8 – Introdução

- Aplicações bem projetadas geralmente expõem uma interface pública e mantêm privados os detalhes de implementação, permitindo assim que mudanças futuras não afetem os usuários finais.

- Ao projetar um banco de dados, é possível obter um resultado semelhante, mantendo sua tabelas privadas e permitindo que seus usuários acessem dados apenas por meio de um conjunto de views.

- Uma view permite que os usuários visualizem os dados sem ter que necessariamente acessar as tabelas do banco de dados.

- Desta forma, é possível que o DBA controle os dados que os usuários irão acessar, bem como quais colunas da tabela.

8.1 – Definição

- Uma view é simplesmente um mecanismo de consulta de dados. Entretanto, diferentemente das tabelas, as views não armazenam os dados. Desta forma, não ocupam espaço em disco.

- Toda view criada no banco de dados possui um nome identificador. Por meio deste nome os usuários podem acessar a view e consultar dados normalmente, como se estivessem utilizando uma tabela (muitas vezes podem nem saber que estão consultando uma view).

- Como exemplo de utilização de uma view, em uma tabela de empregados pode ser necessário que o usuário não visualize o campo salário. Ou visualize apenas dados dos empregados do seu departamento.

- Desta forma, podemos criar uma view da tabela empregado que não contenha o campo salário, e mostre dados apenas dos empregados de um determinado departamento.

8.2 – Estrutura da View

- A primeira parte da view lista os nomes das colunas (das colunas da view, que não precisam ter necessariamente o mesmo nome das colunas da tabela).

- A segunda parte da view contém uma instrução SELECT, que definirá os critérios de consulta e deverá ter uma expressão para cada coluna da view.

- A Sintaxe abaixo representa a estrutura básica de uma view:

CREATE VIEW <nome da view>

(<campos da view>)

AS

<instrução **SELECT**>

8.2 – Estrutura da View

- Quando a instrução CREATE VIEW é executada, o SGBD armazena a definição da view para uso futuro.

- Nenhuma consulta é feita e nenhum dado é recuperado ou armazenado. Uma vez que o usuário efetua a chamada da view, a consulta é feita e os dados da tabela são recuperados e exibidos de acordo com a estrutura definida pela view.

- As views podem ser utilizadas para o desenvolvimento de relatórios e consultas. Algumas linguagens de programação possuem a capacidade de gerar automaticamente estas estruturas baseadas diretamente na view.

8.2 – Estrutura da View

- Caso seja necessário selecionar os campos id_empregado, nome e cargo da tabela de empregados. O Comando abaixo fará esta seleção:

```
SELECT id_empregado, nome, cargo FROM empregado
```

id_empregado	nome	cargo
1	José Cardozo	Presidente
2	Maria da Silva	Diretora Geral
3	Celio Guimarães	Compras
4	Joaquim Manoel	Estagiário
5	Manoel Alves	Consultor

8.2 – Estrutura da View

- Entretanto, caso o usuário deseje selecionar todos os dados, ele não terá dificuldades, basta colocar um * no lugar dos campos para que ele tenha acesso a todos os campos da tabela:

```
SELECT * FROM empregado
```

id_empregado	nome	data_contratacao	data_saida	id_departamento	cargo	salario	bonus
1	José Cardozo	2004-05-07	NULL	6	Presidente	10000	3000
2	Maria da Silva	2006-05-09	NULL	5	Diretora Geral	8000	1000
3	Celio Guimarães	2006-05-02	NULL	1	Compras	3000	500
4	Joaquim Manoel	2014-05-05	NULL	1	Estagiário	300	100
5	Manoel Alves	2014-06-04	NULL	3	Consultor	4000	500

8.2 – Estrutura da View

- Nem sempre pode ser viável dar acesso integral a todos os dados e colunas de uma tabela. Por esse motivo, é possível criar uma view somente com os campos de interesse:

```
CREATE VIEW empregado_vw
(Código,
Nome,
Cargo,
Salário)
AS
SELECT id_empregado, nome, cargo, salario FROM empregado;
```

8.2 – Estrutura da View

- Selecionando todos os campos da view, é possível observar seu funcionamento:

```
SELECT * FROM empregado_vw
```

Código	Nome	Cargo	Salário
1	José Cardozo	Presidente	10000
2	Maria da Silva	Diretora Geral	8000
3	Celio Guimarães	Compras	3000
4	Joaquim Manoel	Estagiário	300
5	Manoel Alves	Consultor	4000

8.3 – Operações de Filtro na View

- Também é possível que filtros sejam acrescentados nas views, para personalizar a consulta. Além dos filtros, é possível também especificar quais campos da view deseja-se que seja exibido como resultado na consulta.

- Entretanto, os campos que pertencem a tabela, mas não estão declarados na view, não poderão ser exibidos.

```
SELECT Nome, Salário FROM empregado_vw WHERE nome like 'M%' OR
Salário >=1000 ORDER BY Nome;
```

Nome	Salário
Celio Guimarães	3000
José Cardozo	10000
Manoel Alves	4000
Maria da Silva	8000

8.4 – View sem Declaração de Campos

- Para criação de uma view é possível que os nomes dos campos (rótulos da view) sejam omitidos. Desta forma, o comando SELECT que será responsável por determinar os campos que a view irá possuir, como ilustra o exemplo abaixo:

```
CREATE VIEW empregado_vw
AS
SELECT id_empregado, nome, salario FROM empregado;
```

id_empregado	nome	salario
1	José Cardozo	10000
2	Maria da Silva	8000
3	Celio Guimarães	3000
4	Joaquim Manoel	300
5	Manoel Alves	4000

8.5 – Operação de Join com View

- Uma view possui basicamente as mesmas características de uma tabela, sendo que ela pode ser agrupada, ordenada e até relacionada com uma tabela ou com outra view.

- Como exemplo, será feito uma junção entre a view cliente_vw e a tabela de cidade, onde as mesmas são relacionadas por meio do campo id_cidade, como ilustra a figura abaixo:

```
CREATE VIEW cliente_vw
(Código,
Nome,
id_cidade)
AS
SELECT id_cliente, nome, id_cidade FROM cliente;
```

8.5 – Operação de Join com View

- O comando abaixo faz uma junção da view cliente_vw com a tabela de cidade:

```
SELECT Código, Nome, cidade.nome FROM cliente_vw JOIN cidade USING(id_cidade)
```

Código	Nome	nome
1	Paulo Soares	Maringá
5	Hugo Menezes	Maringá
6	Johny Percebs	Maringá
7	Fred Kruger	Maringá
3	Paulo Nunes	São Paulo
4	Maria Célia Araujo	Floripa
2	Edmundo da Silva	Guarapuava

8.5 – Operação de Join com View

- Uma alternativa melhor ainda, seria criar a view cliente_vw já relacionada internamente com a tabela de cidades, como ilustra o exemplo abaixo:

```
CREATE VIEW cliente_vw
(Código,
Nome,
Cidade)
AS SELECT id_cliente, cliente.nome, cidade.nome FROM cliente JOIN cidade
USING(id_cidade)
```

Código	Nome	nome
1	Paulo Soares	Maringá
5	Hugo Menezes	Maringá
6	Johny Percebs	Maringá
7	Fred Kruger	Maringá
3	Paulo Nunes	São Paulo
4	Maria Célia Araujo	Floripa
2	Edmundo da Silva	Guarapuava

8.6 – View com Campos Calculados

- Em bancos de dados com um grande número de registros, as operações de consulta podem ficar pesadas, exigindo um alto índice de recursos computacionais, além do tempo de resposta ficar comprometido.

- As funções de agregação (SUM, AVG, MIN, MAX, etc.), as funções escalares, entre outras, costumam exigir altos índices de recursos computacionais. Desta forma, pode ser interessante criar uma tabela com os valores, que são gerados por meio de agregações, já calculados e armazenados no banco.

- Como exemplo, vamos analisar a tabela de empregado da figura abaixo, onde não existe um campo salário total, obrigando o banco a fazer o calculo toda vez que uma consulta é feita.

id_empregado	nome	data_contratacao	data_saida	id_departamento	cargo	salario	bonus
1	José Cardozo	2004-05-07	NULL	6	Presidente	10000	3000
2	Maria da Silva	2006-05-09	NULL	5	Diretora Geral	8000	1000
3	Celio Guimarães	2006-05-02	NULL	1	Compras	3000	500
4	Joaquim Manoel	2014-05-05	NULL	1	Estagiário	300	100
5	Manoel Alves	2014-06-04	NULL	3	Consultor	4000	500

8.6 – View com Campos Calculados

- A view abaixo já mostra valores do salário total do empregado calculados automaticamente, não sendo necessário que este calculo seja feito toda vez:

```
CREATE VIEW empregado_vw
(Código, Nome, Cargo, Salário_Total) AS
SELECT id_empregado, nome, cargo, FORMAT((salario+bonus),2) FROM empregado;
```

```
SELECT * FROM empregado_vw;
```

Código	Nome	Cargo	Salário_Total
1	José Cardozo	Presidente	13,000.00
2	Maria da Silva	Diretora Geral	9,000.00
3	Celio Guimarães	Compras	3,500.00
4	Joaquim Manoel	Estagiário	400.00
5	Manoel Alves	Consultor	4,500.00

8.6 – View com Campos Calculados

- Entretanto, caso a tabela empregado tenha uma quantidade muito grande de dados, a execução dessa view irá requerer um alto índice de recursos computacionais, visto que terá que calcular o salário para todos os empregados.

- Desta forma, uma técnica para agilizar as consultas de grandes tabelas, é a criação de uma tabela a partir de uma view. Esta tabela já armazenará os valores calculados, aliviando a carga do banco nos calculos.

```
CREATE TABLE empregado_salario AS SELECT * FROM EMPREGADO_VW
```

8.7 – Alteração de uma View

- Devido a estrutura da view ser mais complexa, não é possível adicionar ou excluir campos de forma simples como em uma tabela. Para alterar uma view é necessário, praticamente, reescrevê-la integralmente.

- Para alterar uma view, utilizá-se o comando ALTER VIEW, com a nova estrutura da view, como no exemplo abaixo:

```
ALTER VIEW cliente_vw
(ID,
Nome,
Cidade)
AS SELECT id_cliente, cliente.nome, cidade.nome FROM cliente JOIN cidade
USING(id_cidade)
```

8.8 – Exclusão de uma View

- Uma view pode ser excluída por meio do comando DROP VIEW seguido do nome da view que se deseja excluir, observando que nenhum dado será perdido, pois estes estão armazenados nas tabelas e não dentro da view.

- No caso da view empregado_vw, para excluí-la do banco de dados, pode-se utilizar o comando abaixo:

```
DROP VIEW empregado_vw
```

8.9 – Vantagens na Utilização de View

- Muitos desenvolvedores acreditam que a única função da view é servir como uma forma mais fácil de se efetuar uma consulta. Entretanto, ela possui diversas vantagens:

1. **Segurança dos Dados**: Se uma tabela for criada, e o usuário tiver permissão para consulta, ele será capaz de acessar esta tabela integralmente. Com uma view, é possível restringir quais campos e quais registros o usuário poderá acessar.

2. **Agregação de Dados**: Pode ocorrer de aplicações e relatórios requererem dados agregados, como soma de todas as vendas do mês por exemplo. Desta forma, o usuário teria que criar consultas mais complexas. Entretanto, é possível que se crie uma view com todos os dados (sejam calculados ou não), de forma que o usuário apenas chame a view e já tenha o resultado esperado.

8.9 – Vantagens na Utilização de View

3. Escondendo a Complexidade: Uma das razões mais comuns para se utilizar views é proteger os usuários finais da complexidade. Existem consultas que são extremamente complexas, precisando fazer uso de estruturas de agrupamento, subquery, entre outros recursos do SQL. Toda esta complexidade pode ser compilada em uma view, fazendo com que o usuário precisa apenas fazer a chamada desta para obter o resultado necessário.

4. Juntando Dados: Grandes bancos de dados possuem um alto número de tabelas. Por meio de uma view, é possível efetuar junções destas tabelas, facilitando as consultas.

8.9 – Vantagens na Utilização de View

- A principal função de uma view é permitir que os usuários possam recuperar dados de acordo com as suas necessidades, sem que necessariamente precisar ter acesso as tabelas.

- Entretanto, pode acontecer de um usuário ter a necessidade de efetuar algum tipo de alteração nos dados, ou então a inserção de algum registro. A maioria dos SGBD permitem que alterações nos dados sejam feitas por meio de view, desde que algumas condições sejam seguidas:

1. Não podem ser usadas funções de agregação (**AVG(), MIN(), MAX(), SUM()**, etc.)
2. A view não pode utilizar cláusulas **GROUP BY** ou **HAVING**.
3. Não podem existir subconsultas no **SELECT** e no **FROM**. E nenhuma subconsulta no **WHERE** que não se refira a tabelas da cláusula **FROM**.
4. Não pode utilizar **UNION, UNION ALL** ou **DISTINCT**.
5. A cláusula **FROM** deve usar junções internas se houver mais de uma tabela ou view.
6. Não pode atualizar colunas que são derivadas por meio de expressões.