Introduction to Database Systems
Lecture 7

Engr. Madeha Mushtaq
Department of Computer Science
Iqra National University

Relational Model Concepts

- The relational model represents the database as a collection of relations.
- Informally, each relation resembles a table of values or, to some extent, a flat file of records.
- When a relation is thought of as a table of values, each row in the table represents a collection of related data values.
- A row represents a fact that typically corresponds to a real-world entity or relationship.
- The table name and column names are used to help to interpret the meaning of the values in each row.

Relational Model Concepts

- For example, the following table is called STUDENT because each row represents facts about a particular student entity.
- All values in a column are of the same data type.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

Relational Model Concepts

- In the formal relational model terminology, a row is called a tuple.
- A column header is called an attribute, and the table is called a relation.
- The data type describing the types of values that can appear in each column is represented by a domain of possible values.

Basics of RDM

- RDM used mainly for external, conceptual, and to some extent physical schema.
- Separation of conceptual and physical levels makes manipulation much easier, contrary to previous data models.
- The basic structure is relation.
- Both entities and relationships are modeled using tables/relations.

Basics of RDM

- Relations physically represented as tables.
- Table is a two dimensional representation of a relation.
- Consists of rows and columns.
- Columns represent attributes and rows represent records.
- Rows, records and tuples all these terms are used interchangeably.

Basic Properties of a Table

- Each cell of a table contains atomic/single value
- Each column has a distinct name; the name of the attribute it represents.
- The values of the attributes come from the same domain
- The order of the columns is immaterial.
- The order of the rows is immaterial.
- Each row/tuple/record is distinct, no two rows can be same.

Mathematical Relations

- Consider two sets:
  - A = {x, y}B = {2, 4, 6}
- Cartesian product of these sets
  - A X B= {(x,2), (x,4), (x,6), (y,2), (y,4), (y,6)}
- A relation is some subset of this Cartesian product, For example,
  - R1= {(x,2), (y,2),(x,6),(x,4)}
  - R2 = {(x,4), (y,6), (y,4)}

Database Relations

- The same notion of Cartesian product and relations can be applied to more than two sets, e.g. in case of three sets, we will have a relation of ordered triplets.
- Examples of some real world scenario
  - Name = {Ali, Sana, Ahmed, Sara}
  - Age = {15,16,17,18,……..,25}

Database Relations

- Cartesian product of Name & Age:
- Name X Age= {(Ali,15), (Sana,15), (Ahmed,15), (Sara,15), …., (Ahmed,25), (Sara,25)}
- CLASS = {(Ali, 18), (Sana, 17), (Ali, 20), (Ahmed, 19)}

Database Relations

- A domain D is a set of atomic values.
- By atomic we mean that each value in the domain is indivisible.
- A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn.
- It is also useful to specify a name for the domain, to help in interpreting its values.

Database Relations

- Some examples of domains follow:
- Phone_numbers: The set of ten-digit phone numbers.
- Names: The set of character strings that represent names of persons.
- Grade_point_averages: Possible values of computed grade point averages, each must be a real (floating-point) number between 0 and 4.
- Employee_ages: Possible ages of employees in a company; each must be an integer value between 15 and 80.

Database Relations

- A data type or format is also specified for each domain.
- The data type for Employee_ages is an integer number between 15 and 80.
- For Academic_department_names, the data type is the set of all character strings that represent valid department names.
- A domain is thus given a name, data type, and format.

Relation Scheme

- Let A1, A2, A3, …, A_n be some attributes and D1, D2, D3,…, D_n be their domains.
- A relation scheme relates certain attributes with their domain in context of a relation.
- Relation Scheme can be represented as
  - R = (A1:D1, A2:D2, ……, An:Dn)
  - STD = (stId:Text, stName: text, stAdres:Text, doB:Date) OR
  - STD(stId, stName, stAdres, doB)

Relation Scheme

- A relation schema is used to describe a relation; R is called the name of this relation.
- The degree of a relation is the number of attributes n of its relation schema.
- A relation of degree seven, which stores information about university students, would contain seven attributes describing each student as follows:

Relation Scheme

- STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)
- Using the data type of each attribute, the definition is sometimes written as:
- STUDENT(Name: string, Ssn: string, Home_phone: string, Address: string, Office_phone: string, Age: integer, Gpa: real)

Relation Scheme

- According to this scheme we can have a relation (instance of this scheme), like
  - STD={(stId:S001, stName:Ali, stAdres: Lahore, doB:12/12/76), (stId:S003, stName:A. Rehman, stAdres: RWP, doB:2/12/77)}

Relation Scheme

- STD={(S001, Ali, Lahore, 12/12/76), (S002, A. Rehman, RWP, 2/12/77)}

stId	stName	stAdres	doB
S001	Ali	Lahore	12/12/76
S002	A. Rehman	RWP	2/12/77

Characteristics of Relations

- **Ordering of Tuples in a Relation:**
  - A relation is defined as a set of tuples.
  - Mathematically, elements of a set have no order among them; hence, tuples in a relation do not have any particular order.
- **Ordering of Values within a Tuple:**
- **Values and NULLs in the Tuples :**
  - NULL values are used to represent the values of attributes that may be unknown or may not apply to a tuple. A special value, called NULL, is used in these cases.

Domain Constraints

- Domain constraints specify that within each tuple, the value of each attribute *A* must be an atomic value from the domain dom(*A*).
- The data types associated with domains typically include standard numeric data types for integers (such as short integer, integer, and long integer) and real numbers (float and double precision float).
- Characters, Booleans, fixed-length strings, and variable-length strings are also available, as are date, time, timestamp, and money, or other special data types.

Key Constraints

- Primary Key
- Foreign Key
- Unique Key
- Super Key

Constraints on NULL Values

- Another constraint on attributes specifies whether NULL values are or are not permitted.
- For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.
- For Example:
  - Create Table STUDENT (ID int NOT NULL, LastName varchar(255), FirstName varchar(255) NOT NULL);

Entity Integrity Constraint

- The entity integrity constraint states that no primary key value can be NULL.
- This is because the primary key value is used to identify individual tuples in a relation.
- Having NULL values for the primary key implies that we cannot identify some tuples.

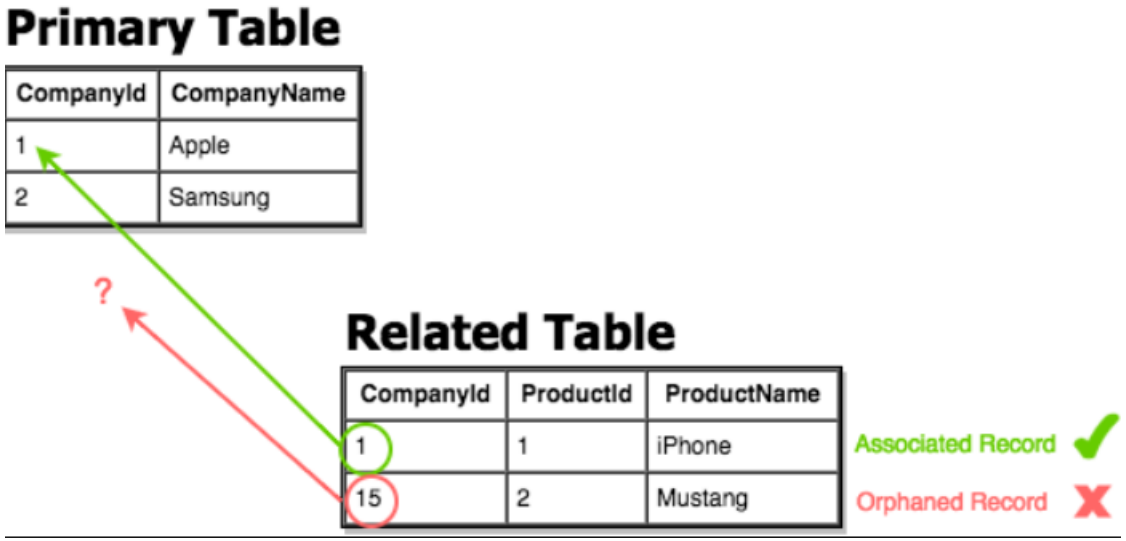
Referential Integrity

- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.
- In relationships, data is linked between two or more tables.
- This is achieved by having the foreign key (in the child table) reference a primary key value (in the primary – or parent – table).
- Because of this, we need to ensure that data on both sides of the relationship remain intact.

Referential Integrity

- So, referential integrity requires that, whenever a foreign key value is used it must reference a valid, existing primary key in the parent table.
- Thus, any primary key field changes must be applied to all foreign keys, or not at all.
- The same restriction also applies to foreign keys in that any updates (but not necessarily deletions) must be propagated to the primary parent key.

Referential Integrity Example



Referential Integrity Example

- In the above example, if we delete record number 15 in primary table, we need to be sure that there's no foreign key in any related table with the value of 15.
- We should only be able to delete a primary key if there are no associated records.
- Otherwise, we would end up with an orphaned record.

Referential Integrity

- So referential integrity will prevent users from:
  - Adding records to a related table if there is no associated record in the primary table.
  - Changing values in a primary table that result in orphaned records in a related table.
  - Deleting records from a primary table if there are matching related records.

Integrity Constraints

- All integrity constraints should be specified on the relational database schema (i.e., defined as part of its definition) if we want to enforce these constraints on the database states.
- Most relational DBMSs support key, entity integrity, and referential integrity constraints.
- These constraints are specified as a part of data definition in the DDL.

End of Slides