Test Questions

Nov 26th

CS425 - Database Organization

Results

Please leave this empty!

1.1  1.2  1.3  1.4  Sum
Part 1.1 Normalization and Functional Dependencies (Total: 0 Points)

Consider the following relation \( R(A, B, C, D) \) and functional dependencies \( F \) that hold over this relation.

\[
F = A \rightarrow B, D \\
C, D \rightarrow B \\
C \rightarrow D \\
B \rightarrow D
\]

Question 1.1.1 (0 Points)
Determine all candidate keys of \( R \).

Solution

\( \{A, C\} \)

Question 1.1.2 (0 Points)
Compute the attribute cover of \( X = \{C, B\} \) according to \( F \).

Solution

\( X^+ = \{B, C, D\} \)
**Question 1.1.3**  (0 Points)

Compute the attribute cover of $F$. Show each step of the generation according to the algorithm shown in class.

**Solution**

1st iteration: 1) Apply union rule to combine right-hand sides:

no union possible

$F_1 = A \rightarrow B, D$

$C, D \rightarrow B$

$C \rightarrow D$

$B \rightarrow D$

1st iteration: 2) Find extraneous attribute:

$D$ is extraneous in $C, D \rightarrow B$

$F_2 = A \rightarrow B, D$

$C \rightarrow B$

$C \rightarrow D$

$B \rightarrow D$

2nd iteration: 1) Apply union rule to combine right-hand sides:

$F_3 = A \rightarrow B, D$

$C \rightarrow B, D$

$B \rightarrow D$

2nd iteration: 2) Find extraneous attribute:

$D$ is extraneous in $A \rightarrow B, D$.

$F_4 = A \rightarrow B$

$C \rightarrow B, D$

$B \rightarrow D$

3rd iteration: 1) Apply union rule to combine right-hand sides:

none apply.

$F_5 = A \rightarrow B$

$C \rightarrow B, D$

$B \rightarrow D$

3rd iteration: 2) Find extraneous attribute:

$D$ is extraneous in $C \rightarrow B, D$.

$F_6 = A \rightarrow B$

$C \rightarrow B$

$B \rightarrow D$

4th iteration: 1) + 2)

none apply.

$F_7 = A \rightarrow B$

$C \rightarrow B$

$B \rightarrow D$

---

**Question 1.1.4**  (0 Points)

In which normal form is relation $R$ (recall that a relation can be in multiple normal forms).

- [ ] 2NF
- [ ] 3NF
- [ ] BCNF
Question 1.1.5  (0 Points)

If $R$ is not in 3NF then decompose it.

**Solution**

Adding relations for each functional dependency:

\[
R_1(A, B) \quad R_2(C, B) \quad R_3(B, D)
\]

Add relation to hold candidate key if necessary:

\[
R_1(A, B) \quad R_2(C, B) \quad R_3(B, D) \quad R_4(A, C)
\]

Remove contained relations (in this case none)

\[
R_1(A, B) \quad R_2(C, B) \quad R_3(B, D) \quad R_4(A, C)
\]

Question 1.1.6  (0 Points)

If you have composed $R$ in the previous step then determine the candidate keys for each relation created during the decomposition.

**Solution**

Relations are

\[
R_1(A, B) \quad R_2(C, B) \quad R_3(B, D) \quad R_4(A, C)
\]

Candidate keys are $R_1 : \{A\}$, $R_2 : \{C\}$, $R_3 : \{B\}$, and $R_4 : \{A, C\}$. 
Part 1.2 Concurrency Control (Total: 0 Points)

Question 1.2.1 (1 Point)

For each of the following schedules determine which properties this schedule has. E.g., a schedule may be recoverable and cascade-less (strict) or conflict-serializable. Consider the following notation for operations of transactions:

- \(w_1(A)\) transaction 1 wrote item \(A\)
- \(r_1(A)\) transaction 1 read item \(A\)
- \(c_1\) transaction 1 commits
- \(a_1\) transaction 1 aborts

\[
S_1 = r_1(A), w_2(A), r_1(B), c_1, w_3(B), r_3(B), w_3(A), c_3, r_2(C), c_2
\]
\[
S_2 = r_1(A), w_2(B), r_1(B), c_1, c_2
\]
\[
S_3 = r_1(A), w_2(B), c_2, r_1(B), w_1(B), c_1
\]
\[
S_4 = w_1(A), w_2(A), c_2, w_1(A), c_1
\]

- ■ \(S_1\) is recoverable
- ■ \(S_1\) is cascade-less
- ■ \(S_1\) is conflict-serializable
- ☐ \(S_2\) is recoverable
- ☐ \(S_2\) is cascade-less
- ■ \(S_2\) is conflict-serializable
- ■ \(S_3\) is recoverable
- ■ \(S_3\) is cascade-less
- ■ \(S_3\) is conflict-serializable
- ■ \(S_4\) is recoverable
- ■ \(S_4\) is cascade-less
- ☐ \(S_4\) is conflict-serializable
Question 1.2.2  Create a Strict Schedule (8 Points)

Consider the following set of transactions:

\[ T_1 = r_1(A), w_1(A), c_1 \]
\[ T_2 = r_2(B), r_2(A), w_2(B), w_2(A), c_2 \]
\[ T_3 = r_3(B), w_3(B) \]

1. Write a cascade-less history involving these three transactions.

Solution

Several solutions are correct. For example,

\[ S = r_1(a), w_1(A), c_1, r_2(B), r_2(A), w_2(B), w_2(A), c_2, r_3(C), w_3(C) \]

In a correct solution, if one transaction \( T_i \) writes an item, then the others cannot read nor write the same item until \( T_i \) commits.

Question 1.2.3  (1 Point)

Check all correct statements below

- In a cascade-less (strict) schedule if a transaction \( T_j \) read a data item written by transaction \( T_i \) then the commit of \( T_i \) has to be before this read operation of \( T_j \)
- A recoverable schedule is also cascade-less
- Not all conflict-serializable schedules are also 2PL
- Under 2PL a transaction is split into three phases, a first growing phase, a shrinking phase, and a second growing phase
- Every SS2PL schedule is also 2PL

Solution