Database solutions

Selected SQL commands – part 1

Marzena Nowakowska
Faculty of Management and Computer Modelling
Kielce University of Technology
room: 3.21 C

SQL syntax

A syntax is like template which tells what is required (permitted) for a given command.

Conventions used for the SQL syntax:

- SQL keywords are in uppercase letters, although in practice they can be in any case,
- words or phrases which should be supplied are in lowercase letters – they are placeholders for values (constants, expressions, identifiers – names of databases, tables, or other database objects),
- curly brackets { } around a list of words or phrases separated by vertical bars means that one of these options must be taken to the command,
- square brackets [ ] indicate optional occurrence of words or phrases.

Each SQL command ends with a semicolon (;).

Fundamental ideas

DML – Data Manipulation Language

Data manipulation is retrieving and modifying data through SQL commands:

- SELECT
- INSERT
- DELETE
- UPDATE

DDL – Data Definition Language

Data definition operates on data structure:

- CREATE
- DROP
- ALTER

DCL – Data Control Language

Data controlling is used to control access to data stored in a database:

- GRANT
- REVOKE
- DENY

SELECT command

SELECT [*modifier*]
{ * | *table*.* | [*table*.]*field1* [**AS** *alias1*] [, [*table*.]*field2* [**AS** *alias2*] [, ...]]}
FROM *table_expression* [, ...]
[**WHERE** *condition*]
[**GROUP BY** *field_list*]
[**HAVING** *group_condition*]
[**ORDER BY** *sort_field1* [**ASC | DESC**][**,** *sort_field2* [**ASC | DESC**][, ...]]];

The command retrieves data from a table or from multiple tables.

SELECT command description

modifier = {ALL | DISTINCT} (ALL is a default value)

specification what to do with duplicates in the resulting records

table name of a table from which records are retrieved; * defines all fields from a given table; ALL is a default value

field either a table field name or a calculated field the value of which is to be presented; calculated fields are expressions created with the use of field names, constants, operators, and functions

alias an alternative name for a field

table_expression

name of a table, a named query, or a result of a join operation (INNER JOIN, LEFT JOIN, RIGHT JOIN)

condition criterion imposed on retrieved records

field_list list of fields separated by comas, by which records are grouped

group_condition

criterion imposed on a group (used with GROUP BY clause)

sort_field either a table field or a calculated field according to which records are sorted

SELECT command – modifier

```
SELECT Book.Title, Author
FROM Book;

SELECT  ALL Book.Title, Author
FROM Book;
```

the same

Result: 25 records

Title	Author
Bazy danych	Jasiński Paweł
Bazy danych	Jasiński Paweł
Bazy danych	Jasiński Paweł
Bazy danych	Jasiński Paweł
Big Data	Manilla John
Encyklopedia of computer scier	Ralston Anthony
English business letters	Kaufmann Walter
Financial MANAGEMENT	Schall Diana
Finite MATHEMATICS	Thompson Norman
Jak wygrać na polskiej giełdzie	Werner Jack
Longman lexicon of contem. En	Mc Arthur Thomas
Meta-analysis of accident data	Elvik Rune
MsWorks 3.0 i 3.0 PL	Widuch Tadeusz
Podatek VAT i akcyzowy	Woźniak Karol
Praca z arkuszem kalkulacyjnym	Praca zbiorwoa
Programowanie w systemie UN	Rochkind Mark
Quality assessment	Elvek Rune
Road safety analysis	Abdel-Aty Mohamed
Safety preformace function	Abdel-Aty Mohamed
Słownik Angielsko-Polski	Stanisławski Jan
Turbo pascal	Marciniak Andrzej
Wartość pieniądza w czasie	Wieczorek Daniel
Wartość pieniądza w czasie	Wieczorek Daniel
Wartość pieniądza w czasie	Wieczorek Daniel
Wartość pieniądza w czasie	Wieczorek Daniel

SELECT command – modifier

```
SELECT DISTINCT Book.Title,
Author
FROM Book;
```

Result: 19 records. Tere are more than one volume of the same title and the same author but only one is shown

Title	Author
Bazy danych	Jasiński Paweł
Big Data	Manilla John
Encyklopedia of computer sc	Ralston Anthony
English business letters	Kaufmann Walter
Financial MANAGEMENT	Schall Diana
Finite MATHEMATICS	Thompson Norman
Jak wygrać na polskiej giełdz	Werner Jack
Longman lexicon of contem.	Mc Arthur Thomas
Meta-analysis of accident da	Elvik Rune
MsWorks 3.0 i 3.0 PL	Widuch Tadeusz
Podatek VAT i akcyzowy	Woźniak Karol
Praca z arkuszem kalkulacyjn	Praca zbiorwoa
Programowanie w systemie	Rochkind Mark
Quality assessment	Elvek Rune
Road safety analysis	Abdel-Aty Mohamed
Safety preformace function	Abdel-Aty Mohamed
Słownik Angielsko-Polski	Stanisławski Jan
Turbo pascal	Marciniak Andrzej
Wartość pieniądza w czasie	Wieczorek Daniel

SELECT command - table_expression

The term *table_expression* defines a recordset that is selected form the dabatase:

- a single table
- multiple tables (sometimes that have very little in common); JOINs are relational operators that combine data from multiple table into a single result table. The source tables are joined in the sense that the resulting table includes information taken from all the source tables.

Consider the following tables

Location

Location_ID	City
1	Boston
3	Miami
5	Chicago

Department

Dept_ID	Location_ID	Dept_name
21	1	Sales
24	1	Admin
27	5	Repair
29	5	Stock

Employee

Emp_ID	Dept_ID	Empl_name
61	24	Kirk
63	27	McCoy

Basic JOIN

SELECT Department.*, Employee.* FROM Department, Employee;

Department.Dept_ID	Location_ID	Dept_name	Emp_ID	Employee.Dept_ID	Empl_name
21	1	Sales	61	24	Kirk
21	1	Sales	63	27	McCoy
24	1	Admin	61	24	Kirk
24	1	Admin	63	27	McCoy
27	5	Repair	61	24	Kirk
27	5	Repair	63	27	McCoy
29	5	Stock	61	24	Kirk
29	5	Stock	63	27	McCoy

The result table is the Cartesian product of the *Employee* and the *Department* tables; it combines every row of *Employee* with every row of *Department.*

Not only it contains considerable redundancy but also it doesn't make much sense.

Those rows in which the department identifier from the *Department* table is equal to department identifier from the *Employee* table indicate a correct combination of information.

Equi-join

This is a basic join with a WHERE clause containing a condition specifying that the value in one column in the first table must be equal to the value of a corresponding column in the second table.

```
SELECT Department.*, Employee.*
FROM Department, Employee
WHERE Department.Dept_ID = Employee.Dept_ID;
```

The version with aliases (attention: depending on a DBMS the table aliases can be defined in a different way) is as follows:

```
SELECT D.*, E.*
FROM Department AS D, Employee AS E
WHERE D.Dept_ID = E.Dept_ID;
```

Department.Dept_ID	Location_ID	Dept_name	Emp_ID	Employee.Dept_ID	Empl_name
24	1	Admin	61	24	Kirk
27	5	Repair	63	27	McCoy

INNER JOIN

The inner join is a special case of an equi-join. A column from one source table is compared with a column of a second source table for equality. The two columns must be the same type.
An inner join discards all rows form the result table that don't have corresponding rows in both source tables. The common fields (responsible for the join) can have different names.

```
SELECT Department_a.*, Employee.*
FROM Department_a INNER JOIN Employee ON Department_a.ID =
Employee.Dept_ID;
```

ID	Location_ID	Dept_name	Emp_ID	Dept_ID	Empl_name
24	1	Admin	61	24	Kirk
27	5	Repair	63	27	McCoy

Outer JOIN

When an inner join is performed on tables, all unmatched rows are excluded from the output. Outer joins, however, do not exclude the unmatched rows.

In a query that includes the JOIN operator, the left table is the one that precedes the keyword JOIN and the right table is the one that follows the keyword JOIN.

The left table may have rows that don't have matching counterparts in the right table. Conversely, the table on the right may have rows that don't have matching counterparts in the left table,

Consider the following tables

Department

Dept_ID	Location_ID	Dept_name
21	1	Sales
24	1	Admin
27	5	Repair
29	5	Stock

AdvAgency

Agency_ID	Agency_name
1	Fergusson
2	Frost-BT
3	Toryon
4	Gyrosign
5	Digit Design

Dep-Advert

Agency_ID	Dept_ID
1	21
4	24
2	29

LEFT (outer) JOIN

The left join preserves unmatched rows from the left table but discards unmatched rows from the right table.

```
SELECT Department.Dept_ID, Department.Dept_name, [Dep-Advert].Dept_ID, AdvAgency.Agency_name, AdvAgency.Agency_ID
FROM
        (Department LEFT JOIN [Dep-Advert]
                ON Department.Dept_ID = [Dep-Advert].Dept_ID)
LEFT JOIN AdvAgency
        ON [Dep-Advert].Agency_ID = AdvAgency.Agency_ID;
```

Department.Dept_ID	Dept_name	Dep-Advert.Dept_ID	Agency_name	Agency_ID
21	Sales	21	Fergusson	1
24	Admin	24	Gyrosign	4
27	Repair			
29	Stock	29	Frost-BT	2

All departments are included, even if no advertisement was ordered by them. For those (non-advert) departments respective fields have null values.

RIGHT(outer) JOIN

The right join preserves unmatched rows from the right table but discards unmatched rows from the left table.

```
SELECT Department.Dept_ID, Department.Dept_name,
[Dep-Advert].Agency_ID, AdvAgency.Agency_name, AdvAgency.Agency_ID
FROM
        (Department RIGHT JOIN [Dep-Advert]
                ON Department.Dept_ID = [Dep-Advert].Dept_ID)
RIGHT JOIN AdvAgency
        ON [Dep-Advert].Agency_ID = AdvAgency.Agency_ID;
```

Dept_ID	Dept_name	Dep-Advert.Agency_ID	Agency_name	AdvAgency.Agency_ID
21	Sales	1	Fergusson	1
29	Stock	2	Frost-BT	2
			Toryon	3
24	Admin	4	Gyrosign	4
			Digit Design	5

All advert agencies are included, even if some of them were not ordered advertisements form the departments. For those (non-ordered) agencies respective fields have null values.

WHERE clause

The clause defines criterion (condition) which have to be satisfied by records. The criterion is a logical expression that can consists of expression of different types, such as text, date, and numerical expression. The following operators can be used to create an expression:

- comparison operators: =, >, >=, <, <=, <> (*not equal to* symbol may depend on a DBMS)
- logical operators: AND (conjunction), OR (alternative), NOT (negation),
- arithmetical operators: + (addition), - (subtraction), * (multiplication), / , \ (division) : 5/2 -> 2.5 whereas 5\2 -> 2
- BETWEEN operator enables to define a range of values in the condition
- IN operator enables to define a list of values specified in the condition
- IS NULL operator enables to indicate null values
- LIKE operator enables to define a patterns of characters; the operator is used with wildcards: * (or %) represents any collection of characters, _ (or ?) represents any single character

WHERE clause examples

Selection of values within a specified range:

```
SELECT * FROM Foods WHERE Calories >99 AND Calories<301;
SELECT * FROM Foods WHERE Calories >=100 AND Calories <=300;
SELECT * FROM Foods WHERE Calories BETWEEN 100 AND 300;
```

} the same

Provided that Calories are integer numbers

Selection of values within a specified list:

```
SELECT Title, Author, Editor FROM Book WHERE Price IN (100, 50, 150);
SELECT Company, Phone FROM Supplier
        WHERE state NOT IN ('CA', 'AZ', 'NM');
```

Selection of values as regards a specified string of characters:

```
SELECT Article, Journal FROM Publications WHERE Abstract LIKE 'intern%';
SELECT * form Customers WHERE Phone  NOT LIKE '503%';
```

Selection of values with the criterion imposed on a calculated field:

```
SELECTK Book.Author, Book.Title, Year(Date())-Book.Edit_year AS Book_age
FROM Book WHERE Year(Date())-Book.Edit_year  > 10;
```

Homework: think out a database table for the illustration of above examples.

ORDER BYclause

The clause is used to display result records in an ascending (ASC) or a descending (DESC) order. Ascending is the default order. There can be more than one field in the order list.

ASC: 1,2,3,…., 100 DESC: 'zero', 'z', 'x', … , 'bogdan', 'beata', 'alabama', 'ala'

Examples

Records are sorted first by date of sale, than for each date, the records are ordered by invoice number. A default order type is ascending.

```
SELECT * FROM Sales ORDER BY Sale_date, Invoice_no;
SELECT * FROM Sales ORDER BY Sale_date ASC, Invoice_no ASC;
```

} the same

Records are sorted first by date of sale discerningly, than for each date, the records are ordered by invoice number (in an ascending way). A default order type is ascending.

```
SELECT * FROM Sales ORDER BY Sale_date DESC, Invoice_no;
```

Homework: think out a database table for the illustration of above examples.

Aggregation

The aggregation operation can be used to all rows in a table expression, to the rows specified by a WHERE clause, or to groups of rows set up by the GROUP BY clause.
The aggregation functions (SQL aggregates):

SUM(expression)	The total sum of values in the numeric expression
AVG(expression)	The average of values in the numeric expression
COUNT(expression)	The number of non-null values in the expression
COUNT(*)	The number of selected rows.
MAX(expression)	The highest (greatest) value in the expression
MIN(expression)	The lowest (smallest) value in the expression

The argument - expression is often a field name, but it can also be a constant, a function call, or any combination of field names, constants, and function calls - all connected by appropriate operators.

Article	Price	PriceCut	PriceAfterCut
AAA	100	0.1	90
BBB	50	0.1	45
CCC	150	0.2	120
	200	0.2	160
EEE	80	0.1	72
FFF	75	0.1	67.5

Count(Article) → 5
Count(Price) → 6
Sum(Price) → 655
Min(PriceCut) →0.1
Max(PriceAfterCut) →160

Aggregation

ID	Title	Price	Edit_year
0001	Wartość pieniądza w czasie	14.00 zł	1983
0003	Programowanie w systemie UN	7.00 zł	1993
0004	Financial MANAGEMENT	114.60 zł	1974
0005	Turbo pascal	22.50 zł	1993
0006	Finite MATHEMATICS	16.32 zł	1983
0007	Bazy danych	99.50 zł	1997
0011	Wartość pieniądza w czasie	14.00 zł	1993
0013	Słownik Angielsko-Polski	48.00 zł	1985
0015	Bazy danych	99.50 zł	1992
0016	Wartość pieniądza w czasie	14.00 zł	1993
0017	Jak wygrać na polskiej giełdzie	7.80 zł	1993
0019	Podatek VAT i akcyzowy	7.30 zł	1993
0022	Encyklopedia of computer scie	160.70 zł	1993
0031	Longman lexicon of contem. En	144.00 zł	1988
0032	Wartość pieniądza w czasie	89.90 zł	1994
0099	English business letters	10.56 zł	1986
0100	Praca z arkuszem kalkulacyjnyn	130.00 zł	1993
0102	MsWorks 3.0 i 3.0 PL	14.10 zł	1994
0105	Bazy danych	99.50 zł	1997
0106	Bazy danych	99.50 zł	2012
0107	Big Data	55.00 zł	2014
0108	Road safety analysis	33.00 zł	2015
0109	Safety preformace function	50.00 zł	2015
0110	Meta-analysis of accident data	100.00 zł	2014
0111	Quality assessment	60.00 zł	2013

← Before aggregation

Example aggregation

```
SELECT Count(Book.ID) AS Books,
Avg(Book.Price) AS AvgPrice,
Min(Book.Price) AS MinimumPrice,
Max(Book.Price) AS MaksimumPrice,
Min(Year(Date())-Book.Edit_year ) AS
AgeYoungest,
Max(Year(Date())-Book.Edit_year) AS
AgeOldest,
FROM Book;
```

After aggregation ↓

Assumed current year: 2020

Books	AvgPric	MinimumPric	MaksimumPrice	AgeYounges	AgeOldest
25	60.43 zł	7.00 zł	160.70 zł	5	46

GROUP BY clause

The clause is intimately connected to aggregates. The clause divides a set of records into subsets, while aggregate functions produce summary values for each subset. There can be more than one GROUP BY clause in a query.

Example

```
SELECT Book.Type, Count(Book.ID) AS Books,
Avg(Book.Price) AS AvgPrice,
Min(Book.Price) AS MinimumPrice,
Max(Book.Price) AS MaksimumPrice,
Min(Year(Date())-Book.Edit_year)  AS AgeYoungest,
Max(Year(Date())-Book.Edit_year) AS AgeOldest,
FROM Book
GROUP BY Book.Type;
```

Type	Books	AvgPrice	MinimumPrice	MaksimumPrice	AgeYounges	AgeOldest
Handbook	16	55.16 zł	7.00 zł	130.00 zł	5	46
Lexicon	3	89.57 zł	48.00 zł	160.70 zł	7	35
User guide	6	59.93 zł	7.30 zł	144.00 zł	6	34

WHERE clause in aggregation queries

WHERE clause used in an aggregation query (either with or without grouping) enables to retrieve records that satisfy the criterion in the WHERE condition first, then the aggregation is done.

Examples

```
SELECT Count(Book.ID) AS Books, Avg(Book.Price) AS AvgPrice,
Min(Book.Price) AS MinimumPrice, Max(Book.Price) AS MaksimumPrice,
Min(Year(Date())-Book.Edit_year) AS AgeYoungest,
Max(Year(Date())-Book.Edit_year) AS AgeOldest,
FROM Book WHERE Book.Type<>"Handbook“;
```

Books	AvgPrice	MinimumPrice	MaksimumPrice	AgeYoungest	AgeOldest
9	69.81 zł	7.30 zł	160.70 zł	6	35

```
SELECT Book.Type, Count(Book.ID) AS Books, Avg(Book.Price) AS AvgPrice,
Min(Book.Price) AS MinimumPrice, Max(Book.Price) AS MaksimumPrice,
Min(Year(Date())-Book.Edit_year) AS AgeYoungest,
Max(Year(Date())-Book.Edit_year) AS AgeOldest,
FROM Book WHERE Book.Type<>"Handbook"
GROUP BY Book.Type;
```

Type	Books	AvgPrice	MinimumPrice	MaksimumPrice	AgeYoungest	AgeOldest
Lexicon	3	89.57 zł	48.00 zł	160.70 zł	7	35
User guide	6	59.93 zł	7.30 zł	144.00 zł	6	34

HAVING clause in queries

The HAVING clause is a criterion used for an aggregate function applied to groups, i.e. for each group an aggregate is calculated, and then a criterion on the aggregate result is imposed. Then the records for which the criterion is satisfied are selected.

Example

```
SELECT Book.Type, Count(Book.ID) AS Books,
Avg(Book.Price) AS AvgPrice,
Min(Book.Price) AS MinimumPrice,
Max(Book.Price) AS MaksimumPrice,
Min(Year(Date())-Book.Edit)_year AS AgeYoungest,
Max(Year(Date())-Book.Edit)_year AS AgeOldest,
FROM Book
GROUP BY Book.Type;
HAVING Avg(Book.Price) > 60;
```

Type	Books	AvgPrice	MinimumPrice	MaksimumPrice	AgeYoungest	AgeOldest
Lexicon	3	89.57 zł	48.00 zł	160.70 zł	7	35

Nested query

It is possible to create a nested query, i.e. to place one query inside another. The inner query is called a subquery and it is evaluated first. Then the outer query can use the results of the subquery to find its results.
The subquery (is usually a part of the WHERE clause) and can be applied in various ways.

Examples

```
SELECT Customer_number, Last_N, First_N, Town FROM Customer
WHERE Credit_limit IN (SELECT DIST Credit_limit FROM Customer WHERE
Town = ‘Kielce’) and Town <> ‘Kielce’;

SELECT Customer_number, Last_N, First_N, Town FROM Customer
WHERE Credit_limit = (SELECT Max(Credit_limit) FROM Customer);

SELECT Author,  Title, Price FROM Book
WHERE Price > (SELECT  AVG(Price) from Book);
```

Homework: think out a database table for the illustration of above examples.