Autor: Luiz Fernando Antonelli Galati

Lista de comandos SQL

CREATE DATABASE nomedabase;

USE nomedabase;

```
CREATE TABLE tabela(
   coluna1 TYPE OUTRAS_COISAS,
   coluna2 TYPE OUTRAS_COISAS,
   ...
   coluna n TYPE OUTRAS_COISAS);
```

DESC tabela;

SELECT coluna1, coluna2, ..., coluna n FROM tabela WHERE condições;

INSER INTO tabela (coluna1, coluna2, ..., coluna n) VALUES (valor1, valor2, ..., valor n);

UPDATE tabela SET coluna1 = x, coluna2 = y, ..., coluna n = z WHERE condições;

DELETE FROM tabela WHERE condições;

DROP TABLE tabela;

ALTER TABLE tabela MODIFY COLUMN coluna TYPE OUTRAS_COISAS;

ALTER TABLE tabela ADD COLUMN coluna TYPE OUTRAS_COISAS;

ALTER TABLE tabela DROP COLUMN coluna;

SELECT coluna1, SUM(coluna2) FROM tabela GROUP BY coluna1 ORDER BY coluna1;

O comando JOIN

SELECT * FROM tabela1 JOIN tabela2 ON tabela1.chave_estrangeira = tabela2.chave_referenciada_pela_estrangeira;

[LF1] Comentário: Também funciona trocar as tabelas antes do "ON': SELECT * FROM tabela2 JOIN tabela1....

JOIN duplo, usado para unir uma 'tabela1' com uma 'tabela2' e uma 'tabela3', sendo que tabela1 contém chave estrangeira tanto para a tabela2 quanto para a 'tabela3':

SELECT * FROM tabela1 JOIN tabela2 ON
tabela1.chave_estrangeira_para_tabela2 =
tabela2.chave_referenciada_pela_estrangeira_para_tabela2 JOIN tabela3 ON
tabela1.chave_estrangeira_para_tabela3 =
tabela3.chave_referenciada_pela_estrangeira_para_tabela_3;

Podemos também usar o JOIN para, partindo de uma tabela1, chegar a uma tabela n a partir de chaves estrangeiras. Por exemplo, suponhamos que desejemos sair da tabela1 e chegar na tabela 5, sendo que tabela1 tem chave estrangeira para tabela2 (mas não tem para as outras), tabela2 tem chave estrangeira para tabela3 (mas não tem para as outras), tabela3 tem chave estrangeira para tabela4 (mas não tem para as outras) e tabela4 tem chave estrangeira para tabela5. Basta fazermos:

SELECT * FROM tabela1 JOIN tabela2 ON
tabela1.chave_estrangeira_para_tabela2 =
tabela2.chave_referenciada_pela_estrangeira_para_tabela2 JOIN tabela3 ON
tabela2.chave_estrangeira_para_tabela3 =
tabela3.chave_referenciada_pela_estrangeira_para_tabela_3 JOIN tabela4 ON
tabela3.chave_estrangeira para tabela4 =
tabela4.chave_referenciada_pela_estrangeira_para_tabela4 JOIN tabela5 ON
tabela4.chave_esntrangeira_para_tabela5 =
tabela5.chave_referenciada_pela_estrangeira_para_tabela5;

De forma menos precisa, mas mais simplificada, podemos escrever:

SELECT * FROM tabela1 JOIN tabela2 ON tabela1.chaveParaTabela2 = tabela2.chave JOIN tabela3 ON tabela2.chaveParaTabela3 = tabela3.chave JOIN tabela4 ON tabela3.chaveParaTabela4 = tabela4.chave JOIN tabela5 ON tabela4.chaveParaTabela5 = tabela5.chave;

Restrições (constraints)

Listar todas as restrições (constraints) de uma tabela

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE TABLE_NAME = "tabela";

Adicionar uma restrição (constraint) em uma tabela

ALTER TABLE tabela ADD CONSTRAINT OUTRAS_COISAS;

- **Exemplo de adição de Foreign Key**

Abaixo um exemplo de adição de uma Foreign Key numa coluna *colunaA* de uma tabela *tabela1*. A Foreign Key referencia a *coluna2* de *tabela2.*

ALTER TABLE tabela1 ADD CONSTRAINT nome_da_restricao FOREIGN KEY (colunaA) REFERENCES tabela2 (coluna2)

Remover uma restrição (constraint) de uma tabela

ALTER TABLE tabela DROP tipo_da_restricao nome_da_restricao;
ALTER TABLE tabela DROP INDEX nome_da_restricao;

Obs: o nome da restrição deve ser escrito sem aspas.

- **Exemplo de remoção de Foreign Key**

Abaixo um exemplo de remoção de uma Foreign Key cujo nome é fk_name:

ALTER TABLE tabela DROP FOREIGN KEY fk_name;
ALTER TABLE tabela DROP INDEX fk_name;

A função EXISTS()

Suponhamos uma tabela1 com diversas instâncias numeradas por um id. Suponhamos agora uma tabela2 que tem uma coluna que é chave estrangeira para o id da tabela1 (digamos que o nome dessa coluna seja tabela1_id).

Quero descobrir quais ids de tabela1 estão presentes (= existem) em tabela2 (mais especificamente, na coluna tabela1_id de tabela2). Para isso, basta executar o seguinte comando:

SELECT tabela1.id FROM tabela1 WHERE EXISTS (SELECT tabela2.id FROM tabela2 WHERE tabela2.tabela1_id = tabela1.id);

[LF2] Comentário: Podemos trocar o id por qualquer campo de tabela1 que desejarmos ou por *. Por exemplo: tabela1.nome, tabela1.email, *, etc

Alternativamente, poderíamos dizer: quero descobrir quais instâncias (= todos os campos das instâncias) de tabela1 estão referenciadas na tabela2 e fazer:

SELECT * FROM tabela1 WHERE EXISTS (SELECT tabela2.id FROM tabela2 WHERE tabela2.tabela1_id = tabela1.id);

Se eu quiser descobrir quais instâncias de tabela1 **não** estão referenciadas na tabela2, basta fazer:

SELECT * FROM tabela1 WHERE NOT EXISTS (SELECT tabela2.id FROM tabela2 WHERE tabela2.tabela1_id = tabela1.id);

Operadores UNION e EXCEPT

O operador UNION é usado para unir, no resultado de uma consulta, o resultado de duas outras consultas. Assim, se o conjuntoA for o conjunto resposta para uma consultaA e o conjuntoB for o conjunto resposta para uma consultaB, consultaA UNION consultaB terá como resposta o conjunto união entre o conjuntoA e o conjuntoB.

Exemplo: *SELECT nome FROM tabela1 UNION SELECT nome FROM tabela2*, para encontrarmos todos os nomes que estão presentes ou na tabela1, ou na tabela2, ou nas duas tabelas.

Já o operador EXCEPT é usado para excluir do resultado de uma consulta o resultado de outra consulta. Assim, se conjuntoA for o conjunto resposta para uma consultaA e conjuntoB for o conjunto resposta para uma consultaB, consultaA EXCEPT consultaB terá como resposta o conjunto de elementos que estão presentes em conjuntoA, mas não estão presentes em conjuntoB.

Exemplo: *SELECT nome FROM tabela1 EXCEPT SELECT nome FROM tabela2*, para encontrarmos todos os nomes que estão presentes na tabela1, mas não estão presentes na tabela2.

O comando IN

A instrução IN pode ser utilizada para substituir uma sequência de ORs, a fim de facilitar a escrita e leitura de uma consulta (query). Por exemplo, em vez de escrever

SELECT * FROM tabela WHERE coluna = “aaa” OR coluna = “bbb” OR coluna = “ccc” OR coluna = “ddd”;

é possível escrever

SELECT * FROM tabela WHERE coluna IN (“aaa”, “bbb”, “ccc”, “ddd”);

Cuidados gerais

- Ao utilizar o GROUP BY, pensar com calma em quais colunas utilizar para fazer o agrupamento. Preciso agrupar por uma só coluna, duas ou mais de duas?
- Sintaxe do HAVING quanto utilizado com GROUP BY e ORDER BY: GROUP BY... HAVING... ORDER BY