POLITÉCNICO
DO PORTO
ESCOLA
SUPERIOR
DE MEDIA ARTES
E DESIGN

P.PORTO

DATABASES
SQL
Introduction

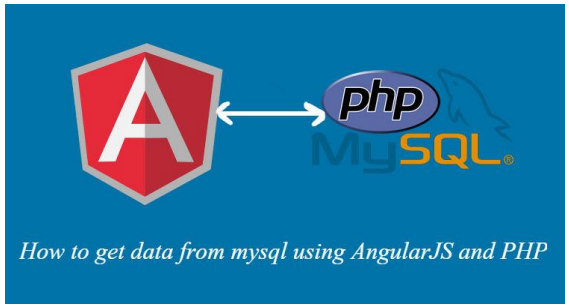
TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

Agenda

- **SQL – Structured Query Language**
  - **Introduction**
  - **Historical Perspective**
  - **Data Definition Language**
  - **Data Manipulation Language**
  - **Data Transaction Language**
  - **Data Control Language**
  - **Data Query language**
- **MySQL: Basic Introduction**
  - **Data Types**
  - **Operators**
  - **Built-in Functions**

INTRODUCTION

- SQL – Structured Query Language



INTRODUCTION

- SQL – Structured Query Language

- Declarative language, which supports the access, interaction and execution of operations in databases

- Standard language, practically universal, which greatly facilitates the transition between different DBMS (DataBase Management Systems) and technologies

- SQL is one of the most widely used languages in applications that use databases

HISTORICAL PERSPECTIVE

- SQL – Structured Query Language
- 1970s - Originally developed by IBM, it was originally called SEQUEL (Structured English Query Language);
- 70s - It was renamed to SQL;
- 1986 - First normalization of the SQL language (SQL86);
- 1992 - International Standard Organization publishes revision in the standard (SQL 92);
- 1999 - Revision of the standard (SQL 99), considerably increasing the functionalities;

HISTORICAL PERSPECTIVE

- SQL – Structured Query Language
- 2006 - XML extensions;
- 2008 - Object-oriented extensions;
- 2011 - Temporary Extensions;
- 2016 – Last language update

SQL LANGUAGES

- SQL – Structured Query Language

The SQL language consists of several groups / families:

- **DATA DEFINITION LANGUAGE (DDL)**

  Data Definition Language allows to define the structure and design of the DB. Create and modify the DB schema, constraints, indexes ...

  CREATE, DROP, ALTER, RENAME…

  - **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
  - **DROP**: This command is used to delete objects from the database.
  - **ALTER**: This is used to alter the structure of the database.
  - **TRUNCATE**: This is used to remove all records from a table, including all spaces allocated for the records are removed.
  - **COMMENT**: This is used to add comments to the data dictionary.
  - **RENAME**: This is used to rename an object existing in the database.

SQL LANGUAGES

- SQL – Structured Query Language

The SQL language consists of several groups / families:

- **DATA MANIPULATION LANGUAGE (DML)**

  Data Manipulation Language allows you to manipulate DB data such as insert, remove and change data; support CRUD operations

  INSERT, UPDATE, DELETE…

- **INSERT** : It is used to insert data into a table.
- **UPDATE**: It is used to update existing data within a table.
- **DELETE** : It is used to delete records from a database table.
- **LOCK:** Table control concurrency.
- **CALL:** Call a PL/SQL

SQL LANGUAGES

- SQL – Structured Query Language

The SQL language consists of several groups / families:

- **DATA TRANSATION LANGUAGE (DTL)**

  Data Transaction Language, allows you to manage the transactions over the DB, terminating them automatically or not.

  BEGIN WORK, COMMIT (permanently save), ROLLBACK (undo changes)

- **COMMIT**: Commits a Transaction.
- **ROLLBACK**: Rollbacks a transaction in case of any error occurs.
- **SAVEPOINT**:Sets a savepoint within a transaction.
- **SET TRANSACTION:** Specify characteristics for the transaction.

SQL LANGUAGES

- SQL – Structured Query Language

The SQL language consists of several groups / families:

- **DATA CONTROL LANGUAGE (DCL)**

  Data Control Language allows you to manage security schemes in accessing data. To grant and take back authority from any database user.

  GRANT, REVOKE

- **GRANT:** This command gives users access privileges to the database.
- **REVOKE:** This command withdraws the user's access privileges given by using the GRANT command.

SQL LANGUAGES

- SQL – Structured Query Language

The SQL language consists of several groups / families:

- **DATA QUERY LANGUAGE (DQL)**

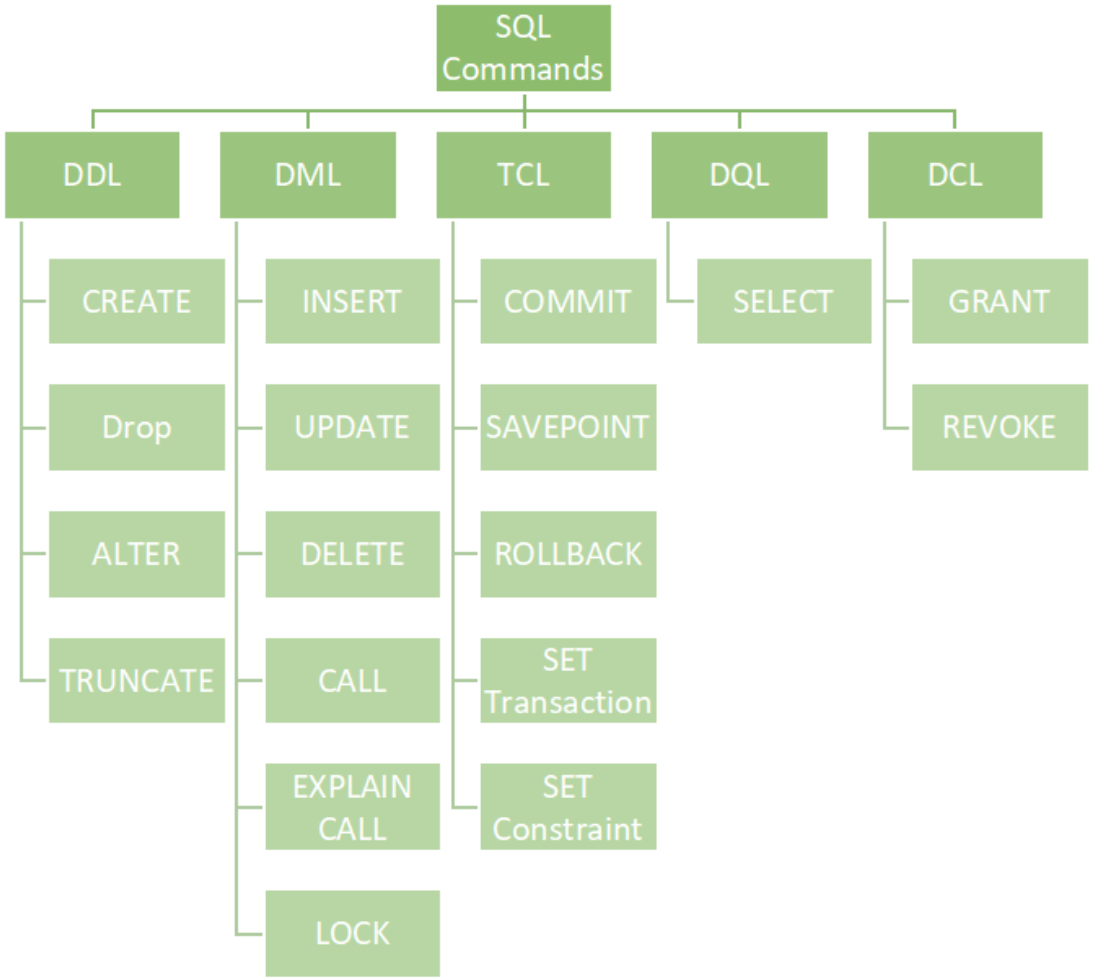
  Data query language is used to fetch data from tables based on conditions that we can easily apply. Retrieve data from database.

  SELECT

  - **SELECT**: It is used to retrieve data from the database.

SQL LANGUAGES

- SQL – Structured Query Language



SQL LANGUAGES

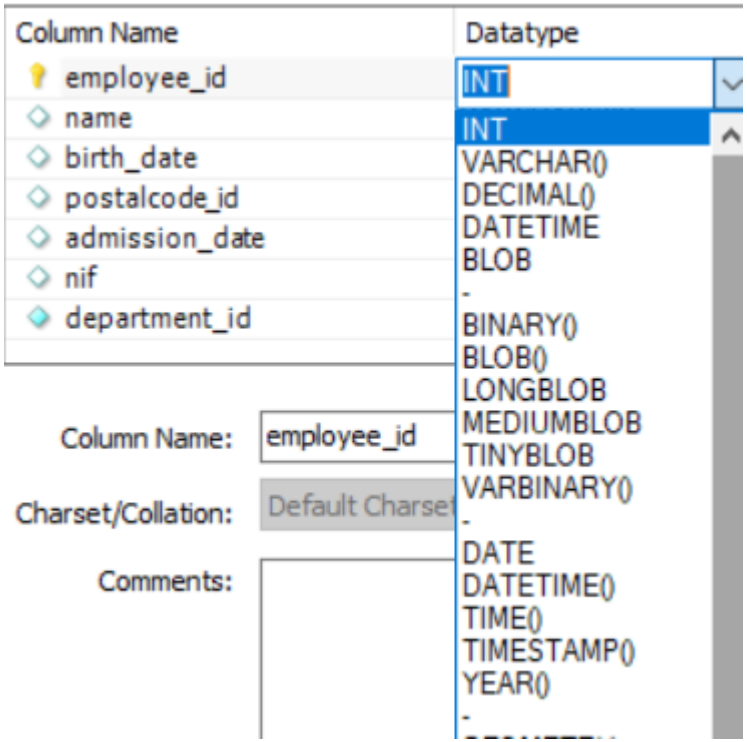
- MySQL

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Numeric
  - String
  - Date and Time
  - JSON
  - BOOLEAN



https://dev.mysql.com/doc/refman/8.0/en/data-types.html

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Numeric Integer Types
    - TYNYINT
    - SMALLINT
    - MEDIUMINT
    - INT
    - BIGINT

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	M Un
TINYINT	1	-128	0	127	25
SMALLINT	2	-32768	0	32767	65
MEDIUMINT	3	-8388608	0	8388607	16
INT	4	-2147483648	0	2147483647	42
BIGINT	8	-2^{63}	0	$2^{63}-1$	2

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Fixed-Point Types
    - DECIMAL | Example: Decimal(5,2): from -999.99 to 999.99
  - Floating-Point Types
    - FLOAT | (4 bytes, precision until 23)
    - DOUBLE | (8 bytes, precision until 53)

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Text
    - CHAR | (size up to 255 characters)
    - VARCHAR | (size up to 65.535 characters)
    - TEXT | (size up to 4.294.967.295 characters)
    - BLOB | A BLOB is a *binary object* that can hold a variable amount of data. The four BLOB types are TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. Useful, f.e., to store images
    - ENUM | Enumeration

```
CREATE TABLE shirts (
    name VARCHAR(40),
    size ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);
```

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Date & Time
    - DATE
    - DATETIME
    - TIMESTAMP
    - TIME
    - YEAR

Type	Usage	Data type format	Range
DATE	Stores only date information in the table column	YYYY-MM-DD format (year, month, and date)	from '1000-01-01' to '9999-12-31'
TIME	Displays only time	HH:MM:SS format (hours, minutes, and seconds)	from '-838:59:59' to '838:59:59'
DATETIME	Stores both date and time in the column	YYYY-MM-DD HH:MM:SS (year, month, and date, and hours, minutes, and seconds)	from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
DATETIME	Stores both date and time values in the column	YYYY-MM-DD HH:MM:SS (year, month, and date, and hours, minutes, and seconds)	from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP	Stores both date and time values in the column. Conversion of the value from the zone of the connection server to UTC takes place.	YYYY-MM-DD HH:MM:SS (year, month, and date, and hours, minutes, and seconds)	from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC
YEAR	Stores only year values in the column	YYYY (year)	from '1901' to '2155'

DATETIME values in 'YYYY-MM-DD hh:mm:ss' format

DATE values in 'YYYY-MM-DD' format

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - Date & Time
    - DATE
    - DATETIME
    - TIMESTAMP
    - TIME
    - YEAR

Data Type	"Zero" Value
DATE	'0000-00-00'
TIME	'00:00:00'
DATETIME	'0000-00-00 00:00:00'
TIMESTAMP	'0000-00-00 00:00:00'
YEAR	0000

DATETIME values in 'YYYY-MM-DD hh:mm:ss' format

DATE values in 'YYYY-MM-DD' format

SQL LANGUAGES

- SQL – Structured Query Language

- Data Types in MySQL
  - JSON

MySQL supports a native JSON data type defined by RFC 7159 that enables efficient access to data in JSON (JavaScript Object Notation) documents.

```
mysql> CREATE TABLE t1 (jdoc JSON);
Query OK, 0 rows affected (0.20 sec)

mysql> INSERT INTO t1 VALUES('{"key1": "value1", "key2": "value2"}');
Query OK, 1 row affected (0.01 sec)
```

SQL LANGUAGES

https://dev.mysql.com/doc/refman/8.0/en/built-in-function-reference.html

- SQL – Structured Query Language
- Operators in MySQL

Name	Description
<	Less than operator
<>, !=	Not equal operator
<<	Left shift
<=	Less than or equal operator
<=>	NULL-safe equal to operator
%, MOD	Modulo operator
*	Multiplication operator
+	Addition operator
-	Minus operator
-	Change the sign of the argument
->	Return value from JSON column after evaluating path; equivalent to JSON_EXTRACT().
->>	Return value from JSON column after evaluating path and unquoting the result; equivalent to JSON_UNQUOTE(JSON_EXTRACT()).
/	Division operator
:=	Assign a value
=	Assign a value (as part of a SET statement, or as part of the SET clause in an UPDATE statement)
=	Equal operator

SQL LANGUAGES

https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html

- SQL – Structured Query Language
- Numeric Function in MySQL

Name	Description
ABS()	Return the absolute value
ACOS()	Return the arc cosine
ASIN()	Return the arc sine
ATAN()	Return the arc tangent
ATAN2(), ATAN()	Return the arc tangent of the two arguments
CEIL()	Return the smallest integer value not less than the argument
CEILING()	Return the smallest integer value not less than the argument
CONV()	Convert numbers between different number bases
COS()	Return the cosine
COT()	Return the cotangent
CRC32()	Compute a cyclic redundancy check value
DEGREES()	Convert radians to degrees
DIV	Integer division
EXP()	Raise to the power of
FLOOR()	Return the largest integer value not greater than the argument

SQL LANGUAGES

https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

- SQL – Structured Query Language
- String Function in MySQL

Name	Description
ASCII()	Return numeric value of left-most character
BIN()	Return a string containing binary representation of a number
BIT_LENGTH()	Return length of argument in bits
CHAR()	Return the character for each integer passed
CHAR_LENGTH()	Return number of characters in argument
CHARACTER_LENGTH()	Synonym for CHAR_LENGTH()
CONCAT()	Return concatenated string
CONCAT_WS()	Return concatenate with separator
ELT()	Return string at index number
EXPORT_SET()	Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
FIELD()	Index (position) of first argument in subsequent arguments
FIND_IN_SET()	Index (position) of first argument within second argument
FORMAT()	Return a number formatted to specified number of decimal places
FROM_BASE64()	Decode base64 encoded string and return result
HEX()	Hexadecimal representation of decimal or string value
INSERT()	Insert substring at specified position up to specified number of characters