SQL SELECT

G51DBS Database Systems

Jason Atkin

jaa@cs.nott.ac.uk

This Lecture

- SQL SELECT
  - WHERE Clauses
  - SELECT from multiple tables
  - JOINs
- Further reading
  - The Manga Guide to Databases, Chapter 4
  - Database Systems, Chapter 6

SQL SELECT Overview

```
SELECT
  [DISTINCT | ALL] <column-list>
  FROM <table-names>
  [WHERE <condition>]
  [ORDER BY <column-list>]
  [GROUP BY <column-list>]
  [HAVING <condition>]
```

([] *optional*, | *or*)

Example Tables

Student

sID	sFirst	sLast
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

Module

mCode	mTitle
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Introduction to AI

Grade

sID	mCode	gMark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

The SQL for these tables

```
CREATE TABLE Student (
        sID VARCHAR(5)
                PRIMARY KEY,
        sFirst VARCHAR(50)
                NOT NULL,
        sLast VARCHAR(50)
                NOT NULL
) ENGINE=InnoDB;

CREATE TABLE Module (
        mCode VARCHAR(7)
                PRIMARY KEY,
        mTitle VARCHAR(50)
                NOT NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE GradeFK (
        sID VARCHAR(5) NOT NULL,
        mCode VARCHAR(7) NOT NULL,
        gMark INT,
CONSTRAINT
        PRIMARY KEY (sID, mCode),
CONSTRAINT fkgrastu
        FOREIGN KEY (sID)
        REFERENCES Student(sID),
CONSTRAINT fkgramod
        FOREIGN KEY (mCode)
        REFERENCES Module(mCode)
) ENGINE=InnoDB;
```

DISTINCT and ALL

- Sometimes you end up with duplicate entries
- Using DISTINCT removes duplicates
- Using ALL retains duplicates
  - ALL is used as a default if neither is supplied
- These will work over multiple columns

```
SELECT ALL sLast
  From Student;
```

sLast
Smith
Jones
Brown
Jones
Brown

```
SELECT DISTINCT sLast
  FROM Student;
```

sLast
Smith
Jones
Brown

WHERE Clauses

- In most cases returning all of the rows is not necessary
- A WHERE clause specifies which rows should be returned
- It takes the form of a condition – only rows that satisfy the condition are returned

- Example conditions:
  - `Mark < 40`
  - `First = 'John'`
  - `First <> 'John'`
  - `First = Last`
  - `(First = 'John')`
    `AND`
    `(Last = 'Smith')`
  - `(Mark < 40)`
    `OR`
    `(Mark > 70)`

WHERE Examples

```
SELECT
*
FROM Grade
WHERE gMark >= 60
```

sID	mCode	gMark
S103	DBS	72
S104	PR1	68
S104	IAI	65
S107	PR1	76
S107	PR2	60

```
SELECT
DISTINCT sID
FROM Grade
WHERE gMark >= 60
```

sID
S103
S104
S107

WHERE Examples

- Given the table:

Grade

sID	mCode	gMark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

- **QUESTION:**
  Write an SQL query to find a list of the ID numbers and Marks for students who have passed IAI (scored 40% or more), i.e.:

sID	gMark
S103	58
S104	65

Solution

```
SELECT sID, gMark FROM Grade
    WHERE (mCode = 'IAI')
    AND (gMark >= 40)
```

SELECT from Multiple Tables

- Often you need to combine information from two or more tables
- You can produce the effect of a Cartesian product using:

```
SELECT * FROM
  Table1, Table2
```

- If the tables have columns with the same name, ambiguity will result
- This can be resolved by referencing columns with the table name:

```
TableName.ColumnName
```

SELECT from Multiple Tables

```
SELECT
  First, Last, Mark
FROM
  Student, Grade
WHERE
  (Student.ID =
     Grade.ID)
  AND (Mark >= 40)
```

Student

ID	First	Last
S103	John	Smith
S104	Mary	Jones
S105	Jane	
S106	Marl	
S107	John	

Grade

ID	Code	Mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

Note: For simplicity I will assume columns are labelled ID, First, etc rather than sID, sFirst, etc. And that no foreign key constraints exist or are enforced.

SELECT from Multiple Tables

```
SELECT ... FROM Student, Grade WHERE ...
```

ID	First	Last	ID	Code	Mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72

SELECT from Multiple Tables

```
SELECT ... FROM Student, Grade
   WHERE (Student.ID = Grade.ID) AND ...
```

ID	First	Last	ID	Code	Mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60
S107	John	Brown	S107	IAI	35

SELECT from Multiple Tables

```
SELECT ... FROM Student, Grade
   WHERE (Student.ID = Grade.ID)AND(Mark >= 40)
```

ID	First	Last	ID	Code	Mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60

SELECT from Multiple Tables

```
SELECT First, Last, Mark FROM Student, Grade
   WHERE (Student.ID = Grade.ID) AND (Mark >= 40)
```

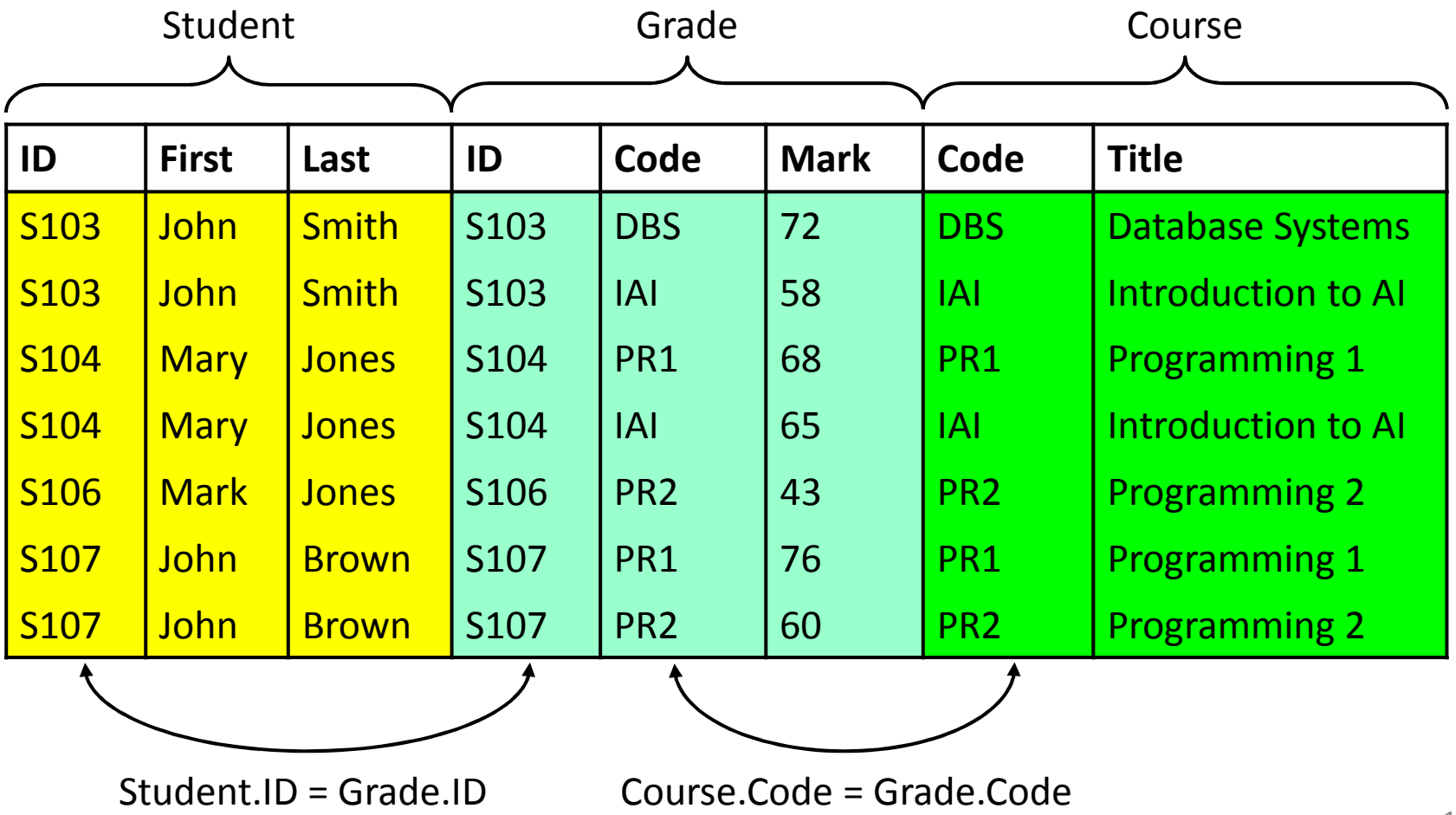
First	Last	Mark
John	Smith	72
John	Smith	58
Mary	Jones	68
Mary	Jones	65
Mark	Jones	43
John	Brown	76
John	Brown	60

SELECT from Multiple Tables

- When selecting from multiple tables, you will almost always use a WHERE clause
  - Or equivalent filter
- You will usually be looking to match columns
- Particularly foreign keys to candidate keys

```
SELECT *
From
   Student, Grade,
   Course
WHERE
   Student.ID =
   Grade.ID
   AND
   Course.Code =
   Grade.Code
```

SELECT from Multiple Tables



Joins

- JOINs can be used to combine tables in a SELECT query
- There are numerous types of JOIN
  - **`CROSS JOIN`**
  - **`INNER JOIN`**
  - **`NATURAL JOIN`**
  - **`OUTER JOIN`**

- **`OUTER JOIN`** will be discussed later – they are linked with NULLs

`A CROSS JOIN B`

- Returns all pairs of rows from A and B
- The Cartesian Product

`A INNER JOIN B`

- Returns pairs of rows satisfying a condition

`A NATURAL JOIN B`

- Returns pairs of rows with common values **in identically named columns**

CROSS JOIN

```
SELECT * FROM
   A CROSS JOIN B
```

- Is the same as

```
SELECT * FROM A, B
```

- ANSI SQL defines a CROSS JOIN as a join without a condition which builds the Cartesian product of two tables

- Often common to use a **WHERE** clause, to avoid huge result sets
- Without a **WHERE** clause, the number of rows produced will be equal to the number of rows in **A multiplied by** the number of rows in **B**.
- **But:** If you use a **WHERE**, an **INNER JOIN** may be a better choice.

CROSS JOIN

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code
123	DBS
124	PRG
124	DBS
126	PRG

```
SELECT * FROM
   Student CROSS JOIN
   Enrolment
```

ID	Name	ID	Code
123	John	123	DBS
124	Mary	123	DBS
125	Mark	123	DBS
126	Jane	123	DBS
123	John	124	PRG
124	Mary	124	PRG
125	Mark	124	PRG

INNER JOIN

- **`INNER JOIN`** specifies a condition that pairs of rows must satisfy

```
SELECT *
   FROM A INNER JOIN B
   USING (col1, col2)
```

```
SELECT *
   FROM A INNER JOIN B
   ON <condition>
```

- If columns have the same names in both **`A`** and **`B,`** use a **`USING`** clause
  - Will output rows with equal values in the specified columns

- If columns are different, or you need a different condition, then use an 'on' clause, e.g.:

```
ON  a.col = b.col
```

INNER JOIN

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code
123	DBS
124	PRG
124	DBS
126	PRG

```
SELECT * FROM
   Student INNER JOIN
   Enrolment USING (ID)
```

ID	Name	Code
123	John	DBS
124	Mary	PRG
124	Mary	DBS
126	Jane	PRG

- A single ID column will be output representing the equal values from both Student.ID and Enrolment.ID

INNER JOIN

Buyer

Name	Budget
Smith	100,000
Jones	150,000
Green	80,000

Property

Address	Price
15 High Street	85,000
12 Queen Street	125,000
87 Oak Lane	175,000

```
SELECT * FROM
   Buyer INNER JOIN Property
   ON Price <= Budget
```

Name	Budget	Address	Price
Smith	100,000	15 High Street	85,000
Jones	150,000	15 High Street	85,000
Jones	150,000	12 Queen Street	125,000

NATURAL JOIN

```
SELECT * FROM
   A NATURAL JOIN B
```

- Is the same as

```
SELECT A.Col1, A.Col2,
   ... , A.Coln,
   [and all other columns
   except for
   B.Col1,...,B.Coln]
FROM A, B
WHERE A.Col1 = B.Col1
   AND ...
   AND A.Coln = B.Coln
```

- Join using **ALL** identically named columns
- A **`NATURAL JOIN`** is effectively a special case of an **`INNER JOIN`** where the **`USING`** clause has specified **all identically named** columns
- Note: Only one copy of each matched column

NATURAL JOIN

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code
123	DBS
124	PRG
124	DBS
126	PRG

```
SELECT * FROM
Student NATURAL JOIN Enrolment
```

ID	Name	Code
123	John	DBS
124	Mary	PRG
124	Mary	DBS
126	Jane	PRG

Every combination of Student and Enrolment, where the ID columns (i.e. same name) match. One copy of each column name.

JOINs vs WHERE Clauses

- Named JOINs are **not** absolutely necessary
  - You can obtain the same results by selecting from multiple tables and using appropriate WHERE clauses
- Should you use named JOIN types?
  - Yes
    - They often lead to more concise and elegant queries
    - Can be much easier to read and understand
    - NATURAL JOINs are extremely common
  - No
    - Support for JOINs can vary between DBMSs
    - Some things might be easier with sub-queries (next lecture)

Examples

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Examples

- Write SQL statements to do the following:
  - Produce a list of all student names and all their enrolments (module codes)
  - Find a list of students who are enrolled on the G52ADS module
  - Find a list of module titles being taken by the student named “Harrison”
  - Find a list of module codes and titles for all modules currently being taken by first year students

Example 1

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Produce a list of all student names and all their enrolments (module codes)

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 1

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Produce a list of all student names and all their enrolments (module codes)

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 1

- Produce a list of all student names and all their enrolments (module codes)

```
SELECT sName, mCode FROM Student
NATURAL JOIN Enrolment;
```

```
SELECT sName, mCode FROM Student
INNER JOIN Enrolment USING
(sID);
```

Example 2

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of students who are enrolled on the G52ADS module

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 2

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of students who are enrolled on the G52ADS module

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 2

- Find a list of students who are enrolled on the G52ADS module

```
SELECT sName FROM Student
NATURAL JOIN Enrolment
WHERE mCode = 'G52ADS';
```

```
SELECT sName FROM Student
INNER JOIN Enrolment
ON Student.sID = Enrolment.sID
WHERE mCode = 'G52ADS';
```

Example 3

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of module titles being taken by the student named "Harrison"

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 3

Student

sID	**sName**	sAddress	sYear
1	**Smith**	5 Arnold Close	2
2	**Brooks**	7 Holly Avenue	2
3	**Anderson**	15 Main Street	3
4	**Evans**	Flat 1a, High Street	2
5	**Harrison**	Newark Hall	1
6	**Jones**	Southwell Hall	1

Module

mCode	mCredits	**mTitle**
G51DBS	10	**Database Systems**
G51PRG	20	**Programming**
G51IAI	10	**Artificial Intelligence**
G52ADS	10	**Algorithms**

Find a list of module titles being taken by the student named "Harrison"

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 3

- Find a list of module titles being taken by the student named “Harrison”

```
SELECT mTitle FROM
Student NATURAL JOIN Enrolment
     NATURAL JOIN Module
WHERE sName = 'Harrison';

SELECT mTitle FROM
Student, Enrolment, Module
WHERE sName = 'Harrison' AND
     Student.sID = Enrolment.sID AND
     Enrolment.mCode = Module.mCode;
```

Example 4

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of module codes and titles for all modules currently being taken by first year students

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 4

Student

sID	sName	sAddress	**sYear**
1	Smith	5 Arnold Close	**2**
2	Brooks	7 Holly Avenue	**2**
3	Anderson	15 Main Street	**3**
4	Evans	Flat 1a, High Street	**2**
5	Harrison	Newark Hall	**1**
6	Jones	Southwell Hall	**1**

Module

mCode	mCredits	**mTitle**
G51DBS	10	**Database Systems**
G51PRG	20	**Programming**
G51IAI	10	**Artificial Intelligence**
G52ADS	10	**Algorithms**

Find a list of module codes and titles for all modules currently being taken by first year students

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 4

- Find a list of module codes and titles for all modules currently being taken by first year students

```
SELECT DISTINCT mCode, mTitle
FROM Student NATURAL JOIN Enrolment
     NATURAL JOIN Module
WHERE sYear = 1;

SELECT DISTINCT Module.mCode, mTitle
     FROM Student, Enrolment, Module
WHERE sYear = 1 AND
     Student.sID = Enrolment.sID AND
     Enrolment.mCode = Module.mCode;
```

Writing Queries

- When writing queries:
  - **There are often many ways to accomplish the same query**
  - Be concerned with **correctness, clarity and conciseness**, in that order
  - **Do not worry hugely about being clever or efficient**
- Most DBMSs have query optimisers
  - Will optimise your query to improve efficiency
  - Simpler queries are easier to optimise
  - A later lecture will cover ways to improve efficiency

Next Lecture

- More SQL SELECT
  - Aliases
  - ‘Self-Joins’
  - Subqueries
  - IN, EXISTS, ANY, ALL
  - LIKE
- Further reading
  - The Manga Guide to Databases, Chapter 4
  - Database Systems, Chapter 6

SQL SELECT II

G51DBS Database Systems

Jason Atkin

jaa@cs.nott.ac.uk

Last Lecture

- WHERE Clauses
- SELECT from multiple tables

```
SELECT * FROM TA, TB;
```

- JOINs
  - CROSS JOIN (Cartesian Product)

```
SELECT * FROM TA CROSS JOIN TB;
```

  - INNER JOIN (Specifies a column or condition)

```
SELECT * FROM TA INNER JOIN TB USING (Col);
SELECT * FROM TA INNER JOIN TB ON (α);
```

  - NATURAL JOIN (Compares columns with identical names)

```
SELECT * FROM TA NATURAL JOIN TB;
```

SQL SELECT Overview

```
SELECT
  [DISTINCT | ALL] <column-list>
  FROM <table-names>
  [WHERE <condition>]
  [ORDER BY <column-list>]
  [GROUP BY <column-list>]
  [HAVING <condition>]
```

([] *optional*, | *or*)

This Lecture

- More SQL SELECT
  - Aliases
  - ‘Self-Joins’
  - Subqueries
  - IN, EXISTS, ANY, ALL
  - LIKE
- Further reading
  - The Manga Guide to Databases, Chapter 4
  - Database Systems, Chapter 6

Aliases

Aliases

- Aliases **rename columns** or **tables**
- Can make names more meaningful
- Can shorten names, making them easier to use
- **Can resolve ambiguous names**
  - Important for self-joins

- Two forms:

Column alias

```
SELECT
column [AS] newName
from table
```

Table alias

```
SELECT * from
table [AS] newName
```

([] *optional*)

Alias Example : Tables

Employee

ID	First
123	John
124	Mary
125	Andy

WorksIn

ID	Dept
123	Marketing
124	Sales
124	Marketing
125	Sales

```
SELECT
   E.ID AS empID,
   E.First, W.Dept
FROM
   Employee AS E,
   WorksIn W,
WHERE
   E.ID = W.ID
```

Note: You normally cannot use a **column** alias in a WHERE clause

Alias Example : Columns

empID	First	Dept
123	John	Marketing
124	Mary	Sales
124	Mary	Marketing
125	Andy	Sales

```
SELECT
   E.ID AS empID,
   E.First, W.Dept
FROM
   Employee AS E,
   WorksIn W
WHERE
   E.ID = W.ID
```

Note: You normally cannot use a **column** alias in a WHERE clause

Self-joins

Aliases and ‘Self-Joins’

- Aliases can be used to copy/rename a table, so that it can **be combined with itself**:

Find all of the people in the same department as ‘Andy’

i.e. find employee names from all rows where the department is the same as the department on the ‘Andy’ row

Employee

Name	**Dept**
John	Marketing
Mary	Sales
Peter	Sales
Andy	Marketing
Anne	Marketing

Aliases and ‘Self-Joins’

Employee A

A

Name	Dept
John	Marketing
Mary	Sales
Peter	Sales
Andy	Marketing
Anne	Marketing

Employee B

B

Name	Dept
John	Marketing
Mary	Sales
Peter	Sales
Andy	Marketing
Anne	Marketing

Aliases and ‘Self-Joins’

```
SELECT ... FROM Employee A, Employee B ...
```

A.Name	A.Dept	B.Name	B.Dept
John	Marketing	John	Marketing
Mary	Sales	John	Marketing
Peter	Sales	John	Marketing
Andy	Marketing	John	Marketing
Anne	Marketing	John	Marketing
John	Marketing	Mary	Sales
Mary	Sales	Mary	Sales

Aliases and 'Self-Joins'

```
SELECT ... FROM Employee A, Employee B ...
WHERE A.Dept = B.Dept
```

A.Name	A.Dept	B.Name	B.Dept
John	Marketing	John	Marketing
~~Mary~~	~~Sales~~	~~John~~	~~Marketing~~
~~Peter~~	~~Sales~~	~~John~~	~~Marketing~~
Andy	Marketing	John	Marketing
Anne	Marketing	John	Marketing
~~John~~	~~Marketing~~	~~Mary~~	~~Sales~~
Mary	Sales	Mary	Sales

Aliases and ‘Self-Joins’

```
SELECT ... FROM Employee A, Employee B
  WHERE A.Dept = B.Dept
```

A.Name	A.Dept	B.Name	B.Dept
John	Marketing	John	Marketing
Andy	Marketing	John	Marketing
Anne	Marketing	John	Marketing
Mary	Sales	Mary	Sales
Peter	Sales	Mary	Sales
Mary	Sales	Peter	Sales
Peter	Sales	Peter	Sales

Aliases and ‘Self-Joins’

```
SELECT ... FROM Employee A, Employee B
  WHERE A.Dept = B.Dept AND B.Name = ‘Andy’
```

A.Name	A.Dept	B.Name	B.Dept
~~John~~	~~Marketing~~	~~John~~	~~Marketing~~
~~Andy~~	~~Marketing~~	~~John~~	~~Marketing~~
~~Anne~~	~~Marketing~~	~~John~~	~~Marketing~~
~~Mary~~	~~Sales~~	~~Mary~~	~~Sales~~
~~Peter~~	~~Sales~~	~~Mary~~	~~Sales~~
~~Mary~~	~~Sales~~	~~Peter~~	~~Sales~~
~~Peter~~	~~Sales~~	~~Peter~~	~~Sales~~

Aliases and ‘Self-Joins’

```
SELECT ... FROM Employee A, Employee B
  WHERE A.Dept = B.Dept AND B.Name = ‘Andy’
```

A.Name	A.Dept	B.Name	B.Dept
John	Marketing	Andy	Marketing
Andy	Marketing	Andy	Marketing
Anne	Marketing	Andy	Marketing

Aliases and ‘Self-Joins’

```
SELECT A.Name FROM
  Employee A,
  Employee B
WHERE A.Dept = B.Dept
  AND B.Name = ‘Andy’
```

A.Name
John
Andy
Anne

- The result is the names of all employees who work in the same department as Andy.

Sub-queries

Subqueries

- A SELECT statement can be nested inside another query to form a subquery
- The results of the subquery are passed back to the containing query
- For example:
  Retrieve a list of names of people who are in Andy's department:

```
SELECT Name
  FROM Employee
 WHERE Dept =
  (SELECT Dept
   FROM Employee
   WHERE Name = 'Andy')
```

Subqueries : Example

```
SELECT Name
  FROM Employee
 WHERE Dept =
  (SELECT Dept
   FROM Employee
   WHERE
    Name = 'Andy')
```

- First the subquery is evaluated, returning 'Marketing'

- This value is passed to the main query:

```
SELECT Name
  FROM Employee
 WHERE Dept =
  'Marketing'
```

Subqueries and sets

Subqueries and sets

- Often a subquery will return a set of values rather than a single value
- We cannot directly compare a single value to a set. Doing so will result in an error
- We need operators which will act on a set
- Options for handling sets
  - IN – checks to see if a value is in a set
  - EXISTS – checks to see if a set is empty
  - ALL/ANY – checks to see if a relationship holds for every/one member of a set
  - NOT can be used with any of the above, to reverse the result

IN

- Using IN we can see if a given value is in a set of values
- NOT IN checks to see if a given value is not in the set
- The set can be given explicitly or can be produced in a subquery

```
SELECT <columns>
  FROM <tables>
 WHERE <value>
    IN <set>
```

```
SELECT <columns>
  FROM <tables>
 WHERE <value>
    NOT IN <set>
```

IN : Example 1

Employee

Name	Dept	Manager
John	Marketing	Chris
Mary	Sales	Chris
Chris	Marketing	Jane
Peter	Sales	Jane
Jane	Management	

```
SELECT *
 FROM Employee
 WHERE Dept IN
('Marketing','Sales')
```

Employee

Name	Dept	Manager
John	Marketing	Chris
Mary	Sales	Chris
Chris	Marketing	Jane
Peter	Sales	Jane

(NOT) IN : Example 2

Employee

Name	Dept	Manager
John	Marketing	Chris
Mary	Sales	Chris
Chris	Marketing	Jane
Peter	Sales	Jane
Jane	Management	

```
SELECT *
 FROM Employee
 WHERE Name NOT IN
  (SELECT Manager
    FROM Employee)
```

(NOT) IN : Example 2

- First the subquery

```
SELECT Manager
   FROM Employee
```

- is evaluated giving

Manager
Chris
Chris
Jane
Jane

- This gives

```
SELECT *
   FROM Employee
   WHERE Name NOT
      IN ('Chris',
          'Jane)
```

Name	Dept	Manager
John	Marketing	Chris
Mary	Sales	Chris
Peter	Sales	Jane

EXISTS

- Using EXISTS we see whether there is at least one element in a set

```
SELECT <columns>
  FROM <tables>
  WHERE EXISTS <set>
```

- NOT EXISTS is true if the set is empty

```
SELECT <columns>
  FROM <tables>
  WHERE NOT EXISTS
     <set>
```

- The set is **always** given by a subquery

EXISTS

Employee

Name	Dept	Manager
John	Marketing	Chris
Mary	Marketing	Chris
Chris	Marketing	Jane
Peter	Sales	Jane
Jane	Management	

```
SELECT *
 FROM Employee AS E1
 WHERE EXISTS (
  SELECT * FROM
    Employee AS E2
    WHERE E2.Manager
          = E1.Name )
```

Name	Dept	Manager
Chris	Marketing	Jane
Jane	Management	

Select employees who are the managers of somebody else

ANY and ALL

- ANY and ALL compare a single value to a set of values
- They are used with comparison operators like:
  = , >, <, <>, >=, <=
- **`val = ANY (set)`** is true if there is at least one member of the set equal to value
- **`val = ALL (set)`** is true if all members of the set are equal to the value

ALL : Example

Name	Salary
Mary	20,000
John	15,000
Jane	25,000
Paul	30,000

Name
Paul

- Find the name(s) of the employee(s) who earn the highest salary

```
SELECT Name
 FROM Employee
 WHERE
Salary >= ALL (
     SELECT Salary
     FROM Employee)
```

ANY : Example

Name	Salary
Mary	20,000
John	15,000
Jane	25,000
Paul	30,000

Name
Mary
Jane
Paul

- Find the name(s) of the employee(s) who earn more than someone else

```
SELECT Name
 FROM Employee
 WHERE
Salary > ANY (
     SELECT Salary
     FROM Employee )
```

Word Searches

Word Searches

- Word Searches : search for words/strings
- Commonly used for searching product catalogues etc.
- Need to search by keywords
- Might need to use partial keywords

- Example:
  Given a database of books, searching for "crypt" might return
  - "*Cryptonomicon*" by Neil Stephenson
  - "*Applied Cryptographer*" by Bruce Schneier

LIKE

- We can use the **LIKE** keyword to perform string comparisons in queries
- Like is not the same as '=' because it allows wildcard characters
- It is **not** normally case sensitive

```
SELECT * FROM books
 WHERE bookName LIKE '%crypt%';
```

LIKE

- The '%' character can represent any number of characters, including none

```
bookName LIKE 'crypt%'
```

- Will return "Cryptography Engineering" and "Cryptonomicon" but **not** "Applied Cryptography"

- The '_' character represents exactly one character

```
bookName LIKE 'cloud_'
```

- Will return "Clouds" but not "Cloud" or "Cloud Computing"

LIKE

- Sometimes you might need to search for a set of words
- To find entries with **all** words you can link conditions with **AND**
- To find entries with **any** words use **OR**

```
SELECT *
 FROM books
 WHERE
  bookName LIKE
  '%crypt%'
 OR
  bookName LIKE
  '%cloud%';
```

Sub-query Examples

Example 1

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of student IDs and Names for students studying G52ADS, without using a JOIN

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 1

- We need to access two tables
- We cannot use a JOIN
- We MUST be using a sub-query

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 1

- If we knew a list of sIDs it would be easy
- How can we SELECT a list of sIDs?
- "Find a list of student IDs and Names for students studying G52ADS"

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Example 1

- SELECT sID FROM Enrolment
  WHERE mCode = ‘G52ADS’

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
~~5~~	~~G51DBS~~
~~5~~	~~G51PRG~~
~~5~~	~~G51IAI~~
4	G52ADS
~~6~~	~~G51PRG~~
~~6~~	~~G51IAI~~

Example 1

- Output the matching Student names
- Question: How do we do this?
  (What is the SQL query?)

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Sub-Query

sID
1
2
4

Example 1

- SELECT sNAME FROM Student WHERE sID IN …

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Sub-Query

sID
1
2
4

Example 1

- SELECT sNAME FROM Student WHERE sID IN (SELECT sID FROM Enrolment WHERE mCode = ‘G52ADS’)

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Sub-Query

sID
1
2
4

Example 2

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Find a list of the names of any students who are enrolled on at least one module alongside 'Evans'

Enrolment

sID	mCode
1	G52ADS
2	G52ADS
5	G51DBS
5	G51PRG
5	G51IAI
4	G52ADS
6	G51PRG
6	G51IAI

Student NATURAL JOIN Enrolment

Student NATURAL JOIN Enrolment

sID	sName	sAddress	sYear	mCode
1	Smith	5 Arnold Close	2	G52ADS
2	Brooks	7 Holly Avenue	2	G52ADS
~~3~~	~~Anderson~~	~~15 Main Street~~	~~3~~	
4	Evans	Flat 1a, High Street	2	G52ADS
5	Harrison	Newark Hall	1	G51DBS
5	Harrison	Newark Hall	1	G51PRG
5	Harrison	Newark Hall	1	G51IAI
6	Jones	Southwell Hall	1	G51PRG
6	Jones	Southwell Hall	1	G51IAI

Question did not say we could not use a join

Example 2: 2 joins and a sub-query

- Find a list of the names of any students who are enrolled on at least one module alongside 'Evans'

```
SELECT DISTINCT S1.sName, E1.mCode
FROM
Student S1 NATURAL JOIN Enrolment E1
WHERE E1.mCode IN
( SELECT E2.mCode FROM
Enrolment E2 NATURAL JOIN Student S2
WHERE S2.sName = 'Evans' );
```

Could also join the two sub-queries on **`E1.mCode = E2.mCode`**

Example 2: Three joins

- Find a list of the names of any students who are enrolled on at least one module alongside ‘Evans’

```
SELECT DISTINCT S1.sName, E1.mCode
FROM
(Student S1 NATURAL JOIN Enrolment E1)
INNER JOIN
(Student S2 NATURAL JOIN Enrolment E2)
USING (mCode)
WHERE S2.sName = 'Evans';
```

Cannot NATURAL JOIN the parts here! (sName, mCode etc would match)

Example 2: Sub queries, get names

```
SELECT DISTINCT S1.sName FROM Student S1
  WHERE S1.sID IN
    ( SELECT E1.sID FROM Enrolment E1
      WHERE E1.mCode IN
        ( SELECT E2.mCode FROM Enrolment E2
          WHERE E2.sID IN
            ( SELECT S2.sID FROM Student S2
              WHERE S2.sName = 'Evans') ) );
```

As long as we only want the student name, not the module id (mCode not in table)

Next Lecture

- More SQL SELECT
  - ORDER BY
  - Aggregate functions
  - GROUP BY and HAVING
  - UNION
- Further reading
  - The Manga Guide to Databases, Chapter 4
  - Database Systems, Chapter 6