1 - Introdução

- Durante a modelagem de um banco de dados é fundamental considerar no projeto a economia de espaço em disco, bem como obter o melhor desempenho futuro de aplicações que irão utilizar os dados armazenados.

- Entretanto, por falta de experiência dos desenvolvedores, ou mesmo por questão de cronograma apertado, alguns banco de dados são mal projetados, gerando inúmeros problemas, como lentidão e gasto excessivo de disco.

- Para evitar esse tipo de problemas, serão analisadas algumas situações relativamente simples, mas que podem melhorar de forma significativa o desempenho do banco de dados.

1.1 – Regras Básicas de um Banco de Dados

- Segue abaixo algumas regras básicas para um projetos de banco de dados. Algumas destas regras já são conhecidas a muito tempo, mas foram relaxadas recentemente em alguns SGBDs.

- Esse relaxamento de algumas regras não melhoraram em nada o desempenho do banco e, preferencialmente, devem ser desconsideradas.

1. **Chave Primária:** Alguns SGBDs permitem a criação de tabelas sem uma chave primária mas não há nenhuma boa razão para isso;

2. **Tipo de Dados da Chave Primária**: A chave primária pode ser de praticamente qualquer tipo de dados, mas não deve exceder o limite de tamanho imposto pelo tipo.

3. **Chave Primária Única:** Caso exista mais de uma possibilidade de chave primária, apenas uma deve ser escolhida. As demais possibilidades são são conhecidas como chaves candidatas;

4. Não Mudar a Chave Primária: Chaves primárias, uma vez criadas, não podem (ou não devem) ser mudadas;

5. Atualização de Valores em Chaves Candidatas: Chaves candidatas (chaves únicas adicionais) podem ter seu valor atualizado (update), pois não se espera que esta alteração tenha qualquer efeito nos relacionamentos com os registros em tabelas filho;

6. Relacionamento Através de Chave Estrangeira: As chaves estrangeiras em tabelas filho formam relacionamento através da chave primária da tabela pai. A tabela filho pode atualizar o valor de sua chave estrangeira para apontar para um registro diferente na tabela pai, mas um registro na tabela pai não pode (ou não deve) alterar o valor de sua chave primária, pois isso afetaria todos os registros da tabela filho a ele relacionados;

1.1 – Regras Básicas de um Banco de Dados

7. Nomes de Objetos: Todos os nomes para bancos de dados, tabelas e colunas devem utilizar o conjunto de caracteres padrão, que é composto de letras, números e sublinhados (underscore – "_"). A utilização de símbolos ou espaços não é uma boa prática.

8. Palavras Reservadas: Todos os bancos de dados contêm uma lista de palavras reservadas que não devem ser usadas nos nomes de banco de dados, tabela, colunas, etc. Alguns bancos de dados permitem que palavras reservadas sejam usadas em nomes de objetos se elas forem colocadas entre caracteres especiais, como aspas (“”).

9. Case Insensitive: Muitos bancos de dados são case insensitive quando se trata de nomes. Isto significa que nomes como “cliente” , “CLIENTE” ou “Cliente” referenciarão a mesma coluna (ou objeto). Entretanto, existem SGBD’s que diferenciam, portanto é importante padronizar o modelo, deixando tudo minúsculo ou maiúsculo.

1.2 – Regras Adicionais de um Banco de Dados

- Existem algumas regras complementares que melhoram o desempenho e a organização do banco de dados, são elas:

1. Coluna Extra para Ser Chave Primária: Se uma chave natural não é aparente, é permitido adicionar uma coluna para este fim. Inclusive, muitos projetistas consideram a melhor prática sempre adicionar um campo para ser chave primária (id_cliente por exemplo).

2. Nomes para as Tabelas: Todas as tabelas devem ser nomeadas no singular, não no plural. Assim, deve-se ter “PRODUTO” ao invés de “PRODUTOS”. A lógica para esta afirmação é porque se refere a ela como “a tabela de produto” e não “a tabela de produtos”.

3. Nomes das Colunas: Todas as colunas devem ter nomes significativos e não códigos curtos e ininteligíveis. Algumas convenções podem ser feitas, como “id_” ou “cod_” para todas as chaves primárias. Entretanto, o mais importante é padronizar.

1.2 – Regras Adicionais de um Banco de Dados

5. Case Insensitive: Para separar nomes de objetos que são compostos, procure usar o underscore (“_”). Por exemplo, use o nome “tipo_cliente” e não “TipoCliente”.

6. Convenção de Nomes, Tipos e Tamanhos: Sempre que possível, os campos com o mesmo conteúdo devem ter o mesmo nome. Por exemplo, não seria bom deixar o nome de um campo como “id_cliente” em uma tabela e “cod_cliente” em outra.

7. Frameworks: Alguns frameworks possuem algumas facilidades caso o nome dos objetos do banco de dados siga uma determinada sintaxe. Desta forma, é importante que seja estudado as regras básicas do framework que será utilizado antes da criação do banco de dados.

1.3 – Boas Práticas de Banco de Dados

- Existem diversas técnicas mais avançadas que também podem contribuir para o melhor desempenho do banco de dados, seguem algumas das principais:

1. **Chaves Técnicas Desnecessárias:** Se já existe um campo natural para ser a chave primária, pode não ser necessário a criação de uma coluna para esse fim, como as colunas "id" ou "cod". No caso da tabela abaixo, já existe um código padrão para cada país, que poderia perfeitamente ser a chave primária.

```
CREATE TABLE pais(
id_pais INT(11) NOT NULL AUTO_INCREMENT,
cod_pais_iso CHAR(3) NOT NULL,
nome VARCHAR(50) NOT NULL,
PRIMARY KEY (id_pais));
```

1.3 – Boas Práticas de Banco de Dados

2. Relacionamento N-N: Para resolver a questão de um relacionamento N:N, é preciso criar uma tabela intermediária, contendo como chave estrangeira as respectivas chaves primárias das 2 tabelas envolvidas no N:N. Um exemplo clássico é o relacionamento PEDIDO – PRODUTO, que precisa de uma tabela de ITEM_PRODUTO como mostrado no quadro abaixo:

```
CREATE TABLE ITEM_PEDIDO
(id_item INT(11) PRIMARY KEY AUTO_INCREMENT,
id_pedido INT(11) NOT NULL,
id_produto INT(11) NOT NULL,
data DATE NOT NULL,
...
```

1.3 – Boas Práticas de Banco de Dados

- A abordagem apresentada na anteriormente funciona bem, entretanto a coluna ID_ITEM poderia ser excluído, deixando como chave primária as colunas ID_PEDIDO e ID_PRODUTO (chave composta).

```
CREATE TABLE ITEM_PEDIDO
(id_pedido INT(11),
id_produto INT(11),
quantidade INT(11) NOT NULL,
preco DOUBLE NOT NULL,
PRIMARY KEY (id_pedido, id_produto));
```

1.3 – Boas Práticas de Banco de Dados

3. Relacionamento Um-para-Um: Apesar de não ser muito comum, eles acontecem quando uma entidade tem dados opcionais em outra tabela. Esse tipo de relacionamento deve ocorrer sempre que colunas opcionais, que raramente são preenchidas, existem em grande quantidade na tabela. O exemplo abaixo mostra a tabela de “CLIENTE”, onde algumas colunas são usadas de forma sazonal:

```
CREATE TABLE CLIENTE(
id_cliente INT(11) PRIMARY KEY,
nome VARCHAR(45),
endereco VARCHAR(45),
...
);
```

```
CREATE TABLE DADOS_EXTRA(
id_cliente INT(11) PRIMARY KEY,
time VARCHAR(45),
sogra VARCHAR(45),
cachorro VARCHAR(45)
);
```

1.3 – Boas Práticas de Banco de Dados

4. Padronização dos Nomes: O mesmo nome não deve ser usado para representar coisas diferentes, como no exemplo abaixo, onde o campo “marca” possui tipos distintos e representam campos diferentes:

```
CREATE TABLE MARCA
(id_marca INT(11) PRIMARY KEY AUTO_INCREMENT,
marca VARCHAR(255) NOT NULL
);


CREATE TABLE PRODUTO
(id_produto INT(11) PRIMARY KEY AUTO_INCREMENT,
...
marca INT(11),
CONSTRAINT FK_PROD FOREIGN KEY (marca) REFERENCES MARCA (id_marca)
);
```