Informatics 1: Data & Analysis

Lecture 8: SQL Queries

Ian Stark

School of Informatics
The University of Edinburgh

Friday 8 February 2013
Semester 2 Week 4

Lecture Plan

Data Representation

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

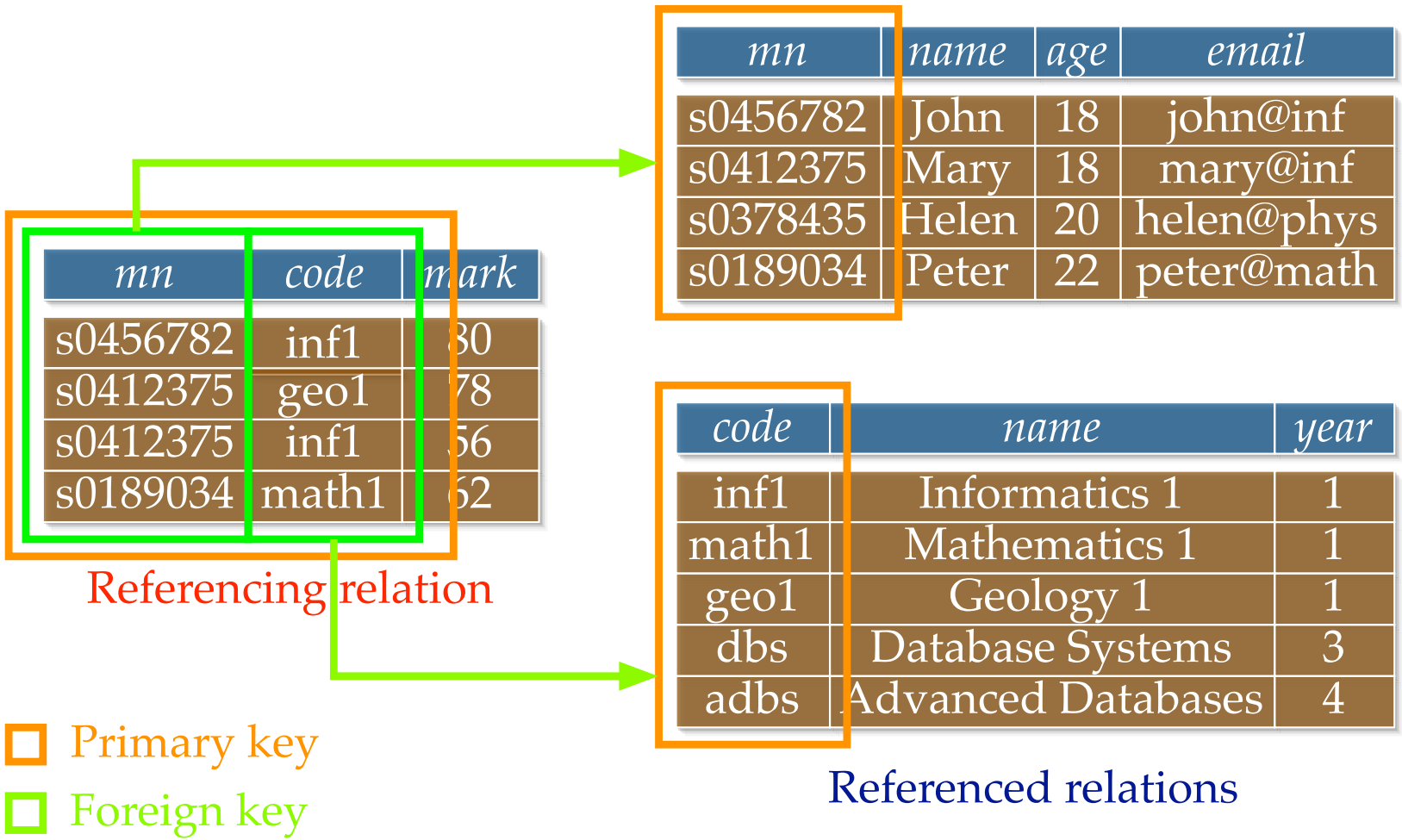
- The *entity-relationship (ER)* model
- The *relational* model

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 from the table for students older than 18:

```
SELECT *
FROM Students
WHERE age > 18
```

This will return a new table, with the same schema as Students, but containing only some of the rows.

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

Example Query

Find the names of all students who are taking Informatics 1

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND C.name = 'Informatics 1'
```

This joins all three tables,

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all students who are taking Informatics 1

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND C.name = 'Informatics 1'
```

This joins all three tables, selects rows that match,

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all students who are taking Informatics 1

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Informatics 1'
```

This joins all three tables, selects rows that match, and projects out a single column.

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all students who are taking Informatics 1

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Informatics 1'
```

This joins all three tables, selects rows that match, and projects out a single column.

name
John
Mary

Example Query

Find the names of all courses taken by (any student called) Mary

```
SELECT C.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND S.name = 'Mary'
```

This joins all three tables,

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all courses taken by (any student called) Mary

```
SELECT C.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND S.name = 'Mary'
```

This joins all three tables, selects rows that match,

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all courses taken by (any student called) Mary

```
SELECT C.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND S.name = 'Mary'
```

This joins all three tables, selects rows that match, and projects out a single column.

mn	name	age	email	code	name	year	mark
s0456782	John	18	john@inf	inf1	Informatics 1	1	80
s0412375	Mary	18	mary@inf	geo1	Geology 1	1	78
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1	56
s0189034	Peter	22	peter@math	math1	Mathematics 1	1	62

Example Query

Find the names of all courses taken by (any student called) Mary

```
SELECT C.name
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND S.name = 'Mary'
```

This joins all three tables, selects rows that match, and projects out a single column.

name
Geology 1
Informatics 1

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.

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 Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
    AND (C.name = 'Informatics 1' OR C.name = 'Mathematics 1')
```

Disjunction Query

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

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Informatics 1'
UNION
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Mathematics 1'
```

Conjunction Query

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

```
SELECT S.name
FROM Students S, Takes T1, Courses C1, Takes T2, Courses C2
WHERE  S.mn = T1.mn AND T1.code = C1.code
    AND S.mn = T2.mn AND T2.code = C2.code
    AND C1.name = 'Informatics 1'
    AND C2.name = 'Mathematics 1'
```

Conjunction Query

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

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Informatics 1'
INTERSECT
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Mathematics 1'
```

Difference Query

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

```
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Informatics 1'
EXCEPT
SELECT S.name
FROM Students S, Takes T, Courses C
WHERE  S.mn = T.mn AND T.code = C.code
    AND  C.name = 'Mathematics 1'
```

Comparison Query

Find the matriculation numbers of all cases where one student scored higher than another in Informatics 1.

```
SELECT S1.mn AS "Higher", S2.mn AS "Lower"
FROM Students S1, Takes T1, Students S2, Takes T2, Courses C
WHERE S1.mn = T1.mn AND T1.code = C.code
    AND S2.mn = T2.mn AND T2.code = C.code
    AND C.name = 'Informatics 1'
    AND T1.mark > T2.mark
```

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.mn) AS "Number",
       AVG(T.mark) AS "Mean Mark",
       MAX(T.mark) AS "Highest"
FROM Students S, Takes T, Courses C
WHERE S.mn = T.mn AND T.code = C.code
  AND C.name = 'Informatics 1'
```