Bases de Dados
SQL – Conceitos e Comandos Básicos

Prof. Renato Bueno
renato@dc.ufscar.br

20 de outubro de 2011

Apresentação baseada no material elaborado
pelo **prof. Dr. Caetano Traina Jr.** - GBDI/USP - São Carlos.

Outline

Introdução a SQL

- A Linguagem SQL – "*Structured Query Language*" foi desenvolvida pelos pesquisadores Donald D. Chamberlin and Raymond F. Boyce a partir de 1972 no Laboratório de Pesquisa da IBM em San Jose, logo depois da introdução do modelo relacional por Edgar F. Codd em 1970.
- Inicialmente chamada "SEQUEL", foi criada para ser a linguagem de consulta do SGBD Relacional "System R", então em desenvolvido no Laboratório. Logo foi renomeada para SQL ("*Structured Query Language*"), por questões de patente.
- Por sua simplicidade e grande poder de consulta, SQL é atualmente o padrão industrial em linguagens de consultas a banco de dados, dominando mais de 95% do mercado de sistemas de gerenciamento de bases de dados.

Introdução

- SQL é uma linguagem de consulta sofisticada, que vem evoluindo continuamente desde sua criação, embora mantendo um nível de padronização muito alto.
- Entre seus principais atrativos está a pequena quantidade de comandos extremamente poderosos, atendendo ao paradigma Relacional, ou seja, o programador expressa em SQL "o que" ele quer recuperar, não "como" deve ser recuperado.
- Padronizado: ANSI e ISO.

Introdução

- SQL é composta por 3 "sub-linguagens":
  1. Linguagem de Definição de Dados - DDL
  2. Linguagem de Manipulação de Dados - DML
  3. Linguagem de Controle de Dados - DCL

Linguagem de Definição de Dados - DDL

- Elementos fundamentais da linguagem:
  - `DATABASE`
  - `USER`
  - `ROLE`
  - `SCHEMA`
  - `TABLESPACE`
  - `TABLE`
  - `INDEX`
  - `DOMAIN`
  - `FUNCTION`
  - `SEQUENCE`
  - `TRIGGER`
  - `VIEW`
- Todos os elementos podem ser criados (`CREATE`), corrigidos (`ALTER`) e removidos (`DROP`).

DDL – Comando CREATE TABLE

Criar uma Tabela no Esquema da Aplicação

Sintaxe:

```
CREATE TABLE <nome da tabela> (
    <definição de Coluna>,...
    <Restrições de Integridade>,...
    );

<definição de Coluna> pode ser:
<nome atr> <tipo de dado>
    [NULL | NOT NULL]
    [ USER | DEFAULT <default- value>
           | COMPUTED BY <expresion>
           ]
```


DDL – Comando CREATE TABLE

Tipos de Dados

Sintaxe:

```
CREATE TABLE <nome da tabela> (
    <definição de Coluna>,...
    <Restrições de Integridade>,...
    );

<tipo de dado> pode ser:
 {SMALLINT | INTEGER | FLOAT | DOUBLE PRECISION}
 | {DECIMAL | NUMERIC} [( precision [, scale])]
 | DATE
 | {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR} [(int)]
 | BLOB
 }
```


DDL – Comando CREATE TABLE

Restrições de Integridade

Sintaxe:

```
CREATE TABLE <nome da tabela> (
    <definição de Coluna>,...
    <Restrições de Integridade>,...
    );
```


DDL – Comando CREATE TABLE

Restrições de Integridade como Declaração de Restrições

Restrições de Integridade são tratadas em SQL como Restrições (`CONSTRAINT`). Elas podem ser restrições de Atributo ou de Tabela.

- Restrições de atributos (ou de colunas) são declaradas para cada atributo:

  ```
  <nome atr> <tipo de dado> CONSTRAINT <nome Constraint>
  {PRIMARY KEY | UNIQUE | FOREIGN KEY ... | CHECK ...}
  ```

- Restrições de tabela são declaradas separadamente, depois que todos os atributos necessários tenham sido declarados:

  ```
  [CONSTRAINT <nome Constraint>]
    {PRIMARY KEY (ATR,...)| UNIQUE (ATR,...)|
     FOREIGN KEY ... | CHECK ...}
  ```


DDL – Comando CREATE TABLE

Criar uma Tabela no Esquema da Aplicação – Exemplo

Exemplo:

```
CREATE TABLE Turma (
   sigla char(7) NOT NULL,
   numero decimal(2) NOT NULL,
   codigo decimal(4) PRIMARY KEY,
   NNalunos decimal(3),
   constraint SiglaDaTurma FOREIGN KEY (Sigla)
      REFERENCES Discip (Sigla)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
   constraint SiglaNumero UNIQUE  (Sigla, Numero),
   constraint LimiteDeVagas CHECK  (NNalunos<50)
);
```


DDL – Comando ALTER TABLE

Modifica tabelas já definidas

Sintaxe:

```
ALTER TABLE <nome da tabela>
  ADD <definição de Coluna>
  ADD <Restrição de integridade> -- Chaves primária,
                                 -- Secund., Estrang.
  DROP <definição de Coluna>
  DROP <Restrição de integridade>

  RENAME <novo nome> -- Renomeia a tabela
  RENAME <Atributo> TO <novo atributo>

                  ...
```

DDL – Comando ALTER TABLE

Modificar tabelas já definidas – Exemplos

```
ALTER TABLE Professor
    ADD CorCabelos CHAR(25) DEFAULT 'Branco';

ALTER TABLE Aluno ADD Altura INT DEFAULT NULL;

ALTER TABLE Aluno DROP Altura;

ALTER TABLE Aluno ADD MonitoraDiscip CHAR(7)

    REFERENCES Disciplina (Sigla)
       ON UPDATE CASCADE  ON DELETE SET NULL;

ALTER TABLE Aluno DROP MonitoraDiscip;
```


DDL – Comando DROP TABLE

Remove completamente uma tabela e sua definição

Sintaxe:

```
DROP TABLE [IF EXISTS] <nome da tabela> [, ...]
[CASCADE | RESTRICT];
```

Exemplo:

```
DROP TABLE Aluno;
```


DDL – Comando CREATE DOMAIN

Cria um tipo de dado definido pelo usuário

Sintaxe:

```
CREATE DOMAIN  <Nome do Domínio> [AS] <Tipo de Dado>
  | [[NOT] NULL]
  | [DEFAULT valor-default]
  | [CHECK ( Condição)]
```


DDL – Comando CREATE DOMAIN

Cria um tipo de dado definido pelo usuário

```
<condição> = {
  VALUE <operador> <val>
  | VALUE [NOT] BETWEEN <val> AND <val>
  | VALUE [NOT] LIKE <val> [ESCAPE <val>]
  | VALUE [NOT] IN ( <val> [, <val> ...])
  | VALUE IS [NOT] NULL
  | VALUE [NOT] CONTAINING <val>
  | VALUE [NOT] STARTING [WITH] <val>
  | (<Condição>)
  | NOT <Condição>
  | <Condição> OR <Condição>
  | <Condição> AND <Condição>
}
```

```
<operador> = {= | < | > | <= | >= | !< | !> | <> | !=}
```


DDL – Comando CREATE DOMAIN

Criar tipos de dado definidos pelo usuário – Exemplos

Exemplos:

```
CREATE DOMAIN DNome_Pessoa CHAR (40) NULL;

CREATE DOMAIN DCodigo  INT  NOT NULL;

CREATE DOMAIN DIdade INT
    CHECK (VALUE BETWEEN 1 AND 120);
```


DDL – Comando DROP DOMAIN

Elimina um tipo de dado já definido pelo usuário

Sintaxe:

```
DROP DOMAIN <Nome do Domínio>
```

Exemplo:

```
DROP DOMAIN Nome_Pessoa;
```

Linguagem de Manipulação de Dados - DML

- A sub-liguagem de Manipulação de Dados tem quatro comandos:
  1. SELECT
  2. INSERT INTO
  3. UPDATE
  4. DELETE

DML – Comando SELECT

Realiza as consultas em uma base de dados

Sintaxe:

```
<Consulta>={
SELECT [ ALL | DISTINCT ] <lista de atributos>
   FROM <lista de Tabelas>
   [WHERE <condição>]
   [GROUP BY <lista de atributos>
     [HAVING <condição>]]
   }
   [ORDER BY <Lista de atributos> [ASC|DESC], ...]
   ;

<Consulta1> UNION [ALL] <Consulta2>; |
<Consulta1> INTERSECT [ALL] <Consulta2>; |
<Consulta1> EXCEPT [ALL] <Consulta2>;
```


DML – Comando SELECT

Exemplo 1

Exemplo: Listar os alunos, disciplinas e notas tiradas em turmas com mais de 10 alunos

```
SELECT A.Nome, T.Sigla Disciplinas, M.Nota
FROM Alunos A, Turma T, Matricula M
WHERE A.RA=M.RA AND
      M.CodigoTurma=T.Codigo AND
      T.NNAlunos>10
ORDER BY A.Nome, T.Sigla;
```


DML – Comando INSERT INTO - DML

Insere tuplas em uma Relação

Sintaxe:

- Formato 1: Insere uma tupla de cada vêz.

```
INSERT INTO <Tabela> [( <Atributo>, ... )]
     VALUES ( expression | DEFAULT, ... );
```

- Formato 2: Insere múltiplas tuplas a partir de uma tabela.

```
INSERT INTO <Tabela> [( <Atributo>, ... )]
     <Comando SELECT>;
```


DML – Comando INSERT INTO - DML

Exemplos

- Formato 1: Insere uma tupla de cada vêz.

```
insert into Professor values ('Antonio','5656','MS-3',33);

insert into Professor ( Nome, Grau, NNfuncional)
          values ('Antoninho', 'MS-3',  '5757');
```

- Formato 2: Insere múltiplas tuplas a partir de uma tabela.

```
INSERT INTO pessoa ( Nome, Idade )
   SELECT nome, idade from Aluno;

INSERT INTO pessoa ( Nome, Idade )
   SELECT nome, idade from Professor;

INSERT INTO Saocarlenses
   SELECT *
   FROM aluno
   WHERE cidade like 'S%Carlos';
```


DML – Comando UPDATE- DML

Altera o valor de atributos de tuplas de uma relação

Sintaxe:

```
UPDATE <tabela>
  SET <Atributo> = <expressão>, ...
  [ WHERE <Condição> ]

<expressão> = {<Atributo>|<constante>|<expr>|NULL|USER}
```

Onde <expr> é qualquer comando SELECT que resulte em apenas uma tupla e uma coluna.

DML – Comando UPDATE- DML

Exemplos

Aumentar em um a idade de todos os alunos.

```
UPDATE Alunos
       SET Idade=Idade+1

UPDATE Turma
   SET NNAlunos= (
       SELECT count (*)
       FROM matricula
       WHERE codigoTurma=101)
   WHERE Codigo=101;
```

Note-se que a cláusula `WHERE` deve selecionar apenas as tuplas de `Turma` com `código=101`.

DML – Comando DELETE FROM

Remove tuplas de uma relação

Sintaxe:

```
DELETE [FROM] <tabela>
  [WHERE <Condição>]
```

Exemplos:
Apaga todas as tuplas do aluno cujo RA vale 1234:

```
DELETE FROM Aluno
  WHERE RA=1234;
```

Remove todos os Alunos em que o atributo `Cidade` tem o valor indicado:

```
DELETE FROM Aluno
  WHERE Cidade = 'Mirim-Guaçu';
```

Apaga todas as tuplas da relação;

```
DELETE FROM Aluno;
```


Linguagem de Controle de Dados – DCL

Os comandos da sub-linguagem de Controle de Dados têm estrutura individual. Alguns exemplos são:

- `CONNECT`: Permite a conexão a uma base de dados através de um gerenciador;
- `DISCONNECT`: Desconecta de uma base de dados;
- `COMMIT`: Torna permanente todas as alterações feitas desde o início da conexão;
- `ROLLBACK`: Descarta todas as alterações feitas desde o início da conexão, ou do último comando `COMMIT` ou `ROLLBACK`.

Bases de Dados
SQL – Conceitos e Comandos Básicos

Prof. Renato Bueno
renato@dc.ufscar.br

20 de outubro de 2011

Apresentação baseada no material elaborado
pelo **prof. Dr. Caetano Traina Jr.** - GBDI/USP - São Carlos.

FIM