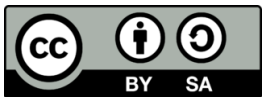
THE UNIVERSITY of EDINBURGH

Data Science in Medicine

Lecture 7: Querying Relational Databases with SQL

Dr Areti Manataki

Usher Institute
The University of Edinburgh



Usher Institute
Population Health Sciences & Informatics

In the previous lecture

- Relational Database: a set of tables with rows and columns, and links between them

Student

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@inf	18
s1489673	Stuart	stuart@law	34
s1473612	Alan	alan@law	20

Course

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Takes

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	inf1
s1489673	sls

In the previous lecture

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student,
    FOREIGN KEY (cid) REFERENCES Course
 )
```

- The primary key uniquely identifies a tuple. No two rows in the Takes table have the same combination of mn and cid.
- Foreign key constraints specify links between tables. The foreign key on mn sspecifies that mn will always take a value that appears in the Student table.

In the previous lecture

- Systematically transform an ER model into a relational one
- Transforming:
  - entity and relationship sets
  - key and participation constraints
  - weak entity sets and hierarchies

In this lecture

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

Inserting rows into a table

```
CREATE TABLE Student (
    mn CHAR(8),
    name CHAR(20),
    email CHAR(25),
    age INTEGER,
    PRIMARY KEY (mn) )
```

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

- The above statement adds a tuple in the Student table.
- We could omit the list of column names and simply list the values in the appropriate order, but it is good practice to include column names.

Deleting and updating rows

- We can delete tuples using the DELETE command

```
DELETE
    FROM Student
    WHERE name = ‘Alan’
```

- We can update the column values in an existing row using the UPDATE command

```
UPDATE Student
    SET name = ‘Alan’
    WHERE mn = ‘s1428571’
```

SQL queries

- SQL allows us to ask questions to the database, such as:
  - Which students are older than 19?
  - What are the names of all students taking the Medical Informatics course?
  - What is the average age of all students born in Europe who are taking the Medical Informatics course but not the Advanced Databases course?

A simple SQL query

- The following query returns all students older than 19.

```
SELECT *
FROM Student
WHERE age > 19
```

- The * means that the table returned has the same schema as Students.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

A simple SQL query

- The following query returns all students older than 19.

```
SELECT *
FROM Student
WHERE age > 19
```

- The * means that the table returned has the same schema as Students.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

A simple SQL query

- The following query returns all students older than 19.

```
SELECT *
FROM Student
WHERE age > 19
```

- The * means that the table returned has the same schema as Students.

mn	name	email	age
s1253477	Jenny	jenny@inf	23
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

SQL query syntax

```
SELECT [DISTINCT] field-list
FROM table-list
[ WHERE qualification ]
```

- Anything in [*square brackets*] is optional.
- SELECT: the columns to be retained in the result
- FROM: the tables from which to take the data
- WHERE: conditions that should hold for the records to be picked out

Variations of a simple SQL query

- Instead of using *, we can explicitly specify the list of fields to be returned. These could be in a different order than in the original table.

```
SELECT *
FROM Student
WHERE age > 19
```

```
SELECT mn, name, email, age
FROM Student
WHERE age > 19
```

Variations of a simple SQL query

- We can specify which tables the fields are from.
- This is particularly useful when the FROM-clause includes several tables.

```
SELECT *
FROM Student
WHERE age > 19
```

```
SELECT Student.mn, Student.name,
       Student.email, Student.age
FROM Student
WHERE Student.age > 19
```

Variations of a simple SQL query

- We can specify which tables the fields are from, while locally abbreviating their names.
- This is particularly useful when the FROM-clause includes several tables.

```
SELECT *
FROM Student
WHERE age > 19
```

```
SELECT S.mn, S.name, S.email,
       S.age
FROM Student S
WHERE S.age > 19
```

Additional SQL queries

- We may choose to select only a subset of the fields of each selected tuple.

```
SELECT S.name
FROM Student S
WHERE S.age > 19
```

- In this case, the table returned has a different schema to that in Student.

name
Jenny
Stuart
Alan

Additional SQL queries

- We may choose not to specify a condition through the WHERE-part of the query.

```
SELECT age
FROM Student
```

age
19
23
18
34
23

- By using DISTINCT, we remove any duplicates from the returned records.

```
SELECT DISTINCT age
FROM Student
```

age
19
23
18
34

Additional SQL queries

- We can include several tables in the FROM-clause.
- The following query returns the email addresses of all students taking Medical Informatics.

```
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'
```

Query evaluation

```
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'
```

1. Take all rows from the tables.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	medinf
s1489673	medinf
s1473612	sls

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
medinf	Medical Informatics	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Query evaluation

```
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'
```

1. Take all rows from the tables.
2. Keep only the row combinations that satisfy the qualification conditions.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	medinf
s1489673	medinf
s1473612	sls

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
medinf	Medical Informatics	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Query evaluation

```
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'
```

1. Take all rows from the tables.
2. Keep only the row combinations that satisfy the qualification conditions.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	medinf
s1489673	medinf
s1473612	sls

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
medinf	Medical Informatics	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Query evaluation

```
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'
```

1. Take all rows from the tables.
2. Keep only the row combinations that satisfy the qualification conditions.
3. Return the specified columns.

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@med	18
s1489673	Stuart	stuart@med	34
s1473612	Alan	alan@law	23

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	medinf
s1489673	medinf
s1473612	sls

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
medinf	Medical Informatics	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Query evaluation

```
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'
```

1. Take all rows from the tables.
2. Keep only the row combinations that satisfy the qualification conditions.
3. Return the specified columns.

email
jenny@inf
stuart@med

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've been introduced to the SQL Data Manipulation Language to:
  - insert, delete and update rows in a table
  - query the database
- General form of a basic SQL query:

```
SELECT [DISTINCT] field-list
FROM table-list
[ WHERE qualification ]
```

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 second part of the course on Relational Databases.