SQL statements for one table

This section contains SQL statements, that are applied for one table. Consider the following situation: We have one table with name table_name. This table has n, n natural number, columns with names column_1, column_2, … , column_n. Every column has m, m natural number, values. The i-th column, $1 \leq i \leq n$, has the values $v_{1, i}, v_{2, i}, \dots , v_{m, i}$.

Selecting columns

```
/* Selecting the first column, with every value in it. */
SELECT colum_1
FROM table_name;
```

```
/* Selecting the i-th column, 1 ≤ i ≤ n, with every value in it. */
SELECT colum_i
FROM table_name;
```

```
/* Selecting the last column with every value in it. */
SELECT colum_n
FROM table_name;
```

```
/* Selecting the first and the second column with every value in it. */
SELECT colum_1,  colum_2
FROM table_name;
```

```
/* Selecting the i-th and j-th column, 1 ≤ i, j ≤ n and i ≠ j,  with every value in it. */
SELECT colum_i,  colum_j
FROM table_name;
```

```
/* Selecting the i_{1}-th, i_{2}-th, ... , i_{j}-th column, j natural number, 1 ≤ i_{1}, i_{2}, … i_{j} ≤ n and  i_{1} ≠ i_{2} ≠ …  ≠ i_{j}. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
```


Document written by Artur Sanin

```
FROM table_name;


/* Selecting every column, with every value in it. */
SELECT column_1, column_2, … , column_n
FROM table_name;
or
/* Selecting every column, with every value in it. */
SELECT *
FROM table_name;
```

Filtering records

This section contains general formulated statements for filtering records and fields from a table. We use the WEHRE keyword with a specified condition, that the records should meet. Conditions can be combined with the AND, OR, NOT keywords.

```
/* Selecting only records in table_name where column_i, 1 ≤ i ≤ n, has value v_{j, i}, 1 ≤ j ≤ m.*/
SELECT *
FROM table_name
WHERE column_i = v_{j, i};
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, or column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2}, i_{2}}, 1 ≤ j_{2} ≤ m. */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) OR (column_i_{2} = v_{j_{2}, i_{2}});
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, and column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2}, i_{2}}, 1 ≤ j_{2} ≤ m. */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) AND (column_i_{2} = v_{j_{2}, i_{2}});
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, or column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2}, i_{2}}, 1 ≤ j_{2} ≤ m or … or column_i_{p}, 1 ≤ i_{p} ≤ n, has value v_{j_{p}, i_{p}}, 1 ≤ j_{p} ≤ m, */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) OR (column_i_{2} = v_{j_{2}, i_{2}}) OR …
OR (column_i_{p} = v_{j_{p}, i_{p}});
```

```
/* Selecting only records in table_name where column_i_{1}, 1 ≤ i_{1} ≤ n, has value
v_{j_{1}, i_{1}}, 1 ≤ j_{1} ≤ m, and column_i_{2}, 1 ≤ i_{2} ≤ n, has value v_{j_{2},
i_{2}}, 1 ≤ j_{2} ≤ m and … and column_i_{p}, 1 ≤ i_{p} ≤ n, has value v_{j_{p}, i_{p}},
1 ≤ j_{p} ≤ m, */
SELECT *
FROM table_name
WHERE (column_i_{1} = v_{j_{1}, i_{1}}) AND (column_i_{2} = v_{j_{2}, i_{2}}) AND
… AND (column_i_{p} = v_{j_{p}, i_{p}});


/* Selecting all records in table_name. 1 ≤ i ≤ n is arbitrary. */
SELECT *
FROM table_name
WHERE (column_i = v_{1, i}) OR (column_i = v_{2, i}) OR … OR (column_i = v_{m, i});


/* General syntax for filtering records. */
SELECT *
FROM table_name
WHERE condition;
        or
/* General syntax for filtering records, j some natural number AND/OR = AND or OR. */
SELECT *
FROM table_name
WHERE condition_1 AND/OR condition_2 AND/OR … AND/OR condition_j;


/* Filtering all records, that do not satisfy the condition. */
SELECT *
FROM table_name
WHERE NOT condition ;


/* General syntax for filtering fields form specific columns. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
WHERE condition;
        or
/* General syntax for filtering fields, p some natural number AND/OR = AND or OR. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
```

```
WHERE condition_1 AND/OR condition_2 AND/OR … AND/OR condition_p;


/* Filtering fields, that do not satisfy a condition. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{j}
FROM table_name
WHERE NOT condition;
```

Ordering results

To sort your query results, use the ORDER BY keyword. Note that the order is ascending by default. If the column, by which you want the results ordered, is numerical, then you get the lowest number first ascending to the highest. If the column, by which you want the results ordered, is a string, then you get an alphabetical order.

Note if ordered by several columns, then your results will be sorted by the first specified column first, if the same value accurse multiple times, then those will be sorted by the next column and so on.

```
/* Selecting all records in table_name ordered by column_i, 1 ≤ i ≤ n, ascending. */
SELECT *
FROM table_name
ORDER BY column_i ASC;
        or
/* Selecting all records in table_name ordered by column_i, 1 ≤ i ≤ n, ascending. */
SELECT *
FROM table_name
ORDER BY column_i;


/* Selecting all records in table_name ordered by column_i, 1 ≤ i ≤ n, descending. */
SELECT *
FROM table_name
ORDER BY column_i DESC;
```

```
/* Selecting all records in table_name ordered by column_i_{1} ascending or descending,
column_i_{2} ascending or descending, … column_i_{p} ascending or descending , 1 ≤
i_{1},  i_{2}, … , i_{p} ≤ n. */
SELECT *
FROM table_name
ORDER BY column_i_{1} ASC/DESC, column_i_{2} ASC/DESC, … , column_i_{p}
ASC/DESC;
```

```
/* Selecting columns column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2}, … ,
i_{p} ≤ n and p natural number, in table_name ordered by column_i, 1 ≤ i ≤ n, ascending. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{p}
FROM table_name
ORDER BY column_i;
```

Note that column_i can be one of the columns column_i_{1}, column_i_{2}, … , column_i_{p}.

```
/* Selecting columns column_i_{1}, column_i_{2}, … , column_i_{p}, 1 ≤ i_{1}, i_{2}, … ,
i_{p} ≤ n and p natural number, in table_name ordered by column_i, 1 ≤ i ≤ n, descending. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{p}
FROM table_name
ORDER BY column_i DESC;
```

```
/* Selecting columns column_i_{1}, column_i_{2}, … , column_i_{q}, 1 ≤ i_{1}, i_{2}, … ,
i_{q} ≤ n and q natural number, in table_name ordered by column_j_{1} ascending or
descending, column_j_{2} ascending or descending, … column_j_{p} ascending or
descending , 1 ≤ j_{1},  j_{2}, … , j_{p} ≤ n. */
SELECT column_i_{1}, column_i_{2}, … , column_i_{q}
FROM table_name
ORDER BY column_j_{1} ASC/DESC, column_j_{2} ASC/DESC, … , column_j_{p}
ASC/DESC;
```

Note that column_j_{1}, column_j_{2}, …. , column_j_{p} can be one of the columns column_i_{1}, column_i_{2}, …. , column_i_{q}.

Databases