**COMP 5120/6120 Database Systems I**

**Spring 2024 Homework #2**

**Due: 2/28/2024**

1. What is a foreign key constraint? Why are such constraints important? What is referential integrity? (10 pts) A foreign key constraint is a rule in a relational database that establishes a relationship between two tables based on the values of their respective columns. It ensures that the values in a column or set of columns (the foreign key) in one table correspond to the values in another table's column (the primary key). The reason that constraints are important because Data Integrity: Foreign key constraints ensure the integrity and accuracy of data within the database. They enforce referential integrity, which means that values in the foreign key column(s) must match existing values in the primary key column(s). This prevents orphaned records, where a foreign key references a non-existent primary key, or inconsistent relationships between tables. Relationship Enforcement: Foreign key constraints establish and enforce relationships between tables. They define dependencies between data in different tables, ensuring that related data is maintained consistently. For example, if there is a foreign key relationship between an "Orders" table and a "Customers" table, the constraint ensures that an order can only be associated with an existing customer. Data Consistency: By enforcing referential integrity, foreign key constraints help maintain consistency in the database. They prevent operations that would result in data inconsistencies, such as deleting a primary key value while there are still related foreign key values, or updating a primary key without updating the corresponding foreign key values. Referential integrity is the concept that ensures the consistency and correctness of relationships between tables in a relational database. It is maintained through the use of foreign key constraints. Referential integrity guarantees that any action performed on the database, such as insertion, update, or deletion, will not violate the defined relationships between tables. It ensures that data remains accurate, valid, and consistent across different tables, promoting data integrity and reliable database operations.
2. Explain the difference between external, internal, and conceptual schemas. How are these different schema layers related to the concepts of logical and physical data independence? (10 pts)

The three schema layers - external, conceptual, and internal - represent different views or perspectives of a database system. Each layer focuses on different aspects of the database and serves distinct purposes. Logical and physical data independence is related to the ability to modify these schema layers without affecting the other layers.

**1. External Schema:**

- Also known as user schema or view schema.

- It represents the specific view of the data that is relevant to a particular user or group of users.

- It defines the portion of the database that is visible and accessible to the users.

- The external schema provides a logical and customized view of the data, hiding unnecessary details and presenting the data in a way that is most useful for the users.

- Changes made to the external schema should not impact the conceptual or internal schemas, ensuring logical data independence.

**2. Conceptual Schema:**

- It represents the overall logical structure and organization of the entire database.

- It defines the entities, their attributes, and the relationships between them in an abstract and conceptual manner.

- The conceptual schema provides a high-level view of the database, independent of any specific database management system.

- It acts as a bridge between the external and internal schemas, providing a logical representation of the data.

- Modifications to the conceptual schema should not require changes to the external or internal schemas, ensuring logical data independence.

**3. Internal Schema:**

- Also known as physical schema.

- It describes how the data is physically stored and organized in the storage medium of the database system.

- It includes details such as storage structures, file organizations, indexing methods, and access paths.

- The internal schema deals with the physical implementation of the database, considering aspects such as disk space allocation, data compression, and data encryption.

- Changes made to the internal schema should not affect the conceptual or external schemas, ensuring physical data independence.

**Logical Data Independence:**

- Logical data independence refers to the ability to modify the conceptual schema without impacting the external schemas or application programs.

- If a change is made to the conceptual schema, such as adding or modifying entities, attributes, or relationships, it should not require rewriting or modifying the external schemas.

- Logical data independence allows for the evolution of the database system without affecting the applications that rely on it.

**Physical Data Independence:**

- Physical data independence refers to the ability to modify the internal schema without affecting the conceptual or external schemas.

- If changes are made to the internal schema, such as reorganizing the storage structure or changing indexing methods, they should not require modifying the conceptual or external schemas.

- Physical data independence allows for efficient storage management and performance improvements without impacting the logical structure or the applications using the database.

In summary, the external schema provides a user-specific view, the conceptual schema represents the overall logical structure, and the internal schema deals with the physical storage details. Logical and physical data independence ensures that modifications to one schema layer can be made without affecting the other layers, allowing for flexibility, maintenance, and evolution of the database system.

1. Consider the following schema:

Suppliers (sid: integer, sname: string, address: string)

Parts (pid: integer, pname: string, color: string)

Catalog (sid: integer, pid: integer, cost: real)

The Catalog relation lists the prices charged for parts by suppliers. Write the following queries in **SQL** (40 pts):

* 1. Find the pnames of parts for which there is some supplier.

SELECT DISTINCT pname

FROM Parts

INNER JOIN Catalog ON Parts.pid = Catalog.pid;

* 1. For each part, find the sname of the supplier who charges the most for that part.

SELECT Parts.pname, Suppliers.sname

FROM Parts

INNER JOIN Catalog ON Parts.pid = Catalog.pid

INNER JOIN Suppliers ON Catalog.sid = Suppliers.sid

WHERE Catalog.cost = (

SELECT MAX(cost)

FROM Catalog

WHERE Catalog.pid = Parts.pid

);

* 1. Find the sids of suppliers who supply only red parts.

SELECT DISTINCT sid

FROM Catalog

WHERE pid IN (

SELECT pid

FROM Parts

WHERE color = ‘red’

)

GROUP BY sid

HAVING COUNT(DISTINCT pid) = (

SELECT COUNT(\*)

FROM Parts

WHERE color = ‘red’

);

* 1. Find the snames of suppliers who supply every part.

SELECT sname

FROM Suppliers

WHERE NOT EXISTS (

SELECT pid

FROM Parts

WHERE NOT EXISTS (

SELECT sid

FROM Catalog

WHERE Catalog.sid = Suppliers.sid AND Catalog.pid = Parts.pid

)

);

1. Consider the following schema:

Employee (person-name, street, city)

Works (person-name, company-name, salary)

Company (company-name, city)

Manages (person-name, manager-name)

Write the following queries in **relational algebra** (40 pts):

* + 1. Find the names of all employees who work for Auburn Bank.

π person-name (σ company-name = 'Auburn Bank' (Works))

π person-name (σ company-name = 'Auburn Bank' (Works))

* + 1. Find the names and cities of residence of all employees who work for Auburn Bank.

π person-name, city (σ company-name = 'Auburn Bank' (Works ⋈ Employee))

* + 1. Find the names, street address, and cities of residence of all employees who work for Auburn Bank and earn more than $50,000 per year.

π person-name, street, city (σ company-name = 'Auburn Bank' ∧ salary > 50000 (Works ⋈ Employee))

* + 1. Find the names of all employees in this database who live in the same city as the company for which they work.

π person-name (Works ⋈ Company) ⋈ Employee where city = Company.city