DATABASE CONCEPTS

INTRODUCTION

Database: A database is an organized collection of data.
Database Management System: Software used to manage databases is called Data Base Management System (DBMS). A database management system is a collection of programs that enables users to create, maintain and use a database. Some examples of DBMS are – MySQL, Oracle, DB2, IMS, IDS etc.

Benefits of using a DBMS are:

a. Redundancy can be controlled
b. Inconsistency can be avoided
c. Data can be shared
d. Security restrictions can be applied.
e. User friendly

Limitations of using DBMS are:

a. High cost
b. Security overheads

Relational Database: A database in which the data is stored in the form of relations (also called tables) is called a Relational Database. In other words a Relational Database is a collection of one or more tables.

RDBMS: A DBMS used to manage Relational Databases is called an RDBMS (Relational Data Base Management System). Some popular RDBMS software available are: Oracle, MySQL, Sybase, Ingress.

MySQL: It is an Open Source RDBMS Software. It is available free of cost.

RDBMS TERMINOLOGY

Relation/Table: A table refers to a two dimensional representation of data arranged in columns (also called fields or ***attributes***) and rows (also called records or ***tuples***).

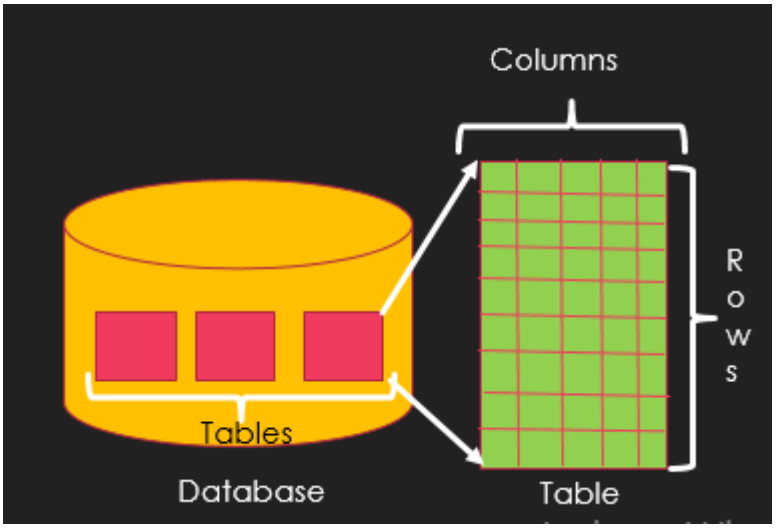

Fig Relation between database and table

For e.g. In a student's database there will be Students, Teachers and Office staff tables for storing respective data.

Degree of a table is the number of columns in the table.

Cardinality of a table is the number of rows in a table.

Domain is the data type of values in each column.

Relation Schema R is denoted by R (A_1 , A_2 , A_3..., A_n) where R is the relation name and A_1 , A_2 , A_3 ,....A_n is the list of attributes.

Relation State is the set of tuples in the relation at a point in time. A relation state r of
relation schema R (A_1, A_2, ..., A_n), denoted r(R) is a set of n-tuples r = {t_1,t_2,...., t_m}, where each n-tuple is an ordered list of values t = <v_1, v_2, ...,v_n>, where v_i is in domain of A_i or is NULL. Here n is the degree of the relation and m is the cardinality of the relation.

For example, consider a table STUDENT

columns

Admn_no	Roll_no	Name	Class
1001	1	Ankita	12
1002	2	Vaibhav	12
1003	3	Utsav	12

rows

Table: STUDENT

In the Student table:

- Cardinality = 3 (there are 3 rows in this table)
- Degree = 4 (there are 4 columns in this table)

- Domain = The domain of attributes of the student relation:
  - Admn_no: Set of 4-digit numbers
  - Roll_no: Integer<50
  - Name: Set of character strings
  - Class: integer <13
- Relation Schema – STUDENT (Admn_no, Roll_no, Name, Class)
- Relation State - {<1001, 1, Ankita,12>,<1002, 2, Vaibhav, 12> ,<1003, 3, Utsav,12> }

Relational Model Constraints

Constraints, are restrictions imposed on the values, based on certain requirements. Various types of constraints in Relational model are:

- **Domain Constraint**: It specifies that the range of values that every attribute in each tuple of the relation must satisfy. For example, the Admn_no must be a 4-digit number. Hence a value such as "11001" or "A123" violates the domain constraint as the former is not 4-digit long and the latter contains an alphabet.

- **Key Constraint**: Let us first understand the terms: superkey, key, candidate key and primary key.

  (i) **Superkey** is a set of attributes in a relation, for which all tuples in a relation have unique values. Every relation must have at least one superkey which is the combination of all attributes in a relation. Thus, for the STUDENT relation, following are some of the superkeys:
  (a) {Name, Employee_ID, Gender, Salary, Date_of_birth} - default superkey consisting of all attributes.
  (b) {Admn_no, Roll_no, Name, Class}
  (c) { Admn_no, Roll_no, Name }
  (d) { Roll_no, Name, Class }
  (e) { Admn_no, Roll_no }
  (f) { Admn_no }

  (ii) **Key** is the minimal superkey, which means it is that superkey of a relation from which if any attribute is removed then it no longer remains a superkey. For example, the superkey { Admn_no, Roll_no } is not a key as if we remove Roll_no from this combination and then what is left { Admn_no } is still a Superkey. Now { Admn_no } is a key as it is a superkey as well as no more removals are possible.

  (iii) **Candidate key**: A key as described above is called candidate key of the relation. For
  example, the STUDENT relation has two candidate keys Admn_no and Roll_no.

  (iv) **Primary Key**: The group of one or more columns used to uniquely identify each row of a relation is called its Primary Key. One of the candidate keys may be designated as Primary key.

If a relation has many candidate keys it is preferable to choose that one as primary key which has least number of attributes. For example in the relation: STUDENT, Admn_no is the primary key.

- **Null Value Constraint**: This constraint when applied to an attribute makes sure that the attribute cannot have null values. For example, if every STUDENT must have a valid name then the Name attribute is constrained to be NOT NULL.

- **Entity Integrity Constraint**: This constraint specifies that primary key of a relation cannot have null value. If we allow null values for a primary key then there can be multiple tuples for which primary key is having null values which means multiple tuples have the same value. This itself violates the definition of primary key.

- **Referential Integrity Constraint**: This constraint is specified between two relations. Let us suppose there are two relations R1 and R2. Foreign key in a relation R1 is the set of attributes in R1 that refer to primary key in another relation R2 provided the domain of foreign key attributes in R1 is same as that of primary key attributes in R2. The value of foreign key can be duplicate and it either occurs as a value of primary key in some tuple of R2 or is NULL.

  R1 is called the referencing relation and R2 is called referenced relation, and a referential integrity constraint holds from R1 to R2.

  The main purpose of this constraint is to check that data entered in one relation is consistent with the data entered in another relation.

  For example, consider two relation schemas:
  STUDENT (Admn_no, Roll_no, Name, Class)
  DEPARTMENT (Dept_ID, Dept_Name, Admn_no)

  Following are the primary keys:
  Admn_no is the primary key of Student relation.
  Dept_ID is the primary key of Department relation.

  Now the Admn_no- the primary key of relation in Student, is also present in relation Department. The Admn_no of Department relation is a foreign key that references primary key of Student relation (Admn_no).