B.Sc. (Computer Science)-Part-III
Subject Name- “DBMS”
Paper-I (Video Part –III)

By- Prof. Dileep Kumar Sahu

Assistant Professor

Department of Computer Application

Govt. Vishwanath Yadav Tamaskar Post Graduate Autonomous College, Durg (C.G.)

Email ID: dileepksahu20@gmail.com

Unit- III: Concept of DBMS and Data Models

By- Prof. Dileep Kumar Sahu

Content

1. Data Modeling
2. Relational Model
3. Relational Constraints
   - Key
   - Domain
   - Integrity

Objective

- Explain the Data Models.
- Describe relational model and its advantages;
- Describe the Relational Constraints and its types

Data Models

How is the data organized in a database?

Data Models

A model in database system basically defines the structure or organization of data and a set of operations on that data.

A database model defines:

- The logical data structure
- Data relationships
- Data consistency constraints

Types of Data Models

Model Types	Example
1. Object-based Models: Use objects as key data representation components.	• **Entity-Relationship Model:** It is a collection of real world objects called entities and their relationships. It is mainly represented in graphical form using E-R diagrams. This is very useful in Database design. • Object-Oriented Model: Defines the database as a collection of objects that contains both data members/values and operations that are allowed on the data. The interrelationships and constraints are implemented through objects, links and message passing mechanisms. Object-Models are useful for databases where data interrelationship are complex, for example, Computer Assisted Design based components.
2. Record based Logical Models: Use records as the key data representation components	Relational Model: It represents data as well as relationship among data in the form of tables. Constraints are stored in a meta-data table. This is a very simple model and is based on a proven mathematical theory. Network Model: In this model data is represented as records and relationship as links. This model is a very good model as far as conceptual framework is concerned but is nowadays not used in database management systems.

Types of Data Models

Model Types	Example
3. Hierarchical Data Representation Model	Hierarchical Model: It defines data as and relationships through hierarchy of data values. These models are now not used in commercial DBMS products.

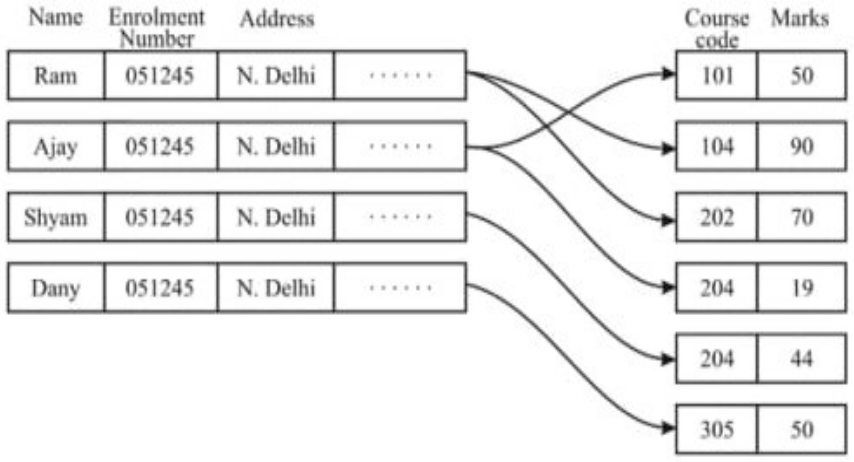

Figure 6: An example of Network Model

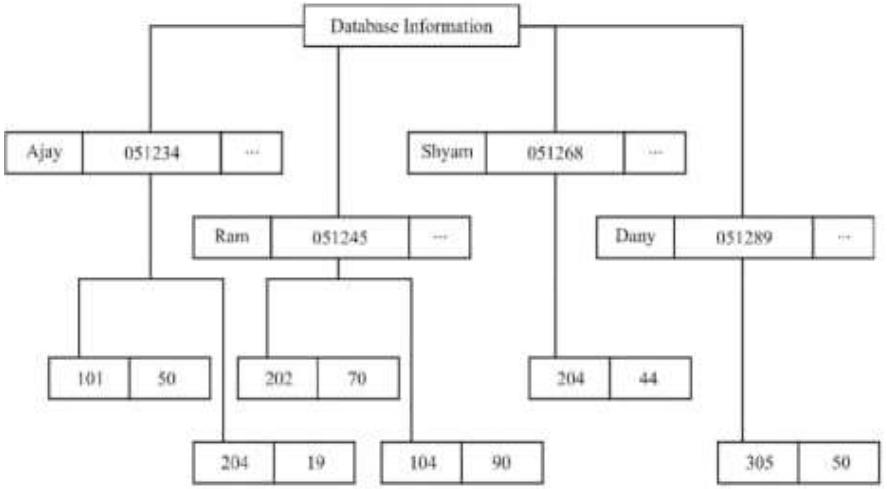

Figure 7: An example of Hierarchical Model

Data Models

- **Relational model** is a simple model in which database is represented as a collection of "Relations", where each relation is represented by a two dimensional table.
- Thus, because of its simplicity it is most commonly used.
- The following table represents a simple relation:

PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

Figure-1

Relational Model

- Advantages of Relational Model:

- **Ease of use:** The simple tabular representation of database helps the user define and query the database conveniently.
  - For example, you can easily find out the age of the person whose first name is “Vibhu”.
- **Flexibility :**
- Since the database is a collection of tables, new data can be added and deleted easily. Also, manipulation of data from various tables can be done easily using various basic operations. For example, we can add a telephone number field in the table at *Figure 1*.
- **Accuracy:** In relational databases the relational algebraic operations are used to manipulate database. These are mathematical operations and ensure accuracy (and less of ambiguity) as compared to other models.

Basic Terms

Domains, Attributes Tule and Relation:

- **Tuple:** Each row in a table represents a record and is called a tuple. A table containing 'n' attributes in a record is called n-tuple.
- **Attribute :** The name of each column in a table is used to interpret its meaning and is called an attribute. Each table is called a relation.
- **Domain :** A domain is a set of permissible values that can be given to an attribute. So every attribute in a table has a specific domain. Values to these attributes cannot be assigned outside their domains. E.g. domain may be age between 1 and 150.

Relation Name: PERSON

PERSON_ID	NAME	AGE	ADDRESS	TELEPHONE
1	Sanjay Prasad	35	b-4,Modi Nagar	011-25347527
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.	023-12245678
3	Vibhu Datt	36	c-2, New Delhi	033-1601138

Figure 2: An extended PERSON relation

Basic Terms: Relation

Relation : A relation consists of:

- Relational Schema
- Relation instance
- **Relational Schema:** A relational schema specifies the relation's name, its attributes and the domain of each attribute. If R is the name of a relation and A^1, $A^2 \ldots A^n$ is a list of attributes representing R then R(A^1, $A^2 \ldots A^n$) is called a relational schema. Each attribute in this relational schema takes a value from some specific domain called Domain (Ai).

For example, the relational schema for relation PERSON as in *Figure 2* will be:

- PERSON(PERSON_ID:integer, NAME: string, AGE:integer, ADDRESS:string)
- Total number of attributes in a relation denotes the degree of a relation. Since the PERSON relation contains 5 attributes, so this relation is of degree 5.

Relation Schema and Instance

Relation Instance or Relation State: A relation instance denoted as **r** is a collection of tuples for a given relational schema at a specific point of time.

- A relation state **r** of the relation schema R (A^1,A^2,…….A^N), also denoted by r(R) is a set of n-tuples
- r = {t^1,t^2,……t^m}
- Where each n-tuple is an ordered list of n values
- t = <v^1,v^2,……., v^n>
- where each v^i belongs to domain (A^i) or contains null values.
- The relation schema is also called *'intension'* and relation state is also called *'extension'*.

Example 1:

RELATION SCHEMA For STUDENT:

STUDENT (RollNo: string, name: string, login: string, age: integer)

STUDENT

	ROLLNO	NAME	LOGIN	AGE
t_1	3467	Shikha	shikha@gmail	20
t_2	4677	Kanu	kanu@gmail	20

Where t_1 = (3467, shikha, shikha@gmail.com , 20) for this relation instance, m = 2 and n = 4.

Relation Schema and Instance

Example 2:

RELATIONAL SCHEMA For PERSON:

PERSON (PERSON_ID: integer, NAME: string, AGE: integer, ADDRESS: string)

RELATION INSTANCE

In this instance, m = 3 and n = 4

PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

KEYS in RELATION

- **Super Key:** A **super key** is an attribute or set of attributes used to identify the records uniquely in a relation.
  - For Example, in the Relation PERSON described earlier PERSON_ID is a super key since PERSON_ID is unique for each person.
  - Similarly (PERSON_ID, AGE) and (PERSON_ID, NAME) are also super keys of the relation PERSON since their combination is also unique for each record.
- **Candidate keys:**
- Super keys of a relation can contain extra attributes. Candidate keys are minimal super key, i.e. such a key contains no extraneous attribute. The following properties must be satisfied by the candidate keys:
- •A candidate key must be **unique**.
  - A candidate key's value must **exist.** It cannot be null. (This is also called entity integrity rule)
  - A candidate key is a minimal set of attributes.
  - The value of a candidate key must be **stable.** Its value cannot change outside the control of the system.
- A relation can have more than one candidate keys and one of them can be chosen as a **primary key.**

KEYS in RELATION

- **Primary Key:**
- A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.
- A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a composite key.
- If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

RELATIONAL CONSTRAINTS

There are three types of constraints on relational database that include:

- DOMAIN CONSTRAINT
- PRIMARY KEY CONSTRAINT
- INTEGRITY CONSTRAINT

- **Domain Constraint:**

Domain Constraint

- It specifies that each attribute in a relation must contain an atomic value only from the corresponding domains. The data types associated with commercial RDBMS domains include:

1) Standard numeric data types for integer (such as short- integer, integer, long integer)

2) Real numbers (float, double precision floats)

3) Characters

4) Fixed length strings and variable length strings.

- Thus, domain constraint specifies the condition that we want to put on each instance of the relation. So the values that appear in each column must be drawn from the domain associated with that column.

Domain Constraint : Example

STUDENT

ROLLNO	NAME	LOGIN	AGE
3467	Shikha	shikha@gmail	20
4677	Kanu	kanu@gmail	20

- Here: AGE of the relation STUDENT always belongs to the integer domain within a specified range (if any), and not to strings or any other domain.
- Within a domain non-atomic values should be avoided.
- This sometimes cannot be checked by domain constraints.
- For example, a database which has area code and phone numbers as two different fields will take phone numb

Area code	Phone
11	29534466

- A non-atomic value in this case for a phone can be 1129534466, however, this value can be accepted by the Phone field.

Key Constraint

- This constraint states that the key attribute value in each tuple must be unique, i.e., no two tuples contain the same value for the key attribute.
- This is because the value of the primary key is used to identify the tuples in the relation.
- **Example 1 :** If A is the key attribute in the following relation R than A1, A2 and A3 must be unique.

R

A	B
A1	B1
A3	B2
A2	B2

- **Example 2:** In relation PERSON, PERSON_ID is primary key so PERSON_ID cannot be given as the same for two persons.

Entity Integrity Constraint

- It states that **no primary key value can be null**. This is because the primary key is used to identify individual tuple in the relation.
- So we will not be able to identify the records uniquely containing null values for the primary key attributes. This constraint is specified on one individual relation.
- Example:

Let R be the Table

A#	B	C
Null	B1	C1
A2	B2	C2
Null	B3	C3
A4	B4	C3
A5	B1	C5

- NOTE: '#' identifies the Primary key of a relation.
- In the relation R above, the primary key has null values in the tuples t^1 & t^3. NULL value in primary key is not permitted, thus, relation instance is an invalid instance.

Referential integrity constraint

- It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
- This constraint is specified on two relations (not necessarily distinct). It uses a concept of foreign key and has been dealt with in more detail in the next unit.
- Example:

R

A#	B	C^
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C3
A5	B1	C5

S

E	C#
E1	C1
E2	C3
E3	C5
E2	C2

Note:

1) '#' identifies the Primary key of a relation.
2) '^' identifies the Foreign key of a relation.

Thank You

By- Prof. Dileep Kumar Sahu