DBMS Week 2 TA Session

Relational Operators

- σ - Select
- π - Project
- $\neg$ - Negation (not)
- $\wedge$ - AND
- $\vee$ - OR
- $\cup$ - Union
- $\cap$ - Intersection
- $\times$ - Cartestion Product
- $-$ - Set Difference
- $\bowtie$ - Natural Join

Domain Types

- `char(n)` - fixed n length of characters
- `varchar(n)` - characters varies from 0 to n(inclusive)
- `numeric(p,d)` - d digits to the right of decimal point and p total no of digits
    - **Example** - `numeric(4, 2)` - 44.22
- `int` - Interger value

DDL (Data Definition Language)

- A language which is used design the schema of the database and also able modifies it.
- **Example**
  - `CREATE`
  - `DROP`
  - `ALTER`

Create a Table

```
CREATE TABLE takes (
    ID varchar(5),
    course_id varchar(8),
    sec_id varchar(8),
    semester varchar(8),
    year_ numeric(4, 0),
    grade varchar(2),
    primary key (ID, course_id, sec_id, semester, year_),
    foreign key (ID) references student,
    foreign key (course_id, sec_id, semester, year_)references section
)
```

DROP

```
DROP TABLE takes
```

DML (Data Manipulation Language)

- The SQL commands that deal with the manipulation of data present in the database
- **Example**
  - `INSERT`
  - `UPDATE`
  - `DELETE`

INSERT

```
INSERT into takes values (1, 'C001', 'CS', 'spring', '2022', 'S')
```

```
INSERT into takes (ID, course_id, sec_id, semester, year_, grade)
values ('1', 'C001', 'CS', 'spring', '2022', 'S')
```

DELETE

```
DELETE from takes WHERE ID=1
```

Basic SQL Queries

```
SELECT <attributes>
FROM <tables>
WHERE <condtion>
```

- Example

```
SELECT dept_name
FROM instructor
WHERE dept_name='Biology'
```

SELECT Clause

- `DISTINCT` - Selects all the distinct values
- `*` - Selects all the attributes
- `as` - Renames the attribute
- `TOP <n>` - selects top n tuples

- **Example**

```
SELECT DISTINCT(name) as Instructor_Name
FROM instructor
WHERE dept_name='Biology'
```

FROM Clause

```
SELECT *
FROM instructor, course
```

- The above query basically performs a **cross-join** between the instructor and course table.

WHERE Clause

```
SELECT *
FROM instructor as i, course as c
WHERE i.course_id=c.course_id and i.dept_name='Biology' and salary>40000
```

String Operations

- `LIKE` - Uses the pattern that are described in like condition and matches with the attribute
- `%` - matches any substring
- `_` - matches any character
- **Example**

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

- The above SQL query matches with the instructor names who has a substring `dar` in their names.

LIKE (Continued)

- `'Intro%'` matches any string beginning with "Intro"
- `'%Comp%'` matches any string containing "Comp" as a substring
- `'%Science'` - matches any string ending with "Science"
- `'_ _ _'` matches any string of exactly three characters
- `'_ _ _%'` matches any string of at least three characters

WHERE Clause Predicates

`BETWEEN a and b`

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

- Select the name of the instructor whose salaries between 90000 and 100000 (both are inclusive)

`IN`

- Acts like shorthand operator for `OR`

```
select name
from instructor
where dept_name in ('Comp Sci', 'Biology')
```

Set Operations

- `UNION` and `UNION ALL`
- `INTERSECT` and `INTERSECT ALL`
- `EXCEPT` and `EXCEPT ALL`

Note

- `UNION ALL`, `INTERSECT ALL` and `EXCEPT ALL` retains the duplicate

Aggregate functions

- **avg** - Average value
- **min** - Minimum value
- **max** - Maximum value
- **sum** - Sum of all value
- **count** - Count of all value

Examples of Aggregate functions

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

```
select count(distinct ID)
from teaches
where semester = 'Spring' and year = 2010;
```

Group By

```
SELECT dept_name, avg(salary)
from instructor
group by dept_name
```

Having

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

Note - the Order of SQL queries

SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY