USN										

Internal Assessment Test II – Oct. 2019

Sub:	Database Management Systems				Sub Code:	17CS53	Branch:	ISE	
Date:	14-10-19	Duration:	90 min's	Max Marks:	50	Sem / Sec:	V-A /B (ISE)		OBE

Scheme and Solution

1. Consider the following tables: [10]MARKS

works (Person_name, Company_name, Salary)
lives (Person_name, Street, City)
located-In (Compny_name, City)
write the following queries in SQL:

i) List the names of the people who work for the company 'Wipro' along with the cities they live in.

Select person_name,city from works w,lives l where w.person_name=l.person_name and Company_name='Wipro'

ii) Find the names of the persons who do not work for 'Infosys'.
Select person_name from works where company_name!='Infosys'

iii) Find the people whose salaries are more than that of all of the 'oracle' employees.
Select person_name from works where salary>(select max(salary) from works where company_name='oracle'

iv) Find the persons who works and lives in the same city.
Select person_name,l.city from works w,lives l, located-in li where w.person_name=l.person_name and l.city=li.city and w.company_name=li.company_name

v) Find the average salary, maximum salary and minimum salary of each company.
Select company_name, max(salary), min(salary),avg(salary) from works group by company_name

2. Consider the company database [10]MARKS

EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno)
DEPARTMENT(Dname ,Dnumber, Mgr_ssn, Mgr_start_date)
DEPT_LOCATIONS(Dnumber, Dlocation)
PROJECT(Pname, Pnumber ,Plocation, Dnum)
WORKS_ON(Essn, Pno, Hours)
DEPENDENT(Essn,Dependent_name, Sex, Bdate, Relationship)
Specify the following queries in SQL on the database schema given above

a. For every project located in Bangalore, list the project number the controlling department number and the department manager's last name, address and birth date.
Select Lname,Address,Bdate,pnumber,dnumber form employee e, department d, project p where e.ssn=d.mgr_ssn and d.dnumber=p.dnum and Dno=Dnumber and plocation='Bangalore'

b. List the names of the employees who have a more than one dependent.
Select Fname,Minit,Laname from employee where ssn in(select essn from dependent group by essn having count(*)>1)

c. For each project, list the project name and the total hours per week (by all employees) spent on that project.
   Select pname, sum(hours) form project p, works_on w where p.pnumber=w.pno group by pname

d. Retrieve the name of each employee who works on all the projects controlled by Research department.
   Select Fname,Minit,Laname ,pname, dname from employee e, department d, project p,works_on w where e. dno=d.dnumber and d.dnumber=p.dnum and e.ssn=w.essn and dname=’Research’

e. Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.

Select **Fname,Minit,Laname ,pname, salary, dno from employee where (dno, salary) in (select dno, max(salary) from employee group by dno)**

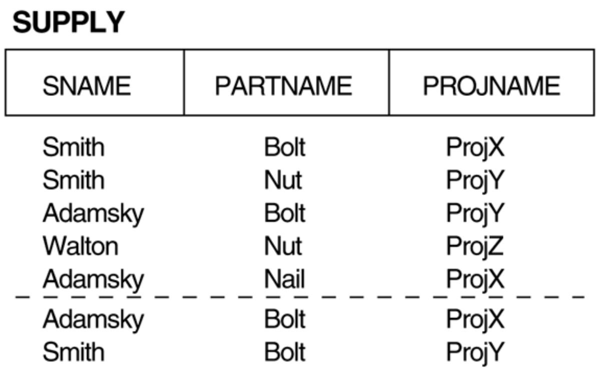
3. Define 4NF and 5NF. Explain with suitable example. [10]MARKS

Fourth Normal Form (4NF)

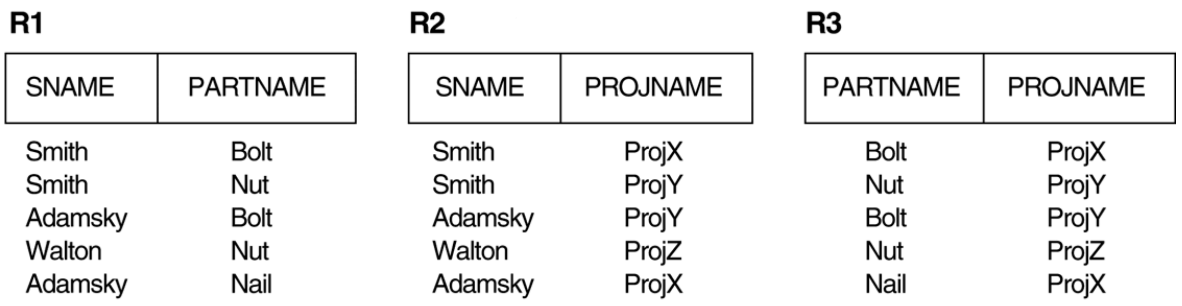
Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation. In this tutorial we will learn about Multi-valued Dependency, how to remove it and how to make any table satisfy the fourth normal form.

A relation schema *R* is in **4NF** with respect to a set of dependencies *F* (that includes functional dependencies and multivalued dependencies) if, for every *nontrivial* multivalued dependency $X \twoheadrightarrow Y$ in F^+, *X* is a superkey for R.

c)The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R1, R2, R3). (d) Decomposing the relation SUPPLY into the 5NF relations R1, R2, and R3
For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

(c)

SUPPLY

SNAME	PARTNAME	PROJNAME
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d)

R1

SNAME	PARTNAME
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R2

SNAME	PROJNAME
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R3

PARTNAME	PROJNAME
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

- A **multivalued dependency** (**MVD**) $X \longrightarrow\!\!> Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties, where we use Z to denote (R 2 (X υ Y)):

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
- $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
- $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.

- An MVD $X \longrightarrow\!\!> Y$ in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) X υ $Y = R$.

s_id	**course**	**hobby**
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

1. As you can see in the table above, student with s_id **1** has opted for two courses, **Science** and **Maths**, and has two hobbies, **Cricket** and **Hockey**.

Fifth normal form (5NF)

A relation schema R is in **fifth normal form** (**5NF**) (or **Project-Join Normal Form** (**PJNF**)) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R_1, R_2, ..., R_n) in F^+ (that is, implied by F), every R_i is a superkey of R.

SUBJECT	**LECTURER**	**SEMESTER**
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.
- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

Lossless Decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

4. Draw the neat diagram of single, two, three tier architecture diagrams. Explain the functional components in those architectures. [10]MARKS

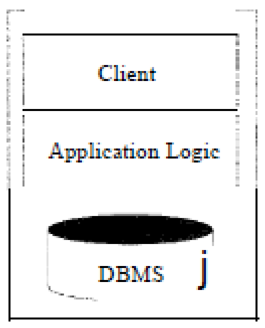
Data-intensive Internet applications can be understood in terms of three different functional components: data management, application logic, and presentation

Single-Tier and Client-Server Architectures

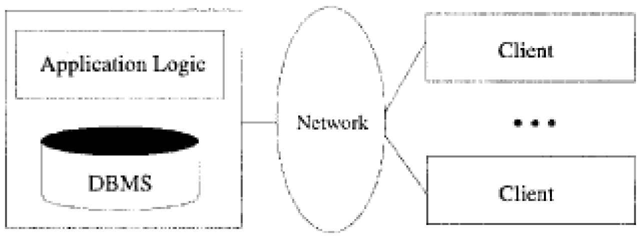
Single-tier architectures have a,n important drawback: Users expect graphical interfaces that require much more computational power than simple dumb terminals. Centralized computation of the graphical displays of such interfaces requires much more computational power than a single server have available, and thus single-tier architectures do not scale to thousands of users. The commoditization of the PC and the availability of cheap client computers led to the development of the two-tier architecture

Two-tier architectures, often also referred to a client-server architectures, consist of a client computer and a server computer, which interact through a well-defined protocol. What part of the functionality the client implements, and what part is left to the server, can vary. In the traditional client-server architecture, the client implements just the graphical user interface, and the server.

The thick-client model has several disadvantages when compared to the thin client model. First, there is no central place to update and maintain the business logic, since the application code runs at many client sites. Second, a large amount of trust is required between the server and the clients.



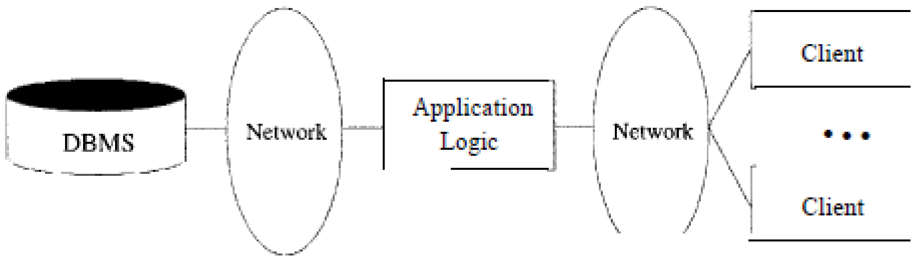
Single Tier



Two Tier

Three Tier Architectures

- Presentation Tier: Users require a natural interface to make requests, provide input, and to see results. The widespread use of the Internet has made Web-based interfaces increasingly popular.
- Middle Tier: The application logic executes here. An enterprise-class application reflects complex business processes, and is coded in a general purpose language such as C++ or Java.
- Data Management Tier: Data-intensive Web applications involve DBMSs, which are the subject of this book.



At the presentation layer, we need to provide forms through which the user can issue requests, and display responses that the middle tier generates.

The middle layer runs code that implements the business logic of the application: It controls what data needs to be input before an action can be executed, determines the control flow between multi-action steps, controls access to the database layer, and often assembles dynamically generated HTML pages from database query results.

Advantages of the Three-Tier Architecture

Heterogeneous Systems: Applications can utilize the strengths of different platforms and different software components at the different tiers. It is easy to modify or replace the code at any tier without affecting the other tiers.

- Thin Clients: Clients only need enough computation power for the presentation layer. Typically, clients are Web browsers.

- Integrated Data Access: In many applications, the data must be accessed from several sources. This can be handled transparently at the middle tier, where we can centrally manage connections to all database systems involved.
- Scalability to Many Clients: Each client is lightweight and all access to the system is through the middle tier. The middle tier can share database connections across clients, and if the middle tier becomes the bottle-neck, we can deploy several servers executing the middle tier code;

5. Give the list of client tier technologies used in presentation Layer. Describe with suitable Example. [10]MARKS

HTML forms are a common way of communicating data from the client tier to the middle tier. The general format of a form is the following:

```
<FORM ACTION="page.jsp" METHOD="GET" NAME="LoginForm">
</FORM>
```

A single HTML document can contain more than one form. Inside an HTML form, we can have any HTML tags except another FORM element.

The FORM tag has three important attributes:

- ACTION: Specifies the URI of the page to which the form contents are submitted; if the ACTION attribute is absent, then the URI of the current page is used. In the sample above, the form input would be submitted to the page named page. j sp, which should provide logic for processing the input from the form
- METHOD: The HTTP/1.0 method used to submit the user input from the filled-out form to the web server. There are two choices, GET and POST; we postpone their discussion to the next section.
- NAME: This attribute gives the form a name. Although not necessary, naming forms is good style. The client-side programs in JavaScript that refer to forms by name and perform checks on form fields.

Inside HTML forms, the INPUT, SELECT, and TEXTAREA tags are used to specify user input elements; a form can have many elements of each type. The simplest user input element is an INPUT field, a standalone tag with no terminating tag. An example of an INPUT tag is the following:

```
<INPUT TYPE=”text" NAME="title">
```

Passing Arguments to Server-Side Scripts

Using the GET method gives users the opportunity to bookmark the page with the constructed URI and thus directly jump to it in subsequent sessions; this is not possible with the POST method. The choice of GET versus POST should be determined' by the application and its requirements

JavaScript

JavaScript is often used for the following types of computation at the client:

- Browser Detection: JavaScript can be used to detect the browser type and load a browser-specific page.
- Form Validation: JavaScript is used to perform simple consistency checks on form fields. For example, a JavaScript program might check whether form input that asks for an email address contains the character '@,' or if all required fields have been input by the user.
- Browser Control: This includes opening pages in customized windows; examples include the annoying pop-up advertisements that you see at many websites, which are programmed using JavaScript.

JavaScript is usually embedded into an HTML document with a special tag, the SCRIPT tag.

```
<SCRIPT LANGUAGE=" JavaScript">
 alert(" Welcome to our bookstore");
 </SCRIPT
```

Style Sheets

```
BODY {BACKGROUND-COLOR: yellow}
H1 {FONT-SIZE: 36pt}
H3 {COLOR: blue}
P {MARGIN-LEFT: 50px; COLOR: red}
```

The use of style sheets has many advantages. First, we can reuse the same document many times and display it differently depending on the context. Second, we can tailor the display to the reader's preference such as font size, color style, and even level of detail. Third, we can deal with different output formats, such as different output devices.

6. Explain the informal guidelines used as a measures to determine the quality of relation schema design. [10]

Informal Design Guidelines for Relation Schemas

Informal guidelines that may be used as measures to determine the quality of relation
schema design:

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples

- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Imparting Clear Semantics to Attributes in Relations

The semantics of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple

Guideline 1. Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation. Intuitively, if a relation schema corresponds to one entity type or one relationship type, it is straightforward to explain its meaning. Otherwise, if the relation
corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

Redundant Information in Tuples and Update Anomalies

Storing natural joins of base relations leads to an additional problem referred to as update anomalies. These can be classified into insertion anomalies, deletion anomalies, and modification anomalies.

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Guideline 2. Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.

NULL Values in Tuples

i. Attributes that are NULL frequently could be placed in separate relations (with the primary key)
ii. Reasons for nulls:

  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable

Guideline 3. As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation

Generation of Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

Guideline 4. Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated.

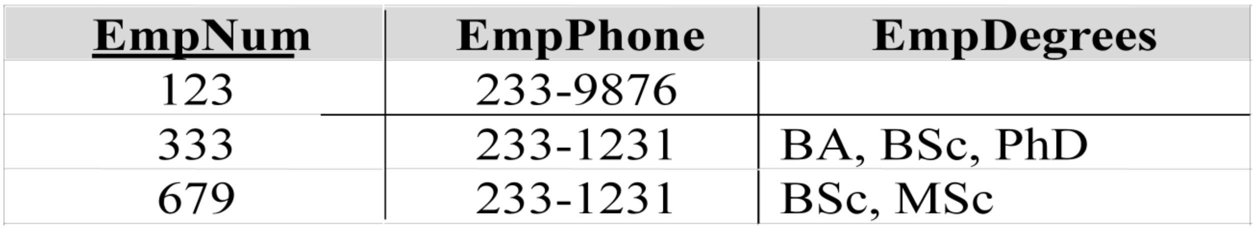
7. Define Normal Form. Explain 1NF, 2NF, 3NF with suitable Examples. [10]MARKS

- Normalization: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- Normal form: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

  – A superkey of a relation schema *R* = {A_1, A_2, ..., A_n} is a set of attributes *S* *subset-of* *R* with the property that no two tuples t_1 and t_2 in any legal relation state *r* of *R* will have $t_1[S] = t_2[S]$
  – A **key** *K* is a superkey with the *additional property* that removal of any attribute from *K* will cause *K* not to be a superkey any more.

First Normal Form

Disallows composite attributes, multivalued attributes, and **nested relations**; attributes whose values *for an individual tuple* are non-atomic



EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

Second Normal Form

– Uses the concepts of **FD**s, **primary key**

<u>Definitions:</u>

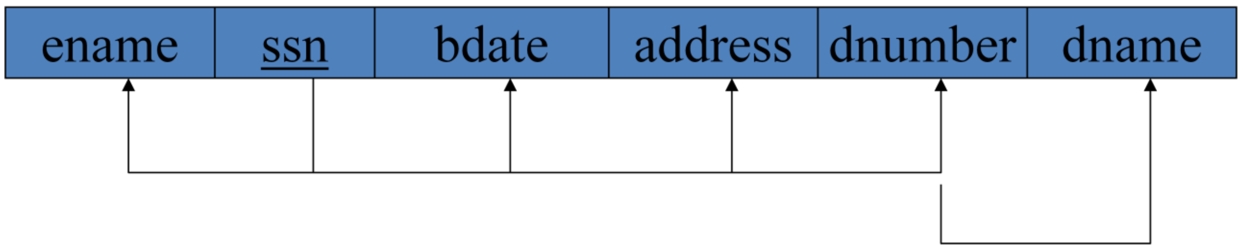
– **Prime attribute -** attribute that is member of the primary key K

– **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

– A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on the primary key

– R can be decomposed into 2NF relations via the process of 2NF normalization

EmployeeDept

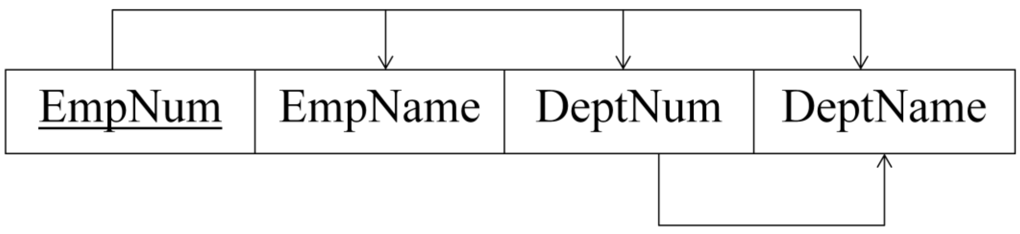


Third Normal Form

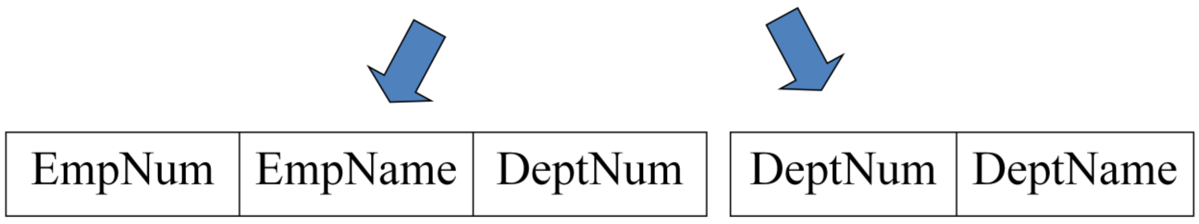
– **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

– A relation schema R is in **third normal form** (**3NF**) if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key

– R can be decomposed into 3NF relations via the process of 3NF normalization.



We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.



8. Write the Short Note on [10]MARKS

i)Triggers

- Objective: to monitor a database and take action when a condition occurs
- Triggers are expressed in a syntax similar to assertions and include the following:
  - event (e.g., an update operation)
  - condition
  - action (to be taken when the condition is satisfied)
- A trigger to compare an employee's salary to his/her supervisor during insert or update operations:

```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
        SALARY, SUPERVISOR_SSN ON EMPLOYEE
        FOR EACH ROW
                WHEN
                (NEW.SALARY> (SELECT SALARY FROM EMPLOYEE
                 WHERE SSN=NEW.SUPERVISOR_SSN))
                INFORM_SUPERVISOR (NEW.SUPERVISOR_SSN,NEW.SSN;
```

ii) Aggregate functions

SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing *aggregate values* such as MIN and SUM. These features represent a significant extension of relational algebra. SQL supports five aggregate operations, which can be applied on any column, say A, of a relation:
1. COUNT ([DISTINCT] A): The number of (unique) values in the A column.
2. SUM ([DISTINCT] A): The sum of all (unique) values in the A column.
3. AVG ([DISTINCT] A): The average of all (unique) values in the A column.
4. MAX (A): The maximum value in the A column.

5. MIN (A): The minimum value in the A column.

SELECT AVG (S.age) FROM Sailors S

iii)Nested Queries

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- The comparison operator **IN** compares a value v with a set (or multi-set) of values V, and evaluates to **TRUE** if v is one of the elements in V
- In general, we can have several levels of nested queries
- A reference to an *unqualified attribute* refers to the relation declared in the *innermost nested query*
- In this example, the nested query is *not correlated* with the outer query
- If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query* , the two queries are said to be *correlated*
- The result of a correlated nested query is *different for each tuple (or combination of tuples) of the relation(s) the outer query*

```
SELECT      E.FNAME, E.LNAME
        FROM        EMPLOYEE AS E
        WHERE       E.SSN IN (SELECT   ESSN
                    FROM DEPENDENT
                    WHERE       ESSN=E.SSN AND
                            E.FNAME=DEPENDENT_NAME
```

iv).Views and their updatability

A view is a “virtual” table that is derived from other tables

Allows for limited update operations (since the table may not physically be stored)

Allows full query operations

A convenience for expressing certain operations

SQL command: CREATE VIEW

a table (view) name

a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)

a query to specify the table contents

```
CREATE View WORKS_ON_NEW AS
        SELECT FNAME, LNAME, PNAME, HOURS
                FROM EMPLOYEE, PROJECT, WORKS_ON
        WHERE SSN=ESSN AND PNO=PNUMBER
```

GROUP BY PNAME;

v) Group by and having clause

we have applied aggregate operations to all (qualifying) rows in a relation. Often we want to apply aggregate operations to each of a number of groups of rows in a relation, where the number of groups depends on the relation instance (i.e., is not known in advance). For example, consider the following query.

we need a major extension to the basic SQL query form, namely, the GROUP BY clause. In fact, the extension also includes an optional HAVING clause that can be used to specify qualifications over groups
(for example, we may be interested only in rating levels> 6.)

The general form
of an SQL query with these extensions is:

```
SELECT [ DISTINCT] select-list
FROM from-list
WHERE 'qualification
GROUP BY grouping-list
HAVING group-qualification
```

```
SELECT S.rating, MIN (S.age)  FROM Sailors S  GROUP BY S.rating
```

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING
COUNT (*) > 1
```

CO's to PO's & PSO's mapping(DBMS-17CS53)

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Identify, analyze and define database objects, enforce integrity constraints on a database using RDBMS.	1,2	3	2	2	-	2	-	-	-	-	-	2	-	2	-	-	-
CO2	Use Structured Query Language (SQL) for database manipulation.	2,3	2	2	3	-	3	-	-	-	-	-	2	-	3	-	-	3
CO3	Formulate, using relational algebra, solutions to a broad range of query problems.	2	3	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CO4	Demonstrate an understanding of normalization theory and apply such knowledge to the normalization of a database	4	2	3	3	3	2	-	-	-	-	-	2	-	2	-	-	3
CO5	Design, build database systems and Develop application to interact with databases	1,2,3,4	3	3	3	3	3	-	-	-	3	-	2	-	3	-	-	3
CO6	Summarize the database recovery techniques and transaction processing	5	2	2	-	-	3	2	-	-	-	-	-	-	-	-	-	3

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Implement and maintain enterprise solutions using latest technologies.				
PSO2	Develop and simulate *wired & wireless network protocols* for various network applications using modern tools.				

PSO3	Apply the knowledge of Information technology and software testing to maintain legacy systems.
PSO4	Apply knowledge of web programming and design to develop *web based applications* using database and other technologies.