# SQL commands for data manipulation

INSERT – data insertion

UPDATE – data changing

DELETE – data deletion

SELECT – data selection

UPDATE, DELETE and SELECT works on all the table rows, which match the given conditions. If no conditions are passed to the command, then the changes will be applied to all the rows.

# INSERT

INSERT has 2 different syntax-es to offer

The long syntax, where all the inserted columns are defined

INSERT INTO table\_name (field1, field2, ..., fieldN)   
 VALUES (value1, value2, ..., valueN)

The short syntax, where you insert all of the columns

INSERT INTO table\_name  
 VALUES (value1, value2, ..., valueN)

When using the second syntax, the number of values should match the number of columns the table has. Value 1 will be put into Columns 1, Value 2 into Column 2, etc. It is not possible, under any syntax, to insert a value into IDENTITY column.

## Example:

INSERT INTO JOBS (job\_id, min\_level, max\_level) VALUES (1,1,8)

INSERT INTO JOBS VALUES (2,'Cleaner', 2,3)

*IDENTITY field can be omitted*

INSERT INTO employee VALUES ('Oto', 'Kalns', 2, 2, '05.05.2010')

INSERT INTO employee (fname, lname, job\_id, hire\_date) VALUES ('Oto', 'Kalns', 2, getdate())

There are two options on how to insert NULL (empty) value into the field. First is to pass NULL as a value in the insert query. Another option is to skip the column altogether.

INSERT INTO employee VALUES ('John', 'Doe', 2, NULL, '05.05.2010')

INSERT INTO employee (fname, lname, job\_id, hire\_date) VALUES ('John', 'Doe', 2, getdate())

Both inserts leave job\_level column blank. If the table definition allows for NULL values for specific column, then NULL value can be inserted into that column. The IDENTITY column cannot have NULL values, because it’s always defined in the schema as NOT NULL (cannot be empty).

# DELETE

DELETE FROM <table\_name>  
 WHERE <condition>

The command removes all the rows from table, which match the condition. If no condition is passed, then all the rows are removed.

DELETE FROM employee

WHERE fname = 'Oto'

This query removes all records from the table, which have fname = ‘Oto’

DELETE FROM employee

The query removes all the records from the table

DELETE FROM employee

WHERE job\_level>100

This query removes all records from the table, which have job\_level greater than 100

DELETE FROM employee WHERE job\_level IN (70, 82, 93)

This query removes all the records from the table, which have job\_level 70, 82 or 93.

DELETE FROM employee

WHERE job\_level IS NULL

Remove records from the table, which have no job\_level specified

# UPDATE

UPDATE <table\_name>   
 SET columns1 = value1, ..., columnN= valueN  
 WHERE condition

Command changes all the rows, which match the condition. The rows are change according to SET statement, only specified column values are changed.

UPDATE employee

SET job\_level=24

WHERE emp\_id=1

For employee with emp\_id 1 change job\_level to 24

UPDATE employee

SET job\_level=4

WHERE job\_level=3

Update all employees with job\_level 3 to job\_level 4

When no condition is passed, all the rows are changed

UPDATE employee

SET job\_level=job\_level+1  
Increase job\_level for all of the employees by 1

To select all the rows from the table, use SELECT statement:

SELECT \* FROM jobs

UPDATE jobs SET min\_level=min\_level+5

WHERE job\_id IN (SELECT job\_id FROM employee WHERE fname='Oto')

For all the jobs, which are performed by employee named Oto, change min\_level to 5