Database Management Lab - 4: Constraints in the Tables; Views and Sequences

Employees in the company, constitute basketball teams together. Each one of the employees can play in a team through a particular time (for instance an employee can play for a team only for two weeks). There is not any obligation such as playing for only one team, for employees. Table "team" is given as:

TEAM

TNUMBER (PK)	NOT NULL	NUMERIC(2)
TNAME	NOT NULL	VARCHAR(15)

Example:
1. The relation table "Team_employee", which shows the relation between the table "Team" and the table "Employee", has attributes as below. Create the relation table "Team_employee" respect to those attributes:

Attributes (columns): Tno (numeric(2)), essn (char(9)), play_time (numeric (2))
Tno: team number; essn: ssn of the employee; play_time: how many weeks that employee plays in that team.
Constraints:

- Tno and Essn should be primary key of the table together.
- **Tno** should reference the column **tnumber** in the table Team. In addition to this if you delete a row in the table Team, the rows of the table Team_employee, which have the same tno should be deleted automatically.
- **Essn** should reference the column **ssn** in the table Employee. In addition to this if you delete a row in the table Employee, the rows of the table Team_employee, which have the same ssn should be deleted automatically.
- The column Play_time should not be greater than 12. You should obstruct this situation automatically. (An employee can not play in a team for more than 12 weeks!)

Table Team creation:
create table team(tnumber numeric(2), tname varchar(15), constraint pk_team primary key(tnumber))

Answers:

The creation of the relation table between Employee and team tables:

```
create table team_employee (Tno numeric(2), essn char(9) , play_time numeric(2),
constraint pk_team_emp primary key(tno, essn) ,
constraint fk_team foreign key(tno) references team(tnumber) on delete cascade,
constraint fk_emp foreign key(essn) references employee (ssn) on delete cascade,
constraint play_time_ck check (play_time <13)  );
```

Explanation of the answer: Definition of the primary key is done in the second row of the answer.
The third row says that "Tno references the column "tnumber" in the table Team" and "**on delete cascade**" says that "if you delete a team in the table **Team**, the rows in the **Team_employee** are also deleted automatically which have the same tno".
The forth row says that "Essn references the column "ssn" in the table Employee" and "**on delete cascade**" says that "if you delete an employee in the table **Employee**, the rows in the **Team_employee** are also deleted automatically which have the same essn".
The last row says that "the column play_time should not get any value which is greater than 12. It should be less than 13".

What is View?

Views are virtual tables or virtual datasets which are constructed by a SQL query. They are not stored physically in the database as tables, they do not occupy any place in the database as tables. They are only stored as queries in the database.

When the data in the tables (the tables which were used in the query of view) are changed, the data of the views are also changed automatically. We can create views as:

CREATE VIEW *viewname*
AS
SELECT *select-list*
FROM table-list
[WHERE search-condition]

Examples:

1. Create a view which includes the name and salary information of the employees whose salary is between 20000 and 40000.
2. Create a view which includes name and sex information of the employees who is working in the department “Sales”.

Answers:

1. We use the table employee when creatşng the view. The name of the view is salary_interval:

create view salary_interval as select fname, lname, salary from employee where salary between 20000 and 40000;

NOTE: For instance if you delete the employees, who has the salary equal to 30000$, from the table employee; the same employees will be deleted from the view automatically.

2. The name of the second view is sales_emps:

create view sales_emps as select fname, lname, sex from employee e, department d where e.dno=d.dnumber and d.dname='Sales'

What is Sequence?

It is a database object which constructs a set (or an array) of numeric value respect to a particular array rule. For instance we can use that object for giving auto-increment values to a unique column such as primary key columns. We can create sequences as:

CREATE SEQUENCE sequence_name

[INCREMENT BY #]

[START WITH #]

[MAXVALUE # | NOMAXVALUE]

[MINVALUE # | NOMINVALUE]

[CYCLE | NO CYCLE] => It **can/can not** return the initial value, when it reaches the last value

[CACHE #] => A particular number of the elements in the sequence can be taken in the memory to speed up the processes

We can get information about an existing sequence as writing “Select * from *sequence_name*” in the SQL editor.

Example:
1. Create a sequence which has a minimum value as 9 and a maximum value as 99 and increments one by one. Use that sequence for giving values to the column tnumber in the table Team. (You should add a new row to the table Team for this process).

Answer: The name of the sequence will be seq:
create sequence seq minvalue 9 maxvalue 99 increment by 1;

insert into team values(nextval('seq'), 'Kitties')

We take the next value of the sequence by “nextval('seq')” and then give that value to tnumber.

Union, Intersect, Except Operations in the Tables

1. Select the name and surname of the employees who are studying at the project named “OperatingSystems” **and** who are working at the department named “Software” (UNION ? / INTERSECT ? / EXCEPT ?)

2. Select the name and surname of the employees who are studying at the project named “OperatingSystems” **or** who are working at the department named “Software” (UNION ? / INTERSECT ? / EXCEPT ?)

3. Select the name and surname of the employees who are studying at the project named “OperatingSystems” **but** who are **not** working at the department named “Software” (UNION ? / INTERSECT ? / EXCEPT ?)

4. Select the name and the first name of the employees those are not the manager of any department or those are not the manager of any employee. (EXISTS ? / NOT EXISTS?)

5. Find the department name of the employees whose name is ‘John’ by using the keyword ‘IN’.

6. How many employees are there in the department named ‘Sales’? And what is the minimum, maximum, average and total salary of the employees who are working at the department named “Sales”.

Answers
1. select fname, lname from employee e, project p, works_on w where pname = 'OperatingSystems' and e.ssn=w.essn and p.pnumber=w.pno
INTERSECT
select fname, lname from employee e, department d where dname = 'Software' and e.dno=d.dnumber;

Explanation: Intersect expression is the equivalent of the word “and”; it represents the intersection of two sets.

2. select fname, lname from employee e, project p, works_on w where pname = 'OperatingSystems' and e.ssn=w.essn and p.pnumber=w.pno
UNION
select fname, lname from employee e, department d where dname = 'Software' and e.dno=d.dnumber;

Explanation: Union expression is the equivalent of the word “or”; it represents the union of two sets.

3. select fname, lname from employee e, project p, works_on w where pname = 'OperatingSystems' and e.ssn=w.essn and p.pnumber=w.pno
EXCEPT
select fname, lname from employee e, department d where dname = 'Software' and e.dno=d.dnumber;

Explanation: Except expression is the equivalent of the word “difference”; it represents the difference of one set, from another set.

4. select fname, lname from employee e where not exists (select null from department d where d.mgrssn = e.ssn) and not exists (select null from employee e2 where e2.superssn = e.ssn);

Explanation: The result set rows of the subquery, which comes after from the “not exists” expression, is eliminated from the result set of the outer query (the main query).

5. select dname from department where dnumber in (select dno from employee where fname ='John')

6. select count(*), min(salary), max(salary), avg(salary), sum(salary) from employee, department where e.dno = d.dnumber and dname = 'Sales'

Explanation: count(*) function returns the number of the rows of the query, min( ) function returns the minimum value of a numeric column, max( ) function returns the maximum value of a numeric column, avg( ) function returns the average value of a numeric column, sum( ) function returns the total value of a numeric column