

## More on SQL and Transactions

ECE 650  
Systems Programming & Engineering  
Duke University, Spring 2016

## Example Relational Database Tables

- COMPANY = {EMPLOYEE, DEPARTMENT, DEPT\_LOCATIONS, PROJECT, WORKS\_ON, DEPENDENT}

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	DEP_NAME	SEX	BDATE	RELATIONSHIP
------	----------	-----	-------	--------------

ECE 650 – Spring 2016

2

## LIKE clause

- Allows comparison conditions on parts of a string
  - Two special characters:
    - '%' replaces an arbitrary number of characters
    - '\_' replaces a single character
- SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE ADDRESS LIKE '%Houston,TX%';
  - Retrieve all employees whose address is in Houston, Texas
- SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE BDATE LIKE '\_\_ \_5 \_ \_ \_ \_ \_';
  - Retrieve all employees who were born during the 1950s
  - Where BDATE format is 'YYYY-MM-DD'

ECE 650 – Spring 2016

3

## Arithmetic Operators

- We can use arithmetic on numeric domains
  - add, subtract, multiply, divide
- SELECT FNAME, LNAME, 1.1\*SALARY  
FROM EMPLOYEE, WORKS\_ON, PROJECT  
WHERE SSN=ESSN AND PNO=PNUMBER AND  
PNAME='ProductX';
  - Want to see effect of giving all employees who work on ProductX a 10% raise

ECE 650 – Spring 2016

4

## Other Operators

- Can append strings with concatenate operator: '||'
- Increment and decrement operators for
  - Date, time, timestamp, interval data types
- BETWEEN operator (for convenience):
- SELECT \*  
FROM EMPLOYEE  
WHERE (SALARY BETWEEN 30000 AND 40000) AND DNO=5;
  - Retrieve all employees in dept 5 whose salary is between \$30,000 and \$40,000

ECE 650 – Spring 2016

5

## ORDER BY Clause

- Sometimes desirable to re-order returned results
- Can use ORDER BY clause
- SELECT DNAME, LNAME, FNAME, PNAME  
FROM DEPARTMENT, EMPLOYEE, WORKS\_ON, PROJECT  
WHERE DNUMBER=DNO AND SSN=ESSN AND  
PNO=PNUMBER  
ORDER BY DNAME, LNAME, FNAME
  - Retrieve a list of employees and projects they are working on
  - Ordered by department, and within each department, ordered alphabetically by last name, first name

ECE 650 – Spring 2016

6

## ORDER BY Clause (2)

- Can also specify ascending or descending order
  - ASC or DESC keyword
- Example:
  - **ORDER BY DNAME DESC, LNAME ASC, FNAME ASC**

## Nested Queries

- Some queries require fetching existing DB values and using them in a comparison condition
- Useful to use nested queries
  - SELECT, FROM, WHERE blocks inside WHERE of other query
  - Other query is called **outer query**

## Nested Query Example

- **SELECT DISTINCT PNUMBER**  
FROM PROJECT  
WHERE PNUMBER **IN** (SELECT PNUMBER  
FROM PROJECT, DEPARTMENT,  
EMPLOYEE  
WHERE DNUM=DNUMBER AND  
MGRSSN=SSN AND LNAME='Smith')
  - OR
  - **PNUMBER IN** (SELECT PNO FROM WORKS\_ON  
WHERE ESSN=SSN AND LNAME='Smith');
- Select project numbers of projects with 'Smith' involved as a manager
- Select project numbers of projects with 'Smith' involved as a worker
- IN** compares a value v with a set, evaluates to true if v is an element in the set

## More on IN Operator

- Can compare tuple of values in parenthesis with a set of union-compatible tuples
- **SELECT DISTINCT ESSN**  
FROM WORKS\_ON  
WHERE (PNO, HOURS) **IN** (SELECT PNO, HOURS  
FROM WORKS\_ON  
WHERE ESSN='123456789');
- Select SSN of employees working the same (project, hours) combination on some project that employee with SSN 123456789 works on

## ANY, SOME, ALL Keywords

- **ANY** and **SOME** operators have same meaning
  - Can use equivalently to IN
  - E.g. WHERE PNUMBER = ANY ...
    - instead of WHERE PNUMBER IN ...
  - Can also combine with operators for comparison (>, >=, <, <=)
- **ALL**
  - Compares a value 'v' to every value in a set
  - **SELECT LNAME, FNAME**  
FROM EMPLOYEE  
WHERE SALARY > ALL (SELECT SALARY FROM EMPLOYEE  
WHERE DNO=5);
  - Returns names of employees whose salary is greater than salary of all employees in department 5

## Correlated Nested Queries

- **Correlated condition:**
  - When condition in WHERE-clause of a nested query refers to some attribute of a relation declared in the outer query
- Consider that the nested query is evaluated once for each tuple in the outer query
- For example –
- **SELECT E.FNAME, E.LNAME**  
FROM EMPLOYEE AS E  
WHERE E.SSN **IN** (SELECT ESSN FROM DEPENDENT  
WHERE E.FNAME=DEPENDENT\_NAME  
AND E.SEX=SEX);

## Correlated Nested Queries (2)

- In general:
  - For query written with nested SELECT, FROM, WHERE blocks
  - And using the = or IN operators
  - Can always be expressed as a single query block
- For example, can rewrite query from previous slide as:
- ```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E, DEPENDENT AS D
WHERE E.SSN=D.ESSN AND E.SEX=D.SEX AND
E.FNAME=D.DEPENDENT_NAME;
```

## EXISTS Function

- Check whether result of correlated nested query is empty
  - Empty means contains no tuples
- ```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE EXISTS IN (SELECT * FROM DEPENDENT
WHERE E.SSN=ESSN AND E.SEX=SEX AND
E.FNAME= DEPENDENT_NAME);
```
- Can also use "NOT EXISTS"
- ```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS (SELECT * FROM DEPENDENT
WHERE SSN=ESSN);
```

Find names of employees who have no dependents

## UNIQUE Function

- **UNIQUE(Q)**
  - Returns true if there are no duplicate tuples in the query Q
  - Otherwise returns false

## Explicit Sets and NULLs

- WHERE-clause may contain explicit set of values
  - Enclosed in parenthesis
- Example:
  - ```
SELECT DISTINCT ESSN
FROM WORKS_ON
WHERE PNO IN(1, 2, 3);
```

All employee SSNs who work on projects 1, 2 or 3
- SQL allows queries to check whether a value is NULL
  - NULL means missing or undefined or not applicable
  - Must use "IS" or "IS NOT" instead of = or ≠
  - ```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE SUPERSSN IS NULL;
```

All employees who do not have supervisors

## Joined Table

- Specify a table resulting from a join operation
  - In the FROM-clause of a query
  - May be easier to follow than mixing together all the select and join conditions in the WHERE-clause
- Example:
  - Retrieve name and address of every employee who works for the 'Research' department
  - ```
SELECT FNAME, LNAME, ADDRESS
FROM (EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER)
```
- Can also use 'NATURAL JOIN':
  - No join condition is specified (e.g. 'ON' clause)

## Aggregate Functions

- Built-in functions:
  - COUNT, SUM, MIN, MAX, AVG
  - COUNT: # of tuples or values specified in a query
- Find sum of salaries of all employees of the 'Research' department, as well as max, min, & average salaries
  - ```
SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'
```
- Retrieve the number of employees in the company
  - ```
SELECT COUNT(*) FROM EMPLOYEE;
```
- Count the # of distinct salary values in the database
  - ```
SELECT COUNT(DISTINCT SALARY) FROM EMPLOYEE;
```

## GROUP BY Clause

- Sometimes want to apply aggregate functions to subgroups of tuples in a relation
  - E.g. find average salary of employees in each department
  - GROUP BY clause specifies the grouping attributes which should also appear in the SELECT-clause
- Example: for each department, retrieve the department number, number of employees in dept., and avg salary
  - ```
SELECT DNO, COUNT(*), AVG(SALARY)
FROM EMPLOYEE
GROUP BY DNO;
```
- Example: retrieve the project number, project name, and the # of employees who work on that project
  - ```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME;
```

ECE 650 – Spring 2016

19

## SQL Views

- Views (also called Virtual Tables)
  - Single table derived from other tables
  - Does not necessarily exist in physical form (e.g. stored in dbase)
  - Can think of as way to specify a table we need to reference often
    - E.g. instead of JOIN on several tables every time for certain query
- Example:
  - ```
CREATE VIEW WORKS_ON1
AS SELECT FNAME, LNAME, PNAME, HOURS
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND PNO=PNUMBER;
```
  - Creates view with first name, last name, project name, and hours for each employee's project

ECE 650 – Spring 2016

20

## Transaction Processing

ECE 650 – Spring 2016

21

## Transaction Processing

- More next class...

ECE 650 – Spring 2016

22