Assignment 2

CMPUT 391 (Winter 2003)

Due Date (in class): Wednesday February 12, 2003 in class.
Percentage overall grade: 3%
Penalties: 20% off a day for late assignments
Maximum Marks: 100

Write clearly your Name, Student Number and Lab Number on the front page of your assignment.

Deliverables:
The answers of the following questions 1, 2, 3, and 4 clearly typed on paper.
The TA will take into account the cleanliness of what was handed in. It is your responsibility to make your assignment readable.

Question 1: (Armstrong Axioms) [10%]

There are inference rules for functional dependencies. Three of those rules are known as the Armstrong's Axioms: Reflexivity, Augmentation and transitivity. These axioms are sound and complete. We saw in class two other rules that are inferred from these axioms. These derived rules are known as the decomposition rule and the union rule.
There is yet another inference rule called pseudotransitive rule that stipulates that:
if X → Y and WY → Z then WX → Z
Prove this rule using the known axioms.

Solution:
1- X → Y (given)
2- WY → Z (given)
3- WX → WY (augmentation on 1.)
4- WX → Z (transitivity on 3. and 2.)

Question 2: (Functional Dependencies) [25%]

1- Consider the following relation schema and set of functional dependancies:
Emp-Dept (SIN, E_Name, B_Date, Address, D_Num, D_Name, D_Manager)
F={SIN→{E_Name, B_Date, Address, D_Num},
D_Num → {D_Name, D_Manager}
}
Calculate the closure of {SIN}⁺ and {D_Num}⁺ with respect to F.
Solution:
Using the algorithm seen in class:

\( \{SIN\}^+ = \{SIN, E_{Name}, B_{Date}, \text{Address}, \text{D}_\text{Num}, \text{D}_{Name}, \text{D}_{Manager}\} \)

\( \{\text{D}_\text{Num}\}^+ = \{\text{D}_\text{Num}, \text{D}_{Name}, \text{D}_{Manager}\} \)

2- Is the set of functional dependencies F minimal? If not, try to find a minimal set of functional dependencies that is equivalent to F (minimal cover). Prove the equivalence.

Solution:
No, the set of functional dependencies F is not minimal since the right-hand side of the rules have more than just one attribute.

The minimal cover G of F is:

- \( \text{SIN} \rightarrow \text{E}_{Name} \)
- \( \text{SIN} \rightarrow \text{B}_{Date} \)
- \( \text{SIN} \rightarrow \text{Address} \)
- \( \text{SIN} \rightarrow \text{D}_\text{Num} \)
- \( \text{D}_\text{Num} \rightarrow \text{D}_{Name} \)
- \( \text{D}_\text{Num} \rightarrow \text{D}_{Manager} \)

To prove that two sets of functional dependencies F and E are equivalent, we either show that \( F^+ = E^+ \) or that E covers F and F covers E.

To show that F covers E, we calculate \( X^+ \) with respect to F for every FD \( X \rightarrow Y \) in E and check whether \( X^+ \) includes the attributes in \( Y \).

Rather than calculating \( G^+ \) and \( F^+ \) we show that the coverage of G and F.

**F covers G**

\( \{\text{SIN}\}^+ = \{\text{SIN, E}_{Name, B}_{Date, Address, D}_{Num, D}_{Name, D}_{Manager}\} \) and \( \{\text{D}_\text{Num}\}^+ = \{\text{D}_\text{Num, D}_{Name, D}_{Manager}\} \) with respect to F (see 2.1). All right-hand side of any FD in G is included.

**G covers F**

\( \{\text{SIN}\}^+ = \{\text{SIN, E}_{Name, B}_{Date, Address, D}_{Num, D}_{Name, D}_{Manager}\} \) and \( \{\text{D}_\text{Num}\}^+ = \{\text{D}_\text{Num, D}_{Name, D}_{Manager}\} \) with respect to G and all right-hand side of any FD in F is included.

3- What update anomalies can happen to Emp-Dept? Give examples.

Solution:

**Insert anomaly**: Adding a new department and new manager, we would need employees in the department first.

**Delete anomaly**: Removing the only employee in a department would remove the department and its manager.

**Update anomaly**: Changing the manager of a populous department would need to update many tuples.

**Question 3:** (Functional Dependencies) [25%]

Consider the relation \( R = \{A, B, C, D, E, F, G, H, I, J\} \) and the set of functional dependencies \( F=\{\{A, B\} \rightarrow C, \)

\( A \rightarrow \{D, E\}, \)

...
B \rightarrow F,
F \rightarrow \{G, H\},
D \rightarrow \{I, J\}

1- What is the key for R? Demonstrate it using the inference rules.
Decompose R into 2NF, then 3NF relations.

Solution:
A \rightarrow DE \text{ (given) } \Rightarrow A \rightarrow D \text{ and } A \rightarrow D
Since A \rightarrow D \text{ and } D \rightarrow IJ \text{ (given) } \Rightarrow A \rightarrow IJ
Using the union rule A \rightarrow ADEIJ, thus AB \rightarrow ABDEIJ \text{ (augmentation)}
Also AB \rightarrow C \text{ (given) } \Rightarrow AB \rightarrow ABCDEIJ.
Since B \rightarrow F \text{ (given) and } F \rightarrow GH \text{ (given), } B \rightarrow GH \text{ (transitivity)}
Thus AB \rightarrow AGH \text{ holds. Also } AB \rightarrow AF \text{ holds from } B \rightarrow F \text{ (given)}
Finally, using the union rule AB \rightarrow ABCDEFGHIJ.
So AB is a key. This can also be determined by calculating AB^+ \text{ with respect to the set } F.

<table>
<thead>
<tr>
<th>2NF</th>
<th>3NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 (A, B, C)</td>
<td>R1 (A, B, C)</td>
</tr>
<tr>
<td>R2 (A, D, E, I, J)</td>
<td>R2.1 (A, D, E) R2.2 (D, I, J)</td>
</tr>
<tr>
<td>R3 (B, F, G, H)</td>
<td>R3.1 (B, F) R3.2 (F, G, H)</td>
</tr>
</tbody>
</table>

2- What is the key for R if F = \{ \{A, B\} \rightarrow C, 
{B, D} \rightarrow \{E, F\}, 
{A, D} \rightarrow \{G, H\}, 
A \rightarrow I, 
H \rightarrow J \}?

Demonstrate it using inference rules. Decompose R in 2NF then 3NF in this case.

Solution:
AD \rightarrow GH \text{ (given) } \Rightarrow ABD \rightarrow ABDGH \text{ (augmentation and reflexivity)}
Since A \rightarrow I \text{ (given) then } ABD \rightarrow ABDI \text{ (augmentation and reflexivity)}
AB \rightarrow C \text{ (given) } \Rightarrow ABD \rightarrow ABCD
BD \rightarrow EF \text{ (given) } \Rightarrow ABD \rightarrow ABDEF
AD \rightarrow GH \text{ (given) } \Rightarrow AD \rightarrow H. \text{ Since } H \rightarrow J \text{ then } AD \rightarrow J. \text{ Thus, } ABD \rightarrow ABDJ.
Finally, using the union rule ABD \rightarrow ABCDEFGHIJ.
So ABD is a key. This can also be determined by calculating ABD^+ \text{ with respect to the set } F.

<table>
<thead>
<tr>
<th>2NF</th>
<th>3NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 (A, B, C)</td>
<td>R1 (A, B, C)</td>
</tr>
<tr>
<td>R2 (B, D, E, F)</td>
<td>R2 (B, D, E, F)</td>
</tr>
<tr>
<td>R3.2 (H, J)</td>
<td></td>
</tr>
</tbody>
</table>
**Question 4: (Query Optimization) [40%]**

Consider the following query Q:

```
SELECT E-Name, Salary
FROM Employee, Works, Project
WHERE P-Type=”design” AND P-Num=PNO AND ESIN=SIN AND B-Date > “1961-01-29"
```

On the following tables:

- Employee(SIN, E-Name, B-Date, Address, Sex, Salary, Supervisor) with 10,000 tuples;
- Works(ESIN, PNO) with 20,000 tuples;
- Project(P-Name, P-Type, P-Num, Location, D-Num) with 500 tuples.

Knowing that one page can accommodate 100 tuples of Employee, 400 tuples of Works, or 120 tuples of Project, and assuming that we have 6 buffers in main memory calculate the cost for evaluating Q if we choose Bloc-Nested Loop joins or Sort-merge joins for both of the two joins, or Bloc-Nested Loop for the first join and Sort-Merge for the second join. The first join is between Project and Works while the second joins the result with Employee. Assume that the same number of tuples of the result of the first join can fit per page as we can fit Project tuples (120). Which plan would be the best? Assume that there are 5 types of projects and ¾ of the employees are born after January 29, 1961. All distributions are uniform. Push selections as early as possible in all cases. Draw your query plans.

**Solution:**

Employee has 10,000 tuples, 100 per page \( \Rightarrow \) there are 10000/100 = 100 pages
Works has 20,000 tuples with 400 per page \( \Rightarrow \) there are 20000/400 = 50 pages
Project has 500 tuples with 120 per page \( \Rightarrow \) there are 500/120 = 4.16 \( \approx \) 5 pages
Since there are 5 types of projects, the selection on projects with type=design will generate 100 tuples fitting in one page.
Since there are ¾ of employees born after 1961, the selection with the birth date constraint will generate 7500 tuples fitting in 75 pages.

Since the selection on Project is smaller than the Works relation, Works should better be the outer relation.

The first select costs 5 I/Os.
Since the result is the size of one buffer, it can reside in main memory to do the join. Thus, the cost of the first join is the cost of scanning Works: 50 I/Os
The result of the first join is estimated at 20000/ 500 *100 = 4000 tuples. This is assuming a uniform distribution (i.e. the number of employees assigned per project is uniform.).
Since the distributions are assumed uniform: we have 20,000 works tuples and 500 projects. That is 40 employees per project. Since we have 100 projects with type “design”, that gives us 4000 tuples.
At 120 tuples per page, the result is about 34 pages (exactly 33 and a third). Thus writing T1 costs 34 I/Os. The cost of the second select is 100 I/Os and writing T2 costs 75 I/Os. The cost of the second join is $34 + \frac{34}{4} \times 75 = 709$ I/Os.
Thus the total cost for this plan is $5 + 50 + 34 + 100 + 75 + 709 = 973$ I/Os.

The first select costs 5 I/Os.
The result can fit in one buffer and can be sorted in main memory.
Sorting Works on PNO costs $2 \times \log_5(50) \times 50 = 2 \times 3 \times 50 = 300$ I/Os. The SM join would cost 50 I/Os since the outer relation fits in memory.
Writing T2 costs 34 I/Os (see above) and sorting T2 on ESIN costs $2 \times 3 \times 34 = 204$ I/Os.
Selecting Employees costs 100 I/Os for scanning and 75 I/Os to write T3. Sorting T3 on SIN costs $2 \times 3 \times 75 = 450$ I/Os. The final join costs $75 + 34 = 109$ I/Os.
Thus the total cost for this plan is $5 + 300 + 50 + 34 + 204 + 100 + 75 + 450 + 109 = 1327$ I/Os.

The first select costs 5 I/Os.
Since the result is the size of one buffer, it can reside in main memory to do the join. Thus, the cost of the first join is the cost of scanning Works: 50 I/Os.
Writing T1 costs 34 I/Os (see above) and sorting T1 on ESIN costs $2 \times 3 \times 34 = 204$ I/Os.
Selecting Employees costs 100 I/Os for scanning and 75 I/Os to write T3. Sorting T3 on SIN costs $2 \times 3 \times 75 = 450$ I/Os. The final join costs $75 + 34 = 109$ I/Os.
Thus the total cost for this plan is $5 + 50 + 34 + 204 + 100 + 75 + 450 + 109 = 1027$ I/Os.

The best plan among these three is to use Bloc-Nested Loops join for both joins.