SQL DML except queries

Overview of this video

A run through of SQL DML (DML = Data Manipulation Language)

- Except for queries, which is more complex and thus has its own videos

Data Manipulation Language (DML)

Insert rows into a table

Delete rows from a table

Update rows in a table

Query a table

Not this video (but they are part of DML)

Insert

Before: Students

name	number	programme
Anna	20171989	G402
John	20174378	G702

```
INSERT INTO Students
VALUES ('Oliver',20171112,'G402');
```

' is used to denote the start or end of a string

After: Students

name	number	programme
Anna	20171989	G402
John	20174378	G702
Oliver	20171112	G402

Insert

Before: Students

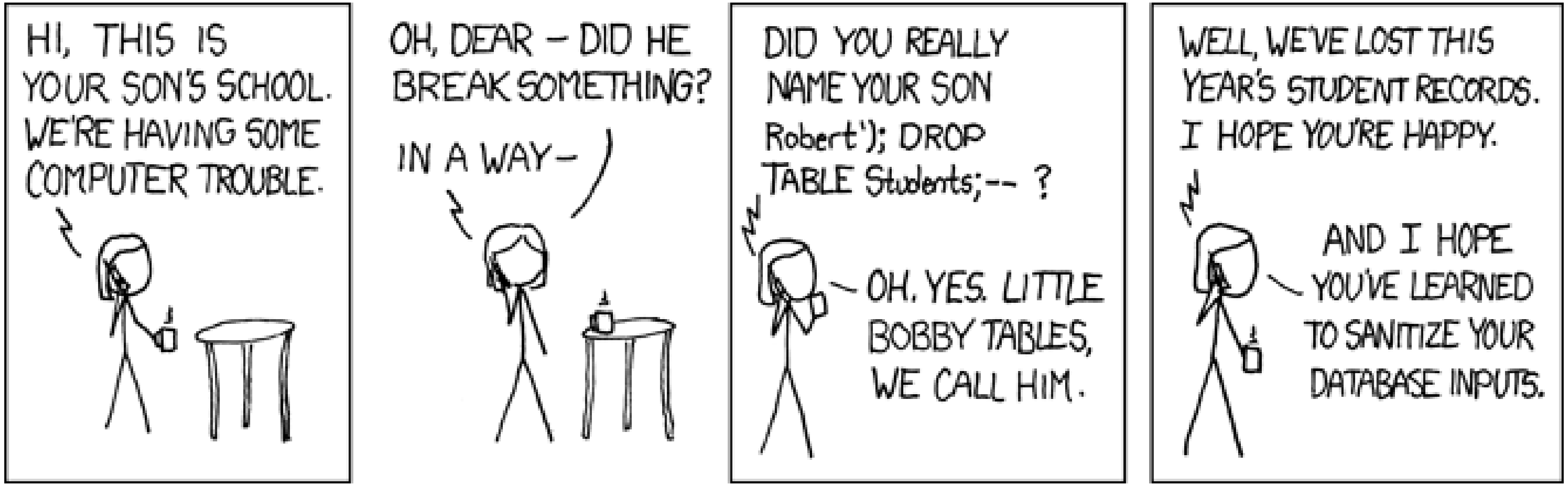
name	number	programme
Anna	20171989	G402
John	20174378	G702
Oliver	20171112	G402

```
INSERT INTO Students(programme,name)
VALUES('G702','Danny');
```

After: Students

name	number	programme
Anna	20171989	G402
John	20174378	G702
Oliver	20171112	G402
Danny	*null*	G702

SQL Injections


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Credits: https://xkcd.com/327/

This is the most common exploit online (~50% of all exploits are SQL Injections)

What happens is: You make a website with a form, you take what people inputs into the form and insert the fields into your SQL database

Insert

~~Students~~

~~name~~	~~number~~	~~programme~~
~~Anna~~	~~20171389~~	~~G402~~
~~John~~	~~20174378~~	~~G702~~
~~Oliver~~	~~20171112~~	~~G402~~
~~Danny~~	~~*null*~~	~~G702~~

```
INSERT INTO Students(programme,name)
VALUES('G702','Robert'); DROP TABLE students; --');
```

-- is used to denote comments

FROM in SQL basically sets the input table (so FROM Students mean set Students as the input table)

How to remo hn

```
DELETE FROM Students
WHERE name='John';
```

Second line means: … where name is equal to John

Warning:

```
DELETE FROM Students;
WHERE name='John'
```

Students

name	number	programme
Anna	20171989	G402
Oliver	20171112	G702
Danny	*null*	G702
Anna	20171234	G702
Anna	20171234	G702

Conditions in WHERE clauses

Comparisons: =,<,<=,>=,>,<> (or != for the last)

- Used for equals, strictly less than, less than or equal, greater than or equal, strictly greater and not equal of e.g. numbers

Conditions can contain:

- AND
  - E.g. if you want both that the name is Oliver and the programme is G402, you write WHERE name = ‘Oliver’ AND programme = ‘G402’
- OR
  - Similar to AND, but used if you want or…
- NOT
  - If you want everything but something in particular
- BETWEEN
  - E.g. “Price BETWEEN 10 AND 20” if you want the price to be between 10 and 20
- LIKE
  - For string matching
  - _ matches any 1 letter and % any number of letters
  - E.g. “Name LIKE ‘O%r’” and “Name LIKE ‘O____r’” matches Oliver

Conditions in WHERE clauses cont.: IN

```
DELETE FROM Students
WHERE name IN (‘John’,’Sebastian’);
```

Special version using queries – see the video on queries – the optional part

Students

name	number	programme
Anna	20171989	G402
Oliver	20171112	G702
Danny	*null*	G702
John	*null*	G702

UPDATE

How to change Oliver

```
UPDATE Students
SET programme=‘G402’
WHERE name=‘Oliver’;
```

Relative changes

```
UPDATE Students
SET number=number+1
WHERE name=‘Oliver’;
```

Therefore, = in WHERE is not the same type as = in SET
I.e. = in WHERE is for comparison (like == in Python or Java)
and = in SET is for change value (like = in Python or Java)

Students

name	number	programme
Anna	20171989	G402
Oliver	20171113	G402
Danny	*null*	G702
Anna	20171234	G702