SQL DML, Physical Data Organization

SALIM A.

Professor
Department of CSE
College of Engineering Trivandrum

February 24, 2025

Course Outcomes

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)
- CO2: Model real world scenarios given as informal descriptions, using Entity Relationship diagrams. (Cognitive Knowledge Level: Apply)

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)
- CO2: Model real world scenarios given as informal descriptions, using Entity Relationship diagrams. (Cognitive Knowledge Level: Apply)
- CO3:Model and design solutions for efficiently representing and querying data using relational model (Cognitive Knowledge Level: Analyze)

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)
- CO2: Model real world scenarios given as informal descriptions, using Entity Relationship diagrams. (Cognitive Knowledge Level: Apply)
- CO3:Model and design solutions for efficiently representing and querying data using relational model (Cognitive Knowledge Level: Analyze)
- CO4:Demonstrate the features of indexing and hashing in database applications (Cognitive Knowledge Level: Apply)

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)
- CO2: Model real world scenarios given as informal descriptions, using Entity Relationship diagrams. (Cognitive Knowledge Level: Apply)
- CO3:Model and design solutions for efficiently representing and querying data using relational model (Cognitive Knowledge Level: Analyze)
- CO4:Demonstrate the features of indexing and hashing in database applications (Cognitive Knowledge Level: Apply)
- CO5:Discuss and compare the aspects of Concurrency Control and Recovery in Database systems (Cognitive Knowledge Level: Apply)

Course Outcomes

- CO1: Summarize and exemplify fundamental nature and characteristics of database systems (Cognitive Knowledge Level: Understand)
- CO2: Model real world scenarios given as informal descriptions, using Entity Relationship diagrams. (Cognitive Knowledge Level: Apply)
- CO3:Model and design solutions for efficiently representing and querying data using relational model (Cognitive Knowledge Level: Analyze)
- CO4:Demonstrate the features of indexing and hashing in database applications (Cognitive Knowledge Level: Apply)
- CO5:Discuss and compare the aspects of Concurrency Control and Recovery in Database systems (Cognitive Knowledge Level: Apply)
- CO6:Explain various types of NoSQL databases (Cognitive Knowledge Level: Understand)

Company: Schema

Company: Schema

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Figure: Company Schema

Company: Schema

Company: Schema

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

SQL: Syntax

SQL: Syntax

- SQL stands for ‘Structured Query Language’

SQL: Syntax

- SQL stands for ‘Structured Query Language’
- First developed in 1970s by two scientists at IBM following a theory of ‘relational algebra’ by Edgar F. Codd.

SQL: Syntax

- SQL stands for ‘Structured Query Language’
- First developed in 1970s by two scientists at IBM following a theory of ‘relational algebra’ by Edgar F. Codd.
- SQL is specialized to handle ‘structured data’ that follows relational model

SQL: Syntax

- SQL stands for 'Structured Query Language'
- First developed in 1970s by two scientists at IBM following a theory of 'relational algebra' by Edgar F. Codd.
- SQL is specialized to handle 'structured data' that follows relational model
- Rigid and structural: requires rigid predefined schema as compared with those of 'noSQL'

The general form of SQL

SQL: Syntax

- SQL stands for 'Structured Query Language'
- First developed in 1970s by two scientists at IBM following a theory of 'relational algebra' by Edgar F. Codd.
- SQL is specialized to handle 'structured data' that follows relational model
- Rigid and structural: requires rigid predefined schema as compared with those of 'noSQL'

The general form of SQL

- SELECT column
  FROM table
  WHERE predicate on rows
  GROUP BY columns
  HAVING predicate on groups
  ORDER BY columns

SQL: Simple Queries

SQL: Simple Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

SQL: Simple Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
- SELECT BDATE, ADDRESS
  FROM EMPLOYEE
  WHERE FNAME='John' AND MINIT='B' AND
  LNAME='Smith'

SQL: Simple Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
- SELECT BDATE, ADDRESS
  FROM EMPLOYEE
  WHERE FNAME='John' AND MINIT='B' AND
  LNAME='Smith'
- Similar to a SELECT-PROJECT-JOIN of relational algebra operations:
  - The SELECT clause specifies the projection attributes
  - The WHERE clause specifies the selection condition
  - The corresponding relational algebra expression:
    $\pi_{BDATE,ADDRESS}(\sigma_{FNAME='John' \wedge MINIT='B' \wedge LNAME='Smith'}(EMPLOYEE$

SQL: Simple Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
- SELECT BDATE, ADDRESS
  FROM EMPLOYEE
  WHERE FNAME='John' AND MINIT='B' AND
  LNAME='Smith'
- Similar to a SELECT-PROJECT-JOIN of relational algebra operations:
  - The SELECT clause specifies the projection attributes
  - The WHERE clause specifies the selection condition
  - The corresponding relational algebra expression:
    $\pi_{BDATE,ADDRESS}(\sigma_{FNAME='John' \wedge MINIT='B' \wedge LNAME='Smith'}(EMPLOYEE$
- However, the result of the query may contain duplicate tuples

SQL: Simple Queries

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
- SELECT BDATE, ADDRESS
  FROM EMPLOYEE
  WHERE FNAME='John' AND MINIT='B' AND
  LNAME='Smith'
- Similar to a SELECT-PROJECT-JOIN of relational algebra operations:
  - The SELECT clause specifies the projection attributes
  - The WHERE clause specifies the selection condition
  - The corresponding relational algebra expression:
    $\pi_{BDATE,ADDRESS}(\sigma_{FNAME='John' \wedge MINIT='B' \wedge LNAME='Smith'}(EMPLOYEE$
- However, the result of the query may contain duplicate tuples
- Retrieve the name and address of all employees who work for the 'Research' department.

SQL: Simple Queries

SQL: Simple Queries

- Retrieve the name and address of all employees who work for the 'Research' department.

SQL: Simple Queries

- Retrieve the name and address of all employees who work for the 'Research' department.
- SELECT FNAME, LNAME, ADDRESS
  FROM EMPLOYEE, DEPARTMENT
  WHERE DNAME='Research' AND DNUMBER=DNO

SQL: Simple Queries

- Retrieve the name and address of all employees who work for the 'Research' department.
- SELECT FNAME, LNAME, ADDRESS
  FROM EMPLOYEE, DEPARTMENT
  WHERE DNAME='Research' AND DNUMBER=DNO
- A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name
  - DNUMBER is both DEPARTMENT and DEPT_LOCATIONS.
  - Qualify by relationname: DEPARTMENT.DNUMBER, DEPT_LOCATIONS.DNUMBER

SQL: Simple Queries

SQL: Simple Queries

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

SQL: Simple Queries

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.
- SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
  FROM PROJECT,DEPARTMENT, EMPLOYEE,
  WHERE DNUM=DNUMBER AND MGSSSN=SSN AND
  PLOCATION ='STAFFORD'

SQL: Simple Queries

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.
- SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
  FROM PROJECT,DEPARTMENT, EMPLOYEE,
  WHERE DNUM=DNUMBER AND MGSSSN=SSN AND
  PLOCATION ='STAFFORD'
- The join condition DNUM=DNUMBER relates a project to its controlling department

SQL: Simple Queries

- For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.
- SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
  FROM PROJECT,DEPARTMENT, EMPLOYEE,
  WHERE DNUM=DNUMBER AND MGSSSN=SSN AND
  PLOCATION ='STAFFORD'
- The join condition DNUM=DNUMBER relates a project to its controlling department
- The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

SQL: ALIASES

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE E S
    WHERE E.SUPERSSN=S.SSN

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE E S
    WHERE E.SUPERSSN=S.SSN
- The alternate relation names **E** and **S** are called **aliases** or **tuple variables** for the EMPLOYEE relation

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE E S
    WHERE E.SUPERSSN=S.SSN
- The alternate relation names **E** and **S** are called **aliases** or **tuple variables** for the EMPLOYEE relation
- We can think of **E** and **S** as two different copies of EMPLOYEE; **E** represents employees in role of supervisees and **S** represents employees in role of supervisors

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE E S
    WHERE E.SUPERSSN=S.SSN
- The alternate relation names **E** and **S** are called **aliases** or **tuple variables** for the EMPLOYEE relation
- We can think of **E** and **S** as two different copies of EMPLOYEE; **E** represents employees in role of supervisees and **S** represents employees in role of supervisors
- We can also write the above query using **AS**

SQL: ALIASES

- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE E S
    WHERE E.SUPERSSN=S.SSN
- The alternate relation names **E** and **S** are called **aliases** or **tuple variables** for the EMPLOYEE relation
- We can think of **E** and **S** as two different copies of EMPLOYEE; **E** represents employees in role of supervisees and **S** represents employees in role of supervisors
- We can also write the above query using **AS**
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM EMPLOYEE AS E, EMPLOYEE AS S
    WHERE E.SUPERSSN=S.SSN

SQL: UNSPECIFIED WHERE-clause

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected
- If more than one relation is specified in the FROM-clause and there is **no join condition**, then CARTESIAN PRODUCT of tuples is selected

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected
- If more than one relation is specified in the FROM-clause and there is **no join condition**, then CARTESIAN PRODUCT of tuples is selected
- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected
- If more than one relation is specified in the FROM-clause and there is **no join condition**, then CARTESIAN PRODUCT of tuples is selected
- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes
  - SELECT * FROM EMPLOYEE WHERE DNO=5

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected
- If more than one relation is specified in the FROM-clause and there is **no join condition**, then CARTESIAN PRODUCT of tuples is selected
- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes
  - SELECT * FROM EMPLOYEE WHERE DNO=5
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used

SQL: UNSPECIFIED WHERE-clause

- A missing WHERE-clause indicates **no condition**; hence, all tuples of the relations in the FROM-clause are selected
- If more than one relation is specified in the FROM-clause and there is **no join condition**, then CARTESIAN PRODUCT of tuples is selected
- To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes
  - SELECT * FROM EMPLOYEE WHERE DNO=5
- To eliminate duplicate tuples in a query result, the keyword DISTINCT is used
  - SELECT SELECT DISTINCT SALARY
    FROM EMPLOYEE

SQL: UNION

SQL: UNION

- UNION, MINUS, INTERSECT are SET operations

SQL: UNION

- UNION, MINUS, INTERSECT are SET operations
- The set operations apply only to union compatible relations; the two relations must have the **same attributes and the attributes must appear in the same order**

SQL: UNION

- UNION, MINUS, INTERSECT are SET operations
- The set operations apply only to union compatible relations; the two relations must have the **same attributes and the attributes must appear in the same order**
- Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

SQL: UNION

- UNION, MINUS, INTERSECT are SET operations
- The set operations apply only to union compatible relations; the two relations must have the **same attributes and the attributes must appear in the same order**
- Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.
  - (SELECT PNUMBER, PNAME
    FROM PROJECT, DEPARTMENT EMPLOYEE
    WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
    LNAME='Smith')
    UNION

SQL: UNION

- UNION, MINUS, INTERSECT are SET operations
- The set operations apply only to union compatible relations; the two relations must have the **same attributes and the attributes must appear in the same order**
- Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.
  - (SELECT PNUMBER, PNAME
    FROM PROJECT, DEPARTMENT EMPLOYEE
    WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
    LNAME='Smith')
    UNION
    (SELECT PNUMBER, PNAME
    FROM PROJECT, WORKS_ON, EMPLOYEE
    WHERE PNUMBER=PNO AND ESSN=SSN AND
    NAME='Smith')

SQL: NESTING OF QUERIES

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)
- Retrieve the name and address of all employees who work for the 'Research' department.

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM EMPLOYEE
    WHERE DNO IN

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM EMPLOYEE
    WHERE DNO IN
    (SELECT DNUMBER
    FROM DEPARTMENT
    WHERE DNAME='Research')

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM EMPLOYEE
    WHERE DNO IN
    (SELECT DNUMBER
    FROM DEPARTMENT
    WHERE DNAME='Research')
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of the nested query

SQL: NESTING OF QUERIES

- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query(Outer Query)
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM EMPLOYEE
    WHERE DNO IN
    (SELECT DNUMBER
    FROM DEPARTMENT
    WHERE DNAME='Research')
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of the 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**

SQL: CORRELATED NESTED QUERIES

SQL: CORRELATED NESTED QUERIES

- 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

SQL: CORRELATED NESTED QUERIES

- 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
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

SQL: CORRELATED NESTED QUERIES

- 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
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE E.SSN IN

SQL: CORRELATED NESTED QUERIES

- 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
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE E.SSN IN
    (SELECT ESSN
    FROM DEPENDENT AS D
    WHERE E.Sex = D.Sex AND
    E.FNAME=D.DEPENDENT_NAME)

SQL: CORRELATED NESTED QUERIES

- 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
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE E.SSN IN
    (SELECT ESSN
    FROM DEPENDENT AS D
    WHERE E.Sex = D.Sex AND
    E.FNAME=D.DEPENDENT_NAME)
- For each employee tuple, evaluate the nested query, which retrieves the ESSN value of of **all DEPENDENT tuple with same sex and first name as EMPLOYEE tuple**, if the SSN value of EMPLOYEE tuple is in the result of the nested query, then select the EMPLOYEE tuple.

SQL: CORRELATED NESTED QUERIES

- 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
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE E.SSN IN
    (SELECT ESSN
    FROM DEPENDENT AS D
    WHERE E.Sex = D.Sex AND
    E.FNAME=D.DEPENDENT_NAME)
- For each employee tuple, evaluate the nested query, which retrieves the ESSN value of of **all DEPENDENT tuple with same sex and first name as EMPLOYEE tuple**, if the SSN value of EMPLOYEE tuple is in the result of the nested query, then select the EMPLOYEE tuple.
- **Evaluated once for each tuple in the outer query**

SQL: =ANY(or =SOME)

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.
- The =ANY(or =SOME) operator returns TRUE if the value **v** is equal to some value in the set **V**, and hence equivalent to **IN**.

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.
- The =ANY(or =SOME) operator returns TRUE if the value **v** is equal to some value in the set **V**, and hence equivalent to **IN**.
- $<, \leq, >, \geq,$ *and* $<>$ can be combined with =ANY(or =SOME). The keyword ALL can also be combined with these operators.

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.
- The =ANY(or =SOME) operator returns TRUE if the value **v** is equal to some value in the set **V**, and hence equivalent to **IN**.
- $<, \leq, >, \geq,$ *and* $<>$ can be combined with =ANY(or =SOME). The keyword ALL can also be combined with these operators.
- Retreive names of all employees whose salary is greater than the salary of all employees in department 5

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.
- The =ANY(or =SOME) operator returns TRUE if the value **v** is equal to some value in the set **V**, and hence equivalent to **IN**.
- $<, \leq, >, \geq,$ *and* $<>$ can be combined with =ANY(or =SOME). The keyword ALL can also be combined with these operators.
- Retreive names of all employees whose salary is greater than the salary of all employees in department 5
  - SELECT FNAME, LNAME
    FROM EMPLOYEE
    WHERE SALARY $\geq$ *ALL*

SQL: =ANY(or =SOME)

- In addition to **IN**, other operators like =ANY(or =SOME) can also be used for multiset comparison.
- The =ANY(or =SOME) operator returns TRUE if the value **v** is equal to some value in the set **V**, and hence equivalent to **IN**.
- $<, \leq, >, \geq,$ *and* $<>$ can be combined with =ANY(or =SOME). The keyword ALL can also be combined with these operators.
- Retreive names of all employees whose salary is greater than the salary of all employees in department 5
  - SELECT FNAME, LNAME
    FROM EMPLOYEE
    WHERE SALARY $\geq$ *ALL*
    (SELECT SALARY
    FROM EMPLOYEE
    WHERE DNO=5)

SQL: EXISTS

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.
- EXISTS and NOT EXISTS are usually used in conjunction with the correlated nested queries.

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.
- EXISTS and NOT EXISTS are usually used in conjunction with the correlated nested queries.
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.
- EXISTS and NOT EXISTS are usually used in conjunction with the correlated nested queries.
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.
- EXISTS and NOT EXISTS are usually used in conjunction with the correlated nested queries.
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS
    (SELECT *
    FROM DEPENDENT AS D
    WHERE E.SSN = D.ESSN AND E.SEX = D.SEX
    AND
    E.FNAME=D.DEPENDENT_NAME)

SQL: EXISTS

- The EXISTS in SQL is used to check wther the result of a correlated nested query is empty or not.
- EXISTS and NOT EXISTS are usually used in conjunction with the correlated nested queries.
- Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS
    (SELECT *
    FROM DEPENDENT AS D
    WHERE E.SSN = D.ESSN AND E.SEX = D.SEX
    AND
    E.FNAME=D.DEPENDENT_NAME)
- For each employee tuple, evaluate the nested query, which retrieves **all DEPENDENT tuple with same SSN, sex and first name as the EMPLOYEE tuple**, if at least one tuple EXISTS in the result of the nested query, then select the EMPLOYEE tuple.

SQL: NOT EXISTS

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.
- NOT EXISTS return TRUE if there is no tuples in the result of nested query

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.
- NOT EXISTS return TRUE if there is no tuples in the result of nested query
- Retrieve the names of employees who have no dependents

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.
- NOT EXISTS return TRUE if there is no tuples in the result of nested query
- Retrieve the names of employees who have no dependents
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.
- NOT EXISTS return TRUE if there is no tuples in the result of nested query
- Retrieve the names of employees who have no dependents
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE E.SSN = ESSN)

SQL: NOT EXISTS

- EXISTS return true **TRUE** if there is at least one tuple in the result of the nested query, else return **FALSE**.
- NOT EXISTS return TRUE if there is no tuples in the result of nested query
- Retrieve the names of employees who have no dependents
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE E.SSN = ESSN)
- For each EMPLOYEE tuple, the subquery selects all DEPENDENT tuples whose ESSN value matches the EMPLOYEE SSN. If subquery **results is empty**, select the EMPLOYEE tuple and retrieve FNAME, LNAME.

SQL: NOT EXISTS

SQL: NOT EXISTS

- List the names of managers who have at least one dependent

SQL: NOT EXISTS

- List the names of managers who have at least one dependent
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS

SQL: NOT EXISTS

- List the names of managers who have at least one dependent
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE E.SSN = ESSN)
    AND

SQL: NOT EXISTS

- List the names of managers who have at least one dependent
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE E.SSN = ESSN)
    AND
    EXISTS (SELECT *
    FROM DEPARTMENT
    WHERE SSN = MGRSSN);

SQL: NOT EXISTS

- List the names of managers who have at least one dependent
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE AS E
    WHERE EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE E.SSN = ESSN)
    AND
    EXISTS (SELECT *
    FROM DEPARTMENT
    WHERE SSN = MGRSSN);
- Two nested correlated subqueries:
  - 1. Select all DEPENDENT tuples related to EMPLOYEE
  - 2. Select all DEPARTMENT tuples managed by the EMPLOYEE
  - If at least one tuple each first and second subquery exists, then that EMPLOYEE tuple is selected.

SQL: NOT EXISTS

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (
    (SELECT PNUMBER
    FROM PROJECT
    WHERE DNUM = 5)
    EXCEPT

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (
    (SELECT PNUMBER
    FROM PROJECT
    WHERE DNUM = 5)
    EXCEPT
    (SELECT PNO
    FROM WORKS_ON
    WHERE SSN = ESSN);

SQL: NOT EXISTS

- Retrieve names of each employee who works on all the projetcs controlled by department 5
  - SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE
    WHERE NOT EXISTS
    (
    (SELECT PNUMBER
    FROM PROJECT
    WHERE DNUM = 5)
    EXCEPT
    (SELECT PNO
    FROM WORKS_ON
    WHERE SSN = ESSN);
- Two nested correlated subqueries:
  - 1. Select all projects controlled by department 5
  - 2. Select all projects that the particular employee being works on
  - If the set difference of first minus second subquery is empty, then that employee works on all projects

Joined Tables in SQL and Outer Joins

Joined Tables in SQL and Outer Joins

- SQL permits users to specify a table resulting from a join operation in the FROM clause of a query

Joined Tables in SQL and Outer Joins

- SQL permits users to specify a table resulting from a join operation in the FROM clause of a query
- Retrieve the name and address of all employees who work for the 'Research' department.

Joined Tables in SQL and Outer Joins

- SQL permits users to specify a table resulting from a join operation in the FROM clause of a query
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE JOIN DEPARTMENT ON DNo= Dnumber)
    WHERE DNAME ='Research'

Joined Tables in SQL and Outer Joins

- SQL permits users to specify a table resulting from a join operation in the FROM clause of a query
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE JOIN DEPARTMENT ON DNo= Dnumber)
    WHERE DNAME ='Research'
- Here the FROM clause conatins a single joined table.

Joined Tables in SQL and Outer Joins

- SQL permits users to specify a table resulting from a join operation in the FROM clause of a query
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE JOIN DEPARTMENT ON DNo= Dnumber)
    WHERE DNAME ='Research'
- Here the FROM clause conatins a single joined table.
- There are several types of join
  1. NATURAL JOIN: No join conditions is specified between the relations **R** and **S**; *an implicite EQUIJOIN condition for each pair of attributes with the same name from* R *and* S *are created*
  2. OUTER JOIN: When the user wants to keep all tuples of **R**, or all those tuples in **S**, or all those in both relations in the result of the JOIN, regardless of whether or not they have matching tuples in other relations

Joined Tables:NATURAL JOIN

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE NATURAL JOIN
    (DEPARTMENT AS DEPT(DName, DNO, MSsn, MSdate)))
    WHERE DNAME ='Research'

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE NATURAL JOIN
    (DEPARTMENT AS DEPT(DName, DNO, MSsn, MSdate)))
    WHERE DNAME ='Research'
- The default type of join is INNER JOIN, a tuple is added if a matching tuple exists in the other relation

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE NATURAL JOIN
    (DEPARTMENT AS DEPT(DName, DNO, MSsn, MSdate)))
    WHERE DNAME ='Research'
- The default type of join is INNER JOIN, a tuple is added if a matching tuple exists in the other relation
- OUTER JOIN

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE NATURAL JOIN
    (DEPARTMENT AS DEPT(DName, DNO, MSsn, MSdate)))
    WHERE DNAME ='Research'
- The default type of join is INNER JOIN, a tuple is added if a matching tuple exists in the other relation
- OUTER JOIN
- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor. All emplyees are to be included, even if no supervisor

Joined Tables:NATURAL JOIN

- If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then apply NATURAL JOIN
- Retrieve the name and address of all employees who work for the 'Research' department.
  - SELECT FNAME, LNAME, ADDRESS
    FROM (EMPLOYEE NATURAL JOIN
    (DEPARTMENT AS DEPT(DName, DNO, MSsn, MSdate)))
    WHERE DNAME ='Research'
- The default type of join is INNER JOIN, a tuple is added if a matching tuple exists in the other relation
- OUTER JOIN
- For each employee, retrieve the employee's name, and the name of his or her immediate supervisor. All emplyees are to be included, even if no supervisor
  - SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
    FROM (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE
    AS S
    ON E.SUPERSSN=S.SSN)

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

- RIGHT OUTER JOIN: Every tuple in the right table must appear in the result; no matching tuple, padded with NULL values for the attributes in the left table.

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

- RIGHT OUTER JOIN: Every tuple in the right table must appear in the result; no matching tuple, padded with NULL values for the attributes in the left table.
- FULL OUTER JOIN All tuples in both relations will be included, non matching with NULL values.

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

- RIGHT OUTER JOIN: Every tuple in the right table must appear in the result; no matching tuple, padded with NULL values for the attributes in the left table.
- FULL OUTER JOIN All tuples in both relations will be included, non matching with NULL values.
- Multiway JOIN

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

- RIGHT OUTER JOIN: Every tuple in the right table must appear in the result; no matching tuple, padded with NULL values for the attributes in the left table.
- FULL OUTER JOIN All tuples in both relations will be included, non matching with NULL values.
- Multiway JOIN
  - For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

Joined Tables:LEFT OUTER, RIGHT OUTER, FULL OUTER JOIN

- RIGHT OUTER JOIN: Every tuple in the right table must appear in the result; no matching tuple, padded with NULL values for the attributes in the left table.
- FULL OUTER JOIN All tuples in both relations will be included, non matching with NULL values.
- Multiway JOIN
  - For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.
  - SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
    FROM ((PROJECT jOIN DEPARTMENT ON DNUM =DNUMBER)
    JOIN EMPLOYEE ON MGRSSN = SSN)
    WHERE PLOCATION ='STAFFORD'

Aggregate Functions in SQL

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.
- Functions : COUNT, SUM, MAX, MIN, and AVG

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.
- Functions : COUNT, SUM, MAX, MIN, and AVG
- Find the sum of salaries of all employees, the maximum salary, minimum salary and average salary

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.
- Functions : COUNT, SUM, MAX, MIN, and AVG
- Find the sum of salaries of all employees, the maximum salary, minimum salary and average salary
  - SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY)
    FROM EMPLOYEE

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.
- Functions : COUNT, SUM, MAX, MIN, and AVG
- Find the sum of salaries of all employees, the maximum salary, minimum salary and average salary
  - SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY)
    FROM EMPLOYEE
- Find the sum of salaries of all employees of RESEARCH department, as well as the maximum salary, minimum salary and average salary

Aggregate Functions in SQL

- Aggregate functions are used to summarize information from multiple tuples into single tuple summary.
- Functions : COUNT, SUM, MAX, MIN, and AVG
- Find the sum of salaries of all employees, the maximum salary, minimum salary and average salary
  - SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY)
    FROM EMPLOYEE
- Find the sum of salaries of all employees of RESEARCH department, as well as the maximum salary, minimum salary and average salary
  - SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY)
    FROM (EMPLOYEE JOIN DEPARTMENT ON DNO =DNUMBER)
    WHERE DNAME ='RESEARCH'

Aggregate Functions in SQL

Aggregate Functions in SQL

- Find the number of employees in Research department

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')
- Count the number of distinct salary values in the database

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')
- Count the number of distinct salary values in the database
  - SELECT (DISTINCT SALARY)
    FROM EMPLOYEE

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')
- Count the number of distinct salary values in the database
  - SELECT (DISTINCT SALARY)
    FROM EMPLOYEE
- Retrieve names of eployees who have two or more dependents

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')
- Count the number of distinct salary values in the database
  - SELECT (DISTINCT SALARY)
    FROM EMPLOYEE
- Retrieve names of eployees who have two or more dependents
  - SELECT LNAME, FNAME
    FROM EMPLOYEE
    WHERE (SELECT COUNT(*)
    FROM DEPENDENT WHERE SSN=ESSN) ≥ 2

Aggregate Functions in SQL

- Find the number of employees in Research department
  - SELECT COUNT(*)
    FROM (EMPLOYEE, DEPARTMENT
    WHERE DNO=DNUMBER AND DNAME ='RESEARCH')
- Count the number of distinct salary values in the database
  - SELECT (DISTINCT SALARY)
    FROM EMPLOYEE
- Retrieve names of eployees who have two or more dependents
  - SELECT LNAME, FNAME
    FROM EMPLOYEE
    WHERE (SELECT COUNT(*)
    FROM DEPENDENT WHERE SSN=ESSN) ≥ 2
- The above correlated query retieves number of dependents of the employee, which used in the WHERE clause of outer query.

GROUP BY and HAVING

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples
- For each department, retrieve the department number, the number of employees in the department, and their average salary.

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples
- For each department, retrieve the department number, the number of employees in the department, and their average salary.
  - SELECT DNO, COUNT(*), AVG(SALARY)
    FROM EMPLOYEE
    GROUP BY DNO

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples
- For each department, retrieve the department number, the number of employees in the department, and their average salary.
  - SELECT DNO, COUNT(*), AVG(SALARY)
    FROM EMPLOYEE
    GROUP BY DNO
- The SELECT clause include only the grouping attribute and aggregate functions to be applied on each group

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples
- For each department, retrieve the department number, the number of employees in the department, and their average salary.
  - SELECT DNO, COUNT(*), AVG(SALARY)
    FROM EMPLOYEE
    GROUP BY DNO
- The SELECT clause include only the grouping attribute and aggregate functions to be applied on each group
- For each project, retrieve the project number, the project name, and the number of employees who work on this project

GROUP BY and HAVING

- Allows to apply aggregate functions to subgroup of tuples
- For each department, retrieve the department number, the number of employees in the department, and their average salary.
  - SELECT DNO, COUNT(*), AVG(SALARY)
    FROM EMPLOYEE
    GROUP BY DNO
- The SELECT clause include only the grouping attribute and aggregate functions to be applied on each group
- For each project, retrieve the project number, the project name, and the number of employees who work on this project
  - SELECT PNUMBER, PNAME, COUNT(*)
    FROM PROJECT, WORK_ON
    WHERE PNUMBER =PNO
    GROUP BY PNUMBER, PNAME

GROUP BY and HAVING

GROUP BY and HAVING

- For each project on which more than two employees work, retrieve the project number, project name, and the number of employees works on the project.

GROUP BY and HAVING

- For each project on which more than two employees work, retrieve the project number, project name, and the number of employees works on the project.
  - SELECT PNUMBER, PNAME, COUNT(*)
    FROM PROJECT, WORK_ON
    WHERE PNUMBER =PNO
    GROUP BY PNUMBER, PNAME
    HAVING COUNT(*) $>$ 2

GROUP BY and HAVING

- For each project on which more than two employees work, retrieve the project number, project name, and the number of employees works on the project.
  - SELECT PNUMBER, PNAME, COUNT(*)
    FROM PROJECT, WORK_ON
    WHERE PNUMBER =PNO
    GROUP BY PNUMBER, PNAME
    HAVING COUNT(*) $>$ 2
- For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project

GROUP BY and HAVING

- For each project on which more than two employees work, retrieve the project number, project name, and the number of employees works on the project.
  - SELECT PNUMBER, PNAME, COUNT(*)
    FROM PROJECT, WORK_ON
    WHERE PNUMBER =PNO
    GROUP BY PNUMBER, PNAME
    HAVING COUNT(*) $>$ 2
- For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project
  - SELECT PNUMBER, PNAME, COUNT(*)
    FROM PROJECT, WORK_ON, EMPLOYEE
    WHERE PNUMBER =PNO AND SSN=ESSN AND DNO=5
    GROUP BY PNUMBER, PNAME

GROUP BY and HAVING

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000
    GROUP BY DNAME HAVING COUNT(*) $>$ 5

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000
    GROUP BY DNAME HAVING COUNT(*) $>$ 5
- This is incorrect, it will select only departments that have more than employees who can earn more than rs. 40000

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000
    GROUP BY DNAME HAVING COUNT(*) $>$ 5
- This is incorrect, it will select only departments that have more than employees who can earn more than rs. 40000
- Here WHERE is executed first, but HAVING is applied later.

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000
    GROUP BY DNAME HAVING COUNT(*) $>$ 5
- This is incorrect, it will select only departments that have more than employees who can earn more than rs. 40000
- Here WHERE is executed first, but HAVING is applied later.
- The correct query is

GROUP BY and HAVING

- Count the total number of employees whose salary exceeds rs.40000 in each department, but only for departments where more than five employees work
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000
    GROUP BY DNAME HAVING COUNT(*) $>$ 5
- This is incorrect, it will select only departments that have more than employees who can earn more than rs. 40000
- Here WHERE is executed first, but HAVING is applied later.
- The correct query is
  - SELECT DNAME, COUNT(*)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER =DNO AND SALARY $>$ 40000 AND
    (SELECT DNO
    FROM EMPLOYEE
    GROUP BY DNAME
    HAVING COUNT(*) $>$ 5)

SPECIFYING CONSTRAINTS IN SQL

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values
    - DNUM INTEGER **NOT NULL CHECK (DNUM >0 and DNUM < 21);**
    - Here deparment numbers are restricted to integers between 1 to 20.

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values
    - DNUM INTEGER **NOT NULL CHECK (DNUM >0 and DNUM < 21);**
    - Here deparment numbers are restricted to integers between 1 to 20.
    - CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21)
    - Now D_NUM is a new attribute type.

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values
    - DNUM INTEGER **NOT NULL CHECK (DNUM >0 and DNUM < 21);**
    - Here deparment numbers are restricted to integers between 1 to 20.
    - CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21)
    - Now D_NUM is a new attribute type.
- Specifying Key and Referential Integrity Constraints

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values
    - DNUM INTEGER **NOT NULL CHECK (DNUM >0 and DNUM < 21);**
    - Here deparment numbers are restricted to integers between 1 to 20.
    - CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21)
    - Now D_NUM is a new attribute type.
- Specifying Key and Referential Integrity Constraints
  - Primary key and foreign key can be specified along with the table definition, CREATE TABLE

SPECIFYING CONSTRAINTS IN SQL

- Specifying attribute constraints
  - NOT NULL : if NULL not permitted for a particular attribute
  - CHECK : Restrict attribute or domain values
    - DNUM INTEGER **NOT NULL CHECK (DNUM >0 and DNUM < 21);**
    - Here deparment numbers are restricted to integers between 1 to 20.
    - CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21)
    - Now D_NUM is a new attribute type.
- Specifying Key and Referential Integrity Constraints
  - Primary key and foreign key can be specified along with the table definition, CREATE TABLE
  - The UNIQUE clause specifies alternate keys

SPECIFYING CONSTRAINTS IN SQL

SPECIFYING CONSTRAINTS IN SQL

- Foreign key constraints

SPECIFYING CONSTRAINTS IN SQL

- Foreign key constraints
  - The default action that SQL takes for integrity violation is to **reject** the update operation

SPECIFYING CONSTRAINTS IN SQL

- Foreign key constraints
  - The default action that SQL takes for integrity violation is to **reject** the update operation
  - The other options are **SET NULL, CASCADE, SET DEFAULT**

SPECIFYING CONSTRAINTS IN SQL

- Foreign key constraints
  - The default action that SQL takes for integrity violation is to **reject** the update operation
  - The other options are **SET NULL, CASCADE, SET DEFAULT**
  - An action must be qualified with either **ON DELETE** or **ON UPDATE**

SPECIFYING CONSTRAINTS IN SQL

- Foreign key constraints
  - The default action that SQL takes for integrity violation is to **reject** the update operation
  - The other options are **SET NULL, CASCADE, SET DEFAULT**
  - An action must be qualified with either **ON DELETE** or **ON UPDATE**
  - CREATE TABLE EMPLOYEE (

    .

    ESSN CHAR(9),
    DNO INTEGER DEFAULT 1,
    SUPERSSN CHAR(9),
    PRIMARY KEY (ESSN),
    FOREIGN KEY (DNO) REFERENCES DEPARTMENT
    ON DELETE SET DEFAULT
    ON UPDATE CASCADE,
    FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE
    ON DELETE SET NULL
    ON UPDATE CASCADE);

CONSTRAINTS IN SQL: ASSERTIONS

CONSTRAINTS IN SQL: ASSERTIONS

- ASSERTION

CONSTRAINTS IN SQL: ASSERTIONS

- ASSERTION
  - CREATE ASSERTION can be used to create other types of constraints that are outside the scope of built in relational model constraints(primary unique keys, entity integrity, referential integrity.

CONSTRAINTS IN SQL: ASSERTIONS

- ASSERTION
  - CREATE ASSERTION can be used to create other types of constraints that are outside the scope of built in relational model constraints(primary unique keys, entity integrity, referential integrity.
  - Each assertion is given with a constraint name and specified via condition similar to a WHERE clause of SQL query.

CONSTRAINTS IN SQL: ASSERTIONS

- ASSERTION
  - CREATE ASSERTION can be used to create other types of constraints that are outside the scope of built in relational model constraints(primary unique keys, entity integrity, referential integrity.
  - Each assertion is given with a constraint name and specified via condition similar to a WHERE clause of SQL query.
  - Specify a constraint that the salary of an employee must not be greater than the salary of the manager of the department that the employee works for

CONSTRAINTS IN SQL: ASSERTIONS

- ASSERTION
  - CREATE ASSERTION can be used to create other types of constraints that are outside the scope of built in relational model constraints(primary unique keys, entity integrity, referential integrity.
  - Each assertion is given with a constraint name and specified via condition similar to a WHERE clause of SQL query.
  - Specify a constraint that the salary of an employee must not be greater than the salary of the manager of the department that the employee works for
  - CREATE ASSERTION SALARY_CONSTRAINT
    CHECK (NOT EXISTS(SELECT *
    FROM EMPLOYEE E, EMPLOYEE M,
    DEPARTMENT D
    WHERE E.Salary > M.Salary
    AND E.Dno =D.DNumber
    AND D.Mgr_ssn =M.Ssn))

TRIGGERS in SQL

TRIGGERS in SQL

- Triggers specify the action to be taken when certain events occur and when certain conditions are satisfied.

TRIGGERS in SQL

- Triggers specify the action to be taken when certain events occur and when certain conditions are satisfied.
- For example, it is useful to specify a condition that, if violated, causes some users to be informed about the violation.

TRIGGERS in SQL

- Triggers specify the action to be taken when certain events occur and when certain conditions are satisfied.
- For example, it is useful to specify a condition that, if violated, causes some users to be informed about the violation.
- We want to check whenever employees salary is greater than the salary of his or her direct supervisor. Several events can trigger this rule :
  - Insertion of new employee tuple
  - Changing an employee's salary
  - Changing an employee's supervisor

TRIGGERS in SQL

- Triggers specify the action to be taken when certain events occur and when certain conditions are satisfied.
- For example, it is useful to specify a condition that, if violated, causes some users to be informed about the violation.
- We want to check whenever employees salary is greater than the salary of his or her direct supervisor. Several events can trigger this rule :
  - Insertion of new employee tuple
  - Changing an employee's salary
  - Changing an employee's supervisor
- Let the acton required is call an external stored procedure.

TRIGGERS in SQL

TRIGGERS in SQL

- Trigger for whenever employees salary is greater than the salary of his or her direct supervisor.

TRIGGERS in SQL

- Trigger for whenever employees salary is greater than the salary of his or her direct supervisor.
  - CREATE TRIGGER SALARY_VIOLATION
    BEFORE INSERT OR UPDATE OF
    SALARY, SUPERVISOR_SSN ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.SALARY > (SELECT SALARY
    FROM EMPLOYEE
    WHERE SSN=NEW.SUPERVISON_SSN))
    INFORM_SUPERVISOR
    (NEW.SUPERVISOR_SSN, NEW.SSN);

TRIGGERS in SQL

TRIGGERS in SQL

- **Event** : These are database update operations that are explicitly applied to the database. Example: Insert of new employee tuple, change of salary,change of supervisor

TRIGGERS in SQL

- **Event** : These are database update operations that are explicitly applied to the database. Example: Insert of new employee tuple, change of salary,change of supervisor
- The events are specified after the keyword **BEFORE or AFTER** :
  - BEFORE : Trigger should executed before the triggering operation is excuted.
  - AFTER: Trigger should be executed after the specified operation.

TRIGGERS in SQL

- **Event** : These are database update operations that are explicitly applied to the database. Example: Insert of new employee tuple, change of salary,change of supervisor
- The events are specified after the keyword **BEFORE or AFTER** :
  - BEFORE : Trigger should executed before the triggering operation is excuted.
  - AFTER: Trigger should be executed after the specified operation.
- **Condition**: Once the triggering event is occured, the optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs. If a condition is specified, it is evaluated, if true, the rule action be executed.

TRIGGERS in SQL

- **Event** : These are database update operations that are explicitly applied to the database. Example: Insert of new employee tuple, change of salary,change of supervisor
- The events are specified after the keyword **BEFORE or AFTER** :
  - BEFORE : Trigger should executed before the triggering operation is excuted.
  - AFTER: Trigger should be executed after the specified operation.
- **Condition**: Once the triggering event is occured, the optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs. If a condition is specified, it is evaluated, if true, the rule action be executed.
- **Action** : Usually sequence of SQL statements; or it can be a stored procedure. Here INFORM_SUPERVISOR

TRIGGERS in SQL

TRIGGERS in SQL

- Trigger for Trigger to maintain consistency of total salary in Department Table, when new employees are added in Employee table

TRIGGERS in SQL

- Trigger for Trigger to maintain consistency of total salary in Department Table, when new employees are added in Employee table
  - CREATE TRIGGER Total_Sal1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.Dno IS NOT NULL)
    UPDATE DEPARTMENT
    SET Total_Sal= Total_Sal + NEW.Salary
    WHERE Dno=NEW.Dno;

TRIGGERS in SQL

- Trigger for Trigger to maintain consistency of total salary in Department Table, when new employees are added in Employee table
  - CREATE TRIGGER Total_Sal1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.Dno IS NOT NULL)
    UPDATE DEPARTMENT
    SET Total_Sal= Total_Sal + NEW.Salary
    WHERE Dno=NEW.Dno;
- **FOR EACH ROW**: Rule is triggered separetely for each tuple, known as **row-level trigger**. If this clause is left out, trigger is known **statement-level trigger**.

TRIGGERS in SQL

- Trigger for Trigger to maintain consistency of total salary in Department Table, when new employees are added in Employee table
  - CREATE TRIGGER Total_Sal1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.Dno IS NOT NULL)
    UPDATE DEPARTMENT
    SET Total_Sal= Total_Sal + NEW.Salary
    WHERE Dno=NEW.Dno;
- **FOR EACH ROW**: Rule is triggered separetely for each tuple, known as **row-level trigger**. If this clause is left out, trigger is known **statement-level trigger**.
- The **statement-level trigger** is triggered once for triggering statement.

Row-level trigger vs Statement-level trigger

Row-level trigger vs Statement-level trigger

- Consider an update statement

Row-level trigger vs Statement-level trigger

- Consider an update statement
  - UPDATE EMPLOYEE
    SET Salary=1.1*Salary
    WHERE Dno=5

Row-level trigger vs Statement-level trigger

- Consider an update statement
  - UPDATE EMPLOYEE
    SET Salary=1.1*Salary
    WHERE Dno=5
- This operation would be an event for that trigger the **Total_Sal** trigger

Row-level trigger vs Statement-level trigger

- Consider an update statement
  - UPDATE EMPLOYEE
    SET Salary=1.1*Salary
    WHERE Dno=5
- This operation would be an event for that trigger the **Total_Sal** trigger
- The above statement could update multiple records, hence a rule using **row-level semantics** is required to trigger for each row. A **statement-level semantics** is triggered only once, which is not suitable for the above update.

SQL Views

SQL Views

- A view is a single table that is derived from other tables

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.
- The base tables tuples are always physically stored in the database.

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.
- The base tables tuples are always physically stored in the database.
- Create a view to list employee name, project and number of hours worked

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.
- The base tables tuples are always physically stored in the database.
- Create a view to list employee name, project and number of hours worked
  - CREATE VIEW WORKS_ON1
    AS SELECT Fname, Lname, Pname, Hours
    FROM EMPLOYEE, PROJECT, WORKS_ON WHERE Ssn
    =Essn AND Pno=Pnumber

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.
- The base tables tuples are always physically stored in the database.
- Create a view to list employee name, project and number of hours worked
  - CREATE VIEW WORKS_ON1
    AS SELECT Fname, Lname, Pname, Hours
    FROM EMPLOYEE, PROJECT, WORKS_ON WHERE Ssn
    =Essn AND Pno=Pnumber
- Create view of each department with department name, no of employees and total salary

SQL Views

- A view is a single table that is derived from other tables
- A view does not necessarily exist in physical form; it is considered to be a virtul table.
- The base tables tuples are always physically stored in the database.
- Create a view to list employee name, project and number of hours worked
  - CREATE VIEW WORKS_ON1
    AS SELECT Fname, Lname, Pname, Hours
    FROM EMPLOYEE, PROJECT, WORKS_ON WHERE Ssn
    =Essn AND Pno=Pnumber
- Create view of each department with department name, no of employees and total salary
  - CREATE VIEW DEPT_INFO(Dept_name, No_of_emps,
    Total_sal)
    AS SELECT Dname, COUNT(*), SUM(Salary)
    FROM DEPARTMENT, EMPLOYEE
    WHERE Dnumber =Dno
    GROUP BY Dname

SQL Views

SQL Views

- We can specify SQL queries on a view

SQL Views

- We can specify SQL queries on a view
- Retrieve the last name and first name of all employees who work for "ProductX" project.

SQL Views

- We can specify SQL queries on a view
- Retrieve the last name and first name of all employees who work for "ProductX" project.
  - SELECT Fname, Lname
    FROM WORKS_ON1
    WHERE Pname ='ProductX'

SQL Views

- We can specify SQL queries on a view
- Retrieve the last name and first name of all employees who work for "ProductX" project.
  - SELECT Fname, Lname
    FROM WORKS_ON1
    WHERE Pname ='ProductX'
- A view is supposed to be always up-to-date. If we modify the tuples in base tables on which the view is defined, the view must automatically reflect the changes.

SQL Views

- We can specify SQL queries on a view
- Retrieve the last name and first name of all employees who work for "ProductX" project.
  - SELECT Fname, Lname
    FROM WORKS_ON1
    WHERE Pname ='ProductX'
- A view is supposed to be always up-to-date. If we modify the tuples in base tables on which the view is defined, the view must automatically reflect the changes.
- An efficient method for automatic update of view table is **incremental update**, where DBMS determine what new tuples must be inserted, deleted or modified in a materialized view table(physically stored).

SQL Views

- We can specify SQL queries on a view
- Retrieve the last name and first name of all employees who work for "ProductX" project.
  - SELECT Fname, Lname
    FROM WORKS_ON1
    WHERE Pname ='ProductX'
- A view is supposed to be always up-to-date. If we modify the tuples in base tables on which the view is defined, the view must automatically reflect the changes.
- An efficient method for automatic update of view table is **incremental update**, where DBMS determine what new tuples must be inserted, deleted or modified in a materialized view table(physically stored).
- To dispose a view: DROP VIEW WORKS_ON

Update of Views and base tables

Update of Views and base tables

- Update through VIEW is possible, provided:

Update of Views and base tables

- Update through VIEW is possible, provided:
  - A view with single defining table is updatable, if the view attributes contain the primary key of the base relation as well as all the attribute with NOT NULL constraint that do not have default values specified.

Update of Views and base tables

- Update through VIEW is possible, provided:
  - A view with single defining table is updatable, if the view attributes contain the primary key of the base relation as well as all the attribute with NOT NULL constraint that do not have default values specified.
  - Views defined on multiple tables using joins are generally not updatable.

Update of Views and base tables

- Update through VIEW is possible, provided:
  - A view with single defining table is updatable, if the view attributes contain the primary key of the base relation as well as all the attribute with NOT NULL constraint that do not have default values specified.
  - Views defined on multiple tables using joins are generally not updatable.
  - Views defined using grouping and aggregate functions are not updatable.