Name:Mohab Yousef
Id:904154154
mey0012

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 database constraint used to establish a link between the data in two tables. It is a field (or collection of fields) in one table (values in field of tuples) that refers to the primary key of another table (values in field of tuples). The table containing the foreign key is called the child table, and the table containing the primary key is called the referenced or parent table.
   Foreign key constraints are important for several reasons:
   Referential Integrity: They ensure that the relationship between two tables remains synchronized during updates and deletes, preventing the existence of orphaned records in the child table.
   Data Consistency and Accuracy: By linking tables together, foreign key constraints help maintain the accuracy and consistency of the data by ensuring that only valid values are inserted or updated in the foreign key column(s).
   Data Integrity: They help avoid anomalies and protect the integrity of the dataset by preventing users from entering incorrect or inconsistent data.
   Relationship Representation: Foreign keys represent relationships between different entities in a database and hence provide a mechanism to navigate between different instances of entities.
   Referential integrity is a property of data stating that all its references are valid. It is a fundamental aspect of database integrity and is enforced through foreign key constraints. Referential integrity ensures that relationships between tables remain consistent, i.e., if a row in table A references a row in table B, then that row must exist in table B. That is, they enforce foreign keys.

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)

   External Schema: data access customization or User-level view of the data, focused on presenting data in a user-friendly manner.
   Conceptual Schema: Overall logical representation of the data, defining entities, relationships, and constraints stored in the database.
   Internal Schema: Physical representation of the data, focused on storage, organization, and management of data on the storage medium.

   The three schema layers—external, conceptual, and internal—are integral to achieving logical and physical data independence in a database system. Logical

data independence is realized through the separation of the external and conceptual schemas, allowing changes in the database's logical structure without affecting user views or interactions. Conversely, physical data independence is achieved by isolating the conceptual and internal schemas, enabling modifications to the physical storage and retrieval mechanisms without altering the logical representation of the data. Together, these layers and concepts facilitate flexibility and robustness in a database system, allowing alterations in data structures and storage without impacting user interfaces or application integrations.

3. 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):

   a.) Find the pnames of parts for which there is some supplier.

```
SELECT DISTINCT P.pname
FROM Parts P, Catalog C
WHERE P.pid = C.pid;
```

   b.) For each part, find the sname of the supplier who charges the most for that part.

```
SELECT P.pid, S.sname
FROM Parts P, Suppliers S, Catalog C
WHERE C.pid = P.pid
AND C.sid = S.sid
AND C.cost = (SELECT MAX (C1.cost)
FROM Catalog C1
WHERE C1.pid = P.pid);
```

   c.) Find the sids of suppliers who supply only red parts.

```
SELECT DISTINCT C.sid
FROM Catalog C
WHERE NOT EXISTS (SELECT *
```

```
FROM Parts P
WHERE P.pid = C.pid AND P.color <> 'Red' );
```

d.) Find the snames of suppliers who supply every part.

```
SELECT S.sname
FROM Suppliers S
WHERE NOT EXISTS (( SELECT P.pid
FROM Parts P )
EXCEPT
( SELECT C.pid
FROM Catalog C
WHERE C.sid = S.sid ));
```

4. 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):

a.) Find the names of all employees who work for Auburn Bank.

$\pi_{\text{person-name}}(\sigma_{\text{company-name='Auburn Bank'}}(\text{Works}))$

b.) Find the names and cities of residence of all employees who work for Auburn Bank.

$\pi_{\text{person-name, city}}((\sigma_{\text{company-name='Auburn Bank'}}(\text{Works}))\bowtie\text{Employee})$

c.) 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.

$\pi_{\text{person-name, street, city}}((\sigma_{\text{company-name='Auburn Bank' ^ salary>50000}}(\text{Works}))\bowtie\text{Employee})$

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

$\pi_{\text{person-name}}(\sigma_{\text{Employee.city = Company.city}}(\text{Employee}\bowtie\text{Works}\bowtie\text{Company}))$
or
$\pi_{\text{person-name}}(\text{Employee}\bowtie\text{Works}\bowtie\text{Company})$