Lesson : Writing SQL Statements

All the exercises use the library.db database in Worksheet 3. A populated copy of the database can be found in the google drive data folder.

UPDATE command

The UPDATE command allows the editing of data values in a database. One or more records may be updated at the same time.

For example, the statement
```
UPDATE Borrower Set Contact = 98764326 WHERE ID=2
```
will update 1 record whose ID = 2

Another example
```
UPDATE Book Set Title = 'Book: ' || Title
```
will update the values of Title in the Book table such that each book title now start with 'Book: '. Note the use of `||` for string concatenation (adding of two strings).

Consider another example. All the books borrowed by Kumara (BorrowerID 3) should have been borrowed by Sarah (BorrowerID 2). Then, the following SQL statement can be carried out to update the Loan table.
```
UPDATE Loan Set BorrowerID = 2 WHERE BorrowerID = 3
```

Exercise:

1. All the books borrowed by Sarah (BorrowerID 2) should have been borrowed by Kumara (BorrowerID 3). Write down the SQL statement to update the Loan table.

   UPDATE Loan SET BorrowerID = 3 WHERE BorrowerID = 2

   -> what if you want to use the names (supposing the name is the primary key and youdon't know the BorrowerID)
   -

2. All the contact numbers of borrowers in the Borrower table must add the international dialling pre-fix of "+65". Write down the SQL statement to update the Borrower table.

   UPDATE Borrower SET Contact = "+65" || Contact

3. Sarah wants to change her name to Sasha Li. Write down the SQL statement to update the record.

   UPDATE Borrower SET FirstName = "Sasha", SurName = "Li"
   WHERE ID = 2

   assuming that Sarah's ID is 2
   alternative solution: when FirstName is "Sarah"
   assuming that there is only one person named "Sarah"

DELETE command

The DELETE command is used to delete existing records in a table.

For example,
```
DELETE FROM Books WHERE Title = 'Life of Pie'
```
This will delete the book 'Life of Pie' from the Books table.
Please note that the text is case-sensitive, which means that the following 2 SQL statements are different.
```
DELETE FROM Book WHERE Title =  'Life of Pie'
DELETE FROM Book WHERE Title =  'Life of pie'.
```

Be careful when using the DELETE statement. For example:
```
DELETE FROM Loan
```
This will delete all entries from the Loans table!

Exercise

1. Which of the following SQL statements are valid? (Tick the statements)

V ■	`DELETE FROM Book WHERE Title = 'H2 Computing Ten Year Series'`
	`DELETE FROM Book WHERE Title = *`
	`DELETE FROM Book WHERE`
V	`DELETE FROM Book`

SELECT statement

The SELECT statement allows the user to retrieve data from the database.

To select all fields, use `*`.

For example, typing
```
SELECT * FROM Book
```
will give you details of all the books in library.

Conditions may be added using WHERE.

For example, `SELECT * from Book WHERE Damaged = 1`
The statement will return all the damaged books.

You can also find NULL values, for example books with no publishers (no value on the PublisherID field), using the IS operator.

```
SELECT * from Book WHERE PublisherID IS NULL
```

You can also search for terms which are not NULL, for example:

```
SELECT * from Book WHERE PublisherID IS NOT NULL
```

This statement returns all records where there is a publisher.

You can insert more than one condition using AND and OR binary operators.
For example,

```
SELECT * FROM Book WHERE Title = 'Life of Pie' AND Damaged = 0
```

This statement returns the books with Title 'Life of Pie' **and** are not damaged.

What if you only need the book titles?
In that case, include the fields you want in the SELECT statement. For example,

```
SELECT Title FROM Book
```

This statement will give you all the Title of the books in the Book table.

Title
The Lone Gatsby
A Winter's Slumber
Life of Pie
A Brief History Of Primates
To Praise a Mocking Bird
The Catcher in the Eye
H2 Computing Ten Year Series

Try out the SQL statements yourself!

Do you know?

You can execute multiple SQL statements, but each statement must end with a colon ;

Exercise

2. Which of the following SQL statements show all data inside the Publisher table? (Tick the statements)

	`Select all FROM Publisher`
V	`Select * FROM Publisher`
	`Select ID, Title FROM Publisher`
	`Select ID, Title, PublisherID, Damaged FROM Publisher`

3. Write down the SQL statement to show all the Names of the publishers inside the Publisher table.

SELECT Name From Publisher

4. What does the following SQL statement do?
   `SELECT Title FROM Book WHERE PublisherID=1 AND Damaged=0`

   print all the book titles that are not damaged, and publisher id is 1

5. Write down the SQL statement to show all the Titles of the books which have PublisherID 1 or 2.

   SELECT Title FROM Book WHERE PublisherID = 1 OR PublisherID = 2

   SELECT Title FROM Book WHERE PublisherID in (1, 2)

Another example is to look at all the loans.

To list all loans, the SQL statement `SELECT * FROM Loan` is used.

ID	BorrowerID	BookID	Date Borrowed
1	3	2	20180220
2	3	1	20171215
3	2	3	20171231
4	1	5	20180111

The list can be ordered by the BookID (in **asc**ending order) instead by using the SQL statement `SELECT * FROM Loan ORDER BY BookID ASC`

The result is as follows.

ID	BorrowerID	BookID	Date Borrowed
2	3	1	20171215
1	3	2	20180220
3	2	3	20171231
4	1	5	20180111

The list can be ordered by the BookID (in **desc**ending order) by using the SQL statement `SELECT * FROM Loan ORDER BY BookID DESC`

The result is as follows.

ID	BorrowerID	BookID	Date Borrowed
4	1	5	20180111
3	2	3	20171231

1	3	2	20180220
2	3	1	20171215

What happens if the following SQL statement is executed?

```
SELECT * FROM Loan ORDER BY BookID
```

By default, it will order by BookID in ascending order.
Try out the SQL statements using DBrowser in SQLite.

Exercise

6. Write down the SQL statement to select all the records in Loan table arranged in ascending order of BookID with BorrowerID = 3.

SELECT * FROM Loan WHERE BorrowerID = 3 ORDER BY BookID

What if you want to find out the total number of loans?

You can use function COUNT to get the answer. The SQL can be the following:

```
SELECT COUNT(*) FROM Loan
```

-> start with what the question wants to retrieve

7. Write down the SQL statement to find all the book titles and their publishers' name that has been damaged (i.e. Damaged = 1).

SELECT Book.Title, Publisher.Name FROM Book INNER JOIN Publisher
ON Book.PublisherID = Publisher.ID WHERE Book.Damaged = 1

8. Write down the SQL statement to find all the book titles borrowed by Kumara and the dates that they were borrowed.

SELECT Book.Title, Loan."Date Borrowed" FROM Borrower INNER JOIN Loan ON Borrower.ID =Loan.BorrowerID
INNER JOIN Book ON Loan.BookID = Book.ID WHERE Borrower.FirstName = "Kumara"

9. Write down the SQL statements to print the names and the number of books they borrowed for all borrowers.

SELECT Borrower.FirstName, COUNT(Loan.BorrowerID) AS "Books Borrowed"
FROM Borrower
LEFT OUTER JOIN Loan ON Borrower.ID = Loan.BorrowerID
GROUP BY Loan.BorrowerID

SELECT FirstName, SurName , COUNT(Loan.Borrower.ID) AS "Books Loan"
From Borrower
Inner Join Loan ON Borrower.ID = Loan.Borrower.ID
Group BY Loan.BorrowerID

APPENDIX

Operators

You have seen some operators being used in the examples earlier. These operators are often used in SELECT statements, but can be used in other statements (like the UPDATE statement example shown earlier). The following are comparison operators, logical operators and arithmetic operators that you need to know.

A. Comparison Operators

Comparison Operator	Description
=	Checks if the values of two operands are equal or not, if yes then the condition becomes true.
!=	Checks if the values of two operands are equal or not, if the values are not equal, then the condition becomes true.
<>	Checks if the values of two operands are equal or not, if the values are not equal, then the condition becomes true.
>	Checks if the values of the left operand is greater than the value of the right operand, if yes then the condition becomes true.
<	Checks if the values of the left operand is less than the value of the right operand, if yes then the condition becomes true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes then the condition becomes true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes then the condition becomes true.

B. Logical Operators

Logical Operator	**Description**
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS	The value exists. For example, IS NULL looks for NULL values.
IS NOT	The value does not exist.
||	String concatenation

C. Arithmetic Operators

Arithmetic Operator	Name	Description
+	Addition	Adds values on either side of the operator
-	Subtraction	Subtracts the right-hand operand from the left-hand operand
*	Multiplication	Multiplies values on either side of the operator
/	Division	Divides the left-hand operand by the right-hand operand
%	Modulus	Divides the left-hand operand by the right-hand operand and returns the remainder

Functions (DISCUSSED IN NEXT LESSON)

Aggregate functions help to count / calculate results from the database.

Function	Description
MIN	Minimum value
MAX	Maximum value
SUM	Sum of all values
COUNT	Number of values

References

Content	Links
SQL	https://www.webucator.com/tutorial/learn-sql/simple-selects.cfm https://www.tutorialspoint.com/sql/sql-and-or-clauses.htm https://www.bbc.com/education/guides/z37tb9q/revision/5 https://www.w3schools.com/sql/default.asp