BANCO DE DADOS

CAP - AULA 4

ExPlica.net.br

DML-SQL

Data Manipulation Language

GRUPOS

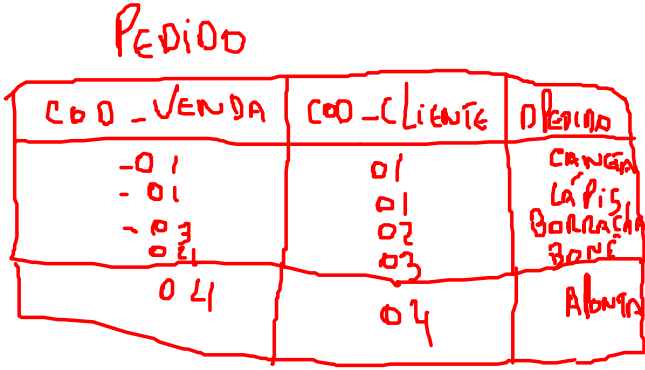
Uma característica muito importante do SQL é o poder de agrupar linhas com base em valores de determinadas colunas. Dessa forma, não estaremos trabalhando na pesquisa em todas as linhas da tabela, como fizemos anteriormente, mas sim em grupos menores. Para isso, utilizamos as funções de grupo já mostradas, com a cláusula GRUPO BY no comando SELECT. A cláusula GRUPO BY deve vir antes da cláusula ORDER BY e depois do WHERE (se houver necessidade de utilizá-los).

```
SELECT CODVENDA, COUNT(*) FROM PEDIDO
GRUPO BY CODVENDA;

SELECT CODCLIENTE, SUM(TOTALPEDIDO) FROM PEDIDO
GROUP BY CODCLIENTE;
```

```
SELECT CODCLIENTE, COUNT(*) FROM PEDIDO
WHERE PRECOVENDA < 15
GROUP BY CODCLIENTE;
```

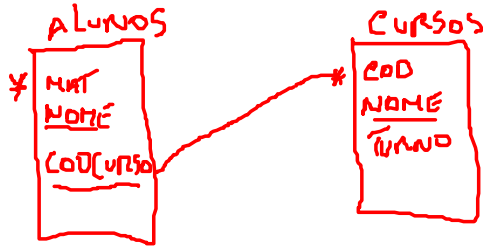
```
SELECT CODCLIENTE, COUNT(*) FROM PEDIDO
GROUP BY CODCLIENTE
HAVING PRECOVENDA < 15;
```



APELIDOS

```
SELECT a.nome, c.nome
FROM alunos AS a, cursos AS c
WHERE a.codCurso = c.cod;
```

```
SELECT a.nome, c.nome
FROM alunos a, cursos c
WHERE a.codCurso = c.cod;
```




```
SELECT ALUNOS.NOME, CURSOS.NOME
FROM ALUNOS, CURSOS
WHERE ALUNOS.COD_CURSO = CURSOS.COD;
```

GRUPOS

```
SELECT c.nome, COUNT(a.mat)
FROM alunos a
INNER JOIN cursos c
ON a.codCurso = c.cod
GROUP BY c.nome;
```

CURSOS | QUANT
NOME
SIN 35
SAD 40
SEF 50
...

```
SELECT a.nome
FROM alunos a
INNER JOIN cursos c
ON a.codCurso = c.cod
GROUP BY a.nome
HAVING COUNT(c.cod) > 2;
```

CONDIÇÃO

- ALEX 3
- JOÃO 5
- MARIA 4

INNER JOIN

```
SELECT *
FROM alunos AS a
INNER JOIN cursos AS c
ON a.codCurso = c.cod;
```

```
WHERE a.cidade = "Rio De Janeiro";
```

DCL-SQL

Data Control Language

GRANT

Privilégios para trabalhar com dados:	
Privilégio	*Descrição*
INSERT	Inserir dados em uma tabela
UPDATE	Atualizar dados em uma tabela
DELETE	Excluir dados de uma tabela
EXECUTE	Executar funções ou procedimentos armazenados
SELECT	Efetuar consultas em uma tabela

Privilégios para modificar a estrutura do banco de dados:

Privilégio	*Descrição*
CREATE	Criar tabela ou banco de dados
ALTER	Modificar uma tabela
DROP	Excluir uma tabela ou um banco de dados
CREATE VIEWS	Criar exibições
TRIGGER	Criar ou excluir um trigger em uma tabela
INDEX	Criar ou excluir um índice
CREATE ROUTINE	Criar uma função ou um procedimento armazenado
ALTER ROUTINE	Alterar ou excluir uma função ou procedimento armazenado

Privilégios Administrativos	
Privilégio	*Descrição*
CREATE USER	Criar contas de usuários
SHOW DATABASES	Ver os nomes dos bancos de dados no servidor
SHUTDOWN	Desligar o servidor
RELOAD	Recarregar as tabelas que armazenam os privilégios dos usuários dos bancos de dados. Assim elas são atualizadas se tiverem sido modificadas.
Outros privilégios	
ALL	Todos os privilégios disponíveis em um determinado nível, exceto GRANT OPTION
GRANT OPTION	Permite dar privilégios a outros usuários
USAGE	Não altera privilégios; usado para tarefas administrativas na conta do usuário.

Níveis dos privilégios

No MySQL os privilégios são atribuídos em quatro níveis diferentes:

- **Global** – O usuário tem acesso a todas as tabelas de todos os bancos de dados
- **Database** – Esse privilégio dá ao usuário acesso a todas as tabelas de um banco de dados específico
- **Table** – O usuário tem aceso a todas as colunas de uma tabela específica em um banco de dados
- **Column** – O usuário possui acesso apenas a colunas especificadas em uma determinada tabela.

Armazenando informações sobre privilégios

O MySQL utiliza tabelas especiais denominada **grant tables** para armazenar informações sobre os privilégios dos usuários, em um banco de dados interno de nome **mysql**. A tabela a seguir detalha essas tabelas de privilégios:

Tabela	Descrição
user	Armazena nomes e senhas de todos os usuários do servidor. Também armazena os privilégios globais que são aplicados a todos os bancos de dados do servidor.
db	Armazena privilégios dos bancos de dados
tables_priv	Armazena privilégios das tabelas
columns_priv	Armazena privilégios de colunas
procs_priv	Armazena privilégios de acesso a funções e stored procedures (procedimentos armazenados).

Usando a declaração GRANT para atribuir privilégios

Sintaxe:

```
GRANT lista_privilégios
ON [nome_banco.]tabela
TO usuário1 [IDENTIFIED BY 'senha1'],
usuário2 [IDENTIFIED BY 'senha2'] ...
[WITH GRANT OPTION]
```

```
GRANT USAGE
ON *.*
TO julia@localhost IDENTIFIED BY '1234';
```

Garantir acesso a um usuário de nome **julia**, sem privilégios:

```
GRANT ALL
ON *.*
TO alexandre IDENTIFIED BY '1234'
WITH GRANT OPTION
```

Dar privilégios globais a um usuário de nome alexandre:

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON db_biblioteca.*
TO ana@localhost;
```

Dar privilégios específicos para execução de comandos DML em todas as tabelas do banco db_biblioteca ao usuário ana:

```
GRANT ALL
ON db_biblioteca.*
TO ana@localhost;
```

Dar todos os privilégios no banco db_biblioteca à usuária ana:

```
GRANT SELECT, INSERT, UPDATE
ON db_biblioteca.tbl_autores
TO julia@localhost;
```

Garantir privilégios de inserção e atualização de registros e efetuar consultas na a tabela **tbl_autores** do banco de dados **db_biblioteca** ao usuário julia:

```
GRANT SELECT (nome_autor, sobrenome_autor), UPDATE (nome_autor)
ON db_biblioteca.tbl_autores
TO fabio@localhost;
```

Garantir o privilégio de consultar nomes e sobrenomes e alterar somente nomes dos autores (coluna nome-autor) da tabela tbl_autores do banco db_biblioteca ao usuário fabio:

```
SHOW GRANTS FOR fabio@localhost;
```

Para verificar:

REVOKE

Revogando privilégios com a declaração REVOKE

Podemos revogar (retirar) privilégios dos usuários usando a declaração **REVOKE**.

Sintaxe:

```
REVOKE lista_privilégios
ON objeto
FROM usuário1, usuário2, ...;
```

```
REVOKE DELETE
ON db_biblioteca.*
FROM ana@localhost;
```

revogar o privilégio de exclusão de dados no banco db_biblioteca à usuária ana:

```
REVOKE UPDATE (nome_autor)
ON db_biblioteca.tbl_autores
FROM fabio@localhost;
```

Retirando o privilégio de atualização da coluna nome_autor do banco db_biblioteca, na tabela de autores, do usuário fabio:

```
REVOKE ALL, GRANT OPTION
FROM alexandre, ana@localhost;
```

Remover todos os privilégios em todos os bancos de dados dos usuários alexandre e ana:

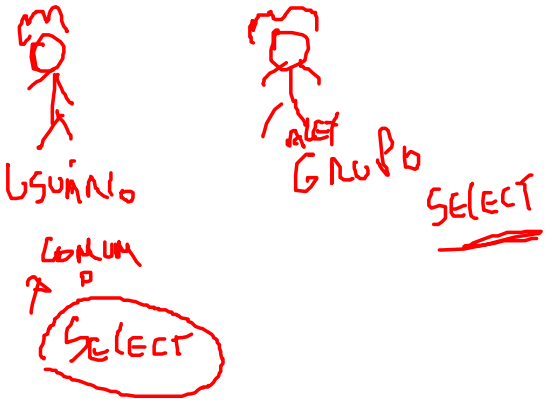
O comando é usado para impedir explicitamente que um usuário receba uma permissão específica.

DENY

```
DENY ALL ON alunos TO gerente;

DENY DELETE ON alunos TO secretaria;

DENY SELECT ON alunos TO diretor;
```



81 – Ao acessar uma base de dados MySQL, usando o comando *GRANT*, os privilégios são dados por meio de palavras reservadas, entre as quais podemos citar:

a) FILE, SHUTDOWN, UPDATE, INDEX, CREATE.
b) FILE, INSERT, PROCESS, CREATE, SORT.
c) FILE, SHUTDOWN, UPDATE, INDEX, MAIL.
d) INDEX, DOWNLOAD,UPDATE, DROP, CREATE.

50 – Assinale a alternativa que completa correta e respectivamente as lacunas da assertiva a seguir.

Na linguagem SQL, privilégio representa um tipo de acesso a _____ ou bases de dados listadas logo após a cláusula "ON" do comando "GRANT".

a) linhas, colunas
b) tabelas, registros
c) colunas, tabelas
d) linhas, tabelas

Modelos de Banco de Dados

Navegacional

Dados são armazenados em **registros** e podem ser gravados ou lidos de **forma conjunta**.

Esses registros são compostos por dados contínuos que podem **formar grupos**, dessa forma um registro pode ter uma construção hierárquica.

Modelos de Banco de Dados

Semiestruturado

Mais recentemente, surgiram bancos onde os dados são guardados e manipulados na forma **XML**, não mais em tabelas.

Alguns **bancos de dados relacionais** do mercado, criaram compatibilidade para tratar dados de forma **semiestruturada**.

Data warehouse

Surgido em **1980**, um **armazém de dados** é um sistema utilizado para armazenar informações sobre **atividades de uma organização** em bancos de dados de forma consolidada. Esses dados vêm dos **sistemas transacionais** (OLTP).

Com isso, é possível gerar **relatórios**, análises dos dados e tomar **decisões estratégicas** importantes.

Dados em um armazém **não podem ser voláteis** e geralmente **imutáveis** (somente leitura), salvo quando necessário fazer correções.

Data warehouse

BIG DATA

GRANDE BD

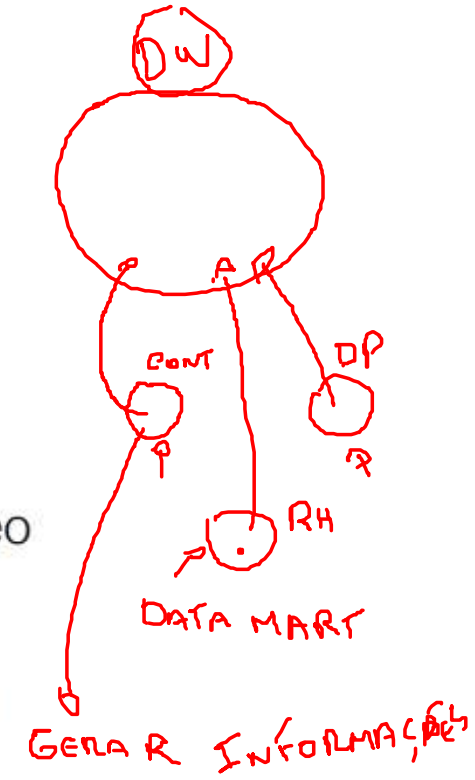
DATA MART

DATA MINNING

Para explorar um data warehouse, usamos uma **ferramenta OLAP** (Online Analytical Processing) ou **Processo Analítico em Tempo Real**.

As ferramentas data warehouse fazem parte do núcleo do mercado de **Business Intelligence** (BI).

Deve-se considerar também um **hardware** potente e **usuários** preparados para esse modelo.



DATA MINNING

MINERAÇÃO

Prospecção de dados ou mineração de dados (também conhecida pelo termo inglês data mining) é o processo de explorar dados à procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados.

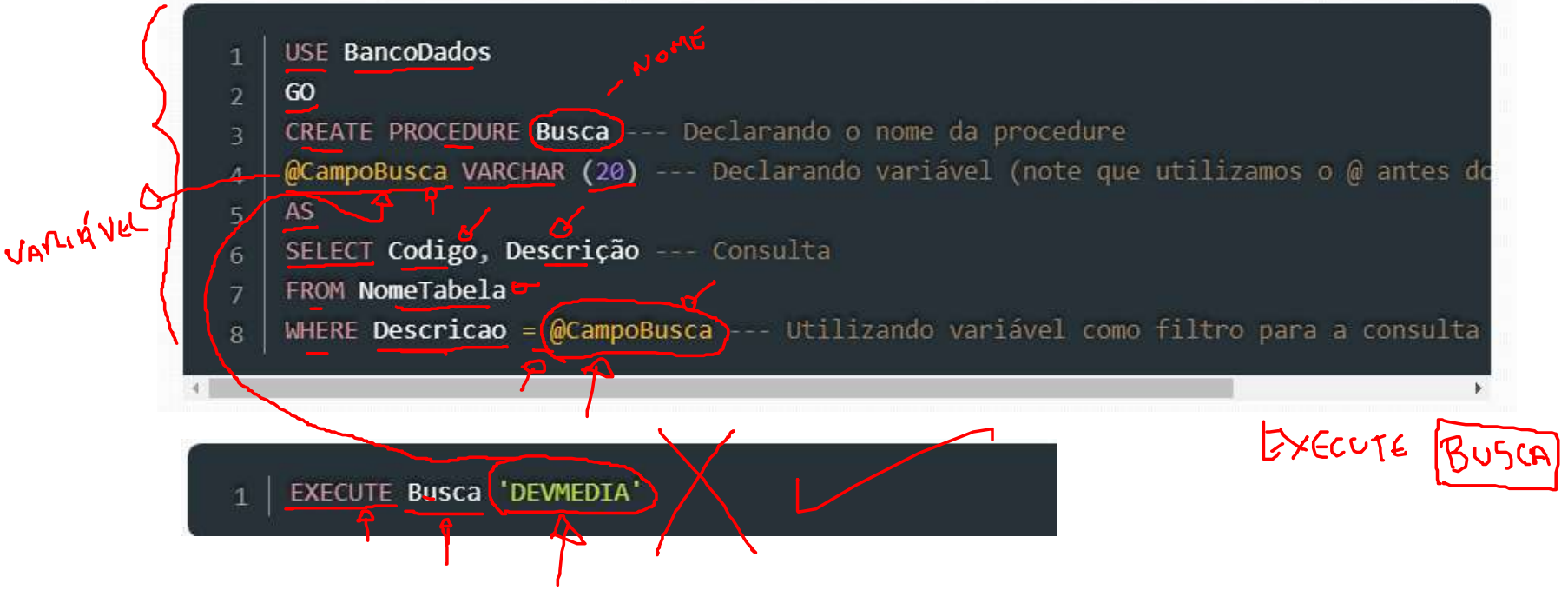
Data mart (repositório de dados) é sub-conjunto de dados de um Data warehouse (ou DW, armazém de dados). Geralmente são dados referentes a um assunto em especial (ex: Vendas, Estoque, Controladoria) ou diferentes níveis de sumarização (ex: Vendas Anual, Vendas Mensal, Vendas 5 anos), que focalizam uma ou mais áreas específicas.

Stored Procedure, que traduzido significa Procedimento Armazenado, é uma conjunto de comandos em **SQL** que podem ser executados de uma só vez, como em uma função. Ele armazena tarefas repetitivas e aceita parâmetros de entrada para que a tarefa seja efetuada de acordo com a necessidade individual.

Este é um exemplo de um stored procedure que executa uma consulta utlizando um filtro por descrição, em uma tabela específica de nosso banco de dados.

```
1  USE BancoDados
2  GO
3  CREATE PROCEDURE Busca --- Declarando o nome da procedure
4  @CampoBusca VARCHAR (20) --- Declarando variável (note que utilizamos o @ antes do
5  AS
6  SELECT Codigo, Descrição --- Consulta
7  FROM NomeTabela
8  WHERE Descricao = @CampoBusca --- Utilizando variável como filtro para a consulta
```

```
1  EXECUTE Busca 'DEVMEDIA'
```



Triggers são pequenas rotinas programadas no banco de dados, semelhantes às conhecidas Stored Procedures. Porém, como o nome indica (trigger = gatilho), uma Trigger pode ser disparada em resposta a um determinado evento. G

Onde se tem os seguintes parâmetros:

- nome: nome do gatilho, segue as mesmas regras de nomeação dos demais objetos do banco.

- momento: quando o gatilho será executado. Os valores válidos são BEFORE (antes) e AFTER (depois).

- evento: evento que vai disparar o gatilho. Os valores possíveis são INSERT, UPDATE e DELETE. Vale salientar que os comandos LOAD DATA e REPLACE também disparam os eventos de inserção e exclusão de registros, com isso, os gatilhos também são executados.

- tabela: nome da tabela a qual o gatilho está associado.

```
CREATE TRIGGER Tgr_ItensVenda_Insert AFTER INSERT
ON ItensVenda
FOR EACH ROW
BEGIN UPDATE Produtos SET Estoque = Estoque - NEW.Quantidade
WHERE Referencia = NEW.Produto; END$
```

TCL (Comando de Transação) COMMIT – Esse comando serve para confirmar uma ação dentro de uma banco de dados, por exemplo:

SAVEPOINT – Esse comando tem uma função parecida com a função “Ponto de Restauração” do Windows.

ROLLBACK – Esse comando é utilizado para desfazer as alteração até um ponto de restauração definido por SAVEPOINT.

```
BEGIN TRANSACTION
INSERT INTO tabela_teste VALUES (1);
SAVEPOINT meu_savepoint;
INSERT INTO tabela_teste VALUES (2);
ROLLBACK TO SAVEPOINT meu_savepoint;
INSERT INTO tabela_teste VALUES (3);
INSERT INTO tabela_teste VALUES (4);
COMMIT;
END;
```