Sai University

Data Base Management Systems

Mid-Semester Examination with Solutions, Semester A, 2023–2024

Date : 4 October, 2023 Marks : 20
Duration : 90 minutes Weightage : 20%

1. Let r be a relation with attribute set R and s be a relation with attribute set S such that $S \subseteq R$. We can define a division operation in the relational algebra as follows: $r \div s$ is the largest relation q such that $q \times s \subseteq r$.

   (a) Let $r(A, B, X, Y)$ and $s(X, Y)$ be the relations displayed below. Compute $r \div s$.

r			
A	B	X	Y
a	b	w	x
c	d	y	z
a	b	y	z
e	f	w	x
c	d	w	x

s	
X	Y
y	z
w	x

(2 marks)

Solution Clearly q should be a relation over $A \times B$ for $q \times s$ to be a subset of r. We need to find all tuples $(\alpha, \beta) \in A \times B$ so that $\{(\alpha, \beta)\} \times s$ lies inside r. Since $s = \{(y, z), (w, x)\}$, this means that $\{(\alpha, \beta, y, z), (\alpha, \beta, w, x)\}$ should be in r.

The only candidate tuples for q are the projections of r onto A, B, which are (a, b), (c, d) and (e, f). Of these, we have $\{(a, b, y, z), (a, b, w, x)\}$ and $\{(c, d, y, z), (c, d, w, x)\}$ in r but $(e, f, y, z) \notin r$, so $r \div s = \{(a, b), (c, d)\}$.

(b) Show that

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

where $A - B$ denotes set difference (the set of elements in A that are not in B).

(4 marks)

Solution $\Pi_{R-S}(r)$ is the projection of r onto the columns $R - S$. This is the set of all tuples that could possibly be in q. We have to eliminate those tuples which do not combine with all tuples in s.

$(\Pi_{R-S}(r) \times s)$ is the set of all tuples created by combining the projection of r on $R - S$ with s. $\Pi_{R-S,S}(r)$ is just r with the columns rearranged. Hence $(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ is the set of tuples from the cross product that are not actually in r. These are the tuples whose $R - S$ projection should be excluded from q.

So we remove from $\Pi_{R-S}(r)$ the $R - S$ projection of $(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$, which is the expression above.

2. An election database includes the following two tables:

Candidates(CandidateID,CandidateName)
Votes(CandidateId,PollingBoothID,NumberOfVotes)

The second table is updated booth by booth. Whenever counting is completed in a polling booth, an entry is added to the second table for each candidate who has got a nonzero number of votes, listing out the number of votes he got at that booth. For candidates who poll zero votes at a booth, no entry is recorded in *Votes*.

We need to periodically compute the current status of *all* candidates by "joining" these two tables to create a table of the form

Status(CandidateID,CandidateName,TotalVotesTillNow)

(a) Explain why a natural join is not adequate for this purpose. *(2 marks)*

Solution We have to join the two tables on *CandidateID*. If a particular candidate has not polled any votes in any of the booths that have been counted so far, that candidate's ID will not be present in the relation *Votes*, and hence will not appear in the joined relation *Status*. Our requirement is that such a candidate should appear in *Status* with a vote count of zero, so a natural join will not suffice.

(b) What kind of join should we use to ensure that *every* candidate is included in the status table, including those who have yet to poll any votes across all booths? *(2 marks)*

Solution We need an outer join — specifically, a natural left outer join — to ensure that all candidates are reported in the final table *Status*.

(c) Write an SQL query to compute the *Status* table. *(4 marks)*

Solution We need to group and aggregate the votes in *Votes* by *CandidateID* and then compute a natural left outer join with the table *Candidates*. Here are two versions of the query.

(i)
```
with CurrentVotes(CandidateID,TotalVotesTillNow) as
    (select CandidateID, sum(NumberOfVotes)
     from Votes
     group by CandidateID)
select CandidateID,CandidateName,TotalVotesTillNow from
    Candidates natural left outer join CurrentVotes
```

(ii)
```
select CandidateID,CandidateName,TotalVotesTillNow from
    Candidates natural left outer join
    (select CandidateID, sum(NumberOfVotes)
     from Votes
     group by CandidateID)
     as CurrentVotes(CandidateID,TotalVotesTillNow)
```

To fully solve the problem, we should create a view `Status` to store the output of the query.

3. A database of movies currently showing in the city has a table of the form *Playing(Title, Theatre, Location)*, where *Title* is the name of the movie, *Theatre* is the name of the cinema theatre and *Location* is the place where the theatre is located (a mall or multiplex).

   We assume that no pair of movies currently running have the same title and no two cinema theatres have the same name. A cinema theatre may be showing different movies in different time slots, but no location has two cinema theatres showing the same movie.

   (a) What functional dependencies can you infer from these constraints? *(2 marks)*

   Solution Since cinema theatres have distinct names, from the theatre name we can uniquely identify the location. Since no movie plays at two theatres in the same location, from the movie name and the location we can uniquely identify the theatre where it is playing. Hence we have two functional dependencies.

   - *Theatre* $\rightarrow$ *Location*
   - *Title, Location* $\rightarrow$ *Theatre*

   (b) Compute a BCNF decomposition of the table *Playing*. *(2 marks)*

   Solution We have a functional dependency *Theatre* $\rightarrow$ *Location* where the left hand side, *Theatre* is not a superkey for *Playing*, so this relation is not in BCNF. The BCNF decomposition gives two relations

   - *R(Theatre, Location)*
   - *S(Title, Theatre)*

   (c) Explain whether your BCNF decomposition is dependency preserving. *(2 marks)*

   Solution The functional dependency *Title, Location* $\rightarrow$ *Theatre* cannot be checked locally in either relation R or relation S resulting from the BCNF decomposition. Hence this decomposition is not dependency preserving.
