Data : Consists of group of characters ,facts, text, graphics, images, sounds etc. that have meaning in the user's environment.

Information : Data that has been processed in such a way that it increases knowledge of user.

Database: An organized collection of logically related data
-organized : refers to the way the data is organized
Collection : refers to a group of data

Database Management systems : consists of database and set of programs (software) to access the data.

RDBMS (relational database management system) applications store data in a tabular form and relation exist among the tables. while DBMS applications stores data in tables but there will be no **"**relation" between the tables, like in a RDBMS

Database Applications
Banking: all transactions
Airlines: reservations, schedules
Universities: registration, grades
Sales: customers, products, purchases
Online retailers: order tracking, customized recommendations
Manufacturing: production, inventory, orders, supply chain
Human resources: employee records, salaries, tax deductions
Databases touch all aspects of our lives

Disadvantages of file system or drawbacks of using file systems/conventional/ traditional file systems to store data:

- Data redundancy and inconsistency
- Multiple file formats, duplication of information in different files
- Difficulty in accessing data
- Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
- Integrity constraints (e.g. account balance > 0) become
  "buried" in program code rather than being stated explicitly
- Hard to add new constraints or change existing ones
- Atomicity of updates
- Failures may leave database in an inconsistent state with partial updates carried out
  Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  Concurrent accessed needed for performance
  Uncontrolled concurrent accesses can lead to inconsistencies
  Example: Two people reading a balance and updating it at the same time
- Security problems
- Hard to provide user access to some, but not all, data
- Database systems offer solutions to all the above problems

Advantages of database management systems

- Program-Data independence
- Centralized control of data(data ownership)
- Data redundancy eliminated
- Data can be shared

- Improved data consistency
- Improved data integrity
- Improved data accessibility and response
- Reduced program maintenance
- Improved security for data
- Multiple access of data
- Protection from failure

Disadvantages of DBMS

- Higher cost -More sophisticated hardware and software, higher operating cost
- Greater complexity
- Backup/recovery procedures complex
- DBMS occupy more memory space
- Application need to process large amount of data

Database Administrator[DBA] responsibilities

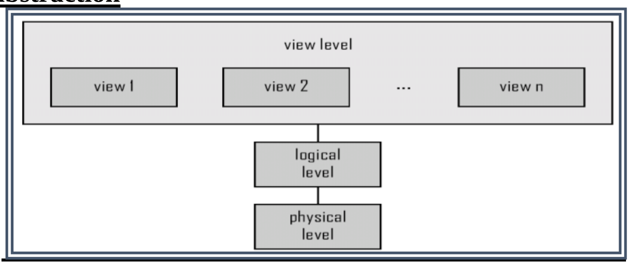
- Creates and maintains database
- Create schema, subschema of database
- Define storage structure and access methods
- Responsible for schema and physical organization modifications
- Grants authorization for data access. Regulates who can access which part of database
- Responsible for
  - -database backup
  - -database recovery
  - -disk space management
  - -performance monitoring
- Responsible for procedures for database integrity and security

Levels of Abstraction

Views of Data

Major purpose of database system is to provide users with an abstract view of data
The system hide certain details of how the data are stored and maintained
Abstraction: Hiding un necessary details and showing only essentials.
Developers hide the complexity from users through several levels of abstraction
Dbms hides from user details of how data stored in database and how data is retrieved

3 levels of abstraction



Physical level

- Lowest level of abstraction
- About **how** data is stored
- Low level data structures , data storage
- Handled by DBMS

Logical level

- Describe what data is stored and **what** relationship exist among the data
- Logical data structure

- Independent of any particular technology
- Handled by DBA

View level
- Highest level of abstraction
- Describes only part of database
- Independent of DBMS technology
- Required for the user to do his work
- Simplifies user's interaction with database
- Defined in programming language terms

Instance & schema

Instance:
- The collection data stored in a database at a particular movement is called data base.
- Database instance change over time as data is inserted/ deleted / modified

Schema
- The overall design of database is called the data schema
- Declaration of variable –Schema
- Value of the variable - Instance

The database have several schemas
- physical schema: describe database design at physical level
- Logical schema: describe the database design at logical level

Subschema: A data base may have several schemas at view level sometimes called subschema

Data independencePhysical Data Independence – The ability to modify a schema definition in one level with out affecting a schema definition in next higher level is called data independence.

- Physical data independence: The ability to modify the physical schema with out causing application program
- Logical data independence: The ability to modify the logical schema without causing application programs

In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others

Data models

- The structure of a database is data model
- A collection of conceptual tools describing the data
- A data model is the underlying structure of database
- Data model is a collection of conceptual tools for describing data, data relationship, data semantics and data constraints
- Various data models that have been processed fall into three different groups

Two different groups
- Object –based logical model
- Record based logical model

Object based logical model
- The models are describing at logical and view levels
  - Entity- relationship model
  - Object oriented model
- While object –based logical models describes data in terms of entities and relationship among entities
- record based logical model are concerned with how these entities and relationships have to be represented in the database system

Record based logical model
- Hierarchical model
- Network model
- Relational model

Hierarchical Model
- Data represented by a collection of records and relationships among data represented by links
- Records are organized as a collection of trees
- Some data redundancy can be avoided
- Complex data structures are need
- Can not be possible to represent many-to-many relationship
- Data represented in parent –child relationship
- One –to many relationship can easily represented
- Eg: IMS (information management system)
- RAMIS- rapid access management information system

Network model
- Data represented by collection of records and relationship among the data represented by links
- Records are organized into a collection of graphs
- Data redundancy is minimum
- High degree of complexity
- Represented many –to – many & one- many relation ship
- Relation ship among the data are represented by links
- It is represented by graphs & more complex
- Eg: Integrated database management system
- ADABAS- Adaptable database system

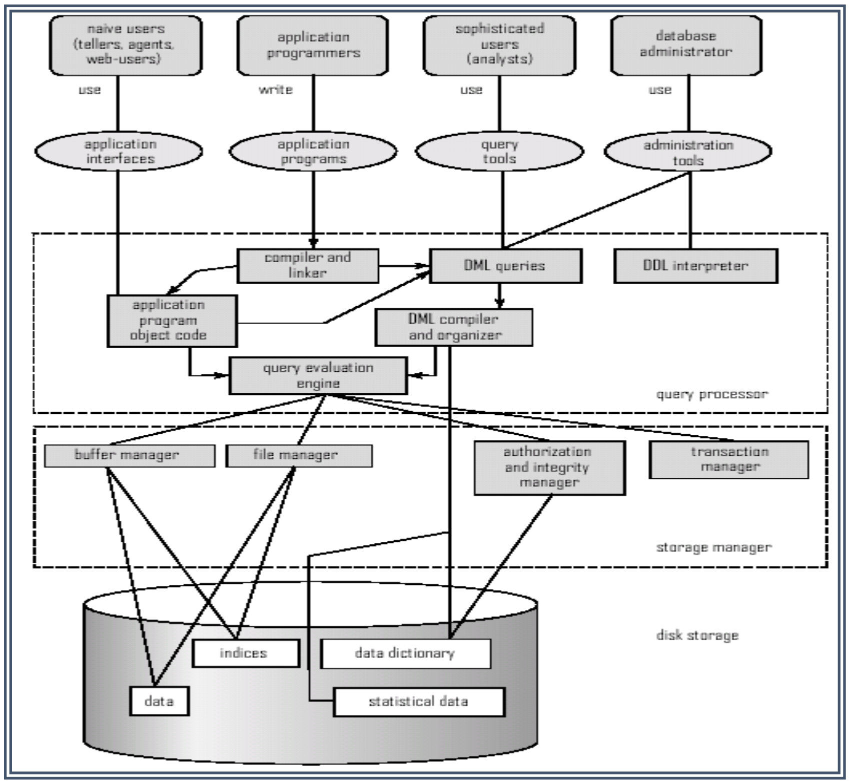
Relational models
- Relational model proposed by Dr.E.F.Codd in 1976
- The relational model uses a collection of tables.
- Data represented in the form of tables(relations)
- Tables are logically equivalent to files and consists of rows
- Rows called **tuples**
- Rows have fixed number of columns called **attributes**

Relational database
- Any database for which the logical organization is based on relational data models
- **RDBMS:** A DBMS that manages the relational database
- An RDBMS is a type of DBMS that stores data in the form of related tables

Some popular RDBMS packages

• RDBMS	Company/Corporation
• Oracle	oracle corporation
• Sybase	sybase corporation
• Informix	informix software Inc
• SQL server	Microsoft
• DB2	IBM
• Ingress II	Computer associates international inc

System structure



Data Definition Language (DDL) Commands
COMMAND : CREATE
This command is used to create the structure of database

Syntax :
Create table <table name > (column_name1 type(size),   Column_name2 type(size), …);
Example

COMMAND :ALTER
This command is used to add a column or to increase the size of the column

Syntax:
alter table <table name> add/modify (column data type(size));
Example

COMMAND: DROP
This command is used to delete the table from database.

Syntax:
drop table <tablename>; Example

COMMAND :TRUNCATE
This command is used delete the rows but structure remains

Syntax:

truncate table <table name>; example

Data manipulation commands (DML) commands:

COMMAND :INSERT

Inserting a single row into a table

Syntax:

insert into <table name> values (value list);

```
SQL>  insert  into s values(‘s3’,’sup3’,’blore’,10)
```

Inserting more than one record using a single insert commands:

INPUT:

```
SQL>  insert into emp
Values(&empno,'&ename','&job','&mgr','&hirdate',&sal,'&comm',&deptno,'&empgen');
Enter value for empno: 23
Enter value for ename: aaa
Enter value for job: clerk
Enter value for mgr: 45
Enter value for hirdate: 12-dec-89
Enter value for sal: 5000
Enter value for comm: 20
Enter value for deptno: 30
Enter value for empgen: m
old   1:  insert into emp
values(&empno,'&ename','&job','&mgr','&hirdate',&sal,'&comm',&deptno,'&emp
new   1:  insert into emp values(23,'aaa','clerk','45','12-dec-89',5000,'20',30,'m')

1 row created.

SQL> /
Enter value for empno: 24
Enter value for ename: bbb
Enter value for job: manager
Enter value for mgr: 46
Enter value for hirdate: 13-jan-89
Enter value for sal: 6000
Enter value for comm: 60
Enter value for deptno: 20
Enter value for empgen: female
old   1:  insert into emp
values(&empno,'&ename','&job','&mgr','&hirdate',&sal,'&comm',&deptno,'&emp
new   1:  insert into emp values(24,'bbb','manager','46','13-jan-89',6000,'60',20,'female')

1 row created.
```

OUTPUT:
emp table

```
EMPNO   ENAME       JOB            MGR    HIREDATE         SAL     COMM.     DEPTNO
-----------  ------------  ----------------  --------  ----------------------        -------  ------------    --------------
7369        SMITH       CLERK   7902     17 – DEC – 80            800                 20
7499        ALLEN       SALESMAN       7698   20 – FEB – 81    1600      300       30
7521        WARD        SALESMAN       7698   22 – FEB – 81    1250       500      30
7566        JONES       MANAGER        7839   02 – APR – 81     2975                20
7654        MARTIN      SALESMAN       7698   28 – SEP – 81    1250     1400       30
7698        BLAKE       MANAGER        7839   01 – MAY – 81     2850              30
7782        CLARK       MANAGER        7839   09 – JUN – 81      2450              10
7788        SCOTT       ANALYST        7566   19 – APR – 87      3000              20
7839        KING        PRESIDENT             17 – NOV – 81       5000             10
7844        TURNER      SALESMAN       7698   08 – SEP – 81      1500     0        30
7876        ADAMS       CLERK          7788   23 - MAY – 87      1100              20
7900        JAMES       CLERK          7698    03 – DEC – 81       950             30
7902        FORD        ANALYST       7566    03 – DEC – 81      3000              20
7934        MILLER      CLERK        7782     23 – JAN – 82      1300             10
```

Skipping the fields while inserting
Syntax:
insert into <tablename>(coln names to which data to be inserted)> values (list of values);
Other way is to give null while passing the values

COMMAND : SELECT
To retrieve all rows from a table

Syntax:
SQL>Select * from table name;

INPUT:
SQL>Select * from emp;

OUTPUT:

```
EMPNO   ENAME       JOB            MGR    HIREDATE         SAL     COMM.     DEPTNO
-----------  ------------  ----------------  --------  ----------------------        -------  ------------    --------------
7369        SMITH       CLERK   7902     17 – DEC – 80              800               20
7499        ALLEN       SALESMAN       7698   20 – FEB – 81     1600     300       30
7521        WARD        SALESMAN       7698   22 – FEB – 81      1250      500     30
7566        JONES       MANAGER        7839   02 – APR – 81       2975              20
7654        MARTIN      SALESMAN       7698   28 – SEP – 81      1250      1400    30
7698        BLAKE       MANAGER        7839   01 – MAY – 81      2850              30
7782        CLARK       MANAGER        7839   09 – JUN – 81       2450              10
7788        SCOTT       ANALYST        7566   19 – APR – 87      3000               20
7839        KING        PRESIDENT             17 – NOV – 81       5000              10
7844        TURNER      SALESMAN       7698   08 – SEP – 81       1500       0        30
7876        ADAMS       CLERK   7788             23 - MAY – 87       1100             20
```

To retrieve specific columns from a table
Syntax:
Select column_name1, …..,column_namen from table name;

INPUT:
```
SQL> select ename, deptno, sal from emp;
```

OUTPUT:
```
ENAME      DEPTNO    SAL
---------- ------- -------     ------
adams         20           1100
allen          30           1600
blake         30            2850
clark         10             2450
ford          20            3000
james        30            950
jones        20           2975
```

KEYWORD : DISTINCT
Elimination of duplicates from the select clause: It prevents retrieving the duplicated values. Distinct keyword is to be used.

Syntax:
SQL>Select DISTINCT col1, col2 from table name;

INPUT:
```
SQL>select distinct job from emp;
```

OUTPUT:
```
JOB
------
CLERK
SALESMAN
MANAGER
ANALYST
PRESIDENT
```

To select specific rows from a table we include 'where' clause in the select command. It can appear only after the 'from' clause
Syntax:
Select column_name1, …..,column_name from table name where condition;

INPUT:
```
SQL> select ename, job from emp where job = 'manager';
```
OUTPUT:
```
ENAME     JOB
```

```
---------- ---------
jones     manager
blake     manager
clark     manager
```

COMMAND: UPDATE

To modify column or group of columns

Syntax:

update <tablename> set field=values where condition;

INPUT:

```
SQL>update std set branch='mca' where sno=03;
1 row updated.
sql> select * from std;
```

OUTPUT:

```
 SNO SNAME      BRANCH
---------- ---------- ----------
     1       ram       B.Tech
     2       ravi      B.Tech
     3       krish     mca
     4       rfde      B.Tech
```

COMMAND :DELETE

The command is used to delete the rows in a table

Syntax:

Delete from <table_name> where conditions;

INPUT:

```
SQL> delete from  emp where  empno=7369;
```

<u>DATA CONTROL LANGUAGE (DCL) COMMANDS IN RDBMS</u>

COMMAND : COMMIT

Syntax:

commit [work] [comment '*comment_text*']
commit [work] [force '*force_text*' [,*int*]]

INPUT:

```
SQL>commit;
```

OUTPUT:

```
Commit complete;
```

COMMAND : ROLLBACK

The ROLLBACK command is the transactional control command to undo the transactions that have not already been committed to the database. The ROLLBACK command can be issued to undo the changes since the last COMMIT or ROLLBACK.

Syntax:
ROLLBACK [WORK] [TO [SAVEPOINT]'savepoint_text_identifier'];
ROLLBACK [WORK] [FORCE 'force_text'];

INPUT:
SQL>rollback;

OUTPUT:
Rollback complete;

INTEGRITY CONSTRAINTS

data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle. Integrity constraints are used to ensure accuracy and consistency of data in a relational database.

TYPES OF INTEGRITY CONSTRAINTS

Various types of integrity constraints are-

- Domain Integrity
- Entity Integrity Constraint
- Key Constraints
- Referential Integrity Constraint

Domain Integrity-

- Domain integrity means the definition of a valid **set of values** for an attribute.
- You define data type, length or size, It also allows null value. It is the value unique or not for an attribute ,the default value, the range (values in between) and/or specific values for the attribute

Entity Integrity Constraint-

- This rule states that in any database relation value of attribute of a primary key can't be null.

Key Constraints

- An attribute or set of attributes that uniquely identifies rows in a table is called as Key
- Constraints can be defined in **two** ways
- 1.column-level
  The constraints can be specified immediately after the column definition. This is called column-level definition.

- 2. Table-level
  The constraints can be specified after all the columns are defined. This is called table-level definition

Not Null Constraint

- By default, all columns in a table allow nulls.
- **Null** means the absence of a value.
- A NOT NULL constraint requires a column of a table contain no null values
- **Syntax :**

  [CONSTRAINT constraint name] NOT NULL

  Example:

```
CREATE  TABLE  employee
( id number(5),
name char(20) CONSTRAINT nm_nn NOT NULL,
dept char(10),
age number(2),
salary number(10),
location char(10)
);
```

Unique Key

- A UNIQUE key integrity constraint requires that every value in a column or set of columns (key) be unique--that is, no two rows of a table have duplicate values in a specified column or set of columns
- A column(s) can have a null value but the values cannot be duplicated

Column level:

Syntax

[CONSTRAINT constraint_name] UNIQUE

Table level:

Syntax

[CONSTRAINT constraint_name] UNIQUE(column_name)

Example**:**

```
CREATE TABLE employee
( id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10) UNIQUE
);
```

or

```
CREATE TABLE employee
( id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10) CONSTRAINT loc_un UNIQUE
);
```

table level:

```
CREATE TABLE employee
( id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT loc_un UNIQUE(location)
);
```

Primary key

- An attribute or a set of attributes which uniquely identify an entity
- No two rows of a table have duplicate values in the specified column or set of columns.
- The primary key columns do not allow nulls. That is, a value must exist for the primary key columns in each row.
- Each table have at most only one primary key

- **Syntax**:
- [CONSTRAINT constraint_name] PRIMARY KEY (column_name1,column_name2,..)
- **Note**: The syntax within the bracket i.e. [CONSTRAINT constraint_name] is optional
- **column level**

```
        CREATE TABLE employee
( id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10)

  );
```

```
        CREATE TABLE employee
( id number(5) CONSTRAINT emp_id_pk PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10)
);
```

Table level:

```
CREATE TABLE employee
( id number(5),
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY (id)
);
```

-
  ```
  CREATE TABLE employee
  ( id number(5), NOT NULL,
  name char(20),
  dept char(10),
  age number(2),
  salary number(10),
  ```

```
location char(10),
ALTER TABLE employee ADD CONSTRAINT PK_EMPLOYEE_ID PRIMARY KEY (id)
);
```

- A primary key can consist of one or more fields on a table.

composite primary key:

- When multiple fields are used as a primary key, they are called a composite primary key.

Foreign key or Referential Integrity

- Foreign key is the key i.e. attribute which refers to another table primary key.
- Reference key is the primary key of table referred by another table.
- Foreign keys help you create logical ties within the information in a database

Foreign Key at column level:

-
  ```
  CREATE TABLE product
  ( product_id number(5) CONSTRAINT pd_id_pk PRIMARY KEY,
  product_name char(20),
  supplier_name char(20),
  unit_price number(10)
  );
  ```

-
  ```
  CREATE TABLE order_items
  ( order_id number(5) CONSTRAINT od_id_pk PRIMARY KEY,
  product_id number(5) CONSTRAINT pd_id_fk REFERENCES, product(product_id),
  product_name char(20),
  supplier_name char(20),
  unit_price number(10)
  );
  ```

Foreign Key at table level:

```
CREATE TABLE order_items
( order_id number(5) ,
product_id number(5),
product_name char(20),
supplier_name char(20),
unit_price number(10)
CONSTRAINT od_id_pk PRIMARY KEY(order_id),
CONSTRAINT pd_id_fk FOREIGN KEY(product_id) REFERENCES product(product_id)
);
```

check constraint

- It specifies a range of values with in which the attribute should take
- It allows you to specify a condition on each row in a table
- A check constraint can be defined in either an SQL CREATE TABLE statement or an SQL ALTER TABLE statement.

Syntax:

- CREATE TABLE table_name (column1 datatype null/not null, column2 datatype null/not null, ... CONSTRAINT constraint_name CHECK (column_name condition) [DISABLE]);

Example:

- CREATE TABLE suppliers (supplier_id numeric(4), supplier_name varchar2(50), CONSTRAINT check_supplier_id CHECK (supplier_id BETWEEN 100 and 9999));

Alter command to add a constraint

- ALTER TABLE Persons ADD PRIMARY KEY (ID);
- ALTER TABLE Persons ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);

DROP Constraint

- ALTER TABLE Persons DROP CONSTRAINT PK_Person;

ALTER TABLE Persons DROP PRIMARY KEY