Sayantan Chattopadhyay, 1000818145
Jay Ganguli, 1001575191

CSC343 Assignment 3

Question 1: Indexes

1. PersonType ON Person(Occupation, DateOfBirth)
   We often need to find students or other people types and the *Person* table is vary large. This would be a clustered secondary index (since it does not contain a key for *Person*). This index would be stored using a tree rather than a hash table since it needs to compare an age range to identify students (which is the sub-query we ran the most in Assignment 2). In particular, this would help with q2, q3, q11 and q12.

2. TakeOn ON Take(Date)
   This would again be a clustered secondary index (Date is not a key for *Take*, and entries are only added and not modified in this table). If you consider searching for trip records over a range of dates, then this index must be stored in a B+ tree.
   This is another situation where we can expect the table we're indexing to get really big over time. An index here would shorten the work and not be too difficult to maintain. Inparticular, this would help with q3, q7(part c), q8, q9 and q11.

Question 2: Database Design

(a) Decomposition: $C \rightarrow AFH \implies C \rightarrow H$
Transitivity: $DF \rightarrow AC$, $C \rightarrow H \implies DF \rightarrow ACH$
Decomposition: $C \rightarrow ACG \implies C \rightarrow G$
Transitivity: $DF \rightarrow ACH$, $C \rightarrow G \implies DF \rightarrow ACHG$
Transitivity: $DF \rightarrow ACHG$, $A \rightarrow B \implies DF \rightarrow ABCHG$
Decomposition: $H \rightarrow EA \implies H \rightarrow E$
Transitivity: $DF \rightarrow ABCHG$, $H \rightarrow E \implies DF \rightarrow ABCEGH$

Transitivity: $H \rightarrow AE$, $AEH \rightarrow BD \implies H \rightarrow AEBD$
Decomposition: $ABD \rightarrow FGH \implies ABD \rightarrow FG$
Transitivity: $H \rightarrow ABDE$, $ABD \rightarrow FG \implies H \rightarrow ABDEFG$
Decomposition: $DF \rightarrow AC \implies DF \rightarrow C$
Transitivity: $H \rightarrow ABDEFG$, $DF \rightarrow C \implies H \rightarrow ABCDEFG$

Union: $C \rightarrow AFH$, $C \rightarrow ACG \implies C \rightarrow AFGH$
Transitivity: $C \rightarrow AFGH$, $H \rightarrow E \implies C \rightarrow AEFGH$

Transitivity: $C \to AEFGH$, $AEH \to BD \implies C \to ABDEFGH$

Augmentation: $E \to F \implies DE \to DF$
Transitivity: $DE \to DF$, $DF \to ABCEGH \implies DE \to ABCFGH$

Augmentation: $A \to B \implies AD \to BD$
Transitivity: $AD \to BD$, $ABD \to FGH \implies AD \to BFGH$
Transitivity: $AD \to BFGH$, $H \to ABCDEFG \implies AD \to BCEFGH$

Therefore, the candidate keys of R are {C, H, AD, DE, DF}.

(b) λ = {DF → AD, AEH → BD, C → AFH, C → ACG, E → F, BC → EH, A → B, DE → HB, ABD → FGH, H → EA}

Decompose RHS into singleton (also listed in alphabetical order and duplicates removed):
λ^+ = {A → B, ABD → F, ABD → G, ABD → H, AEH → B, AEH → D, BC → E, BC → H, C → A, C → F, C → H, C → G, DE → H, DE → B, DF → A, E → F, H → A, H → E}

Minimize LHS:
Since $A \to B$

$$ABD \to F \implies AD \to F$$
$$ABD \to G \implies AD \to G$$
$$ABD \to H \implies AD \to H$$

Since $H \to A$ and $H \to E$

$$AEH \to B \implies H \to B$$
$$AEH \to D \implies H \to D$$

Now λ^+ = {A → B, AD → F, AD → G, AD → H, BC → E, BC → H, C → A, C → F, C → H, C → G, DE → H, DE → B, DF → A, E → F, H → A, H → B, H → D, H → E}

Finally, delete redundant FDs:
$BC \to H$ is implied by $C \to H$ and can be removed.
Since $C \to A$ and $C \to F$, we can remove $AD \to F$.
Since $C \to A$ and $C \to G$, we can remove $AD \to G$.
Since $C \to A$ and $C \to H$, we can remove $AD \to H$.
$BC \to E$ is implied by $C \to H, H \to E$ and can be removed.
Since $C \to A$ and $C \to H$, we can remove $H \to A$.
Since $C \to A$ and $A \to B$, we can remove $H \to B$.
Since $H \to D$ and $H \to E$, we can remove $DE \to H$.

Since $C \to F$ we can remove $E \to F$.
Since $C \to A$ and $A \to B$, we can remove $DE \to B$ and $DF \to A$.

Now λ^+ = {A → B, C → A, C → F, C → G, C → H, H → D, H → E}

Thus, we can decompose R into $R_1 = (\underline{A}, B)$, $R_2 = (\underline{C}, A, F, G, H)$ and $R_3(\underline{H}, D, E)$ with dependencies $F_1 = \{A \to B\}, F_2 = \{C \to A, C \to F, C \to G, C \to H\}$ and $F_3 = \{H \to D, H \to E\}$.

Question 3: Database Design

(a) Dependencies $F\{D \to ABCEF\}$ would decompose into $F_1\{D \to ABC\}$ and $F_2\{D \to EF\}$.
Both R_1 and R_2 still have the same primary key D. The union of F_1 and F_2 would give us back $D \to ABCEF$.

(b) Dependencies $F\{D \to ABCEF, A \to F\}$ would decompose into $F_1\{D \to ABC\}$ and $F_2\{D \to EF\}$.
Both R_1 and R_2 still have the same primary key D, but we lose the dependency $A \to F$ so it is not dependency-preserving.

(c) Dependencies $F\{A \to BCD, E \to DF\}$ would decompose into $F_1\{A \to BCD\}$ and $F_2\{E \to DF\}$.
The common attribute of both F_1 and F_2 is not a key for either relation, so it is not a lossless join. The union of F_1 and F_2 would still give us the original dependency set F.

(d) Let $F_1 = \{B \to ACDEF\}$ and $F_2 = \{C \to A\}$

  (i) Armstrong's axioms do not remove attributes from the LHS. In F_1 and F_2, the attribute A was removed from the LHS and a different attribute has taken its place.

  (ii) F_1 still satisfies BCNF since the LHS B is a superkey in this case.

  (iii) F_1 works in 3NF as well (as it satisfies the same criteria as above). F_2 works only in 3NF and not in BCNF since its LHS C is not a superkey, but its RHS A is a key for S.

Question 4: Transactions and Concurrency

(a)
$$S = T1 : w(A), T2 : r(A), T1 : r(A)$$

This schedule is not possible with 2PL since $T1$ must release the exclusive lock on A prior to $T2$ locking and reading it before $T1$ needs to lock A again prior to reading it again.

We can however conflict serialize this schedule because we can find a conflict equivalent.

$$S = T1 : w(A), T1 : r(A), T2 : r(A)$$

(b) Every schedule that is allowed in Strict 2PL locking protocol is allowed in 2PL locking protocol. Since the locking in 2PL is more tolerant with items not be locked until they are committed as in the case of Strict 2PL, the following schedule is possible for 2PL but not Strict 2PL.

$$T1 : R(A), W(A), T2 : R(A), W(A), R(B), W(B), Commit, T1 : Abort$$

Because this schedule is not possible in Strict 2PL, A remains locked until it is either committed or aborted, but since 2PL does not have this restriction this schedule is allowed under 2PL.

Question 5: Transactions and Concurrency

1. This schedule is serializable, conflict serializable, and regardless of commit or abort it is also view serializable. Since commit or abort of the schedule is not given it not possible to know whether the schedule is recoverable or not. We do know that it is not strict and does not avoid cascading aborts.

2. This schedule does not belong to the strict class, besides that it does belong to all the others it is serializable, conflict serializable, view serializable, recoverable, and avoids cascading aborts.

3. This schedule belongs to all classes, it is serializable, conflict serializable, view serializable, recoverable,avoids cascading aborts, and is strict.

4. This schedule is serializable, conflict serializable, and view serializable. Since it is not recoverable, it is not strict and does not avoid cascading aborts.

5. This schedule is recoverable, avoids cascading aborts, and strict. It is not however serializable, view serializable, and conflict serializable.