AULA 15 – CHAVES ESTRANGEIRAS E JOIN

Quando criamos uma tabela utilizando o MySQL, a gente precisa escolher uma coisa chamada **ENGINE**, que é a máquina que vai poder criar registros

Exemplo:

```
DROP TABLE IF EXISTS `cursos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `cursos` (
  `idcurso` int(11) NOT NULL DEFAULT '0',
  `nome` varchar(30) NOT NULL,
  `descricao` text,
  `carga` int(10) unsigned DEFAULT NULL,
  `totaulas` int(10) unsigned DEFAULT NULL,
  `ano` year(4) DEFAULT '2016',
  PRIMARY KEY (`idcurso`),
  UNIQUE KEY `nome` (`nome`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

A **InnoDB** é uma engine que vai permitir a criação de tabelas com algumas características que iremos precisar
A principal característica desta engine é suportar chaves estrangeiras

Algumas engines:

- ✓ **MyISAM**
- ✓ **InnoDB**
- ✓ **XtraDB**

→ A InnoDB e a XtraDB suportam o que chamamos de **ACID**, que são 4 principais regras de uma boa transação

A → ATOMICIDADE – não pode ser dividida em subtarefas. Exemplo: eu tenho uma tarefa a ser feita: ou toda ela é feita, ou nada será considerado

C → CONSISTÊNCIA – se antes de fazer a transação o banco de dados estava ok, depois que terminar a transação ele também tem que estar ok. Se ocorrer inconsistências ou falhas, tudo é desfeito para o estado anterior

I → ISOLAMENTO – quando eu tenho duas transações acontecendo em paralelo, elas devem ser executadas de forma isolada. No MySQL eu posso ter dois usuários ao mesmo tempo, por exemplo, e se por acaso de houver duas transações em paralelo, elas não devem interferir uma na outra

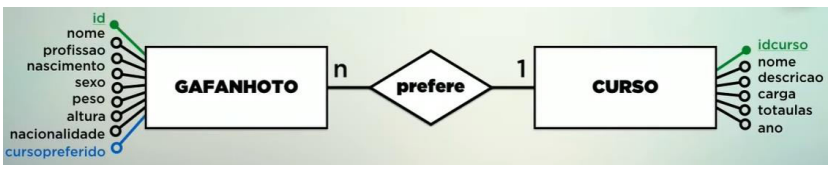
D → DURABILIDADE – ela deve durar o tempo que for necessário. Exemplo: salvei um dado de um cliente: eu quero que este dado fique lá o tempo necessário que eu precise dele

→ O MyISAM não possui a compatibilidade com esses 4 conceitos de transação!

→ No MySQL, caso eu não declarar a engine (a depender da versão), ele já considera a InnoDB!

Adicionando a foreign key (chave estrangeira)

Exemplo que iremos utilizar:



Exemplo de UM-PARA-MUITOS
cursopreferido é a minha chave estrangeira

```
ALTER TABLE gafanhotos
ADD COLUMN cursopreferido int;
```

Lembrando: a chave estrangeira tem que ser do mesmo tio da chave primária que veio do curso, que é do tipo INT

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
nome	varchar(30)	NO		NULL	
profissao	varchar(20)	YES		NULL	
nascimento	date	YES		NULL	
sexo	enum('M','F')	YES		NULL	
peso	decimal(5,2)	YES		NULL	
altura	decimal(3,2)	YES		NULL	
nacionalidade	varchar(20)	YES		Brasil	
cursopreferido	int	YES		NULL	

→ O que devemos fazer agora é dizer que esse curso preferido é uma chave estrangeira!

```
ALTER TABLE gafanhotos
ADD FOREIGN KEY (cursopreferido)
REFERENCES cursos(idcurso);
```

Isso significa que o **cursopreferido** da tabela **gafanhotos** está ligado com o **idcurso** da tabela de

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
nome	varchar(30)	NO		NULL	
profissao	varchar(20)	YES		NULL	
nascimento	date	YES		NULL	
sexo	enum('M','F')	YES		NULL	
peso	decimal(5,2)	YES		NULL	
altura	decimal(3,2)	YES		NULL	
nacionalidade	varchar(20)	YES		Brasil	
cursopreferido	int	YES	MUL	NULL	

Esse MUL é a representação de uma **chave múltipla** que, no caso, é uma chave estrangeira

→ Agora temos que cadastrar os cursos preferidos de cada aluno!

Exemplo: fazer com que o Daniel Morais (id = 1) prefira o curso de MySQL (idcurso = 6)

```
UPDATE gafanhotos
SET
    cursopreferido = '6'
WHERE
    id = '1'
```

id	nome	profissao	nascimento	sexo	peso	altura	nacionalidade	cursopreferido
1	Daniel Morais	Auxiliar Administrat	1984-01-02	M	78.50	1.83	Brasil	6

→ Em teoria, temos que dar um update para cada um dos alunos. Podemos fazer isso digitando o código de uma só vez ou então editar de forma manual!

Integridade referencial

→ Tentar apagar o curso de MySQL

```
DELETE FROM cursos
WHERE idcurso = '6';
```

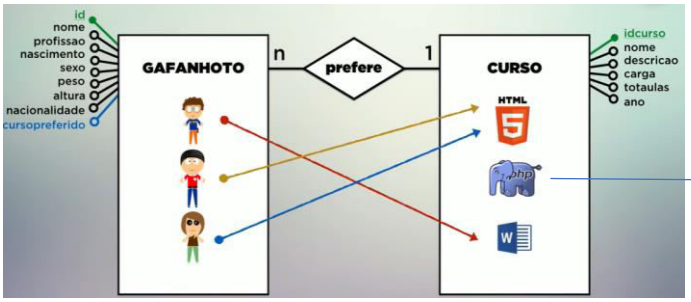
→ Ao executar este comando, o MySQL irá alegar um erro

```
8  17:29:09  delete from cursos where idcurso = '6'  Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`cadastro`.`gafanhotos`, ...
```

→ Esse erro foi um erro de **integridade referencial**

Isso acontece, pois, esse curso já tem relação com alguém. Se algum curso não tiver relação com ninguém, este poderia ser deletado

Exemplo:



O curso de php não tem relação com ninguém, então se eu quiser, ele pode ser deletado!
Agora, o de HTML5 e o de Word, eles não poderiam ser deletados

Outro exemplo:

id	nome	profissao	nascimento	sexo	peso	altura	nacionalidade	cursopreferido
1	Daniel Morais	Auxiliar Administrat	1984-01-02	M	78.50	1.83	Brasil	6
2	Talita Nascimento	Farmacêutico	1999-12-30	F	55.20	1.65	Portugal	22
3	Emerson Gabriel	Programador	1920-12-30	M	50.20	1.65	Moçambique	12
4	Lucas Damasceno	Auxiliar Administrat	1930-11-02	M	63.20	1.75	Irlanda	7
5	Leila Martins	Farmacêutico	1975-04-22	F	99.00	2.15	Brasil	1
6	Letícia Neves	Programador	1999-12-03	F	87.00	2.00	Brasil	8
7	Janaína Couto	Auxiliar Administrat	1987-11-12	F	75.40	1.66	EUA	4
8	Carlisson Rosa	Professor	2010-08-01	M	78.22	1.98	Brasil	5
9	Jackson Telles	Programador	1999-01-23	M	55.75	1.33	Portugal	3
10	Danilo Araujo	Dentista	1975-12-10	M	99.21	1.87	EUA	30

Nota-se que o curso 28 é preferido por ninguém, por exemplo

```
DELETE FROM cursos
WHERE
    idcurso = '28';
```

Como o curso 28 não tinha relação com ninguém, ele foi deletado!

27	Modelagem...	Curso de Modelagem de Dados	30	12	2020
29	PHP7	Curso de PHP, versão 7.0	40	20	2020
30	PHP4	Curso de PHP, versão 4.0	30	11	2010

→ Um problema: ao analisar a tabela, vemos que não temos o nome do curso em que o gafanhoto prefere, apenas o id do curso. Para isso, precisamos aprender sobre as **junções do SQL**

Junções

```
SELECT
    gafanhotos.nome,
    gafanhotos.cursopreferido,
    cursos.nome,
    cursos.ano
FROM
    gafanhotos
        JOIN
    cursos;
```

Aqui estou juntando gafanhotos com cursos

O que aconteceu:

nome	cursopreferido	nome	ano
Daniel Morais	6	PHP4	2010
Daniel Morais	6	PHP7	2020
Daniel Morais	6	Modelagem de Dados	2020
Daniel Morais	6	Magento	2019
Daniel Morais	6	Joomla	2019
Daniel Morais	6	WordPress	2019
Daniel Morais	6	After Effects	2018
Daniel Morais	6	Premiere	2017
Daniel Morais	6	SEO	2017
Daniel Morais	6	Segurança	2018

Mostrou o Daniel com todos os cursos, assim como os outros gafanhotos cadastrados

Acabou não juntando de forma inteligente: cada aluno vai ser junto com cursos

O que devemos fazer é **filtrar** essas coisas!

→ Para que eu crie um filtro, tenho uma cláusula sempre que eu utilizar o JOIN: **Cláusula ON**

```
SELECT
    gafanhotos.nome,
    gafanhotos.cursopreferido,
    cursos.nome,
    cursos.ano
FROM
    gafanhotos
        JOIN
    cursos ON cursos.idcurso = gafanhotos.cursopreferido;
```

Ligação da minha chave primária (idcurso) com minha chave estrangeira (cursopreferido)

nome	cursopreferido	nome	ano
Leila Martins	1	HTML5	2014
Jackson Telles	3	Photoshop5	2014
Janaína Couto	4	PHP	2015
Carlisson Rosa	5	Java	2015
Daniel Morais	6	MySQL	2016
Lucas Damasceno	7	Word	2016
Letícia Neves	8	Python	2017
Emerson Gabriel	12	C++	2017
Talita Nascimento	22	Premiere	2017
Andreia Delfino	22	Premiere	2017

Resultado final!

→ SEMPRE que eu utilizar o JOIN, eu DEVO utilizar o ON também, para dar sentido!

→ Nota-se também que no resultado não aparece o nome de todos os alunos, somente aqueles que tem relação com algum curso, isso porque quando eu utilizo um join (somente a palavra sozinha), eu estou fazendo um **INNER JOIN**, que é um join somente com as relações!

Eu posso escrever INNER JOIN no comando ou somente JOIN

OBS: eu posso trabalhar com apelidos de coluna!

```
SELECT
    g.nome,
    g.cursopreferido,
    c.nome,
    c.ano
FROM
    gafanhotos AS g
        JOIN
    cursos AS c ON c.idcurso = g.cursopreferido;
```

Comando **AS**

Além do INNER JOIN, temos o **OUTER JOIN**

→ Este, vai tratar dos conceitos do INNER junto com aqueles dados que não tem relação com nenhuma outra tabela!

```
SELECT
    g.nome,
    g.cursopreferido,
    c.nome,
    c.ano
FROM
    gafanhotos AS g
        JOIN
    cursos AS c ON c.idcurso = g.cursopreferido;
```

À esquerda do JOIN, temos a tabela **gafanhotos**

À direita do JOIN, temos a tabela **cursos**

→ Se eu quiser mostrar TODOS os gafanhotos, inclusive aqueles que não preferem nada:

```
SELECT
    g.nome, g.cursopreferido, c.nome, c.ano
FROM
    gafanhotos AS g
        LEFT JOIN
    cursos AS c ON c.idcurso = g.cursopreferido;
```

nome	cursopreferido	nome	ano
Daniel Morais	6	MySQL	2016
Talita Nascimento	22	Premiere	2017
Emerson Gabriel	12	C++	2017
Lucas Damasceno	7	Word	2016
Leila Martins	1	HTML5	2014
Letícia Neves	8	Python	2017
Janaína Couto	4	PHP	2015
Carlisson Rosa	5	Java	2015
Jackson Telles	3	Photoshop5	2014
Danilo Araujo	30	PHP4	2010

Valter Vilmerson	NULL	NULL	NULL
Allan Silva	NULL	NULL	NULL
Rosana Kunz	1	HTML5	2014
Josiane Dutra	NULL	NULL	NULL
Elvis Schwarz	NULL	NULL	NULL
Paulo Narley	NULL	NULL	NULL
Joade Assis	NULL	NULL	NULL
Nara Matos	NULL	NULL	NULL
Marcos Dissotti	NULL	NULL	NULL

→ Se eu quiser mostrar TODOS os cursos, inclusive aqueles que nenhum aluno prefere:

```
SELECT
    g.nome, g.cursopreferido, c.nome, c.ano
FROM
    gafanhotos AS g
        RIGHT JOIN
    cursos AS c ON c.idcurso = g.cursopreferido;
```

nome	cursopreferido	nome	ano
Leila Martins	1	HTML5	2014
Rosana Kunz	1	HTML5	2014
NULL	NULL	Algoritmos	2014
Jackson Telles	3	Photoshop5	2014
Janaína Couto	4	PHP	2015
Carlisson Rosa	5	Java	2015
Daniel Morais	6	MySQL	2016
Lucas Damasceno	7	Word	2016
Letícia Neves	8	Python	2017
NULL	NULL	POO	2016
NULL	NULL	POO	2016
NULL	NULL	Excel	2017
NULL	NULL	Responsivi...	2018
Emerson Gabriel	12	C++	2017
NULL	NULL	C#	2017
NULL	NULL	Android	2018
NULL	NULL	JavaScript	2017
NULL	NULL	PowerPoint	2018
NULL	NULL	Swift	2019