**24/03/2015**

# SQL IN ACCESS

**QUERY1:***Students with the status 'MS' or 'Ph' in city 'ankara'*

```
SELECT city, sname, sid
FROM students
WHERE (city='ankara') And (status In ('MS','PhD'));
```

**Aggregate Operators**

SUM,AVG,MIN,MAX,COUNT

**QUERY2:** *Gives the sum of department capacitywhich has the department name "EngineeRing"(Case insensitive so that it also searches in department name Engineering),(*something* the stars are the wildcard characters, anything can comes before and after it)*

```
SELECT SUM(d_capacity) AS EngineeringCapacity
FROM department
WHERE d_name LIKE '*EngineeRing*';
```

**QUERY3:***Students takes the course CS281 and their students id's are equal to enrollment id's.(AS is optional)*

```
SELECT *
FROM students AS s, enrollment AS e
WHERE s.sid=e.sid And e.cid="CS281";
```

**QUERY4:***Students average CGPAwho have a faculty member,Zeynep Nadir, as advisor*

```
SELECT AVG(cgpa) AS avgCGPA
FROM students AS s, facultyMember AS f
WHERE s.advisor_id=f.fid And f.fname='Zeynep Nadir';
```

**QUERY5:***Shows the course id's, how many students have taken it and the average CGPA of that courses.*

```
SELECT cid, COUNT(*) AS totalStudents, AVG(cgpa) AS avgCGPA
FROM students AS s, enrollment AS e
WHERE s.sid=e.sid
GROUP BY cid
HAVING count(*)>0;
```

**QUERY 6:** *Finding the number of students who are undergraduate, graduate and phD.*

```
SELECT s.status, COUNT(*) AS totalStudents
```

```
FROM students AS s

GROUP BY s.status;
```

**QUERY7:** *Showing the students who are below average.*

```
SELECT *
FROM students AS s
WHERE s.CGPA < (SELECT AVG(CGPA) FROM students);
```

**QUERY8:** *Showing the students' ids who donot get F,FX or FZ from ECON 101 course and get A or A- from CS 281.*

```
SELECT s.sid
FROM students AS s, enrollment AS e
WHERE s.sid = e.sid AND e.cid = "CS281" AND e.grade IN ("A", "A-") AND s.sid NOT IN
(
SELECT s.sid FROM students s, enrollment e WHERE s.sid = e.sid AND e.cid =
"ECON101" AND e.grade IN ("F", "FX", "FZ")
);
```

(This is a nested query)

**QUERY9:** *Showing the departments who do not have any faculty member.*

```
SELECT *
FROM department AS d
WHERE NOT EXISTS
(
SELECT *
FROM facultyMember f
WHERE d.dept_id = f.dept_id
);
```

(This is corrolated query)

**QUERY10:** *Showing the students who are enrolled in CS 281 course in s13 or s12.*

```
SELECT s.sid
FROM students s, enrollment e
WHERE s.sid = e.sid AND e.cid = "CS281" AND e.semester = "S13"
UNION SELECT s.sid
FROM students s, enrollment e
WHERE s.sid = e.sid AND e.cid = "CS281" AND e.semester = "S12";
```

**QUERY11:** *Showing students who have taken cs 281 only once.*

```
SELECT s.sid
FROM students AS s, enrollment AS e1
WHERE s.sid = e1.sid AND e1.cid = "CS281" AND s.sid NOT IN
(
SELECT e2.sid
FROM enrollment e2
WHERE e2.cid = "CS281"
GROUP BY e2.sid
HAVING COUNT (*) > 1
);
```

**CHAPTER 19**

**Schema Refinement**

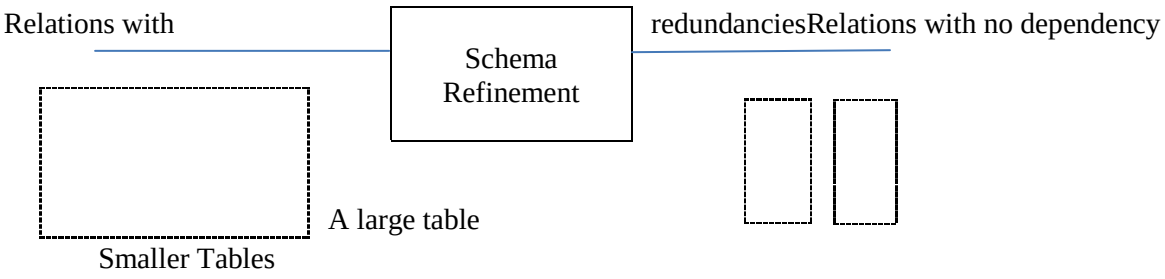
-Normalization

- Functional Dependency (FD)

-Normal Forms: First Normal Form (1NF), 2NF, 2NF

Boyce Code Normal Form: BCNF

-Armstrong Axioms




**<u>Problems due to Redundancy:</u>**

1. <u>Update Anomaly:</u> unnecessary update of several tuples
2. <u>Deletion Anomaly:</u> delete something, due this deletion we delete another thing that we need
3. <u>Insertion Anomaly:</u> In order to insert a necessary information, we need to enter unnecessary information

Ex: employee(<u>eno</u>, ename, city, status)

| | | | |
|---|---|---|---|
| E1 | Ali | 34 | 1 |
| E2 | Ayşe | 34 | 1 |

E3 Veli 35 1
E4 Zeynep 35 1
E5 Oya 06 2

Update Anomaly: Change status of İstanbul from 1 to 5

-need to update several tuples

Deletion Anomaly: Delete Oya then we will lose the information that is status for Ankara

Insertion Anomaly:In order to include status of Eskişehir(26), we need to have an employee in Eskişehir. So, we may use a fake employee name.

**SOLUTION:**

*Relation Decomposition*

Make sure that we have a losless decomposition

FD: they are integrity constraints

R1(eno, ename, city) R2(city, status)

E1 Ali 34 34 1
E2 Ayşe 34 35 1
E3 Veli 35 06 2
E4 Zeynep 35
E5 Oya 06

Employee= R1 ⋈ R2
We obtain the original employee relation, therefore, we have a losless decomposition

*Lossy Decomposition*

Ra (eno, ename, status) Rb(city, status)

E1 Ali 34 1 34 1
E2 Ayşe 34 1 35 1
E3 Veli 35 1 06 2
E4 Zeynep 35 1
E5 Oya 06 2

Ra ⋈ Rb = Rab (eno, ename, status, city)

E1 Ali 1 34
E2 Ali 1 35
E2 Ayşe 1 34
. . . .
. . . .