SQL – Part 4

Advanced SQL Queries – Set Operations

- SQL incorporates several set operations: UNION (set union) and INTERSECT (set intersection), and sometimes EXCEPT (set difference / minus).
- S
- S
  a
  or

```
(SELECT * FROM STUDENT WHERE Em
 UNION
(SELECT * FROM STUDENT WHERE Em                         ');
```

- For example, the following query will not work

```
(SELECT StudentID, Name FROM STUDENT)
 UNION
(SELECT Email FROM STUDENT);
```

Advanced SQL Queries – Join Operations

- When we want to retrieve data from *more than one relations*, we often need to use **join** operations.
- C
  tw

  -
  - List all students, and their enrolled courses i

STUDENT			
StudentID	Name	DoB	Email

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

Advanced SQL Queries – Inner Join

- **Inner Join**: tuples are included in the result only if there is at least one matching in both relations.
- For the query "list the names of all courses which have been enrolled by at le

o

No	Cname	Unit
COMP2400	Relational D	
COMP3900	Advanced Datab	

ENROL				
StudentID	CourseNo	Semes		
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

- Result:

Cname
Relational Databases

Advanced SQL Queries – Outer Join

- **Outer Join** includes **Left Join** and **Right Join**.
- **Left/Right Join**: all tuples of the left/right table are included in the result, e

Advanced SQL Queries – Outer Join

- **Left Join**: A left join retains all rows of the left table regardless of whether there is a row that matches on the right table.

ENROL1		
StudentID	CourseNo	Semester

456	Tom	25/01/19	
458	Peter	20/02/19	

```
SELECT *
  FROM Student s LEFT JOIN E
                              ON s.StudentID=e.StudentID;
```

StudentID	Name	DoB	Email	StudentID	CourseNo	Semester
456	Tom	25/01/1988	tom@gmail.com	456	COMP1130	2016 S1
456	Tom	25/01/1988	tom@gmail.com	456	COMP2400	2016 S2
458	Peter	20/02/1991	peter@hotmail.com	null	null	null

Advanced SQL Queries – Outer Join

- **Right Join**: A right join retains all rows of the right table regardless of whether there is a row that matches on the left table.

ENROL1		
StudentID	CourseNo	Semester
456	COMP1130	2016 S1

456	Tom	25/01/1988	
458	Peter	20/02/1991	

```
SELECT *
 FROM STUDENT s RIGHT JOIN ENROL
                                    ON s.StudentID=e.StudentID;
```

StudentID	Name	DoB	Email	StudentID	CourseNo	Semester
456	Tom	25/01/1988	tom@gmail.com	456	COMP1130	2016 S1
null	null	null	null	457	COMP1130	2016 S1
456	Tom	25/01/1988	tom@gmail.com	456	COMP2400	2016 S2

Advanced SQL Queries – Outer Join

- For the query “list all students, and their enrolled courses if any”, we can use either of the following statements:

```
                                                      ;
      FROM Enrol1 e RIGHT JOIN S
                         ON e.S
```

- If we have 1000 tuples in Student, then th least 1000 tuples (one tuple in Student may occur multiple times) with the following attributes:

StudentID	Name	DoB	Email	CourseNo	Semester
...	...	...	...	...	...

Advanced SQL Queries – Natural Join

- **Motivation**: An inner join retains all the data of the two tables for , with duplication
- SELECT *

;

457	COMP1130	2016 S1
456	COMP24	

STUDENT			
StudentID	Name	DoB	
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

- Result:

StudentID	Name	DoB	Email	StudentID	CourseNo	Semester
456	Tom	25/01/1988	tom@gmail.com	456	COMP1130	2016 S1
456	Tom	25/01/1988	tom@gmail.com	456	COMP2400	2016 S2

Advanced SQL Queries – Natural Join

- **Natural Join**: A natural join retains all the data of the two tables for only the matched rows, without duplication.

-
  ```
  SELECT *
  ```

[illegible]		
[illegible]	[illegible]	[illegible]
457	COMP1130	2016 S1
456	COMP24	

STUDENT			
StudentID	Name	DoB	[illegible]
456	Tom	25/01/19	
458	Peter	20/02/1991	peter@hotmail.com

- Result:

StudentID	Name	DoB	Email	CourseNo	Semester
456	Tom	25/01/1988	tom@gmail.com	COMP1130	2016 S1
456	Tom	25/01/1988	tom@gmail.com	COMP2400	2016 S2

Advanced SQL Queries – Natural Join

- **Natural Join**: One kind of inner join, in which two relations are joined implicitly by comparing all attributes of the same names in both relations.
- For the query “list all students who have enrolled and their courses”, use:

[illegible]				
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
457	COMP1130	2016 S1	active	25/02/2016

STUDENT			
StudentID	Name	DoB	[illegible]
456	Tom	25/01/19	[illegible]
458	Peter	20/02/19	

- Result:(STUDENT.StudentID=ENROL.StudentID is used in the query)

StudentID	Name	DoB	Email	CourseNo	Semester	Status	EnrolDate
456	Tom	25/01/1988	tom@gmail.com	COMP1130	2016 S1	active	25/02/2016

Advanced SQL Queries – Subqueries

- **Subqueries** are just queries that are used where a relation is required.
- Subqueries can be specified within the `FROM`-clause (usually in conjunction wi ry)
- S

  - `IN` *subquery* tests if tuple occurs in the result of the subquery
  - `EXISTS` *subquery* tests whether the sub relation

  - using `ALL`, `SOME` or `ANY` before a subqu comparison formulae
  - in all these cases the condition involving the subquery can be negated using a preceding `NOT`

Subqueries – In

- Recall that, for the query "list all students who have enrolled and their courses", we have:

```
      SELECT *
```

- N [illegible] *that* *h* [illegible]", we have

```
      SELECT s.*, e1.CourseNo
        FROM STUDENT s NATURAL JOIN
       WHERE e1.CourseNo IN
                     (SELECT e2.CourseNo
                        FROM ENROL e2
                    GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

Subqueries – Exists

- For the query: “list all students who have enrolled in at least one course”, we have

```
SELECT s.*
```

- For the query: “list all students who have have

```
SELECT s.*
  FROM STUDENT s
 WHERE NOT EXISTS (SELECT *
                     FROM ENROL e
                    WHERE s.StudentID=e.StudentID);
```

Subqueries – More Complicated

- For the query: “list the courses that have the largest number of students enrolled in Semester 2 2016”, we have

```
SE

          GROUP BY e1.CourseNo) e
 WHERE e.NoOfStudents =
          (SELECT MAX(e2.NoOfStude
             FROM (SELECT e1.CourseN
                     FROM Enrol e1
                    WHERE e1.Semester = '2016 S2'
                    GROUP BY e1.CourseNo) e2);
```

Subqueries – More Complicated

- For the query: “list all the courses that have more students enrolled than at least one other course in Semester 2 2016”, we have

```
SE
```

```
         GROUP BY e1.CourseNo) e
 WHERE e.NoOfStudents > ANY
            (SELECT e2.NoOfStudents
               FROM (SELECT e1.CourseN
                       FROM ENROL e1
                      WHERE e1.Semester = '2016 S2'
                      GROUP BY e1.CourseNo) e2);
```

Views in SQL

- A view in SQL is a virtual table that is derived from other tables in the same database or previously defined views.
- H
  - ,

```
CREATE VIEW ENROL1
AS SELECT s.StudentID, s.Name, e.CourseNo, e
FROM STUDENT s, ENROL e
WHERE s.StudentID=e.StudentID;
```