# Database Management (COIY028H6) – 2020

## Model Answers

1. (a) There would be 3 tables named vehicle, car and motorbike in each system. In a conventional relational database, the schemas of car and motorbike would have to include license and make explicitly. In an object-relational database, one would define a type for each table, with carType and motorbikeType being declared as UNDER vehicleType. Similarly, the tables would be declared as UNDER the vehicle table. Only the additional attributes for each of car and motorbike would need to be declared for each table. (8 marks)

   (b) i. The conditions would be that there must not be an existing row with the same Supplier and Part values as in the INSERT statement, and that the `Quantity` attribute must be allowed to take NULL values. The command results in the tuple ('ABC Ltd', 'Axle', NULL) being inserted into the `Supply` relation. (5 marks)

   ii. We could find all suppliers from the Supply table, and then remove from those the suppliers in the nonSupplier view by using EXCEPT or NOT IN. (4 marks)

   (c) If $T_2$ commits, then this doesn't cause a problem for $T_1$. If $T_2$ aborts and $T_1$ requires perfectly accurate information, then this will be a problem because $T_1$ will have read an incorrect value. If $T_1$ only needs an estimate (say it is summing total revenues), then $T_2$ aborting may not be a problem. (3 marks)

2. (a) i. A key is given by $\{$*Track, Artist, Role*$\}$. This functionally determines all attributes because *Track* $\rightarrow$ *TrackTitle*, *Track* $\rightarrow$ *Album*, and *Album* $\rightarrow$ *AlbumTitle*. No subset of these attributes determines the full set. (4 marks)

   ii. The three FDs form a canonical cover. Each then forms a relation schema as follows: (*Track, TrackTitle*), (*Album, AlbumTitle*) and (*Track, Album*). No schema is a subset of another. Finally, we add another relation schema which forms a key, namely, (*Track, Artist, Role*). (5 marks)

   (b) i. The necessary PHP code would be

   ```
   $table = $_GET['tablename'];
   ```

   (or it could use `POST`). (3 marks)

   ii. If the user entered "Pubs; delete * from Pubs" in the text box, then the database system would delete all the rows from the table. (2 marks)

   iii. The new query would be `select * from ?`. No, it would not solve the problem, because arbitrary strings entered by the user would still be sent to the database system. (3 marks)

   (c)

| "supplier='ABC Ltd"' | "not(supplier='ABC Ltd') and quantity $>$ 50" |
|---|---|
| true | false |
| false | true |
| unknown | unknown |

(3 marks)

 

3. (a) In the definition of the `Player` table, `team` would be declared as a foreign key referencing the (primary key of the) `Team` table. The `team` value for any row inserted in `Player` must already exist as a `name` value in the `Team` table. A row cannot be deleted from the `Team` table if the `name` value appears as a `team` value for a row in the `Player` table. (6 marks)

   (b) The names of players and their teams where the team is based in England and the player is British. (3 marks)

   (c) `FROM Team JOIN Player ON Team.name=Player.team` (2 marks)

   (d) (1) The system could use nested loops to scan through both relations, testing the values of the `country` and `nationality` attributes. This would probably be the least efficient method. (2) The system might use a sort-merge. This would require sorting `Team` on `name` and `Player` on `team`. This would probably be more efficient than (1). (3) The system could scan the `Player` relation, testing the value of the `nationality` attribute. When that matches, it could use an index lookup on the `team` value, since `name` is the primary key in the `Team` relation. This would probably be the most efficient method. (9 marks)

4. (a) We start with $AB$, can add $C$ using the first FD, and then $D$ from the third FD, so the closure is $ABCD$. For $BC$, we can add $A$ and $D$ from the second and third FDs, respectively, so the closure is $ABCD$. (4 marks)

   (b) i. Abbreviating the attributes, the keys are $TSP$ and $TFP$. Using the closure algorithm on $TSP$, we start with $TSP$, can add $W$ (from first FD), can add $Y$ (from second FD), and can add $F$ (from third FD). No subset of $TSP$ determines all attributes since we need all three to determine $Y$. A similar argument holds for $TFP$. (4 marks)

   ii. It violates BCNF because the lefthand side is not a superkey. It does not violate 3NF because the righthand side is prime ($F$ is part of a key). (4 marks)

   iii. The amount of flour for each size of loaf is repeated for every type of bread. (2 marks)

   iv. One can start with $S \rightarrow F$ which violates BCNF. Decomposition yields $\{S, F\}$ and $\{T, S, P, Y, W\}$. The FD $TS \rightarrow W$ violates BCNF in the second schema since $TS$ is not a superkey. Decomposition yields $\{T, S, W\}$ and $\{T, S, P, Y\}$. Both are in BCNF. (6 marks)

 