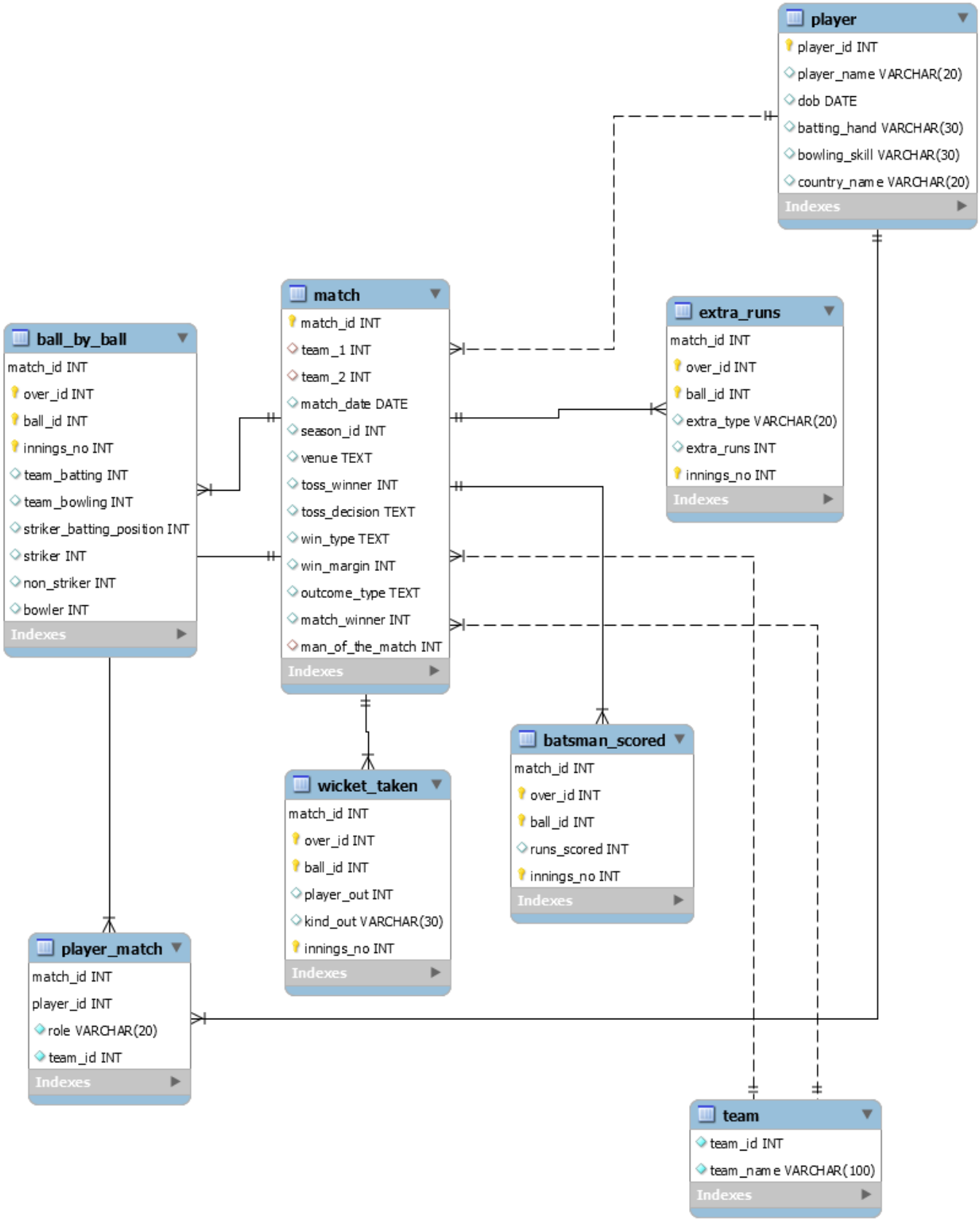
CS 432 Databases

Assignment - 3

Instructor : Prof. Mayank Singh

Devanshu Thakar
nilesh.thakar@iitgn.ac.in
18110174

The image of the Schema diagram is shown below :



Q1a For all the matches_id(entire IPL), find the minimum runs scored in any over and the bowler who bowled that over. Sort by increasing match_id, followed by increasing innings_no, then finally by increasing over_ids. Output:

Q1a

Limit to 1000 rows

```
1  select player_name as bowler_name, net_runs as runs_scored from ball_by_ball natural join
2  (select * from
3  (select match_id, innings_no, over_id,
4  case
5  when extras is not null then extras + over_run
6  else over_run end as net_runs
7  from ( select match_id, over_id, innings_no, over_run, extras
8  from ( select match_id, over_id, innings_no, sum(runs_scored) as over_run
9  from batsman_scored group by match_id, over_id, innings_no) as over_runs
10 natural left outer join (select  match_id, over_id, innings_no, sum(extra_runs) as extras
11 from extra_runs where extra_type in ('noballs', 'wides') group by match_id, over_id, innings_no) as extra_runs)
12 as net_over)  as minOve where net_runs= ( select min(net_runs) from (select match_id, innings_no, over_id,
13 case when extras is not null then extras + over_run  else over_run end as net_runs
14 from ( select match_id, over_id, innings_no, over_run, extras
15 from ( select match_id, over_id, innings_no, sum(runs_scored) as over_run
16 from batsman_scored group by match_id, over_id, innings_no) as over_runs
17 natural left outer join (select  match_id, over_id, innings_no, sum(extra_runs)
18 as extras from extra_runs where extra_type in ('noballs', 'wides') group by match_id, over_id, innings_no)
19 as extra_runs) as net_over) as minOV ) ) as minOVer join player on bowler=player_id
20 group by match_id, over_id, innings_no;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

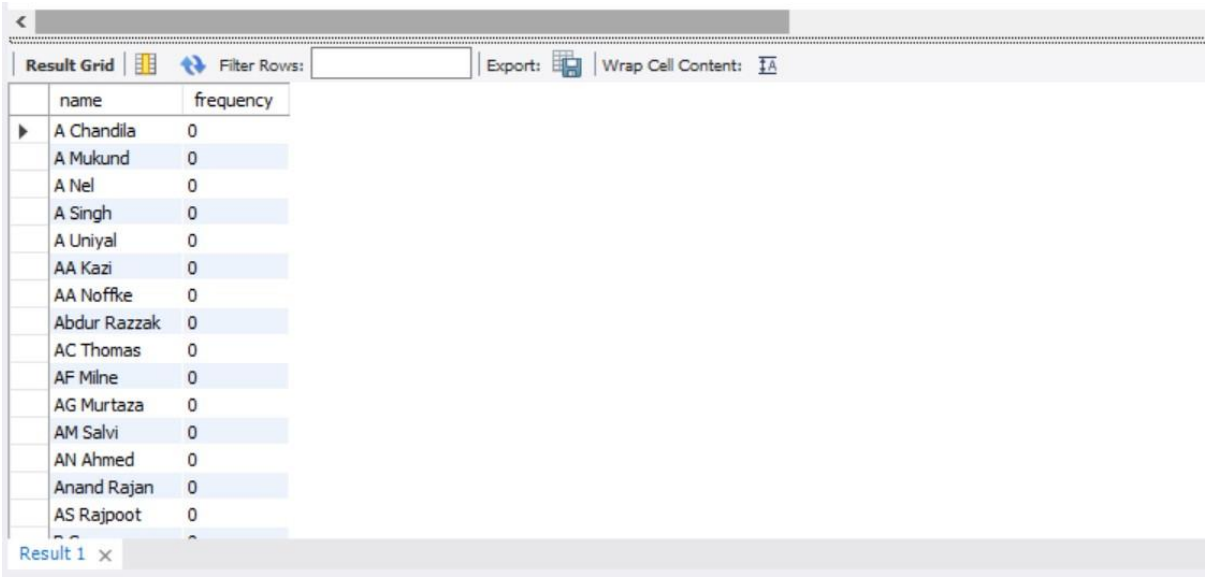
bowler_name	runs_scored
GD McGrath	0
I Sharma	0
GD McGrath	0
MS Gony	0
IK Pathan	0
SK Warne	0
RP Singh	0
ST Jayasuriya	0
SR Watson	0
M Ntini	0
IK Pathan	0
P Kumar	0
DJ Bravo	0

Q1b Find the names of all the batsmen(players) and the frequency of their “caught” out in increasing order of the number of “caught”. If a tie occurs, sort names alphabetically. Hint: Frequency can be 0 too. <names><frequency>

SQL File 6 | Q1b

Limit to 1000 rows

```
1  select player_name as name,
2  case
3      when frequency is not null then frequency
4      else 0
5  end as frequency
6  from
7  (select player_out, count(kind_out) as frequency, kind_out
8  from wicket_taken where kind_out='caught' group by player_out) as caught_out
9  right outer join player on player_id=player_out
10 order by frequency asc, player_name asc;
```

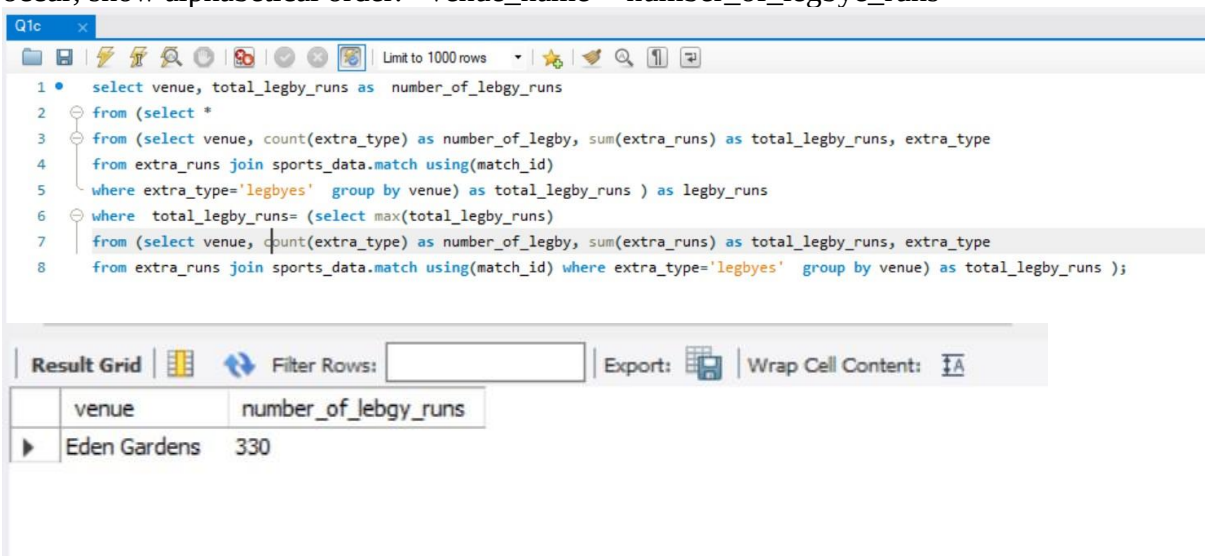


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	frequency
A Chandila	0
A Mukund	0
A Nel	0
A Singh	0
A Uniyal	0
AA Kazi	0
AA Noffke	0
Abdur Razzak	0
AC Thomas	0
AF Milne	0
AG Murtaza	0
AM Salvi	0
AN Ahmed	0
Anand Rajan	0
AS Rajpoot	0

Result 1

Q1c List the stadium(s) where the maximum number of "legbyes" (runs) is taken. If ties occur, show alphabetical order. <venue_name><number_of_legbye_runs>



Q1c

Limit to 1000 rows

```
1  select venue, total_legby_runs as  number_of_lebgy_runs
2  from (select *
3  from (select venue, count(extra_type) as number_of_legby, sum(extra_runs) as total_legby_runs, extra_type
4  from extra_runs join sports_data.match using(match_id)
5  where extra_type='legbyes'  group by venue) as total_legby_runs ) as legby_runs
6  where  total_legby_runs= (select max(total_legby_runs)
7  from (select venue, count(extra_type) as number_of_legby, sum(extra_runs) as total_legby_runs, extra_type
8  from extra_runs join sports_data.match using(match_id) where extra_type='legbyes'  group by venue) as total_legby_runs );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

venue	number_of_lebgy_runs
Eden Gardens	330

Q1d Find the bowler(s)(players) who has the best average(no. of runs given/wickets taken) in edition 5. If a tie occurs, sort names alphabetically. <bowler_name><average>

Q1d

```
-- Ignore the commented queries
-- Finding the runs given by a bowler
-- Finding all the wickets taken by a boweler in season=5
select player_name as bowler_name, (runs_given/wicket_taken) as average from

( (select sum(net_runs) as runs_given, bowler, season_id  from ( (ball_by_ball natural join
        (select *,
        case
        when extra_runs is not null then runs_scored+extra_runs
        else runs_scored  end as net_runs
        from batsman_scored natural left outer join extra_runs) as net_runs ) join
        (select match_id, season_id from sports_data.match
        where season_id=5) as season5 using(match_id))
group by bowler) as runs_given join

(select bowler, count(*) as wicket_taken from
ball_by_ball natural join wicket_taken join  (select match_id, season_id from sports_data.match
    where season_id=5 ) as season5 using(match_id)
    where kind_out not in ('run out','obstructing the field', 'retired hurt')
group by bowler) as wickect_take using(bowler) ) join player on player_id=bowler
where runs_given/wicket_taken =
```

Q1d

```
( select min((runs_given/wicket_taken)) as average from
(  (select sum(net_runs) as runs_given, bowler, season_id  from ( (ball_by_ball natural join
        (select *,
        case
        when extra_runs is not null then runs_scored+extra_runs
        else runs_scored  end as net_runs
        from batsman_scored natural left outer join extra_runs) as net_runs ) join
        (select match_id, season_id from sports_data.match
        where season_id=5) as season5 using(match_id))
group by bowler) as runs_given join

(select bowler, count(*) as wicket_taken from
ball_by_ball natural join wicket_taken join  (select match_id, season_id from sports_data.match
    where season_id=5 ) as season5 using(match_id)
    where kind_out not in ('run out','obstructing the field', 'retired hurt')
group by bowler) as wickect_take using(bowler) ) join player on player_id=bowler );
```

Result Grid | Filter Rows:

bowler_name	average
DAJ Bracewell	10.6667
DS Kulkarni	10.6667

Q1e Find out the names of all batsmen(players) who scored more than 100 runs in a match and, their runs scored. Sort names alphabetically. (if multiple entries of the same player, show the one with the highest runs).<batsmen_name><runs>

Q1e

```
select player_name as batsman_name, max(runs_scored) as runs
from player join
(select striker, match_id , sum(runs_scored) as runs_scored
    from batsman_scored natural join ball_by_ball group by striker, match_id  having sum(runs_scored)>100)
as runs_scored on player_id=striker
group by player_name
order by player_name asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

batsman_name	runs
A Symonds	117
AB de Villiers	133
AC Gilchrist	109
AM Rahane	103
BB McCullum	158
CH Gayle	175
DA Miller	101
DA Warner	109
DPMD Jayawardene	110
KP Pietersen	103
M Vijay	127
MEK Hussey	116
MK Pandey	114
PC Valthaty	120
Q de Kock	108

Q1 f Find out the top 3 batsmen(players) whose [number of runs scored/number of matches played] is the best in edition 2. Sort alphabetically. <batsman_name><value>

Q1f

```
-- Ignore the commented queries
select * from (
select player_name as batsman_name, (runs_scored/match_played) as value  from
(select sum(runs_scored) as runs_scored, striker,  season_id  from ( (ball_by_ball natural join batsman_scored) join
        (select match_id, season_id from sports_data.match
        where season_id=2) as season5 using(match_id))
group by striker) as runs_scored join
(select count(distinct match_id) as match_played, striker,  season_id  from ( (ball_by_ball natural join batsman_scored) join
        (select match_id, season_id from sports_data.match
        where season_id=2) as season5 using(match_id))
group by striker) as matchs_played using(striker)
join
player on striker=player_id
order by (runs_scored/match_played) desc
limit 3 ) as top_3
```

Result Grid | Filter Rows: | Export: | Wrap Cel

batsman_name	value
AB de Villiers	35.7692
MK Pandey	42.0000
ML Hayden	47.6667

Q1g Find out the batting average(as calculated in the above question (f)) of all players. Then only show the list of the top 3 countries with the highest country batting average(∑batting average/Total number of players in that country)<country><value>

Q1g

```
use sports_data;
select avg(value) as value, country_name from
 (select player_name as batsman_name, country_name, (runs_scored/match_played) as value
 from  (select sum(runs_scored) as runs_scored, striker  from  (ball_by_ball natural join batsman_scored)
        group by striker) as runs_scored join
(select count(distinct match_id) as match_played, striker  from  (ball_by_ball natural join batsman_scored)
group by striker) as matchs_played using(striker)
join
player on striker=player_id ) as new_tab
group by country_name
order by value desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

value	country_name
16.70926429	England
16.22694706	West Indies
15.52380000	Netherlands
14.87828333	Australia
14.48165135	South Africa
13.62034500	New Zealand
10.33330000	Zimbabwea
10.24532308	Pakistan
10.20853651	India
9.92118500	Sri Lanka
4.90000000	Bangladesh

Q1h Write down a simple query to make a copy of the player table(with data).

SQL File 6 | Q1h

```
create table player_copy
select * from player;
```

MySQL 8.0 Command Line Client

```
mysql> show tables;
+----------------------+
| Tables_in_sports_data |
+----------------------+
| ball_by_ball         |
| batsman_scored       |
| extra_runs           |
| match                |
| player               |
| player_copy          |
| player_match         |
| team                 |
| wicket_taken         |
+----------------------+
9 rows in set (0.13 sec)
```

Q1i Using view, create a table say “indian_players” which contains information about the total runs scored by all the Indian players till now and sort them alphabetically.<name><runs>

Q1i

```
create view India_Players as
(select player_name as name, sum(runs_scored) as runs
from ball_by_ball natural join batsman_scored join player on player_id=striker
where country_name='India'
group by striker
order by player_name, runs);
-- select * from India_Players;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	runs
A Ashish Reddy	266
A Chandila	4
A Chopra	53
A Kumble	35
A Mishra	291
A Mithun	34
A Mukund	19
A Nehra	41
A Singh	2
A Uniyal	4
AA Bilakhia	69
AA Chavan	12
AA Jhunjhunwala	217
AB Agarkar	178
AB Barath	42
AB Dinda	19

Q1j List all captains who scored more than 50 runs in edition 3. Sort names alphabetically <name><runs>

Q1j

```
1  select player_name as name, total as runs
2  from player natural join
3  (select player_id, sum(match_runs) as total
4  from
5  (select match_id, striker as player_id, season_id, sum(runs_scored) as match_runs
6          from ( (ball_by_ball natural join batsman_scored) natural join
7          (select match_id, season_id from sports_data.match
8          where season_id=3) as season3)
9  group by striker, match_id) as season_3 natural join player_match
10 where role = 'Captain' or role='Captainkeeper'
11 group by player_id) as Captain_score
12 where total>50
13 order by player_name asc;
```

Result Grid | Filter Rows: | Export:

name	runs
AC Gilchrist	289
G Gambhir	277
KC Sangakkara	342
KD Karthik	82
MS Dhoni	287
SC Ganguly	493
SK Raina	58
SR Tendulkar	617

Q2 Suppose a user creates a new relation r1 with a foreign key referencing another relation r2. What authorization privilege does the user need on r2? Why should this not simply be allowed without any such authorization? (max 500 words) (4 marks)

Ans. The user needs references privilege on r2.

SQL provides a references privilege that allows a user to declare foreign keys when creating relations. It may appear that why there is a need to have a references privilege. However, not every user can be allowed to declare a foreign key. Suppose the user who created the new relation r_1 inserts a tuple t into r_1. Now tuple t will have a foreign key referring some tuple t' (say) in relation r_2. It is known that foreign key constraints restricts delete and update operations on the referenced relation. Now, it will not be possible to delete or update tuple t' in r_2 without modifying the tuple t in referencing relation r_1. Thus, the foreign key defined by user prevents future modification to the referenced relation. Therefore there is a need for an authorization privilege for declaring foreign keys.

Q3 Explain the difference between integrity constraints and authorization constraints. (explain them with examples) (max 500 words) (4 marks)

Ans. Integrity constraints ensures that changes made to the database do not results into loss of data consistency. Integrity constraints are generally defined while defining the schema of a relation. However, they can also be added later provided that there is no violation. Not Null, Unique, check <predicate> , referential integrity, assertions are different integrity constraints available in SQL.

Example – Consider a relation *student* containing attributes like student_name, dob, student_department, email. Consider another relation named *department* having tuples for every department. Referential integrity (foreign key) ensures that every tuple in *student* realtion has an entry in student_departement which corresponds to some tuple in *department* relation.

Authorization constraints allows the database administrator to provide or restrict the authority to modify database by the users. SQL provides authorization to read, insert, update and delete data. Each type of authorization constraint is known as privilege. When a user executes a query, the SQL implementation first checks whether the user is authorized to perform the query, only then the query is executed. Integrity constraints are related to the consistency of database as defined. Authorization constraints are related to the privileges of the user to run queries.

Example – Suppose a user creates a table T and he tries to creates a foreign key attribute in T. He can declare a foreign key only if he has been granted the references privileges.

Q4. Consider a set of users A, B, C, D, and E. Suppose the user A creates a table T and thus is the owner of T. Now suppose the following set of statements is executed in order:

1. User A: grant select on T to B, C with grant option
2. User B: grant select on T to C
3. User C: grant select on T to D, E
4. User A: grant select on T to E
5. User A: revoke select on T from B restrict
6. User A: revoke select on T from C cascade

When does D not have SELECT ON T privilege? Justify your answer. (3 marks)

Ans. D will have SELECT ON T privilege after statement-3. D does not have SELECT ON T privilege after statement-6. D got its SELECT privilege from C, and C got its SELECT with grant privilege from A. In statement-6 user A revokes privileges from C with cascade. Therefore the revoke will cascade from C to D and D will not have SELECT ON T privilege after statement-6.

At the end of statement-5 C will have the permission of SELECT ON T with grant. C was granted SELECT ON T permission from A. B also granted SELECT ON T permission to C. But since C got SELECT ON T permission from A, it will retain its permission even when B is revoked. Moreover, revoke on B is executed with restrict there will no cascading of revoke to C due to revoke on B.