Desenvolvimento de Sistemas

“Visão” (*view*): conceito, comandos de criação e manipulação, aplicação

Quem já trabalhou com administração de banco de dados como o MySQL ou qualquer outro com base de dados concentrada em SGBD (Sistema Gerenciador de Banco de Dados) conhece a rotina de escrever e reescrever consultas para fazer testes e verificações no sistema ou no próprio SGBD diariamente, repetindo algumas dessas consultas várias vezes por dia.

Pode-se precisar do resultado de várias tabelas relacionadas, na quais se deve fazer consultas com inúmeros *joins*, utilizando e conhecendo os índices aplicados para todas as tabelas relacionadas.

Outro fator que deve ser considerado é o desempenho do SGBD com inúmeros relacionamentos entre tabelas, pois a utilização de muitos relacionamentos interfere proporcionalmente no tempo de resposta da consulta SQL (*structured query language*).

Por outro lado, sabe-se que escrever a mesma consulta SQL exige que as ordens em que aparecem as tabelas, as ordens de campos e ordenação sejam rigorosamente iguais em todas as consultas, pois a alteração de qualquer ordem pode afetar diretamente a saída dos resultados.

Todos esses quesitos implicam na perda ou no ganho de desempenho, e é com relação a este que se começa a pensar em um recurso chamado “visão”, ou *view*.

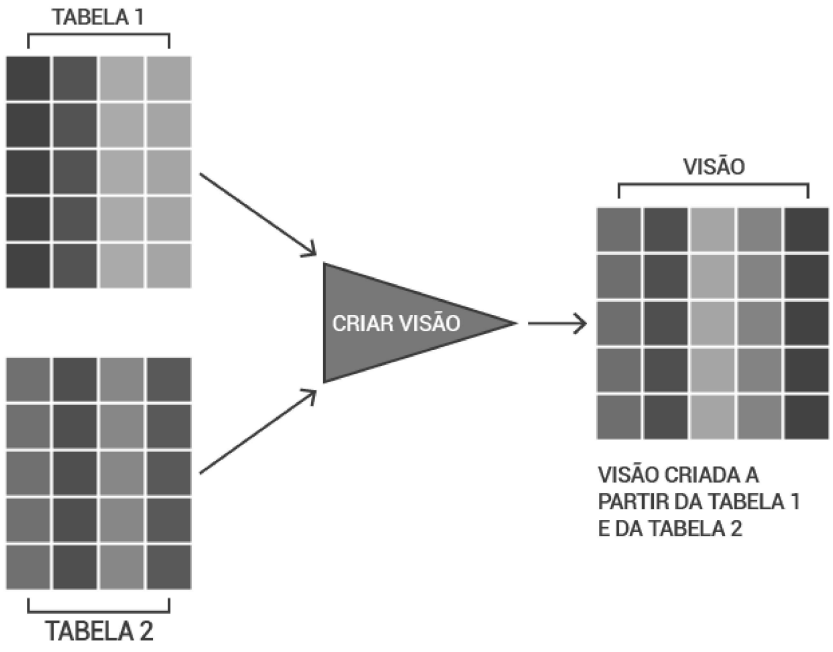

Figura 1 – Visões, ou *views*, formadas a partir de consultas que utilizam uma ou mais tabelas
Fonte: <https://www.datacamp.com/community/tutorials/views-in-sql>. Acesso em: 16 mar. 2022.

O que é a *view*?

Em síntese, pode-se descrever as *views* como **tabelas virtuais** baseadas no conjunto de resultados de uma instrução SQL, que contêm linhas, colunas e campos como os de uma tabela real. Uma *view* simplifica a estrutura de banco de dados para quem a utiliza sem a definição ou a estrutura de uma tabela, representando apenas uma consulta dos campos de uma ou mais tabelas reais no banco de dados.

Para compreender melhor a questão do desempenho, imagine fazer uma consulta SQL em uma tabela do banco de dados com 500.000 linhas e composta por 60 colunas. Nessa consulta específica, devem ser listados apenas três campos de todas essas colunas. Imagine agora o custo computacional para que apenas os dados dos três campos sejam apresentados. Isso despenderá um tempo significativo!

É a partir desses conceitos e considerando o custo computacional que são criadas as *views*. São notáveis os argumentos para utilização de *views* em projetos e você já deve estar pensando na sua criação e utilização, não é mesmo?

Para construir e utilizar *views*, é muito importante que você já esteja familiarizado com as instruções SQL que foram aprendidas anteriormente, pois serão utilizados comandos de seleção para fazer a construção das *views*.

Criando uma *view*

A criação de uma *view* é extremamente simples, porém é preciso um banco de dados criado a fim de obter as informações necessárias para a execução do processo dessa criação.

Então, abra o Workbench ou o editor de sua preferência para acompanhar os *scripts* que virão a seguir.

Observe os comandos executados:

```
CREATE DATABASE Eventos;
USE Eventos;
CREATE TABLE participantes(
    idParticipante INT (9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(200),
    dt_nasc DATE,
    genero VARCHAR(2),
    logradouro VARCHAR(200),
    bairro VARCHAR(200),
    cidade VARCHAR(200),
    complemento VARCHAR(200),
    uf VARCHAR(2),
    cep VARCHAR(200),
    comunidade VARCHAR(200),
    participa VARCHAR(1),
    movimento VARCHAR(1),
    escola VARCHAR(200),
    sabendo VARCHAR(200)
);
```

Após a criação do banco de dados e da tabela, pode-se fazer a inserção de registros.

```
INSERT INTO participantes(
    nome, dt_nasc, genero, logradouro, bairro, cidade, complemento, uf, cep,
    comunidade, participa, movimento, escola, sabendo
) VALUES (
    "Amanda", "2000-10-10", "F", "Rua das Andradas", "Centro", "Porto Alegre", "Final da rua",
    "RS", "911111000", "Do bairro", "S", "S", "Dom Francisco", "Internet"),
    ("Ricardo", "2000-10-10", "M", "Rua das Nereidas", "Centro", "Porto Alegre", "Bloco C",
    "RS", "911000000", "Do bairro", "S", "S", "João XXIII", "Internet"),
    ("Amadeu", "2010-05-20", "M", "Rua Gal Osório", "Porto", "Porto Alegre", "Bloco C",
    "RS", "911009999", "Do bairro", "N", "N", "João XXIII", "Revista"),
    ("Teobaldo", "1995-05-01", "M", "Rua Félix da Cunha", "Partenon", "Porto Alegre", "Casa 2",
    "RS", "900009999", "Do bairro", "N", "N", "Dohms", "Jornal"),
    ("Cremilda", "1980-05-01", "F", "Rua Silvério Souto", "Teresópolis", "Porto Alegre", "Casa 2",
    "RS", "946309999", "Do bairro", "N", "N", "Padre Pio", "Internet"
);
```

Tente aumentar a inclusão de registros para 20 linhas, seguindo o exemplo.

Agora que os registros estão inseridos na tabela, pode-se fazer a consulta SQL para verificar a inserção dos registros, conforme segue:

```
Select * from participantes;
```

Neste momento, deve aparecer a listagem dos participantes inseridos no banco de dados, conforme tabela a seguir:

1	Amanda	2000-10-10	F	Rua das Andradas	Centro	Porto Alegre	Final da rua	RS	911111000	Do bairro	S	S	Dom Francisco	Internet
2	Ricardo	2000-10-10	M	Rua das Nereidas	Centro	Porto Alegre	Bloco C	RS	911000000	Do bairro	S	S	João XXIII	Internet
3	Amadeu	2010-05-20	M	Rua Gal Osório	Porto	Porto Alegre	Bloco C	RS	911009999	Do bairro	N	N	João XXIII	Revista
4	Teobaldo	1995-05-01	M	Rua Félix da Cunha	Partenon	Porto Alegre	Casa 2	RS	900009999	Do bairro	N	N	Dohms	Jornal
5	Cremilda	1980-05-01	F	Rua Silvério Souto	Teresópolis	Porto Alegre	Casa 2	RS	946309999	Do bairro	N	N	Padre Pio	Internet

Assim que você executar o comando **SELECT** no Workbench, visualizará todos os registros inseridos na aba **Result Grid.**

Com a criação do banco de dados e a inserção dos itens na tabela, será possível começar a criação da *view* com o comando:

```
CREATE VIEW
```

Esse é o comando básico para a criação, mas ele não pode ser executado sozinho, pois é necessário informar em qual tabela serão buscados os dados em conjunto com o comando **SELECT.**

A sintaxe para a criação de uma *view* é a seguinte:

```
CREATE VIEW nome_do_VIEW AS SELECT * FROM nome_tabela;
```

O **nome_do_VIEW** é o nome que você escolherá, entretanto, é muito importante que o nome seja condizente com a tabela, pois será mais fácil para todos os DBAs (administradores de bancos de dados) reconhecerem rapidamente o que é *view* e o que é tabela.

O **nome_tabela** é a tabela responsável pelos dados, ou seja, é a partir dela que serão extraídos os dados. Saiba que a consulta pode ser mais complexa, utilizando mais de uma tabela.

No trecho a seguir, será criada uma *view* simples para uma consulta sobre a tabela “Participantes”:

```
CREATE VIEW view_participante AS SELECT * FROM participantes;
```

No Workbench, ao lado esquerdo, você tem a aba chamada **SCHEMAS**, na qual constam todos os bancos de dados criados, as suas funções, seus procedimentos, as visualizações e as tabelas.

Pode-se abrir o diretório e acessar todas as informações criadas no banco de dados e fazer a certificação da criação da *view*.

Figura 2 – Visualização da *view* criada

A figura mostra a aba Schemas selecionada, composta de diversos diretórios e com o banco de dados “evento” selecionado. Está aberto o diretório “Views”, no qual se pode visualizar o nome “view_participante”, que é o view criado.

Para se certificar da criação da *view*, você pode também utilizar o comando:

```
SHOW TABLES;
```

Com esse comando, serão listadas todas as tabelas e *views* que constam no banco de dados.

É importante que você anote esse comando, pois ele costuma ajudar muito nas consultas SQL, principalmente quando se começa a trabalhar com um banco de dados novo.

Após se certificar de que a *view* foi criada, é hora de fazer a consulta dela com o comando:

```
SELECT * FROM VIEW_PARTICIPANTE;
```

Note que, após a consulta SQL, os dados da tabela “Participante” estão sendo mostrados como um espelho dos dados dessa tabela.

1	Amanda	2000-10-10	F	Rua das Andradas	Centro	Porto Alegre	Final da rua	RS	911111000	Do bairro	S	S	Dom Francisco	Interne
2	Ricardo	2000-10-10	M	Rua das Nereidas	Centro	Porto Alegre	Bloco C	RS	911000000	Do bairro	S	S	João XXIII	Interne
3	Amadeu	2010-05-20	M	Rua Gal Osório	Porto	Porto Alegre	Bloco C	RS	911009999	Do bairro	N	N	João XXIII	Revist
4	Teobaldo	1995-05-01	M	Rua Félix da Cunha	Partenon	Porto Alegre	Casa 2	RS	900009999	Do bairro	N	N	Dohms	Jornal
5	Cremilda	1980-05-01	F	Rua Silvério Souto	Teresópolis	Porto Alegre	Casa 2	RS	946309999	Do bairro	N	N	Padre Pio	Interne

Agora que você já sabe que a *view* é construída para ajudar principalmente no desempenho e na visualização dos dados, é importante que a sua consulta SQL na criação da *view* seja personalizada. Confira o exemplo:

```
CREATE VIEW view_participante2 AS SELECT nome, cidade, escola FROM participantes;
```

Neste novo cenário, está sendo criada a **view_participante2**, com a instrução de **SELECT SQL** buscando apenas três campos da tabela “Participantes”.

Observe a saída do comando **SELECT** da *view*:

```
Select * from view_parcipante2;
```

Saída que pode ser conferida na aba **Result Grid** do Workbench:

Amanda	Porto Alegre	Dom Francisco
Ricardo	Porto Alegre	João XXIII
Amadeu	Porto Alegre	João XXIII
Teobaldo	Porto Alegre	Dohms
Cremilda	Porto Alegre	Padre Pio

Agora, o resultado da consulta SQL tem apenas os campos que foram inseridos na instrução **SELECT** no momento da construção da *view*.

Pode-se fazer consultas normalmente como se faz em tabelas. Veja o exemplo a seguir:

```
Select nome from view_parcipante2;
```

Esta é a saída que pode ser conferida no **Result Grid** do Workbench:

Amanda
Ricardo
Amadeu
Teobaldo
Cremilda

Neste caso, a consulta SQL trouxe apenas o nome, conforme instrução do SQL.

Agora que você já sabe utilizar e entende o conceito de *view*, poderá fazer a construção de uma. Está pronto para este desafio?

- Crie um banco de dados chamado ESCOLA e uma tabela chamada ALUNO.
- Na tabela ALUNO, crie os campos **Id_aluno**, **nome**, **idade**, **e-mail**, **telefone1**, **telefone2**, **cidade**, **estado**, **cep**, **rua**, **bairro**, **situação**. Todos esses campos devem ter o tipo de dados recomendado.
- Após a criação da tabela, insira nela alguns registros.
- Construa a *view* utilizando os campos **nome**, **idade** e **e-mail**.
- Agora, faça consultas na *view* utilizando o comando **WHERE** e o campo **IDADE** como referência, ou seja, liste todos os alunos maiores de 18 anos.

Criando *view* com múltiplas tabelas

A lógica de criação de uma *view* utilizando múltiplas tabelas é a mesma e segue os mesmos passos. Você apenas deve se preocupar com a consulta SQL na hora da criação.

Novamente serão utilizados *scripts* SQL para a criação do banco de dados e das tabelas.

Execute os *scripts* no seu editor para validar e aprender na prática todos os comandos e sequências.

```
CREATE DATABASE Agencia;
USE Agencia;
CREATE TABLE Usuario (
    id INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR (128),
    login VARCHAR (128),
    senha VARCHAR (128),
    datanascimento DATETIME NOT NULL,
    PRIMARY KEY (id)
);
CREATE TABLE Destino (
    id INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR (128),
    origem VARCHAR (128),
    destino VARCHAR (128),
    atrativos VARCHAR (1024),
    saida DATETIME NOT NULL,
    retorno DATETIME NOT NULL,
    id_usuario INT,
    PRIMARY KEY (id),
    FOREIGN KEY (id_usuario) REFERENCES Usuario (id)
);
```

Depois de criados o banco de dados e as tabelas, deve-se persistir dados a fim de popular as tabelas para os demais testes.

Veja os *scripts* de inserção de dados:

```
INSERT INTO Usuario (nome, login, senha, datanascimento)
VALUES
('Ana', 'Ana', '123', '2000-10-10'),
('João', 'João', '123', '1995-05-05'),
('Tiburcio','Tiburcio','123','1975-02-23');

INSERT INTO Destino (nome, origem, destino, atrativos, saida, retorno, id_usuario)
VALUES (
'Rio de Janeiro', 'Curitiba', 'Rio de Janeiro', 'Cidade Maravilhosa', '2022-02-20', '2022-02-28', 2);
INSERT INTO Destino (nome, origem, destino, atrativos, saida, retorno, id_usuario)
VALUES (
'Londres', 'Curitiba', 'Londres', 'Cidade Inglesa', '2022-02-20', '2022-02-28', 1 );
INSERT INTO Destino (nome, origem, destino, atrativos, saida, retorno, id_usuario)
VALUES (
'Nova Zelandia', 'Curitiba', 'Londres', 'Ilha turistica', '2022-02-20', '2022-02-28', 3);
```

Com os dados inseridos nas tabelas, você poderá fazer a consulta SQL para verificar se eles estão aparecendo corretamente. Observe:

```
SELECT * FROM DESTINO;
```

Esta é a saída que pode ser conferida no **Result Grid** do Workbench:

1	Rio de Janeiro	Curitiba	Rio de Janeiro	Cidade Maravilhosa	2022-02-20	2022-02-28	2
2	Londres	Curitiba	Londres	Cidade Inglesa	2022-02-20	2022-02-28	1
3	Nova Zelandia	Curitiba	Londres	Ilha turistica	2022-02-20	2022-02-28	3

A tabela “Destino” está correta, então falta conferir a tabela “Usuario”. Observe:

```
SELECT * FROM USUARIO;
```

Esta é a saída que pode ser conferida no **Result Grid** do Workbench:

1	Ana	Ana	123	2000-10-10
2	João	João	123	1995-05-05
3	Tiburcio	Tiburcio	123	1975-02-23

A tabela “Usuário” também está com os dados inseridos.

Como você já deve ter percebido, existe a coluna **ID_USUARIO** na tabela “Destino”, ou seja, elas estão relacionadas.

Com a relação entre as tabelas, é possível construir um comando SQL para listar os itens das duas tabelas, utilizando o comando **INNER JOIN.**

Veja:

```
SELECT D.nome AS Pacote, D.Destino AS Cidade, U.Nome AS responsavel
FROM Destino AS D
INNER JOIN Usuario AS U
ON D.ID_usuario = U.ID
```

Esta é a saída que pode ser conferida no **Result Grid** do Workbench:

Londres	Londres	Ana
Rio de Janeiro	Rio de Janeiro	João
Nova Zelandia	Londres	Tiburcio

Perceba que aqui foi feita a junção dos itens das tabelas “Usuário” e “Destino”, mas a consulta é grande e sempre há mais chances de errar consultas com o comando **INNER JOIN.**

Agora, será construída uma *view* com a mesma instrução SQL e comparado o resultado, conforme segue:

```
CREATE VIEW VIAGEM AS
SELECT D.nome AS Pacote, D.destino AS Cidade, U.nome AS responsável
FROM destino AS D
INNER JOIN Usuario AS U
ON D.ID_Usuario = U.ID
```

Com a *view* criada, pode-se fazer a consulta dela, visualizando o resultado:

```
SELECT * FROM VIAGEM
```

Esta é a saída que pode ser conferida no **Result Grid** do Workbench:

Londres	Londres	Ana
Rio de Janeiro	Rio de Janeiro	João
Nova Zelandia	Londres	Tiburcio

Perceba que neste momento apenas foi criada a *view*. Em seguida, só um comando básico de **SELECT** trará todas as informações necessárias.

Outros comandos da *view*

As *views* são estruturas simples e não têm muitos comandos agregados, mas é possível utilizar o comando **ALTER** para fazer a modificação.

Veja:

```
ALTER VIEW nome_VIEW AS SELECT * FROM nome_outra_tabela;
```

Nesse comando, a *view* criada terá o mesmo nome, mas há uma mudança na tabela em que são consultados os dados.

Para fazer a exclusão de uma *view* do banco de dados, utiliza-se o comando **DROP**, conforme o que segue:

```
DROP VIEW nome_VIEW;
```

Encerramento

Neste material, você aprendeu sobre a utilização da *view* em termos práticos e teóricos, desde a preocupação com o custo computacional até as melhores consultas utilizando as *views*.

Com todas as instruções e os estudos até aqui, pode-se concluir que a *view* é um ótimo recurso quando se precisa otimizar consultas para oferecer o máximo de desempenho no banco de dados.

Certamente, você utilizará a *view* em breve no mercado de trabalho!