SQL SELECT

Database Systems
Michael Pound

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

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

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

Course

Code	Title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Introduction to AI

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 Last
  From Student
```

Last
Smith
Jones
Brown
Jones
Brown

```
SELECT DISTINCT Last
  FROM Student
```

Last
Smith
Jones
Brown

WHERE Clauses

- In most cases returning all the rows is not necessary
  - A WHERE clause restricts rows that are 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 Mark >= 60
```

ID	Code	Mark
S103	DBS	72
S104	PR1	68
S104	IAI	65
S107	PR1	76
S107	PR2	60

```
SELECT DISTINCT ID
  FROM Grade
  WHERE Mark >= 60
```

ID
S103
S104
S107

WHERE Examples

- Given the table:

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

- Write an SQL query to find a list of the ID numbers and Marks for students who have passed (scored 40% or more) in IAI

ID	Mark
S103	58
S104	65

Solution

```
SELECT ID, Mark FROM Grade
    WHERE (Code = 'IAI')
    AND (Mark >= 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	[illegible]
S106	Mari	[illegible]
S107	John	[illegible]

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

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
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65

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, it is almost always best to use a **WHERE** clause to find common values

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

SELECT from Multiple Tables

Student | Grade | Course

ID	First	Last	ID	Code	Mark	Code	Title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	IAI	58	IAI	Introduction to AI
S104	Mary	Jones	S104	PR1	68	PR1	Programming 1
S104	Mary	Jones	S104	IAI	65	IAI	Introduction to AI
S106	Mark	Jones	S106	PR2	43	PR2	Programming 2
S107	John	Brown	S107	PR1	76	PR1	Programming 1
S107	John	Brown	S107	PR2	60	PR2	Programming 2

Student.ID = Grade.ID Course.Code = Grade.Code

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

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
```

- Usually best 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**.

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
126	Jane	124	PRG
123	John	124	DBS
124	Mary	124	DBS

INNER JOIN

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

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

- Can also use a **USING** clause that will output rows with equal values in the specified columns

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

- **col1** and **col2** must appear in both **A** and **B**

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

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 row will be output representing the equal values from both Student.ID and Enrolment.ID

NATURAL JOIN

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

- Is the same as

```
SELECT A.Col1, A.Col2, ...
  , A.Coln, [and columns
  from B with names
  distinct from those in
  A]
FROM A, B
WHERE A.Col1 = B.Col1
  AND ...
  AND A.Coln = B.Coln
```

- A **NATURAL JOIN** is effectively a special case of an **INNER JOIN** where the **USING** clause has specified all identically named columns

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

JOINs vs WHERE Clauses

- JOINs are not absolutely necessary
  - You can obtain the same results by selecting from multiple tables and using appropriate WHERE clauses
  - Should you use JOINs?
- Yes
  - The often lead to concise and elegant queries
  - NATURAL JOINs are extremely common
- No
  - Support for JOINs can vary between DBMSs
  - 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

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