CSE 311L(Database Management System)

Topics:

- Basic SELECT Statement
- Selecting All Columns, Specific Columns
- Arithmetic Expressions, Using Arithmetic Operators, Parenthesis
- Defining a Column Alias

BASIC QUERIES IN SQL

- SQL has one basic statement for retrieving information from a database; the SLELECT statement
- This is *not the same as* the SELECT operation of the relational algebra
- Important distinction between SQL and the formal relational model;
- SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a *multi-set* (sometimes called a bag) of tuples; it is *not* a set of tuples
- SQL relations can be constrained to be sets by using the CREATE UNIQUE INDEX command, or by using the DISTINCT option
- Basic form of the SQL SELECT statement is called a *mapping* of a *SELECT-FROM-WHERE block*

  SELECT <attribute list> FROM <table list> WHERE <condition>
- <attribute list> is a list of attribute names whose values are to be retrieved by the query
- <table list > is a list of the relation names required to process the query
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

SIMPLE SQL QUERIES

Basic SQL queries correspond to using the following operations of the relational algebra:

SELECT

PROJECT

JOIN

Example of a simple query on one relation (company2.sql)

Basic SELECT Statement

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table;
```

Arithmetic Operators

```
SELECT last_name, salary, 12*(salary+100)
FROM emps;
```

Using Column Aliases

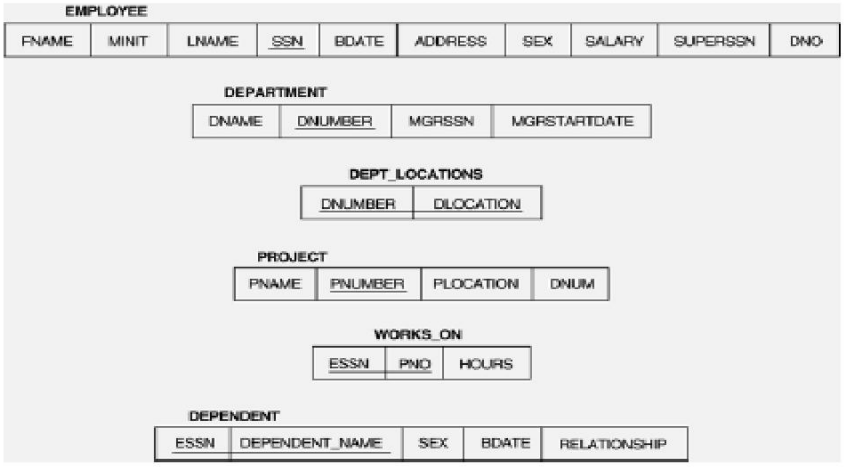
```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM emps;
```

Activity 01:

Write a query that displays the last name , weekly salary, department number of the employees. Name the salary column as "Weekly Salary".

Run: Populate the table with data given and running company,sql in the mysql promt

All subsequent examples uses COMPANY database as shown below:



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
	888665555	20	null

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

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Example of a simple query on one relation

Query 0: Retrieve the birth date and address of the employee whose name is 'John B. Smith'.

```
Q0: SELECT BDATE, ADDRESS FROM EMPLOYEE
WHERE FNAME='John' AND MINIT='B‘ AND LNAME='Smith‘
```

The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition However, the result of the query may contain duplicate tuples

Example of a simple query on two relations

Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND DNUMBER=DNO
```

Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra) (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

Example of a simple query on three relations

Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
Q2: SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS FROM PROJECT,
DEPARTMENT, EMPLOYEE WHERE DNUM=DNUMBER AND MGRSSN=SSN
AND PLOCATION='Stafford'
```

In Q2, there are two join conditions 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

ALIASES, * AND DISTINCT, EMPTY WHERE-CLAUSE

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in different relations
- 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 **Example:** EMPLOYEE.LNAME, DEPARTMENT.DNAME

- Some queries need to refer to the same relation twice. In this case, aliases are given to the relation name

Example

Query 3: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q3: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE E S WHERE E.SUPERSSN=S.SSN

In Q3, 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

Aliasing can also be used in any SQL query for convenience. Can also use the AS keyword to specify aliases

Q3: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE AS E, EMPLOYEE AS S WHERE E.SUPERSSN=S.SSN

UNSPECIFIED WHERE-clause

A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected. This is equivalent to the condition WHERE TRUE

Example:

Query 4: Retrieve the SSN values for all employees.

Q4: SELECT SSN FROM EMPLOYEE

If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

Example:

Q5: SELECT SSN, DNAME FROM EMPLOYEE, DEPARTMENT

Note: It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

USE OF *

To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes

Examples:

Retrieve all the attribute values of EMPLOYEES who work in department 5.

Q1a: SELECT * FROM EMPLOYEE WHERE DNO=5

Retrieve all the attributes of an employee and attributes of DEPARTMENT he works in for every employee of 'Research' department.

Q1b: SELECT * FROM EMPLOYEE, DEPARTMENT WHERE DNAME='Research'
AND DNO=DNUMBER

USE OF DISTINCT

SQL does not treat a relation as a set; duplicate tuples can appear. To eliminate duplicate tuples in a query result, the keyword DISTINCT is used

Example: the result of **Q1c** may have duplicate SALARY values whereas **Q1d** does not have any duplicate values

Q1c: SELECT SALARY FROM EMPLOYEE

Q1d: SELECT **DISTINCT**
SALARY FROM EMPLOYEE

Activity 02:

Find the results in SQL for these queries:

a) Find the first name and Last name of the employees who are supervised by "Franklin Wong"?
b) Find the last and first name of the female employees who have a dependent with the same first name as themselves?
c) For each department find out the department manager's last name, his start date and the name his dependents (if any)?
d) For each employee find out the employee's last and first name, the department name in which he works and the project name he works in and the number of hours he work in those projects.

Restricting and Sorting Data (Part A is based on Company2 schema)

Topics:

- Limiting the Rows Selected
- Restricting with Character Strings and Dates
- Comparison Conditions
- Other Comparison Conditions,

Limiting the Rows Selected

```
SELECT employee_id, last_name, job_id, department_id
FROM emps
WHERE department_id = 90 ;
```

Character Strings and Dates

```
SELECT last_name, job_id, department_id
FROM emps
WHERE last_name = 'WHALEN';
```

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive),
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

```
SELECT last_name, salary
FROM emps
WHERE salary <= 3000;
```

Other (

```
SELECT last_name, salary
FROM emps
WHERE salary BETWEEN 2500 AND 3500;
```

```
SELECT employee_id, last_name, salary, manager_id
FROM emps
WHERE manager_id IN (100, 101, 201);
```

ORDER BY Clause

```
SELECT last_name, job_id, department_id, hire_date
FROM emps
ORDER BY hire_date DESC ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98

Sorting by Multiple Columns

```
SELECT last_name, department_id, salary
FROM emps
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3500
Davies	50	3100
Matos	50	2600
Vargas	50	2500

Activity 03:

Display the employee last name, job ID, and start date of employees hired between February 20, 1998, and May 1, 1998. Order the query in ascending order by start date.

Activity 04:

Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.

SET OPERATIONS (Based on Company.sql)

SQL has directly incorporated some set operations such as union operation (UNION), set difference (MINUS) and intersection (INTERSECT) operations. The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result. 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

Query 5: 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.

```
Q5: (SELECT PNAME FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE
DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')
UNION

(SELECT PNAME FROM PROJECT, WORKS_ON, EMPLOYEE WHERE
PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')
```

NESTING OF QUERIES

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified in an alternative form using nesting

Query 6: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q6: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE WHERE DNO IN
(SELECT DNUMBER FROM DEPARTMENT WHERE DNAME='Research' )
```

Note: The nested query selects the number of the 'Research' department. The outer query selects 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

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. The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

Query 7: Retrieve the name of each employee who has a dependent with the same first name as the employee.

Q7: 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)

In Q7, the nested query has a different result in the outer query. A query written with nested SELECT... FROM… WHERE... blocks and using the **= or IN** comparison operators can *always* be expressed as a single block query. For example, Q7 may be written as in Q7a

Q7a: SELECT E.FNAME, E.LNAME FROM EMPLOYEE E, DEPENDENT D
WHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT_NAME

THE EXISTS FUNCTION

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. We can formulate Query 7 in an alternative form that uses EXIST.

Q7b: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE **EXISTS** (SELECT * FROM DEPENDENT WHERE SSN=ESSN AND FNAME=DEPENDENT_NAME)

Query 8: Retrieve the names of employees who have no dependents.

Q8: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE **NOT EXISTS**

(SELECT * FROM DEPENDENT WHERE SSN=ESSN)

Note: In Q8, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

EXPLICIT SETS

It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query

Query 9: Retrieve the social security numbers of all employees who work on project

number 1, 2, or 3.

Q9: SELECT DISTINCT ESSN FROM WORKS_ON WHERE PNO **IN (1, 2, 3)**

NULLS IN SQL QUERIES

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

Query 10: Retrieve the names of all employees who do not have supervisors.

Q10: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE SUPERSSN IS NULL

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

SUBSTRING COMPARISON

The LIKE comparison operator is used to compare partial strings. Two reserved characters are used: **'%'** (or '*' in some implementations) replaces an arbitrary number of characters, and **'_'** replaces a single arbitrary character.

Query 18: Retrieve all employees whose address is in Houston, Texas. Here, the value of the

ADDRESS attribute must contain the substring 'Houston,TX‘ in it.

Q18: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE ADDRESS LIKE '%Houston,TX%'

Query 19: Retrieve all employees who were born during the 1950s.

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_____5_', with each underscore as a place holder for a single arbitrary character.

Q19: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE BDATE **LIKE** '_____5_‘

Note: The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible. Hence, in SQL, character string attribute values are not atomic

Using the LIKE Condition (based on company2.schema)

- Use the LIKE condition to perform wildcard searches of valid search string values.

- Search conditions can contain either literal characters or numbers:
  - % denotes zero or many characters.
  - _ denotes one character.

```
SELECT last_name
FROM emps
WHERE last_name LIKE '_o%';
```

The ESCAPE Option (based on company2.schema)

```
SELECT employee_id, last_name, job_id
FROM emps
WHERE job_id LIKE '%SA\_%' ESCAPE '\';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

Using the NULL Conditions (based on company2.schema)

```
SELECT last_name, manager_id
FROM emps
WHERE manager_id IS NULL;
```

Logical Conditions

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

```
SELECT employee_id, last_name, job_id, salary
FROM emps
WHERE salary >=10000
AND job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Using the NOT Operator

```
SELECT last_name, job_id
FROM emps
WHERE job_id
NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP

All Activities are based on Company2 Schema

Activity 05:

Display the last name and hire date of every employee who was hired in 1994.

Activity 06:

Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.Title.

Activity 07:

Display the last name of all employees who have an *a* and an *e* in their last name.