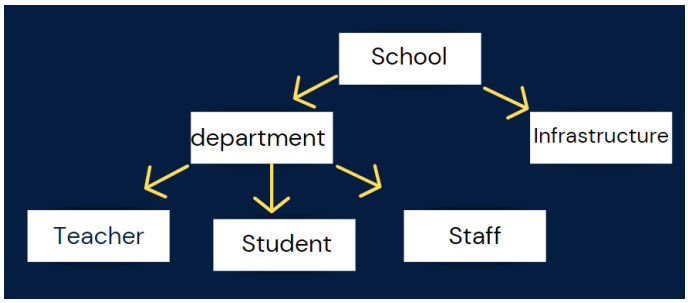
**Data Model (Definition)**

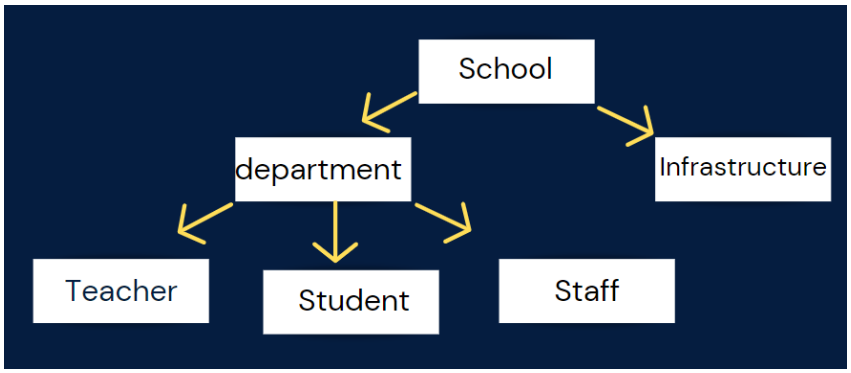
- **Summary**: A data model is an abstract representation of how data is structured, organized, and related within a database.
- **Description**: It outlines the logical arrangement and connections between data components, serving as a crucial link between real-world entities and their storage in the database. Data models are vital for understanding and designing databases.

**Types of Data Models**

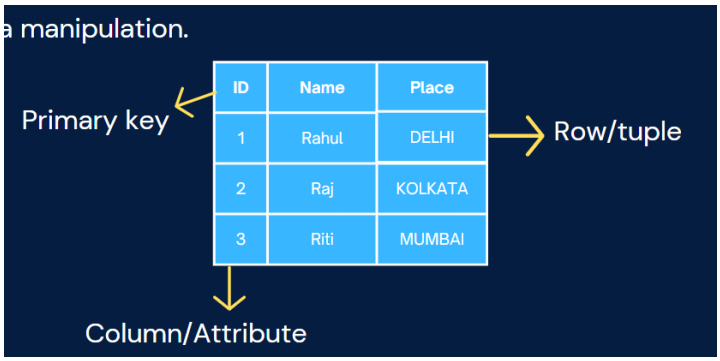
- **Summary**: The document lists several types of data models used in DBMS.
- **Description**:
    - **Hierarchical Data Model**: Represents data in a tree-like structure with parent-child relationships. Primarily used in older systems.



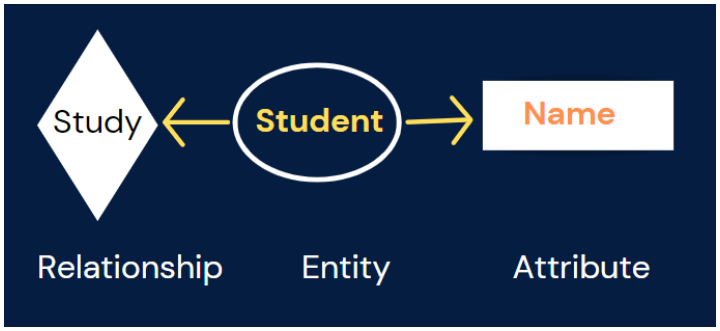

    - **Network Data Model**: Similar to hierarchical but allows multiple parent-child relationships, forming a graph-like structure. More flexible than the hierarchical model.




    - **Relational Data Model**: Organizes data into tables (relations) with rows and columns. It's the most common model, based on set theory, and uses SQL for manipulation.



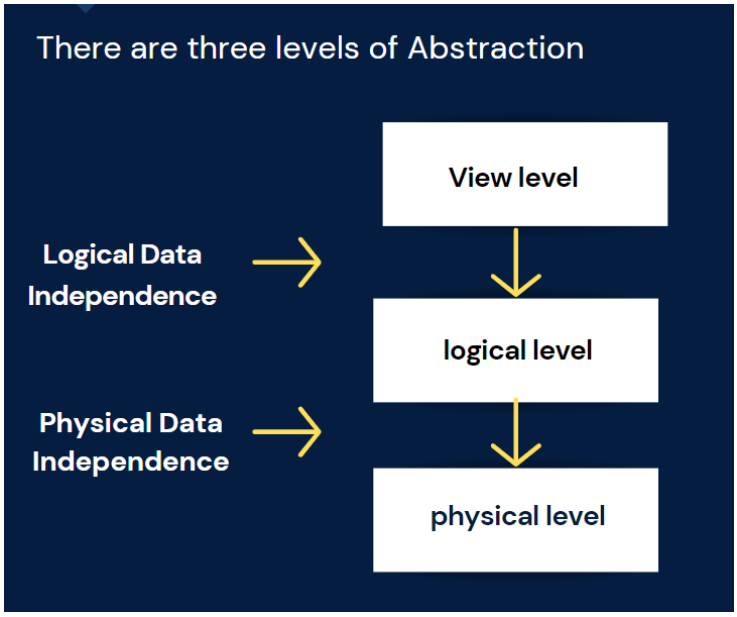

- **Entity-Relationship (ER) Model**: Used for designing relational databases. Represents data using entities (objects), attributes (properties), and relationships between entities.




- **Object-Oriented Data Model**: Extends object-oriented programming concepts to databases, representing data as objects with attributes and methods, supporting inheritance and encapsulation.
- **NoSQL Data Models**: A diverse category including document-oriented (e.g., MongoDB), key-value (e.g., Redis), column-family (e.g., Cassandra), and graph (e.g., Neo4j) models. Designed for scalability and flexibility with large, unstructured/semi-structured data.

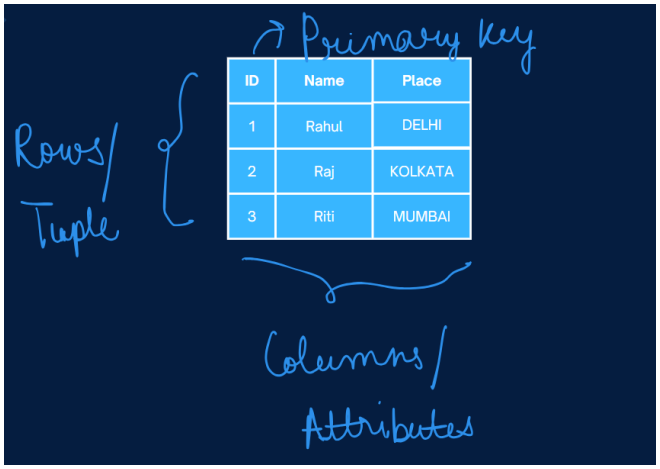
**Data Independence**

- **Summary**: The separation between the logical and physical aspects of data storage and management in a DBMS.
- **Description**: This principle provides benefits like flexibility, security, and easier maintenance. It relates to the levels of abstraction:
    - **Logical Data Independence**: The ability to change the logical schema without changing the external views or application programs.
    - **Physical Data Independence**: The ability to change the physical schema (how data is stored) without changing the logical schema.



**Essential Components of Tables (Relational Model)**

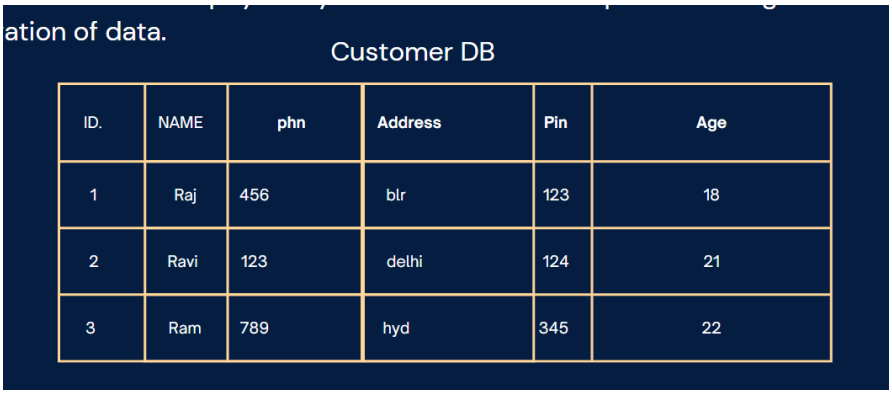
- **Summary**: Describes the fundamental parts of a table in the relational model.
- **Description**:
    - **Row/Tuple**: Represents a single record or data entry within the table.
    - **Cardinality**: The number of rows (tuples) in a table.
    - **Column/Attribute**: Represents a specific property or characteristic of the data being stored (e.g., "Name", "Age").
    - **Degree**: The number of columns (attributes) in a table.




    - **Constraints**: Rules that data within the table must satisfy (e.g., uniqueness, nullability, check constraints, default values).
    - **Keys**: Attributes used for identification and establishing relationships.
        - **Primary Key**: Uniquely identifies each row in a table.
        - **Foreign Key**: A field that links to the primary key of another table, establishing a relationship.

**Views in DBMS**

- **Summary**: A view is a virtual table derived from one or more base tables.
- **Description**: It doesn't store data physically but provides a logical representation based on a query. Views can simplify complex queries, provide security by restricting access, and present data in a specific format.


ation of data.

Customer DB

| ID. | NAME | phn | Address | Pin | Age |
|---|---|---|---|---|---|
| 1 | Raj | 456 | blr | 123 | 18 |
| 2 | Ravi | 123 | delhi | 124 | 21 |
| 3 | Ram | 789 | hyd | 345 | 22 |

**Keys in DBMS**

- **Summary**: Keys are attributes or sets of attributes used to ensure data integrity, uniqueness, and efficient data retrieval.
- **Description**: Types include:
    - **Candidate Key**: An attribute or set of attributes that can uniquely identify a tuple (row). A table can have multiple candidate keys. ***Ex:*** An example Student table shows ID, Roll no, Aadhar Card as potential candidate keys.
    - **Primary Key**: The candidate key chosen to uniquely identify records in a table. It must be unique and cannot contain null values.***Ex:*** An example table highlights the 'ID' column as the Primary Key.
    - **Foreign Key**: An attribute in one table that refers to the primary key of another table, establishing a link or relationship. ***Ex:*** Diagrams show a "Student" table (Referenced Table with Primary Key 'Roll no') and a "Subject" table (Referencing Table with Foreign Key 'Roll no'), illustrating the relationship.
    - **Super Key**: A set of one or more attributes that, collectively, uniquely identify a tuple. Any set of attributes that includes a candidate key is a super key. A candidate key is a minimal super key.

**Referential Integrity**

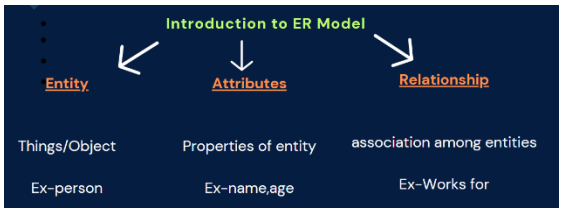
- **Summary**: Ensures that relationships between tables remain consistent.
- **Description**: When one table has a foreign key referencing another table, referential integrity ensures that you cannot add a record to the referencing table unless a corresponding record exists in the referenced table. It also dictates actions (like CASCADE DELETE/UPDATE, SET NULL) when records in the referenced table are deleted or updated.

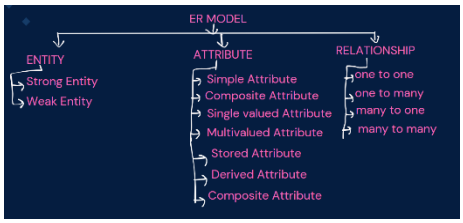
**Integrity Constraints in DBMS**

- **Summary**: Rules enforced on data columns to ensure accuracy and reliability.
- **Description**: Types include:
    - **Domain Integrity**: Ensures values in a column adhere to a specified domain (valid data type, format, range). Ex: Date column must contain valid dates.
    - **Entity Integrity**: Ensures each row has a unique, non-null primary key.
    - **Referential Integrity**: Ensures foreign key values match existing primary key values in the referenced table or are null.
    - **Key Constraint**: Ensures the primary key is unique.
    - **Check Constraint**: Enforces a specific condition for each row (e.g., age > 18).
    - **Null Constraint**: Specifies whether a column can accept NULL values.
    - **Unique Constraint**: Ensures values in a column (or set of columns) are unique across all rows (allows NULL, unlike primary key).
    - **Default Constraint**: Assigns a default value to a column if no value is provided during insertion.

**ER Model (Entity-Relationship Model)**

- **Summary**: A conceptual modeling approach for database design, visually representing entities, attributes, and their relationships.




- **Description**: Key components:
    - **Entity**: A real-world object or concept (e.g., Person, Course, Department).
        - **Strong Entity**: Has its own primary key and exists independently. Represented by a single rectangle.
        - **Weak Entity**: Does not have a primary key of its own and depends on a related strong ("owner") entity for its identification. Represented by a double rectangle.
    - **Attribute**: A property or characteristic of an entity (e.g., Name, Age, ID). Represented by an ellipse.
        - **Simple**: Atomic, cannot be divided (e.g., Age).
        - **Composite**: Can be divided into smaller parts (e.g., Name into FirstName, LastName). Represented by an ellipse connected to other ellipses.
        - **Single-Valued**: Holds only one value per entity instance (e.g., ID).
        - **Multi-Valued**: Can hold multiple values per entity instance (e.g., PhoneNumber). Represented by a double ellipse.
        - **Stored**: Directly stored in the database (e.g., DateOfBirth).
        - **Derived**: Value calculated from other attributes (e.g., Age derived from DateOfBirth). Represented by a dashed ellipse.
        - **Complex**: Attributes that are nested combinations of composite and multi-valued attributes.
    - **Relationship**: An association between two or more entities (e.g., "Works for", "Enrolls"). Represented by a diamond.
        - **Strong Relationship**: Typically links strong entities.
        - **Weak Relationship**: Links a weak entity to its owner strong entity. Often represented by a double diamond.

**ER Model - Degree & Cardinality**

- **Summary**: Describes the number of entities involved in a relationship (degree) and the number of instances related (cardinality).
- **Description**:
    - **Degree of Relationship**: Number of participating entity types. Unary (1), Binary (2), Ternary (3), n-ary (>3).
    - **Cardinality Ratios**: Specify the number of instances of one entity that can relate to instances of another entity.
        - **One-to-One (1:1)**: Each entity instance in one set is related to exactly one instance in the other set. (e.g., Student Enrolls Course, if a student can enroll in only one course and a course can have only one student - specific scenario).
        - **One-to-Many (1:N)**: One instance in the first set can relate to multiple instances in the second, but each instance in the second set relates to only one in the first. (e.g., Author Reads Books - one author can write many books, but a specific book copy is written by one author).
        - **Many-to-One (N:1)**: Multiple instances in the first set can relate to one instance in the second. (e.g., Employees works Department - many employees work in one department).
        - **Many-to-Many (N:N)**: Instances in the first set can relate to multiple instances in the second, and vice-versa. (e.g., Students Enrolls Courses - a student can enroll in many courses, and a course can be enrolled by many students).

**ER Model - Participation Constraints**

- **Summary**: Define whether an entity's existence depends on its relationship with another entity.
- **Description**:
    - **Total Participation (Mandatory)**: Every instance of an entity set *must* participate in the relationship. Represented by a double line connecting the entity to the relationship diamond. (e.g., If every employee must belong to a department).
    - **Partial Participation (Optional)**: Instances of an entity set may or may not participate in the relationship. Represented by a single line. (e.g., If an employee is not required to manage a department).

**Extended ER (EER) Features**

- **Summary**: Enhance the basic ER model to represent more complex relationships, particularly hierarchies.
- **Description**: Needed for real-world data exhibiting hierarchical structures, improving reusability, integrity, and reducing complexity.
    - **Specialization**: Top-down process of defining subtypes (subclasses) from a supertype (superclass) based on distinguishing characteristics. Subtypes inherit attributes/relationships from the supertype but can have their own unique ones. (e.g., Vehicle specialized into Car, Truck). Represents an "is-a" relationship.
    - **Generalization**: Bottom-up process of identifying common features among entity sets and creating a generalized supertype. (e.g., Identifying common attributes of Car and Truck to create a Vehicle supertype). Represents an "is-a" relationship.
    - **Aggregation**: Treats a relationship between entities as a higher-level entity itself. Allows relationships to participate in other relationships. Used to model "part-of" or "composed-of" scenarios abstractly.

**Steps to Draw an ER Model**

- **Summary**: The document outlines a 10-step process for creating an ERD.
- **Description**: 1. Recognize entities. 2. Specify attributes. 3. Discover relationships (including mapping/participation constraints). 4. Define relationship cardinality. 5. Construct ERD. 6. Annotate. 7. Review/refine. 8. Document. 9. Validate with stakeholders. 10. Implement schema.

**ER Model Example (Instagram)**

- **Summary**: The document walks through applying the ER modeling steps to design a simplified Instagram database.
- **Description**: It identifies entities (userProfile, userFriends, userPost, userLogin, userLikes), lists attributes for each (specifying types like primary key, composite, multi-valued), and defines relationships between them with cardinalities (e.g., userProfile has userPost (1:N, total participation), userProfile has userLogin (1:1)).

**Relational Model**

- **Summary**: A data model where data is organized into tables (relations).
- **Description**: Key terms: Table (Relation), Row (Tuple), Column (Attribute), Domain (allowed values for an attribute), Degree (number of columns), Cardinality (number of rows). It emphasizes relationships via foreign keys and data integrity. Values should ideally be atomic (indivisible). Each row must be unique (enforced by keys).

**Converting ER Model to Relational Model**

- **Summary**: Describes the process of translating an ER diagram into a set of relational tables.
- **Description**:
    - **Step 1: Map Entities**: Each strong entity becomes a table with its simple attributes as columns. Weak entities also become tables, including the primary key of the owner entity as a foreign key, forming part of its own composite primary key.
        - ***PDF Visual (Page 98):*** Shows mapping a Strong Entity 'Person' and a Weak Entity 'Address' (dependent on Person) to tables.
    - **Step 2: Map Attributes**: Handle composite attributes (map sub-parts) and multi-valued attributes (create a separate table).
    - **Step 3: Select Keys**: Define primary keys for each table.
    - **Step 4: Map Relationships**:
        - **1:1**: Can merge into one table or keep two tables with a foreign key on either side.
        - **1:N / N:1**: Keep two tables; the foreign key goes on the "many" side, referencing the "one" side's primary key.
        - **N:N**: Create three tables: one for each entity and a third "junction" or "linking" table containing foreign keys from both related entities (forming a composite primary key for the junction table).