SPRING MID SEMESTER EXAMINATION-2024

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Database Management System
Solution

1.Answer all the questions.

a)We can convert weak entity sets into strong entity sets by simply adding the appropriate attributes. Then, why are weak entity sets needed?

Solution - It is true that adding appropriate attributes can convert any weak entity set to a strong entity set. However, we must consider doing this to a weak entity set can lead us to a redundant data and data inconsistencies as the values of the attributes are repeatedly inserted in the strong entity set.Since weak entity sets are always dependent on the identifying or owner entity set, and does not have to have descriptive attributes, then this helps us avoid data redundancy. Using weak entity sets also allows us to easily map the data within the database. In addition, since an entity in a weak entity set refers to a strong entity in a strong entity set, deleting strong entities automatically removes the dependent weak entities.

Note : marks should be awarded on the basis of keywords/synonyms of the words redundancy and dependency on data

b) If a user creates relation r1, and includes a foreign key referencing another relation r2, what authorization privilege is required for the user regarding r2? Additionally, why is it not advisable to permit this without the need for such authorization?

Solution - Referential integrity constraints,

It is not advisable to permit the creation of foreign keys without the need for such authorization for several reasons:

Data Integrity: Foreign keys are crucial for maintaining data integrity in a relational database. They ensure that references between tables are valid, preventing the creation of orphaned records and maintaining consistency in the database.
Consistency and Structure: By requiring explicit authorization for creating foreign keys, database administrators can better control the database's structure and maintain a consistent schema. Unrestricted creation of foreign keys could lead to a less organized and more error-prone database.

Note: half marks should be awarded if mentioned “Referential integrity constraints” and half marks is for the 2nd half part of the question

c) Relationships can have attributes. What are they called and give one example where this is required.

Solution - They are called descriptive attributes
(Example of a descriptive attributes)

Note: half marks for mentioning “descriptive attributes” and half for one example of descriptive attributes

d) What is the difference between compound key and composite key?

Solution - The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

e) Consider a relation R with attributes $\{a_1, a_2, a_3, \ldots, a_n\}$. determine the total number of super keys for relation R.

Solution - $2^n - 1$

Maximum Super keys = (2^n) – 1.

Proof: There are n attributes in R. So the total number of possible subsets/combination of attributes of R is (2^n)

Now to be a Super key, there should be at least one attribute present i.e. the NULL set or the set with no attribute can’t be a super key.

So, maximum possible number of Super keys of R = (2^n) – 1.

Note: Full marks should be awarded if mentioned “$2^n - 1$” otherwise zero

2. The division operator in relational algebra, denoted by ÷ is defined as follows. Let r(R) and s(S) be the relations with attribute sets R and S respectively. Let S ⊆ R (that is, every attribute in s is also present in r). Then r ÷ s is a relation on R-S (that is, that result consists of attributes in R - S). A tuple t∈r÷s iff:

- $t \in \Pi_{R-S}(r)$
- For every tuple $t_s \in s$, there is a tuple $t_r \in r$ satisfying: i) $t_r[S] = t_s[S]$ (that is the tuples match on the attribute set S), and, ii) $t_r[R - S] = t$

A. Write a relational algebra expression using the division operator to find the IDs of all students who have taken all Comp. Sci. courses using the university schema(given in question number 5). [1 Marks]

B. Without using division, show how to write the above query in relational algebra. [2 Marks]

C. Write the pseudocode to efficiently evaluate the division operator. [2 Marks]

Solution -

a. $\Pi_{Id,\, course-id}(takes) \div \Pi_{course-id}(\sigma_{dept-name\, =\, "Comp.\, Sci."}(Course))$

b. $\Pi_{Id}(takes) - \Pi_{Id}((\Pi_{Id}(takes) \times (\sigma_{dept-name\, =\, "Comp.\, Sci."}(Course)) - \Pi_{Id,\, course-id}(takes))$

c. Let $R(r_1, r_2, r_3..., r_n)$ and $S(s_1,\ s_2,... s_n)$ are two relations

So to implement division operator on $R(r_1, r_2) \div S(s_1)$ where r_2 is foreign key of S i.e r_2 And s_1 contain same unique values

```
Step 1. Select column $r_1$ from relation R
        $op_1 <-- \Pi_{r_1}(R)$
Step 2. Select $s_1$ from relation S
        $op_2 <-- \Pi_{s_1}(S)$
Step 3. Perform cartesian product on $op_1$ and $op_2$
        $cp <-- op_1 \times op_2$
        $cp$ will store values of form $(r1_{v1}, s1_{v1}),\ (r1_{v2}, s1_{v2})....$
Step 4. Select column $r_1, r_2$ From relation R
        $op_3 <-- \Pi_{(r_1, r_2)}(R)$
Step 5. Subtract $op_3$ from $cp$
        $op_4 <-- cp - op_3$
Step 6. Now select $r_1$ from $op_4$
        $op_5 <-- \Pi_{r_1}(op_4)$
Step 7. Now subtract $op_5$ from $op_1$
        $op_6 <-- op_1 - op_5$
$op_6$ will be the final result after division
```

Note: In part C the pseudo code can be in any language even in english marks should be awarded on the basis of logic

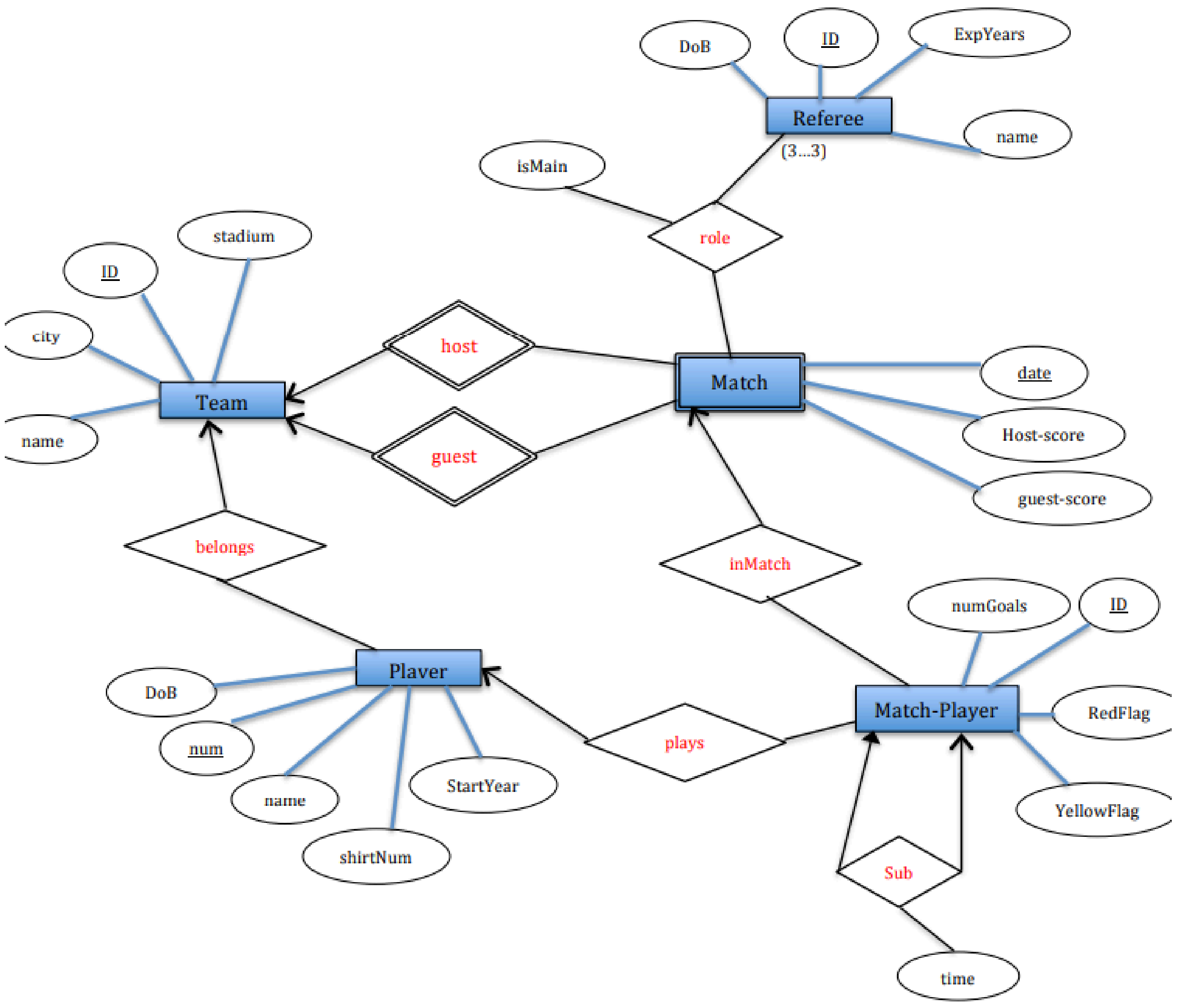
3. Assume we have the following application that models soccer teams, the games they play, and the players in each team. In the design, we want to capture the following:

- We have a set of teams, each team has an ID (unique identifier), name, main stadium, and to which city this team belongs.
- Each team has many players, and each player belongs to one team. Each player has a number (unique identifier), name, DoB, start year, and shirt number that he uses.
- Teams play matches, in each match there is a host team and a guest team. The match takes place in the stadium of the host team.
- For each match we need to keep track of the following:
    - The date on which the game is played
    - The final result of the match
    - The players participated in the match which will be stored in match_players. For each player, how many goals he scored, whether or not he took a yellow card, and whether or not he took a red card.
    - During the match, one player may substitute another player. We want to capture this substitution and the time at which it took place.
- Each match has exactly three referees. For each referee we have an ID (unique identifier), name, DoB, years of experience. One referee is the main referee and the other two are assistant referee.

A. Design an ER diagram to capture the above requirements. Make sure cardinalities and primary keys are clear. [3 Marks]

B. Convert the above ER diagram into relational schemas with specifying primary keys. [2 Marks]

Solution -



Team(<u>ID</u>, city, name, stadium)
Player(<u>num</u>, DoB, name, startYear, ShirtNum, TeamID)
Referee(<u>ID</u>, DoB, Name, ExpYear)
Match(<u>HostId</u>, <u>GuestId</u>, <u>Date</u>, Host_Score, Guest_Score)
RefereeRole(<u>HostId</u>, <u>GuestId</u>, <u>Date</u>, <u>RefId</u>, isMain)
Match-Player(ID, PlayerNum, MatchDate, HostId, GuestId, numGoals, redFlag, yellowFlag, subId, subTime)

Note: There can be multiple solution of this question please check accordingly

4. Explain the significance of each level and data independence at each level in the 3-Level Abstraction of a Database, and support your explanation with a diagram. [5 Marks]

Solution -

- **Physical or Internal Level:**
  The physical or internal layer is the lowest level of data abstraction in the database management system. It is the layer that defines how data is actually stored in the database. It defines methods to access the data in the database. It defines complex data structures in detail, so it is very complex to understand, which is why it is kept hidden from the end user. Data Administrators (DBA) decide how to arrange data and where to store data. The Data Administrator (DBA) is the person whose role is to manage the data in the database at the physical or internal level. There is a data center that securely stores the raw data in detail on hard drives at this level.

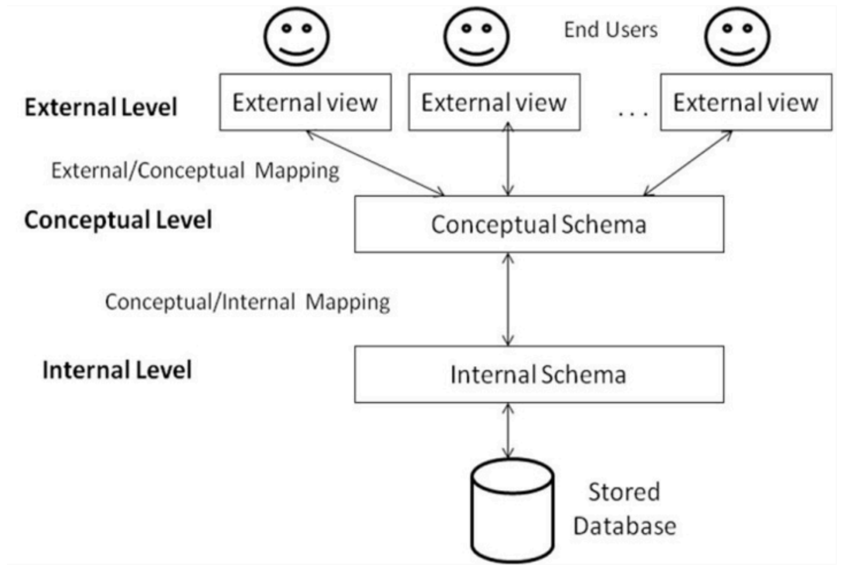
- **Logical or Conceptual Level:**
  The logical or conceptual level is the intermediate or next level of data abstraction. It explains what data is going to be stored in the database and what the relationship is between them.

  It describes the structure of the entire data in the form of tables. The logical level or conceptual level is less complex than the physical level. With the help of the logical level, Data Administrators (DBA) abstract data from raw data present at the physical level.

- **View or External Level:**
  View or External Level is the highest level of data abstraction. There are different views at this level that define the parts of the overall data of the database. This level is for the end-user interaction; at this level, end users can access the data based on their queries.

- **Physical Data Independence :**
  The physical data independence is basically used to separate conceptual levels from the internal/physical levels. It is easy to achieve physical data independence. With this type of independence, user is able to change the physical storage structures or the devices which have an effect on the conceptual schema.

- **Logical Data Independence :**
  Logical Data Independence is used to change the conceptual scheme without changing the following things :
    - External views
    - External API or programs



Note: Physical or Internal Level, Logical or Conceptual Level, View or External Level, (Physical Data Independence and Logical Data Independence) and diagram each contains 1 mark

5. Write the following queries either in relational algebra or SQL (choose only one), using the following university schema.

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

I. Find the ID and name of each instructor in the Physics department.
II. Find the set of all courses taught in the autumn semester 2023, the spring semester 2021, or both.
III. Find the ID and name of each student who has taken at least one course in the “Comp. Sci.” department.
IV. Find the names of all instructors in the computer science department together with the course id of all courses they taught.
V. Find the ID and name of each student who has not taken any course section in the year 2023.

[5 Marks]

Solution -

a. $\Pi_{ID,name}(\sigma_{dept-name="physics"}(instructor))$

SELECT ID, name
FROM instructor
WHERE dept_name = “Physics”;

b. $\Pi_{course-id}(\sigma_{semester = "autumn" \wedge year = 2023}(Section)) \cup \Pi_{course-id}(\sigma_{semester = "spring" \wedge year = 2021}(Section))$

Select distinct course_id
FROM section
WHERE (semester = “Autumn” AND year = 2023)
OR (semester = “Spring” AND year = 2021);

c. $\Pi_{ID,name}(student \bowtie_{student.ID=takes.ID} takes \bowtie_{takes.course-id=course.course-id} \sigma_{dept-name="comp. sci."}(course))$

SELECT takes.ID, student.name
FROM takes
INNER JOIN student ON student.ID = takes.ID
INNER JOIN course ON takes.course_id = course.course_id
WHERE Course.dept_name = “Comp. Sci.”;

d. $\Pi_{name,course-id}(\sigma_{dept-name = "computer\ science"}(Instructor \times Teaches))$

SELECT distinct instructor.name, teaches.Course_id
FROM Instructor
LEFT JOIN Teaches ON Teaches.Id = Instructor.Id
WHERE instructor.dept_name = “Computer Science”;

e. $\Pi_{ID,name}(student) - \Pi_{ID,name}(student \bowtie_{student.ID=takes.ID}(\sigma_{year=2023}(takes)))$

SELECT student.ID,student.name
FROM student
LEFT JOIN takes on student.ID = takes.ID
WHERE year = 2023 and course_id IS NULL;

Note: Each part contains 1 mark