DAMG6210 - Data Management and Database Design

Homework 04

1. **What is the purpose of logical database design?**

Logical database design is a critical phase in the database development process that focuses on defining the structure and relationships of data elements within a database system. It serves as a blueprint for how data will be organized and accessed, ensuring efficient and effective data management.

Key Objectives of Logical Database Design:
Structure and Organization:
- **Data Element Definition:** Clearly define the essential data elements (attributes) to be stored and managed within the database.
- **Relationship Modeling:** Establish meaningful connections and dependencies between data elements using an Entity-Relationship (ER) model.
- **DBMS Compatibility:** Ensure the design aligns with the capabilities and requirements of the chosen Database Management System (DBMS).

Data Independence:
- **Separation of Concerns:** Separate the logical data structure from its physical storage to allow for changes without affecting applications.
- **Flexibility:** Enable adaptability to evolving business needs while maintaining data integrity.

Integrity and Constraints:
- **Data Quality:** Implement integrity constraints to ensure data accuracy and consistency.
- **Primary Keys:** Use unique identifiers for each record.
- **Foreign Keys:** Establish relationships between entities through references to primary keys.
- **Unique Constraints:** Enforce unique values for specific attributes.
- **Data Validation:** Prevent data entry errors, inconsistencies, and anomalies to maintain data quality.

2. List and briefly describe the five properties of relations (tables) in a relational database.

Five Properties of Relations in a Relational Database:

1. Unique Name: Each relation (or table) within a database must have a distinct and unique name. This helps to identify and reference specific tables within the database schema.

2. Atomicity: Each entry at the intersection of a row and column must be atomic or single-valued. This means that there can only be one value associated with each attribute on a specific row. Multivalued attributes are not allowed in a relational database.
3. Row Uniqueness: Every row in a relation must be unique. No two rows can have identical values for all their attributes. This ensures that each record represents a distinct entity.
4. Unique Attribute Names: Each attribute (or column) within a table must have a unique name. This avoids confusion and ambiguity when referencing specific data elements.
5. Order Independence: The sequence of columns (left to right) and rows (top to bottom) is insignificant. The order of these elements can be changed without affecting the meaning or use of the relation. This flexibility allows for different sorting and presentation options without altering the underlying data structure.

3. Explain a property of candidate keys that make them suitable for unique identification.

Candidate keys are a fundamental concept in relational database design. They serve as unique identifiers for individual rows within a table. This property of uniqueness is crucial for ensuring data integrity and efficient data retrieval.

Uniqueness Property:

- A candidate key must guarantee that no two rows in a table have the same values for all attributes within the key. This ensures that each row can be uniquely identified based on its combination of values in the candidate key.
- *Example*: Consider a table named Employees with the following attributes: EmployeeID, FirstName, LastName, and Department. EmployeeID is likely a suitable candidate key for this table. If each employee is assigned a unique employee ID, no two employees will have the same value for this attribute. Therefore, EmployeeID can uniquely identify each row in the Employees table.

Why Uniqueness is Important:

- Data Integrity: Uniqueness prevents duplicate records, ensuring that each row represents a distinct entity.
- Efficient Data Retrieval: When searching for a specific row, the candidate key can be used as an efficient index to locate the row quickly.
- Relationships: Candidate keys are often used as foreign keys in other tables to establish relationships between entities.

▪ Multiple Candidate Keys: A table can have multiple candidate keys. For example, if no two employees have the same combination of FirstName and LastName, then FirstName and LastName together could also be a candidate key.

- Primary Key: One of the candidate keys is typically chosen as the primary key. The primary key is the primary identifier for the table and is often used as a foreign key in other tables.

4. What is the role of referential integrity constraints in the relational model, and how do they maintain data integrity?

Referential Integrity–rule states that any foreign key value (on the relation of the many side) MUST match a primary key value in the relation of the one side. (Or the foreign key can be null).

Role of Referential Integrity Constraints:
Referential integrity constraints ensure consistency and data accuracy between related tables in a relational database. They establish and maintain relationships between tables by enforcing rules that govern how data can be inserted, updated, and deleted.

How referential integrity constraints maintain data integrity:

1. *Foreign Key-Primary Key Relationship:* Referential integrity constraints define a relationship between a foreign key in one table and a primary key in another table. This ensures that data in the foreign key table references valid data in the primary key table.
2. *Data Consistency:* By enforcing this relationship, referential integrity constraints prevent inconsistent data. For example, you cannot insert a new order without a corresponding customer in the customer table.
3. *Data Validation:* They validate data during updates and deletions, preventing actions that would violate the defined relationship. For instance, you cannot delete a customer if there are existing orders associated with that customer.
4. *Data Accuracy:* Referential integrity constraints help maintain data accuracy by ensuring that related data is consistent and valid.
5. *Data Protection:* They protect against accidental data loss or corruption by preventing actions that could lead to orphaned or inconsistent data.

Example:
Consider an Orders table and a Customers table. The Orders table might have a CustomerID foreign key that references the CustomerID primary key in the Customers table. This ensures that every order is associated with a valid customer.
Delete Rules:

- Restrict: Prevents the deletion of a customer if there are existing orders associated with them.
- Cascade: Automatically deletes associated orders when a customer is deleted.
- Set-to-Null: Sets the CustomerID foreign key in the Orders table to null if the corresponding customer is deleted.

5. Explain the concept of multi-valued attributes in logical database design. Provide an example to illustrate the concept.

Multivalued attributes are attributes that can have multiple values for a single entity. For example, an employee might have multiple skills, or a product might be available in multiple colors.

Handling Multivalued Attributes:

When dealing with multivalued attributes in logical database design, it's generally recommended to create a separate relation to store the multivalued data. This approach provides better flexibility, normalization, and data management capabilities.

To effectively handle multivalued attributes in logical database design, follow these steps:

1. Create a New Relation: Create a new relation that specifically stores the multivalued attribute.
2. Include a Foreign Key: Include a foreign key in the new relation that references the primary key of the original entity. This establishes a one-to-many relationship between the two relations.
3. Define Primary Key: The primary key of the new relation should typically consist of the foreign key (from the original entity) and the multivalued attribute itself.

Example:

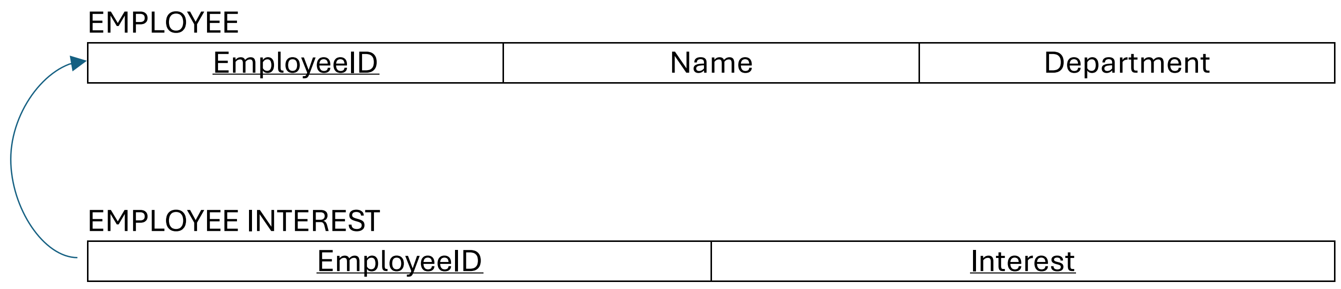
Original Entity: Employee

- EmployeeID (primary key)
- Name
- Department
- Interest (multivalued attribute)

New Relations:

1. **Employee Table:**
    - EmployeeID (primary key)
    - Name
    - Department
    - Interest
2. **Employee Interest Table:**
    - EmployeeID (foreign key, referencing EmployeeID from Employee table)
    - Interest (primary key, along with EmployeeID)

EMPLOYEE
EmployeeID
Name
Department
Interest



6. What are data integrity constraints, and why are they essential in a database?

Data integrity constraints are rules that enforce data accuracy and consistency within a relational database. They help prevent errors, maintain data quality, and ensure the reliability of information stored in the database.

There are several types of data integrity constraints, including:

1. **Domain Constraints:** These constraints define the allowable values for each attribute. They specify the data type, size, and range of acceptable values. For example, a "Date of Birth" attribute might have a domain constraint requiring a valid date format.
2. **Entity Integrity:** This constraint ensures that every relation (table) has a primary key and that the primary key values are not null. The primary key uniquely identifies each row in the table.
3. **Referential Integrity:** This constraint maintains consistency between related tables by enforcing rules about foreign keys. It ensures that foreign key values in one table reference valid primary key values in another table. This prevents orphaned or inconsistent data.

Data integrity constraints are essential in a database because:

- **Data Accuracy:** Constraints help prevent incorrect or invalid data from being entered into the database.
- **Data Consistency:** They ensure that data is consistent across related tables, preventing inconsistencies and anomalies.
- **Data Reliability:** By maintaining data integrity, constraints contribute to the overall reliability and trustworthiness of the database.
- **Error Prevention:** Constraints can help prevent common errors like entering invalid data or deleting related data that should be preserved.
- **Data Quality:** They contribute to improving data quality by ensuring that data is accurate, consistent, and complete.

7. What the difference between primary key and unique key constraints.

Feature	Primary Key Constraint	Unique Key Constraint

Purpose	Uniquely identifies each record in a table	Ensures uniqueness within specified column(s) or sets of columns
Number of Columns	Only one primary key constraint per table, can consist of one or multiple columns	Multiple unique key constraints can exist in a table; each unique key can consist of one or multiple columns
Automatic Indexing	Typically creates a clustered index by default	Typically creates a non-clustered index by default (may vary depending on database system)
Null Values	Cannot contain null values; every row must have a value for the primary key column(s)	Can allow one null value; except in multi-column unique keys where the combination of null values must be unique
Relationships	Often used as a reference in foreign key constraints to establish relationships between tables	Typically, not used as a reference in foreign key constraints, but can be used in other scenarios
Data Modification	Updating or deleting a primary key value can affect related data (e.g., foreign key constraints)	key value can affect related data (e.g., foreign key constraints) Updating or deleting a unique key value does not directly affect related data
Example	Order ID in an Orders table, uniquely identifying each order	Email in a Customers table, ensuring that each customer has a unique email address

8. What is a foreign key constraint, and how does it maintain referential integrity in a relational database?

Foreign key constraints are rules that maintain consistency and data accuracy between related tables in a relational database. They establish relationships between tables by defining a connection between a foreign key in one table and a primary key in another table.

How Foreign Key Constraints Maintain Referential Integrity:

1. ***Data Validation:*** Foreign key constraints ensure that data inserted or updated in a table references valid data in another table. This prevents inconsistencies and errors.
2. ***Relationship Enforcement:*** They establish and enforce relationships between tables, ensuring that data in one table is consistent with related data in another.
3. ***Data Protection:*** Foreign key constraints help protect against accidental data loss or corruption by preventing actions that could lead to orphaned or inconsistent data.

Example:

Consider two tables: Orders and Customers. The Orders table has a CustomerID foreign key that references the CustomerID primary key in the Customers table. This establishes a one-to-many relationship between the two tables, where an order must be associated with a valid customer.

- Data Validation: When inserting a new order, the database system will check if the specified CustomerID exists in the Customers table. If it doesn't, the insertion will be rejected, preventing an inconsistent order.
- Relationship Enforcement: If a customer is deleted from the Customers table, any orders associated with that customer will be affected based on the defined referential integrity rules. Common options include:
    - Restrict: Prohibiting the deletion of the customer if there are associated orders.
    - Cascade: Automatically deleting the associated orders when the customer is deleted.
    - Set Null: Setting the CustomerID foreign key in the Orders table to null if the customer is deleted.