DATA MODELS

Definition of data models

In simple terms, a data model is a way to organize and understand information. A data model is a blueprint for organizing data in a way that's easy to understand and use (structuring and organizing data).

Concept of data model

Imagine a blueprint for a house. It shows the rooms, their sizes, and how they connect. This blueprint helps builders know where to put walls, doors, and windows.

A data model is like a blueprint for data. It shows the different pieces of information (like rooms in a house) and how they fit together. For example, a customer might have a name, address, and phone number. These pieces of information are connected because they belong to the same person.

Think of a library. The books are organized by subject, author, or title. This makes it easy for people to find the books they need. A data model for a library would show the different types of books, who wrote them, and how they are organized.

In essence, **Data modeling is about creating a structured plan for organizing and managing information**

Approach in Data Models

(a) Conceptual Data Model; (b) Logical Data Model; (c) Physical Data Model

1. **Conceptual Data Model**

Focusing on high-level relationships between entities.

2. **Logical Data Model**

Focusing on detailed representation of entities, attributes, and relationships, Including primary keys, foreign keys, and normalization.

3. **Physical Data Model**

Specifies how the logical data model will be realized in a database, including data types, and table structure.

Types of Data Models

1. **Flat Model**
    - **Simple Structure:** No defined relationships between records.
    - **Common Use:** Suitable for small, simple datasets.

- **Example:** A spreadsheet with a list of names, addresses, and phone numbers.

2. **Hierarchical Model**

- **Tree-like Structure:** One-to-many relationships.
- **Limitations:** Rigid structure, difficulty handling complex relationships.
- **Example:** Organizational charts or family trees.

3. **Network Model**

- **Extended Hierarchical:** Many-to-many relationships.
- **Complexity:** More complex than hierarchical, but can handle more relationships.
- **Example:** Social networks where users can have multiple friends and follow multiple users.

4. **Relational Model**

- **Tables and Relationships:** Based on tables and relationships between them.
- **Normalization:** Often involves normalization to reduce redundancy.
- **Example:** Most modern database systems, such as MySQL, PostgreSQL, and Oracle.

5. **Object-Relational Model (ORD)**

- **Combination:** Combines relational and object-oriented features.
- **Benefits:** Supports complex data structures and object-oriented programming paradigms.
- **Example:** Databases that support both relational operations and object-oriented constructs.

6. **Star Schema**
   - **Dimensional Modeling:** A straightforward way to show data.
   - Example**:** A Star Schema might have a "Sales" table with columns for "Date," "Product," "Customer," and "Quantity."
7. **Snowflake Schema:**
   - **Dimensional modeling**: Is an extension of a star schema, a more detailed way to show data.
   - **Example:** Following the same sales table above, could break down "Product" into "Category" and "Subcategory" for more details.

Assignment: *Why do we need data models?*

DATA MODELING

Creating a Simple Data Model in MS Access

FGC Minna: Data processing SS2 First Term

Objects in MS Access:

- **Tables:** The foundation of a database. They store data in a structured format with rows (records) and columns (fields). Each record represents a specific entity (e.g., a student), and each field holds a particular data type (e.g., name, text; grade, number).
- **Queries:** Allow you to retrieve specific data from one or more tables based on certain criteria. It's like asking a question of the database to get the information you need.
- **Forms:** User interfaces for interacting with data. They provide a way to enter, edit, or view data in a more user-friendly format than raw tables.
- **Reports:** Present data in a formatted and organized way, often for printing or sharing. They are useful for summarizing or analyzing data.

Steps:

A. **Creating a Table**:
   - Open MS Access and select "Blank Database."
   - Type the database name (School Database) and click on "Create"
   - Define table name and fields with names and data types
   - Save the table.

Example: Three tables named "Students, "Teachers" and "Courses", add the field with each entities

- **Students:** Student ID, Name, Grade, Contact Information.
- **Teachers:** Teacher ID, Name, Subject(s), Contact Information.
- **Courses:** Course ID, Course Name, Description, Credits.

With any of the corresponding data types below:

1. **Text**
2. **Purpose:** Stores alphanumeric data like names, addresses, or short descriptions.
3. **Length:** Up to 255 characters.
4. **Memo**
5. **Purpose:** Stores larger amounts of text, such as lengthy descriptions or notes.
6. **Length:** Up to 64,000 characters.
7. **Number**
8. **Purpose:** Stores numerical data, including integers, decimals, and currency values.
9. **Types:** Integer, Decimal, Currency, Single, Double, etc.
10. **AutoNumber**
11. **Purpose:** Automatically generates unique sequential or random numbers.
12. **Usage:** Often used as primary keys to uniquely identify records.
13. **Yes/No**
14. **Purpose:** Stores Boolean values (true or false).
15. **Usage:** Useful for indicating binary choices.
16. **Date/Time**
17. **Purpose:** Stores dates and times.

18. **Formats:** Various formats, such as MM/DD/YYYY, DD/MM/YYYY, etc.
19. **Currency**
20. **Purpose:** Stores monetary values with a specific number of decimal places.
21. **Usage:** For financial data.
22. **OLE Object**
23. **Purpose:** Stores objects from other applications, such as Word documents, Excel spreadsheets, or PowerPoint presentations.
24. **Hyperlink**
25. **Purpose:** Stores web addresses or file paths.
26. **Usage:** For linking to external resources.
27. **Attachment**
28. **Purpose:** Stores files of various types, such as images, PDFs, or documents.

B. **Creating a Form**:
   - Click on "Create" and select "Form."
   - Customize the form layout and fields as needed.
   - Save the form with a descriptive name.

C. **Creating a Query**:
   - Click on "Create" and select "Query Wizard."
   - Choose the type of query and specify the fields and criteria.
   - Save the query with a meaningful name.

D. **Creating a Report**:
   - Click on "Create" and select "Report."
   - Choose the data source and customize the report layout.
   - Save the report.

DATA MODELING

Significance of Data model

Imagine building a LEGO castle. You have all the pieces, but without a plan, it would be a chaotic mess! A plan, or blueprint, helps you organize the pieces and build a beautiful castle.

Data modeling is like a blueprint for your data. It helps you understand how different pieces of information (like the LEGO bricks) fit together.

- **Organization:** Just like a blueprint keeps your LEGO castle organized, data modeling keeps your information organized and easy to find.
- **Understanding:** It helps you understand how different pieces of information relate to each other.
- **Building Systems:** Data models are used to create computer systems that store and manage data, just like the instructions for building your LEGO castle.
- **Decision Support:** Well-structured data models provide the foundation for effective data analysis and decision-making.
- **Communication:** Data models can serve as a common language for stakeholders, improving communication and collaboration.

- **Cost-Efficiency:** By optimizing data storage and retrieval, data modeling can help reduce costs associated with data management.
- **Scalability**: Standardized models are scalable, allowing businesses to manage increasing volumes of data effectively as they grow.
- **Data Quality:** Data modeling helps ensure data quality by defining rules and constraints, preventing inconsistencies and errors.

Examples of standard data model

Imagine you're building a LEGO castle. To keep it organized and easy to build, you follow a blueprint. This blueprint is like a standard data model.

Standard data models are widely used in a particular industry like blueprints for different types of data; they provide a set of rules and guidelines on how to organize and structure information.

Examples:

Examples of standard data models representing a different domain or industry. Each of these domains has its own set of entities and relationships that are essential for managing and analyzing data effectively

- **IMS Global Learning Consortium:** A non-profit organization that develops standards for e-learning and educational technology. **Entities**: Courses, Learning Objectives, Learners, Assessments
  ISO 10303 (STEP) – A standard that helps engineers share product and design data between different computer programs.

  ISO 15926 – A standard used in oil and gas plants to keep data about equipment and operations for the whole life of the plant.

  IDEAS Ontology – A shared system used by the military of several countries to make sure their data can work together.

  EN 12896 (Transmodel) – A standard for public transport data, such as bus schedules, fares, and passenger information.

  Seabed Survey Data Model (SSDM) – A standard way to record and share data about the seabed, mostly used in oil and gas exploration.

  Common Education Data Standards (CEDS) – A U.S. standard that makes student and school data easy to use and share across the education system.

  SIF (Schools Interoperability Framework) – A standard that allows schools in countries like the U.S., UK, and Australia to share student information between systems.

Normal Form

The normal forms defined in relational database theory represent guidelines for record design.

Types of Normal forms

A. First Normal Forms

- **Rule:** Each cell in a table should contain a single, atomic value.
- **Violation:** A cell containing multiple values (e.g., comma-separated list).
- **Example:** A table where a single cell stores both a student's name and their class.

Example:

- Not in 1NF:

StudentID	Name	Courses
1	Alice Johnson	Math, Science, History
2	Bob Smith	English, Art

- In 1NF:

StudentID	Name	Course
1	Alice Johnson	Math
1	Alice Johnson	Science
1	Alice Johnson	History
2	Bob Smith	English
2	Bob Smith	Art

Explanation: The updated table ensures that each cell holds a single piece of information, making the data atomic and the table easier to manage.

B. Second Normal Form (2NF)

- **Rule:** A table must be in 1NF and all non-key attributes must depend on the entire primary key.
- **Violation:** A non-key attribute depends on only part of the primary key.
- **Example:** A table where a student's major depends on their department but not their student ID (if the primary key is a composite of student ID and department).

Example:

Consider a table:

Student ID	Department	Major
101	CS	Computer Science
102	CS	Computer Science
103	Math	Mathematics

In this example, the primary key is (Student ID, Department). However, the "Major" attribute depends only on the "Department" attribute, not the entire primary key. This violates 2NF.

To achieve 2NF, you would split the table into two:

Student ID	Department
101	CS
102	CS
103	Math

Department	Major
CS	Computer Science
Math	Mathematics

By separating the "Major" attribute into a separate table, we ensure that it depends on the entire primary key of its own table. This eliminates the partial dependency and brings the database into 2NF.

C. Third Normal Form (3NF)

- **Rule:** A table must be in 2NF and no non-key attribute should transitively depend on another non-key attribute.
- **Violation:** A non-key attribute depends on another non-key attribute through a transitive relationship.

Example:

Imagine a table that records employee details:

EmployeeID	EmployeeName	DepartmentID	DepartmentName	DepartmentLocation
1	Alice Smith	101	HR	New York
2	Bob Johnson	102	IT	San Francisco
3	Carol Davis	101	HR	New York

Here's the breakdown of the table:

Primary Key: EmployeeID

Non-Key Attributes: EmployeeName, DepartmentID, DepartmentName, DepartmentLocation

Violation Explanation:

DepartmentName and DepartmentLocation depend on DepartmentID.

DepartmentID depends on EmployeeID.

Thus, DepartmentName and DepartmentLocation transitively depend on EmployeeID through DepartmentID.

Correction to Achieve 3NF:

To achieve 3NF, we should eliminate the transitive dependency by splitting the table into two tables:

1. **Employees Table**: Contains information specific to employees.

EmployeeID	EmployeeName	DepartmentID
1	Alice Smith	101
2	Bob Johnson	102
3	Carol Davis	101

2. **Departments Table**: Contains information about departments.

DepartmentID	DepartmentName	DepartmentLocation
101	HR	New York
102	IT	San Francisco

Assignment: (a) *What are the benefits of using Normal Forms (b) with the aid of a table explain insertion, deletion and update anomaly in a table.*

Normal Form

Determinant of Normal Form (Unique and Non Unique Determinant)

- **Unique Determinant**: An attribute or a set of attributes that uniquely identifies a record and determines the values of other attributes. Typically, the primary key is a unique determinant.
- **Non-Unique Determinant**: An attribute or a set of attributes that determines other attributes but does not uniquely identify a record.

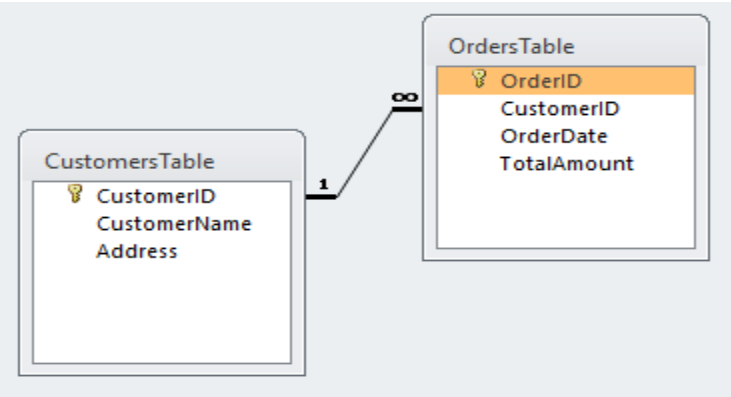
Concept of Foreign Key

A foreign key is a field in a relational database table that references the primary key of another table.

Example: Consider two tables:

- **CustomersTable:** CustomerID (primary key), CustomerName, Address
- **OrdersTable:** OrderID (primary key), CustomerID (foreign key), OrderDate, TotalAmount

The CustomerID in the Orders table is a foreign key that references the CustomerID in the Customers table. Note, when creating the two tables, set the datatypes to text in all the field of both table. Close all open tables, click on database tools from the menu bar, select relationships, Add the two tables from show table, click CustomerID and drag it to CustomerID in ordersTable, check enforce Referential Integrity, click create, select relationship report.



Differences between Primary Key and Foreign Key
Primary Key:

- **Unique Identifier:** A column or combination of columns that uniquely identifies each row in a table.
- **Non-Null:** Cannot contain null values.
- **Enforced Uniqueness:** Ensures that no two rows have the same primary key value.
- **Example:** In a "Customers" table, "CustomerID" might be the primary key.

Foreign Key:

- **Reference to Another Table:** A column or combination of columns that references the primary key of another table.
- **Relationship:** Creates a relationship between the two tables.
- **Referential Integrity:** Ensures data consistency by preventing invalid references.
- **Example:** In an "Orders" table, "CustomerID" might be a foreign key referencing the "CustomerID" in the "Customers" table.

Entity Relationship Model

It consists of entities (which represent objects or concepts) and the relationships between them. A graphical representation of the entities, attributes, and relationships that make up a database system.

Components of entity relationship

a. **Entities:** represent objects or concepts within a data model, each represent a rectangles in ER diagrams.. For example:

- **Student**: Represents individual students in the system.
- **School**: Represents different schools.

b. **Attributes**: are characteristics or properties of an entity. The key attribute is often the unique identifier.

Here's how attributes could be outlined for **Student** entities:

- **Student ID** (Primary Key)
- Surname
- First Name

- Date of Birth
- Telephone
- Address

c. **Relationships:** These represent the connections between entities

Relationship between Entities, and Attribute

1. You can use Gliffy and Lucid Chart Software to create ERD. ERDs use symbols like rectangles, diamonds, ovals, and lines to represent the connections between entities, relationships, and attributes in two ways: **Chen notation** and **Crow's notation**

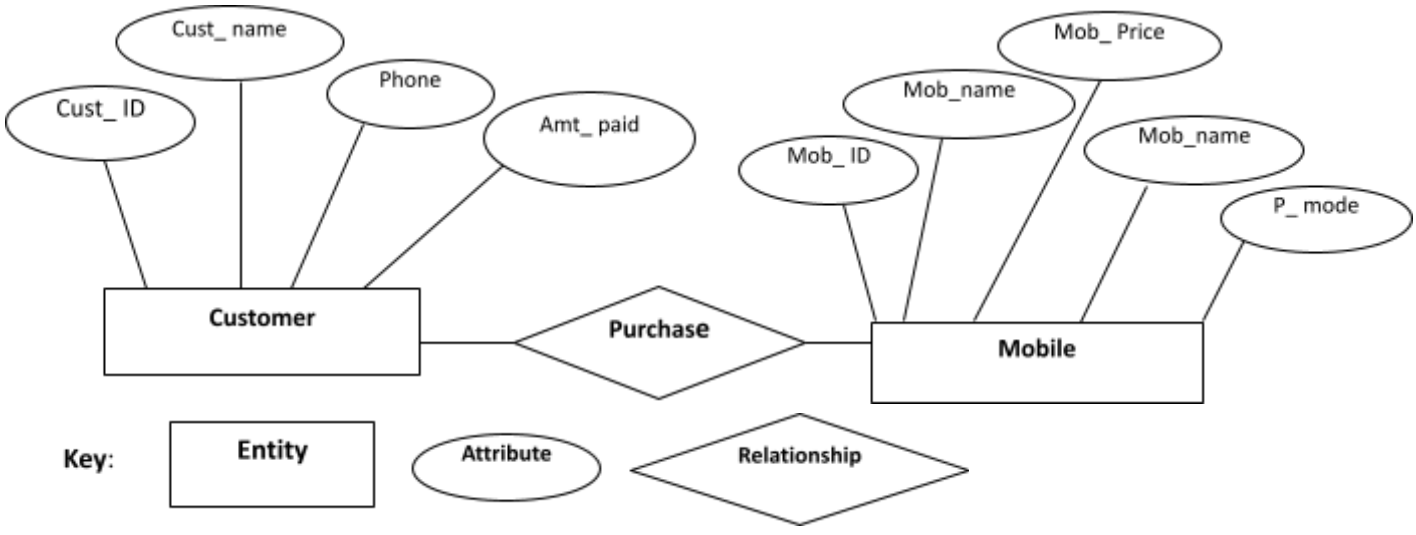
Crow's notation

Figure A

Relationship

Book
Book ID Title Author Reserve Status
Reservation
Reservation ID Book ID Reservation Date Redeemed Date

Figure B



Entity Relationship Model

FGC Minna: Data processing SS2 First Term

Additional features of Entity Model: Connectivity and Cardinality

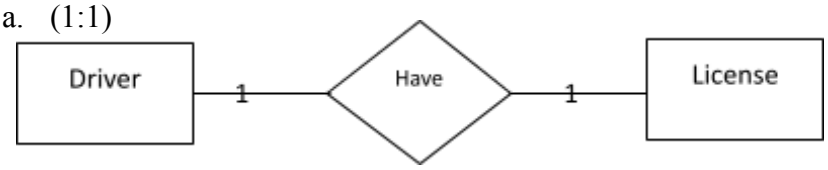
(a) **Connectivity**: It specifies the type of relationship that exists between entities such as:

1. **One-to-One (1:1) relationship:** means each entity is associated with exactly one instance of another entity, such as driver having one license, and each license is being assigned to one driver.
2. **One-to-Many (1:N) relationship:** one entity has a single event, while the other can have multiple occurrences; for example, a customer can place many orders.
3. **Many-to-One (M:1) relationship:** is a type of association between two entities where multiple instances of one entity can relate to a single instance of another entity; for example, Many developers work on a project
4. **Many-to-Many (M:N) relationship:** is a type of association between two entities where each instance of one entity can relate to multiple instances of the other entity; Many tourists visit, and many cities are visited by tourists.

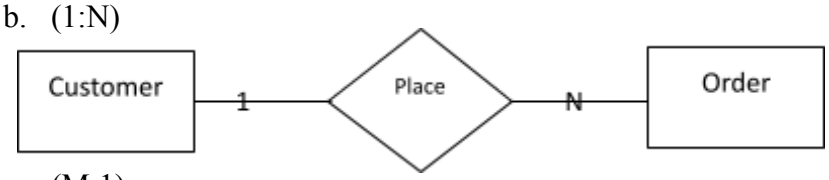
(b) **Cardinality**: Indicates the interactions between entities in times of number expressed as ratios in ERD. Examples using Chen notation and Crow's notation

- **Chen notation:** (1:1), (1:N), (M:1), and (M:N)

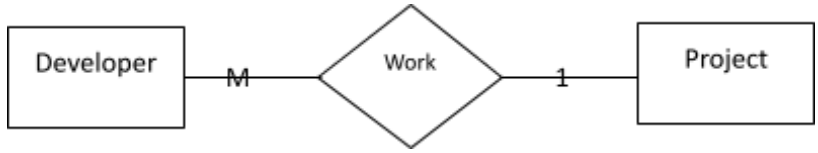
a. (1:1)



b. (1:N)



c. (M:1)

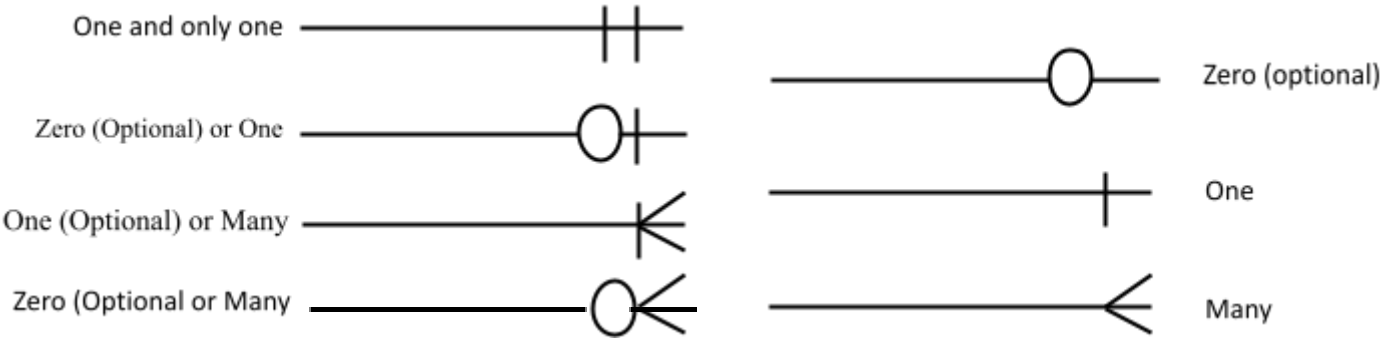


d. (M:N)

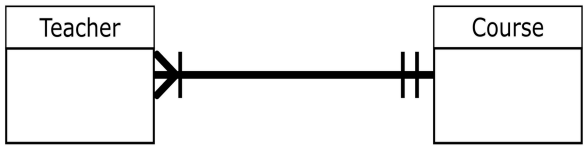
FGC Minna: Data processing SS2 First Term

- **Crow's notation**

Here are the symbols associated with the crow's foot notation:

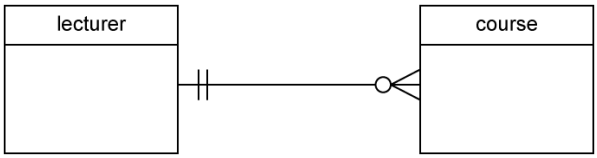


Note: For one entity, a minimum and maximum number define its relationship with another entity.



Second 1 is the minimum in both entities while the first is notation in both is the maximum (Many to one)

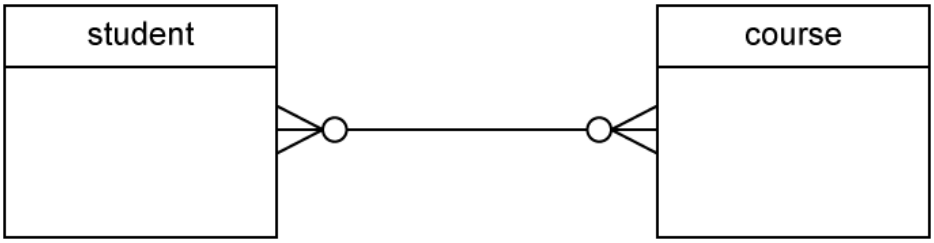
Minimum and maximum of one student can seat vice versa (one to one)



Minimum and maximum of one lecturer and minimum of zero and maximum of many lecturers (one to many)

Assignment

1. Analyze and Interpret the following crow's notation



Relational Model

Creating and Modifying Relation Using SQL

SQL, or Structured Query Language, is a computer language used for storing, manipulating, and retrieving data in relational database tables. It is the standard database language employed by various relational database management systems, including MySQL, MS Access, Oracle, Sybase, Informix, PostgreSQL, and SQL Server.

Creating Relation Using SQL

How to Type SQL Statements in Microsoft Access

1. Launch Microsoft Access, select "More" to open an existing database or click "Blank Database" to create a new one. If creating, type a name in the “File Name” box and click “Create.” (For existing databases, browse to locate the file.)
2. Click “Create” in the top menu.
3. Select the “Query Design” button.
4. Close the “Show Table” dialog box without selecting any tables.
5. Click the “SQL View” or “SQL” button near the top left.
6. If needed, use the dropdown on the button to select “SQL View.”
7. Type your SQL commands in SQL View and click the "Run" button to execute them.

CREATE TABLE

To create a new table, you use the CREATE TABLE statement. Here’s an example:

```
CREATE TABLE employees (
   CREATE TABLE employees (
   employee_id INT,
   first_name VARCHAR (50),
   last_name VARCHAR(50),
   hire_date DATE,
   salary INT,
   PRIMARY KEY (employee_id)
);
```

INSERT (Adding Basic Record)

```
INSERT INTO employees (employee_id, first_name, last_name, hire_date, salary)
VALUES (1, 'John', 'Musa', '2023-01-15', 50000);
```

Modifying Relation Using SQL

Adding a New Column

```
ALTER TABLE employees
```

ADD email VARCHAR(100)

;

Dropping a Column

ALTER TABLE employees

DROP COLUMN email;

Dropping a Table

DROP TABLE employees;

Delete

delete from EMPLOYEE

where last_name = “Musa”;

Update

UPDATE employee

set salary = salary+1000

where first_name='John' and last_name='Musa';

Relational Model

Integrity constraints over relations

Integrity ensures the accuracy and consistency of data in a relational database. Data integrity is maintained through various constraints, with referential integrity being a key concept. Referential integrity ensures that relationships between tables remain valid, typically enforced by foreign key constraints.

Types of integrity constraints over relations

1. **UNIQUE constraint**: constraint specifies that no two records can have the same value in a particular column. They must all be unique.
2. **NOT NULL constraint**: which specifies that a column can't be left blank
3. PRIMARY KEY constraint: constraint defines a unique identification of each record (or row) in a table.
4. **FOREIGN KEY constraint:** uniquely identified rows/records in any other database table.
5. **INDEX constraint:** use to create and retrieve data from the database very quickly.

6. **DEFAULT constraint:** provides a default value for a column when none is specified.
7. **CHECK constraint:** ensures that all values in a column satisfy certain conditions.

Enforcing Integrity Constraints

Enforcing integrity constraints in a database is crucial for maintaining accurate and reliable data. Here are some common types of integrity constraints you might encounter, along with how to enforce them:

1. **PRIMARY KEY Constraint:** Ensures that each row in a table is unique. Primary key must contain unique values. A table can have only one primary key which may consist of single or multiple fields called a **COMPOSITE KEY.**
   Examples:

```
CREATE TABLE employees (
   employee_id INT NOT NULL,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
    PRIMARY KEY (ID)
);
```

2. **Foreign Key Constraint:** Ensures referential integrity between tables.

Department Table

```
CREATE TABLE departments (

   department_id INT NOT NULL,
   department_name VARCHAR(50),
   PRIMARY KEY (department_id)
);
```

Employee Table

```
CREATE TABLE employee (

   employee_id INT NOT NULL,
   first_name VARCHAR(50),
   last_name VARCHAR(50) NOT NULL,
   department_id INT,
   PRIMARY KEY (employee_id)
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

SQL to show how to enforce integrity constraints

Enforcing integrity constraints in SQL often involves defining rules that govern how data can be added, updated, or deleted. The NO ACTION option is commonly associated with foreign key constraints and specifies that if an action would violate the referential integrity of the database, no changes should be made.

Foreign Key with NO ACTION

This ensures that no changes are made to the referencing table if the referenced data would be invalidated.

```
CREATE TABLE departments (
    department_id INT,
    department_name VARCHAR(50),
    PRIMARY KEY (department_id)
);

CREATE TABLE employees (
    employee_id INT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    department_id INT,
    PRIMARY KEY (department_id),
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

In this example:

If you try to delete a department that has employees, the delete will fail (no action is taken) because it would violate the integrity of the employees table.

Other Foreign Key Actions

Besides `NO ACTION`, there are other actions you can use with foreign keys:

Unique Constraint

To enforce uniqueness on a column:

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   email VARCHAR(100) UNIQUE
);
```

Not Null Constraint

To ensure that a column cannot contain NULL values:

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL
);
```

Explains how to use SQL and query a relational data e.g. to create a view statement

Creating a view in Microsoft Access is a great way to simplify your data retrieval. In Access, a view is typically referred to as a "query." Here's how you can create a view using a SQL statement in Access:

Example 1

```
SELECT*
FROM Studentstable;
```

This shows the entire field in the table

Examples 2

```
SELECT StudentsID, StudentsName, Age, Email, PhoneNo

FROM Studentstable

WHERE StudentsID = [Enter StudentsID];
```

This require user to enter students ID to View a student record

Example 3

```
SELECT StudentsID, StudentsName, Age, Email, PhoneNo

FROM Studentstable

WHERE Age = "16";
```

This view student record that are 16 years old

Example 4

```
SELECT StudentsID, StudentsName, Age, Email, PhoneNo

FROM Studentstable

WHERE StudentsID IN (1, 4);
```

File Organization

File Organization: refers to the way data is stored within a storage system. It determines how data is accessed, modified, and managed.

Methods of file organization: Heap, Sequential, Hash, and B-tree files.

Explanation and characteristics of each file structure

Heap Files: Data is stored in no particular order (randomly). Adding new records is quick, but finding specific records takes longer since you might need to scan the entire file.

Sequential Files: Records are stored in a specific, sorted order based on a key field. Searching for records is fast, but adding or removing records can be slow because the order must be maintained.

Hash Files: Each record is stored using a special formula called a hash function. When searching for a record is using a key (like an ID), the hash function converts that ID into a number indicating where to find the record. If two different IDs produce the same number (collision), it needs to be handled properly.

FGC Minna: Data processing SS2 First Term

B-Tree Files: Use a tree structure to keep data organized, allowing for fast searches and easy access to ranges of records. Because of this structure, you can also find a group of records (like all students in a certain age range) quickly and efficiently.

Take home Assignment

1. *List and briefly explain the ways file can be organized in storage medium*
2. *What is the medium and file access methods of the following storage media: hard disk, solid state drive, cassette, cd/dvd, memory card, pen drive, floppy disk*