LAB 10

DATA MANIPULATION: INSERT, UPDATE, DELETE DATA

❖ **Main contents:** The statement updates data

- INSERT statement
- UPDATE statement
- DELETE statement

1. INSERT statement

The `INSERT` statement allows adding data rows to a specific table. There are two variants of the `INSERT` statement: the first way is to add a row of values, the second way is to add a set of rows returned from a `SELECT` statement.

Insert a value line

```
   INSERT INTO table_name
[(column_name,...)]
      VALUES((expression | DEFAULT),...),(...),...
```

`INSERT` creates a new row in the table <*table_name*>. The new line contains the values specified by the expressions in the `VALUES` list. If *column_name* is not included, the sequence of columns in the table <*table_name*> is used. If *column_name* is included, this way, new data is added to the table by specifying the column names and data for each column.

Example: Add a record to the *offices* table

```
INSERT INTO classicmodels.offices
    (officeCode,
    city,
    phone,
    addressLine1,
    addressLine2,
    state,
```

```
    country,
    postalCode,
    territory
    )
    VALUES
    ('8',
    'Boston',
    '+1 215 837 0825',
    '1550 dummy street',
    NULL,
    'MA,
    'USA',
    '02107',
    'NA'
    );
```

If the value is unknown, you can use the keyword NULL. Use the default value with the keyword DEFAULT.

The results of the above command can be verified using the query statement:

```
SELECT * FROM classicmodels.offices;
```

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
5	Tokyo	+81 33 224 5000	4-1 Kioicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA
8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

The result is a new row of data written to the end of the data table.

Note: If the column names are not specified, when the order of the columns changes, SQL may put the values in the wrong places. Therefore a good way to avoid this is to determine the column names associated with the data when adding data to the table.

Add multiple lines with the SELECT statement

Instead of providing data directly, it is possible to select from other tables using the `SELECT` statement.

```
INSERT INTO table_name
    [(column_name,...)]
    <SELECT statement>;
```

Unlike the previous way, this allows to create multiple data lines with. The list of result columns of the `SELECT` statement must match the list of columns of the table. Just like the previous way, the unspecified columns will be assigned implicit values of the column.

Example: Create a temporary table and add it to all offices in the US

```
INSERT INTO temp_table
SELECT *
FROM classicmodels.offices
WHERE country = 'USA';
```

The result of the above command can be verified using the query statement:

```
SELECT * FROM classicmodels.temp_table;
```

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

2. UPDATE statement

`UPDATE` statement is used to update the data that already exists in the database tables. The statement can be used to change the values of a row, a group of rows, or even all rows in a table.

Structure of the `UPDATE` statement:

```
UPDATE table_name [, table_name...]
 SET column_name1=expr1
     [, column_name2=expr2 ...]
```

```
[WHERE condition]
```

- After the `UPDATE` keyword, the name of the change table. The `SET` clause specifies the column to change and the value to change. The changed value can be a fixed value, an expression or even a subquery.
- The `WHERE` clause identifies the table rows to be updated. If the `WHERE` clause is omitted, all table rows will be updated.
- `WHERE` *clause is very important and should not be ignored. If we only want to change a row of a table, but forget the* `WHERE` *clause will update the entire table.*
- If an `UPDATE` statement violates any integrity constraints, MySQL will not perform an update and issue an error message.

Example: In the employees table, if we want to update Diane Murphy's email with employeeNumber is 1002 to diane-murphy@classicmodelcars.com, execute the following query:

```
SELECT firstname,
       lastname,
       email
FROM employees
WHERE employeeNumber = 1002;
```

	firstname	lastname	email
▶	Diane	Murphy	dmurphy@classicmodelcars.com

Result has updated the new email diane-murphy@classicmodelcars.com

```
UPDATE employees
SET email = 'diane-murphy@classicmodelcars.com'
WHERE employeeNumber = 1002;
```

Executing the `SELECT` query again, we will see the email changes to the new value

	firstname	lastname	email
▶	Diane	Murphy	diane-murphy @classicmodelcars.com

3. DELETE statement

To delete data rows of a database table, use the `DELETE` statement.

Structure of the `DELETE` statement:

```
 DELETE FROM table_name
[WHERE conditions]
```

- After `DELETE FROM` is the name of the table you want to delete records. `WHERE` clause specifies the condition to restrict the lines to be removed. If a record meets the `WHERE` condition, it is removed from the database table.
- If the `WHERE` clause is omitted from the `DELETE` statement, all table rows will be deleted. To reduce the dangers of statements like `DELETE` or `UPDATE`, *always check the* `WHERE` *condition in a* `SELECT` *T statement before executing the* `DELELE` *or* `UPDATE` *statement.*

Example: Delete all employees in the office with *officeNumber* code of 6, execute the following query:

```
DELETE
FROM employees
WHERE officeCode = 6;
```

Perform the query again on the `employee` table. There are no more rows in the table with `officeCode = 6`.

	employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
▶	1002	Murphy	Diane	x5800	diane-murphy @classicmodelcars.com	1	NULL	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
	1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep

Note: If you remove the WHERE condition

```
DELETE FROM employees;
```

Will delete all rows of the *employees* table. Therefore, it is important to note the condition in the `WHERE` clause when executing the `DELETE` statement.

MySQL also supports deleting records from many different tables.

Example: delete all employees who work for an office with *officecode 1* and also delete that office.

```
DELETE employees,offices
FROM employees,offices
WHERE employees.officeCode = offices.officeCode  AND
      offices.officeCode = 1;
```

After executing the above data deletion command, please check the data tables again
The *employees* table no longer has employee lines with *officeCode = 1*.

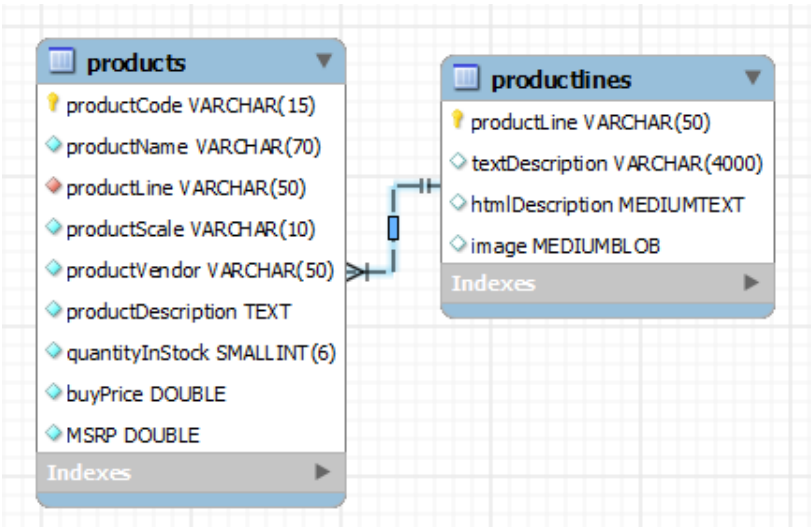
employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep

Offices table no longer has *officeCode = 1*.

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
5	Tokyo	+81 33 224 5000	4-1 Kioicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA
8	Boston	+1 215 837 0825	1550 dummy street	NULL	MA	USA	02107	NA

4. Update bound data

There may be constraints between data tables, for example, foreign key constraints between *products* and *productlines* tables.



If we delete a row of data in the *productlines* table and there are still rows in the *products* table that refer to this row, the default is not allowed.

Example: Delete product lines with the code ‘Ships’

```
DELETE FROM productlines
WHERE productLine='Ships';
```

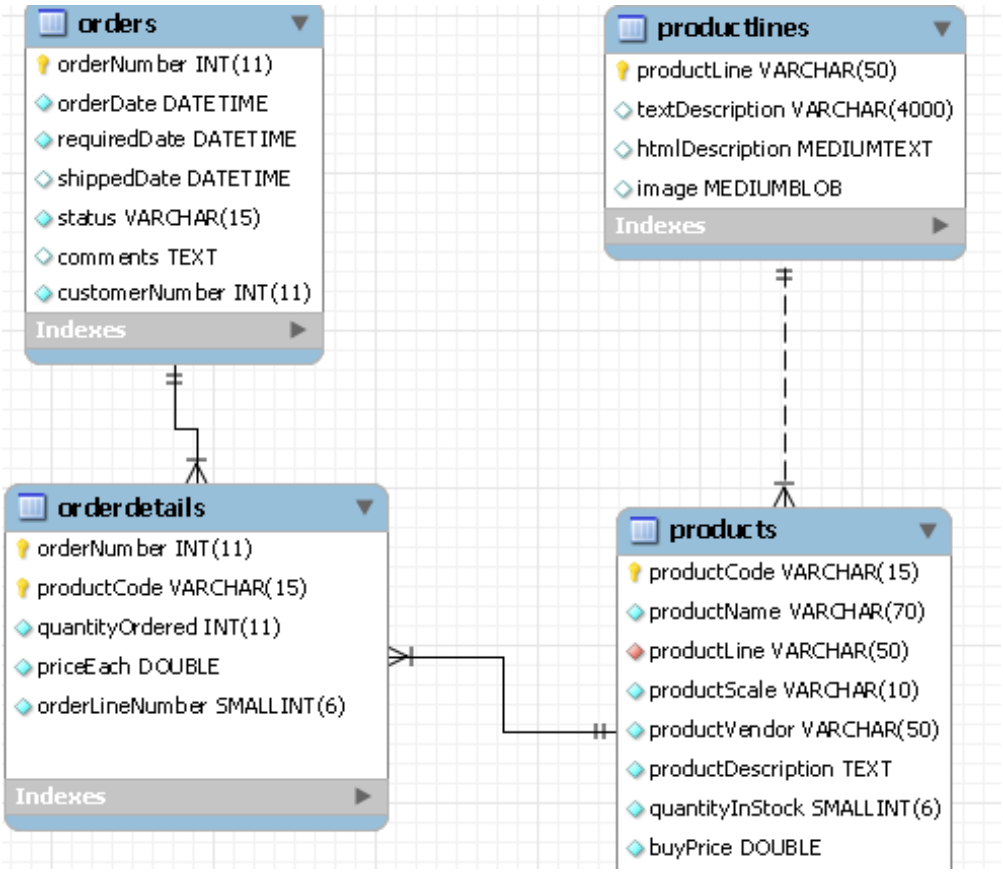
An error message "Cannot delete or update a parent row: a foreign key constraint fails (` classicmodels`.`products`, CONSTRAINT `fk_products_productlines` FOREIGN KEY (` productLine`) REFERENCES `productlines` (` productLine`) ON DELETE is displayed. NO ACTION ON UPDATE NO ACTION) ”

If the foreign key declaration is with the ON DELETE CASCADE option, the system will automatically delete the data rows in the products table that refer to this data stream.

If the foreign key is declared with the ON DELETE SET NULL option, the productLine foreign key of the reference rows will be set to NULL.

❖ Practical Exercises

1. Practice the INSERT, UPDATE and DELETE commands on the tables in the image below of the *classicmodels* database.



2. Create a table named *temp_orderdetails*, then add the data from the latest month in the *orderdetails* table to the table above.

3. Replace the entire productline's name 'Cars' with 'Automobiles'.