### **SQL** Lecture II

G51DBI – Databases and Interfaces
Yorgos Tzimiropoulos
yorgos.tzimiropoulos@nottingham.ac.uk

#### Overview of weeks 2-4

We will see how to translate English to Relational Algebra and SQL queries **and** vice versa

English: "Find all universities with > 20000 students" Relational Algebra:  $\pi_{uName}(\sigma_{enr>20000}(University))$ 

**SQL**: Select uName From University Where

University.enr>20000

Theory is easy and simple

**But** a sequence of simple operations is not always so obvious!

#### This Lecture

- ➤ SQL SELECT
  - WHERE Clauses
  - SELECT from multiple tables
- ➤ More SQL SELECT
  - Aliases
  - 'Self-Joins'
  - Subqueries
  - IN, EXISTS, ANY, ALL

### **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		
Course				

S107	John	Brown			
Course					
Code	Title				
DBS	Database	Systems			
PR1	Programming 1				
PR2	Programn	ning 2			
IAI	Introduct	ion to Al			
DBS PR1 PR2	Database Programr Programr	ning 1 ning 2			

Code	Mark
DBS	72
IAI	58
PR1	68
IAI	65
PR2	43
PR1	76
PR2	60
IAI	35
	DBS IAI PR1 IAI PR2 PR1 PR2

### **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: Smith Jones Brown Jones Brown

SELECT DISTINCT Last FROM Student;

Smith Jones Brown

#### **WHERE Clauses**

- 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
- In most cases returning 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 SELECT DISTINCT ID WHERE Mark >= 60; FROM Grade WHERE Mark >= 60;

### **WHERE Examples**

SELECT \* FROM Grade WHERE Mark >= 60;

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

SELECT DISTINCT ID FROM Grade WHERE Mark >= 60;

> 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 50% or more) in IAI

ID	Mark
S103	58
S104	65

### Solution

SELECT ID, Mark FROM Grade WHERE (Code = 'IAI') AND (Mark >= 50);

### **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 with Merit (Marks in [60, 69])

ID	Mark
S104	68
S104	65
S107	60

#### Solution

SELECT ID, Mark FROM Grade WHERE (Mark >=60 AND Mark < 70);

### Solution (only in MySQL!)

SELECT ID, Mark FROM Grade WHERE Mark BETWEEN 60 AND 69;

### **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 students IDs for both the IAI and PR2 modules

ID
S103
S104
S106
S107
S107

### Solution

SELECT ID FROM Grade WHERE (Code = 'IAI' OR Code = PR2';

### **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

First, Last, Mark

AND (Mark >= 40);

Student, Grade

# **SELECT from Multiple Tables**

Student First Last S103 John Smith Grade S104 Mark Code S105 S103 DBS S106 S103 IAI 58 S107 John S104 PR1 68 (Student.ID = Grade.ID) S104 IAI 65 PR2 S106 43 S107 PR1 76 S107 PR2 60 IAI 35 S107

## **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)

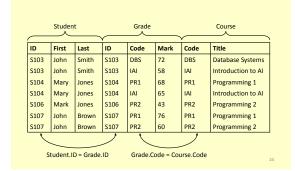
First	Last	Mark			
John	Smith	72			
John	Smith	58			
Mary	Jones	68			
Mary	Jones	65			
Mark	Jones	43			
John	Brown	76			
John	Brown	60			

### **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**



### **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

mCode	
G52ADS	
G52ADS	
G51DBS	
G51PRG	
G51IAI	
G52ADS	
G51PRG	
G51IAI	
	G52ADS G52ADS G51DBS G51PRG G51IAI G52ADS G51PRG

### **Examples**

- Write SQL statements to do the following:
  - 1. Produce a list of all student names and all their enrolments (module codes)
  - 2. Find a list of module titles being taken by the student named "Harrison"
  - 3. Find a list of module codes and titles for all modules currently being taken by first year students

#### **Solutions**

- 1. SELECT sName, mCode FROM Student, Enrolment WHERE Student.sID = Enrolment.sID;
- 2. SELECT mTitle FROM Module, Student, Enrolment WHERE (Module.mCode = Enrolment.mCode) AND (Student.sID = Enrolment.sID) AND Student.sName = "Harrison";
- 3. SELECT Module.mCode, mTitle FROM Enrolment, Module, Student WHERE (Module.mCode = Enrolment.mCode) AND (Student.sID = Enrolment.sID) AND sYear = 1;

## **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 gueries are easier to optimise

#### **SQL SELECT Overview**

#### SELECT

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

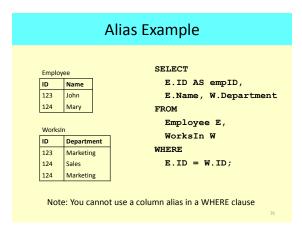
([] optional, | or)

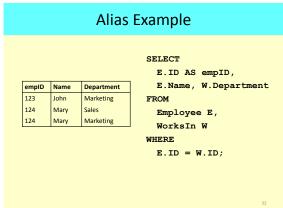
### Aliases

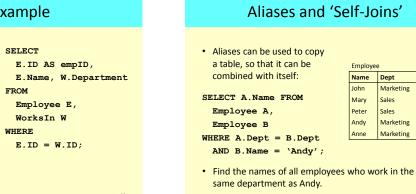
- Aliases rename columns
   Two forms: or tables
- · Can make names more meaningful
- · Can shorten names, making them easier to
- · Can resolve ambiguous

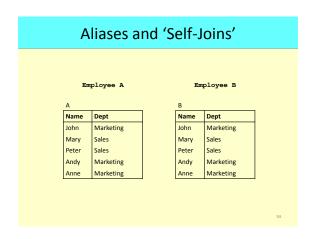
- Column alias SELECT column [AS] new-col-name
- · Table alias SELECT \* FROM table [AS] new-table-name

([] optional)













Dept

Sales

Sales

Marketing

Marketing

Marketing

#### 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';



• Names of all employees who work in the same department as Andy.

### **Subqueries**

- A SELECT statement can For example, retrieve a be nested inside another query to form a subquery
- The results of the subquery are passed back to the containing query
- 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**

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**

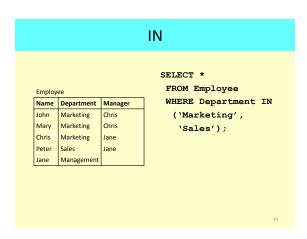
- Often a subquery will return a set of values rather than a single value
- We cannot directly compare a single value to a set. Doing so will result in an error
- · Options for handling sets
- IN checks to see if a value is in a set
- · EXISTS checks to see if a set is empty
- ALL/ANY checks to see if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

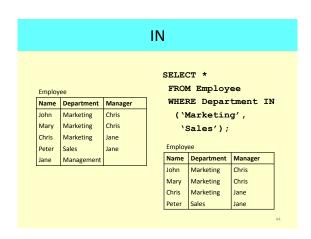
IN

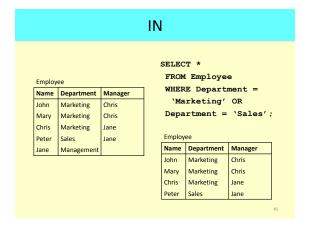
- Using IN we can see if a given value is in a set of values
- · NOT IN checks to see if a given value is not in the set
- The set can be given explicitly or can be produced in a subquery

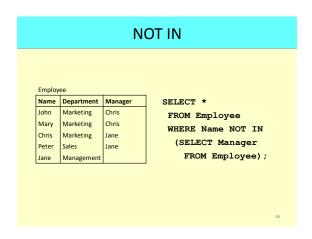
SELECT columns FROM tables WHERE value IN set;

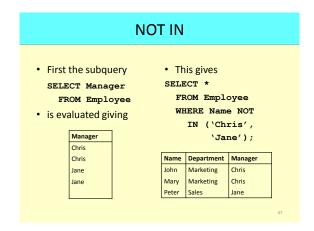
SELECT columns FROM tables WHERE value NOT IN set;

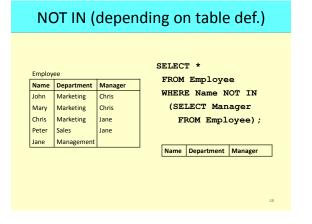












### Correct query

#### Employee

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

SELECT \* FROM Employee
WHERE Name NOT IN
(SELECT Manager FROM
Employee WHERE
Manager IS NOT NULL);

**EXISTS** 

 Using EXISTS we can see whether there is at least one element in a given set SELECT columns
FROM tables
WHERE EXISTS set;

 NOT EXISTS is true if the set is empty SELECT columns
FROM tables
WHERE NOT EXISTS

 The set is always given by a subquery

set;

#### **EXISTS**

SELECT \*

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

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

 Retrieve all the info for those employees who are also managers.

#### **EXISTS**

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

#### Employee E1

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

- 1 --

Employee E2					
Name	Dept	Manager			
John	Marketing	Chris			
Mary	Marketing	Chris			
Chris	Marketing	Jane			
Peter	Sales	Jane			
Jane	Management				

#### **EXISTS**

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

#### **EXISTS**

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

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

Name	Dept	Manager
Chris	Marketing	Jane
Chris	Marketing	Jane
Jane	Management	
Jane	Management	

#### **EXISTS**

#### Employee

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

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

Name Dept		Manager
Chris	Marketing	Jane
Jane	Management	

### **ANY and ALL**

- ANY and ALL compare a single value to a set of values
- They are used with comparison operators like = , >, <, <>, >=, <=

Name Salary

Mary

John

Jane

Paul

20,000

15,000

25,000

30,000

- 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

Name Salary Mary 20,000 John 15,000 Jane 25,000 Paul 30,000 • Find the name(s) of the employee(s) who earn the highest salary

### ALL

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

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

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

> Name Mary Jane

• 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 Search**

- Word Search
  - Commonly used for searching product catalogues etc.
  - Need to search by keywords
  - Might need to use partial keywords
- For 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
- The '\_' character represents exactly one character

bookName LIKE "crypt%"

bookName LIKE "cloud "

- Will return "Cryptography Engineering" and "Cryptonomicon" but not "Applied Cryptography"
- 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%";

## Example

Track							
cdID	Num	Track_title Time		alD			
1	1	Violent	239	1			
1	2	Every Girl	410	1			
1	3	Breather	217	1			
1	4	Part of Me	279	1			
2	1	Star	362	1			
2	2	Teaboy	417	2			

	CD							
	cdID	Т	itle	Price				
	1	Mix		9.99				
	2	Compilation		12.99				
	Artist							
	aID		Name					
	1		Stellar					
	2		Cloudboy					

Write a query to find any track title containing either the string 'boy' or 'girl'

### Example

SELECT Track\_title FROM Track
WHERE Track\_title LIKE "%boy%"
OR Track title LIKE "%girl%";

66

### Date, Time, Datetime

• 3 different types for a time column

туре	Description	Example
DATE	A Day, Month and Year	'1981-12-16' or '81-12-16'
TIME	Hours, Minutes and Seconds	'15:24:39'
DATETIME	Combination of above	'1981-12-16 15:24:39'

- Timestamp: as Datetime, usually used to display current date and time ('2014-11-04 15:30:43')
- Usual conditions may be used on WHERE clauses

```
• SELECT * FROM table-name

WHERE date-of-event < `2012-01-01';

Or WHERE date-of-event LIKE `2014-11-%';
```

## Take home messages

- 1. SELECT query to retrieve information
- 2. WHERE clause to specify condition
- 3. Cartesian product to combine tables
- 4. Different ways to write SELECT queries
  - a. Some more elegant than others

Thanks for your attention!

Any questions??