
Photo: Unsplash

THE UNIVERSITY OF WESTERN AUSTRALIA

Topic Three: SQL Basics

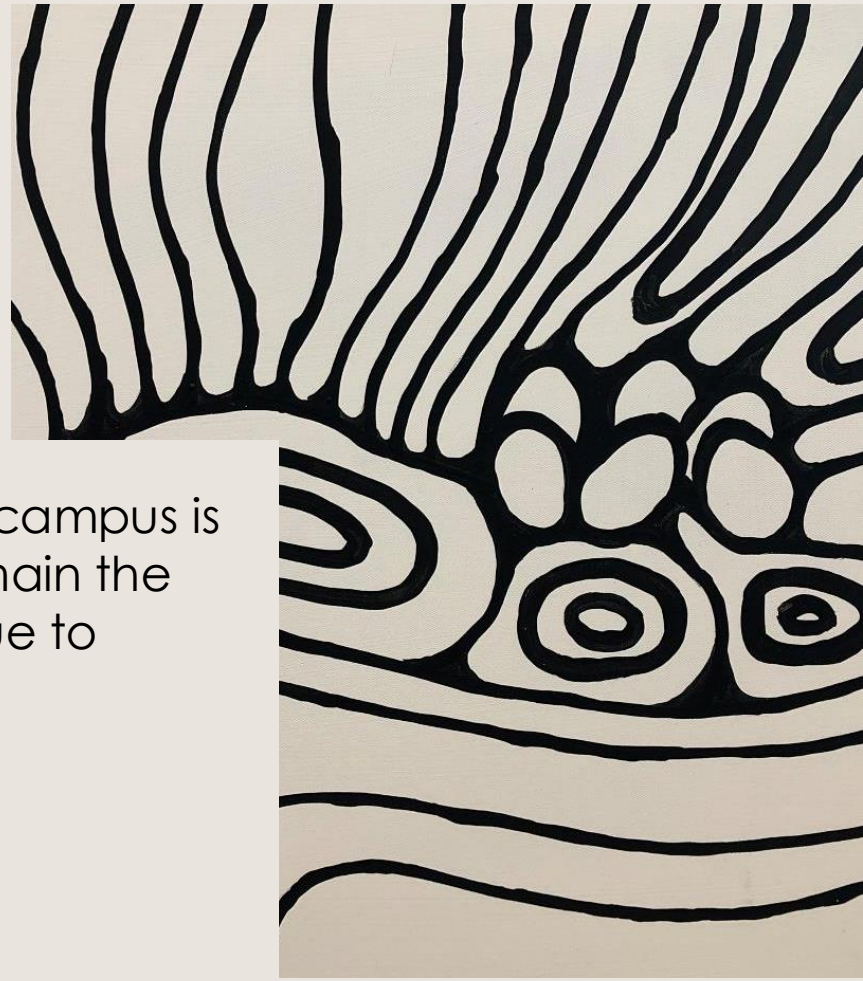
INMT5526: Business Intelligence

Semester Two, 2024

Tristan W. Reed
UWA Business School

Acknowledgement of country

The University of Western Australia acknowledges that its campus is situated on Noongar land, and that Noongar people remain the spiritual and cultural custodians of their land, and continue to practise their values, languages, beliefs and knowledge.



Artist: Dr Richard Barry Walley OAM

Actually building the database

THE UNIVERSITY OF WESTERN AUSTRALIA

- So far, we have looked at *how* we would go about building a database.
  - Considering the entities we may wish to model, their relationships and their attributes.
- However, if we want to actually harness the power of a database, we must actually build it (and populate it with data)!
  - *Recall* why we design a database before we build it?
- There are many steps required when building the database!
  - Setting it up, creating the data model, populating the data…

Changing your password

THE UNIVERSITY OF WESTERN AUSTRALIA

- It is vitally important that you change your password on the MySQL server.
  - By default, both your username and password are your Student ID.
  - This could mean others can access your data – ruining what work you do and creating academic integrity concerns in terms of the assessments.
- Consider this in terms of being a wider security concern – what issue is it?
  - If we stored private data, we aren't storing it securely – likely the first password tried would be just the username again!

“CRUD” operations

THE UNIVERSITY OF WESTERN AUSTRALIA

- It is simply an acronym referring to the four (types of) functions required to implement a computerised storage system:
  - **Create:** add new data to the system (and could consider data modelling);
  - **Read:** view the data (or model) that is already in the system;
  - **Update:** change the data (or model) that already exists in the system;
  - **Delete:** remove a record (or table, or database) in the system;
- We can manipulate data in any way by considering these four (types of) functions!

“ACID” principles

THE UNIVERSITY OF WESTERN AUSTRALIA

- While we are in the area of acronyms, there is another important one we should know about with regards to ensuring valid data remains in our database:
  - **Atomicity**: each command is treated as a whole single statement – all or nothing;
  - **Consistency:** commands issued must ensure the data(base) stays valid;
  - **Isolation:** concurrency ensures the same result as sequential execution;
  - **Durability:** results of commands change the original data and stay this way.
- Much like attributes and properties are used interchangeably, we may also use the terms command, transaction and statement interchangeably.

Creating a database

THE UNIVERSITY OF WESTERN AUSTRALIA

- In MySQL (and most relational and non-relational database systems), our basic building block is a *database*.
  - A database can contain one or more tables – and these can contain some amount of rows and columns (and hence data) within them. *When do we create a new database?*
  - Hence, before we can create a table to model our data (and put data within it), we must first create a database to store the data and models within.
  - Your permissions are restricted so only you can deal with a database starting with "**`<StudentID>_`**". This is a good thing for our unit!

Creating a table

THE UNIVERSITY OF WESTERN AUSTRALIA

- Now that we have a *database*, we can create one or more *tables* within it.
  - We must prefix our tables with the name of our database to put the table within the database (and be allowed to do so) – i.e. **`<StudentID>_db.<TableName>`**.
- We will always refer to the table by the full name (with database prefix) above.
  - Hopefully it is obvious what to substitute above in that table name!
- We then specify (separated by a comma) each of the attributes (columns) within the table, as well as their data type and any special *modifiers*.
  - Can we think of one *modifier* in particular that we should focus on and include!

Table and database relationships

- Our relational database model has a hierarchy of ‘layers’ within it:
  - Our Database Management System may contain zero or more databases;
  - Each database may contain zero or more tables;
  - Each table will contain columns (attributes) and rows (instances/records).
- Our permissions may vary to different tables and databases.


Photo: Unsplash

Topic Three: Data Models

INMT5526: Business Intelligence

Semester Two, 2024

Tristan W. Reed
UWA Business School

Creating a table: example

THE UNIVERSITY OF WESTERN AUSTRALIA

- For readability, we specify this over multiple lines:

```
CREATE TABLE 12345678_db.TestTable(
    id int NOT NULL,
    name varchar(255),
    creationDate date,
    PRIMARY KEY (id)
);
```

MySQL data types

THE UNIVERSITY OF WESTERN AUSTRALIA

- Like most DBMS, MySQL has a list of supported *data types* that specify what type of data can be stored in what attribute.
  - What is the benefit of specifying this?
- The (major) data types and their uses are as follows:
  - **`DATE`**: a date in the **`YYYY-MM-DD`** format (use single quotes in queries around it);
  - **`TIME`**: a time in the **`HH:MM:SS`** format (similar caveat to the above);
  - **`DATETIME`**: a date in the above format, as well as a time in the above format;

(Further) MySQL data types

THE UNIVERSITY OF WESTERN AUSTRALIA

- The (major) data types and their uses are as follows:
  - **`VARCHAR(X)`**: a *variable length* string of X characters (benefits?);
  - **`CHAR(X)`**: a fixed length string of X characters;
  - **`BOOLEAN`**: a **`TRUE`** or **`FALSE`** value (stored as **`0`** or **`1`**);
  - **`INT`**: an integer (whole) number;
  - **`FLOAT`**: a floating point (with decimal portion) number;

“Esoteric” MySQL data types

THE UNIVERSITY OF WESTERN AUSTRALIA

- There exists many more ‘esoteric’ data types:
  - e.g. JSON documents, binary strings and blobs (can store images), spatial geometries;
- We will look into some of these later.
  - There are certainly benefits in using some of these formats;
  - However, their use cases are more specialised than the other formats;
  - It can be seen how the above formats handle ‘usual’ cases;
  - However, we have multiple types of numeric formats (e.g. **`TINYINT`**).

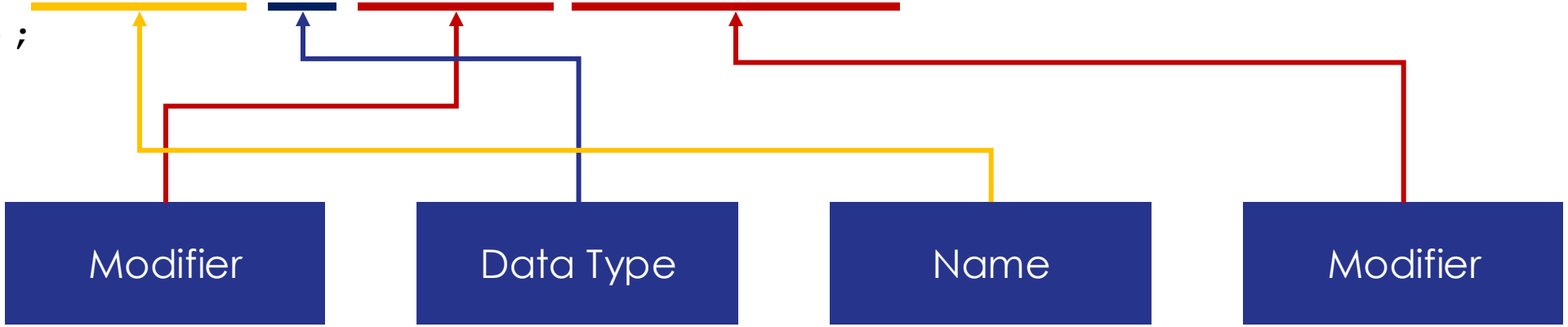
Attribute (Field) modifiers

THE UNIVERSITY OF WESTERN AUSTRALIA

- There are a few modifiers (there are many more) that we can add:
  - **`PRIMARY KEY`** to specify that a field(s) is a primary key;
  - **`AUTO_INCREMENT`** to automatically increase an integer field for each row;
  - **`NOT NULL`** to ensure a value is specified for each observation (row);
  - **`FOREIGN KEY`** to specify a foreign key relationship (next week's job).
- MySQL goes about this a bit unusually in case of the keys.
  - These are *generally* specified at the end of the statement, rather than in the 'middle' like they are with other DBMS – one of the quirks that I was talking about!

Putting it all together

- When *creating* a new table (i.e. specifying the data model), for each of the attributes (i.e. columns or fields) we specify the name, then the data type and then after that any modifiers that we have, as seen below:
  -
```
CREATE TABLE 12345678_db.ExampleTable (
    fieldName INT NOT NULL AUTO_INCREMENT
);
```



Modifying a table

THE UNIVERSITY OF WESTERN AUSTRALIA

- We may want to adjust our data model after we have created it.
  - However, we must consider any issues this creates with respect to the integrity of the data within the table – especially if we add a new column.
- This is achieved through an **`ALTER TABLE`** command:
  - **`ALTER TABLE <TableName> ADD COLUMN <ColumnName> <DataType> <Modifiers>;`** to add a new column **`<ColumnName>`** to **`<TableName>`** (with **`<DataType>`** and **`<Modifiers>`**);
  - **`ALTER TABLE <TableName> DROP COLUMN <ColumnName>;`** to remove a column **`<ColumnName>`** from **`<TableName>`**.

Altering a field

THE UNIVERSITY OF WESTERN AUSTRALIA

- Alternatively (pun somewhat intended) we can change or modify one of the columns without recreating it using the following commands/queries:
  - **`ALTER TABLE <TableName> RENAME COLUMN <OldColumnName> TO <NewColumnName>;`** to change just the column name from **`<OldColumnName>`** to **`<NewColumnName>`**.
  - **`ALTER TABLE <TableName> MODIFY COLUMN <ColumnName> <DataType> <Modifiers>;`** to change the data type (**`<DataType>`**) and/or modifiers (**`<Modifiers>`**) for column **`<ColumnName>`.**
    - If you wish to keep something the same, specify the same value for what they were before.
- Alternatively, an **`ALTER TABLE`** with the **`CHANGE`** command can be used to do both:

```
ALTER TABLE <TableName> CHANGE <OldColumnName> <NewColumnName>
<DataType> <Modifiers>;
```

Do we need `COLUMN`?

THE UNIVERSITY OF WESTERN AUSTRALIA

- The word **`COLUMN`** in these commands is technically optional in MySQL.
  - For this unit, we will always include it for clarity.
- Following on from this – yes, some of the commands can be a bit verbose and we are forced to write them out exactly in MySQL Workbench.
  - However, this is great for practice and reinforcing our learnings.
- We may have systems and tools around us to make this job easier.
  - Graphical interfaces, object-relational models, copy-and-paste and so on…

Deleting a table

THE UNIVERSITY OF WESTERN AUSTRALIA

- We may wish to get rid of a table – but we must be careful!
  - This will delete all of the data within the table, not just the model itself!
- We achieve this through a **`DROP TABLE`** command.
  - **`DROP TABLE <TableName>;`**
- This is why we should keep track of the commands that we issue to the MySQL Workbench to manipulate our database!
  - You may wish to save (export) your queries as you go.

Viewing (reading) a table

THE UNIVERSITY OF WESTERN AUSTRALIA

- We can examine the data types and attribute (field/column) names of our table by issuing the following command:
  - **`SHOW COLUMNS FROM <TableName>;`**
- Remember that we must specify the database name as part of the "table name" (e.g. **`12345678_db.<TableName>`**)!

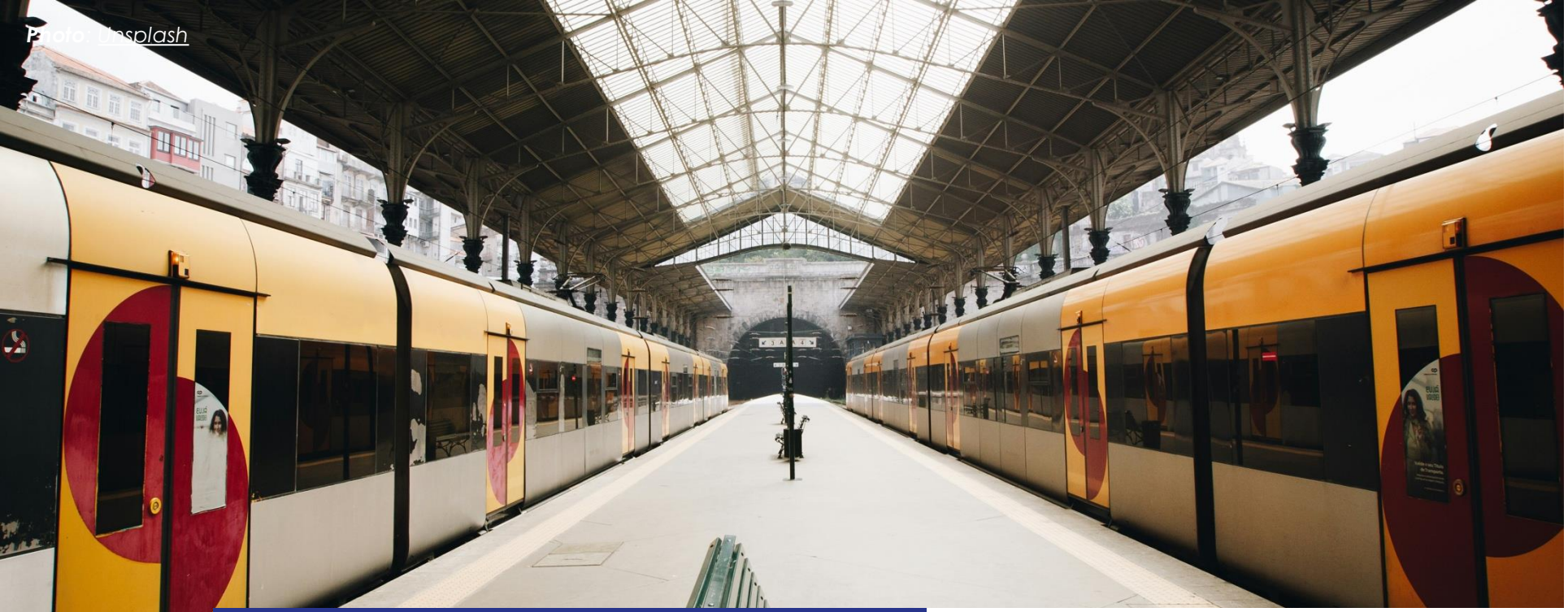


Photo: Unsplash

THE UNIVERSITY OF WESTERN AUSTRALIA

Topic Three: Data Itself

INMT5526: Business Intelligence

Semester Two, 2024

Tristan W. Reed
UWA Business School

Adding data

THE UNIVERSITY OF WESTERN AUSTRALIA

- So far, we have just looked at building our data model.
  - Specifying *how* the data in the database should look.
- We now need to focus on actually *inserting* the data into the database tables.
  - What use is a database with no data?
- Generally, this would be achieved through bulk or automated means.
  - However, we will be doing this manually to emphasise our understanding and to examine how each command works.

Adding data: example

THE UNIVERSITY OF WESTERN AUSTRALIA

- The command to insert (add/create) data within a table is relatively straightforward:
  - **`INSERT INTO <TableName> (<ColumnNameOne>, <ColumnNameTwo>, …)`**
    `VALUES (<ValueOne>, <ValueTwo>, …);`
- This would create a single new record within **`<TableName>`**, where the value of **`<ColumnNameOne>`** is **`<ValueOne>`** and the value of **`<ColumnNameTwo>`** is **`<ValueTwo>`**.
  - Repeat for however many columns that we have – only required if **`NOT NULL`**.

Changing and updating data

THE UNIVERSITY OF WESTERN AUSTRALIA

- The command to update (change) data in MySQL is relatively straightforward and similar to the command to *create* data in MySQL.
  - **`UPDATE <TableName> SET <ColumnNameOne> = <NewValueOne>, <ColumnNameTwo> = <NewValueTwo>, … WHERE <ColumnName> = <Value>;`**
- We can specify a **`WHERE`** clause after this (before the semicolon as shown in navy above) to only affect (change) certain rows only.
  - We would want to do this for at least e.g. **`id = 2`** so we only affect one row.

Reading data

THE UNIVERSITY OF WESTERN AUSTRALIA

- To read (or view) data within the table in MySQL, we use the "Select" statement in one of the two ways seen below:
  - **`SELECT <ColumnNameOne>, <ColumnNameTwo> FROM <TableName>;`**
  - **`SELECT * FROM <TableName>;`**
- We can also view a subset of our data by specifying a **`WHERE`** clause in the same manner as with changing data.

Deleting data

THE UNIVERSITY OF WESTERN AUSTRALIA

- The command to delete data is relatively similar to deleting a table.
  - **`DELETE FROM <TableName> WHERE <ColumnNameOne> = <ValueOne>;`**
- If we do not specify a **`WHERE`** clause, then everything in the table is deleted;
  - Tread carefully – you may wish to do a **`SELECT`** first to test!
- We won't delete any data for now.
  - This is just for your reference!

THE UNIVERSITY OF WESTERN AUSTRALIA

Questions and Answers

tristan.reed@uwa.edu.au