Criando e Gerenciando Tabelas / Implementando Constraints

Curso de Introdução a Oracle 10g: SQL

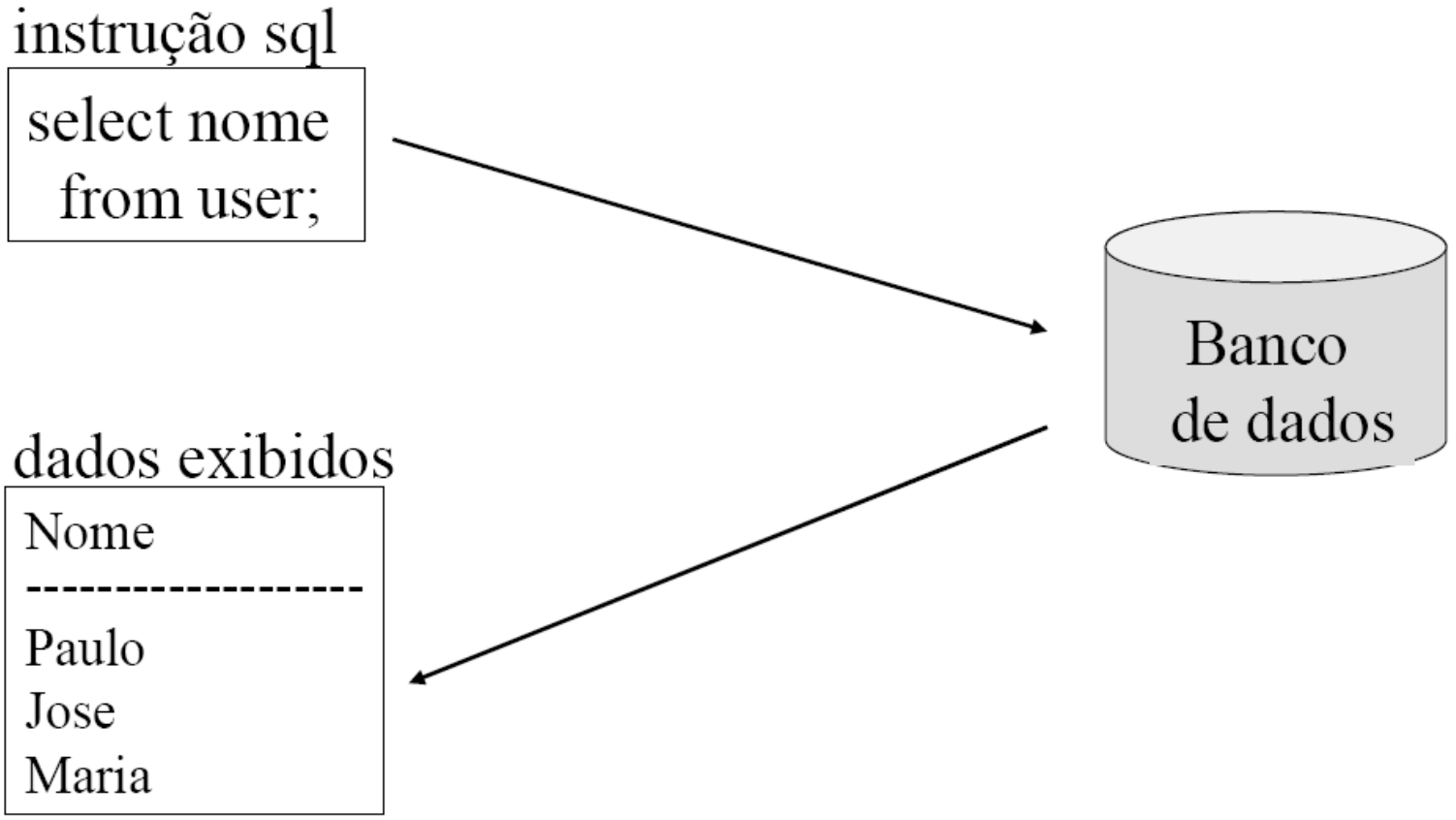
Prof.: Marlon Mendes Minussi
marlonminussi@gmail.br

STRUCTURED QUERY LANGUAGE - SQL

- A SQL surgiu no início da década de 70, por uma iniciativa da IBM.
- Nos últimos anos tornou-se a linguagem mais popular para acesso a bancos de dados, juntamente com a difusão de SGBDs relacionais.
- Existem iniciativas de padronizar a SQL, surgindo padrões SQL-86, 89, 92 e SQL-101.

SQL

- A SQL consiste em uma que permite a interface básica para comunicação com o banco de dados.



SQL

- Um SGBD realiza alguns processos que podem ser efetuados por meio da linguagem SQL.
- DEFINIÇÃO: criação descritiva do esquema que atenderá as necessidades no BD;
- CONSTRUÇÃO: inserção das instâncias inicias no banco de dados;
- MANIPULAÇÃO: realização de consultas e atualizações sobre os dados decorrentes do dia-a-dia.

ESQUEMA

- A palavra ESQUEMA representa uma coleção de objetos que pertence a um usuário e tem o mesmo nome do usuário.
- Exemplo:
- Os esquemas podem ser:
  - tabelas;
  - views;
  - sequencias;
  - entre outros.

Objetos do Banco de Dados

Objeto	Descrição
Table	Unidade básica de armazenamento composta por linhas e colunas.
View	Representação lógica de um subconjunto dos dados de uma ou mais tabelas.
Sequence	Gera valores para as chaves primárias.
Index	Aumenta a performance das consultas.
Synonym	Fornece nomes alternativos para objetos.

Convenções de nomes

- Nomeie as tabelas e colunas de acordo com o padrão de nomenclatura para qualquer objeto do banco de dados:
  - Nomes de tabelas e colunas devem começar com uma letra e podem ter de 1 até 30 caracteres de tamanho.
  - Nomes devem conter somente os caracteres A-Z, a-z, 0-9, _(underscore), $ e #.
  - Nomes não devem possuir o mesmo nome de outro objeto criado pelo mesmo usuário do Servidor Oracle.
  - Nomes não devem ser um palavra reservada do Oracle.

Diretrizes de Nomenclatura

- Utilize nomes descritivos para tabelas e outros objetos do banco de dados.

- Nota: Nomes não fazem distinção entre maiúsculas e minúsculas.

  – Por exemplo, TCLIENTES é tratado da mesma forma que tCLIENTES ou tclienteS.

Tipos de Dados

- Para iniciar a definição de um BD é necessário conhecer os tipos de dados que o BD manipula. De uma maneira geral serão manipulados os tipos:
  - numéricos (-45 | 0 | 25.57)
  - literais (José Roberto | porcelana | Bom dia!)
  - datas (dia, mês e ano | 23 / 12 / 01 | horário)
- A sintaxe que identifica estes tipos são basicamente:
  - number(*n*,*d*) – numéricos
  - varchar2(*n*) – caracteres variáveis até 4000
  - date – de 1/1/4712 A.C. até 31/12/9999 D.C.
  - *n* corresponde ao comprimento ou tamanho
  - *d* corresponde a quantidade de dígitos decimais
  - alguns outros tipos serão abordados no decorrer deste material

Tipo de Dado Caractere

- CHARACTER(x) (CHAR): representa um string de tamanho x. Se xfor omitido, então é equivalente a CHAR(1). Se um string a ser armazenado é maior que x, então o restante é preenchido com brancos.
- VARCHAR2 (<n>): armazena string de tamanho variável. É possível armazenar strings de até 4000 bytes. Subtipo: STRING.
- CHARACTER VARYING(x) (VARCHAR): é o sinônimo de VARCHAR2 que representa um string de tamanho x. Armazena exatamente o tamanho da string (tam <= x) sem preencher o resto com brancos. Neste caso x é obrigatório.
- CHARACTER LARGE OBJECT (CLOB): armazena strings longos. Usado para armazenar documentos.

Tipo de Dado Numérico

- Numéricos exatos:
- INTEGER(INT) e SMALLINT: para representar inteiros.
- NUMERIC(p,s): tem um precisão e uma escala (número de dígitos na parte fracionária). A escala não pode ser maior que a precisão. Muito usado para representar dinheiro.
- DECIMAL: também tem precisão e escala. A precisão é fornecida pela implementação

Tipo de Dado Numérico

- Números aproximados:
- REAL: ponto flutuante de precisão simples.
- DOBLE: ponto flutuante com precisão dupla.
- FLOAT(p): permite especificar a precisão que se quer. Usado para transportar(portability) aplicações

Tipo de Dado Data

- DATE: armazena ano (quatro dígitos), mês (dois dígitos) e dia (dois dígitos).
- TIME: armazena hora (dois dígitos) minutos (dois dígitos) e segundo (dois dígitos).
- TIMESTAMP: DATE + TIME

Comando CREATE TABLE

- Para criar tabelas para armazenar dados executando o comando SQL CREATE TABLE.
- Este comando é um dos comandos DDL (Data Definition Language) que é um subconjunto dos comandos SQL, utilizado para criar, modificar ou remover estruturas de banco de dados.
- Estes comando tem efeito imediato no banco de dados, e eles também registram informações no dicionário de dados.
- Para criar uma tabela, o usuário deve possuir o privilégio CREATE TABLE e uma área de armazenamento na qual criará os objetos.

Comando CREATE TABLE

- Sintaxe:
  - schema é igual ao nome do usuário dono do objeto.
  - table é o nome da tabela.
  - column é o nome da coluna.
  - datatype tipo de dado e tamanho da coluna.
    - Onde:
    - indica a definição do tipo de atributo (integer(n), char(n), real(n,m), date...).

      n- número de dígitos ou de caracteres

      m- número de casas decimais

Comando CREATE TABLE

- Sintaxe:
- CREATE TABLE schema.< nome_tabela >
- (nome_atributo1 < tipo > [NOT NULL],
- nome_atributo2 < tipo > [NOT NULL],
- .....
- nome_atributoN < tipo > [NOT NULL]) ;

Comando CREATE TABLE

EX.:

```
CREATE TABLE paciente (
        cod_pac integer PRIMARY KEY,
        nome varchar2(30),
        dt_nasc date,
        rg number(10),
        fone varchar2(12),
        est_civil varchar2(10),
        end varchar2(50));
```

Para conferir a criação da tabela:

Ex.: DESCRIBE paciente; ou DESC paciente;

Nota: Uma vez que o comando de criação é do tipo DDL, um commit automático ocorre quando este comando é executado.

Criando Tabelas

- Os tipos de dado freqüentemente utilizados são **integer** para números inteiros, **numeric** para números possivelmente fracionários, **varchar** para cadeias de caracteres, **timestamp** para data e horas, entre outros.

Consultado o Dicionário de Dados

- Visualizando as tabelas criadas pelo usuário:
  - SELECT table_name FROM user_tables;
- Visualizar os tipos de objetos distintos criados pelo usuário:
  - SELECT DISTINCT object_type FROM user_objects;
- Visualizar as tabelas, visões, sinônimos e sequences criadas pelo usuário:
  - SELECT * FROM user_catalog;
  - SELECT * FROM tab;

Criando uma tabela utilizando uma Sub_consulta

- Pode-se criar uma tabela com os dados selecionados de uma outra tabela.

- Por exemplo:

```
CREATE TABLE temp_paciente;
   AS SELECT * FROM paciente;
```

  - O comando mostrado anteriormente cria uma tabela chamada temp_paciente com a mesma estrutura da tabela paciente e com os seus dados existentes.

DROP TABLE

- Existe um limite de quantas colunas uma tabela pode conter.
- Dependendo dos tipos das colunas, pode ser entre 250 e 1600, entretanto definir uma tabela com estas quantidades de colunas é muito raro e, geralmente, torna o projeto questionável.
- Se uma tabela não é mais necessária, pode ser removida pelo comando DROP TABLE. Por exemplo:
  - DROP TABLE name_table;
  - DROP TABLE t_ temp_paciente;

ALTER TABLE

- Se for necessário modificar uma tabela existente execute o comando ALTER TABLE.

- Esse comando é utilizado para modificar o esquema ou as restrições sobre relações já existentes.

ALTER TABLE

- Para adicionar novos atributos:

```
ALTER TABLE nome_tabela
        ADD nome_campo datatype;
```

  - Por exemplo:

```
ALTER TABLE paciente
        ADD sexo char(1) ;
```

- Para eliminar atributos de uma relação deve-se digitar:

```
ALTER TABLE nome_tabela
        DROP COLUMN nome_campo;
```

- Para modificar atributos existentes em uma tabela utilizamos a cláusula MODIFY:

```
ALTER TABLE nome_tabela
        MODIFY(nome_campo datatype);
```

  - Por exemplo:

```
ALTER TABLE temp_paciente
    MODIFY(sexo varchar2 (10) NOT NULL);
```

ALTER TABLE

- Em que:
- A é o nome de uma atributo na relação r (em ORACLE deve-se escrever alter table r drop column A).
- Para renomear uma coluna de uma tabela deve-se executar:

```
ALTER TABLE  tabela
RENAME COLUMN nome_atual_coluna to novo_nome_coluna;
```

- Para renomear o nome de uma tabela deve-se digitar:

```
ALTER TABLE tabela
        RENAME to novo_nome_tabela;
```

RENAME TABLE

- A instrução que modifica o nome de uma relação é:
- RENAME <nome_atual> TO <novo_nome>;

TRUNCATE TABLE

- Outro comando DDL, utilizado para remover todas as linhas de uma tabela e liberar espaço e armazenamento por ela.

- O comando DELETE também pode remover todas as linhas de uma tabela, porém ele não libera espaço de armazenamento ocupado por ela.

CONSTRAINTS

- O Servidor Oracle utiliza constraints para prevenir entrada de dados inválidos em tabelas.
- Garantir regras à nível de tabela sempre que uma linha for inserida atualizada ou apagada desta tabela.
- A constraint deve ser satisfeita para a que a operação seja bem sucedida.
- Prevenir a exclusão de uma tabela se existir dependências em outras tabelas.

CONSTRAINTS

Constraint	Descrição
NOT NULL	não permite valor nulo.
UNIQUE Key	exige que cada valor seja único (exclusivo) em todos os registros de uma tabela.
PRIMARY KEY	define uma chave primária para uma tabela.
FOREIGN KEY	cria um relacionamento entre tabelas.
CHECK	define uma condição a ser satisfeita.

CONSTRAINTS

- Sintaxe:

```
CREATE TABLE [schema.]table
                (column datatype
                [column_constraint],
                ...
                [table_constraint]);
```

Nota:
Normalmente as constraints são criadas ao mesmo tempo que a tabela.
Podem ser adicionadas constraints a uma tabela após sua criação e também podem ser temporariamente desabilitadas.

Nível de Constraint	Descrição
Coluna	**Referencia uma única coluna e é definida dentro da especificação da própria coluna; pode-se definir qualquer tipo de constraint de integridade.**
Tabela	**Referenciada uma ou mais colunas e é definida separadamente das definições de colunas na tabela; pode-se definir qualquer tipo de constraint, exceto NOT NULL.**

CONSTRAINTS

- Exemplo:

```
CREATE TABLE paciente (
        cod_pac INTEGER NOT NULL,
        nome VARCHAR2(30) NOT NULL,
        dt_nasc DATE,
        ...
        CONSTRAINT cod_pac_pk PRIMARY KEY(cod_pac));
```

CONSTRAINTS NOT NULL

- A constraint NOT NULL assegura que valores nulos não são permitidos na coluna.
- Colunas sem a constraint NOT NULL podem conter valores nulos.
- A constraint NOTT NULL pode ser especificada somente a nível de coluna, e não a nível de tabela.

CONSTRAINTS NOT NULL

- O exemplo anterior aplica a constraint NOT NULL para as colunas COD e NOME da tabela PACIENTE.
- Uma vez que estas constraints não possuem nomes definidos, o Servidor Oracle criará nomes para elas.
- Você pode especificar o nome de uma constraint na própria especificação:
- ...nome_paciente VARCHAR2(35) CONSTRAINT nome_paciente_nn NOT NULL..

CONSTRAINTS UNIQUE KEY

- Uma constraint de integridade do tipo UNIQUE faz com que cada valor em um coluna seja único, ou seja, duas linhas de uma colunas não podem ter valores duplicados na coluna.

- Constraint UNIQUE permite a inserção de nulos a menos que você também defina uma constraint NOT NULL para a mesma coluna.

CONSTRAINTS UNIQUE KEY

- Exemplo:

```
CREATE TABLE paciente (
        cod_pac INTEGER NOT NULL,
        nome VARCHAR2(30) NOT NULL,
        dt_nasc DATE,
        rg NUMBER(10),
        ...
        CONSTRAINT rg_uk UNIQUE KEY(rg) NOT NULL);
```

- O exemplo acima aplica a constraint UNIQUE para a coluna RG da tabela PACIENTE. O nome da constraint é definido como RG_UK.

CONSTRAINTS PRIMARY KEY

- Uma constraint de PRIMARY KEY cria uma chame primária para a tabela.
- Somente uma chave primária pode ser criada o para cada tabela.
- A constraint PRIMARY KEY é uma coluna ou conjunto de colunas que identificam de forma única cada linha em uma tabela.
- Esta constraint obriga a unicidade da coluna e assegura que nenhuma coluna que faça parte da chave primária possa conter um valor nulo.

CONSTRAINTS PRIMARY KEY

- Exemplo:

```
CREATE TABLE paciente (
        cod_pac INTEGER NOT NULL,
        nome VARCHAR2(30) NOT NULL,
        dt_nasc DATE,
        rg NUMBER(10),
        ...
        CONSTRAINT rg_uk UNIQUE KEY(rg) NOT NULL,
        CONSTRAINT cod_pac_pk PRIMARY KEY (cod_pac);
```

- O exemplo acima define uma constraint PRIMARY KEY na coluna COD_PAC da tabela PACIENTE. O nome da constraint é definido como COD_PAC_PK.
- Nota: Um índice único é criado automaticamente para uma coluna de chave primária.

CONSTRAINTS FOREIGN KEY

- Uma FOREIGN KEY, ou constraint de integridade referencial, designa uma coluna como chave estrangeira e estabelece um relação entre uma chave primária ou uma chave única.

- Um valor de chave estrangeira deve corresponder a um valor existente na tabela pai ou ser nulo (se for filha de uma UNIQUE).

CONSTRAINTS FOREIGN KEY

- Exemplo:

```
CREATE TABLE atendimento (
        cod_consulta integer,
        cod_pac integer,
        cod_med integer,
        dt_consulta date,
        ...
         CONSTRAINT cod_pac_fk FOREIGN KEY (cod_pac) REFERENCES paciente (cod_pac),
        CONSTRAINT cod_consulta_pk PRIMARY KEY (cod_consulta));
```

- O exemplo acima define uma constraint FOREIGN KEY na coluna COD_PAC da tabela ATENDIMENTO. O nome da constraint é definido como COD_PAC_FK.

CONSTRAINTS FOREIGN KEY

- A chave estrangeira é definida na tabela dependente (filha), a tabela que contém a coluna referenciada é a tabela pai.
- Uma chave estrangeira é definida utilizando-se de uma combinação das seguintes palavras:
  - FOREIGN KEY: utilizada para definir a coluna da tabela filha quando a constraint for definida a nível ao nível de tabela.
  - REFERENCES: identifica a tabela e coluna da tabela referenciada (pai).
  - ON DELETE CASCADE: indica que quando uma linha da tabela pai é apagada, as linhas dependentes da tabela filha também serão apagadas.

CONSTRAINTS CHECK

- A constraint CHECK define uma condição que cada linha deve satisfazer.

- Por Exemplo:

```
...desconto number(10,2),
    CONSTRAINT desconto_ck CHECK (desconto BETWEEN 0
AND 100000),...
```

CONSTRAINTS

- Para adicionar uma constraint a uma tabela existente deve-se usar o comando ALTER TABLE.
- Sintaxe Geral:

```
ALTER TABLE nome_tabela
    ADD CONSTRAINT nome_constraint type (column);
```

- Por Exemplo:

Constraint Check

```
ALTER TABLE atendimento
    ADD CONSTRAINT valor_ck CHECK (valor > 0);
```

CONSTRAINTS

- Constraints de chave primária:

```
ALTER TABLE medico
    ADD CONSTRAINT medico_pk PRIMARY KEY  (cod_medico);
```

- Constraints de chave estrangeira:
  - O exemplo mostrado a seguir cria uma restrição de chave estrangeira para o campo cod_medico da tabela ATENDIMENTO, com a tabela de MEDICO em que o cod_medico é a chave primária. A restrição pode ser nomeada como o exemplo mostrado a seguir, cujo nome é COD_MEDICO_FK.

CONSTRAINTS

```
ALTER TABLE atendimento
    ADD CONSTRAINT cod_medico_fk
    FOREIGN KEY    (cod_medico) REFERENCES medico;
```

- Pode-se definir a restrição em que o nome é dado automaticamente, como mostrado a seguir:

```
ALTER TABLE atendimento
    ADD (FOREIGN KEY (cod_medico)
        REFERENCES  medico);
```

CONSTRAINTS

- Para remover restrições (definidas previamente com um nome):

```
ALTER TABLE nome_tabela
    DROP CONSTRAINT nome_constraint;
```

- Por Exemplo:

```
ALTER TABLE atendimento
    DROP CONSTRAINT valor_ck;
```

CONSTRAINTS

- Desabilitando um constraint:

```
ALTER TABLE atendimento
   DISABLE CONSTRAINT  valor_ck;
```

- Ativando um constraint:

```
ALTER TABLE atendimento
   ENABLE CONSTRAINT valor_ck;
```

- Visualizando todas as constraints de usuários:

```
SELECT * FROM user_constraints;


SELECT constraint_name, constraint_type, search_condition
FROM user_constraints
WHERE table_name='paciente‘;
```