**INNODB**

CREATE TABLE produto

(cd\_produto INTEGER,

nm\_produto VARCHAR(30),

cd\_categoria INTEGER REFERENCES categoria (cd\_categoria),

PRIMARY KEY (cd\_produto)

);

**MYISAM**

CREATE TABLE produto\_myisam

(cd\_produto INTEGER,

nm\_produto VARCHAR(30),

cd\_categoria INTEGER REFERENCES categoria\_myisam(cd\_categoria),

PRIMARY KEY (cd\_produto)

)

ENGINE = MYISAM;

Não é recomendado que se misture os tipos de tabela, feitos com diferentes engines. Os comandos não funcionam corretamente alterando.

INSERT INTO categoria VALUES(1, ‘categoria 1’);

INSERT INTO categoria VALUES(2, ‘categoria 2’);

INSERT INTO produto VALUES(1, ‘produto 1’, 1);

INSERT INTO produto VALUES(2, ‘produto 2’, 2);

INSERT INTO produto VALUES(3, ‘produto 3’, 3); -- acusa erro semântico, utilizando o INNO DB

Sintaxe – na formulação das estruturas há erro. Semântica: ao tentar aplicar o código, alguma referência está inválida.

**SHOW ENGINES; -- mostra as engines disponíveis. Na padrão, estará escrito default (normalmente Inno DB)**

INSERT INTO categoria\_myisam VALUES (1, ‘categoria 1’);

INSERT INTO categoria\_myisam VALUES (2, ‘categoria 2’);

INSERT INTO produto\_myisam VALUES (1, ‘produto 1’, 1);

INSERT INTO produto\_myisam VALUES (2, ‘produto 2’, 2);

INSERT INTO produto\_myisam VALUES (3, ‘produto 3’, 3); semanticamente não gera ação de erro. Insere o dado na tabela produto, mesmo se a referência da categoria não existe.

DELETE FROM categoria\_myisam WHERE cd\_categoria = 1; - permite exclusão, mesmo tendo uma tabela filha que a tem como referência. Para o InnoDB, seria necessário primeiro excluir ou alterar a categoria da tabela filha e, somente depois, seria possível fazer a exclusão do registro da tabela mãe.

**AO DESENVOLVER UM PRODUTO, SEMPRE VERIFICAR SE A ENGINE UTILIZADA REALMENTE É O INNODB. MYISAM É PENSADO EM GRANDES VOLUMES DE DADOS E PERFORMANCE, MAS, ACABA NÃO FAZENDO AS VALIDAÇÕES. MYISAM NÃO POSSUI CONTROLE A NÍVEL DE BANCO, SOMENTE A NÍVEL DE APLICAÇÃO.**

**SHOW GLOBAL VARIABLES LIKE ‘default\_storage\_engine’; -- mostra todas as variáveis globais.**

Tabelas MyISAM criam 3 tipos de arquivo na pasta da database, dentro de data. Os arquivos InnoDb criam somente 2. Entretanto, A InnoDB também armazena arquivos na pasta data.

MyISAM – MYI – arquivo de índice, MYD – arquivo de dados, FRM – arquivo de header.

-- DROP TABLE categoria\_myisam;

-- poderia ser executado, a referência de produtos seria perdida.

**CSV**

CREATE TABLE categoria\_csv

(cd\_categoria INTEGER NOT NULL,

nm\_categoria VARCHAR(30) NOT NULL,

-- PRIMARY KEY (cd\_categoria) NÃO PODE SER CRIADA PRIMARY KEY PARA CSV.

) ENGINE = CSV;

TODOS OS CAMPOS DEVEM ESTAR PREENCHIDOS COMO NOT NULL.

Arquivo .frm é o header, .csv é o próprio arquivo de dados, .csm é arquivo de controle.

INSERT INTO categoria\_csv

SELECT \* FROM categoria;

Pode-se também exportar os dados. Botão direito na tabela – exportar banco de dados ou exportar o select realizado em algum formato específico.

CREATE TABLE produto\_csv ENGINE = CSV

AS

SELECT \* FROM produto;

-- PARA REALIZAR ESTA OPERAÇÃO, É NECESSÁRIO QUE TODOS OS CAMPOS DE PRODUTO SEJAM NOT NULL.

ALTER TABLE produto

MODIFY COLUMN nm\_produto VARCHAR(30) NOT NULL;

ALTER TABLE produto

MODIFY COLUMN cd\_categoria INTEGER NOT NULL;

CREATE TABLE produto\_csv ENGINE = CSV

AS

SELECT \* FROM produto;

-- AGORA A AÇÃO PODE SER EXECUTADA.

ALTER TABLE produto\_csv ENGINE=CSV;

**MEMORY**

CREATE TABLE categoria\_memory

(cd\_produto INTEGER PRIMARY KEY,

nm\_produto VARCHAR(30),

)

ENGINE = MEMORY;

INSERT INTO categoria\_memory VALUES (1,’categoria 1 na memória’);

INSERT INTO categoria\_memory VALUES (2,’categoria 2 na memória’);

Arquivo gerado é apenas o .frm

SELECT tabs.TABLE\_SCHEMA, tabs.TABLE\_NAME tabs.`ENGINE` --usa crase mesmo. Ou pode ser tirada a crase.

FROM information\_schema .`TABLES` tabs

WHERE tabs.TABLE\_SCHEMA = ‘bd\_teste\_1’;

CREATE TABLE teste1

(cd\_tste1 INTEGER PRIMARY KEY,

ds\_teste1 varchar(50)

)

ENGINE = MYISAM;

CREATE TABLE teste2

(cd\_tste2 INTEGER PRIMARY KEY,

ds\_teste2 varchar(50)

)

ENGINE = MYISAM;

INSERT INTO teste1 VALUES (1,’teste1 – A’);

INSERT INTO teste1 VALUES (2,’teste1 – B’);

INSERT INTO teste2 VALUES (1,’teste2 – A’);

INSERT INTO teste2 VALUES (2,’teste2 – B’);

**MERGE**

CREATE TABLE testes

(cd\_teste INTEGER PRIMARY KEY,

ds\_teste VARCHAR(50)

)

ENGINE = MERGE

UNION = (teste1, teste2)

INSERT\_METHOD = LAST;

INSERT INTO testes VALUES(5,’teste? - E’);

-- Comando irá inserir os dados na última tabela informada. Neste caso, tabela 2

SELECT\*FROM teste1;

SELECT\*FROM teste2;

SELECT\*FROM testes;

Tipo merge gera o .frm (header) e .mrg (tabelas que está unindo).

-- PARA SE UTILIZAR UMA TABELA DO TIPO MERGE, TODAS DEVEM SER DO TIPO MYISAM.

--Assemelha- se a uma VIEW.

UNION, INTERSECT e EXCEPT podem ser

UNION

Em alguns bancos, UNION elimina as redundâncias – oque estiver em ambos, mostra somente uma vez. UNION ALL (somente alguns bancos) mostra todos os registros de A e de B, mesmo que o mesmo registro esteja duplicado nas duas tabelas.

INTERSECT

Mostra somente oque é comum entre ambas tabelas.

EXCEPT

Mostra somente oque está na primeira tabela informada, porém, que não está na segunda também.

SELECT ds\_teste1 FROM teste1

UNION

SELECT ds\_teste2 FROM teste2;

-- Exclui registros repetidos

SELECT ds\_teste1 FROM teste1

UNION ALL

SELECT ds\_teste2 FROM teste2;

-- Não exclui registros repetidos

SELECT ds\_teste1 FROM teste1

INTERSECT

SELECT ds\_teste2 FROM teste2;

SELECT ds\_teste1 FROM teste1

EXCEPT

SELECT ds\_teste2 FROM teste2;