



Module 3

Databases

SQL-99: Queries

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the SELECT 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 specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query



Retrieval Queries in SQL (cont.)

- Basic form of the SQL SELECT statement is called a *mapping* or 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



Relational Database Schema--Figure 5.5

EMPLOYEE

FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
-------	----------------	--------	--------------

DEPT_LOCATIONS

<u>DNUMBER</u>	<u>DLOCATION</u>
----------------	------------------

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM
-------	----------------	-----------	------

WORKS_ON

<u>ESSN</u>	<u>PNO</u>	HOURS
-------------	------------	-------

DEPENDENT

<u>ESSN</u>	<u>DEPENDENT_NAME</u>	SEX	BDATE	RELATIONSHIP
-------------	-----------------------	-----	-------	--------------



Populated Database--Fig.5.6

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

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

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

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

Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra
- All subsequent examples use the COMPANY database
- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate 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'
```

- Similar to a SELECT-PROJECT pair of relational algebra operations; 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



Simple SQL Queries (cont.)

- 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)



Simple SQL Queries (cont.)

- 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 birthdate.

```
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.BDATE, DEPENDENT.BDATE

ALIASES

- Some queries need to refer to the same relation twice
- In this case, *aliases* are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

**Q8: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
 FROM EMPLOYEE E, EMPLOYEE S
 WHERE E.SUPERSSN=S.SSN**

- In Q8, 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*

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
- Query 9: Retrieve the SSN values for all employees.

**Q9: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



UNSPECIFIED WHERE-clause (cont.)

- Example:

**Q10: SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT**

- 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:

Q1C: SELECT *
FROM EMPLOYEE
WHERE DNO=5

Q1D: 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
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

**Q11: SELECT SALARY
FROM EMPLOYEE**

	⚡ SALARY
1	30000
2	40000
3	25000
4	43000
5	38000
6	25000
7	25000
8	55000

Q11A

	⚡ SALARY
1	55000
2	38000
3	30000
4	40000
5	43000
6	25000

**Q11A: SELECT DISTINCT SALARY
FROM EMPLOYEE**

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (**UNION**), and in *some versions* of SQL there are 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



SET OPERATIONS (cont.)

- Query 4: Make a list of all project names 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.

```
Q4:  (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 LNAME='Smith')
```




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

**Q14: SELECT FNAME, LNAME
 FROM EMPLOYEE
 WHERE SUPERSSN IS NULL**

	FNAME	LNAME
1	James	Borg

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



NULLS IN SQL QUERIES

- The NVL() function returns a specified value if the expression is NULL.
- If the expression is NOT NULL, this function returns the expression.
 - NVL(*expression*, *alt_value*)
- Query 15: Retrieve the names of all employees and the SSN of their supervisors. In case no supervisor, return “without supervisor”

**Q15: SELECT FNAME, LNAME, NVL(SUPERSSN, ‘without supervisor’)
 FROM EMPLOYEE;**

7	Ahmad	Jabbar	987654321
8	James	Borg	without supervisor

Joined Relations Feature in SQL2

- Can specify a "joined relation" in the FROM-clause
- Looks like any other relation but is the result of a join
- Allows the user to specify different types of joins (regular "theta" JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, etc)



Joined Relations Feature in SQL2 (cont.)

- Q1:

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

- could be written as:

Q1:

```
SELECT FNAME, LNAME, ADDRESS  
FROM EMPLOYEE JOIN DEPARTMENT ON (DNUMBER=DNO)  
WHERE DNAME='Research'
```

or as:

In a natural join the columns names
must have the same name

Q1:

```
SELECT FNAME, LNAME, ADDRESS  
FROM EMPLOYEE NATURAL JOIN (SELECT DNAME, DNUMBER AS DNO FROM DEPARTMENT)  
WHERE DNAME='Research'
```



Joined Relations Feature in SQL2 (cont.)

- Another Example;

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

Q2 could be written as follows; this illustrates multiple joins in the joined tables

- **Q2:** **SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS**
 FROM PROJECT JOIN DEPARTMENT ON (DNUM=DNUMBER) JOIN
 EMPLOYEE ON (MGRSSN=SSN)
 WHERE PLOCATION='Stafford'



Self join

- Join a table with itself
- The use of table aliases is mandatory
- Q16: Employees working in the same department
- **Q16(1):** SELECT E1.FNAME, E1.LNAME, E2.FNAME, E2.LNAME
FROM EMPLOYEE E1, EMPLOYEE E2
WHERE E1.DNO = E2.DNO
- **Q16(2):** SELECT E1.FNAME, E1.LNAME, E2.FNAME, E2.LNAME
FROM EMPLOYEE E1, EMPLOYEE E2
WHERE E1.DNO = E2.DNO AND E1.SSN != E2.SSN

	FNAME	LNAME	FNAME_1	LNAME_1
1	John	Smith	John	Smith
2	Franklin	Wong	John	Smith
3	Ramesh	Narayan	John	Smith
4	Joyce	English	John	Smith
5	John	Smith	Franklin	Wong
6	Franklin	Wong	Franklin	Wong
7	Ramesh	Narayan	Franklin	Wong
8	Joyce	English	Franklin	Wong
9	Alicia	Zelaya	Alicia	Zelaya
10	Jennifer	Wallace	Alicia	Zelaya
11	Ahmad	Jabbar	Alicia	Zelaya
12	Alicia	Zelaya	Jennifer	Wallace
13	Jennifer	Wallace	Jennifer	Wallace

	FNAME	L...	FNAME_1	LNAME_1
1	Franklin	Wong	John	Smith
2	Ramesh	Narayan	John	Smith
3	Joyce	English	John	Smith
4	John	Smith	Franklin	Wong
5	Ramesh	Narayan	Franklin	Wong
6	Joyce	English	Franklin	Wong
7	Jennifer	Wallace	Alicia	Zelaya
8	Ahmad	Jabbar	Alicia	Zelaya
9	Alicia	Zelaya	Jennifer	Wallace
10	Ahmad	Jabbar	Jennifer	Wallace
11	John	Smith	Ramesh	Narayan
12	Franklin	Wong	Ramesh	Narayan
13	Joyce	English	Ramesh	Narayan

Self join

- Q16(2): `SELECT E1.FNAME, E1.LNAME, E2.FNAME, E2.LNAME
FROM EMPLOYEE E1, EMPLOYEE E2
WHERE E1.DNO = E2.DNO AND E1.SNN < E2.SNN`

- Always use the primary key!!!!

	FNAME	LNAME	FNAME_1	LNAME_1
1	John	Smith	Franklin	Wong
2	Jennifer	Wallace	Alicia	Zelaya
3	Ahmad	Jabbar	Alicia	Zelaya
4	John	Smith	Ramesh	Narayan
5	Franklin	Wong	Ramesh	Narayan
6	Joyce	English	Ramesh	Narayan
7	John	Smith	Joyce	English
8	Franklin	Wong	Joyce	English
9	Jennifer	Wallace	Ahmad	Jabbar



Joined Relations Feature in SQL2 (Outer join)

- Examples:

Q8: **SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME**
 FROM EMPLOYEE E, EMPLOYEE S
 WHERE E.SUPERSSN=S.SSN

can be written as:

Q8: **SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME**
 FROM (EMPLOYEE E LEFT OUTER JOIN EMPLOYEE S
 ON E.SUPERSSN=S.SSN)

we can use the NVL function

Q8: **SELECT E.FNAME, E.LNAME, NVL(S.FNAME, ' '), NVL(S.LNAME, ' ')**
 FROM (EMPLOYEE E LEFT OUTER JOIN EMPLOYEE S
 ON E.SUPERSSN=S.SSN)

	FNAME	LNAME	FNAME_1	LNAME_1
1	Ramesh	Narayan	Franklin	Wong
2	John	Smith	Franklin	Wong
3	Joyce	English	Franklin	Wong
4	Franklin	Wong	James	Borg
5	Jennifer	Wallace	James	Borg
6	Alicia	Zelaya	Jennifer	Wallace
7	Ahmad	Jabbar	Jennifer	Wallace

Employees without supervisor
are not in the list

	FNAME	LNAME	FNAME_1	LNAME_1
1	John	Smith	Franklin	Wong
2	Ramesh	Narayan	Franklin	Wong
3	Joyce	English	Franklin	Wong
4	Alicia	Zelaya	Jennifer	Wallace
5	Ahmad	Jabbar	Jennifer	Wallace
6	Franklin	Wong	James	Borg
7	Jennifer	Wallace	James	Borg
8	James	Borg	(null)	(null)

For Employees without supervisor
the result is null

	FNAME	LNAME	NVL(S.FNAME, ' ')	NVL(S.LNAME, ' ')
1	John	Smith	Franklin	Wong
2	Ramesh	Narayan	Franklin	Wong
3	Joyce	English	Franklin	Wong
4	Alicia	Zelaya	Jennifer	Wallace
5	Ahmad	Jabbar	Jennifer	Wallace
6	Franklin	Wong	James	Borg
7	Jennifer	Wallace	James	Borg
8	James	Borg		



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



SUBSTRING COMPARISON (cont.)

- Query 17: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX'.

```
Q16: SELECT FNAME, LNAME  
        FROM EMPLOYEE  
        WHERE ADDRESS LIKE '%Houston,TX%'
```

SUBSTRING COMPARISON (cont.)

- Query 18: Retrieve all employees who were born during the 1950s. Here, '5' must be the 7th character of the string (according to our format for date dd/mm/yy), so the BDATE value is '____5_', with each underscore as a place holder for a single arbitrary character.

**Q26: SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE BDATE LIKE '____5_'**

- 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

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-', '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query 19: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

**Q27: SELECT FNAME, LNAME, 1.1*SALARY
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE SSN=ESSN AND PNO=PNUMBER AND PNAME='ProductX'**



ORDER BY

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)
- Query 28: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

**Q28: SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
ORDER BY DNAME, LNAME**

1	Administration	Jabbar	Ahmad	Computerization
2	Administration	Jabbar	Ahmad	Newbenefits
3	Administration	Wallace	Jennifer	Reorganization
4	Administration	Wallace	Jennifer	Newbenefits
5	Administration	Zelaya	Alicia	Computerization
6	Administration	Zelaya	Alicia	Newbenefits
7	Headquarters	Borg	James	Reorganization
8	Research	English	Joyce	ProductX
9	Research	English	Joyce	ProductY
10	Research	Narayan	Ramesh	ProductZ
11	Research	Smith	John	ProductY
12	Research	Smith	John	ProductX
13	Research	Wong	Franklin	ProductY
14	Research	Wong	Franklin	ProductZ
15	Research	Wong	Franklin	Computerization
16	Research	Wong	Franklin	Reorganization

ORDER BY (cont.)

- The default order is in ascending order of values
- We can specify the keyword **DESC** if we want a descending order; the keyword **ASC** can be used to explicitly specify ascending order, even though it is the default