INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 9
SQL Summary

Coverage

- A Summary of SQL
- Extending your knowledge
  - DML
    - Comparison & Logic Operators
    - Set Operations
    - Multiple record INSERTs
    - INSERT from a table; UPDATE, DELETE, REPLACE
    - Views
  - DDL
    - ALTER and DROP, TRUNCATE, RENAME
  - DCL
    - GRANT and REVOKE
  - Other commands
- How to think about SQL
  - Problem Solving & Aggregate Functions

SQL Language: Review

- Provides the following capabilities (just what the DBMS needs to run)
  - Data Definition Language (DDL)
    - To define and set up the database
    - CREATE, ALTER, DROP
      - Also TRUNCATE, RENAME
  - Data Manipulation Language (DML)
    - To maintain and use the database
    - SELECT, INSERT, DELETE, UPDATE
      - MySQL also provides others…. Eg REPLACE
  - Data Control Language (DCL)
    - To control access to the database
      - GRANT, REVOKE
  - Other Commands
    - Administer the database
    - Transaction Control

Things to Remember about SQL

- SQL keywords are case insensitive.
  - We try to CAPITALISE them to make them clear
- Table names are Operating System Sensitive
  - If case sensitivity exists in the operating system, then the table names are case sensitive! (ie Mac, Linux)
    - Account <> ACCOUNT
- Field names are case insensitive
  - ACCOUNTID == AccountID == AcCoUnTID
- You can do maths in SQL…
  - SELECT 1*1+1/1-1;

A 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 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

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
  - SQL supports
    - AND, NOT, OR
      - SELECT * FROM Furniture WHERE ((Type="Chair" AND Colour = "Black") OR (Type = "Lamp" AND Colour = "Black"))

Set Operations

- We can combine results from queries that return the same number of columns
  - although it only makes sense if they are the same columns
- UNION
  - Show all rows returned from the queries
- INTERSECT
  - Show only rows that are common in the queries
- EXCEPT
  - Show only rows that are different in the queries
- [UNION/INTERSECT/EXCEPT] ALL
  - If you want duplicate rows shown in the results you need to use the ALL keyword.. UNION ALL etc.

- In MySQL only UNION and UNION ALL are supported

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

UNION Example (2)

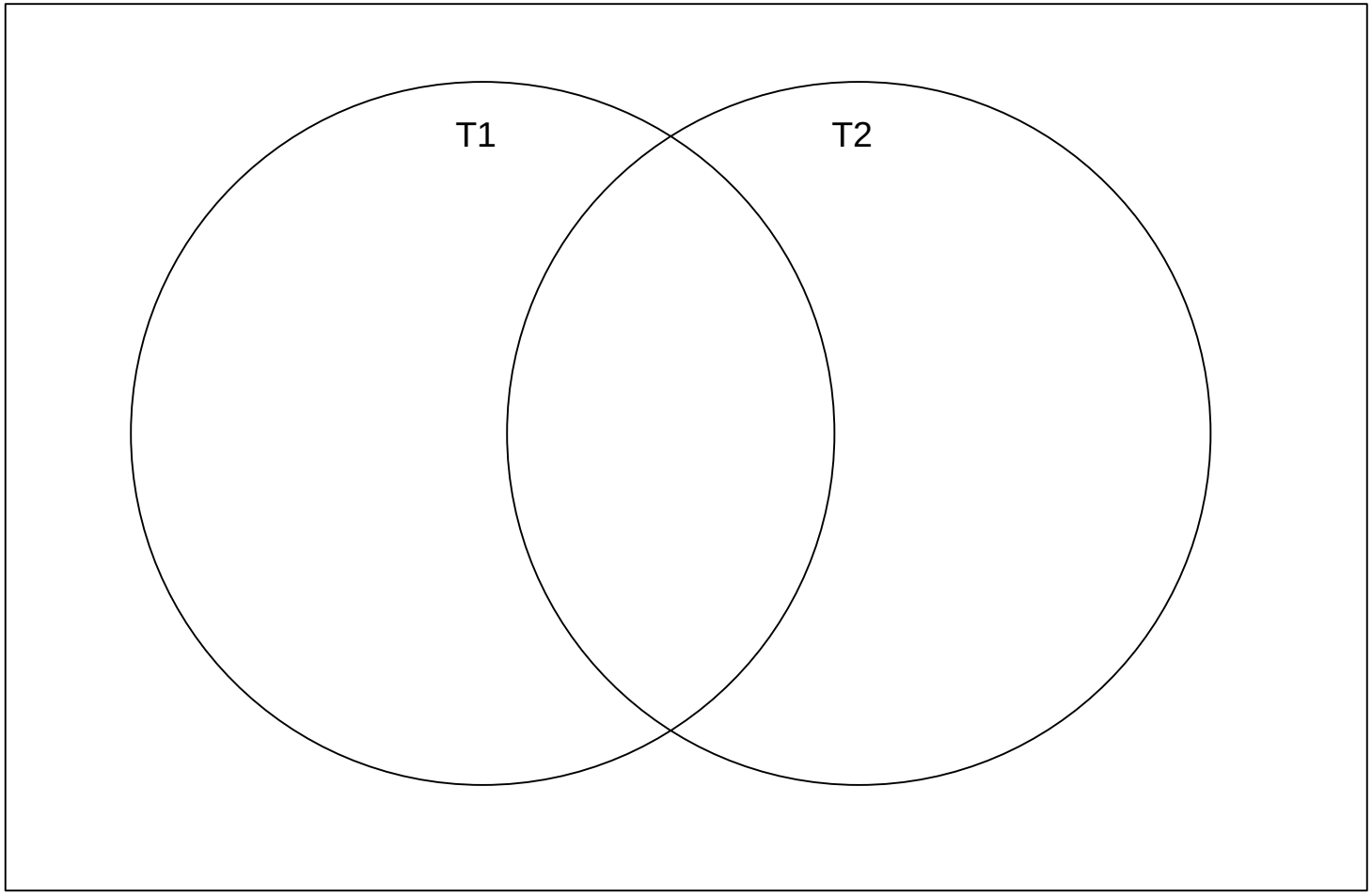
```sql
SELECT ItemID, ItemName
    FROM Item
    WHERE ItemID < 5
UNION
SELECT ItemID, ItemColour
    FROM Item
    WHERE ItemID < 5;
```

ItemID	ItemName
1	Boots - snakeproof
2	Camel saddle
3	Compass
4	Elephant polo stick
1	Green
2	Brown
3	-
4	Bamboo
NULL	NULL

Some Other Useful Functions

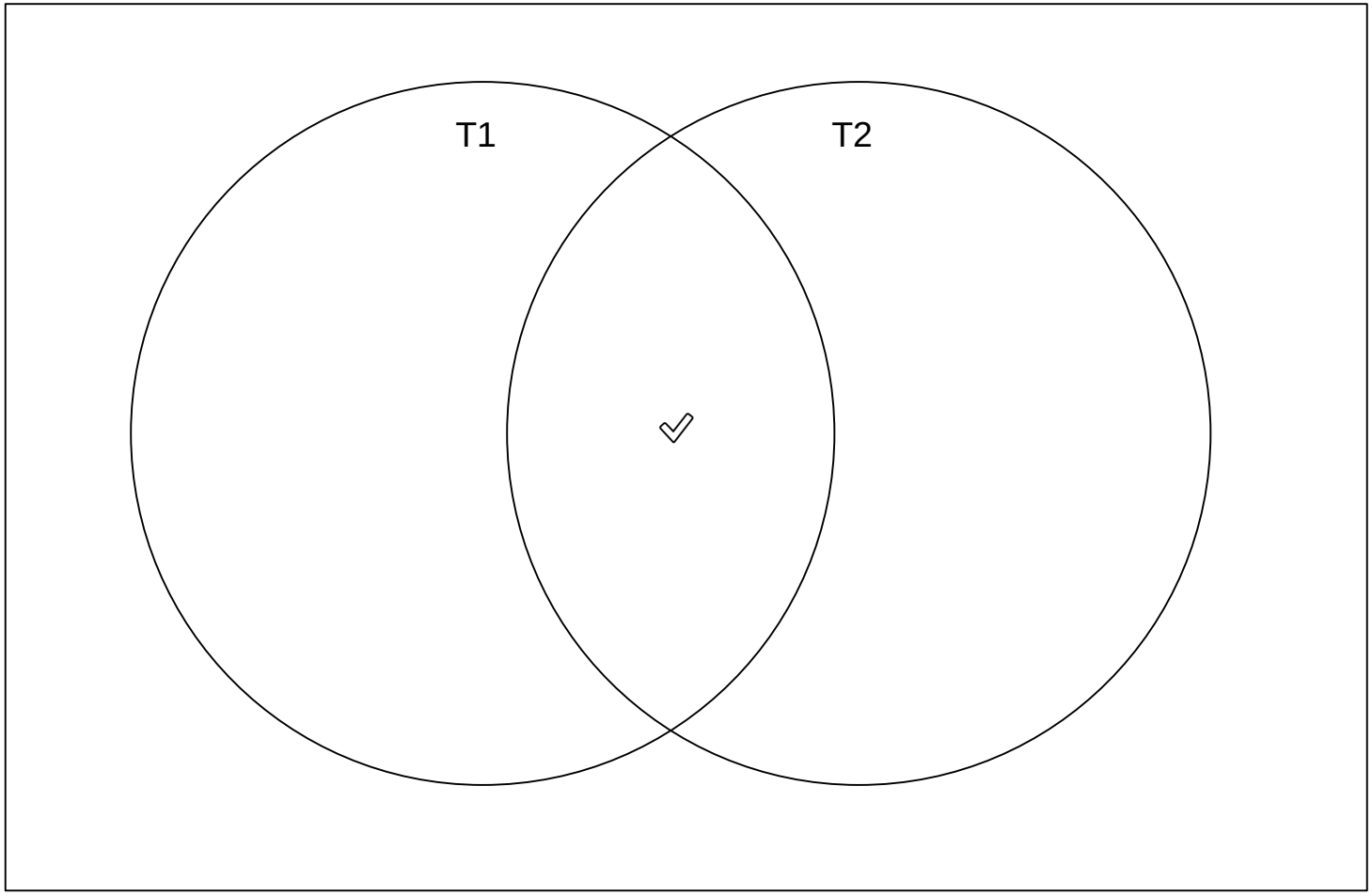
- FORMAT()
  - Allows us to format the output a little better
- UPPER()
  - Change to upper case
- LOWER()
  - Change to lower case
- LEFT()
  - Take the left X characters from a string
- RIGHT()
  - Take the X right characters from a string

JOINS depicted as Venn Diagrams



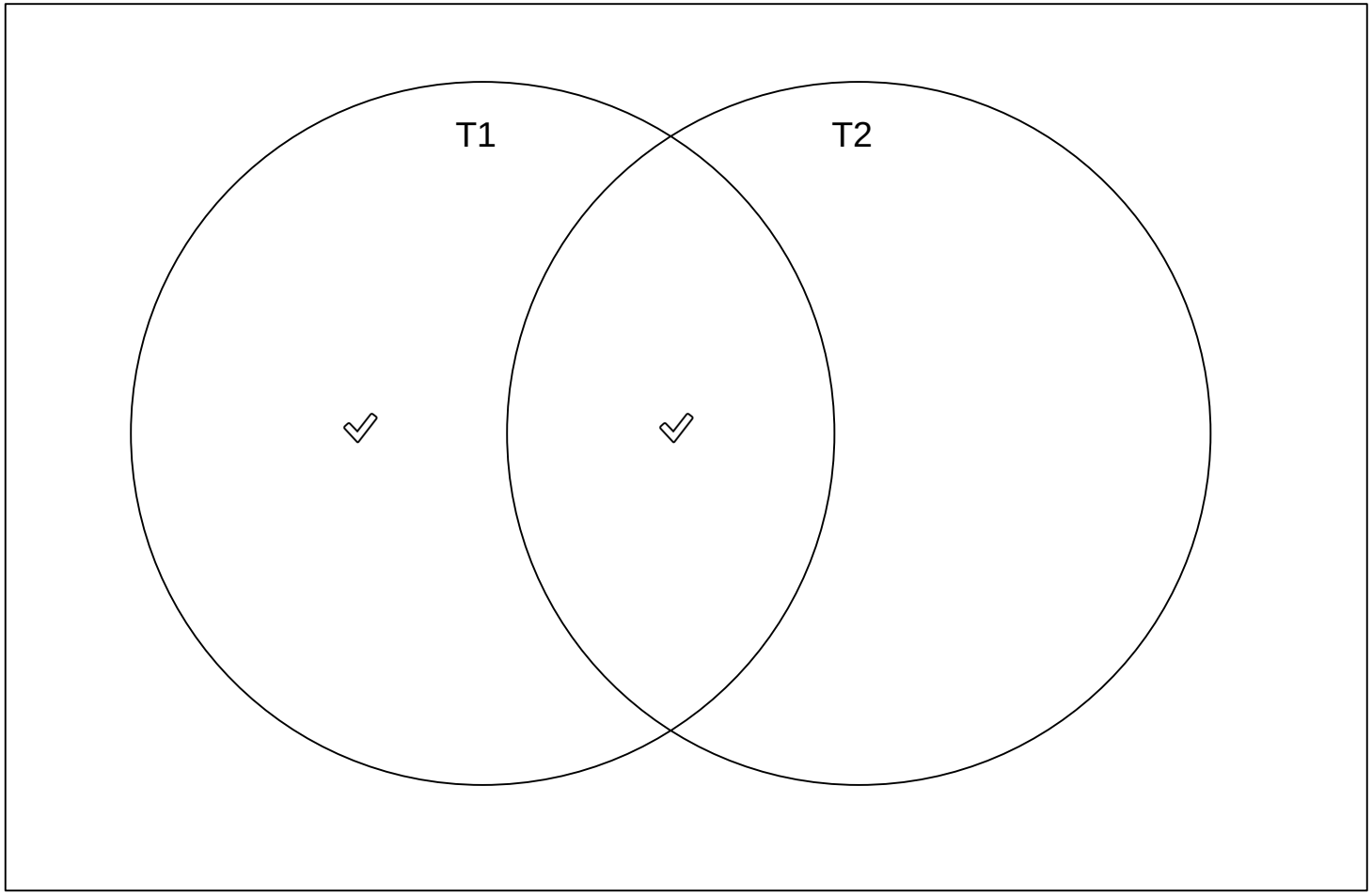
JOINS depicted as Venn Diagrams

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



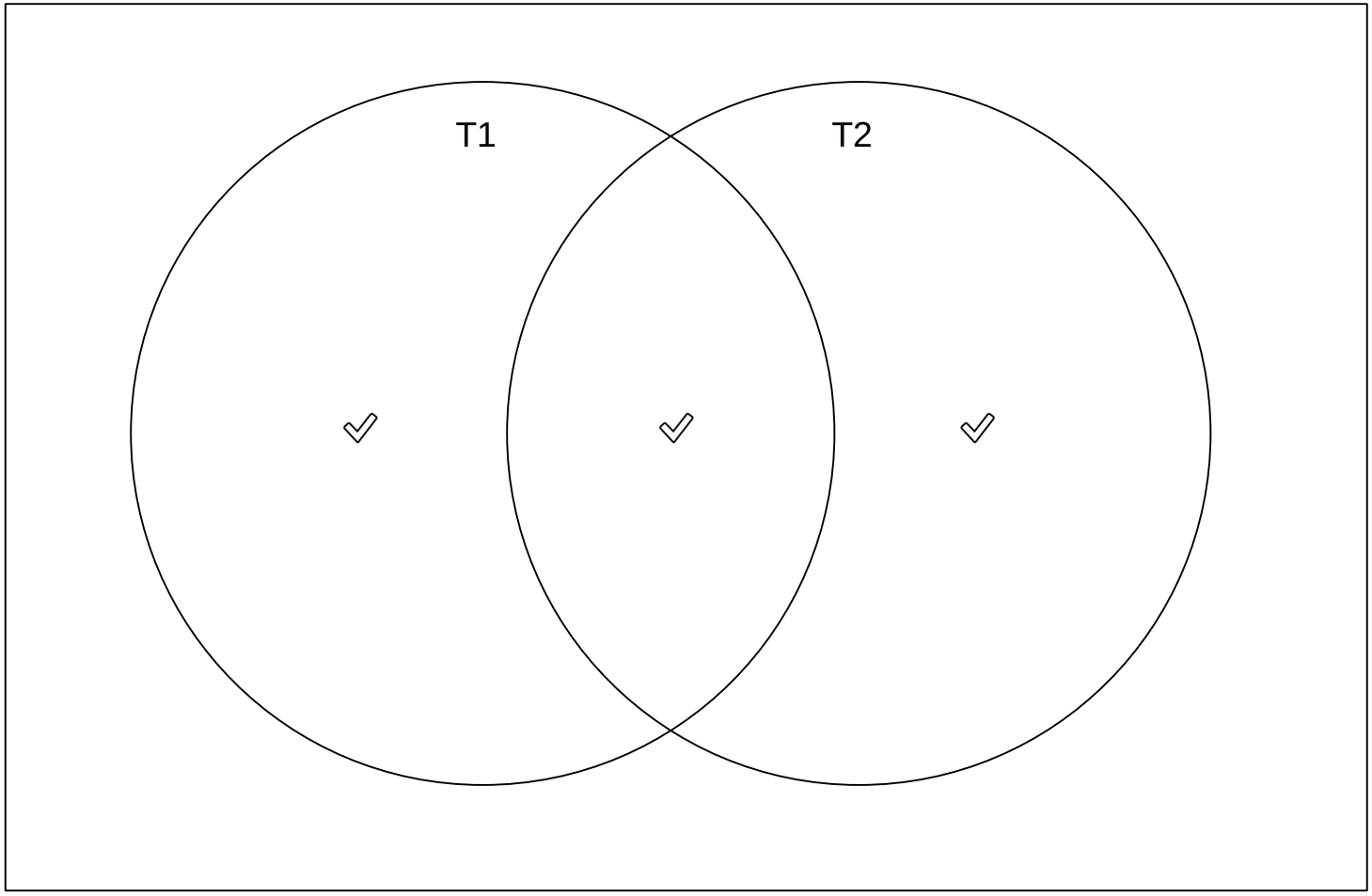
JOINS depicted as Venn Diagrams

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



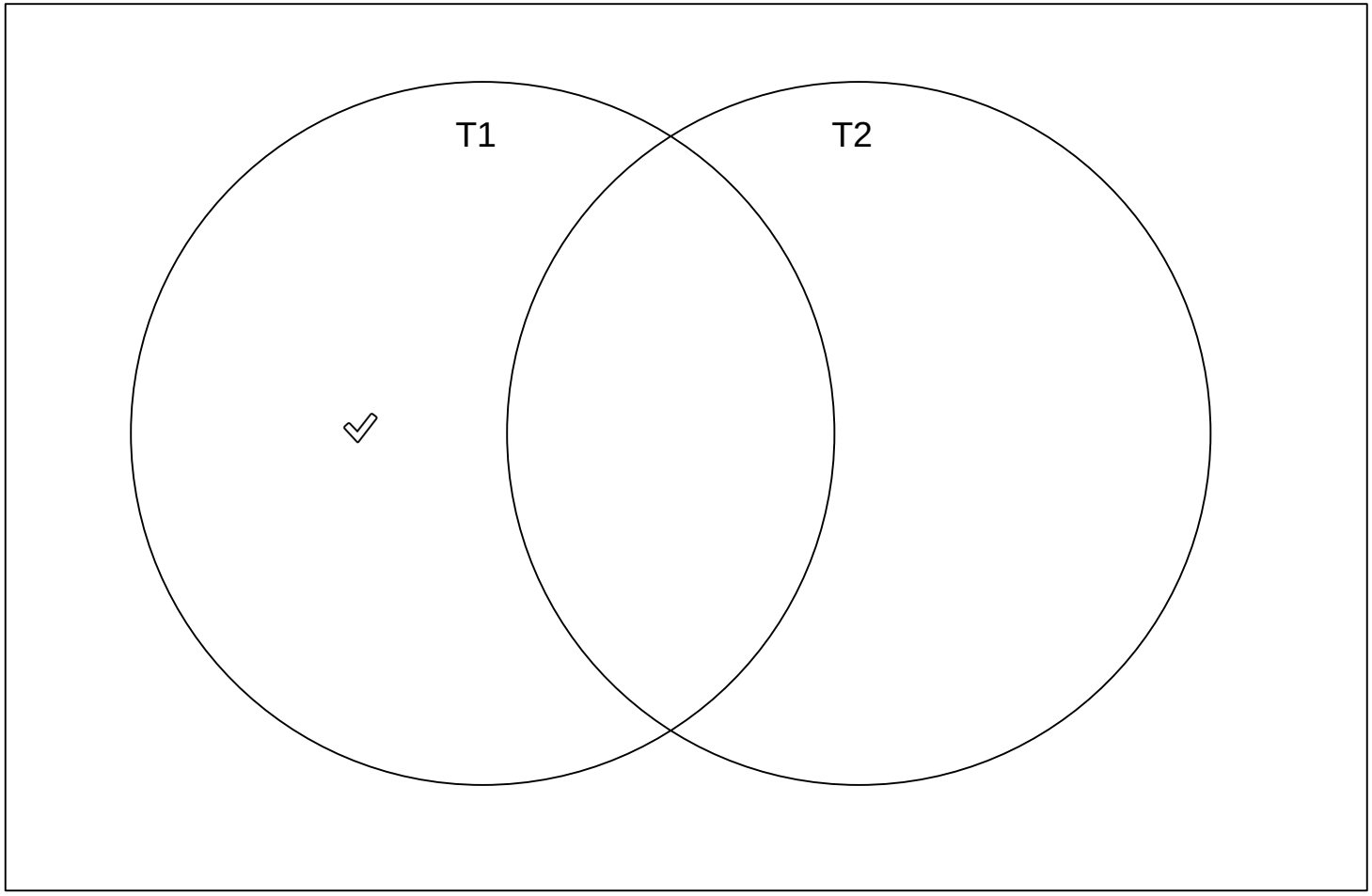
JOINS depicted as Venn Diagrams

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



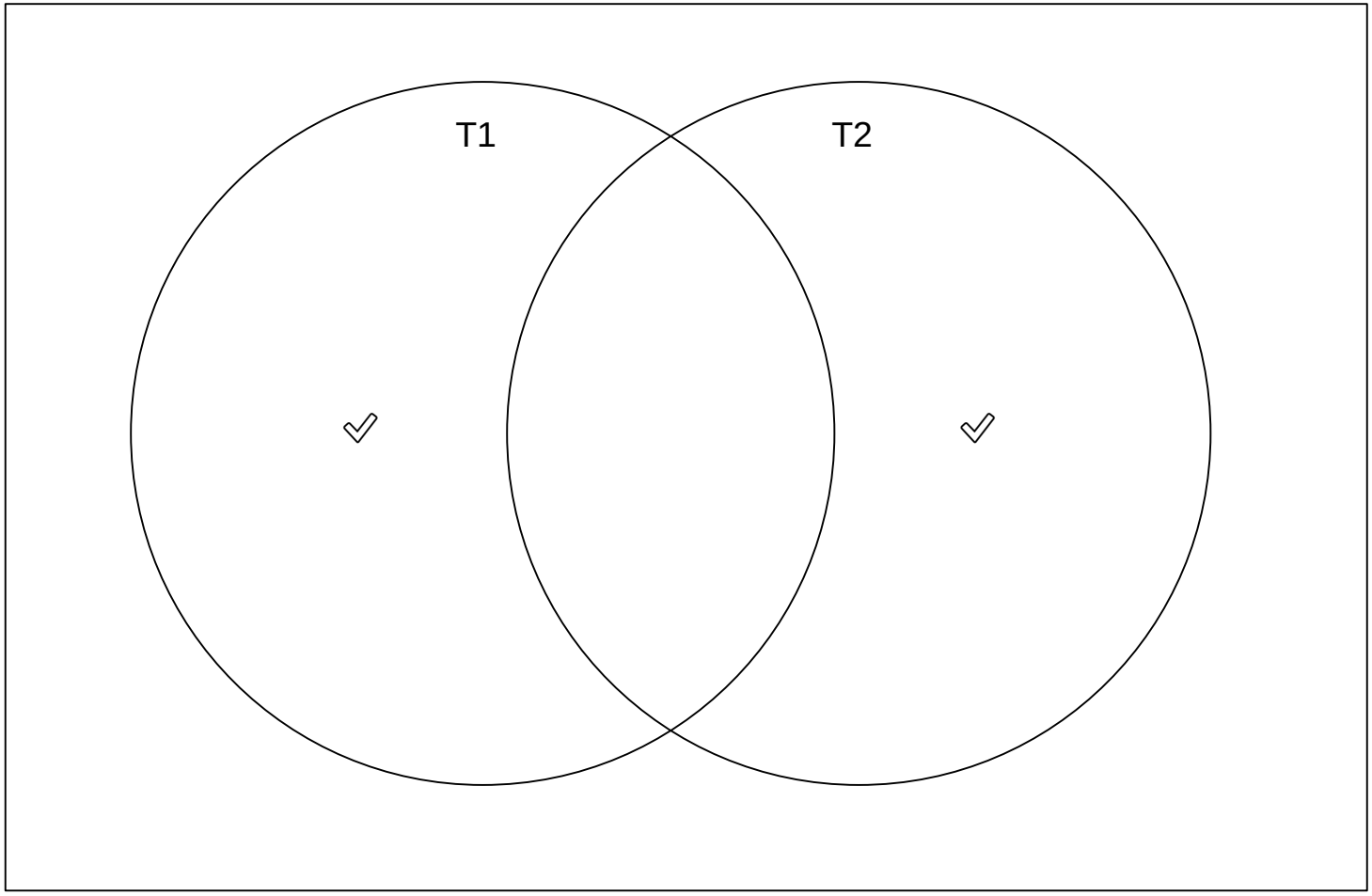
JOINS depicted as Venn Diagrams

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



JOINS depicted as Venn Diagrams

- ?



More on INSERT

- Inserting records from a table
  - Note: table must already exist

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

- Multiple record inserts

```
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");
```

```
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 data in tables!
  - Order 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;
```

- 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 UPDATE Statement - CASE

- A better solution is to use the CASE command

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

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 - What does this do…

```
DELETE FROM Employee;
```

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

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

  - If you delete a row that has rows in other tables dependent on it, either:
    - the dependent rows are deleted too, or
    - the dependent rows get ‘null’ or a default, or
    - your attempt to delete is blocked
    - you decide what action to take when you set up the tables
      - ON DELETE CASCADE or ON DELETE RESTRICT…

Views

- Any relation that is not in the conceptual and logical models, but is made available to the "user" as a virtual relation is called a view.
- Views are good for a number of reasons
  - They help to hide the complexity of queries from users
    - This also hides the structure of the data from users
  - They help to hide data from users
    - Different users use different views
      - Prevents someone accessing the employee tables from accessing employee salaries for instance
    - One way of improving database security
- To create a view…
  - **CREATE VIEW** nameofview **AS** validsqlstatement
- Once a view is defined
  - Its definition is stored in the database (not the data)
  - Can be used just like any other table

Create View Example

```
CREATE VIEW EmpPay AS
SELECT  Employee.ID, Employee.Name, DateHired,
        EmployeeType, HourlyRate AS Pay
        FROM Employee INNER JOIN Hourly
        ON  Employee.ID = Hourly.ID
UNION
SELECT  Employee.ID, Employee.Name, DateHired,
        EmployeeType, AnnualSalary  AS Pay
        FROM Employee INNER JOIN Salaried
        ON  Employee.ID = Salaried.ID
UNION
SELECT  Employee.ID, Employee.Name, DateHired,
        EmployeeType, BillingRate  AS Pay
        FROM Employee INNER JOIN Consultant
        ON    Employee.ID = Consultant.ID;
```

Using a View

```
SELECT * FROM EmpPay;
```

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

ID	Name	DateHired	EmployeeType	Pay
3	Alice	2012-12-02	H	23.43
4	Alan	2010-01-22	H	29.43
1	Sean	2012-02-02	S	92000.00
2	Linda	2011-06-12	S	92300.00
5	Peter	2010-09-07	C	210.00
6	Rich	2012-05-19	C	420.00

```
SELECT * FROM EmpPay
    WHERE EmployeeType = "H" OR EmployeeType = "C"
```

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

Fetched 4 records. Duration: 0.000 sec, fetc

ID	Name	DateHired	EmployeeType	Pay
3	Alice	2012-12-02	H	23.43
4	Alan	2010-01-22	H	29.43
5	Peter	2010-09-07	C	210.00
6	Rich	2012-05-19	C	420.00

Using UPDATE with a VIEW

- Conditions that must be satisfied:
  - The select clause only contains attribute names
    - no expressions, aggregates or distinct specification
  - Any attributes not listed in the select clause can be set to null
  - The query does not have a group by or having clause

Using UPDATE with a VIEW Example

customer = (**customer_name**, customer_street, customer_city)

account = (**account_number**, **branch_name**, balance)

depositor = (**customer_name**, **account_number**)

```
CREATE VIEW custacctinfo AS
SELECT d.account number, d.customer name
FROM depositor d, account a
WHERE d.account number = a.account number
AND a.branch name = 'Deer Park'
```

- Is the above view updateable?

  INSERT INTO custacctinfo VALUES ('L-37', 'John Smith')

  is equivalent to inserts into 3 tables:

  ```
  INSERT INTO depositor VALUES ('John Smith', 'L-37')
  INSERT INTO account VALUES ('L-37', 'Deer Park', null)
  INSERT INTO customer VALUES ('John Smith', null, null)
  ```

- No primary key or foreign key constraints are violated, AND no NOT NULL values are omitted, so this update is valid.

More DDL Commands

- There are more than CREATE!
- ALTER
  - Allows us to add or remove attributes (columns) from a relation (table)
    - ALTER TABLE TableName ADD AttributeName AttributeType
    - ALTER TABLE TableName DROP AttributeName
      - Not supported by many databases (MySQL supports it)
- RENAME
  - Allows the renaming of tables (relations)
    - RENAME TABLE CurrentTableName TO NewTableName

More DDL Commands

- TRUNCATE
  - Same as DELETE * FROM table;
  - 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

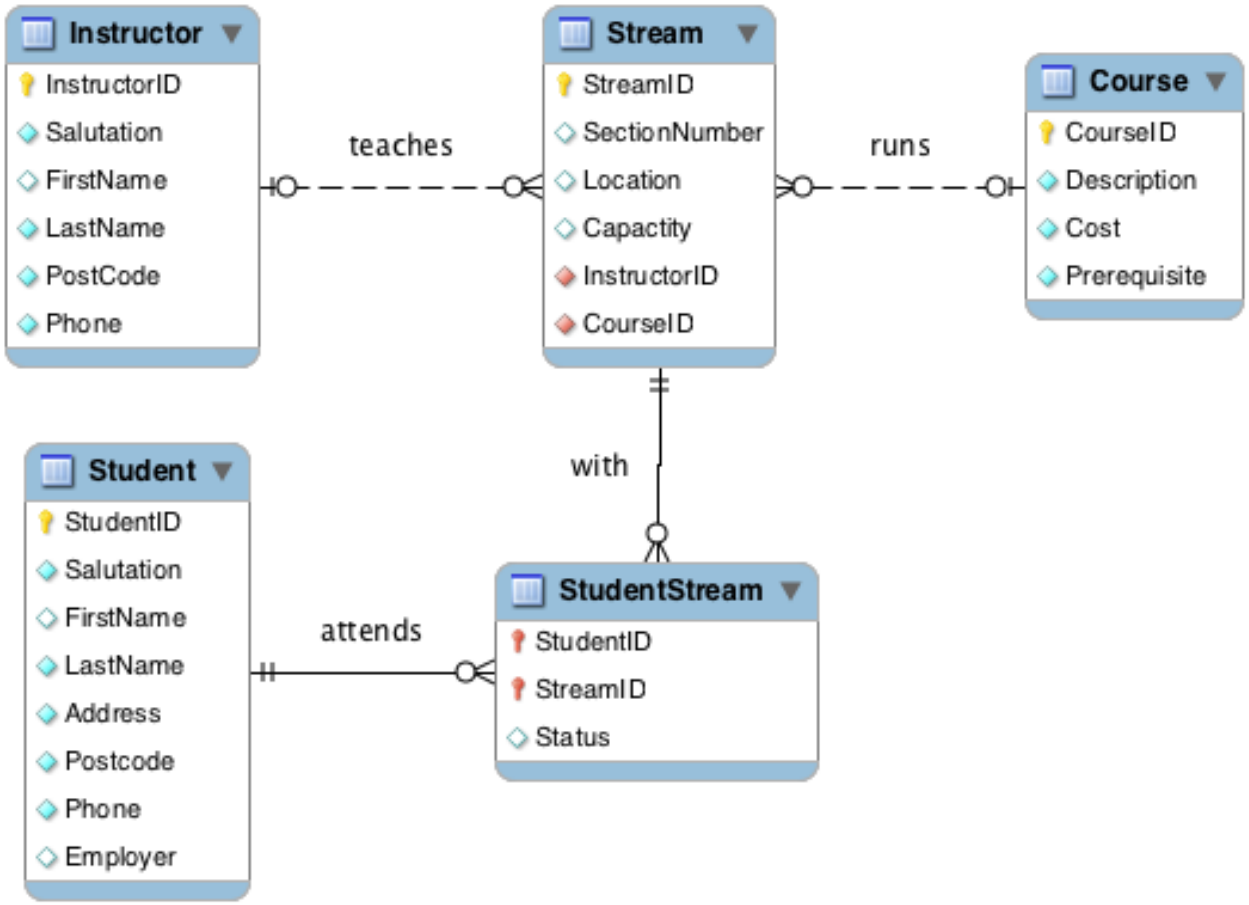
Data Control Language / Other Commands

- DCL
  - Users and permissions
    - **CREATE USER, DROP USER**
    - **GRANT, REVOKE**
    - **SET PASSWORD**
- Other Commands
  - Database administration
    - **BACKUP TABLE, RESTORE TABLE**
    - **ANALYZE TABLE**
  - Miscellaneous
    - **DESCRIBE tablename**
    - **USE db_name**
- MySql calls these ‘Database Administration Statements’

How to Think like SQL

- It's going to be critical for you to think like SQL to handle the queries you will need to write…
- Hopefully the following discussion will help you in this endeavour…
- Firstly, USE the database design as a MAP to help you when you are formulating queries!
- Secondly, USE the structure of the SELECT statement as a template
- Thirdly, FILL out parts of the SELECT structure and BUILD the query…
- Let's try it!

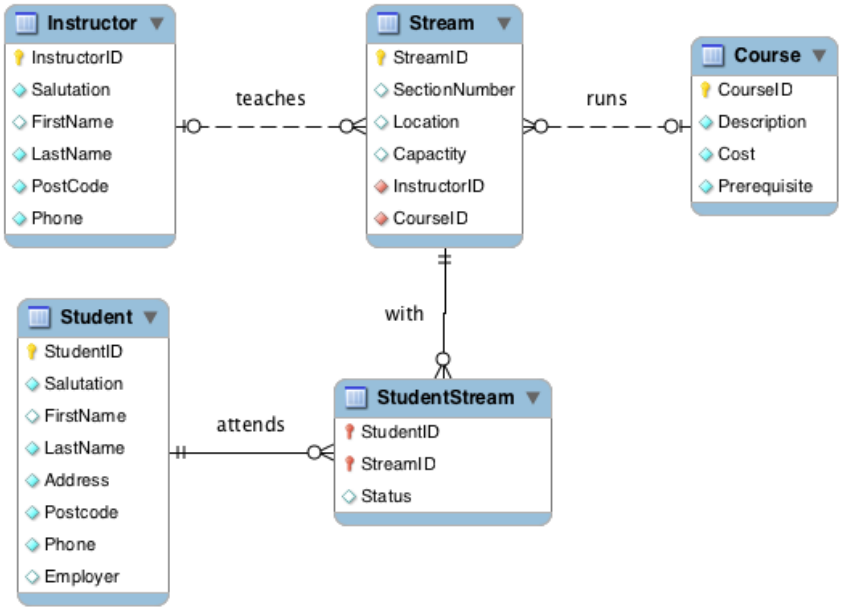
Ad Hoc Query – An Example



- Which employers employ students who are doing a course in locations where the capacity is greater than 20 persons, and what are those locations?

Ad Hoc Query – An Example

- Which employers employ students who are doing a course in locations where the capacity is greater than 20 persons, and what are those locations?
- What is this asking for (what fields)?
  - A report showing
  - Employer, Location, Capacity
  - But only if the capacity > 20
- Need to identify where these things are
- Lets try to use the structure of the SELECT statement now…




```
SELECT Employer, Location, Capacity
FROM Student INNER JOIN StudentStream
ON Student.StudentID = StudentStream.StudentID
INNER JOIN Stream
ON StudentStream.StreamID = Stream.StreamID
WHERE Capacity > 20;
```

Summary

- Expanded your knowledge
  - Various DML Commands
  - Various DDL Commands
  - Various DCL Commands
  - Various Other SQL commands
- How to think about SQL

What's Examinable

- You need to know how to write and think about SQL

Next lecture

- Storage and indexing
  - How is data stored and accessed in a DBMS
  - Alternative types of indexes
  - Going “under the hood” of a DBMS