THE UNIVERSITY OF MELBOURNE

SQL
Part 2

Database Systems & Information Modelling INFO90002.

Week 5 - SQL

Dr Tanya Linden
Dr Renata Borovica-Gajic
David Eccles

This Lecture Objectives

Extending your knowledge

- DML
  - Comparison & Logic Operators
  - Set Operations
  - Subquery
  - Multiple record INSERTs
  - INSERT from a table, UPDATE, DELETE, REPLACE
  - Views
- DDL
  - ALTER and DROP, TRUNCATE, RENAME
  - CTAS

How to think about SQL

- Problem Solving

Things to Remember about SQL

entity save in different file

SQL keywords are case insensitive

- We try to CAPITALISE them to make them clear
- Improve readability of your statements

Table names are Operating System Sensitive

- If case sensitivity exists in the operating system, then the table names are case sensitive! (i.e. Linux, Unix)
  - Account != ACCOUNT

column

Field names are case insensitive

- ACCOUNTID == AccountID == AcCoUnTID

column name is ok

You can do maths in SQL…

- SELECT 1*1+1/1-1;

You can create your own columns that are not in the table

- SELECT '123459999' as MyID

assignment

Note On SELECT

The select statement's job is just to return rows of data, it doesn't care about the order of these rows unless you specify the ORDER BY clause

So what order do rows come out in if you don't specify the ORDER BY clause?

- Any order
- Possibly the order the records were created in
- It is undefined
  - Because SQL may optimise the query which may change the order of results…

So make sure you get into the habit of using the ORDER BY clause *if* you need a particular order

- If you don't need order, don't use it – it's going to slow down the execution

HAVING Clause: Revisited

The HAVING clause was added to SQL because the WHERE keyword **cannot be used** with **aggregate** functions.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Example:

List the number of customers of each country, but ONLY include countries with more than 5 customers

```
SELECT  CountryName, COUNT(CustomerID)
FROM Customers
GROUP BY  CountryName
HAVING COUNT(CustomerID) > 5;
```

Condition over the aggregate

provide alias

```
SELECT  CountryName,
        COUNT(CustomerID) AS CountOfCustomers
FROM Customers
GROUP BY  CountryName
HAVING  CountOfCustomers > 5;
```

use alias

Comparison and Logic Operators

Comparison:

Operator	Description
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<> *OR* !=	Not equal to (depends on DBMS as to which is used)

Logic:

- AND, OR, NOT

Example 1

```
SELECT *
FROM Staff
WHERE (Age>=18 AND Age<=65);
```

Example 2

```
SELECT *
FROM Staff
WHERE LastName='Nguyen' OR LastName='Smith'
```

Example 3

```
SELECT *
FROM Staff
WHERE (LastName='Nguyen' OR LastName='Smith') AND DeptNo=170
```

Some Useful String Functions

UPPER()

- Change to upper case

LOWER()

- Change to lower case

Characters outside the range **A-Z / a-z** are **not affected**

postcode of VIC

LEFT() specific how many number

- Take the left X characters from a string

Left (1) = 3

RIGHT()

- Take the X right characters from a string

The functions do NOT change data in the tables

Many more examples are in the labs!

Set Operations

the two table are not related, you cannot use inner or outer join you have to use union.

UNION

- combine the results of two queries (or tables) into a single result set
- The **number** and **data types of the columns** selected by each component query **must be the same**, but the column lengths can be different

INTERSECT doesn't work on our server

- Shows only rows that are common in the queries (or the tables)

[UNION/INTERSECT] ALL

- If you want to have duplicate rows in the result set you need to use the ALL keyword.. UNION ALL etc.

In MySQL only UNION and UNION ALL are supported

UNION ALL Example

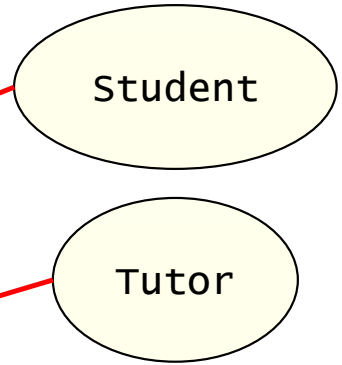
A **Union ALL** operator causes **ALL** rows to be added. Duplicates may occur.

```
SELECT name FROM Student;
Alice Arron
Bella Barton
Connie Chang
Sean Shannon
```

```
SELECT name FROM Tutor;
Jacob Jones
Connie Chang
Luke Laurence
```

```
SELECT name FROM Student
UNION ALL
SELECT name FROM Tutor;

Alice Arron
Bella Barton
Connie Chang
Sean Shannon
Jacob Jones
Connie Chang
Luke Laurence
```



Names that are in both tables are ***duplicated***

UNION Example

```sql
SELECT  Employee.Name, EmployeeType
        FROM Employee INNER JOIN Hourly
        ON  Employee.ID = Hourly.ID
UNION
SELECT  Employee.Name, EmployeeType
        FROM Employee INNER JOIN Salaried
        ON  Employee.ID = Salaried.ID;
```

Output | Snippets | Query 1 Result | Lecture7.sql Result

Fetched 4 records. Durati

Name	EmployeeType
Alice	H
Alan	H
Sean	S
Linda	S

SELECT with Literals

When dealing with non matching columns between the tables, you may need to utilise **literal** values

A **literal** value is a value 'hardcoded' into the query; the value is not generated from the table.

```
SELECT  name,  gender
FROM    customer
WHERE   gender IS NOT NULL;
```

Liz	F
John	M
Ella	F
Rose	M

instead of gender

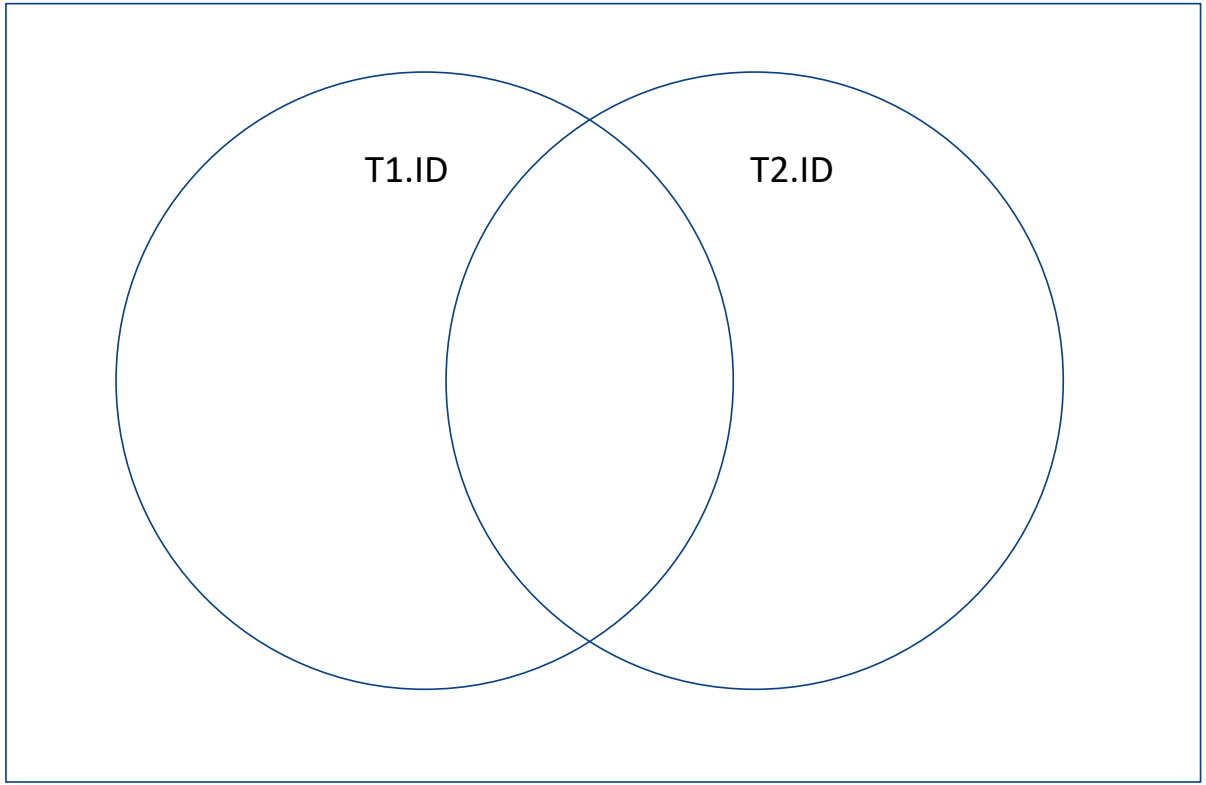
```
SELECT  name, 'Unknown'
FROM    customer
WHERE   gender IS NULL
```

Tom	Unknown
Brian	Unknown
Mary	Unknown

```
SELECT  name, 'Unknown' FROM customer WHERE  gender IS NULL
UNION
SELECT  name, gender FROM customer WHERE  gender IS NOT NULL
```

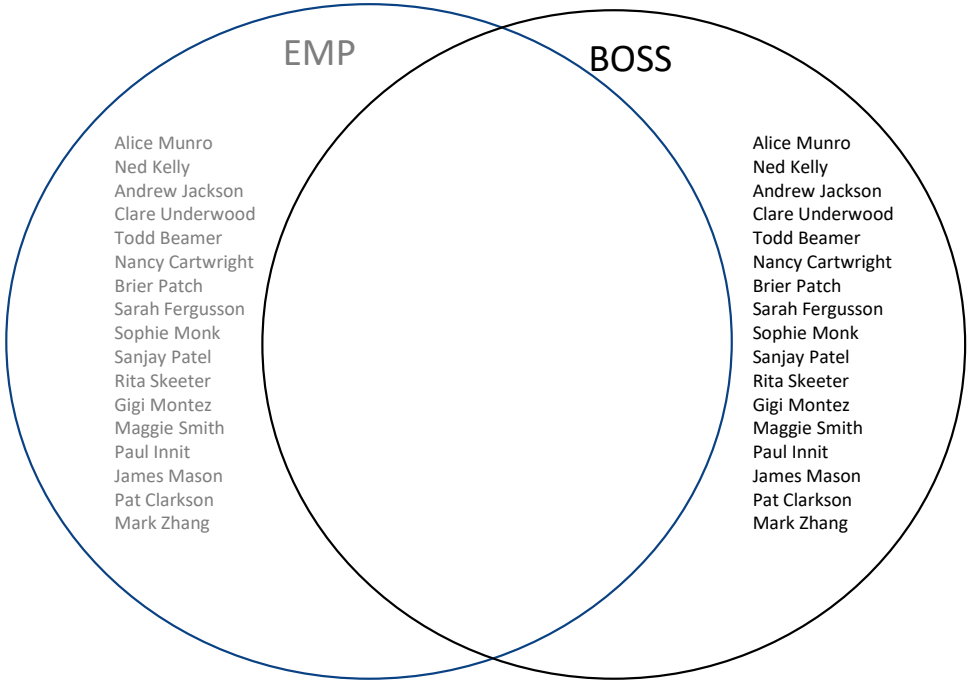
Liz	F
Tom	Unknown
John	M
…	

JOINS depicted as Venn Diagrams



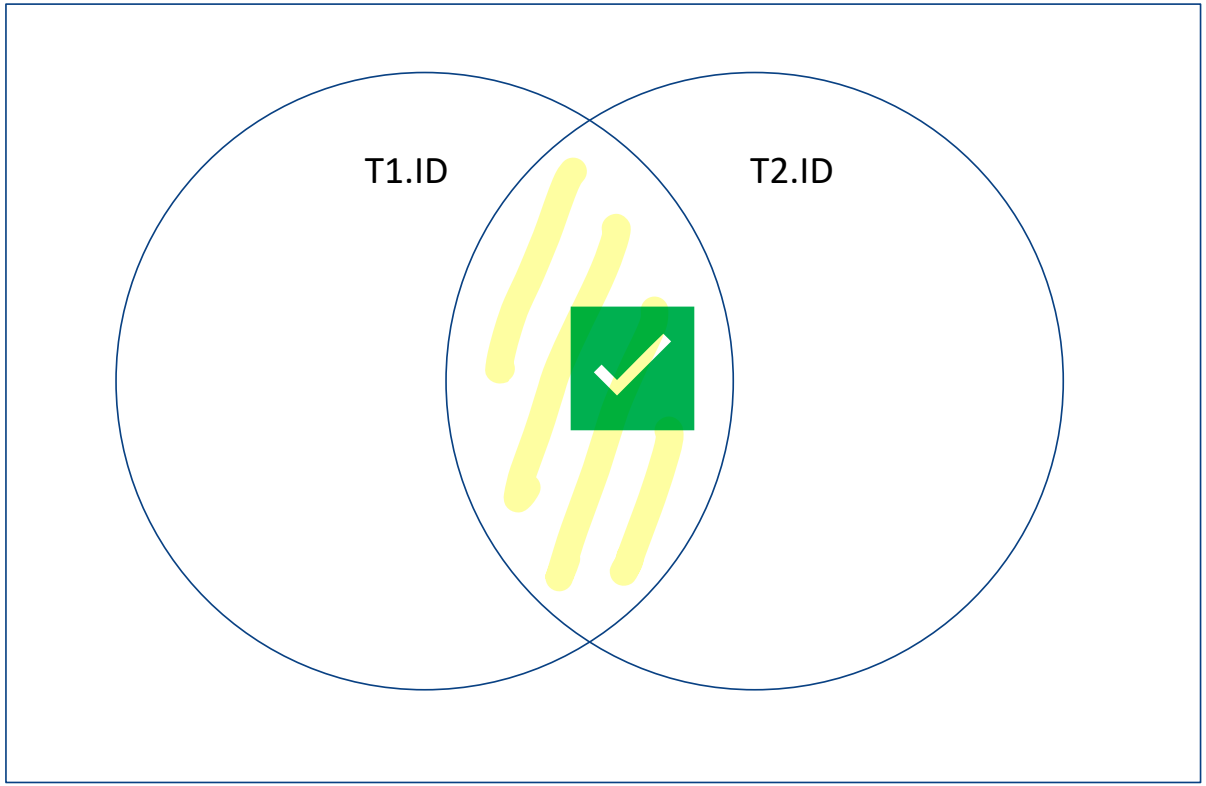
INNER Join – Labs demo

Unary join of the Employee table aliased as EMP, BOSS



JOINS depicted as Venn Diagrams

T1 INNER JOIN T2 ON T1.ID = T2.ID
T1 NATURAL JOIN T2



INNER Join – Labs demo

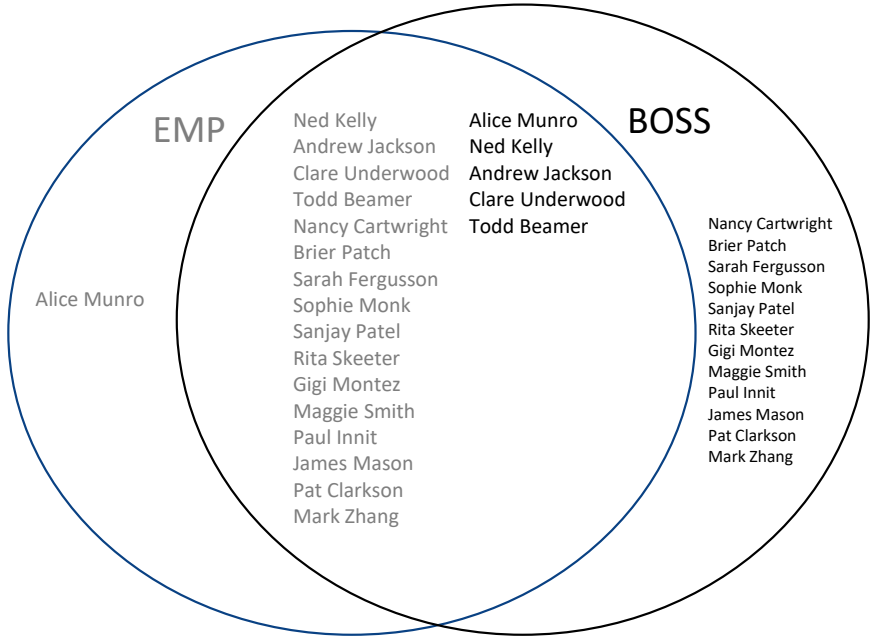
SELECT emp.firstname **AS** efirst, emp.lastname **AS** elast, boss.firstname **AS** bfirst, boss.lastname **AS** blast

FROM employee emp

INNER JOIN employee boss

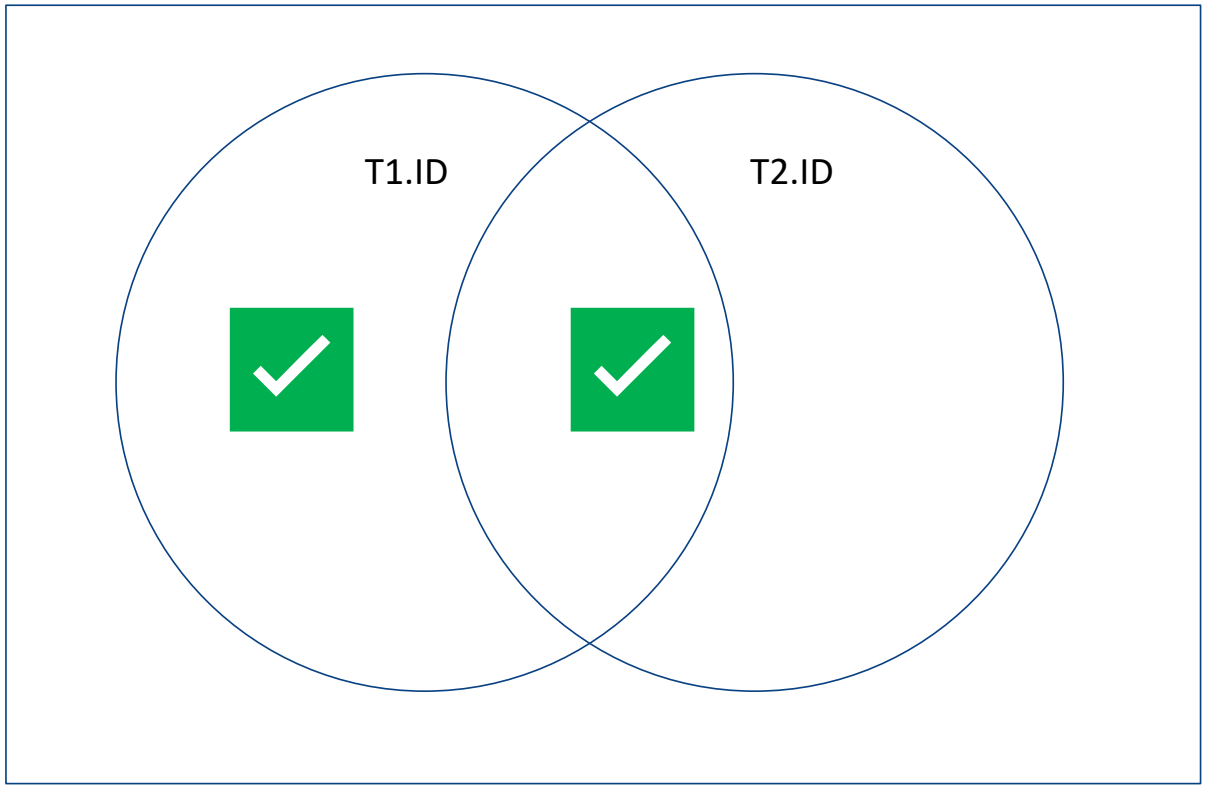
ON emp.bossid = boss.employeeid;

efirst	elast	bfirst	blast
Ned	Kelly	Alice	Munro
Andrew	Jackson	Ned	Kelly
Clare	Underw…	Ned	Kelly
Todd	Beamer	Alice	Munro
Nancy	Cartwright	Todd	Beamer
Brier	Patch	Alice	Munro
Sarah	Fergusson	Brier	Patch
Sophie	Monk	Alice	Munro
Sanjay	Patel	Andrew	Jackson
Rita	Skeeter	Clare	Underw…
Gigi	Montez	Clare	Underw…
Maggie	Smith	Clare	Underw…
Paul	Innit	Andrew	Jackson
James	Mason	Andrew	Jackson
Pat	Clarkson	Andrew	Jackson
Mark	Zhang	Andrew	Jackson



JOINS depicted as Venn Diagrams

T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID



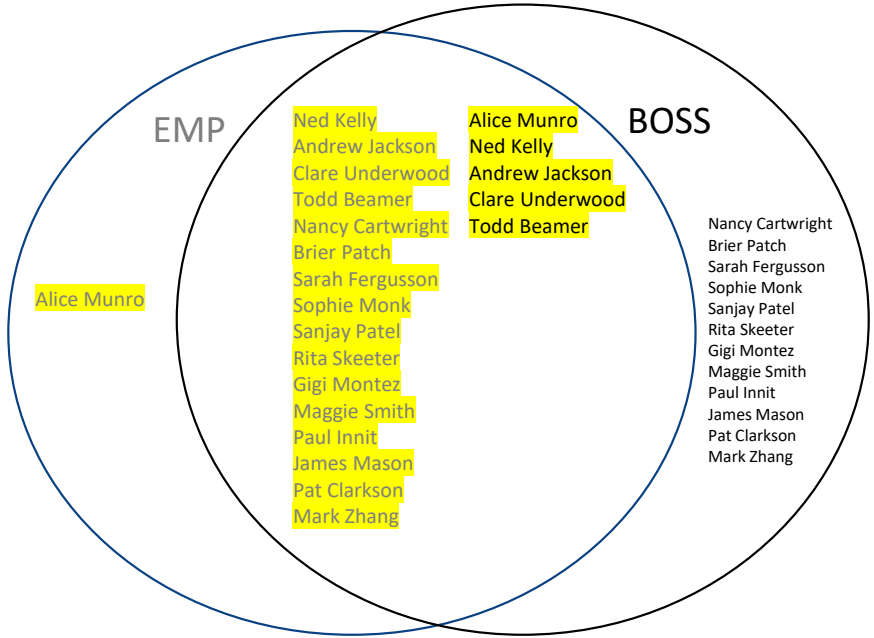
LEFT OUTER JOIN – Labs demo

SELECT emp.firstname **AS** efirst, emp.lastname **AS** elast, boss.firstname **AS** bfirst, boss.lastname **AS** blast

FROM employee emp **LEFT OUTER JOIN** employee boss

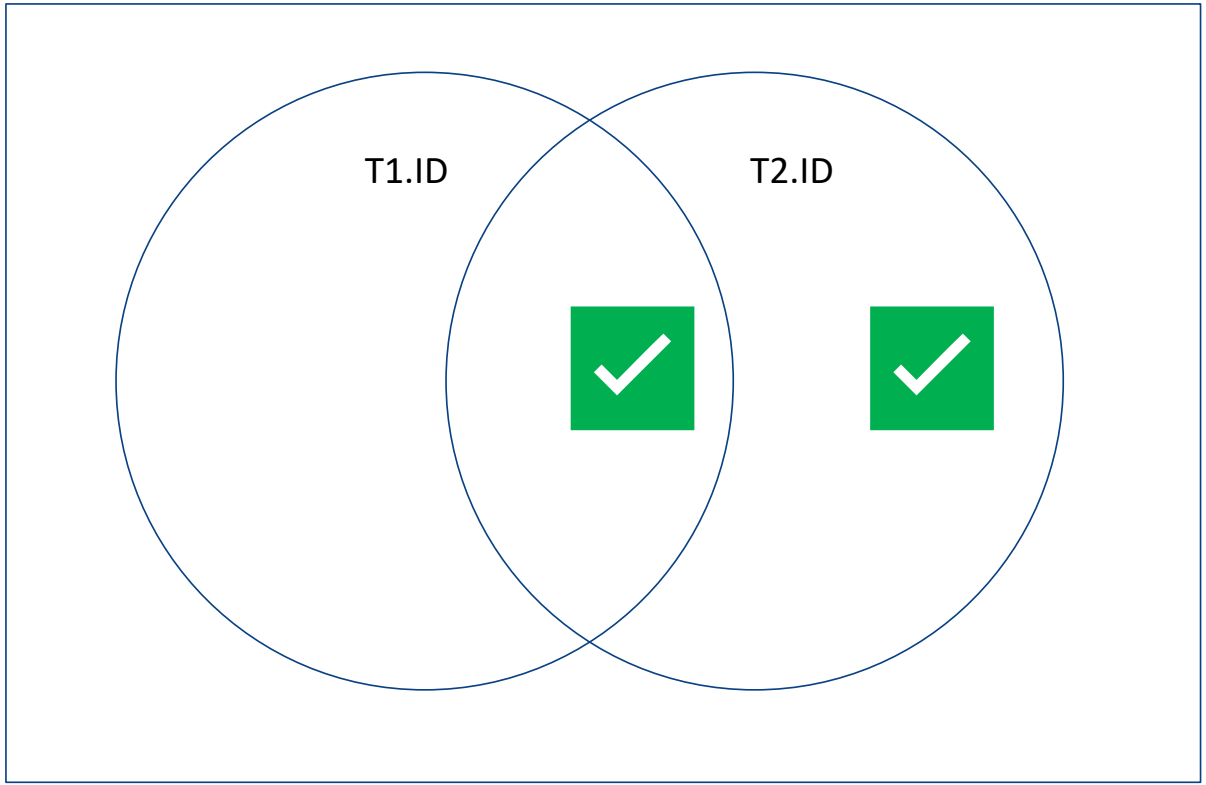
ON emp.bossid = boss.employeeid;

efirst	elast	bfirst	blast
Alice	Munro	NULL	NULL
Ned	Kelly	Alice	Munro
Andrew	Jackson	Ned	Kelly
Clare	Underwood	Ned	Kelly
Todd	Beamer	Alice	Munro
Nancy	Cartwright	Todd	Beamer
Brier	Patch	Alice	Munro
Sarah	Fergusson	Brier	Patch
Sophie	Monk	Alice	Munro
Sanjay	Patel	Andrew	Jackson
Rita	Skeeter	Clare	Underwood
Gigi	Montez	Clare	Underwood
Maggie	Smith	Clare	Underwood
Paul	Innit	Andrew	Jackson
James	Mason	Andrew	Jackson
Pat	Clarkson	Andrew	Jackson
Mark	Zhang	Andrew	Jackson



JOINS depicted as Venn Diagrams

T1 RIGHT OUTER JOIN T2 ON T1.ID = T2.ID

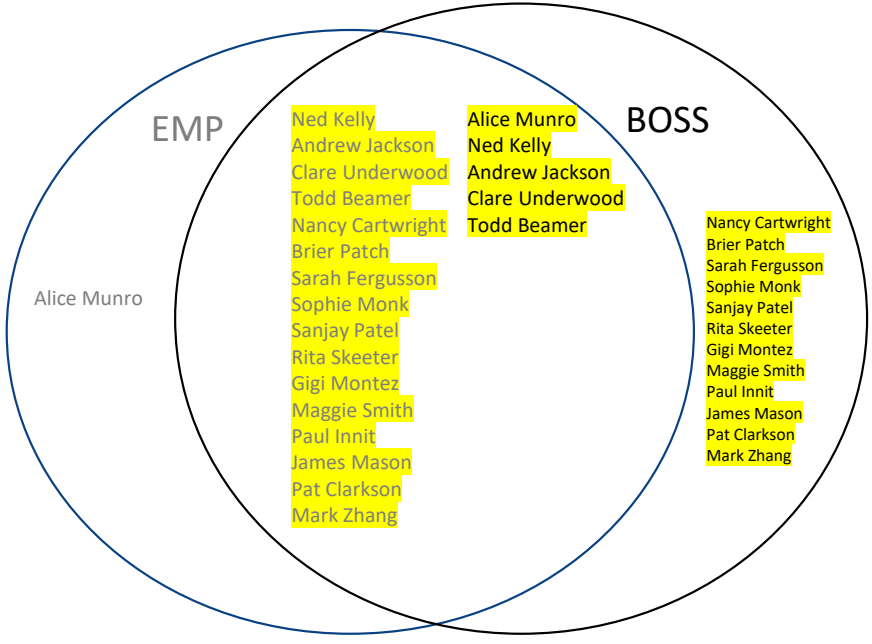


RIGHT OUTER JOIN – Labs demo

SELECT emp.firstname **AS** efirst, emp.lastname **AS** elast, boss.firstname **AS** bfirst, boss.lastname **AS** blast
FROM employee emp **RIGHT OUTER JOIN** employee boss
ON emp.bossid = boss.employeeid;

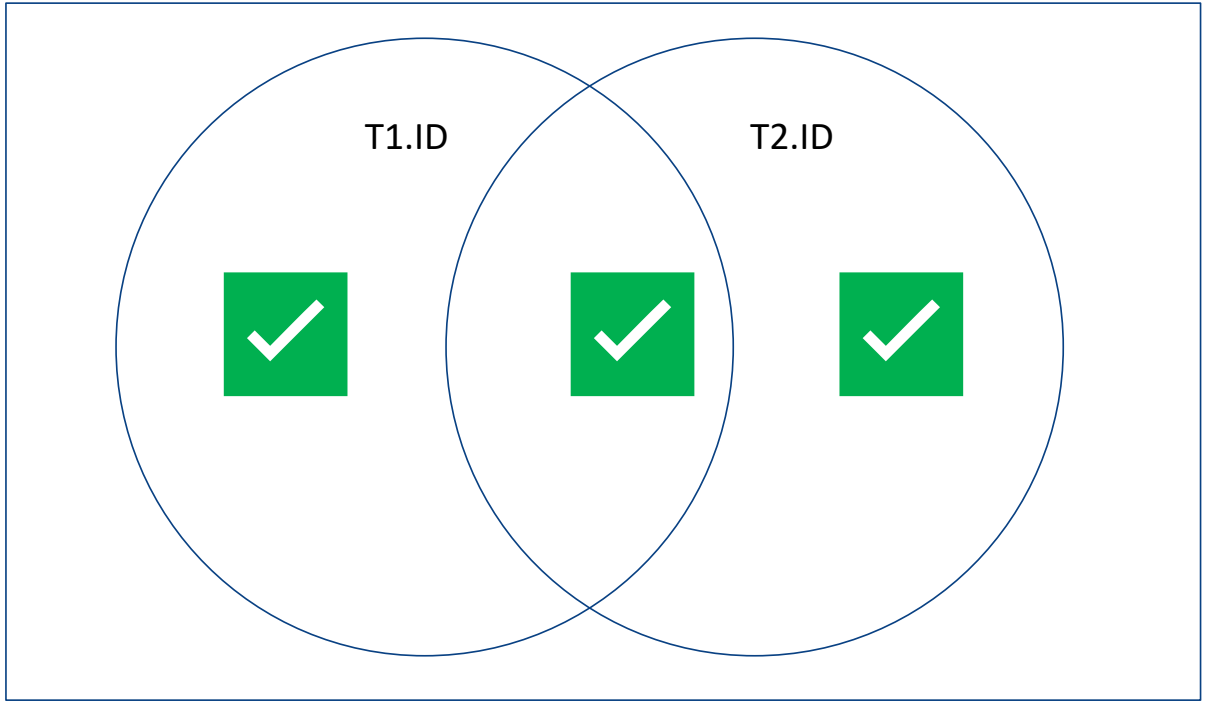
- Asks list every employee of every manager
- Hence Sarah, Sophie, Rita, Mark etc have no names against them

efirst	elast	bfirst	blast
Ned	Kelly	Alice	Munro
Todd	Beamer	Alice	Munro
Brier	Patch	Alice	Munro
Sophie	Monk	Alice	Munro
Andrew	Jackson	Ned	Kelly
Clare	Underwood	Ned	Kelly
Sanjay	Patel	Andrew	Jackson
Paul	Innit	Andrew	Jackson
James	Mason	Andrew	Jackson
Pat	Clarkson	Andrew	Jackson
Mark	Zhang	Andrew	Jackson
Rita	Skeeter	Clare	Underwood
Gigi	Montez	Clare	Underwood
Maggie	Smith	Clare	Underwood
Nancy	Cartwright	Todd	Beamer
NULL	NULL	Nancy	Cartwright
Sarah	Fergusson	Brier	Patch
NULL	NULL	Sarah	Fergusson
NULL	NULL	Sophie	Monk
NULL	NULL	Sanjay	Patel
NULL	NULL	Rita	Skeeter
NULL	NULL	Gigi	Montez
NULL	NULL	Maggie	Smith
NULL	NULL	Paul	Innit
NULL	NULL	James	Mason
NULL	NULL	Pat	Clarkson
NULL	NULL	Mark	Zhang



JOINS depicted as Venn Diagrams

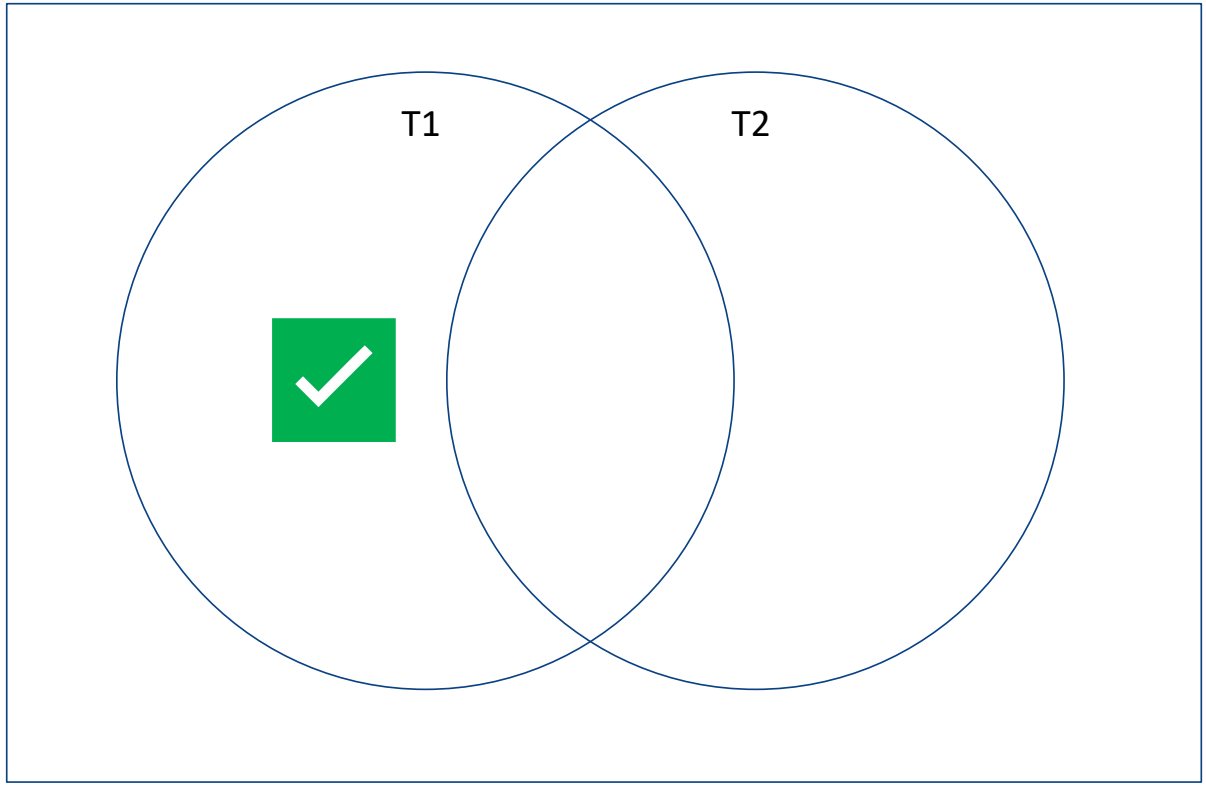
T1 FULL OUTER JOIN T2 ON T1.ID = T2.ID



MySQL DBMS server does not support full outer joins

JOINS depicted as Venn Diagrams

T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID
WHERE T2.ID IS NULL



INNER JOIN – Labs demo

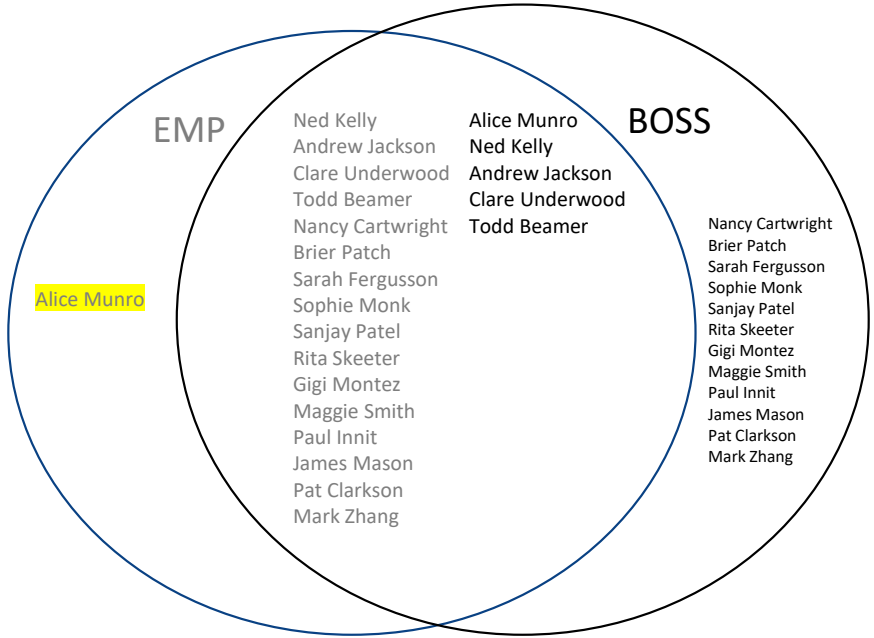
SELECT emp.firstname **AS** efirst, emp.lastname **AS** elast, boss.firstname **AS** bfirst, boss.lastname **AS** blast

FROM employee emp

LEFT OUTER JOIN employee boss

ON emp.bossid = boss.employeeid

WHERE boss.employeeid **IS NULL;**



Effectiveness and readability

Query must run effectively

- INNER JOIN is faster than NATURAL JOIN
- UNION is faster than JOINs
- JOINs are faster than subqueries

avoid use subqueries. because uneffective

```
SELECT boss.firstname AS bfirst, boss.lastname AS blast
FROM employee emp
RIGHT OUTER JOIN employee boss
ON emp.bossid = boss.employeeid
WHERE emp.firstname IS NOT NULL
GROUP BY bfirst, blast
ORDER BY bfirst, blast
```

Versus

```
SELECT FirstName, LastName FROM
employee
WHERE employeeID IN
  (SELECT BossID
   FROM employee)
ORDER BY FirstName, LastName
```

subqueries.

this one is slower to excecute

Effectiveness and readability (cont)

- The EXISTS clause is much faster than IN when the subquery results in a very large set. Conversely, the IN clause is faster than EXISTS when the subquery results is a very small set of rows.

  they will skip null

  Note, the IN clause **can't** compare values with NULLs,
  the EXISTS clause **can** compare values with NULLs.

- Calling functions causes processing overhead so create an alias for the resulting column and use it

```
SELECT col1, COUNT(col3) AS colName
FROM table
GROUP BY col1
HAVING colName>10
```

More on INSERT

Inserting records from a table:

```
INSERT INTO NewEmployee
    SELECT * FROM Employee;
```

you have newemployee create before

- Note: table must already exist

Multiple record inserts:

All columns must be inserted

```
INSERT INTO Employee VALUES
    (DEFAULT, "A", "A's Addr", "2012-02-02", NULL, "S"),
    (DEFAULT, "B", "B's Addr", "2012-02-02", NULL, "S"),
    (DEFAULT, "C", "C's Addr", "2012-02-02", NULL, "S");
```

Specific columns will be inserted

```
INSERT INTO Employee
    (Name, Address, DateHired, EmployeeType)
    VALUES
        ("D", "D's Addr", "2012-02-02", "C"),
        ("E", "E's Addr", "2012-02-02", "C"),
        ("F", "F's Addr", "2012-02-02", "C");
```

The UPDATE Statement

Changes *existing* data in tables

- Order of statements is important
- Specifying a WHERE clause is important
  - Unless you want it to operate on the whole table

```
UPDATE Hourly
    SET HourlyRate = HourlyRate * 1.10;
```

Example: *Increase all salaries greater than $100000 by 10% and all other salaries by 5%*

```
UPDATE Salaried
    SET AnnualSalary = AnnualSalary * 1.05
    WHERE AnnualSalary <= 100000;
UPDATE Salaried
    SET AnnualSalary = AnnualSalary * 1.10
    WHERE AnnualSalary > 100000;
```

Any problems with this?

★ The people with salary 95000 x1.05 then immediate they will time 1.10.

⇒ they time twice.
do twice.

The UPDATE Statement: CASE

A better solution in this case is to use the **CASE** command

```
UPDATE Salaried
    SET AnnualSalary =
        CASE
            WHEN AnnualSalary <= 100000
            THEN AnnualSalary * 1.05
            ELSE AnnualSalary * 1.10
        END;
```

If salary is lower than 100000 increase it by 5%, otherwise increase it by 10%

DELETE, REPLACE

REPLACE

- REPLACE works identically as INSERT
  - Except if an old row in a table has a key value the same as the new row, then it is overwritten…

DELETE

- The DANGEROUS command – deletes *ALL* records

```
DELETE FROM Employee;
```

  - The better (safer) version (unless you are really, really sure)

```
DELETE FROM Employee
    WHERE Name = "Grace";
```

- Be aware of the foreign key constraints
  - ON DELETE CASCADE or ON DELETE RESTRICT (lab practice)

THE UNIVERSITY OF MELBOURNE

ALTER (ADD/DROP), RENAME, TRUNCATE, CTAS

SQL DDL

More DDL Commands

ALTER

- Allows us to add or remove attributes (columns) from a relation (table)

ALTER TABLE TableName ADD AttributeName AttributeType

ALTER TABLE TableName DROP AttributeName

RENAME

- Allows the renaming of tables (relations)

RENAME TABLE CurrentTableName TO NewTableName

Because of FK issues.

More DDL Commands

TRUNCATE

- Same as DELETE * FROM table;
- Faster but cannot ROLL BACK a TRUNCATE command
  - Have to get data back from backup…

DROP

- Potentially DANGEROUS
  - Kills a relation – removes the data, removes the relation
  - There is NO UNDO COMMAND! (have to restore from backup)

DROP TABLE TableName

CTAS (CREATE TABLE as SELECT)

```
CREATE TABLE New_BankHQ
AS
SELECT *
FROM BankHQ
```

To create table structure with no rows

```
CREATE TABLE New_BankHQ
AS
SELECT *
FROM BankHQ
WHERE 1=0;
```

no rows will not equal in this

What’s examinable

- **SELECT**
- **DML (INSERT, UPDATE, DELETE, REPLACE)**
- **DDL (CREATE, ALTER, DROP)**

THE UNIVERSITY OF MELBOURNE

Thank you