



UNIVERSIDADE  
VILA VELHA  
ESPÍRITO SANTO

DML

Prof. Jean-Rémi Bourguet

Banco de Dados (Projeto Físico)

- ▶ Os **padrões SQL** mais recentes são divididos em:
  - ▶ uma **especificação núcleo**;
  - ▶ **extensões especializadas**.
- ▶ **Núcleo** deve ser **implementado** por **todos** os fornecedores de **SGBD**.
- ▶ Assim eles ficam compatíveis com SQL.

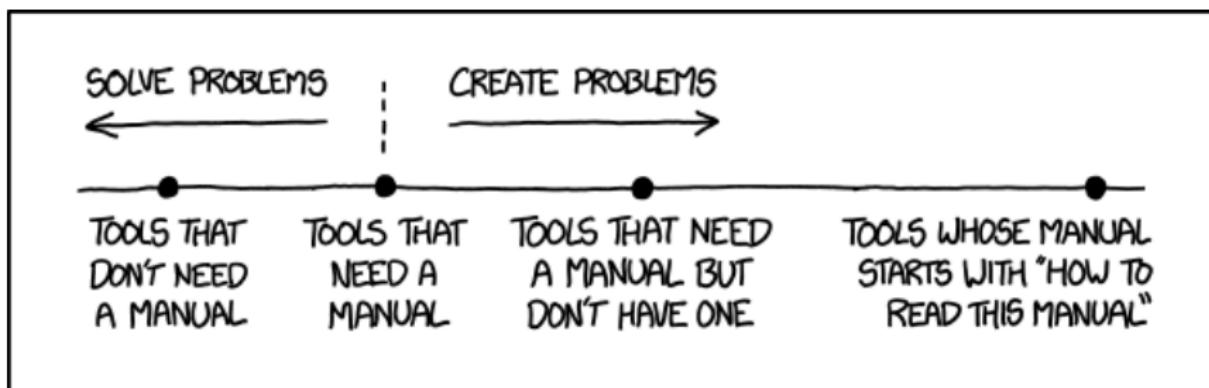


# Backus Naur Form

SQL

Os seguintes caracteres nas sintaxes dos comandos significam:

- <> Indica que o **conteúdo** entre eles deve ser **definido pelo usuário**.
- [ ] Indica que o **conteúdo** entre eles é **opcional**.
- | Indica que será **válida** qualquer uma das **opções separadas por |**.
- ... Indica **continuidade**.



[http://sql1602.sourceforge.net/helpdir-en/html/HE\\_SQL2\\_POPISSYNTAXE.htm](http://sql1602.sourceforge.net/helpdir-en/html/HE_SQL2_POPISSYNTAXE.htm)

- DML significa Data Manipulation Language.
- Esta é a Linguagem de Manipulação de Dados.
- Permite interagir diretamente com os dados dentro das tabelas.
- Os comandos da DML são INSERT, UPDATE e DELETE.

Dividida em 4 agrupamentos

DML - Data Manipulation Language  
INSERT, DELETE, UPDATE e SELECT

DDL - Data Definition Language  
CREATE, ALTER e DROP

DCL - Data Control Language  
GRANT, REVOKE, DENY

DTL - Data Transaction Language  
BEGIN TRANSACTION, COMMIT, ROLLBACK

# Lembranças da DDL

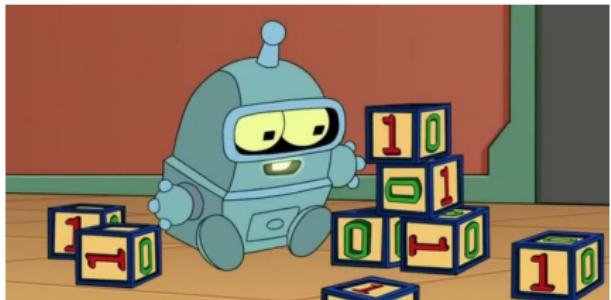
- Em UML, pode especificar o **valor padrão** que um atributo assume...
- ... quando uma **instância** de um classificador é **criada**.
- O **valor inicial** é representado como **atribuição de variável**.



<b>&lt;&lt;Table&gt;&gt;</b>
<b>Professor</b>
<b>&lt;&lt;PK&gt;&gt;</b> professor_id: SMALLINT
primeiroNome: VARCHAR(20)
sobrenome: VARCHAR(20)
telefone: VARCHAR(11) [0..1] = '(27) 3421-2001'
titulo: CHAR(3) [0..1]
cargaHoraria: TINYINT [0..1]

# Lembranças da DDL

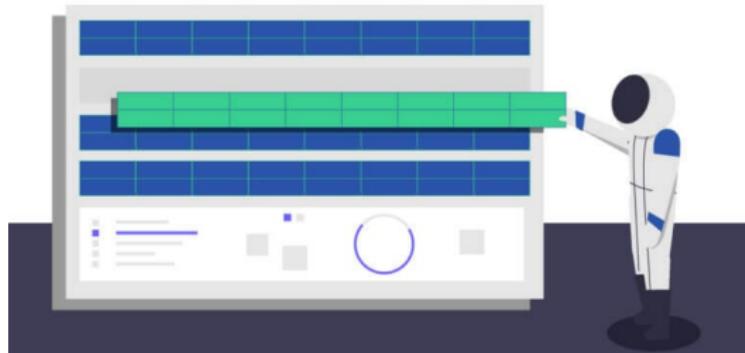
```
mysql> CREATE SCHEMA Exam;  
mysql> USE Exam;  
CREATE TABLE Professor (  
    professor_id      SMALLINT          NOT NULL,  
    primeiroNome     VARCHAR(20)        NOT NULL,  
    sobrenome         VARCHAR(20)        NOT NULL,  
    telefone          VARCHAR(15)        DEFAULT '(27) 3421-2001',  
    titulo            CHAR(3),  
    cargaHoraria      TINYINT,  
    CONSTRAINT PK_Professor PRIMARY KEY (professor_id)  
)
```



# O comando INSERT

- O comando para **incluir dados** em uma tabela é o **INSERT INTO**.
- Indica **nome da tabela, nomes das colunas e respectivos valores**.

```
INSERT INTO <nome_tabela>
[(<nome_Coluna1>, <nome_Coluna2>, ...)]
VALUES
(<valor_Coluna1_Tupla1>, <nome_Coluna2_Tupla1>, ...),
[(<valor_Coluna1_Tupla2>, <nome_Coluna2_Tupla2>, ...),]
...
```



# O comando INSERT

- A **forma** mais simples **INSERT** acrescenta uma **única tupla** na tabela.
- Especificar: **nome da tabela** (com/sem todos os atributos) e **valores**.
- Valores na **mesma ordem** dos **atributos** (**INSERT** ou **CREATE TABLE**).

```
mysql> INSERT INTO Professor VALUES(0,'Ada','Lovelace','+44 1223337733','BSc',30);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO
Professor(telefone,primeiroNome,sobrenome,cargaHoraria,titulo,professor_id)
VALUES('+35 321490300','George','Boole',40,'PhD',1);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM Professor;
+-----+-----+-----+-----+-----+-----+
| professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |
+-----+-----+-----+-----+-----+-----+
| 0 | Ada | Lovelace | +44 1223337733 | BSc | 30 |
| 1 | George | Boole | +35 321490300 | PhD | 40 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

# O comando INSERT

- ▶ Note que é **possível inserir** em uma tabela **múltiplas tuplas...**
- ▶ ...separadas por vírgulas em um **único comando INSERT**.

```
mysql> INSERT INTO Professor  
VALUES(2,'Alan','Turing','+44 1613066000','PhD',40),  
      (3,'Claude','Shannon','+1 6172531000','PhD',30);
```

```
mysql> INSERT INTO  
Professor(primeiroNome,sobrenome,professor_id,telefone,cargaHoraria,titulo)  
VALUES('Alan','Turing',2,'+44 1613066000',40,'PhD'),  
      ('Claude','Shannon',3,'+1 6172531000',30,'PhD');
```

```
mysql> SELECT * FROM Professor;
```

professor_id	primeiroNome	sobrenome	telefone	titulo	cargaHoraria
0	Ada	Lovelace	+44 1223337733	BSc	30
1	George	Boole	+35 321490300	PhD	40
2	Alan	Turing	+44 1613066000	PhD	30
3	Claude	Shannon	+1 6172531000	PhD	40

4 rows in set (0.00 sec)

# O comando INSERT

- Existe uma **segunda forma** de instrução INSERT.
- Permite que o usuário especifique **nomes de atributos explícitos**...
- ...que correspondem aos **valores fornecidos** no comando INSERT.

```
mysql>
INSERT INTO Professor(professor_id,primeiroNome,sobrenome,telefone,titulo)
VALUES(4,'Dennis','Ritchie','+1 6174951000','MSc');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> SELECT * FROM Professor;
+-----+-----+-----+-----+-----+-----+
| professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |
+-----+-----+-----+-----+-----+-----+
|       0 | Ada          | Lovelace  | +44 1223337733 | BSc   |      30 |
|       1 | George        | Boole     | +35 321490300  | PhD   |      40 |
|       2 | Alan          | Turing    | +44 1613066000 | PhD   |      40 |
|       3 | Claude         | Shannon   | +1 6172531000  | PhD   |      30 |
|       4 | Dennis         | Ritchie  | +1 6174951000  | MSc   |      NULL |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

# O comando INSERT

- Deve dar atenção maior para colunas que não aceitam **valores nulos**.
- Os atributos com **NULL permitidos**, **DEFAULT** ou **AUTOINCREMENT**...
- ...são aqueles que **podem ser omitidos**.

```
mysql> INSERT INTO Professor(primeiroNome,sobrenome,professor_id)
VALUES('Ken','Thompson',5);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM Professor;
```

professor_id	primeiroNome	sobrenome	telefone	titulo	cargaHoraria
0	Ada	Lovelace	+44 1223337733	BSc	30
1	George	Boole	+35 321490300	PhD	40
2	Alan	Turing	+44 1613066000	PhD	40
3	Claude	Shannon	+1 6172531000	PhD	30
4	Dennis	Ritchie	+1 6174951000	MSc	NULL
5	Ken	Thompson	(27) 3421-2001	NULL	NULL

```
6 rows in set (0.00 sec)
```

# Filtros e operadores

- A DML permite a **criação de filtros** para alguns de seus comandos.
- Os filtros são utilizados por meio da **cláusula WHERE**.
- Eles utilizam os seguintes **operadores relacionais e lógicos...**
- ...além de alguns **operadores especiais**.

Operador	Comparação
<code>==</code>	Igual
<code>!= ou &lt; &gt;</code>	Diferente
<code>&lt;</code>	Menor
<code>&gt;</code>	Maior
<code>&lt;=</code>	Menor Igual
<code>&gt;=</code>	Maior Igual



# Filtros e operadores

- Eles são utilizados **junto à cláusula WHERE** com os comandos:  
**UPDATE, DELETE** e **SELECT**.

The slide features a dark background with large, bold, colorful text on the left side. The text is arranged in three rows: 'SELECT' in light blue, 'UPDATE' in yellow, and 'DELETE' in pink. To the right of the text is a screenshot of a 'Book Details' form. The form has a purple border and contains fields for Book ID, Book Name, Author Name, Language, Publisher Name, Edition, Actual Stock, Book Description, and various date and price fields. It also includes sections for 'Genre' (with options like Action, Adventure, Comic book, Self Help, Motivation) and 'Pages'. At the bottom are three buttons: 'Add' (green), 'Update' (yellow), and 'Delete' (red). A small icon of stacked books is at the top of the form.

# Filtros e operadores

- ▶ Outro operador comumente utilizado é o **LIKE**.
- ▶ Sua função é **validar expressões** com **dados semelhantes**.
- ▶ Permite a **comparação** com **valores aproximados**.

```
<nome_atributo> LIKE "<string_comparação>"
```



# Filtros e operadores

- O **operador LIKE** pesquisa um **padrão especificado** em uma coluna.
- Existem **dois curingas** geralmente usados **no operador LIKE**:
  - O **sinal de porcentagem (%)** representa **zero, um ou vários caracteres**.
  - O **sinal de sublinhado (\_)** representa um **único caractere**.
- Esses sinais também podem ser **usados em combinações!**



- ▶ Junto com a **string de comparação** é usado o **símbolo percentual (%)**.
- ▶ Ele é usado como **curinga** substituindo **parte da string** desconhecida.
- ▶ O caracter curinga pode substituir qualquer parte da string.

## Wildcard in SQL



WHERE column LIKE  
'AAAA%'

SELECT \* FROM  
Employees

SELECT FROM  
table\_name

WHERE City LIKE '\_elhi';

WHERE City LIKE 'the%';

WHERE City NOT LIKE  
'[abc]%'

educba.com

🔗 [https://www.w3schools.com/sql/sql\\_like.asp](https://www.w3schools.com/sql/sql_like.asp)

# Filtros e operadores

- ▶ Toda **expressão de comparação** tem como resultado **TRUE ou FALSE**.
- ▶ A **comparação** realizada **com valor NULL** resulta em **UNKNOWN**.
- ▶ Para **compararmos valores NULL**, usamos o **predicado IS**.
- ▶ As duas formas de uso são **IS NOT NULL** ou **IS NULL**.



# O comando UPDATE

- ▶ O comando **UPDATE** queremos atualizar dados em uma tabela.
- ▶ **Modificar valores** de atributo de **uma ou mais tuplas** selecionadas.
- ▶ **WHERE** seleciona as **tuplas a serem modificadas** em uma única tabela.

```
UPDATE <nome_tabela>
```

```
SET <nome_Coluna1> = expr1 [,<nome_Coluna2> = expr2 ...]
```

```
[WHERE <condição>];
```



# O comando UPDATE

- A cláusula **SET** atribui um **novo valor** a um **determinado atributo**.
- **Especifica os atributos** a serem **modificados** e seus **novos valores**.

```
mysql> UPDATE Professor  
SET cargaHoraria = 35  
WHERE cargaHoraria IS NULL;  
Query OK, 2 rows affected (0.00 sec)  
Rows matched: 2    Changed: 2    Warnings: 0
```

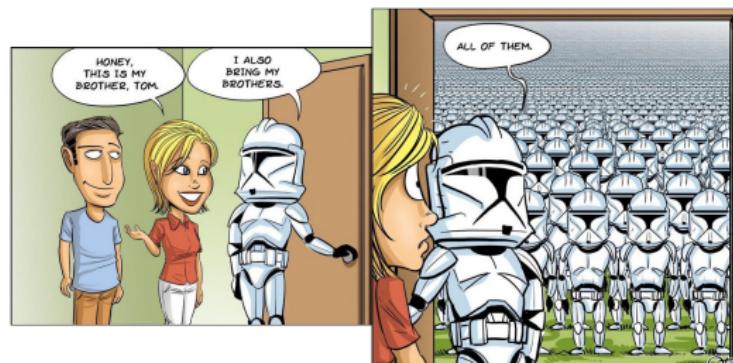
```
mysql> SELECT * FROM Professor;  
+-----+-----+-----+-----+-----+-----+  
| professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |  
+-----+-----+-----+-----+-----+-----+  
|       0 | Ada          | Lovelace  | +44 1223337733 | BSc   |           30 |  
|       1 | George        | Boole     | +35 321490300  | PhD   |           40 |  
|       2 | Alan          | Turing    | +44 1613066000 | PhD   |           40 |  
|       3 | Claude         | Shannon   | +1 6172531000  | PhD   |           30 |  
|       4 | Dennis         | Ritchie  | +1 6174951000  | MSc   |           35 |  
|       5 | Ken           | Thompson  | (27) 3421-2001 | NULL  |           35 |  
+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

# O comando UPDATE

- Vamos **clonar** esta **tabela**!

```
mysql> CREATE TABLE Professor2 SELECT * FROM Professor;  
Query OK, 5 rows affected (0.00 sec)  
Records: 5 Duplicates: 0 Warnings: 0
```

```
mysql> CREATE TABLE Professor3 SELECT * FROM Professor;  
Query OK, 5 rows affected (0.00 sec)  
Records: 5 Duplicates: 0 Warnings: 0
```



# O comando UPDATE

- Várias tuplas podem ser **modificadas** com um **único UPDATE**.
- **WHERE** restringe atualizações aos **registros satisfazendo condições**.

```
mysql> UPDATE Professor  
SET cargaHoraria = 40  
WHERE titulo = 'PhD';  
Query OK, 1 row affected (0.00 sec)  
Rows matched: 3    Changed: 1    Warnings: 0
```

```
mysql> SELECT * FROM Professor;  
+-----+-----+-----+-----+-----+-----+  
| professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |  
+-----+-----+-----+-----+-----+-----+  
|       0 | Ada          | Lovelace   | +44 1223337733 | BSc    |           30 |  
|       1 | George        | Boole       | +35 321490300  | PhD    |           40 |  
|       2 | Alan          | Turing      | +44 1613066000 | PhD    |           40 |  
|       3 | Claude         | Shannon     | +1 6172531000  | PhD    |           40 |  
|       4 | Dennis         | Ritchie    | +1 6174951000  | MSc    |           35 |  
|       5 | Ken           | Thompson   | (27) 3421-2001 | NULL   |           35 |  
+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

# O comando UPDATE

- Várias tuplas podem ser **modificadas** com um comando **UPDATE** só.
- Utiliza-se **expressões aritméticas** para efetuar **alteração nos dados**.
- Vamos supor que houve um aumento de 20% das cargas horárias.

```
mysql> UPDATE Professor  
SET cargaHoraria = cargaHoraria *1.2;  
Query OK, 5 rows affected (0.03 sec)  
Rows matched: 5    Changed: 5    Warnings: 0
```

```
mysql> SELECT * FROM Professor;  
+ professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |  
+-----+-----+-----+-----+-----+-----+  
| 0 | Ada | Lovelace | +44 1223337733 | BSc | 36 |  
| 1 | George | Boole | +35 321490300 | PhD | 48 |  
| 2 | Alan | Turing | +44 1613066000 | PhD | 48 |  
| 3 | Claude | Shannon | +1 6172531000 | PhD | 48 |  
| 4 | Dennis | Ritchie | +1 6174951000 | MSc | 42 |  
| 5 | Ken | Thompson | (27) 3421-2001 | NULL | 42 |  
+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

# O comando UPDATE

- ▶ **UPDATE** refere-se explicitamente a apenas uma **única tabela**.
- ▶ Para **modificar várias tabelas**, emite-se **vários comandos UPDATE**.
- ▶ Possível **especificar NULL ou DEFAULT** como o novo valor do atributo.

```
mysql> UPDATE Professor  
SET telefone = DEFAULT, cargaHoraria = NULL  
WHERE professor_id = 1;  
Query OK, 1 row affected (0.01 sec)  
Rows matched: 1    Changed: 1    Warnings: 0
```

```
mysql> SELECT * FROM Professor;  
+ professor_id | primeiroNome | sobrenome | telefone | titulo | cargaHoraria |  
+-----+-----+-----+-----+-----+-----+  
| 0 | Ada | Lovelace | +44 1223337733 | BSc | 36 |  
| 1 | George | Boole | (27) 3421-2001 | PhD | NULL |  
| 2 | Alan | Turing | +44 1613066000 | PhD | 48 |  
| 3 | Claude | Shannon | +1 6172531000 | PhD | 48 |  
| 4 | Dennis | Ritchie | +1 6174951000 | MSc | 42 |  
| 5 | Ken | Thompson | (27) 3421-2001 | NULL | 42 |  
+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

# O comando UPDATE

- A atualização de uma **chave primária** pode ser **propagada...**
- ...para os **valores de chave estrangeira** das tuplas em **outras tabelas...**
- ...se tal ação de **disparo referencial** for **especificada** na DDL.

```
CREATE TABLE Exame (
    primeiroNome      VARCHAR(20)  NOT NULL,
    sobrenome         VARCHAR(20)  NOT NULL,
    cpf               BIGINT(11)   NOT NULL AUTO_INCREMENT,
    email             VARCHAR(255) UNIQUE,
    resultado         DECIMAL(4,2),
    professor         SMALLINT,
    feedback          TEXT,
    CONSTRAINT PK_Exame PRIMARY KEY (cpf),
    CONSTRAINT FK_ExameProfessor FOREIGN KEY(professor)
        REFERENCES Professor(professor_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

# O comando UPDATE

```
mysql> INSERT INTO Exame(primeiroNome,sobrenome,cpf,professor)
VALUES('Edgar','Codd',98765432100,0);
```

```
mysql> SELECT * FROM Exame;
+-----+-----+-----+-----+-----+-----+-----+
| primeiroNome | sobrenome | cpf | email | resultado | professor | feedback |
+-----+-----+-----+-----+-----+-----+-----+
| Edgar | Codd | 98765432100 | NULL | NULL | 0 | NULL |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> UPDATE Professor
SET professor_id = 6
WHERE professor_id = 0;
```

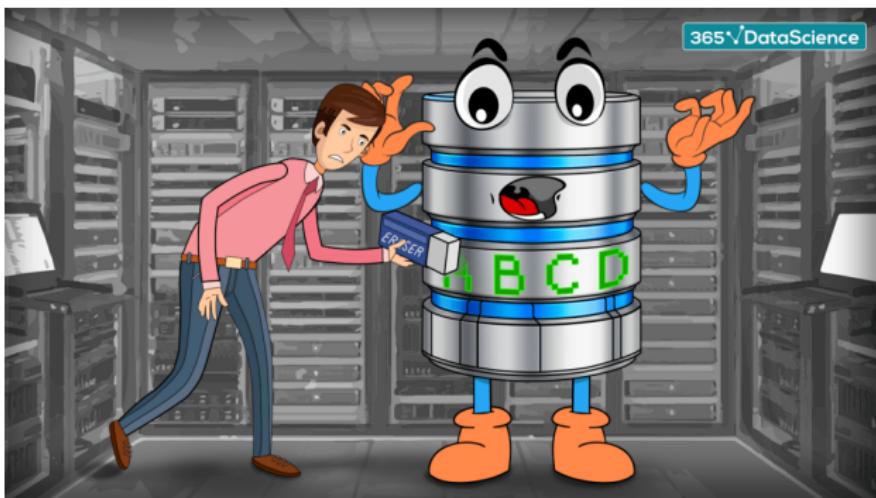
```
mysql> SELECT * FROM Exame;
+-----+-----+-----+-----+-----+-----+-----+
| primeiroNome | sobrenome | cpf | email | resultado | professor | feedback |
+-----+-----+-----+-----+-----+-----+-----+
| Edgar | Codd | 98765432100 | NULL | NULL | 6 | NULL |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

# O comando DELETE

- Para a **exclusão de dados** em uma tabela utiliza-se **DELETE**:

```
DELETE FROM <nome_tabela>
[WHERE <condição>];
```

- WHERE** restringe os registros nos quais DELETE atuará.



# O comando DELETE

- O comando **DELETE remove tuplas** de uma tabela.
- Inclui uma cláusula WHERE para **selecionar** as **tuplas** a serem **excluídas**.
- As **tuplas** são **explicitamente excluídas** de apenas uma tabela por vez.

```
mysql> DELETE FROM Professor  
WHERE cargaHoraria > 42;  
Query OK, 3 rows affected (0.02 sec)
```

```
mysql> SELECT * FROM Professor;
```

professor_id	primeiroNome	sobrenome	telefone	titulo	cargaHoraria
1	George	Boole	(27) 3421-2001	PhD	NULL
4	Dennis	Ritchie	+1 6174951000	MSc	42
5	Ken	Thompson	(27) 3421-2001	NULL	42
6	Ada	Lovelace	+44 1223337733	BSc	36

4 rows in set (0.00 sec)

# O comando DELETE

- A exclusão pode se **propagar** para as **tuplas em outras tabelas...**
- ...se **ações de disparo referencial** forem especificadas na DDL.

```
mysql> DELETE FROM Professor  
WHERE professor_id = 6;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM Professor;
```

professor_id	primeiroNome	sobrenome	telefone	titulo	cargaHoraria
1	George	Boole	(27) 3421-2001	PhD	NULL
4	Dennis	Ritchie	+1 6174951000	MSc	42
5	Ken	Thompson	(27) 3421-2001	NULL	42

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM Exame;
```

primeiroNome	sobrenome	cpf	email	resultado	professor	feedback
Edgar	Codd	98765432100	NULL	NULL	NULL	NULL

1 row in set (0.00 sec)

# O comando DELETE

- Dependendo do **número de tuplas selecionadas** no WHERE...
- ...**zero, uma** ou **várias tuplas** podem ser **excluídas por DELETE**.
- Se **WHERE não for especificada, apagará todas as tuplas** da tabela.
- Porem, a **tabela permanece** no BD como uma **tabela vazia**.



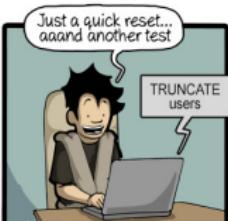
```
mysql> DELETE FROM Professor;  
Query OK, 3 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM Professor;  
Empty set (0.00 sec)
```

# O comando TRUNCATE TABLE

- ▶ O comando **TRUNCATE TABLE** exclui os dados de uma tabela...
- ▶ ...mas não a própria tabela.

```
mysql> TRUNCATE TABLE Professor3;  
Query OK, 6 row affected (0.00 sec)
```



Script with sophisticated update function



CommitStrip.com



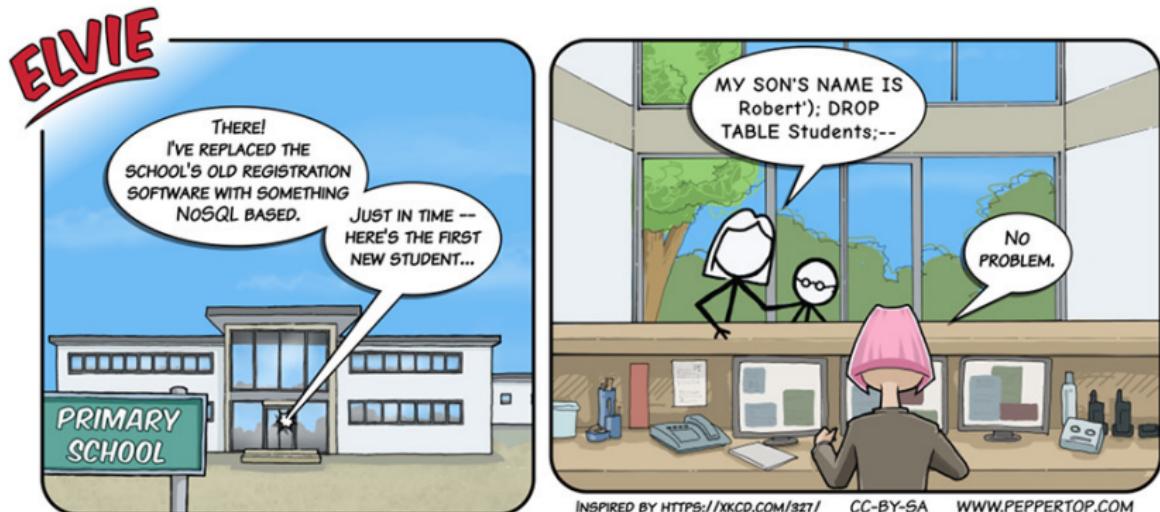
Script with TRUNCATE and INSERT all



[https://w3schools.com/sql/sql\\_ref\\_drop\\_table.asp](https://w3schools.com/sql/sql_ref_drop_table.asp)

# Clonar, Duplicar, Apagar, Salvar e Restaurar

- A **tabela é vazia** mas **ainda existe!**
- Para **remover a tabela**, é necessário passar o **comando DROP...**
- Porém a **tabela Exam** possui uma **restrição de integridade...**
- ...com uma **chave estrangeira** que **aponta** para a **tabela Professor**.



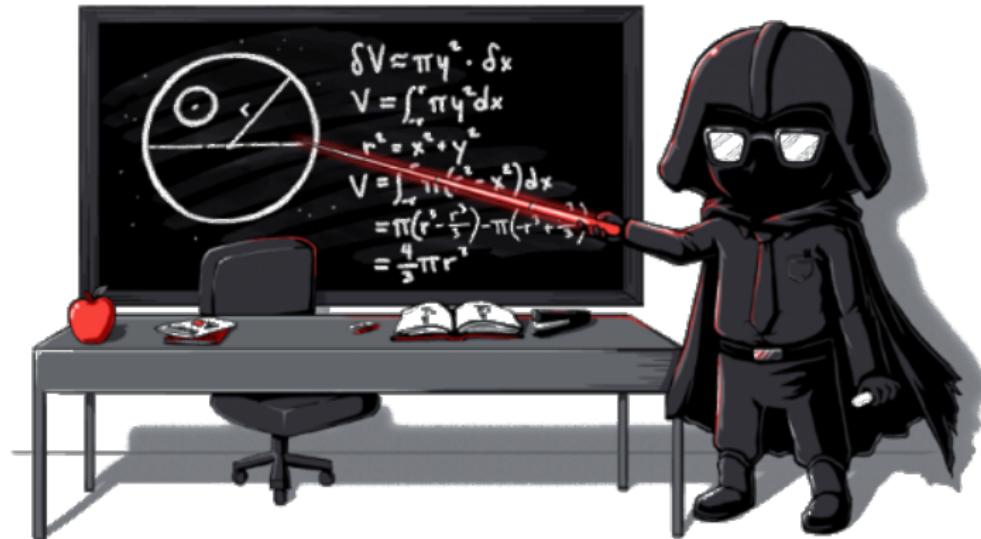
# Clonar, Duplicar, Apagar, Salvar e Restaurar

- Então, vamos somente **restaurar** os dados da tabela Professor.

```
mysql> INSERT INTO Professor SELECT * FROM Professor2;
```

```
Query OK, 6 rows affected (0.11 sec)
```

```
Records: 6    Duplicates: 0    Warnings: 0
```

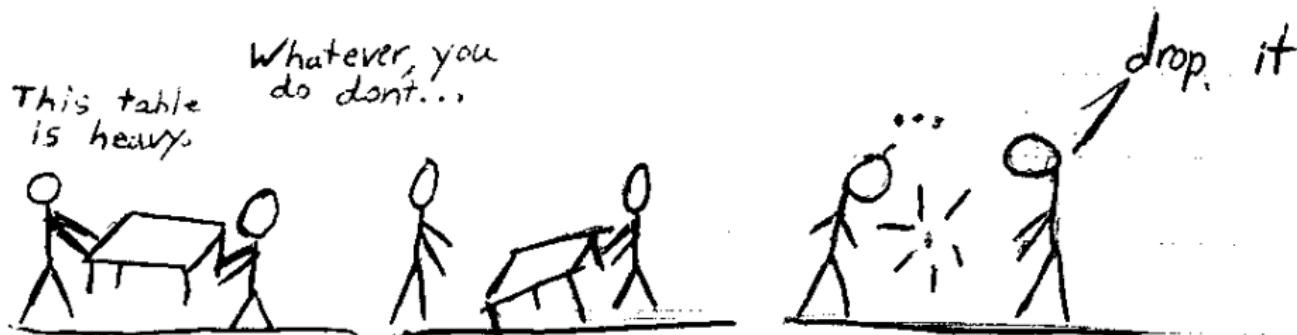


# Clonar, Duplicar, Apagar, Salvar e Restaurar

- Vamos remover as tabelas Professor2 e Professor3!

```
mysql> DROP TABLE Professor2;  
Query OK, 0 rows affected (0.17 sec)
```

```
mysql> DROP TABLE Professor3;  
Query OK, 0 rows affected (0.17 sec)
```



# Clonar, Duplicar, Apagar, Salvar e Restaurar

- Vamos **adicionar exames** de alunos examinados **pelos professores**.

```
mysql> UPDATE Exame
```

```
    SET professor = 1;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Rows matched: 1    Changed: 1    Warnings: 0
```

```
mysql> INSERT INTO Exame(primeiroNome,sobrenome,professor)
```

```
VALUES('Peter','Chen',2),
```

```
('Gordon','Everest',2),
```

```
('Clive','Finkelstein',4),
```

```
('James','Martin',4),
```

```
('Richard','Barker',2),
```

```
('Grady','Booch',4),
```

```
('James','Rumbaugh',5),
```

```
('Ivar','Jacobson',1);
```

```
Query OK, 8 rows affected (0.00 sec)
```

```
Records: 8    Duplicates: 0    Warnings: 0
```

# Clonar, Duplicar, Apagar, Salvar e Restaurar

- Vamos **fazer um dump** do banco.
- A **opção -B** é usada para **declarar o nome do esquema** no script.

```
$ mysqldump -u root -p -B exam > /path/to/dump.sql  
Enter password:
```



# Clonar, Duplicar, Apagar, Salvar e Restaurar

- Podemos **restaurar este backup**.
- Basta que **redirecionemos a entrada padrão para o mysql**.

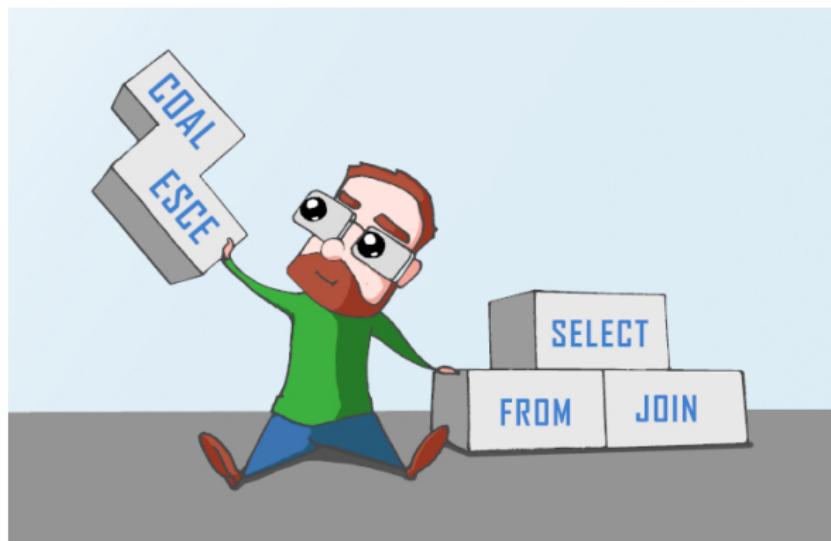
```
mysql> create database Exame;  
mysql> use Exame;  
mysql> source /path/to/dump.sql;
```

ou

```
$ mysql -u root -p < /path/to/dump.sql
```



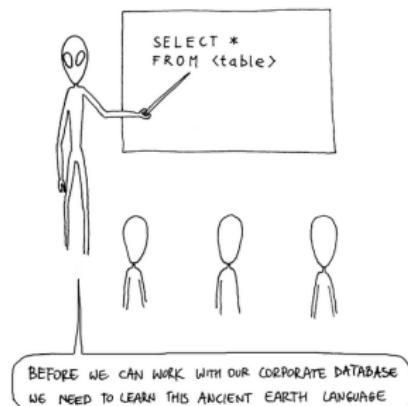
- DQL significa Data Query Language.
- Esta é a Linguagem de Consulta de Dados.
- Permite consultar os dados do banco.
- O comando principal da DQL é SELECT.



# O comando SELECT

- O comando **SELECT** é utilizado para **selecionar os dados** das **tabelas**.

```
SELECT <nome_Coluna1>, [<nome_Coluna2>, ...] | *
FROM <nome_tabela1>, [<nome_tabela2> ...]
[WHERE <condição>]
[GROUP BY <nome_coluna>]
[ORDER BY <nome_coluna> ASC | DESC];
```



IT'S NOT EXACTLY WHAT I MEANT WHEN  
I SAID I NEED AN SQL NINJA...

# O comando SELECT

- A partir da **tabela, ou tabelas, listada(s)** na **cláusula FROM...**
- ...as **consultas executam** o que foi especificado no **WHERE e SELECT**.



```
SELECT * from Contacts  
WHERE behavior = 'nice'
```

**SQL CLAUSE**  
is coming to town!

# O comando SELECT

- A partir dessas cláusulas, elas sempre **produzem** uma **outra tabela**.
- O **SELECT** nos proporciona uma **grande variedade** de **consultas**.



<https://dev.mysql.com/doc/refman/8.0/en/select.html>

# O comando SELECT

- Vamos **recuperar todos os dados** de uma das tabelas do nosso BD.
- Ao invés de **discriminarmos os atributos**, colocamos o **sinal asterisco**.
- Indica que queremos **visualizar todos os atributos** contidos na tabela.

```
mysql> SELECT * FROM Exame;
```

primeiroNome	sobrenome	cpf	email	resultado	professor	feedback
Edgar	Codd	98765432100	NULL	NULL	1	NULL
Peter	Chen	98765432101	NULL	NULL	2	NULL
Gordon	Everest	98765432102	NULL	NULL	2	NULL
Clive	Finkelstein	98765432103	NULL	NULL	4	NULL
James	Martin	98765432104	NULL	NULL	4	NULL
Richard	Barker	98765432105	NULL	NULL	2	NULL
Grady	Booch	98765432106	NULL	NULL	4	NULL
James	Rumbaugh	98765432107	NULL	NULL	5	NULL
Ivar	Jacobson	98765432108	NULL	NULL	1	NULL

9 rows in set (0.00 sec)



# O comando SELECT

- Se quisermos **visualizar apenas** os nomes dos professores ou alunos...
- **trocaremos** o asterisco pelo(s) **nome(s)** do(s) **atributo(s)** adequado(s).

```
mysql> SELECT primeiroNome, sobrenome FROM Professor;
```

primeiroNome	sobrenome
Ada	Lovelace
George	Boole
Alan	Turing
Claude	Shannon
Dennis	Ritchie
Ken	Thompson

6 rows in set (0.00 sec)



# O comando SELECT

- A expressão de **condição** da **cláusula WHERE** serve como **filtro**.
- Execução da **seleção** apenas nas **tuplas** que **satisfazam a condição**.
- Se agora quisermos **visualizar** os **nomes e números de telefone**...
- ...dos **professores** com uma **carga horária** de **menos de 40 horas**.

```
mysql> SELECT sobrenome, telefone  
        FROM Professor  
       WHERE cargaHoraria < 40;
```

sobrenome	telefone
Lovelace	+44 1223337733
Shannon	+1 6172531000
Ritchie	+1 6174951000
Thompson	(27) 3421-2001

4 rows in set (0.00 sec)

# O comando SELECT

- A expressão de **condição** da **cláusula WHERE** pode ser **composta**.
- Se agora quisermos **visualizar** os **nomes e números de telefone**...
- ...dos **MSc** com uma **carga horária de mais de 30 horas**.

```
mysql> SELECT sobrenome, telefone FROM Professor  
        WHERE titulo = 'MSc' AND cargaHoraria > 30;
```

sobrenome	telefone
Ritchie	+1 6174951000

1 row in set (0.00 sec)



# O comando SELECT

- Para **listar os nomes de todos professores** que iniciam com a letra "T":

```
mysql> SELECT sobrenome  
        FROM Professor  
       WHERE sobrenome LIKE 'T%';  
+-----+  
| sobrenome |  
+-----+  
| Turing    |  
| Thompson |  
+-----+  
2 rows in set (0.00 sec)
```



# O comando SELECT

- Para **listar** os **nomes dos professores** que não iniciam com a letra "T":

```
mysql> SELECT sobrenome  
        FROM Professor  
       WHERE sobrenome NOT LIKE 'T%';
```

sobrenome
Lovelace
Boole
Shannon
Ritchie

4 rows in set (0.00 sec)

One of These Is  
Not Like the Others



BARNEY SALTZBERG

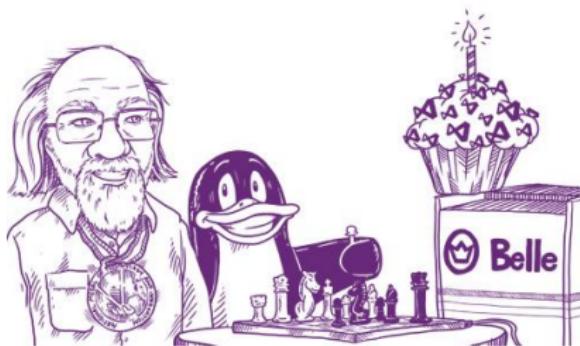
# O comando SELECT

- Para verificar os **professores** que não possuem **cadastro de título**:

```
mysql> SELECT primeiroNome, sobrenome  
        FROM Professor  
       WHERE titulo IS NULL;
```

primeiroNome	sobrenome
Ken	Thompson

1 row in set (0.00 sec)



# O comando SELECT

- Para verificar os **professores** que possuem **cadastro de título**:

```
mysql> SELECT primeiroNome, sobrenome  
        FROM Professor  
       WHERE titulo IS NOT NULL;
```

primeiroNome	sobrenome
Ada	Lovelace
George	Boole
Alan	Turing
Claude	Shannon
Dennis	Ritchie

5 rows in set (0.00 sec)

