SELECT statement

Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ (c) 2011 - 2016

Introduction

- Most important SQL statement
- Retrieves (filters) data from DB
  - Selected rows (`WHERE`)
  - Selected columns (`SELECT`)
- syntax:

```
SELECT    what
FROM      from_where
WHERE     what_we_want
ORDER BY  ordering_column(s)
DESC;
```

Examples of SELECT

```
SELECT * FROM student;

SELECT name, surname
FROM   student;

SELECT    name, surname
FROM      student
ORDER BY surname, name;

SELECT name, surname
FROM   student
WHERE  name = 'John';
```

SELECT DISTINCT

- Table in RDMS is a multiset (bag) - a set that allows multiple same tuples (duplicates)
  - Math relation in relational model is a set
- `SELECT DISTINCT` - removes duplicates
- example:

```
SELECT DISTINCT name
FROM student;
```

Basic operators

- Relational operators
  `=`, `<`, `>`, `<>`, `!=`, `<=`, `>=`
- Arithmetic operators
  `+`, `-`, `*`, `/`, `MOD(a,b)`
- `||` - string concatenation
- `IS [NOT] NULL` - testing to `NULL`
- Expression can be:
  - In WHERE clause to specify selection condition
  - In list of columns projection after SELECT clause
  - In ORDER BY clause

Special `DUAL` table

- Single-row and single-column table usable for queries that does not need `FROM` clause
  - Oracle specific (Oracle requires using `FROM`)
- example:

```
SELECT 1+1 FROM DUAL;
```

- instead of

```
SELECT 1+1;
```

Basic functions I.

- **UPPER**('Ahoj') - conversion to uppercase
- **LOWER**('Ahoj') - conversion to lowercase
- **SUBSTR**('Ahoj', 2, 2) - substring
- **TRIM**(' ahoj ') - trimming whitespaces
- **INITCAP**('milan NOSAL') - initial capital letters
- **REPLACE**('Milan','lan','chal') - replacing substring
- **LENGTH**('Ahoj') - length of string
- **NVL**(to_char(birthday), 'neznáme') - NULL value substitution

Basic functions II.

- **`CURRENT_DATE`**
- **`CURRENT_DATE+1`** - following day
- **`CURRENT_TIMESTAMP`**
- **`MONTHS_BETWEEN(date1, date2)`** - number of months between `date1` and `date2`
- **`EXTRACT`**`(month FROM CURRENT_DATE)`
- **`TO_CHAR`**`(CURRENT_DATE, 'DD.MM.YYYY')` - converts `DATETIME` to string (opposite direction with `TO_DATE`)

Basic functions III.

- **`NEXT_DAY('26-7-1996', 'SUNDAY')`** - closest next sunday from the given date
- **`ABS`**`(-5)` - absolute value
- **`CEIL`**`(15.7)`/**`FLOOR`**`(15.7)` - closest bigger/smaller integer
- **`MOD`**`(123,5)` - modulo
- **`ROUND`**`(15.7)` - rounding
- **`SIGN`**`(-5)` - returns sign of the value
- **`TRUNC`**`(165.54, 1)` - truncating decimal places
- **`GREATEST`**`(1,5,7,33)` - biggest item
- **`LEAST`**`(1,5,7,33)` - smallest item

Basic logic operators

- **AND**

```
name='Johnny' AND
surname='Hrasko'
```

- **OR**

```
course='SQL' OR course='DBS'
course IN ('SQL', 'DBS')
```

- **NOT** - negation

```
course NOT IN ('SQL', 'DBS')
```

`LIKE` operator

- **`LIKE`** - serves for basic pattern matching
  - `%` - 0 to N arbitrary characters
  - `_` - a single arbitrary character
- syntax:

```
SELECT surname, name
FROM student
WHERE surname LIKE '%mith'
    AND name LIKE '_ill'
```

BETWEEN operator

- **BETWEEN** - defines an inclusive range of values or a time range (dates)
- syntax:

```
SELECT surname, name
FROM student
WHERE class BETWEEN 1 AND 5;

SELECT * FROM invoice
WHERE dateOfInvoice NOT BETWEEN
'01.01.2013' AND '31.12.2013';
```

Sorting of results

- Sorting by values of attributes
  - Ascending order (`ASC` - default)
  - Descending order (`DESC`)
- examples:

```
SELECT surname, name
FROM student
ORDER BY surname;

SELECT surname, name
FROM student
ORDER BY surname, name DESC;
```

NULL value

- Special value representing unknown or undefined value
- Specific behavior when used in WHERE clause, aggregate functions, etc.
- SQL uses 3-valued logic
- F - false, T - true, U - unknown

NOT(A)

A	¬A
F	T
U	U
T	F

AND(A, B)

A ∧ B		B		
		F	U	T
A	F	F	F	F
	U	F	U	U
	T	F	U	T

OR(A, B)

A ∨ B		B		
		F	U	T
A	F	F	U	T
	U	U	U	T
	T	T	T	T

Aliases

- Allow to temporarily rename a column (with AS keyword) or a table (without special keyword, sometimes called table variables)
  - Shorter notation (readibility)
  - Enables to differentiate columns with the same names (in joining, subqueries, ...)
  - “Nice” name of a column
- example:

```
SELECT RPAD(f.surname,10) || ' '
|| f.name AS meno FROM fbuser f;
```

Pseudocolumn `ROWNUM`

- Number of the row in the result (`LIMIT` in other implementations)
- The value is assigned after filtering by the WHERE clause, but before aggregation or ordering (sorting)
  - In such a case you need to use subquery
- You can limit number of result tuples to a specific number:

```
SELECT * FROM student
WHERE ROWNUM < 11;
```

Creating view from `SELECT` result

- View can be considered a named `SELECT` stored in the database
  - virtual (derived) table
- syntax:

```
CREATE VIEW view_name AS
    SELECT columns
    FROM tables
    WHERE conditions;
```

Creating new table from `SELECT` result

- A real table that supports all DML statements
- Modifications to the new table will not affect the original table
- syntax

```
CREATE TABLE table_name AS
    SELECT columns
    FROM tables
    WHERE conditions;
```

INSERT with SELECT

- Inserts select result to the table (SELECT result has to have the same schema as the table)
- syntax:

```
INSERT INTO table
select statement;
```

- example:

```
INSERT INTO follows
SELECT * FROM friendsWith;
```

Questions?