

SQL – Introdução

Nesta aula inicial sobre Bancos de Dados em laboratório, apresentaremos os elementos básicos de SQL necessários para a realização das atividades desta disciplina. É o primeiro passo de uma sequência de aulas sobre o mesmo tema. Estes elementos tem a finalidade de experimentação e conhecimento da linguagem e também servem de apoio a outras atividades cujos objetivos não é a linguagem, mas outros aspectos relacionados com sistemas de bancos de dados.

Os operandos dos comandos da linguagem SQL são tabelas ou relações (no sentido matemático). Vejamos as suas propriedades.

1. Tabela

- Estruturas de dados nas quais os bancos de dados armazenam os seus dados.
- É um conjunto de tuplas (linhas). Cada tupla é uma sequência de valores contendo os dados elementares. Cada valor da sequência pertence ao domínio de valores do atributo correspondente. Os valores do i -ésimo elemento das sequências formam uma coluna da tabela.
- Tabela é o nome popular para o que, matematicamente, chamamos de relação. Este nome empresta o qualificativo “relacional” para bancos de dados que são organizados segundo esta estrutura. Ou seja, a palavra “relacional” não tem origem na expressão “relacionamento” entre os dados, uma vez que esta propriedade é compartilhada por bancos de dados de todos os modelos.
- No exemplo abaixo, a tabela é um conjunto de cinco triplas ordenadas. O i -ésimo elemento de cada tripla pertence ao mesmo domínio de valores.
- Como é um conjunto de tuplas, não existe ordem nas linhas de uma tabela, ao menos conceitualmente. Mesmo fisicamente, é melhor não pressupor uma possível ordenação porque vários fatores podem influenciar nesta propriedade, não somente a ordem em que os dados foram incluídos na tabela.
- Como são tuplas, os elementos de cada tupla formam uma sequência. É claro que, como sequência, os valores são ordenados.

Estrutura da tabela (diferente da tabela)

- Chamamos de esquema relacional ao que, popularmente, chamamos de estrutura da tabela.
- A estrutura da tabela é uma sequência de atributos.
- Os atributos estão associados aos respectivos domínios dos valores.
- Embora, por conveniência em alguns momentos, tratemos a sequência de atributos da estrutura de uma tabela como conjunto de atributos, esta situação deve ser explicitamente assinalada para evitar interpretações errôneas.

Incorreções

Estão incorretas as seguintes afirmações, pois elas contradizem a, pelo menos, uma das propriedades acima:

- Tabela é um conjunto de colunas.
- Tabela é um conjunto de atributos.
- Tabela é um conjunto de domínios.
- Tabela é uma sequência de linhas ou tuplas.

Exemplo de tabelas, na representação matemática e tabular:

$T = \{ (1, a, x), (2, b, y), (3, c, z) \}$

```
tb_pessoas={
  ('C001','Letsgo Daqui','30/04/2005','Marte'),
  ('C002','Hericlapiton dos Santos','18/06/1999','Jijoca de Jericoacoara'),
  ('C003','Méki Um Dois e Três','04/09/2001','Uauá'),
  ('C004','Simplicio Simplório Ains Tain','01/01/2000','Saturno'),
  ('C005','Índia Ana Jones Brasil','27/11/1985','Plutão')
  ('C006','Cosette Fantine de Myriel Valjean','29/03/1994','Urano')
  ('C007','Sofia Virgilia Cubas de Quincas Borba','19/10/1986','Netuno')
}
```

Visualmente, a tabela “tb_pessoas” fica mais apresentável da seguinte forma, e sempre que possível, vamos utilizar este formato:

codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplicio Simplório Ains Tain	01/01/2000	Saturno
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno

Tabela tb_pessoas

2. Comando SQL para criação da tabela

O comando SQL que cria a estrutura preliminar da tabela acima é a seguinte:

```
create table tb_pessoas(  
    codigo char(10) not null,  
    nome varchar(30) not null,  
    data_nasc date not null,  
    cidade_natal varchar(20) not null  
)
```

A estrutura é preliminar porque outras propriedades precisam ser atribuídas ao esquema da tabela.

3. Especificação da restrição de chave primária

A restrição da chave primária indica que a lista de colunas associada a ela tem a propriedade de identificar univocamente cada “tupla” da tabela. Além disso, é uma lista minimal (diferente de mínima) com esta propriedade. Pode ocorrer de a chave primária ser constituída por mais que um atributo. Veremos, em outro ponto, os conceitos de “superchave”, “chave candidata” e chave primária.

```
alter table tb_pessoas  
    add constraint ct_pessoas_pk  
        primary key (codigo)
```

Como observação, a chave primária pode ser definida sem o nome da restrição (constraint”), ou seja, o trecho:

```
constraint ct_pessoas_pk
```

não é obrigatória, no entanto, o nome é fortemente indicado para que mensagens retornadas do banco de dados referentes às restrições sejam mais facilmente identificadas, caso contrário, o próprio gerenciador atribuirá um nome, nem sempre compatível com o significado, gerando dificuldades em futuras manutenções e na interpretação das mensagens reportadas pelo sistema gerenciador de banco de dados. Em alguns casos, a mesma especificação pode gerar nomes distintos em ambientes diferentes.

Uma forma alternativa de especificação da chave primária é fazê-la na própria criação da tabela, como mostrado a seguir:

```
create table tb_pessoas(  
    codigo char(10) not null,  
    nome varchar(30) not null,  
    data_nasc date not null,  
    cidade_natal varchar(20) not null,  
    constraint ct_pessoas_pk primary key(codigo)  
)
```

Embora a especificação da chave primária possa ser feita na própria coluna, esta não é a minha recomendação porque este método falha quando a chave for composta. Por exemplo, a criação da tabela abaixo, em geral, na maioria dos produtos falhará alegando que uma tabela pode ter somente uma chave primária.

```
create table t(
    c1      int      not null primary key,
    c2      int      not null primary key,
    c3      int      not null
)
```

De outro modo, seguindo o método recomendado, não haverá problema e poderá ser utilizado em todas as situações como mostrado abaixo:

```
create table t(
    c1      int      not null,
    c2      int      not null,
    c3      int      not null,
    constraint ct_t_pk primary key(c1, c2)
)
```

4. Domínio dos valores dos atributos.

Vamos falar rapidamente das propriedades das colunas que estão presentes no comando de criação da tabela acima. Outros aspectos não serão tratados neste momento.

O conjunto de valores que um atributo pode assumir é chamado de domínio de forma consistente com o sentido matemático de domínio de uma função. Podemos até mesmo entender que um atributo é uma função que mapeia cada elemento com o seu respectivo valor. A forma mais simples de definir o domínio dos valores de um atributo é através da especificação do tipo dos dados que armazenam.

Exemplos:

char	- sequência de caracteres com comprimento fixo.
varchar	- sequência de caracteres com comprimento variável.
smallint	- inteiro de 16 bits (-32768 a 32767).
int	- inteiro de 32 bits (-2^{15} a $2^{15} - 1$)
bigint	- inteiro de 64 bits (-2^{63} a $2^{63} - 1$)
decimal	- número decimal dependente da precisão desejada.
numeric	- número decimal dependente da precisão desejada.
float	- número em ponto flutuante (depende do sistema).
real	- número em ponto flutuante (depende do sistema).
date	- data
time	- horário
double precision	- número em ponto flutuante (depende do sistema).
LOB	- Armazenamento de imagens, grandes textos, etc. Pode receber outros nomes conforme o produto, tais Como: BLOB, CLOB, text, image

Existem outros tipos de dados disponíveis em vários produtos que não seguem a norma ANSI. No entanto, às vezes, é necessário utilizá-las de acordo com o produto. Exemplo: RAW, binary, datetime, timestamp, bit, byte, unichar, univarchar, etc.

A definição conceitual pode ser diferente do físico, pois este é específico das propriedades do ambiente em que opera. Por exemplo, o usuário informa que um determinado atributo é do tipo caractere com no máximo 30 posições, mas você precisa decidir se, no nível físico, será utilizado o tipo “char”, “varchar”, ou alguma variação do formato “Unicode” de tamanho fixo ou variável. Outro exemplo, o usuário informou que um determinado atributo é numérico inteiro que varia de 0 a 10000, mas você precisa escolher entre algumas possibilidades como “int”, “smallint” ou “decimal”. Esta escolha implica em conhecer as características do ambiente operacional, incluindo o SGBD.

Os domínios dos valores dos atributos podem ser mais restritivos do que os especificados genericamente pelos tipos de dados. Veremos à frente como essas restrições são especificadas.

5. Admissibilidade de valores nulos

O valor nulo é atribuído quando ele é indefinido, desconhecido, inaplicável ou não informado. Para que este valor especial seja destinado ao valor de uma coluna é necessário que esta propriedade seja associada a ela.

As colunas que admitem valores indefinidos, desconhecidos, inaplicáveis ou não informados devem ser seguidas da palavra reservada “NULL” que estabelece esta propriedade. Caso contrário, informar “not NULL”. A especificação da propriedade “NULL” ou “not NULL” não é obrigatória, mas como a interpretação em caso de ausência é dependente do sistema gerenciador do banco de dados ou de suas configurações, recomenda-se fortemente que seja especificada. Nas nossas aulas de laboratório, ela é obrigatória. As situações em que o valor NULL é empregado podem ser assim descritas:

- i) Desconhecido: O responsável pela informação sabe que existe, mas não sabe o valor.
- ii) Indefinido: O dado não existe, mas é passível de existir. Por exemplo, o número de passaporte para indivíduos que ainda não o solicitaram.
- iii) Inaplicável: O atributo não é aplicável a alguns elementos da tabela. Por exemplo, para as mulheres não é aplicável o número da carteira de reservista. Pelo menos, por enquanto.
- i) Não informado: Por exemplo, um cliente prefere não informar o valor da renda mensal numa pesquisa populacional, embora ele saiba o valor.

Considere a tabela tb_exemplo a seguir com as seguintes colunas:

Coluna	Tipo	Admite valores indefinidos
col1	int	Não
col2	int	Sim
col3	int	Não

É necessário especificar que a nova coluna admite valores nulos porque a tabela pode conter linhas e não é possível indicar os valores para a nova coluna neste momento. Alternativamente, pode-se especificar um valor “default”. Alguns produtos verificam a condição de tabela vazia e neste caso, admitem que a nova coluna seja declarada como NOT NULL.

Em alguns sistemas, é necessário colocar a lista das colunas entre parênteses.

Na especificação ANSI, a palavra chave “column” após “add” é opcional, mas alguns produtos podem requerer esta especificação sintática.

b) Alteração do domínio dos valores dos atributos e admissibilidade de valores indefinidos

Suponha que a coluna “nome” da tabela tb_pessoas foi declarada como “varchar(30)” e foi detectada a necessidade de alterar para varchar(50). Do mesmo modo, as colunas “estado_civil” e “renda_mensal” não podem admitir valores nulos. É claro que as alterações requerem que os dados atuais sejam compatíveis com novo domínio dos valores. Do mesmo modo, os dados das duas últimas devem ter sido acertados com valores definidos para que a alteração seja possível.

```
alter table tb_pessoas modify nome varchar(32),
                             estado_civil not null,
                             renda_mensal not null
```

c) Remoção de colunas:

```
alter table tb_pessoas drop nome_avo,
                          estado_civil,
                          renda_mensal
```

Remove as colunas especificadas da tabela tb_pessoas.

Na especificação ANSI, a palavra chave “column” após “drop” é opcional, mas alguns produtos podem requerer esta especificação sintática.

Como observação, tomando como exemplo o item da alteração do domínio dos valores dos atributos, o resultado final seria o mesmo se a seguinte sequência fosse executada:

```
alter table tb_pessoas modify nome varchar(32)
alter table tb_pessoas modify estado_civil not null
alter table tb_pessoas modify renda_mensal not null
```

Apesar de atingir o mesmo resultado final, aplique as alterações num único comando porque todas as verificações e eventuais alterações físicas são realizadas uma única vez. Dependendo do produto e do tipo de alteração, o processo pode ser demorado e isto justifica mais ainda o agrupamento das alterações.

Como citado em algumas partes do texto, alguns produtos requerem que o comando de alteração da tabela apresente a lista dos elementos entre parênteses.

8. Restrição de domínio

A especificação do domínio dos valores de um atributo por meio do tipo do dado pode ser muito abrangente em relação ao que se deseja. Por exemplo, suponha que o atributo “nota” da tabela de notas dos alunos comporte somente valores de 0 a 10. Neste caso, é visível que o tipo de dado é muito extenso para o que queremos representar. Isto é feito através de uma restrição de domínio (check constraint). Vamos apresentá-la através de exemplo. A mesma observação sobre a atribuição de nome à restrição também se aplica neste caso.

No seguinte exemplo, a restrição de domínio é apresentada na especificação da própria coluna.

```
create table tb_notas(  
    matricula    char(10)    not null,  
    cod_disc     char(7)     not null,  
    nota         smallint    not null  
                constraint ct_nota_ck check(nota >= 0 and nota <= 10),  
    ano         smallint    not null  
                constraint ct_ano_ck  check(ano between 1980 and 2025),  
    semestre    smallint    not null  
                constraint ct_semestre_ck check(semestre in(1,2)),  
    constraint ct_notas_pk primary key (matricula, cod_disc, ano, semestre)  
)
```

No exemplo abaixo, a restrição de domínio é feita na criação da tabela, mas na parte seguinte à especificação das colunas.

```
create table tb_notas(  
    matricula    char(10)    not null,  
    cod_disc     char(7)     not null,  
    nota         smallint    not null,  
    ano         smallint    not null,  
    semestre    smallint    not null,  
    constraint ct_nota_ck check(nota >= 0 and nota <= 10),  
    constraint ct_ano_ck  check(ano between 1980 and 2025),  
    constraint ct_semestre_ck check(semestre in(1,2)),  
    constraint ct_notas_pk primary key (matricula, cod_disc, ano, semestre)  
)
```

O que ocorre se uma restrição de domínio é adicionada após a criação da tabela e com dados já existentes. Não existe dúvida com relação à verificação da conformidade dos dados novos, mas a validação sobre os dados já existentes nem sempre é aplicada por conta da duração e do impacto sobre a performance que isto poderia causar. Portanto, informe-se sobre o modo de operação do seu sistema. No ambiente do laboratório, novas restrições de domínio validam dados novos. A conferência dos dados antigos deve ser feita antes ou depois por um mecanismo menos invasivo ao ambiente.

Inclusão de restrição de domínio após a criação da tabela. A explicação segue na forma de exemplo.

```
alter table tb_notas(  
    add constraint ct_nota_ck check(nota >= 0 and nota <= 10)
```

9. Valores default

Os valores que um atributo pode assumir constituem-se no seu domínio. Em alguns casos, pode ser eleito um valor deste conjunto como preferencial, quando nenhum valor for especificado. A este valor chamamos de “valor default” de um atributo. Por exemplo, o atributo “estado_civil” da tabela de estudantes tem como valor preferencial “S” (solteiro) e este será o valor quando nenhum for informado o que é justificável no universo estudantil. Analogamente, o estado de nascimento preferencial é “SP” visto que a instituição localiza-se no estado de São Paulo e a maioria dos estudantes é originária deste estado. A coluna “hora_inicio” armazena a hora em que as aulas do estudante têm início. Concluindo, a coluna “total_creditos” indica a quantidade total de créditos que os alunos precisam completar no curso.

Por exemplo:

```
create table tb_estudantes(  
    matricula      char(10)          not null,  
    nome           varchar(32)       not null,  
    data_nasc      date              not null,  
    estado_civil   char(1)           default 'S'   not null,  
    estado_nasc    char(2)           default 'SP'   not null,  
    hora_inicio    time              default '19:00' null,  
    total_creditos int              default 2000    not null,  
    constraint ct_estudantes_pk primary key(matricula)  
)
```

Colunas com valores “default” podem ser incluídas em tabelas já existentes da mesma forma como ocorre com as colunas sem esta propriedade. Exemplo:

```
alter table tb_estudantes  
    add período varchar(10)          default 'matutino'    not null
```

10. Colunas computadas (atributos derivados no nível conceitual)

Os atributos derivados assinalados na representação dos dados no nível Conceitual são especificados como colunas computadas (derivadas ou calculadas) na implementação física. Outras formas de mapeamento de atributo derivado do nível Conceitual para físico podem ser utilizadas, no entanto consideramos o atributo derivado como o modo mais imediato. Quando esta forma de mapeamento não for possível, outros mecanismos precisam ser utilizados.

As colunas computadas podem ser materializadas ou não. As do primeiro tipo são fisicamente armazenadas; as do segundo tipo (não materializadas) são virtuais e, deste modo, avaliadas cada vez que forem citadas. As colunas computadas materializadas são reavaliadas somente quando as colunas base da sua definição têm os valores alterados.

Existem sutilezas dependentes de produto às quais você precisa se atentar. Por exemplo, se as colunas são indexáveis e, em caso positivo, como eles se comportam quando são criados sobre colunas computadas virtuais (não materializadas); se as suas propriedades podem ser modificadas; como as expressões não determinísticas podem ser utilizadas nas definições de colunas computadas; etc. Algumas características são herdadas das expressões que especificam estas colunas, por exemplo, o tipo do resultado e também a admissibilidade de valores indefinidos.

A declaração em SQL que as efetivam estão nos exemplos abaixo, lembrando que eles não esgotam o assunto.

a) Colunas computadas materializadas:

```
create table tb_computed_column(  
    column1 int not null,  
    column2 int not null,  
    column3 compute column1 + column2 materialized,  
    constraint CT_computed_column_PK primary key(column1))
```

b) Colunas computadas não materializadas:

```
create table tb_computed_column(  
    column1 int not null,  
    column2 int not null,  
    column3 compute column1 + column2 not materialized,  
    constraint CT_computed_column_PK primary key(column1))
```

c) Para incluir uma coluna computada numa tabela existente, o procedimento é análogo ao de uma coluna regular.

```
alter table tb_computed_column  
    add column3 compute column1 + column2 not materialized
```

Observação: Existem outras variações que não estão aqui contempladas.

Um atributo derivado no nível conceitual deve ser protegido para evitar inconsistência. No nível lógico, na modalidade relacional, ele torna-se uma coluna computada e precisa manter esta propriedade de modo que é possível inserir diretamente um valor para a coluna. Assim, o valor não é informado e a lista de colunas deve ser fornecida. Alguns produtos podem admitir o comando de inserção sem a lista de colunas desde que a lista de valores seja compatível com a

lista implícita de colunas não computadas. Comportamento semelhante é apresentado no comando de “update” de forma que a coluna não pode ter o valor alterado diretamente.

11. Restrição de chave estrangeira

Este tema é tratado extensamente nas aulas de teoria, mas vamos rapidamente fazer alguns comentários sobre o assunto.

No nosso curso utilizamos o Diagrama de Entidades e Relacionamentos para representar os dados no Nível Conceitual, seguindo o método proposto pelo MER. O mapeamento desta representação para o nível Lógico depende do modelo com que os dados serão representados. A representação no nível Lógico será diferente se escolhermos, por exemplo, o modelo de redes (network / grafo), hierárquico ou relacional. O entendimento desta particularidade é um dos motivos que determina a diferenciação das representações Lógica e Conceitual e, sobretudo, a não representação de elementos presentes somente no modelo relacional no Nível Conceitual.

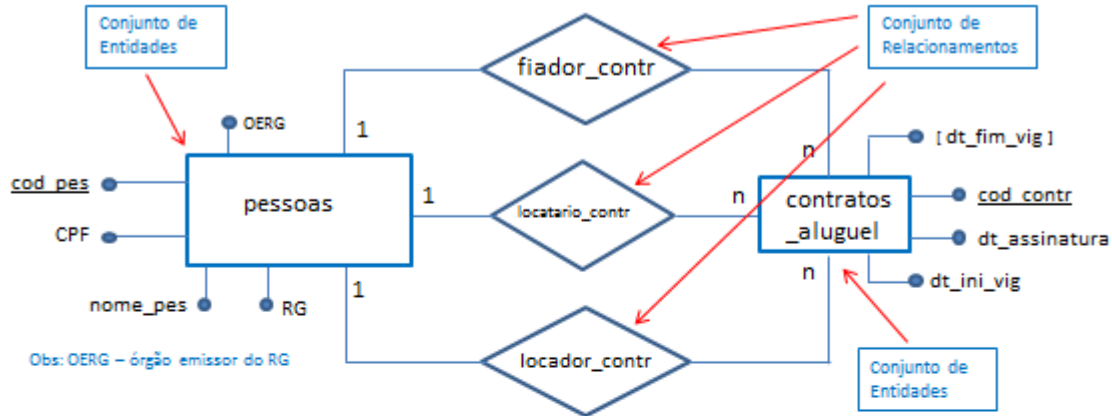
Quando a representação escolhida para o Nível Lógico for relacional e, somente neste caso, as relações (tabelas) receberão colunas que não estão presentes no Nível Conceitual como forma de estabelecer os relacionamentos. Em outros modelos, eles podem ser estabelecidos através de arestas de um grafo ou árvore.

A seguir, um exemplo de representação dos dados no Nível Conceitual e a correspondente derivação para o Nível Lógico na modalidade Relacional. A descrição refere-se a Contratos de Aluguel e as pessoas relacionadas na condição de Locatário, Locador e Fiador, supondo que, no universo do exemplo, as cardinalidades presentes na figura estejam corretas, embora elas possam ser alteradas desde que sejam mapeadas corretamente para o Nível Lógico.

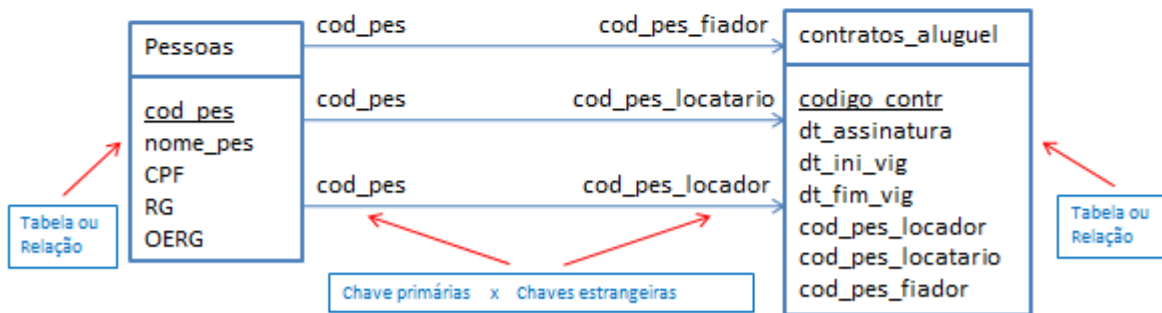
Como observação, em alguns contratos de aluguel a data final de vigência é facultativa.

Na figura, podemos ver o aparecimento de chaves estrangeiras que é o mecanismo através do qual os relacionamentos do Nível Conceitual são representados no Nível Lógico na modalidade Relacional. As chaves estrangeiras é uma restrição (“constraint”) no sentido de integridade dos dados, indicando que é uma referência válida.

Representação dos Dados no Nível Conceitual - DER



Representação dos Dados no Nível Lógico – Diagrama Relacional



Atenção: Na figura, as setas indicam a movimentação da Chave Primária para a relação na qual ela se transforma em Chave Estrangeira. Não representa a direção da referência como ocorre, na maioria das representações. Alternativamente, podemos representar o Diagrama Relacional por um conjunto de Esquemas Relacionais.

11.1. Especificação da restrição de Chave Estrangeira

Conforme a figura acima, as referências estabelecidas pelas chaves estrangeiras são especificadas na forma abaixo:

```
create table tb_pessoas (
  cod_pes          char(5)      not null,
  nome_pes         varchar(40)  not null,
  CPF              char(12)     not null,
  RG               char(10)     not null,
  OERG             char(10)     not null,
  constraint ct_pessoas_pk
    primary key(cod_pes)
)
```

```

create table tb_contratos_aluguel(
    cod_contr      char(10)      not null,
    dt_assinatura   date          not null,
    dt_ini_vig      date          not null,
    dt_fim_vig      date          null,
    cod_pes_locador  char(5)       not null,
    cod_pes_locatario char(5)     not null,
    cod_pes_fiador   char(5)       not null,
    constraint ct_contratos_aluguel_pk
        primary key(cod_contr),
    constraint ct_contratos_aluguel_x_pessoas_locador_fk
        foreign key(cod_pes_locador) references tb_pessoas (cod_pes),
    constraint ct_contratos_aluguel_x_pessoas_locatario_fk
        foreign key(cod_pes_locatario) references tb_pessoas (cod_pes),
    constraint ct_contratos_aluguel_x_pessoas_fiador_fk
        foreign key(cod_pes_fiador) references tb_pessoas (cod_pes),
)

```

Analogamente à especificação da chave primária, a chave estrangeira pode ser definida sem o nome da restrição (constraint”), no entanto, é fortemente indicada e nas aulas ele é obrigatório. Ou seja, o trecho:

```

constraint ct_contratos_aluguel_x_pessoas_locador_fk

```

não é obrigatório, mas o nome deve ser indicado para que todos os apontamentos associados a esta restrição sejam mais facilmente identificados. Sem isto, em geral, o Sistema Gerenciador de Banco de Dados fará a designação com um nome que pode ser estranho ao seu entendimento. Além disso, em alguns casos, o nome pode conter uma parte numérica gerada randomicamente fazendo com que a mesma tabela seja recriada com restrições com nomes distintos em diferentes momentos.

11.2. Especificação após a criação da tabela

É óbvio que para a especificação da chave estrangeira é necessário que a tabela referenciada tenha que ser criada anteriormente. Isto requer uma ordem na criação dos objetos do banco de dados. O problema maior é quando existe a referência cruzada. Em razão disto e para tornar a ordem irrelevante, a minha sugestão é que se criem primeiramente todas as tabelas com a especificação das chaves primárias e depois, todas as chaves estrangeiras, como mostra o exemplo a seguir para a tabela de contratos de aluguel.

```

create table tb_contratos_aluguel(
    cod_contr      char(10)      not null,
    dt_assinatura   date          not null,
    dt_ini_vig      date          not null,
    dt_fim_vig      date          null,
    cod_pes_locador  char(5)       not null,
    cod_pes_locatario char(5)     not null,
    cod_pes_fiador   char(5)       not null,
    constraint ct_contratos_aluguel_pk
        primary key(cod_contr)
)

```

```
alter table tb_contratos_aluguel add
    constraint ct_contratos_aluguel_x_pessoas_locador_fk
        foreign key(cod_pes_locador) references tb_pessoas (cod_pes),
    constraint ct_contratos_aluguel_x_pessoas_locatario_fk
        foreign key(cod_pes_locatario) references tb_pessoas (cod_pes),
    constraint ct_contratos_aluguel_x_pessoas_fiador_fk
        foreign key(cod_pes_fiador) references tb_pessoas (cod_pes)
```

Em alguns sistemas, os elementos adicionados precisam estar entre parênteses.

Entre as duas formas de especificação, no comando de criação da tabela ou fora dele, prefira a segunda maneira. Existem pelo menos duas razões:

- a) Podem ocorrer referências cruzadas quando então não será possível utilizar o primeiro método para a primeira tabela criada;
- b) Algumas vezes, deseja-se alterar ou corrigir a definição da chave estrangeira. Estando separada, a declaração pode ser reutilizada. De outra forma, será necessário um pequeno ajuste.

Na declaração de chave estrangeira o trecho:

```
references tb_referenced_table(y1, y1)
```

poderia ser substituída por

```
references tb_referenced_table
```

O resultado seria o mesmo porque as colunas referenciadas estão implicitamente definidas.

Em que situação você entende que seria necessário especificar as colunas da tabela referenciada? Seria por causa da diferença entre os nomes das colunas que referenciam das que são referenciadas? Não! Considere as seguintes possibilidades:

- a) As colunas referenciadas compõem uma chave candidata não primária;
- b) Por alguma razão, a ordem das colunas que referenciam é diferente da ordem das colunas na tabela referenciada. Isto, embora indesejável, pode ocorrer se novas colunas são inseridas posteriormente segundo as necessidades do negócio.

Repetindo o que foi dito, no caso da criação das chaves estrangeira, a recomendação é que se utilize a segunda modalidade para que se evite a dependência na ordem na especificação dos objetos. Além disso, às vezes, é impossível fazê-lo de outro modo se ocorre a dependência cruzada.

Seguem dois lembretes importantes sobre condições necessárias à especificação de chave estrangeira:

- a) Para que a restrição seja efetivada, o usuário que a especifica precisa ter a permissão específica de referência sobre a tabela referenciada. Isto ocorre em sistemas que permitem esquemas de diferentes proprietários e a chave estrangeira associa tabelas de esquemas distintos;
- b) É óbvio que o domínio de valores das colunas envolvidas deve ser o mesmo nas duas tabelas;

12. Declaração da restrição de chave candidata não primária:

A definição de chave primária está diretamente associada às definições de super chaves e chaves candidatas. Embora tenha visto que muitas pessoas não fazem, é recomendável que as chaves candidatas não primárias sejam especificadas. A sintaxe é a seguinte:

```
alter table tb_exemplo add constraint ct_nome_tabela_uk1  
unique (coluna1, coluna2,...)
```

Lembre-se que a chave candidata é uma propriedade do Nível Conceitual enquanto índices são elementos físicos. No entanto, é muito mais simples a utilização de índices para a verificação da condição de chave candidata com a propriedade de unicidade; é desta maneira que os Sistemas Gerenciadores de Banco de Dados se comportam criando automaticamente um índice sobre as colunas que compõem qualquer chave candidata (é obvio que isto inclui a chave primária). Então, qual a diferença entre chave candidata e índices com a propriedade de unicidade? Podemos fazer as seguintes considerações:

- a) Chave candidata é uma propriedade do Nível Conceitual e índice um elemento do nível físico porque a sua existência atende a questões de desempenho;
- b) Por definição de chave candidata que estudamos na disciplina de teoria, as colunas não podem ter valores nulos;
- c) Conceitualmente, a chave candidata é uma restrição (“constraint”) de integridade sobre os dados;
- d) O índice pode ser removido com comando próprio de remoção de índice, no entanto para excluir o índice associado à restrição de chave candidata não primária é necessário remover a restrição;
- e) Sob algumas condições específicas, um índice com a propriedade de unicidade pode ser especificado sobre colunas que admitem valores nulos, mas isto não é permitido na especificação de chaves candidatas, inclusive na chave primária que, como vimos, é uma das chaves candidatas;
- f) É importante ressaltar que estas considerações podem variar de acordo com os sistemas gerenciadores de bancos de dados.

Conforme as considerações acima, uma restrição de chave candidata não primária não deveria ser passível de exclusão pela remoção do índice, uma vez que esta é uma estratégia de implementação com vista ao desempenho, mas uma restrição é mais que um índice; é uma restrição conceitual de negócio, de modo que para excluí-la é necessário remover a restrição (constraint).

13. Remoção de constraint

Qualquer constraint de chave primária, estrangeira, candidata não primária ou de domínio é removida por comando próprio, supondo que devem ser mantidos a estrutura e os dados.

```
alter table tb_exemplo drop constraint ct_nome_tabela_coluna_ck
```

```
alter table tb_exemplo drop constraint ct_nome_tabela1_tabela2_fk1  
  
alter table tb_exemplo drop constraint ct_nome_tabela_pk  
  
alter table tb_exemplo drop constraint ct_nome_tabela_uk1
```

Por estes exemplos, você pode ver a relevância de se atribuir um nome às restrições que são especificadas, pois qualquer referência a elas são feitas pelo nome, seja para remoção ou informações de erro de violação.

14. Criação de índice

Diferentemente da chave candidata não primária, um índice pode ser criado com comando específico que não produz uma restrição (“constraint”). Sem entrar em detalhes mais complexos (porque podem ser muito complexos), um índice pode ser criado na versão simplificada com o seguinte comando conforme exemplos:

c) Exemplo 1

```
create index ix_contratos_aluguel_01  
on tb_contratos_aluguel (cod_pes_locador, dt_assinatura)
```

ii) Exemplo 2

```
create unique index ix_contratos_aluguel_02  
on tb_contratos_aluguel (cod_contr, dt_assinatura)
```

Veja neste exemplo que o índice foi criado com a cláusula de unicidade e é óbvio que o seja pela composição das colunas, mas não é uma chave candidata, não merecendo, portanto, uma declaração de restrição. O objetivo desta estrutura adicional atende somente a requisitos de performance dependentes dos tipos de acessos que a aplicação exige e constitui-se numa propriedade física, não fazendo parte da descrição dos dados no Nível Conceitual.

Este índice específico do exemplo pode ser objeto de contestação com relação à sua utilidade visto que os atributos sobre os quais está definido é uma super chave de modo que o índice de chave primária, já definida, seria suficiente para uma busca extremamente eficiente. Reflita sobre o tema e vamos deixá-lo para uma discussão futura.

15. Especificação dos dados

Para informar os dados, tanto no momento da inserção deles como para informar os dados desejados, seguem-se as regras usuais de outras linguagens. Deste modo, os números podem ser especificados em várias formas. Exemplos:

```
797
-797
797.123      // Não aplicável a números inteiros
-797.123     // Não aplicável a números inteiros
7.797123E3   // Somente para números em ponto flutuante
```

As cadeias de caracteres são expressas utilizando como delimitadores o símbolo de “apóstrofo” (“single quotes”). Na nossa língua, apóstrofo corresponde ao “single quotes”. Exemplo:

```
'abcxyz'
'ab  c  xyz  '
```

Em geral, a notação acima é a seguida pelos gerenciadores de bancos de dados, mas alguns aceitam também o símbolo “aspas” (“double quotes”) indistintamente. Em outros, a utilização pode ter significados diferentes. Deste modo, prefira a notação acima para especificar cadeias de caracteres.

Em geral, datas e horários podem ser especificados em diferentes formatos e precisam ser adequados ao idioma e localização definido para o sistema e para o cliente. Além disso, delimite com o símbolo “apóstrofo”, embora a observação anterior sobre “aspas” também se aplique para este caso. Por exemplo:

```
'2013/12/31'           /* date      */
'2013/12/31 23:15:59'  /* datetime */
'31/12/2013'           /* date      */
'12/31/2013'           /* date      */
'23:15:59'             /* time      */
```

Em algumas situações, é necessário informar junto com o valor o formato em que ele é especificado, através de algum mecanismo de “casting/conversão” ou análogo. Em geral, os primeiros dois exemplos são aceitos pelos sistemas indistintamente da localização e será utilizado no laboratório.

A palavra reservada NULL indica que um valor é indefinido, inaplicável, desconhecido ou não foi informado. Lembre-se que NULL é diferente de 'NULL'. Você deve utilizar sem os delimitadores.

16. Especificação dos dados versus formato de armazenamento

Às vezes, as pessoas confundem o formato de apresentação com o de armazenamento. O exemplo mais comum desta situação ocorre com datas. O fato de inserir os dados em diferentes apresentações ou como eles são mostrados num comando de recuperação não altera em nada a forma como os dados são armazenados. O formato interno de armazenamento é uma característica do produto, uma decisão de projeto do SGBD. Por exemplo, não importa se uma determinada data é inserida numa das formas seguintes que o resultado no banco de dados será o mesmo.

```
'2013/12/31'
'12/31/2013'
```

Neste exemplo específico, o único cuidado é que devemos nos assegurar que o sistema interpreta corretamente as duas formas de apresentação.

17. Conceito de transação

Deixaremos para uma futura aula o estudo do conceito de transação e suas implicações num ambiente concorrente. No entanto, as operações que você realiza no banco de dados precisam ser confirmadas para que sejam efetivadas e os seus resultados disponíveis para utilização. Em razão disso, vamos operar na modalidade de confirmação automática para que possamos trabalhar mesmo adiando o estudo do conceito pleno de transação.

18. Query (or Data retrieval):

Em inglês, sinônimo de "question" / "inquiry", ou seja, perguntas/consultas realizadas no banco de dados. No entanto, você verá este mesmo termo ser utilizado para qualquer comando de manipulação de dados.

DML: Linguagem de manipulação de dados. Na terminologia de alguns sistemas de bancos de dados, contempla somente comandos que alteram os bancos de dados, mas outros incluem todos que tratam com dados, inclusive consultas.

Uma consulta (query) aos dados, em termos de álgebra relacional, é assim apresentada:

Projeção

codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplício Simplório Ains Tain	01/01/2000	Saturno
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno

Tabela tb_pessoas

19. Comando select

O comando SQL para apresentação de dados pode ser bastante extensa. Vamos apresentá-la, neste momento, de forma bastante simplificada.

```
select select_list
  from table_list
 where search_conditions
```

1. A cláusula “select” (select_list) especifica quais colunas devem ser apresentadas no resultado. Corresponde à projeção da álgebra relacional e na figura acima está assinalada como “projection”. Vide mais abaixo o que pode ser especificado nesta cláusula.
2. A cláusula “from” (table_list) especifica as tabelas das quais os dados devem obtidos.
3. A cláusula “where” (search_conditions) especifica quais linhas devem ser apresentadas. É constituída por um predicado, ou seja, uma expressão que pode ser avaliada como verdadeiro ou falso e estas expressões podem ser conectados por operadores lógicos, formando outras expressões lógicas. Corresponde à seleção da álgebra relacional.

Doravante, vamos utilizar as expressões “select_list”, “table_list” e “search_conditions” para referirmos aos termos do comando “select”.

Exemplo:

```
select nome,
       cidade_natal
  from tb_pessoas
 where data_nasc = '18/06/1999'
        or cidade_natal = 'Saturno'
        or codigo = 'C0006'
```

Colunas Projetadas


codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplicio Simplório Ains Tain	01/01/2000	Saturno
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno

19.1. O símbolo (*) asterisco no select_list

O `select_list` pode ser substituído pelo símbolo asterico (*) para indicar todas as colunas. Esta é uma forma prática indicada para comandos ‘ad-hoc’, mas não deve ser utilizada em programas e aplicações, porque reduz a independência deles em relação aos dados. Uma coluna nova, ou uma que não mais existe não causa nenhum impacto na aplicação se ela não as referencia e a lista das colunas necessárias for especificada.

```
select *
  from tb_pessoas
 where nome = 'Hericlapiton dos Santos'
```

Todas as colunas projetadas




codigo	nome	data_nasc	cidade_natal
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara

19.2. Ordem e nome das colunas no resultado de consultas

Pode-se alterar os nomes das colunas na apresentação dos resultados. Não tem a ver com a estrutura da tabela. Da mesma forma, a ordem das colunas.

```
select nome as "Nome Completo",
       cidade_natal as "Cidade de Origem",
       data_nasc as "Aniversário"
  from tb_pessoas
 where nome = 'Hericlapiton dos Santos'
        or nome = 'Méki Donal di Yuessei'
        or cidade_natal = 'Marte'
```

Título e ordem das colunas



Nome Completo	Cidade de Origem	Aniversário
Letsgo Daqui	Marte	30/04/2005
Hericlapiton dos Santos	Jijoca de Jericoacoara	18/06/1999
Méki Donal di Yuessei	Uauá	04/09/2001

19.3. Expressões na lista de colunas

Cada elemento do “`select_list`”, genericamente, pode ser uma expressão, no sentido amplo. Ou seja, nome de colunas, constantes, funções e operações entre eles, desde que resulte num valor. Exemplo:

```
select "Exemplo",
       pi() * raio,
```

```

sin(angulo) + cos (angulo2),
1000.00,
power(x,3),
substring(descricao,1,20)
from tb_mathematics
where x >= 10
and raio != 0

```

19.4. Ordenação dos resultados


O resultado de um comando de consulta pode ser ordenado indicando as colunas que servirão de critério para ordenação:

```

select * from tb_pessoas
order by cidade_natal

```

Ordenado pela coluna cidade_natal



codigo	nome	data_nasc	cidade_natal
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0001	Letsgo Daqui	30/04/2005	Marte
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0004	Simplicio Simplório Ains Tain	01/01/2000	Saturno
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano

Você pode escolher entre ordem descendente ou ascendente através das cláusulas

desc

ou

asc

e, além disso, a ordenação pode ocorrer por várias colunas. Exemplo:

```

select * from tb_pessoas
order by cidade_natal asc,
       nome desc,
       data_nasc asc

```

Alternativamente, a ordenação das linhas resultantes pode ser estabelecida pela sequência numérica das colunas. Exemplo:

```
select * from tb_pessoas
order by 2, 4 desc, 3 asc
```

20. Inserção de dados numa tabela

O comando utilizado é o seguinte:

```
insert into tb_pessoas(código,nome,data_nasc, cidade_natal)
values('C0008','Epa Minon das Costas','01/01/2000',Júpiter')
```

codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplicio Simplório Ains Tain	01/01/2000	Saturno
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno
C0008	Epa Minon das Costas	01/01/2000	Júpiter

← Linha Nova

Parece absurdo, mas pelas experiências anteriores, preciso lembrar que não podem ser incluídos os valores de cada coluna, separadamente, por comando. A inserção é da tupla/linha; não de cada elemento da sequência que forma a tupla.

A lista de colunas da tabela indica a ordem dos valores que serão fornecidos na lista. Ela não é obrigatória e, se ausente, indica que todos os valores serão informados na ordem das colunas no momento da criação da tabela. Ela se torna obrigatória se:

- A lista de valores contempla somente parte das colunas. É claro que neste caso, as colunas cujos valores não foram fornecidos admitem valores nulos, ou possuem valor “default”. Dependendo do produto utilizado, outras condições podem propiciar esta situação, por exemplo, colunas de auto incremento.
- Os valores serão fornecidos em ordem diferente da lista das colunas quando a tabela foi criada.

A minha orientação é a seguinte a seguinte:

- Para comandos ‘ad-hoc’ executados manualmente, por facilidade, pode-se omitir a lista de colunas da tabela, desde que os valores sejam fornecidos para todas as colunas e na mesma ordem das colunas da tabela.

- ii. Para comandos utilizados em aplicações ou programas, especificar as colunas porque propicia maior independência dos dados em relação às aplicações. Por exemplo, se uma coluna for acrescentada na tabela, a sua aplicação não terá que ser alterada se ela não faz uso desta informação. No caso específico do comando “insert”, para que esta observação seja válida, a coluna precisa admitir valores nulos, ou seja, indefinidos, desconhecidos ou inaplicáveis.

Observações:

- i. Não assumir uma possível ordenação da nova tupla relação às demais linhas já existentes.
- ii. A palavra reservada “into” é opcional em muitos gerenciadores de bancos de dados.

20.1. Como inserir o indicador especial NULL

Para inserir NULL para uma coluna numa linha, basta não informar a coluna na lista que terão os valores incluídos ou, expressamente, assinalar NULL na posição correspondente da cláusula “values”.

a) Forma 1:

```
insert into tb_exemplo (col1,col3) values(1,100)
```

b) Forma 2

```
insert into tb_exemplo (col1,col2,col3) values(1,NULL,100)
```

20.2. Como utilizar o valor “default”

```
insert into tb_estudantes(matricula, nome, data_nasc)
values('RA999999999','Aluno 999999999', '1987/12/18')
```

Considerando a tabela tb_estudantes especificada no texto acima, o estado civil do estudante não foi informado no comando de inserção. Conforme explicado, o estado civil que será armazenado no banco de dados é 'S', ou seja, solteiro.

21. Inserção de múltiplas tuplas resultantes de um comando de seleção de dados

Na sintaxe usual do comando “insert” somente uma linha pode ser inserida em cada execução. Existe uma outra sintaxe que é contemplada na norma padrão que permite a inserção de múltiplos dados que são resultantes de um comando de “select”. Por exemplo:

```
insert into tb_pessoas(codigo,
                        nome,
                        data_nasc,
                        cidade_natal)
```

```
select cod,
       nome_pessoa,
       data_aniversario,
       cidade_nascimento
from tb_pessoas_professor
where data_nasc > '2001/01/01'
```

Este comando insere dados na tabela

tb_pessoas

os dados resultantes da seleção sobre a tabela

tb_pessoas_professor

com a restrição sobre a data de nascimento.

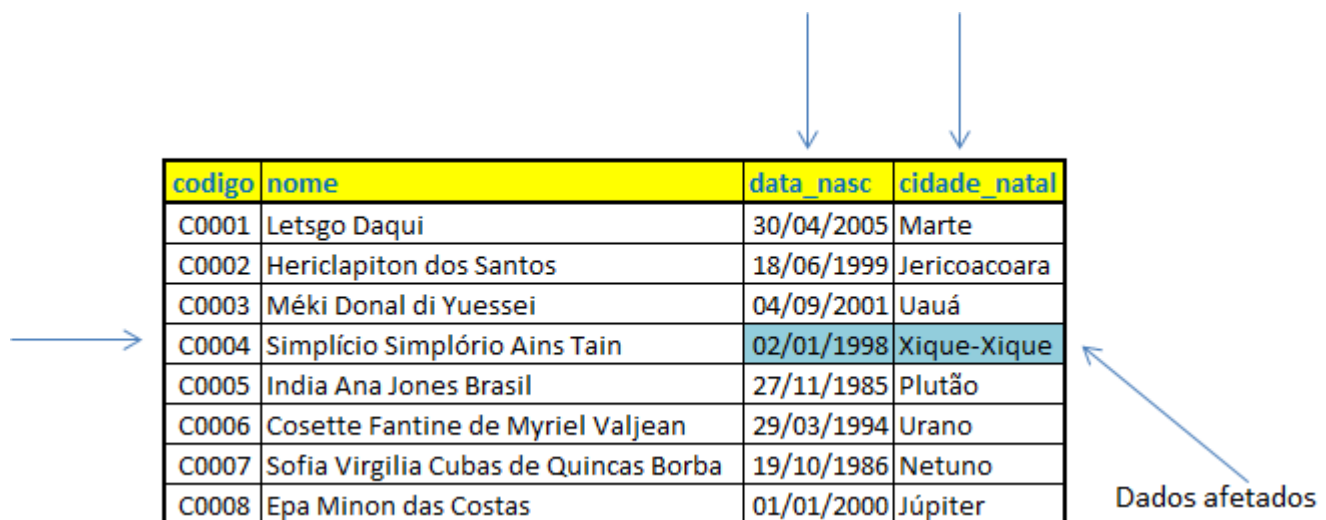
É importante ressaltar que neste caso, o comando de inserção pode incluir uma ou muitas linhas ou nenhuma. Qualquer uma destas situações caracteriza erro. Além disso, o “select” não é um comando mas parte do comando “insert”.

Todas as considerações sobre a lista de colunas do comando “insert” e valores indefinidos permanecem válidas nesta modalidade. É importante ressaltar que os dados de origem já estão no banco de dados e este mecanismo gera uma redundância, mesmo que temporária.

22. Alteração dos dados armazenados numa tabela.

O comando utilizado para alteração é o seguinte:

```
update tb_pessoas
set cidade_natal = 'Xique-Xique',
    data_nasc = '02/01/1998'
where codigo = 'C0004'
```



codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplício Simplório Ains Tain	02/01/1998	Xique-Xique
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno
C0008	Epa Minon das Costas	01/01/2000	Júpiter

Dados afetados

Atenção: A forma de especificação de datas merece cuidado. Além disso, é necessário distinguir a forma de especificação e apresentação do formato de armazenamento, pois são completamente diferentes.

23. Exclusão de dados de uma tabela

O comando utilizado para excluir linhas de uma tabela é o seguinte:

```
delete from tb_pessoas
where cidade_natal = 'Uauá'
```

Linha removida



codigo	nome	data_nasc	cidade_natal
C0001	Letsgo Daqui	30/04/2005	Marte
C0002	Hericlapiton dos Santos	18/06/1999	Jericoacoara
C0003	Méki Donal di Yuessei	04/09/2001	Uauá
C0004	Simplicio Simplório Ains Tain	02/01/1998	Xique-Xique
C0005	India Ana Jones Brasil	27/11/1985	Plutão
C0006	Cosette Fantine de Myriel Valjean	29/03/1994	Urano
C0007	Sofia Virgilia Cubas de Quincas Borba	19/10/1986	Netuno
C0008	Epa Minon das Costas	01/01/2000	Júpiter

24. Laboratório: Conexão ao sistema de banco de dados

Para realizar as atividades, você precisará conectar-se ao Sistema de Banco de Dados. Para isto, ative o script

```
exec_query.php,
```

a partir do caminho indicado para a disciplina. Será necessário informar seu “usuário” (“login”) e senha entregue pelo professor.

25. Letras maiúsculas / minúsculas

Em geral, os bancos de dados são “case sensitive”. No laboratório, o sistema de banco de dados foi preparado de forma “case insensitive”, exceto para a senha.

26. Atividades (Vide material próprio)

Prof. Satoshi Nagayama.