Medical Informatics

Lecture 7: More SQL

Dr Areti Manataki

Nanjing Medical University

In the previous lecture

- We learnt how to use the SQL Data Manipulation Language to
  - insert, delete and update rows in a table

```
INSERT
    INTO Student (mn, name, email, age)
    VALUES (‘s1253477’, ‘Jenny’,
            ‘jenny@sms.ed.ac.uk’, 23)
```

In the previous lecture

- We learnt how to use the SQL Data Manipulation Language to
  - insert, delete and update rows in a table
  - query the database

```
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Medical Informatics’
```

In this lecture

- We'll learn how to formulate more expressive SQL queries with the use of:
  - SQL set operators
  - nested queries
  - aggregate operators

Set operations in SQL

- SQL provides three set-operation constructs that extend the basic form of a query:
  - UNION: A or B
  - INTERSECT: A and B
  - EXCEPT: A but not B

UNION in SQL

- Find the email addresses of all students taking Medical Informatics or Advanced Databases.

```
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Medical Informatics’
UNION
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Advanced Databases’
```

UNION in SQL

- Find the email addresses of all students taking Medical Informatics or Advanced Databases.

```
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND (C.title = ‘Medical Informatics’ OR
C.title = ‘Advanced Databases’)
```

INTERSECT in SQL

- Find the email addresses of all students taking Medical Informatics and Advanced Databases.

```
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Medical Informatics’
INTERSECT
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Advanced Databases’
```

INTERSECT in SQL

- Find the email addresses of all students taking Medical Informatics and Advanced Databases.

```
SELECT S.email
FROM Student S, Takes T1, Course C1, Takes T2,
Course C2
WHERE S.mn = T1.mn  AND  T1.cid = C1.cid
   AND S.mn = T2.mn  AND  T2.cid = C2.cid
   AND C1.title = ‘Medical Informatics’
   AND C2.title = ‘Advanced Databases’
```

EXCEPT in SQL

- Find the email addresses of all students taking Medical Informatics but not Advanced Databases.

```
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Medical Informatics’
EXCEPT
SELECT S.email
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Advanced Databases’
```

Nested queries

- Queries that have other queries embedded within them.
- The idea is to use the result of one query to build another one.
- The following query returns the names of all students that have a mark higher than 70 in any course.

```
SELECT DISTINCT S.name
FROM Student S
WHERE S.mn IN ( SELECT T.mn
                FROM Takes T
                WHERE T.mark > 70 )
```

Nested queries

- Queries that have other queries embedded within them.
- The idea is to use the result of one query to build another one.
- The following query returns the names of all students that have a mark higher than 70 in any course.

```
SELECT DISTINCT S.name
FROM Student S
WHERE S.mn IN ( SELECT T.mn
                FROM Takes T
                WHERE T.mark > 70 )
```

Nested queries

- Queries that have other queries embedded within them.
- The idea is to use the result of one query to build another one.
- The following query returns the names of all students that have a mark higher than 70 in any course.

```
SELECT DISTINCT S.name
FROM Student S
WHERE S.mn IN ( SELECT T.mn
                FROM Takes T
                WHERE T.mark > 70 )
```

Nested queries

- We can prefix IN with NOT.
- Find the email addresses of all students that did not take any courses in 2012.

```
SELECT S.email
FROM Student S
WHERE S.mn NOT IN ( SELECT T.mn
                    FROM Takes T
                    WHERE T.year = 2012 )
```

Aggregate operators in SQL

- SQL also allows us to compute aggregate values rather than simply retrieve data.
- Five aggregate operations are available:
  - COUNT([DISTINCT] *field-name*): The number of (unique) values in a particular field
  - SUM([DISTINCT] *field-name*): The total of all (unique) values in a particular field
  - AVG([DISTINCT] *field-name*): The mean of all (unique) values in a particular field
  - MAX(*field-name*): The maximum value in a particular field
  - MIN(*field-name*): The minimum value in a particular field

Aggregate operators in SQL

- Find the average age of all students taking Medical Informatics.

```
SELECT AVG(S.age)
FROM Student S, Takes T, Course C
WHERE S.mn = T.mn
   AND T.cid = C.cid
   AND C.title = ‘Medical Informatics’
```

Aggregate operators in SQL

- Find the number of students taking Medical Informatics in 2016, their average mark and their highest mark.

```
SELECT COUNT(DISTINCT T.mn), AVG(T.mark),
       MAX(T.mark)
FROM Takes T, Course C
WHERE T.cid = C.cid
   AND C.title = ‘Medical Informatics’
   AND T.year = 2016
```

Conclusions

- We got to formulate more expressive SQL queries with the use of:
  - SQL set operators (e.g. UNION)
  - nested queries
  - aggregate operators, (e.g. AVG)
- This concludes the first part of the course on Relational Databases.

Acknowledgements

The content of these slides was originally created for the Medical Informatics course from The University of Edinburgh, which is licensed under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license.