122COM: Databases

Coventry University

Overview

1 **Introduction**
- SQL
- SQLite

2 **Code**
- Dynamic queries
  - SQL injection

3 **Recap**

4 **Further reading**

Coventry
University

122COM: Databases
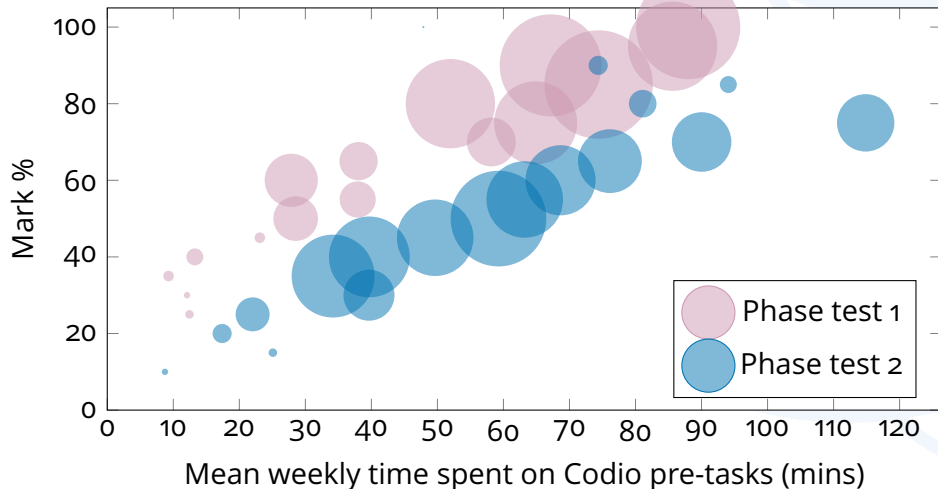
Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

Expectations

You have all attempted the green Codio exercises for this week.

122COM results 2016-17 September starters.

Database *(*noun) - a collection of information that is organized so that it can easily be accessed, managed, and updated.

SQL C

Database *(*noun) - a collection of information that is organized so that it can easily be accessed, managed, and updated.

- Pronounced S-Q-L or Sequel.
  - Structured Query Language.
- Used to query relational databases.
- Theoretically it doesn't matter what underlying database is.
  - MS SQL Server, Oracle, PostgreSQL, MySQL, SQLite.
  - In reality lots of minor variations.

Coventry
University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

Relational Databases C

Built around tables.

- Can be imagined like a spreadsheet.

Row/record →

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

↑
Column/attribute

Coventry University

Many types of query.

- SELECT - Get information from the database.
- INSERT - Add information to the database.
- DELETE - Remove information.

Also used for database administration.

- CREATE - Create a whole new table/schema/function.
- ALTER - Modify a table/schema/function.
- DROP - Delete a whole table/schema/function.

Coventry University

SELECT

Used to retrieve information from the database.

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

SELECT

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

Coventry University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further
reading

SELECT C

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
SELECT * FROM staff;
```

∗ means everything.

Coventry
University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

SELECT C

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
SELECT * FROM staff;
```

* means everything.

#	id	forename	surname	job
1	0	Malcolm	Reynolds	Captain
2	4	Zoe	Washburne	Co-captain
3	11	Hoban	Washburne	Pilot
4	23	Kaywinnet	Frye	Mechanic

Coventry University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further
reading

SELECT

C

Used to retrieve information from the database.

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

Coventry
University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

SELECT   C

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
SELECT * FROM staff WHERE surname = "Washburne";
```

Only return the records WHERE something is true.

122COM:
Databases

Introduction
SQL
  SQLite
Code
  Dynamic queries
    SQL injection
Recap
Further
reading

SELECT C

Used to retrieve information from the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```sql
SELECT * FROM staff WHERE surname = "Washburne";
```

Only return the records `WHERE` something is true.

#	id	forename	surname	job
1	4	Zoe	Washburne	Co-captain
2	11	Hoban	Washburne	Pilot

Coventry
University

What if we want to now how many records there are?

- count() function.
- More efficient.
    - Minimum amount of data.

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

count()

What if we want to now how many records there are?

- count() function.
- More efficient.
    - Minimum amount of data.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

Coventry University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

count()

What if we want to now how many records there are?

- count() function.
- More efficient.
    - Minimum amount of data.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
SELECT count(*) FROM staff;
```

Coventry University

count()

What if we want to now how many records there are?

- count() function.
- More efficient.
  - Minimum amount of data.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
SELECT count(*) FROM staff;
```

#	count(*)
1	4

Coventry
University

INSERT

Used to add information to the database.

122COM:
Databases

Introduction
SQL
 SQLite

Code
 Dynamic queries
  SQL injection

Recap

Further
reading

INSERT

Used to add information to the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

Used to add information to the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
INSERT INTO staff VALUES (42, 'Simon', 'Tam', 'Doctor');
```

122COM:
Databases

Introduction
SQL
  SQLite
Code
  Dynamic queries
    SQL injection
Recap
Further
reading

INSERT

Used to add information to the database.

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic

```
INSERT INTO staff VALUES (42, 'Simon', 'Tam', 'Doctor');
```

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic
42	Simon	Tam	Doctor

Coventry
University

Don't have to supply values for all the columns.

- Depends on the table design.

Coventry
University

Don't have to supply values for all the columns.

- Depends on the table design.

```
INSERT INTO staff (forename, id, surname)
       VALUES ('River', 43, 'Tam');
```

Coventry University

Don't have to supply values for all the columns.

- Depends on the table design.

```
INSERT INTO staff (forename, id, surname)
      VALUES ('River', 43, 'Tam');
```

id	forename	surname	job
0	Malcolm	Reynolds	Captain
4	Zoe	Washburne	Co-captain
11	Hoban	Washburne	Pilot
23	Kaywinnet	Frye	Mechanic
42	Simon	Tam	Doctor
43	River	Tam	

Coventry
University

Why use databases at all?
Why not just use dictionaries and lists or similar?

Databases...
- Have structure.
    - Easy to organise the data.
- Scale.
    - Can handle a LOT of data.
- Multi-user.
    - Can have lots of people working on the same data.
- Fault tolerant.
    - Can recover if things go wrong.

Coventry University

Using SQLite3 in labs.

- Not a fully featured database.
   - But has all the basic features.
   - SQL.
- Good for small/non-urgent databases.
   - $\leq$ gigabytes of data.
- Efficient
   - Don't need to waste resources on a 'real' database.
- Convenient.
   - Don't need to install, configure, manage a 'real' database.
   - Portable, 1 file.
- No network.
   - Single user only.

Python

How to use SQL queries in Python?

```python
import sqlite3 as sql                      # sqlite module

con = sql.connect( 'firefly.sqlite' )      # open database
cur = con.cursor()

cur.execute( '''SELECT * FROM staff;''' )  # run query
for row in cur:                            # loop over results
    print( row )

con.close()                                # close database
```

lec_select.py

Coventry
University

Python

How to use SQL queries in Python?

```python
import sqlite3 as sql          # sqlite module

con = sql.connect( 'firefly.sqlite' )   # open database
cur = con.cursor()

cur.execute( '''SELECT * FROM staff;''' )   # run query
for row in cur:                          # loop over results
    print( row )

con.close()                              # close database
```

lec_select.py

```
(0, 'Malcolm', 'Reynolds', 'Captain')
(4, 'Zoe', 'Washburne', 'Co-captain')
(11, 'Hoban', 'Washburne', 'Pilot')
(23, 'Kaywinnet', 'Frye', 'Mechanic')
```

Coventry
University

122COM:
Databases

Introduction
  SQL
  SQLite

Code
  Dynamic queries
  SQL injection

Recap

Further
reading

C++

How to use SQL queries in C++?

```cpp
#include "libsqlite.hpp"                    // sqlite library

int main()
{
    sqlite::sqlite db( "firefly.sqlite" );   // open database

    auto cur = db.get_statement();          // create query
    cur->set_sql( "SELECT * FROM staff;" );
    cur->prepare();                          // run query

    while( cur->step() )                     // loop over results
        cout << cur->get_int(0) << " " << cur->get_text(1) << endl;
}
```

lec_select.cpp

```
0  Malcolm
4  Zoe
11 Hoban
23 Kaywinnet
```

Break

Static queries

So far looked at static queries.

- Same query is run every time.
- Real power is in dynamic queries.
  - Code creates changes the SQL to ask new questions.

**122COM:
Databases**

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further
reading

Dynamic queries

```python
import sqlite3 as sql

con = sql.connect('firefly.sqlite')
cur = con.cursor()

question = input('Who is the...')

cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))

for row in cur:
    print('%s %s' % row)
```

lec_dynamic.py

```
        Who is the...Captain
        Malcolm Reynolds
```

Coventry
University

Dynamic queries C++

Using sqlitepp.

- 3rd party wrapper around default SQLite3 API.
- Simplified use.

```cpp
sqlite::sqlite db( "firefly.sqlite" );

string question;
cout << "Who is the...";
cin >> question;

auto s = db.get_statement();
s->set_sql( "SELECT forename, surname FROM staff "
            "WHERE job = ?;" );
s->prepare();
s->bind( 1, question );

while( s->step() )
{
    string forename = s->get_text(0);
    string surname = s->get_text(1);
    cout << forename << " " << surname << endl;
}
```
lec_dynamic.cpp

Coventry
University

Bad dynamic queries

Dynamic queries should **ALWAYS** use placeholders (i.e. ?).

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))
```

Dynamic queries must **NEVER** be created by manipulating strings.

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "%s";''' % question )
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "{}";'''.format( question) )
```

- User could input anything, e.g. SQL commands!.
  - Captain"; DROP TABLE staff; -
- Sanitise your inputs.

Coventry University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

Bad dynamic queries

Dynamic queries should **ALWAYS** use placeholders (i.e. ?).

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))
```

Dynamic queries must **NEVER** be created by manipulating strings.

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "%s";''' % question )
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "{}";'''.format( question) )
```

- User could input anything, e.g. SQL commands!.
  - `Captain"; DROP TABLE staff; -`
- Sanitise your inputs.
- Always use placeholders.

Bad dynamic queries

Dynamic queries should **ALWAYS** use placeholders (i.e. ?).

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))
```

Dynamic queries must **NEVER** be created by manipulating strings.

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "%s";''' % question )
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "{}";'''.format( question) )
```

- User could input anything, e.g. SQL commands!.
  - `Captain"; DROP TABLE staff; -`
- Sanitise your inputs.
- Always use placeholders.
  - No exceptions.

Coventry
University

122COM: Databases

Introduction
SQL
SQLite
Code
Dynamic queries
SQL injection
Recap
Further reading

Bad dynamic queries

Dynamic queries should **ALWAYS** use placeholders (i.e. ?).

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))
```

Dynamic queries must **NEVER** be created by manipulating strings.

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "%s";''' % question )
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = "{}";'''.format( question) )
```

- User could input anything, e.g. SQL commands!.
  - `Captain"; DROP TABLE staff; -`
- Sanitise your inputs.
- Always use placeholders.
  - No exceptions.
  - NO EXCEPTIONS!

122COM: Databases

Introduction
 SQL
 SQLite

Code
 Dynamic queries
 SQL injection

Recap

Further reading

Bad dynamic queries

Dynamic queries should **ALWAYS** use placeholders (i.e. ?).

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = ?;''', (question,))
```

Dynamic queries must **NEVER** be created by manipulating strings.

```
cur.execute('''SELECT forename, surname FROM staff
               WHERE job = '%s';''' % question)

cur.execute('''SELECT forename, surname FROM staff
               WHERE job = '{};'''.format(question))
```

- User could input anything, e.g. SQL commands!.
  - Captain"; DROP TABLE staff; -
- Sanitise your inputs.
- Always use placeholders.
  - No exceptions.
  - NO EXCEPTIONS!

Coventry University
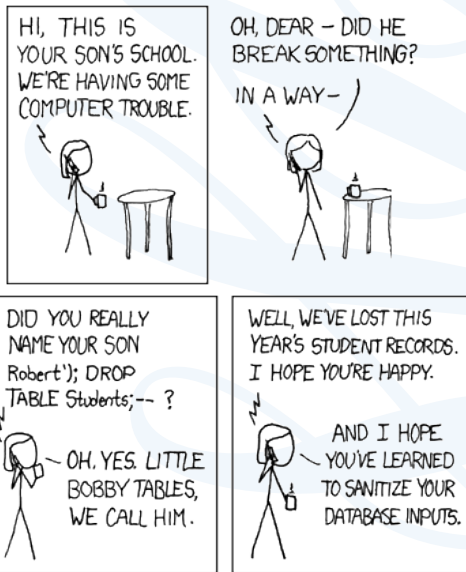
SQL injection   A

Around since at least 1998.

Notable SQL injection attacks.

- 2017 Equifax - 143,000,000 US consumers potentially impacted.
  - Or to put it another way, half of America.
- 2015 TalkTalk - 160,000 customers' details.
- 2014 Hold security - found 420,000 vulnerable websites.
- 2011 MySql - mysql.com compromised.
- 2008 Heartland Payment - 134,000,000 credit cards.

Many, many more.

https://xkcd.com/327/

Coventry University

Injection attacks are **STILL** No. 1 on Open Web Application Security Project (OWASP) Top 10 list.

- How is this still a thing?

- Do **NOT** write code that is vulnerable to this.
    - Do **NOT** write code that execute user input directly.
    - Just use placeholders! Problem solved.

- SQL injection is a critical bug and I **WILL** mark down code that is vulnerable.

Coventry
University

Recap

- SQL used to query databases.
- Databases are...
    - fault tolerant.
    - multi user.
    - scalable.
- Always use place holders in dynamic queries.
    - Say no to SQL injection!

Coventry
University

122COM: Databases

Introduction
SQL
SQLite

Code
Dynamic queries
SQL injection

Recap

Further reading

Why do I care?

- Everyone
  - Structured Query Language (SQL) is widely used, most in demand language[1].
  - Should be aware of and able to defend against SQL injection.
  - Experience in using 3rd party libraries/modules in software.
- Computing - SQL is a vital for much of the web. Heard of LAMP servers?, the M is for MySQL.
- Ethical Hackers - need to understand SQL injection.
- ITB - SQL is widely used in business applications, especially for generating reports.
- Games Tech & MC- SQL is used in games, i.e. for save games.

[1]According to Indeed.com

Further reading

- Introduction to SQL - `http://www.w3schools.com/sql/sql_intro.asp`
- SQL injection hall of shame - `http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/`
- Efficient inserting - the `executemany()` method.

Coventry
University

- Complete the yellow Codio exercises for this week.
- Revise for Phase Test 1. Worth 20% of your 122COM marks.

- If you have spare time attempt the red Codio exercises.

- If you are having issues come to the PSC.
  ```
  https://gitlab.com/coventry-university/
  programming-support-lab/wikis/home
  ```

Coventry
University

The End