Database Engineering

Lecture #2

Data Models

Presented By:

Dr. Suvasini Panigrahi

Associate Professor, Department of CSE,

VSSUT, Burla

Data Models

- One important characteristics of the database approach is that it provides some level of data abstraction by hiding the implementation of operations and storage details
- A database model is a collection of logical constructs that are used to represent the data structure and the data relationships found within the database.
- Data model provides a means to achieve data abstraction
- It can be used to describe the structure of a database and the constraints that the database should obey

Categories of Data Models

- There are three categories of data models based on the type of concepts they use to describe the database structure:
  - **High-Level/Conceptual Data Model** focus on the logical nature of the data representation. They are concerned with *what* is represented rather than *how* it is represented.
  - **Representational/Implementational Data Model** place the emphasis on *how* the data are represented in the database or *how* the data structures are implemented.
  - **Low-level/Physical Data Model** provides concepts that describe the details of how the data are stored in the database

Data Model Operations

- Operation on the data model includes the following:
  - Basic Data Model Operations
  - User-Defined Operations

Basic Building Blocks of a Data Model

- Entity - anything about which data are to be collected and stored
- Attribute - a characteristic of an entity
- Relationship - describes an association among entities
- Constraint - a restriction placed on the data

The Evolution of Data Models

- Hierarchical database model
- Network database model
- Relational database model
- Entity relationship database model

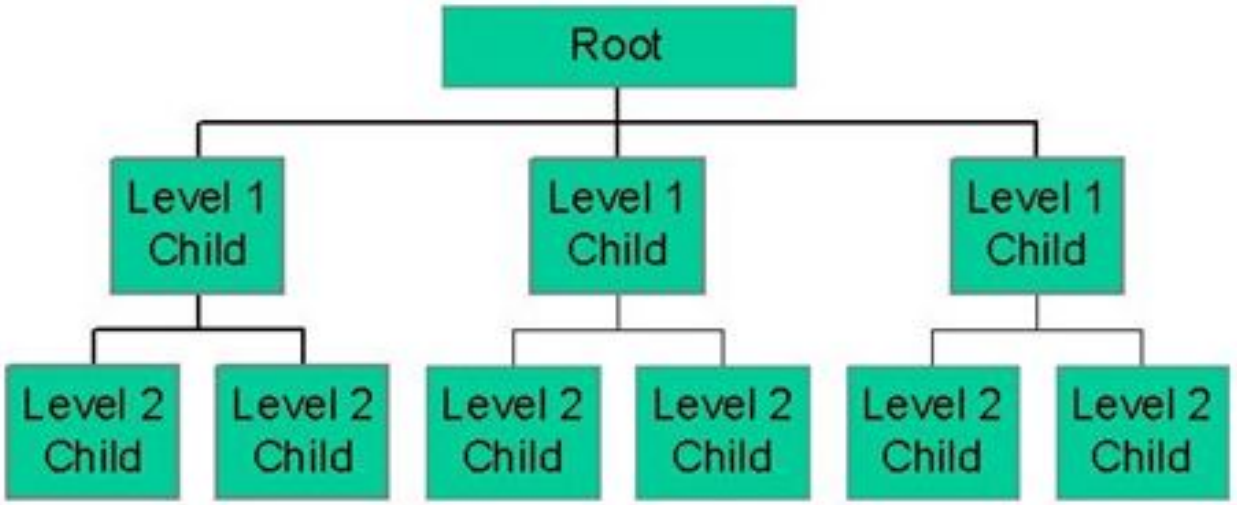
The Hierarchical Model

- Developed by IBM in 1950s to manage large amounts of data for complex manufacturing projects
- In a hierarchical model, data are viewed as a collection of tables, or segments that form a hierarchical relation.
- The data is organized into a tree-like structure where each record consists of one parent record and many children records.
- The records are collection of various fields which are connected through links
- Each field can contain only one value.

The Hierarchical Model (continued)

- The hierarchical structure contains levels, or segments
- This model depicts a set of one-to-many (1:M) relationships between a parent and its children segments
  - Each parent can have many children
  - Each child has only one parent

Hierarchical database model



- In this model, there is a main directory (root node) which contains other subdirectories (child nodes).
- Each subdirectory contains more files and directories.
- They may be directories or other files as well.

The Hierarchical Model (continued)

- In this model, only one parent must be there for each child node but a parent node can have more than one child nodes.
- Multiple parents are not allowed in this structure.
- The first node of the tree is called the root node.
- When data needs to be retrieved then the whole tree is traversed starting from the root node till the specific data is found.

The Hierarchical Model (continued)

- Advantages
  - Many of the hierarchical data model's features formed the foundation for current data models
  - The **model** allows easy addition and deletion of new information.
  - **Data** at the top of the **hierarchy** can be accessed very fast.

The Hierarchical Model (continued)

- Disadvantages
  - Difficult to manage
  - Lacks structural independence (structural changes to the database is very difficult)
  - Complex to implement

The Network Model

- The network database model was created to solve the shortcomings of the hierarchical database model.

- In this type of model, a child can be linked to multiple parents, a feature that was not supported by the hierarchical data model.
- The parent nodes are known as owners and the child nodes are called members.

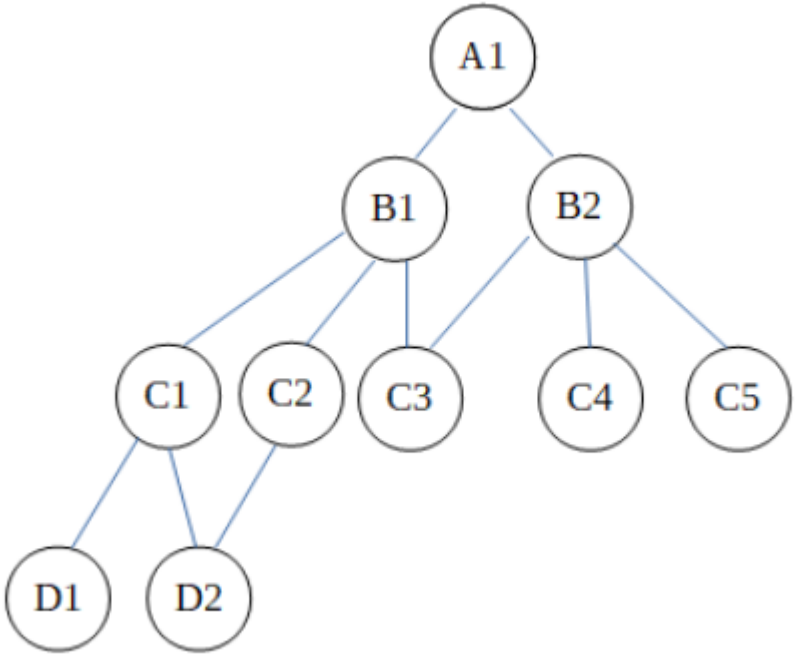
The Network Model (continued)

- The network data model was formalized by the Database Task group in the 1960s.
- In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model
- This model is the generalization of the hierarchical model.
- The model was created to represent complex data relationships more effectively than the hierarchical model.
- This database model gives a flexible way of representing objects and their relationships.

The Network Model (continued)

- The distinguishing feature of this model is that the database schema is viewed as a graph in which object types are nodes and relationship types are arcs, which is not restricted to a hierarchical structure.
- The network model replaces the hierarchical model with a graph thus allowing more general connections among the nodes.
- The main difference of the network model from the hierarchical model is its ability to handle many-to-many relationships.
- In other words it allow a record to have more than one parent.

A Network Data Model



- The network model can support many to many relationships as seen in the diagram.
- D2 and C3 each have multiple parents.
- The parents for D2 are C1 and C2 while for C3 are B1 and B2.
- In this way, the network data model can handle many to many relationships whereas the hierarchical data model didn't.

Network Data Model (continued)

- **Advantages**
  - Handles more relationship types. Hence, it is more useful than the hierarchical data model
  - It has multi-parent support
  - Deals with even larger amounts of information than the hierarchical model.
  - Supports many-to-many relationships.

Network Data Model (continued)

- **Disadvantages**
  - Much more complex than the hierarchical data model. So it is difficult to handle and maintain.
  - The structure of the Network Model is quite complicated and so the programmer has to understand it well in order to implement or modify it.
  - Lack structural independence (structural changes to the database is very difficult).

The Relational Model

- The relational model was developed by E. F. Codd in 1970.
- It is used to model data in the form of relations or tables.
- Once the conceptual model of a database is designed using Entity-Relationship (E-R) diagram, it is required to convert the conceptual model into a relational model.
- The relational model can then be implemented by using any Relational DBMS (RDBMS) language like Oracle, MySQL, etc.
- The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.
- The relational model uses a collection of tables to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has a unique name.

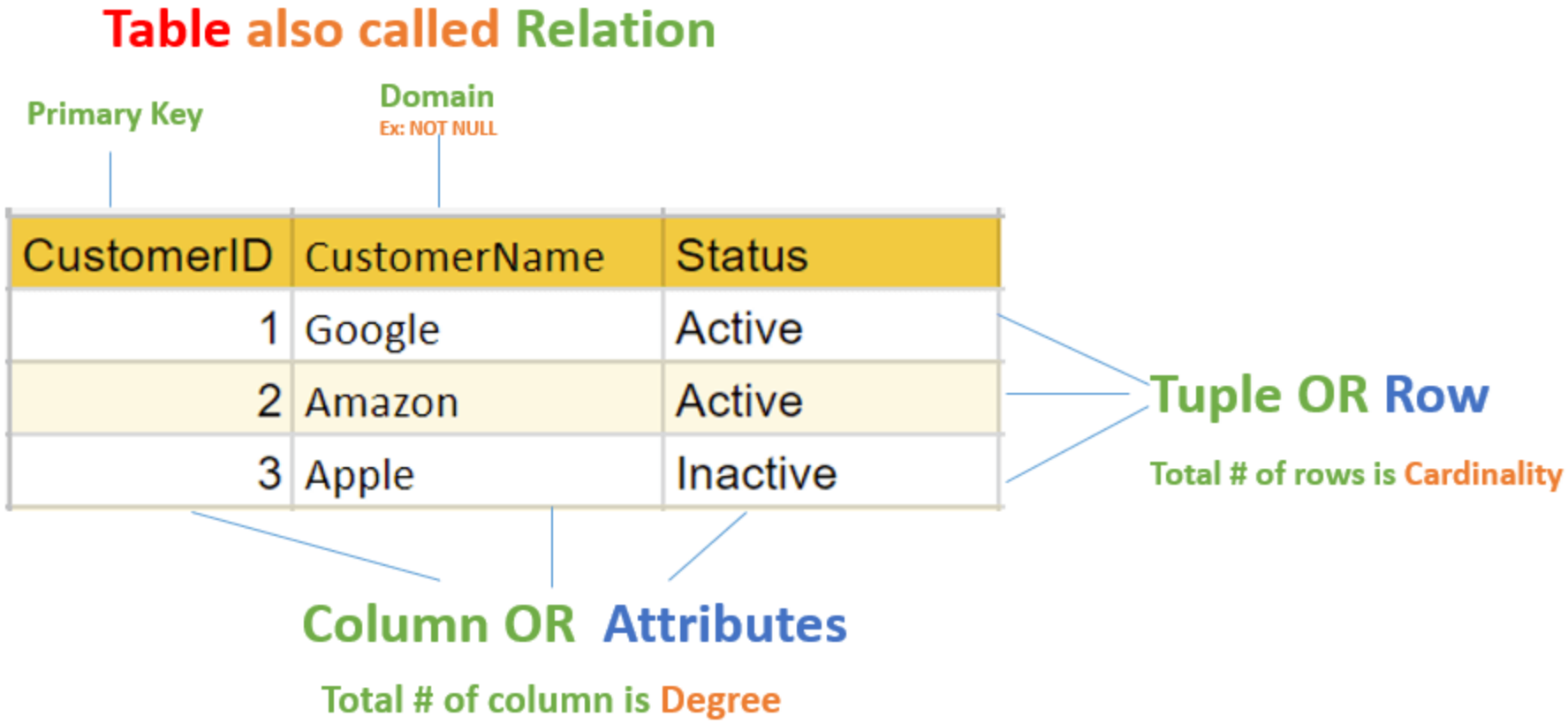
Relational Model Concepts

- **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME, etc.
- **Tables** – In the relational model the, the relations are saved in the table format. A table has two properties rows and columns. Rows represent records and columns represent attributes.
- **Tuple** – It is nothing but a single row of a table, which contains a single record.
- **Relation Schema:** A relation schema represents the name of the relation along with its attributes.
- **Degree:** The total number of attributes in a relation is called the degree of that relation.

Relational Model Concepts

- **Cardinality:** Total number of rows present in a table.
- **Column:** The column represents the set of values for a specific attribute.
- **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key** - Every relation has one or combination of two or multiple attributes to uniquely identify each tuple, which is called relation key.
- **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

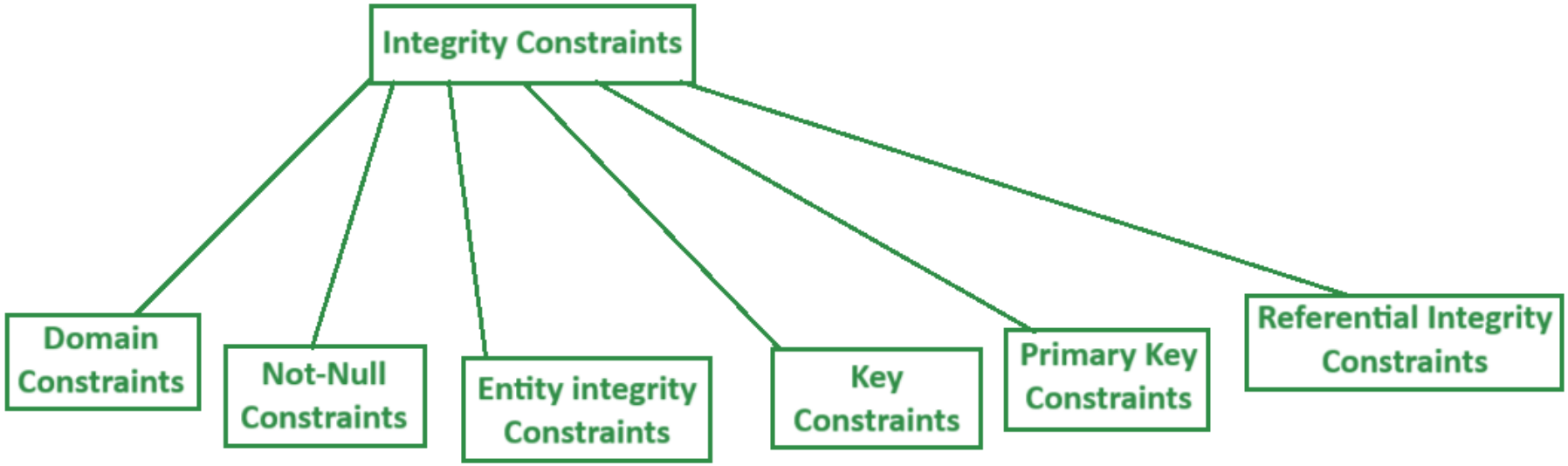
Relational Model



Relational Integrity Constraints

- Integrity constraints are the set of pre-defined rules that are used to maintain the quality of information in a database.
- Integrity constraints ensure that the data insertion, data updating, data deleting and other processes have to be performed in such a way that the data integrity is not affected.
- They act as guidelines ensuring that data in the database remain accurate and consistent.

Types of Relational Integrity Constraints



Domain Constraints

- Domain constraints can be defined as a valid set of values for an attribute.
- The data type of domain includes integer, character, string, time, date, etc.
- The value of the attribute must be available in the corresponding domain.

Domain Constraints

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Not-Null Constraint

- Not-null constraint specifies that within a tuple, attributes over which not-null constraint is specified must not contain any null value.
- This constraint enforces a column to NOT accept NULL values.
- It ensures a field to always contain a value, which means that we cannot insert a new record, or update a record without adding a value to this field.

Not-Null Constraint

- **For Example:**
  - Suppose, the not-null constraint is specified on the "Semester" attribute in the given relation/table,
  - Then the data entry of 4th tuple will violate this integrity constraint, because the "Semester" attribute in this tuple contains null value

Student_id	Name	Semester	Age
21CSE100	Ramesh	5th	20
21CSE101	Kamlesh	5th	21
21CSE102	Akash	5th	22
21CSE103	Mukesh		20

Entity Integrity Constraint

- The entity integrity constraint states that primary key value cannot be null.
- This is because the primary key value is used to identify individual rows in a relation uniquely
- If the primary key has a null value, then we cannot identify those rows.
- A table can contain a null value in it other than the primary key field.

Entity Integrity Constraint

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Key Constraint

- An attribute that can uniquely identify a tuple in a relation is called the key of the table.
- The value of the key attribute for different tuples in the relation has to be unique.
- **Example:**
- In the given table, CustomerID is a key attribute of Customer Table. So, there is a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Key Constraint

- Keys are the attribute(s) that are used to identify an entity uniquely within its entity set.
- An entity set can have multiple keys, but out of which one key will be the primary key.
- A primary key contains a unique and null value value in the relational table.
- Other keys are called secondary keys.
- **Example:**

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Primary Key Constraint

- Primary key constraint states that the primary key attributes are required to be unique and not null.
- That is, primary key attributes of a relation must not have null values and primary key attributes of two tuples must never be same.
- This constraint is specified on the primary key attributes to ensure that no two tuples are same.

Primary Key Constraint

Example

In the example below, the Student_id is the primary key attribute. The data entry of 4^{th} tuple violates the primary key constraint that is specifies on the database schema and therefore this instance of database is not a legal instance.

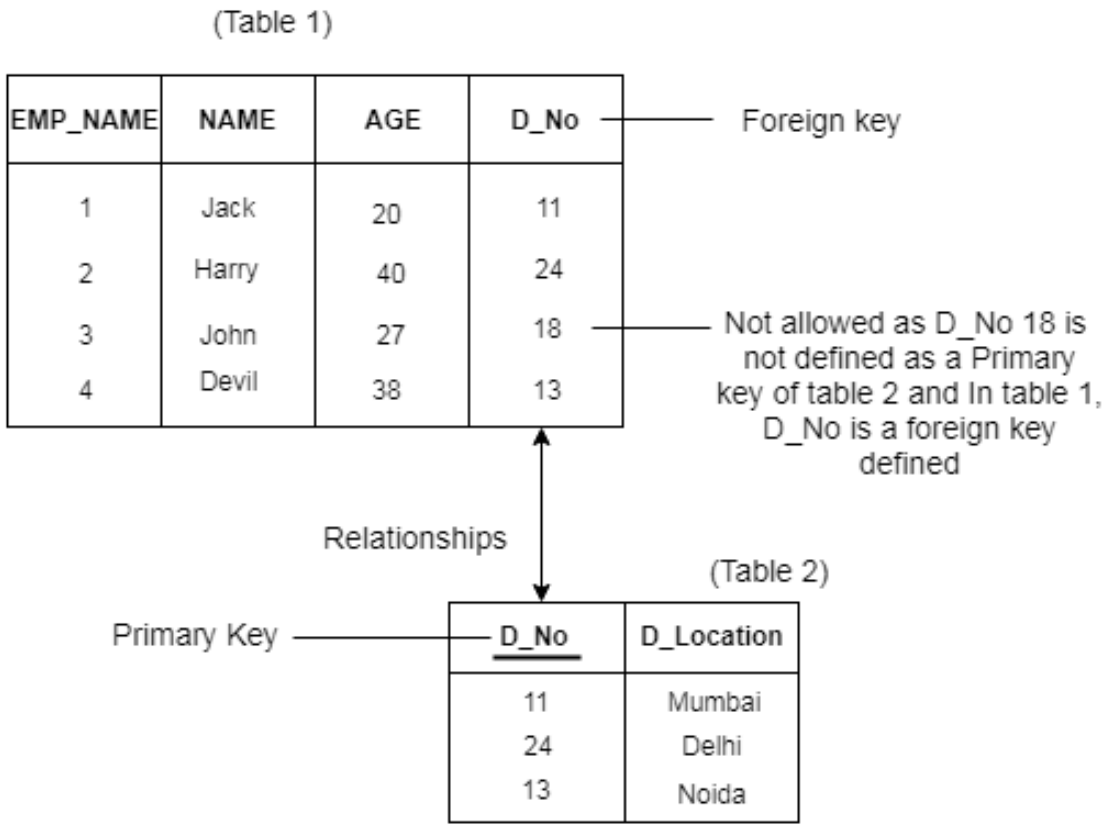
Student_id	Name	Semester	Age
21CSE101	Ramesh	5th	20
21CSE102	Kamlesh	5th	21
21CSE103	Akash	5th	22
21CSE103	Mukesh	5th	20

Referential Integrity Constraint

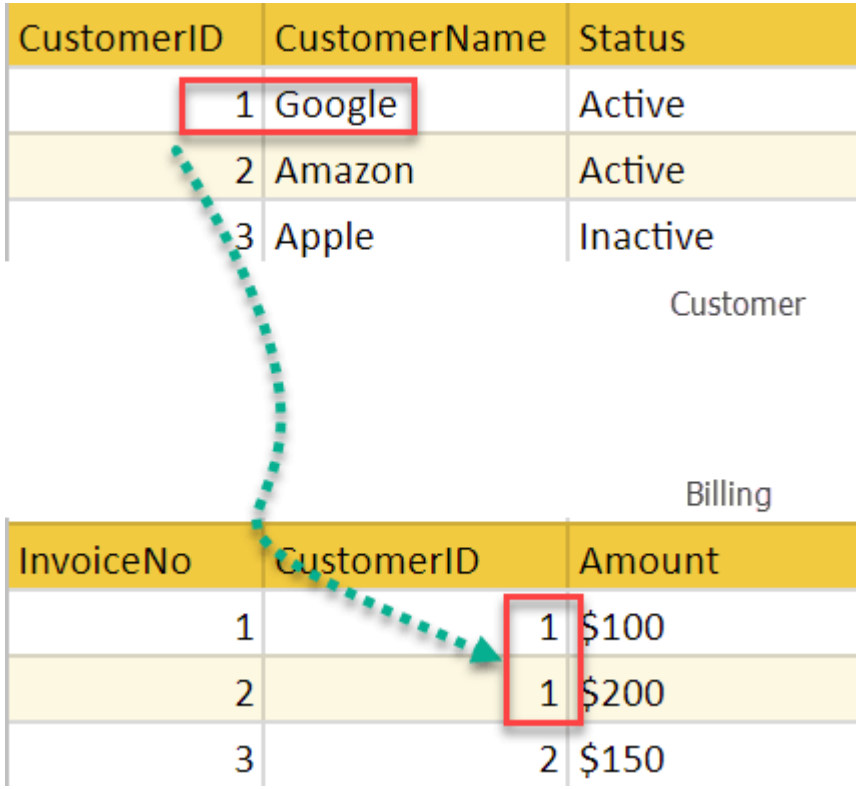
- Referential Integrity constraints in DBMS are based on the concept of Foreign Keys.
- It requires that a foreign key must have a matching primary key or it must be null.
- A referential integrity constraint is specified between the two tables.
- It maintains the correspondence between rows in these tables.
- It means the reference from a row in one table to another table must be valid.

Referential Integrity Constraint

Example: If the foreign key in Table 1 refers to the primary key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.



Referential Integrity Constraint



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	$100
2	1	$200
3	2	$150

In the above example, we have 2 relations, Customer and Billing.
Tuple for CustomerID = 1 is referenced twice in the relation Billing.
So, for CustomerName = Google, the billing amount is found to be $300.

Types of Keys in Relational Model

- Keys are one of the basic requirements of a relational database model.
- It is widely used to identify the tuples (rows) uniquely in the table.
- **Different Types of Database Keys**
  - Candidate Key
  - Primary Key
  - Super Key
  - Alternate Key
  - Foreign Key

Candidate Key

- The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD_NO in STUDENT relation.
- A table can have multiple candidate keys but only one primary key.
- The candidate key can be simple (having only one attribute) or composite as well.
- **Example:**

  {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

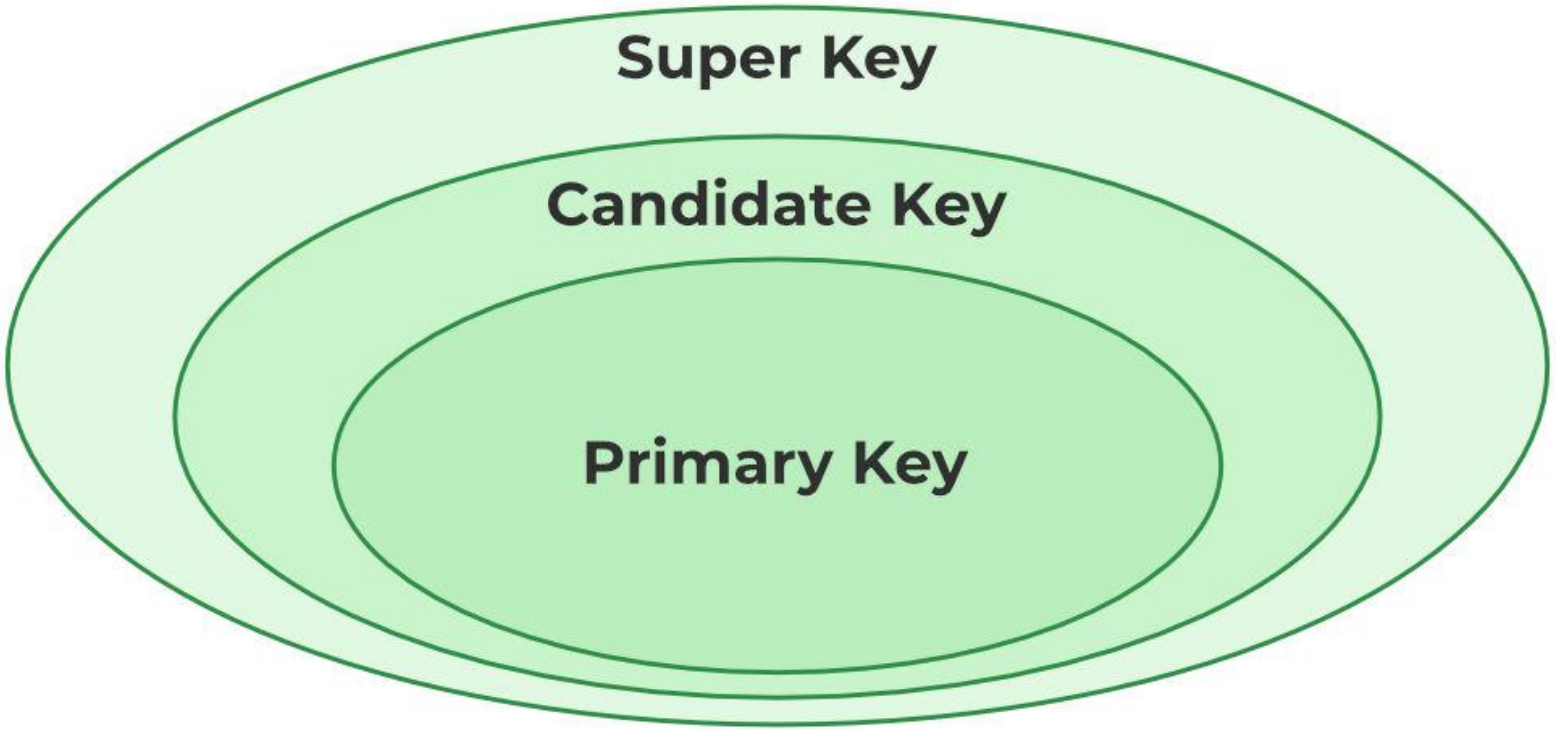
Primary Key

- There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- For Example, STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key
- It is a unique key.
- It can identify only one tuple (a record) at a time.
- It has no duplicate values, it has unique values.
- It cannot be NULL.

Super Key

- The set of attributes that can uniquely identify a tuple is known as Super Key.
- For Example, STUD_NO, (STUD_NO, STUD_NAME), (STUD_NO, PHONE_NO) are super keys of STUDENT relation
- A super key is a group of single or multiple keys that identifies rows in a table.
- Adding zero or more attributes to the candidate key generates the super key.
- Candidate key is a minimal super key (subset of a super key).
- A candidate key is a super key but vice versa is not true.

Relation between Primary Key, Candidate Key, and Super Key



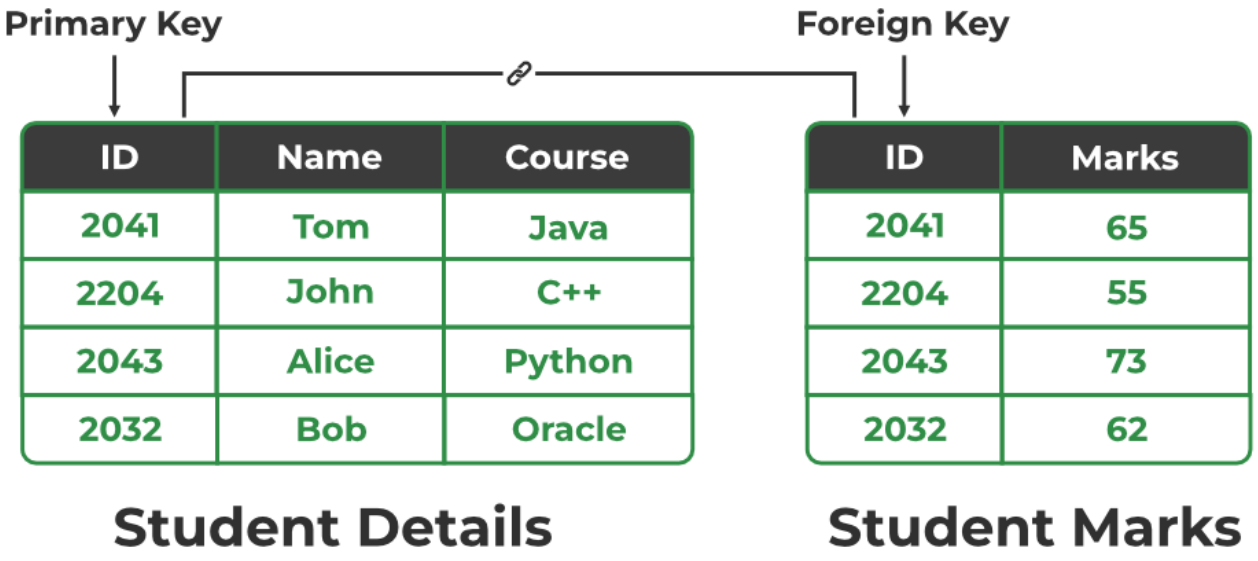
Alternate Key

- The candidate key other than the primary key is called an alternate key.
- All the keys which are not primary keys are called alternate keys.
- It is a secondary key.
- It contains two or more fields to identify two or more records.
- **Example:**

  STUD_NO, as well as STUD_PHONE both, are candidate keys for relation STUDENT but STUD_PHONE will be an alternate key

Foreign Key

- Foreign key is a key that acts as a primary key in one table and it acts as secondary key in another table.
- It combines two or more relations (tables) at a time.
- Foreign key act as a cross-reference between the tables.



Operations in Relational Model

- Four basic update operations performed on relational database model are:
  I. Insert – It is used to insert data into the relation
  II. Delete – It is used to delete tuples from the table.
  III. Update/Modify – It allows you to change the values of some attributes in existing tuples.
  IV. Select – It allows you to choose a specific range of data.
- Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

- The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Update Operation

- You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Delete Operation

- To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.
- In the given example, CustomerName= "Apple" is deleted from the table.
- The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from another table in the same database.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

Select Operation

- In the given example, CustomerName="Amazon" is selected

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
2	Amazon	Active

Best Practices for creating a Relational Model

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name
- No two rows can be identical in a table
- The values of an attribute should be from the same domain

Advantages of using Relational Model

- **Simplicity**: A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use**: The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand
- **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence**: The Structure of Relational database can be changed without having to change any application.
- **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of using Relational Model

- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- There are some forms of data and knowledge which the relational model cannot accommodate easily and adequately.
- Special forms of data:
  - Temporal data
  - Spatial data
  - Multimedia data
  - Unstructured data (warehousing/mining)
  - Document libraries (digital libraries)

ANY DOUBTS?

THANK YOU!