SQL

MC536/MC526
Profs. Anderson Rocha e André Santanché

Based on original by: *Babak Bagheri Hariri*

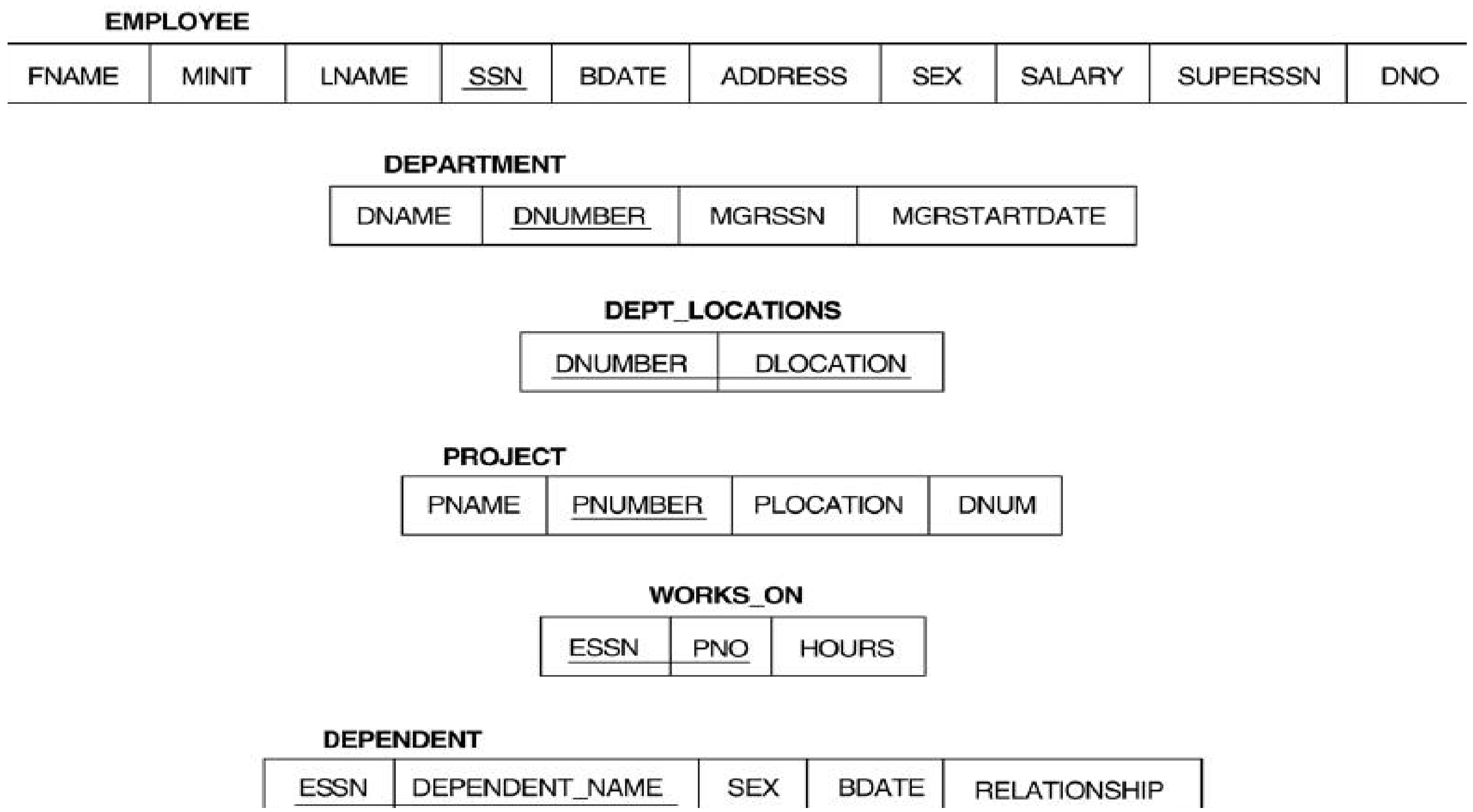
Queries in SQL

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



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

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

Queries (i)

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’
```

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
```

Queries (ii)

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'
```

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
```

Queries (iii)

Query 4: 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.

```
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')
```

Queries (iv)

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

Query 5: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q5:     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)
Q5A:    SELECT E.FNAME, E.LNAME
        FROM EMPLOYEE E, DEPENDENT D
        WHERE E.SSN=D.ESSN AND
                E.FNAME=D.DEPENDENT_NAME
```

Queries (v) – EXISTS

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not

```
Q5B:    SELECT FNAME, LNAME
        FROM EMPLOYEE
        WHERE EXISTS (SELECT *
                FROM DEPENDENT
                WHERE SSN=ESSN AND
                FNAME=DEPENDENT_NAME)
```

Queries (vi)
explicit (enumerated) set of values

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

Query 6: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q6:    SELECT DISTINCT ESSN
       FROM WORKS_ON
       WHERE PNO IN (1, 2, 3)
```

Queries (vii)

The **CONTAINS** operator compares two *sets of values* , and returns TRUE if one set contains all values in the other set (reminiscent of the *division* operation of algebra).

Query 7: Retrieve the name of each employee who works on *all* the projects controlled by department number 5.

```
Q7:     SELECT FNAME, LNAME
        FROM EMPLOYEE
        WHERE ( (SELECT PNO
                FROM WORKS_ON
                WHERE SSN=ESSN)
        CONTAINS
                (SELECT PNUMBER
                FROM PROJECT
                WHERE DNUM=5) )
```

Queries (viii) – Null Value

SQL uses **IS** or **IS NOT** to compare NULLs because it considers each NULL value distinct from other NULL

Query 8: Retrieve the names of all employees who do not have supervisors.

```
Q8:     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

Queries (ix) – JOIN

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

Can be written as:

QTA:
```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM (EMPLOYEE E LEFT OUTER JOIN EMPLOYEES
ON E.SUPERSSN=S.SSN)
```

Queries (x) – JOIN

```
Q9:     SELECT FNAME, LNAME, ADDRESS
        FROM EMPLOYEE, DEPARTMENT
        WHERE DNAME='Research' AND DNUMBER=DNO
```

Can be written as:

```
Q9A:    SELECT FNAME, LNAME, ADDRESS
        FROM (EMPLOYEE JOIN DEPARTMENT
        ON DNUMBER=DNO)
        WHERE DNAME='Research’
```

Or as:

```
Q9B:    SELECT FNAME, LNAME, ADDRESS
        FROM (EMPLOYEE NATURAL JOIN
                DEPARTMENT AS DEPT(DNAME, DNO, MSSN, MSDATE)
        WHERE DNAME='Research’
```

Joined Relations Feature in SQL2

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 B:   SELECT PNUMBER, DNUM,
        LNAME, BDATE, ADDRESS
        FROM (PROJECT JOIN
                DEPARTMENT ON

        DNUM=DNUMBER) JOIN
                        EMPLOYEE ON
                        MGRSSN=SSN) )
        WHERE PLOCATION='Stafford'
```

AGGREGATE FUNCTIONS

Include **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**

Query 10: Find the maximum salary, the minimum salary, and the average salary among all employees.

```
Q10:    SELECT MAX(SALARY),  MIN(SALARY), AVG(SALARY)
        FROM EMPLOYEE
```

Query 11: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
Q11:    SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
        FROM EMPLOYEE, DEPARTMENT
        WHERE DNO=DNUMBER AND
                  DNAME='Research'
```

Group by

SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

Query 12: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
Q12:    SELECT DNO, COUNT (*), AVG (SALARY)
        FROM EMPLOYEE
        GROUP BY DNO
```

Query 13: For each project, retrieve the project number, project name, and the number of employees who work on that project.

```
Q13:    SELECT PNUMBER, PNAME, COUNT (*)
        FROM PROJECT, WORKS_ON
        WHERE PNUMBER=PNO
        GROUP BY PNUMBER, PNAME
```

Group by cont. Having

The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

Query 14: For each project *on which more than two employees work*, retrieve the project number, project name, and the number of employees who work on that project.

```
Q14:    SELECT PNUMBER, PNAME, COUNT (*)
        FROM PROJECT, WORKS_ON
        WHERE PNUMBER=PNO
        GROUP BY PNUMBER, PNAME
        HAVING COUNT (*) > 2
```

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only

the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

SELECT <attribute list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]

Summary of SQL Queries (cont.)

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes

Summary of SQL Queries (cont.)

- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
- A query is evaluated by first applying the WHERE-clause, then
- GROUP BY and HAVING, and finally the SELECT-clause

More complex Select “SQL Server”

SELECT *select_list*
[INTO *new_table*]
FROM *table_source*
[WHERE *search_condition*]
[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

Select Clause:

SELECT [ALL | DISTINCT]
[TOP *n* [PERCENT] [WITH TIES]]
< select_list >
< select_list > ::=
{ *
| { *table_name* | *view_name* | *table_alias* }**.***
| { *column_name* | *expression* | IDENTITYCOL | ROWGUIDCOL }
[[AS] *column_alias*]
| *column_alias* **=** *expression*
} [**,**...*n*]

From Clause:

[FROM { < table_source > } [**,**...*n*]]
< table_source > ::=
table_name [[AS] *table_alias*] [WITH **(** < table_hint > [**,**...*n*] **)**]
| *view_name* [[AS] *table_alias*]
| *rowset_function* [[AS] *table_alias*]
| OPENXML
| *derived_table* [AS] *table_alias* [**(** *column_alias* [**,**...*n*] **)**]
| < joined_table >
< joined_table > ::=
< table_source > < join_type > < table_source > ON < search_condition >
| < table_source > CROSS JOIN < table_source >
| < joined_table >
< join_type > ::=
[INNER | { { LEFT | RIGHT | FULL } [OUTER] }]
[< join_hint >]
JOIN
Arguments
< table_source >

More complex Select “SQL Server” Cont.

Where Clause:

[WHERE < search_condition > | < old_outer_join >]

< old_outer_join > ::=
column_name { * = | = * } *column_name*

Group by clause:

[GROUP BY [ALL] *group_by_expression* [,...*n*]
[WITH { CUBE | ROLLUP }]
]

Having:

[HAVING < search_condition >]

Order By Clause:

[ORDER BY { *order_by_expression* [ASC | DESC] } [,...*n*]]

Compute Clause:

[COMPUTE
{ { AVG | COUNT | MAX | MIN | STDEV | STDEVP | VAR | VARP | SUM }
(*expression* **)** } [**,**...*n*]
[BY *expression* [**,**...*n*]]
]

Compute

Row aggregate function	Result
AVG	Average of the values in the numeric expression
COUNT	Number of selected rows
MAX	Highest value in the expression
MIN	Lowest value in the expression
STDEV	Statistical standard deviation for all values in the expression
STDEVP	|Statistical standard deviation for the population for all values in the expression
SUM	Total of the values in the numeric expression
VAR	Statistical variance for all values in the expression
VARP	Statistical variance for the population for all values in the expression