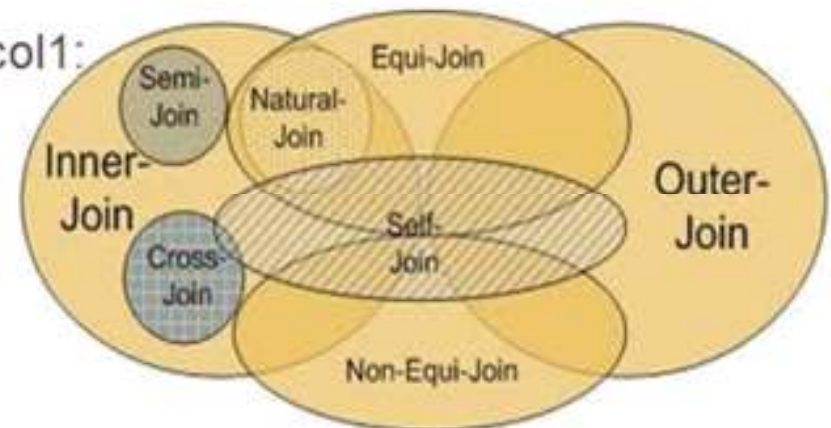
SQL:

SQL Joins

- SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.

- The Syntax for joining two tables is:**

- SELECT col1, col2, col3...
FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1:



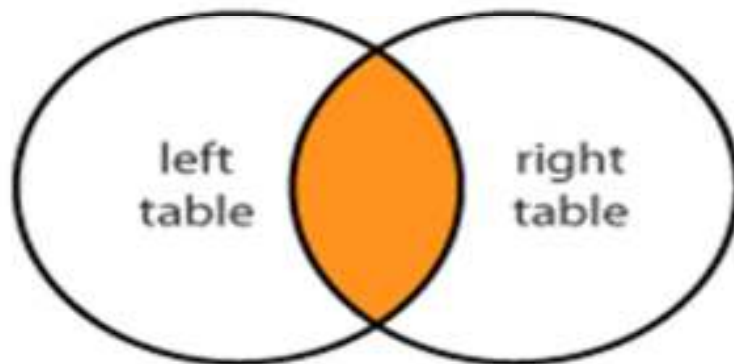
SQL:

JOIN TYPES:

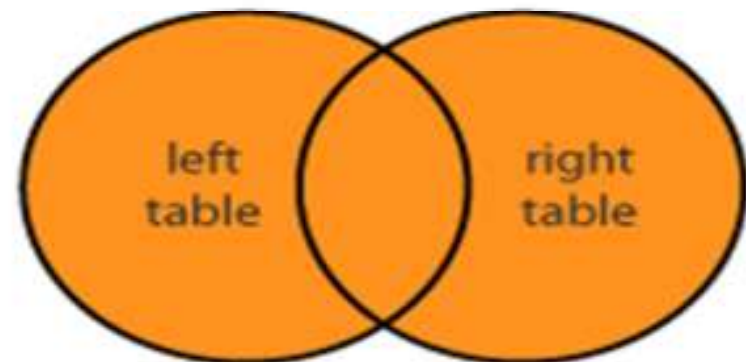
- **INNER JOIN:** Returns all rows when there is at least one match in BOTH tables
- **LEFT JOIN:** Return all rows from the left table, and the matched rows from the right table
- **RIGHT JOIN:** Return all rows from the right table, and the matched rows from the left table
- **FULL JOIN:** Return all rows when there is a match in ONE of the tables

SQL:

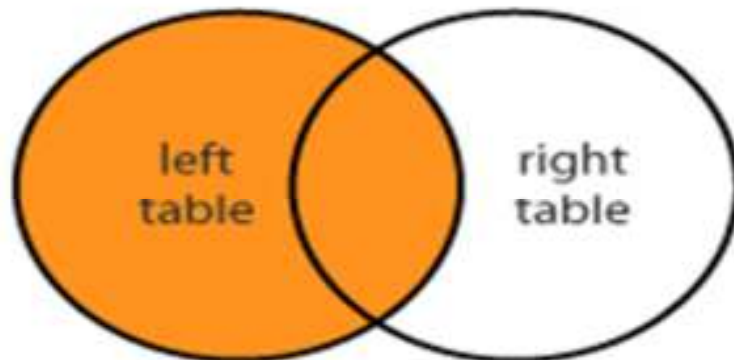
INNER JOIN



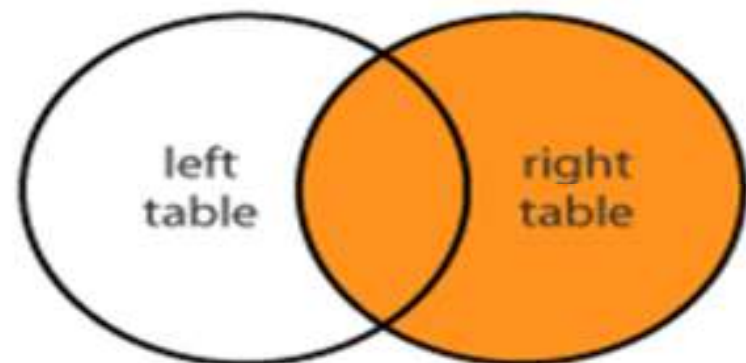
FULL JOIN



LEFT JOIN



RIGHT JOIN



SQL:

INNER JOIN SYNTAX:

```
SELECT column_name(s)
  FROM table1
  INNER JOIN table2
  ON table1.column_name=table2.column_name;
```

LEFT JOIN SYNTAX:

```
SELECT column_name(s)
  FROM table1
  LEFT JOIN table2
  ON table1.column_name=table2.column_name
```

SQL:

RIGHT JOIN SYNTAX:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

FULL JOIN SYNTAX:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

SQL:

Product

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Vediocoon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

Order_items

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS

EXAMPLE:

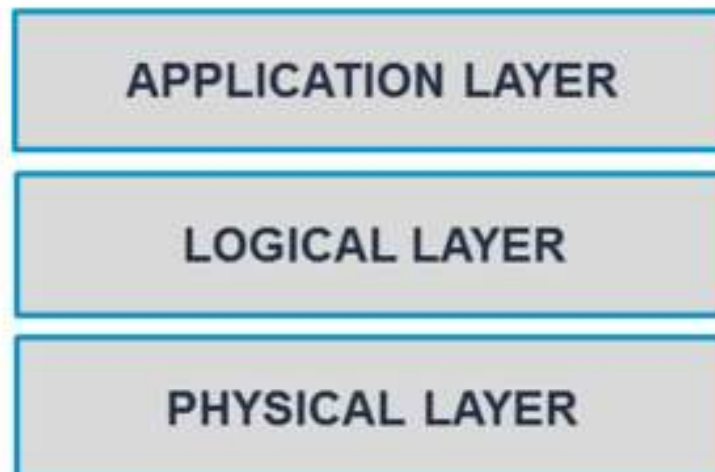
```
SELECT order_id, product_name, unit_price, supplier_name, total_units FROM product, order_items  
WHERE order_items.product_id = product.product_id;
```

(or)

```
SELECT o.order_id, p.product_name, p.unit_price, p.supplier_name, o.total_units FROM product p,  
order_items o WHERE o.product_id = p.product_id;
```

General RDBMS Architecture

It was found in all of the consulted resources that all database systems, can be viewed as a three layered architecture at the highest level of abstraction. It has three main components as outlined in the following diagram:



Database Layered Architecture

General RDBMS Architecture (Cont...)

Application layer:

- The application layer represents the interface for all the users of the system; it essentially provides the means by which the outside world can interact with the database server. In general, it has been found that users can be categorized into four main groups:
- **Sophisticated users** interact with the system without using any form of application; they form their requests directly with the use of a database query language.
- **Specialized users** are application programmers who write specialized database applications that do not fit into the traditional data-processing framework.
- **Naive users** are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been previously written.
- **Database Administrators** have complete control over the entire database system. They have a wide variety of responsibilities, including schema definition, the granting of access authorization, as well as the specification of integrity constraints.

General RDBMS Architecture (Cont...)

Logical layer:

- The core functionality of the RDBMS is represented in the logical layer of the architecture
- logical architecture is a more detailed structure defines what has to be done to support the user services.
- It defines the processes that perform functions and the information or data flows that are shared between these processes.
- They do include any business services, application names and details, and other relevant information for development purposes.

General RDBMS Architecture (Cont...)

Physical layer:

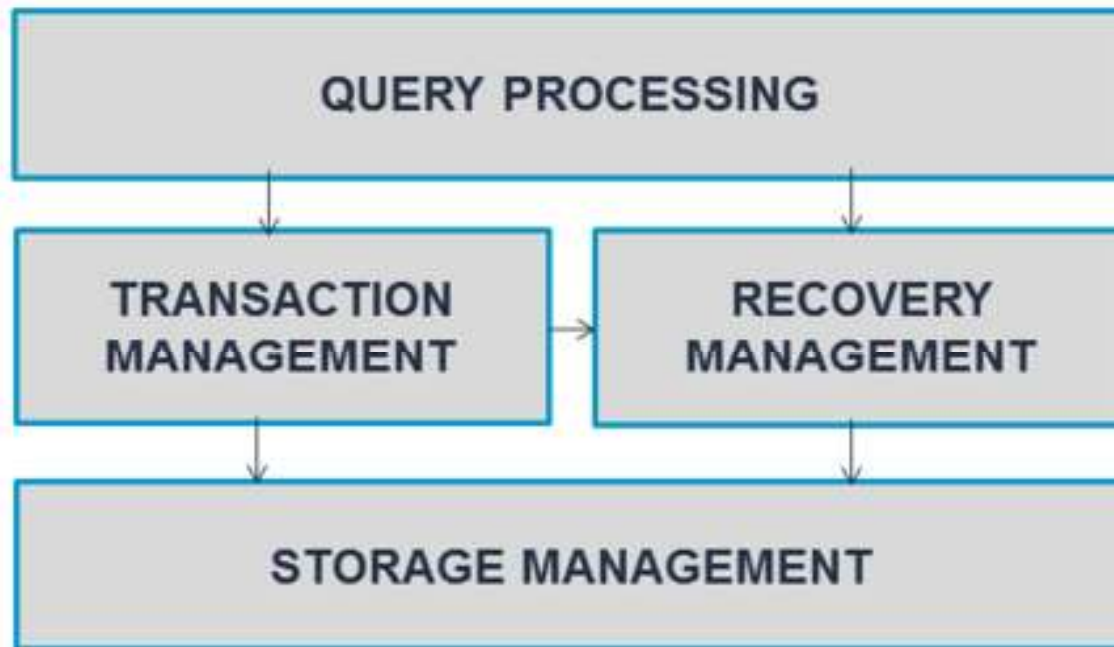
The RDBMS is responsible for the storage of a variety of information, which is kept in secondary storage and accessed via the storage manager.

The main types of data kept in the system are:

- **Data files**, which store the user data in the database
- **Data dictionary**, which stores metadata about the structure of the database
- **Indices**, which provide fast access to data items that hold particular values
- **Statistical Data**, which store statistical information about the data in the database; it is used by the query processor to select efficient ways to execute a query.
- **Log Information**, used to keep track of executed queries such that the recovery manager can use the information to successfully recover the database in the case of a system crash.

General RDBMS Architecture (Cont...)

Logical Modules of RDBMS:



General High Level RDBMS Logical Modules

Top 5 databases

- MySQL



- PostgreSQL



- Oracle



- SQLite



- Microsoft SQL Server

Feature Comparison

Feature	Oracle	MySQL	SQL Server
Interface	GUI, SQL	SQL	GUI, SQL, Various
Language support	Many, including C, C#, C++, Java, Ruby, and Objective C	Many, including C, C#, C++, Java, Ruby, and Objective C	Java, Ruby, Python, VB, .Net, and PHP
Operating System	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris	Windows
Licensing	Proprietary	Open source	Proprietary

Basic Terminologies to be known for Administration

- Database Instance
- Datafile
- Logfile
- Tablespace
- Backup and Recovery



People matter, results count.

www.capgemini.com



The information contained in this presentation is proprietary.
Rightshore® is a trademark belonging to Capgemini.
© 2015 Capgemini. All rights reserved.