Informatics 1: Data & Analysis

Lecture 8: SQL Queries

Ian Stark

School of Informatics
The University of Edinburgh

Friday 7 February 2014
Semester 2 Week 4

Teaching Weeks !

This is Inf1-DA Lecture 8, in Week 4.

Next week is Teaching Week 5, with lectures and tutorials as usual.

After that is Innovative Learning Week (ILW). All lectures, tutorials, labs and coursework are suspended for the week, and replaced by a series of alternative events across the University.

http://www.ed.ac.uk/innovative-learning

Check the ILW calendar and sign up now: some activities run all week, some are one-off events.

The following week, from Monday 24 February, is Teaching Week 6.

Not all courses have noticed that this is the week numbering scheme.

Lecture Plan for Weeks 1–4

Data Representation

This first course section starts by presenting two common data representation models.

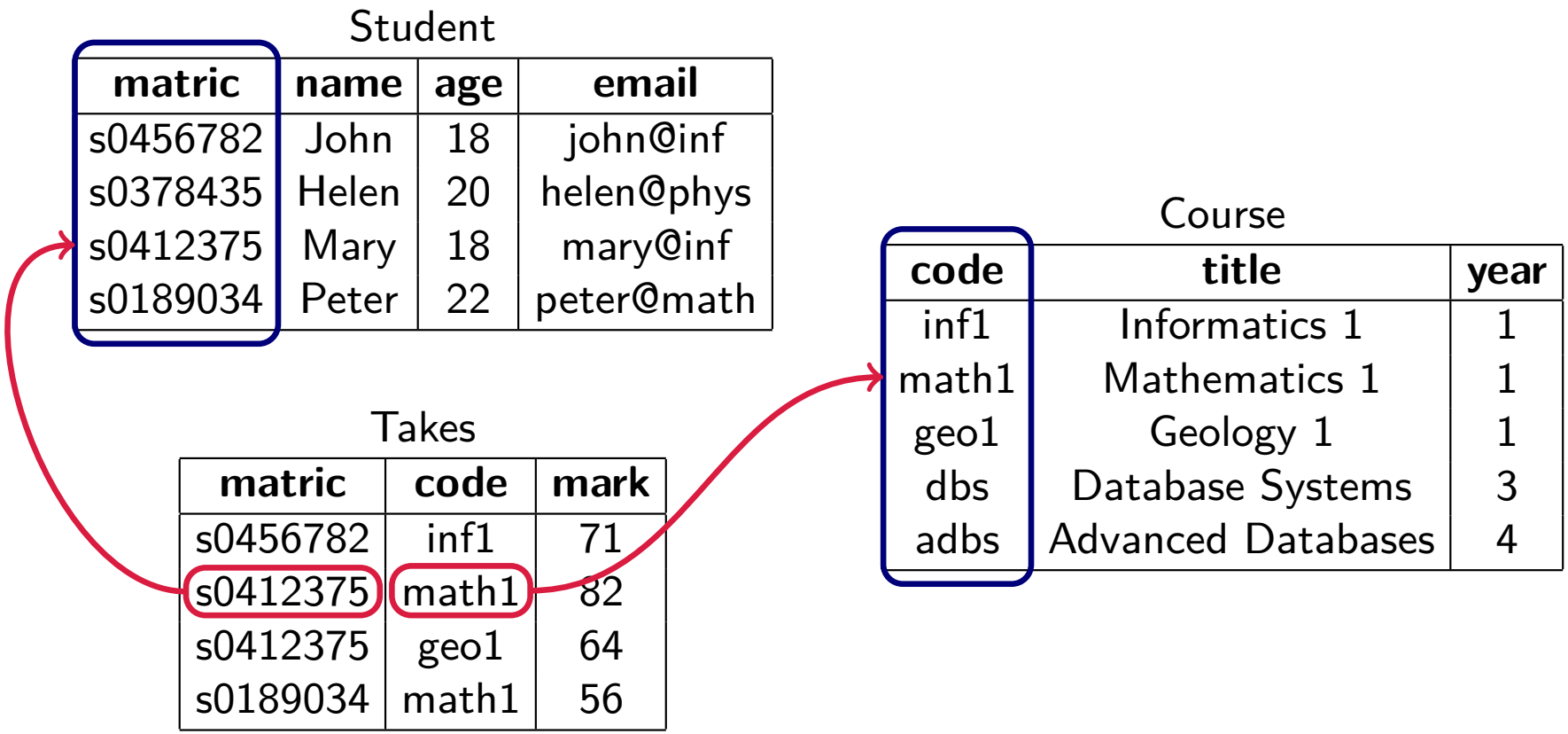
- The *entity-relationship (ER)* model
- The *relational* model

Data Manipulation

This is followed by some methods for manipulating data in the relational model and using it to extract information.

- *Relational algebra*
- The *tuple-relational calculus*
- The query language *SQL*

Students and Courses



Simple Query

Extract all records for students older than 19.

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

Returns a new table, with the same schema as Student, but containing only some of its rows.

$$\{ S \mid S \in \text{Student} \wedge S.\text{age} > 19 \}$$

matric	name	age	email
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once,

Student

matric	name	age	email
s0456782	John	18	john@inf
s0378435	Helen	20	helen@phys
s0412375	Mary	18	mary@inf
s0189034	Peter	22	peter@math

Takes

matric	code	mark
s0456782	inf1	71
s0412375	math1	82
s0412375	geo1	64
s0189034	math1	56

Course

code	title	year
inf1	Informatics 1	1
math1	Mathematics 1	1
geo1	Geology 1	1
dbs	Database Systems	3
adbs	Advanced Databases	4

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test,

Student

matric	name	age	email
s0456782	John	18	john@inf
s0378435	Helen	20	helen@phys
s0412375	Mary	18	mary@inf
s0189034	Peter	22	peter@math

Takes

matric	code	mark
s0456782	inf1	71
s0412375	math1	82
s0412375	geo1	64
s0189034	math1	56

Course

code	title	year
inf1	Informatics 1	1
math1	Mathematics 1	1
geo1	Geology 1	1
dbs	Database Systems	3
adbs	Advanced Databases	4

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test,

Student

matric	name	age	email
s0456782	John	18	john@inf
s0378435	Helen	20	helen@phys
s0412375	Mary	18	mary@inf
s0189034	Peter	22	peter@math

Takes

matric	code	mark
s0456782	inf1	71
s0412375	math1	82
s0412375	geo1	64
s0189034	math1	56

Course

code	title	year
inf1	Informatics 1	1
math1	Mathematics 1	1
geo1	Geology 1	1
dbs	Database Systems	3
adbs	Advanced Databases	4

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test, and return the named columns.

Student

matric	name	age	email
s0456782	John	18	john@inf
s0378435	Helen	20	helen@phys
s0412375	Mary	18	mary@inf
s0189034	Peter	22	peter@math

Takes

matric	code	mark
s0456782	inf1	71
s0412375	math1	82
s0412375	geo1	64
s0189034	math1	56

Course

code	title	year
inf1	Informatics 1	1
math1	Mathematics 1	1
geo1	Geology 1	1
dbs	Database Systems	3
adbs	Advanced Databases	4

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test, and return the named columns.

name
Mary
Peter

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test, and return the named columns.

Expressed in tuple-relational calculus:

$$\{ \mathrm{R} \mid \exists S \in \mathrm{Student}, \mathrm{T} \in \mathrm{Takes}, \mathrm{C} \in \mathrm{Course} \,.$$
$$\mathrm{R.name} = S.\mathrm{name} \wedge S.\mathrm{matric} = \mathrm{T.matric}$$
$$\wedge \mathrm{T.code} = \mathrm{C.code} \wedge \mathrm{C.title} = \text{"Mathematics 1"} \}$$

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test, and return the named columns.

Implemented in relational algebra,

$$\pi_{\text{name}}(\sigma_{\begin{array}{r}\text{Student.matric} = \text{Takes.matric} \\ \wedge\ \text{Takes.code} = \text{Course.code} \\ \wedge\ \text{Course.name} = \text{"Mathematics 1"}\end{array}}(\text{Student} \times \text{Takes} \times \text{Course}))$$

Example Query

Find the names of all students who are taking Mathematics 1

```
SELECT Student.name
FROM Student, Takes, Course
WHERE  Student.matric = Takes.matric
    AND  Takes.code = Course.code
    AND  Course.title = 'Mathematics 1'
```

Take rows from all three tables at once, pick out only those row combinations which match the test, and return the named columns.

Implemented in relational algebra, in several possible ways:

$$\pi_{\text{name}}(\sigma_{\text{title}=\text{"Mathematics 1"}}(\text{Student} \bowtie \text{Takes} \bowtie \text{Course}))$$

$$\pi_{\text{name}}(\text{Student} \bowtie (\text{Takes} \bowtie (\sigma_{\text{title}=\text{"Mathematics 1"}}(\text{Course}))))$$

Travelling Salesman

Travelling Salesman

Given a table of travel times between all n^2 pairs of towns:

- What is the quickest route to visit them all?
- Is there any route shorter than X?

Checking any route is fast.

However, there a lot of routes to check ($n!$).

Can we do it faster?

Not much, no. The fastest algorithm known takes time related to 2^n.

This exponential growth means that a small problem can quickly become very large.

Travelling Salesman

This travelling salesman problem is more widely applicable than the name suggests, and it also arises in:

- Commercial distribution logistics;
- Optimising chip layout;
- Assembling DNA sequence fragments;

etc.

Travelling Salesman

The travelling salesman problem is **NP-hard**: one of a large class of equivalent computational challenges where:

- Checking a potential solution is quick. . .
- . . . but there are a lot of potential solutions. . .
- . . . and we don't know how to do it any faster.

There's a bounty on these. You find a fast way to solve travelling salesman, or a proof that there isn't one, and you collect $1M.

Apply to the *Clay Mathematics Institute*, 10 Memorial Boulevard, Providence, Rhode Island, USA.

Travelling Salesman

P.S. If you are an actual salesman:

- Distances based on real space simplify the general problem;
- There are fast algorithms which with very high probability return an answer very close to the optimum.

Other NP-hard problems are not be so easy to work around.

Limits of Computation

Various physical factors set upper bounds on how much computation is possible:

- The speed of light, c;
- Planck's constant, $\hbar$;
- Universal gravitational constant, G;
- Quantisation of energy states;
- Heisenberg uncertainty in observation;
- Mass/energy equivalence.

For a 1kg computer, this sets a limit on computation of 10^{50} operations per second on 10^{31} bits.

Limits of Computation

Various physical factors set upper bounds on how much computation is possible:

- The speed of light, c;
- Planck's constant, $\hbar$;
- Universal gravitational constant, G;
- Quantisation of energy states;
- Heisenberg uncertainty in observation;
- Mass/energy equivalence.

For a 1kg computer, this sets a limit on computation of 10^{50} operations per second on 10^{31} bits.

Working at 10^9K, "the ultimate laptop looks like a small piece of the big bang".

[Seth Lloyd, *Ultimate Physical Limits to Computation*, Nature 406:1045–154, August 2000]

Limits of Computation

Matter organised to provide the greatest possible computing power is fancifully known as computronium.

In the 1960's Hans-Joachim Bremermann was one of the first people to estimate upper limits to computation.

Limits of Computation

Matter organised to provide the greatest possible computing power is fancifully known as computronium.

In the 1960's Hans-Joachim Bremermann was one of the first people to estimate upper limits to computation.

His *Bremermann limit* is the computation which could be performed using the earth, over the period of its existence so far.

Limits of Computation

Matter organised to provide the greatest possible computing power is fancifully known as computronium.

In the 1960's Hans-Joachim Bremermann was one of the first people to estimate upper limits to computation.

His *Bremermann limit* is the computation which could be performed using the earth, over the period of its existence so far.

This is around 10^{93} bits of computation.

Limits of Computation

Matter organised to provide the greatest possible computing power is fancifully known as computronium.

In the 1960's Hans-Joachim Bremermann was one of the first people to estimate upper limits to computation.

His *Bremermann limit* is the computation which could be performed using the earth, over the period of its existence so far.

This is around 10^{93} bits of computation.

That's enough to solve the travelling salesman problem for 300 cities.

Limits of Computation

Matter organised to provide the greatest possible computing power is fancifully known as computronium.

In the 1960's Hans-Joachim Bremermann was one of the first people to estimate upper limits to computation.

His *Bremermann limit* is the computation which could be performed using the earth, over the period of its existence so far.

This is around 10^{93} bits of computation.

That's enough to solve the travelling salesman problem for 300 cities.

But just the once.

Disjunction Query

Find the names of all students who are taking *either* Informatics 1 *or* Mathematics 1.

```
SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND (C.title = 'Informatics 1'
            OR C.title = 'Mathematics 1')
```

Disjunction Query

Find the names of all students who are taking *either* Informatics 1 *or* Mathematics 1.

```
SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Informatics 1'

UNION

SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Mathematics 1'
```

Conjunction Query

Find the names of all students who are taking *both* Informatics 1 *and* Mathematics 1.

```
SELECT S.name
FROM Student S, Takes T1, Course C1, Takes T2, Course C2
WHERE  S.matric = T1.matric AND T1.code = C1.code
    AND S.matric = T2.matric AND T2.code = C2.code
    AND C1.title = 'Informatics 1'
    AND C2.title = 'Mathematics 1'
```

Conjunction Query

Find the names of all students who are taking *both* Informatics 1 *and* Mathematics 1.

```
SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Informatics 1'

INTERSECT

SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Mathematics 1'
```

Difference Query

Find the names of all students who are taking Informatics 1 *but not* Mathematics 1.

```
SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Informatics 1'

EXCEPT

SELECT S.name
FROM Student S, Takes T, Course C
WHERE  S.matric = T.matric AND T.code = C.code
    AND  C.title  = 'Mathematics 1'
```

Comparison Query

Find the students' names in all cases where one person scored higher than another in Mathematics 1.

```
SELECT S1.name AS "Higher", S2.name AS "Lower"
FROM Student S1, Takes T1, Student S2, Takes T2, Course C
WHERE S1.matric = T1.matric AND T1.code = C.code
    AND S2.matric = T2.matric AND T2.code = C.code
    AND C.title = 'Informatics 1'
    AND T1.mark > T2.mark
```

Higher	Lower
Mary	Peter

Aggregates: Operations on Multiple Values

SQL includes a range of mathematical operations on individual values, like T1.mark $>$ T2.mark.

SQL also provides operations on whole collections of values, as returned in a **SELECT** query. There are five of these standard aggregate operations:

COUNT(val)	The number of values in the val field
SUM(val)	The total of all values in the val field
AVG(val)	The mean of all values in the val field
MAX(val)	The greatest value in the val field
MIN(val)	The least value in the val field

Particular RDBMS implementations may refine and extend these with other operations.

Aggregates: Operations on Multiple Values

SQL includes a range of mathematical operations on individual values, like T1.mark > T2.mark.

SQL also provides operations on whole collections of values, as returned in a **SELECT** query. There are five of these standard aggregate operations:

COUNT(**DISTINCT** val)	The number of distinct values in the val field
SUM(**DISTINCT** val)	The total of the distinct values in the val field
AVG(**DISTINCT** val)	The mean of the distinct values in the val field
MAX(val)	The greatest value in the val field
MIN(val)	The least value in the val field

Particular RDBMS implementations may refine and extend these with other operations.

Aggregating Query

Find the number of students taking Informatics 1, their mean mark, and the highest mark.

```
SELECT COUNT(DISTINCT T.matric) AS "Number",
       AVG(T.mark) AS "Mean Mark",
       MAX(T.mark) AS "Highest"
FROM Student S, Takes T, Course C
WHERE S.matric = T.matric AND T.code = C.code
  AND C.title = 'Informatics 1'
```

Who Writes SQL?

SQL is one of the world's most widely used programming languages, but programs in SQL come from many sources. For example:

- Hand-written by a programmer
- Generated by some interactive visual tool
- Generated by an application to fetch an answer for a user
- Generated by one program to request information from another

Most SQL is written by programs, not directly by programmers.

The same is true of HTML, another domain-specific language.

Also XML, Postscript,...

Homework

Read one of these two articles.

Inside Google spanner, the largest single database on earth.
C. Metz. *Wired Enterprise*, November 2012.
http://www.wired.com/wiredenterprise/2012/11/google-spanner-time/

Spanner: Google's globally-distributed database.
J. C. Corbett et al. In *OSDI '12: Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation*, pages 251–264. The USENIX Association, 2012.
http://research.google.com/archive/spanner.html

Try some SQL by hand using one of these web demonstrators.

w3schools.com	http://is.gd/trysql
SQL Fiddle	http://sqlfiddle.com
SkyServer	http://skyserver.sdss3.org

SQL Statement: Edit the SQL Statement, and click "Run SQL" to see the result.

```
SELECT * FROM Products WHERE Price>50
```

Run SQL »

Result:

Number of Records: 7

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5
20	Sir Rodney's Marmalade	8	3	30 gift boxes	81
29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123.79
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5
51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53
59	Raclette Courdavault	28	4	5 kg pkg.	55

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Restore Database

SQL Fiddle MySQL 5.5.32

```
CREATE TABLE Student (
  matric VARCHAR(8),
  name   VARCHAR(20),
  age    INTEGER,
  email  VARCHAR(25),
  PRIMARY KEY (matric) );

INSERT
  INTO Student (matric, name, age, email
  VALUES ('s1428751', 'Bob', 20, 'bob@sm
```

Build Schema | Edit Fullscreen | Browser | [;]

```
SELECT matric, name
FROM Student
WHERE age > 19
```

Run SQL | Edit Fullscreen | Format Code | [;]

MATRIC	NAME
s1428751	Bob

Record Count: 1; Execution Time: 0ms | View Execution Plan | link

Did this query solve the problem? If so, consider donating $5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

SDSS-III Survey Instruments

Apache Point Observatory
(SDSS 2.5m telescope at left)

DR10 Tools

SQL Search

This page allows you to directly submit a SQL (Structured Query Language) query to the SDSS database server. You can modify the default query as you wish, or cut and paste a query from the SDSS Sample Queries page.

Please note: To be fair to other users, queries run from SkyServer search tools are restricted in how long they can run and how much output they return, by **timeouts** and **row limits**. Please see the Query Limits help page. To run a query that is not restricted by a timeout or number of rows returned, please use the CasJobs batch query service.

Clear Query

```
-- Find some stars near a certain spot in the sky
SELECT top 10
 p.objId,
 p.run, p.rerun, p.camcol, p.field, p.obj,
 p.type, p.ra, p.dec
FROM PhotoTag p, fGetNearbyObjEq(40.433,0.449,3) n
WHERE n.objID=p.objID and p.type=3
```

Submit | Check Syntax Only? | Output Format: (•) HTML ( ) XML ( ) CSV ( ) JSON ( ) VOTable ( ) FITS | Reset

To find out more about the database schema use the Schema Browser.

For an introduction to the Structured Query Language (SQL), please see the Searching for Data How-To tutorial. In particular, please read the Optimizing Queries section.

The inclusion of the imaging and spectro columns for SAS upload in your query (as in the default query on this page) will ensure that when you press **Submit**, the appropriate button(s) are displayed on the query results page to allow you to upload the necessary information to the SAS to retrieve the FITS file data corresponding