Parte 1

PROJETO FÍSICO DE BANCO DE DADOS

A primeira etapa a fazer ao começa a trabalhar com banco de dados é estudar as técnicas utilizadas para se criar um projeto de banco de dados: Modelo Entidade-Relacionamento, Modelo Lógico usando ferramenta Case, Normalização, Estratégias de Projeto de Banco de Dados, etc.

Conhecedores dos conceitos de Codd, Chen, e outros autores que definiram o padrão relacional para banco de dados relacionais, é então, possível avançar mais um passo: implementar o banco de dados em uma ferramenta apropriada. Mas antes, é necessário conhecer alguns conceitos do Projeto Físico e dos Sistemas Gerenciadores de Banco de Dados especificamente.

O Modelo Físico de Dados é a especificação em SQL - *Structure Query Language* – Linguagem de Consulta Estruturada - do esquema relacional para o Sistema Gerenciador de Banco de Dados escolhido. Nesta fase, as estruturas de armazenamento e de recuperação de informações, bem como os mecanismos de acesso devem ser escolhidos, visando sempre o aprimoramento da performance dos aplicativos de Banco de Dados.

Nesta fase devem ser especificados não apenas as tabelas criadas, mas também os índices necessários, as restrições de integridade (*checks* e *triggers*), algumas operações de inclusão, exclusão e atualização de dados para cada tabela, bem como as consultas que a aplicação deve realizar.

1.1 FATORES QUE INFLUENCIAM O PROJETO FÍSICO DE BANCO DE DADOS

Esta fase do projeto de Banco de Dados objetiva não só propor uma apropriada estruturação de dados, mas fazê-lo de maneira que garanta um bom desempenho. Sendo assim, não é possível tomar decisões significativas sobre projetos físicos e análises de desempenho, até que conheçamos as consultas, as transações e as aplicações que devem ser executadas no banco de dados.

Dessa forma, vamos discutir alguns fatores:

a. Analisar as consultas e transações do Banco de Dados

É importante ter uma idéia das intenções de uso do banco de dados, definindo as consultas e transações que esperamos que sejam realizadas em alto nível, especificando o seguinte:

- Os arquivos que serão acessados pela consulta;

- Os atributos nos quais quaisquer condições de seleção para a consulta estejam especificadas;

- Os atributos nos quais quaisquer condições de junção ou condições para ligar múltiplas tabelas ou objetos para a consulta estejam especificadas;

- Os atributos cujos valores serão trazidos através da consulta.

Para cada operação ou transação de atualização, devemos especificar o seguinte:

- Os arquivos que serão atualizados;

- O tipo de operação em cada arquivo (*insert, update, delete*);

- Os atributos nos quais as condições de seleção para uma exclusão ou atualização estejam especificadas.

- Os atributos cujos valores serão alterados através de uma operação de atualização.

b. Analisar a freqüência esperada de solicitação (execução) de consultas e transações

É necessário considerar as taxas de solicitação (execução), usadas de forma estatística em situações práticas para grandes volumes de processamento.

c. Analisar as restrições de tempo de consulta e transações

Neste caso, algumas consultas e transações podem ter rigorosas restrições de desempenho. Por exemplo, uma transação que deve ser interrompida se não for concluída em 20 segundos.

d. Analisar as freqüências esperadas de operações de atualização

Um número mínimo de caminhos de acesso aos dados deve ser especificado para um arquivo que seja freqüentemente atualizado, uma vez que atualizar os próprios caminhos de acesso desacelera as operações de atualização.

e. Analisar as restrições de Unicidade em Atributos

Caminhos de acesso aos dados devem ser especificados em todos os atributos candidatos a chave ou chave primária. A existência de um índice facilita a pesquisa nos arquivos de dados, pois este define um caminho de dados.

Realizadas as análises iniciais, outra decisão importante é definir qual Sistema Gerenciador de Banco de Dados usar. É relevante levar em consideração vários fatores, como:

- O modelo físico, visto que determinará qual a carga de trabalho que será exigida pelo Sistema Gerenciador de Banco de Dados, isso já eliminará algumas opções.

- O custo. Uma locadora, por exemplo, irá adquirir uma licença Oracle ou SQL Server?

- Plataforma de Sistema Operacional. Esta é uma decisão que não será analisada neste momento.

1.2 SQL - *STRUCTURE QUERY LANGUAGE* - LINGUAGEM ESTRUTURADA DE CONSULTA

Como visto no módulo passado, é muito simples criar um *script* SQL através de uma ferramenta Case. No entanto, este não será o foco desta sessão. Aqui aprenderemos a criá-lo sem auxílio de ferramentas case. Para tanto, é necessário conhecermos a linguagem SQL.

Apesar do nome, fazer consultas a bancos de dados não é a única função de SQL. Ela é utilizada para **criar tabelas, inserir, excluir e alterar** dados no banco de dados, além de outras utilizações.

A SQL foi desenvolvida na década de 70 pela IBM em uma das primeiras tentativas de desenvolver um banco de dados relacional. Tornou-se padrão de fato depois de 1986, quando a ANSI (*American National Standards Institute*) endossou como **linguagem padrão para Banco de Dados relacionais** e desde então, já sofreu três atualizações oficiais: em 1989, 1992 e 1999. Nesta última, algumas das mudanças mais significativas estão relacionadas com a definição dos padrões para Banco de Dados Objeto-Relacionais.

Devido ao sucesso da forma de consultar e manipular dados da SQL, dentro de um ambiente de banco de dados, a utilização da SQL foi se tornando ainda maior. Cabe ressaltar que cada implementação de SQL possui uma adaptação da sintaxe para resolver determinados problemas, portanto, qualquer comando mostrado pode ser usado de forma diferente em um determinado Sistema Gerenciador de Banco de Dados.

Recomenda-se a leitura do manual do fabricante para maiores informações sobre o uso da linguagem SQL em Sistema Gerenciador de Banco de Dados comerciais.

Atualmente, a linguagem SQL assume um papel muito importante nos sistemas de gerenciamento de banco de dados, podendo ter muitos enfoques. Através de comandos SQL, é possível montar consultas poderosas sem a necessidade da criação de um programa, ou utilizar comandos SQL embutidos em programas de aplicação que acessam os dados armazenados.

Devido ao fato de possuir várias aplicações, a linguagem SQL provê suporte a várias funções. Que consiste em:

- DDL (Linguagem de definição de dados), onde os dados a serem armazenados são definidos e estruturados; (*Create, Alter e Drop*).

- DML (Linguagem de manipulação de dados), que permite a inclusão, remoção, seleção ou atualização de dados armazenados no banco de dados; Controle de acesso, permitindo proteção dos dados de manipulações não autorizadas; (*Select, Insert, Update, Delete, Commit e Rollback*).

- DCL (Linguagem de controle de dados) uma subclasse da DML responsável pelas permissões de acesso ao banco; (*Grant e Revoke*).

Restrições de Integridade, que auxiliam no processo de definição da integridade dos dados, protegendo contra corrupções, inconsistências e falhas do sistema de computação.

Além dessas características principais, ainda podemos citar:

- Visões, onde são especificadas as consultas disponíveis através de tabelas virtuais (*Views*).

- Gatilhos a fim de automatizar processos no banco (*Triggers*).

Outra característica é a capacidade de cancelar uma série de atualizações ou gravar depois de concluir uma série de atualizações.

1.3 CRIANDO UM BANCO DE DADOS

Esta etapa trata da *Data Definition Language* (DDL). O processo básico consiste simplesmente em atribuir tipo de dado e tamanho para cada um dos atributos que foram identificados.

Atenção:

- Cada Sistema Gerenciador de Banco de Dados adota um terminador de comando. O mais comum é o ponto e vírgula (;)

- Quanto aos comandos apresentados, os Sistemas Gerenciadores de Banco de Dados não são sensíveis a letras maiúsculas ou minúsculas. Contudo, **o conteúdo das colunas normalmente é sensível**, portanto muito cuidado.

1.3.1 Desnormalização de dados

Em uma análise mais profunda, é muitas vezes conveniente avaliar a necessidade de alguns campos redundantes na tabela. Esse processo é chamado de desnormalização de dados. Ocorre que, quando idealizamos o modelo de dados, utilizamos o conceito de processador perfeito, em que as informações são transmitidas sem custo instantaneamente. No entanto, mesmo bancos de dados avançados possuem limitações.

Esta técnica pode ser exemplificada pelo caso dos campos calculados que são eliminados na 3ª Forma Normal. Quando se tem um grande banco de dados que trata de Notas Fiscais, por questões de performance costuma-se “driblar” um pouco a regra e coloca-se o valor total da nota como um campo na tabela. Isso pode ser feito quando o banco de dados possui mecanismo de controle de transações, em que uma eventual gravação no banco de dados implique necessariamente a gravação na outra ponta. Esta operação deve ser feita por meio de gatilhos (*triggers*) para garantir a integridade das informações.

1.3.2 Definição de dados

Antes de criar as tabelas no banco de dados, é preciso definir quais são as características de cada um dos campos. As características que o SQL exige são o tipo do dado e o tamanho de cada campo. As informações aqui apresentadas são utilizadas pela maioria dos bancos de dados. Eventuais mudanças devem ser consultadas na documentação do Sistema Gerenciador de Banco de dados adotado.

TIPO DE DADO	DESCRIÇÃO
Integer ou Int	Número positivo ou negativo inteiro. O número de bytes utilizado varia de acordo com o banco de dados utilizado.
Smallint	Mesma função do Inteiro, mas ocupa cerca da metade do espaço

Numeric	Número positivo ou negativo de ponto flutuante. Deve-se informar tamanho do campo e a quantidade de casas decimais.
Decimal	Semelhante ao Numeric, mas, em alguns bancos de dados, poderá ter uma maior precisão após a vírgula.
Real	Número de ponto flutuante de simples precisão. A diferença básica é que os valores serão armazenados em representação exponencial, portanto será arredondado para o nível mais próximo de precisão.
Double Precision	Número de ponto flutuante de dupla precisão. Comporta-se como o Real, mas permite maior aproximação de resultados.
Float	Número de ponto flutuante em que você define o nível de precisão (número de dígitos significativos)
Bit	Armazenamento de um número fixo de bits. O número de bits deve ser indicado, do contrário o padrão será 1.
Bit Varying	Igual ao Bit, permitindo armazenar valores maiores. Normalmente, **utiliza-se para armazenar imagens**.
Date	Permite armazenar datas
Time	Permite armazenar horários
Timestamp	Permite armazenar uma combinação de data e hora
Character ou char	Permite armazenar cadeias de caracteres (letras, símbolos e números). O tamanho informado é fixo e indica o tamanho máximo da cadeia de caracteres.
Character Varying ou Varchar	Permite armazenar cadeias de caracteres, mas com tamanho variável. Neste caso, especifica-se o tamanho máximo da coluna. Se for utilizado menos espaço que o máximo definido, o espaço restante não será ocupado.
Interval	Intervalo de data ou hora

1.3.3 Adaptando o modelo de dados

Com base nas informações acima, adaptaremos nosso modelo de dados. Para nossos estudos utilizaremos o modelo lógico abaixo:

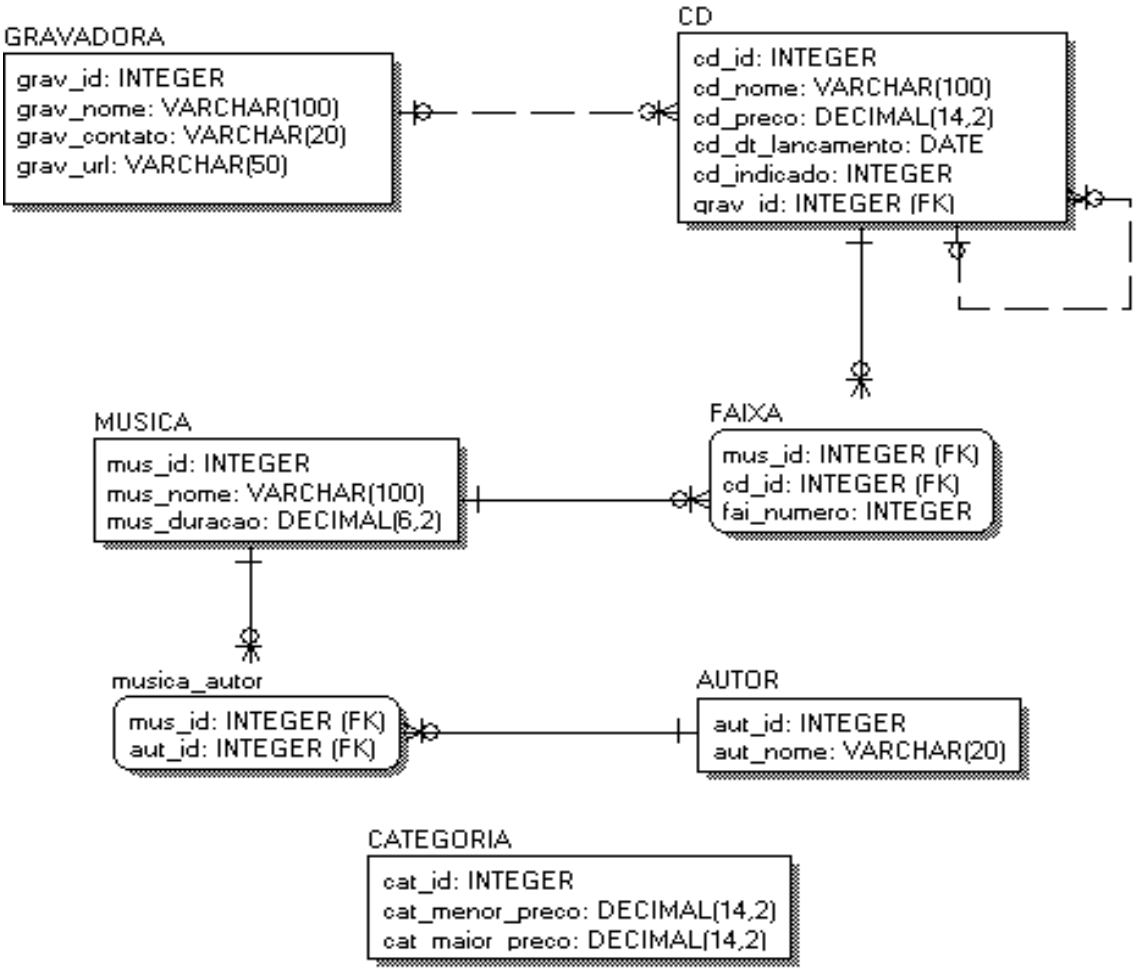
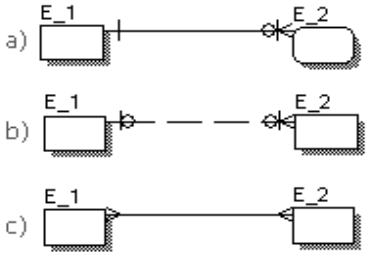


Figura 1- Modelo de dados

Notação dos relacionamentos



1:n – FK obrigatória. Parte da chave primária

1:n – FK não obrigatória

n:m – Muitos para muitos (cria-se a tabela associativa)

1.3.4 Criação de Tabelas

Tabelas são as estruturas mais importantes de um banco de dados. Nas tabelas estará o conteúdo que representa cada objeto do mundo real.

As tabelas criadas no banco de dados ficam armazenadas em outras tabelas internas do gerenciador de banco de dados denominadas **Dicionário de Dados**.

Sintaxe:	**Exemplo:**

`CREATE TABLE nome_da_tabela`
`(coluna1 tipo_de_dado constraint,`
`coluna2 tipo_de_dado constraint,`
`coluna3 tipo_de_dado constraint);`
	`CREATE departamento`
	`(dep_id INTEGER NOT NULL,`
	`dep_nome VARCHAR(100),`
	`dep_atividade VARCHAR(100),`
	`empr_id INTEGER NOT NULL);`

1.3.5 *Constraints* – Integridade Referencial – Restrição de Integridade (RI)

Constraints são regras agregadas a colunas ou tabelas. Assim, pode-se definir um campo como obrigatório ou não, ou aceitar apenas alguns valores predefinidos. No caso de regras aplicadas a tabelas, tem-se a definição de chaves primárias (PK) e estrangeiras (FK).

Um bom Sistema Gerenciador de Banco de Dados deve evitar a entrada de informação incorreta ou inconsistente em sua base de dados, garantindo, com isso, a qualidade da informação inserida. Uma restrição de integridade (RI) é uma condição especificada no esquema da base de dados para restringir a informação a ser armazenada.

As RI são especificadas e conferidas em dois momentos diferentes:

- Na especificação da RI: se dá na definição do esquema da base de dados pelo usuário ou pelo administrador da base de dados (DBA);

- Na conferência da RI: é feita pelo banco de dados toda vez que uma relação é modificada por uma aplicação sendo executada.

1.3.5.1 TIPOS DE *CONSTRAINTS* MAIS COMUNS

As *constraints* podem variar muito de um banco para outro.

a) Chave Primária: é a coluna identificadora de um registro na tabela. Para representá-la basta acrescentar a palavra chave *PRIMARY KEY* seguida do nome da coluna.

Exemplo:

```
...
PRIMARY KEY (cliente_id),
...
```

b) Chave Estrangeira: é o campo que estabelece o relacionamento entre duas tabelas. Dessa forma, deve-se especificar na tabela que contém a chave estrangeira quais são essas colunas e a qual tabela está relacionada.

Ao determinar este tipo de relacionamento, fica garantida a integridade das informações. Os valores presentes na coluna definida com chave estrangeira devem ter um correspondente na outra tabela.

Exemplo1:

```
...
FOREING KEY nome_chave (coluna)
      REFERENCES (tabela)
      ON UPDATE ação
      ON DELETE ação
```

Ação: Determina qual ação o banco de dados deve tomar quando for excluída ou alterada uma linha da tabela que contém referência a esta chave. Pode ser:

- *SET NULL*: Altera o conteúdo da coluna para nulo, perdendo a referência, sem deixar valores inconsistentes.

- *SET DEFAULT*: Altera o conteúdo da coluna pra um valor especificado na cláusula DEFAULT, se houver.

- *CASCATE*: Exclui ou altera todos os registros que se relacionam com a ele.

- *NO ACTION*: Em caso de alteração, não modifica os valores que se relacionam a eles.

- *RESTRICT*: Não permite a exclusão da PK.

Exemplo2:

Dept_id	Dept_nome	Dept_ramal
1	Informática	224
2	Compras	225
3	Pessoal	226

Fun_id	Fun_nome	Fun_admissao	Dept_id
1	Prachedes	12.12.2005	1
2	Nicodócio	21.02.2000	3

```
  CREATE TABLE funcionario (fun_id INTEGER, fun_nome VARCHAR(50),
fun_admissao DATE, dept_id INTEGER,
  PRIMARY KEY (fun_id),
  FOREIGN KEY (dept_id) REFERENCES departamento ON DELETE CASCADE ON
UPDATE NO ACTION
```

c) *DEFAULT*: Atribui um conteúdo padrão a uma coluna da tabela.

Exemplo:

```
...
prod_quantidade INTEGER DEFAULT 1,
...
```

d) *NOT NULL*: Indica que o conteúdo de uma coluna não poderá ser Nulo. Lembre-se, em banco de dados SQL, colunas sem valor atribuído possuem conteúdo Nulo.

Exemplo:

```
...
cliente_nome VARCHAR(50) NOT NULL,
...
```

e) *UNIQUE*: Indica que não pode haver repetição no conteúdo da coluna. Não é a mesma coisa que chave primária. A chave primária, além de não permitir repetição, não pode conter valores nulos, dentre outras característica não existentes nesta restrição.

Exemplo:

```
...
cliente_CPF NUMERIC(11) UNIQUE,
...
```

f) *CHECK*: Definição de domínio: Um domínio é uma expressão de valores possíveis para o conteúdo de uma coluna.

Exemplo:

```
...
sexo CHAR(1) CHECK (UPPER(sexo) = 'M' OR UPPER(sexo) = 'F'),
...
```

1.3.6 Alteração da estrutura da tabela

Para alterar a estrutura de uma tabela, utilizamos o comando ALTER TABLE.

1.3.6.1 ACRESCENTAR NOVAS COLUNAS

O comando utilizado para acrescentar novas colunas é muito semelhante ao da criação de colunas em uma tabela:

Sintaxe:

```
ALTER TABLE nome_da_tabela ADD
 coluna1 tipo_de_dado constraint,
 coluna2  tipo_de_dado  constraint,
...
```

Exemplo:

```
ALTER TABLE departamento ADD
dep_ramal NUMBER(4) UNIQUE
```

1.3.6.2 ACRESCENTAR NOVAS *CONSTRAINTS*

O comando utilizado para acrescentar novas *constraints* é muito semelhante ao da criação de *constraints* em uma tabela:

Sintaxe:	Exemplo:
ALTER TABLE nome_da_tabela ADD (constraint)	ALTER TABLE departamento ADD PRIMARY KEY (dep_id)

1.3.6.3 MODIFICAR COLUNAS

O comando utilizado para modificar qualquer característica de uma coluna.

Sintaxe:	Exemplo:
ALTER TABLE nome_da_tabela MODIFY (nome-coluna tipo_dado constraint)	ALTER TABLE departamento MODIDY dep_ramal INTEGER NOT NULL

Não é necessário repetir o que não estamos modificando.

1.3.6.4 EXCLUINDO ELEMENTOS

Pelo padrão SQL, deveria ser possível excluir colunas ou *constraints* de uma tabela. Alguns bancos de dados não permitem a exclusão de colunas.

Sintaxe:	Exemplo:
ALTER TABLE nome_da_tabela DELETE elemento	ALTER TABLE departamento DELETE dep_ramal ALTER TABLE departamento DELETE PRIMARY KEY

Abaixo um exemplo onde várias alterações estão sendo realizadas em um só comando:

```
ALTER TABLE fornecedores
ADD cgc NUMBER(14),
DROP tipofornecedor,
ADD CONSTRAINT ck_email CHECK (e_mail CONTAINING '@' OR e_mail IS NULL)
```

1.3.7 Eliminando uma Tabela

Para eliminar uma tabela do banco de dados, utilizamos o comando DROP TABLE seguido do nome da tabela. Alguns bancos somente permitirão esta operação se esta não estiver relacionada à outra tabela.

Sintaxe:	**Exemplo:**
`DROP TABLE nome_da_tabela`	`DROP TABLE departamento`

5. EXERCÍCIOS

Usando a Linguagem SQL crie o *Schema* de banco de dados para o seguinte modelo lógico: