Introduction to SQL

Lecture 4

Dr. Reda M. Hussien

Self Join Example

- Relation emp-super

person	supervisor
Bob	Alice
Mary	Susan
Alice	David
David	Mary

  - Find the supervisor of “Bob”
  - Find the supervisor of the supervisor of “Bob”
  - Find ALL the supervisors (direct and indirect) of “Bob

String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator like uses patterns that are described using two special characters:
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any character.
- Find the names of all instructors whose name includes the substring “dar”.

```
select name  from instructor where name like '%dar%'
```

- Match the string “100%

```
like '100\%'  escape  '\'
```

- in that above we use backslash (\) as the escape character.

String Operations (Cont.)

- Patterns are case sensitive.
- Pattern matching examples:
  - ‘Intro%’ matches any string beginning with “Intro”.
  - ‘%Comp%’ matches any string containing “Comp” as a substring.
  - ‘_ _ _’ matches any string of exactly three characters.
  - ‘_ _ _ %’ matches any string of at least three characters.
- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

```
select distinct name  from instructor order by name
```

- We may specify desc for descending order or asc for ascending order, for each attribute; ascending order is the default.
  - Example: order by name desc
- Can sort on multiple attributes
  - Example: order by dept_name, name

Where Clause Predicates

- SQL includes a between comparison operator
- Example: Find the names of all instructors with salary between \$90,000 and \$100,000 (that is, ≥ \$90,000 and ≤ \$100,000)

```
select name from instructor where salary between 90000 and 100000
```

- Tuple comparison

```
select name, course_id from instructor, teaches
where (instructor.ID, dept_name) = (teaches.ID, 'Biology');
```

Duplicates

- In relations with duplicates, SQL can define how many copies of tuples appear in the result.
- Multiset versions of some of the relational algebra operators – given multiset relations r1 and r2:
  - 1. $\sigma\theta$ (r1): If there are c1 copies of tuple t1 in r1, and t1 satisfies selections $\sigma\theta$,, then there are c1 copies of t1 in $\sigma\theta$ (r1).
  - 2. ΠA (r): For each copy of tuple t1 in r1, there is a copy of tuple ΠA (t1) in ΠA (r1) where ΠA (t1) denotes the projection of the single tuple t1.
  - 3. r1 x r2: If there are c1 copies of tuple t1 in r1 and c2 copies of tuple t2 in r2, there are c1 x c2 copies of the tuple t1. t2 in r1 x r2

Duplicates (Cont.)

- Example: Suppose multiset relations r1 (A, B) and r2 (C) are as follows:

  r1 = {(1, a) (2,a)} r2 = {(2), (3), (3)}

- Then ΠB(r1) would be {(a), (a)}, while ΠB(r1) x r2 would be

  {(a,2), (a,2), (a,3), (a,3), (a,3), (a,3)}

- SQL duplicate semantics:

  select A1,, A2, ..., An

  from r1, r2, ..., rm
  where P

- is equivalent to the multiset version of the expression: $\prod_{A_1, A_2, \ldots, A_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$

Set Operations

- Find courses that ran in Fall 2009 or in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
union
(select course_id from section where sem = 'Spring' and year = 2010)
```

- Find courses that ran in Fall 2009 and in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
intersect
(select course_id from section where sem = 'Spring' and year = 2010)
```

- Find courses that ran in Fall 2009 but not in Spring 2010

```
(select course_id from section where sem = 'Fall' and year = 2009)
except
(select course_id from section where sem = 'Spring' and year = 2010)
```

Set Operations (Cont.)

- Find the salaries of all instructors that are less than the largest salary.

```
select distinct T.salary from instructor as T, instructor
as S where T.salary < S.salary
```

- Find all the salaries of all instructors

```
select distinct salary from instructor
```

- Find the largest salary of all instructors.
    - (select "second query") except (select "first query")

Set Operations (Cont.)

- Set operations union, intersect, and except
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates use the corresponding multiset versions union all, intersect all and except all.
- Suppose a tuple occurs m times in r and n times in s, then, it occurs:
  - m + n times in r union all s
  - min(m,n) times in r intersect all s
  - max(0, m – n) times in r except all s

Null Values

- It is possible for tuples to have a null value, denoted by null, for some of their attributes
- null signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving null is null
  - Example: 5 + null returns null
- The predicate is null can be used to check for null values.
  - Example: Find all instructors whose salary is null.

```
select name from instructor where salary is null
```

Null Values and Three Valued Logic

- Three values – true, false, unknown
- Any comparison with null returns unknown
  - Example: 5 < null   or   null <> null   or   null = null
- Three-valued logic using the value unknown:
  - OR: (unknown or true)   = true,
    (unknown or false) = unknown
    (unknown or unknown) = unknown
  - AND: (true and unknown) = unknown,
    (false and unknown) = false,
    (unknown and unknown) = unknown
  - NOT: (not unknown) = unknown
  - "P is unknown" evaluates to true if predicate P evaluates to unknown
- Result of where clause predicate is treated as false if it evaluates to unknown

End of Lecture 4