Module - II

SQL - Structured Query Language

Lecture - 5

Set Operations | Null Values | Aggregate Functions

SQL - RETRIEVAL

- Set Operations
- Null Values
- Aggregate Functions

Set Operations

- **UNION:** Return all Distinct rows selected by either query.
- **UNION ALL:** Return all rows selected by either query, including duplicates.
- **INTERSECT:** Returns all Distinct rows selected by both queries.
- **MINUS:** Return all Distinct rows selected by first query but not in the second.

Set Operations

- Find courses that ran in Fall 2017 or in Spring 2018

```
(select course_id  from section where sem =
'Fall' and year = 2017)
union
(select course_id  from section where sem =
'Spring' and year = 2018)
```

Set Operations

- Find courses that ran in Fall 2017 and in Spring 2018

```
(select course_id  from section where sem =
'Fall' and year = 2017)
intersect
(select course_id  from section where sem =
'Spring' and year = 2018)
```

Set Operations

- Find courses that ran in Fall 2017 but not in Spring 2018

```
(select course_id  from section where sem =
'Fall' and year = 2017)
except
(select course_id  from section where sem =
'Spring' and year = 2018)
```

Set Operations

- Set operations **union**, **intersect**, and **except**
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates use the
  - **union all**,
  - **intersect all**
  - **except all**

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
```

- The predicate **`is not null`** succeeds if the value on which it is applied is not null.

Null Values

- SQL treats as **`unknown`** the result of any comparison involving a null value (other than predicates **`is null`** and **`is not null`**).
- Example: `5 < null` or `null <> null` or `null = null`

Null Values

- The predicate in a **where** clause can involve Boolean operations (**and**, **or**, **not**); thus the definitions of the Boolean operations need to be extended to deal with the value **unknown**.

```
and: (true and unknown)    = unknown,
     (false and unknown)   = false,
     (unknown and unknown) = unknown

or:  (unknown or true)     = true,
     (unknown or false)    = unknown
     (unknown or unknown)  = unknown
```

- Result of **where** clause predicate is treated as *false* if it evaluates to *unknown*

Aggregate Functions

- These functions operate on the multiset of values of a column of a relation, and return a value

  `avg:` average value
  `min:` minimum value
  `max:` maximum value
  `sum:` sum of values
  `count:` number of values

Aggregate Functions - Examples

- Find the average salary of instructors in the Computer Science department

```
select avg (salary)
from instructor
where dept_name= 'Comp. Sci.';
```

Aggregate Functions - Examples

- Find the total number of instructors who teach a course in the Spring 2018 semester

```
select count (distinct ID)
from teaches
where semester = 'Spring' and year = 2018;
```

Aggregate Functions - Examples

- Find the number of tuples in the *course* relation

```
select count (*) from course;
```

Aggregate Functions – Group By

- Find the average salary of instructors in each department

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Functions – Group By

- Attributes in select clause outside of aggregate functions must appear in group by list

```
/* erroneous query */
select dept_name, ID, avg (salary)
from instructor
group by dept_name;
```

Aggregate Functions – Having Clause

- Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name
having avg (salary) > 42000;
```

Note: predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups