SQL

Structured Query Language

Profª Ms. Eveline B. Rodrigues

Linguagem SQL

- Quando os bancos de dados relacionais estavam sendo desenvolvidos, foram criadas linguagens destinadas à sua manipulação.
- O departamento de pesquisas da IBM, desenvolveu a SQL como forma de interface pra o sistema de BD relacional denominado SYSTEM R, inicio dos anos 70.
- Em 1986 o American National Standard Institute (ANSI), publicou o padrão SQL.
- A SQL estabeleceu-se como linguagem padrão de Banco de Dados Relacional.

Linguagem SQL

- A SQL ou *Structured Query Language* (Linguagem de Consulta Estruturada) é uma linguagem criada para a manipulação e o controle dos Banco de Dados relacionais.
- Alguns de seus principais recursos são:
- Alto poder de consulta
- Gerenciamento de índices
- Construção de visões
- Execução de instruções em blocos

Linguagem SQL

Características:

- A Linguagem SQL tem como grandes virtudes sua capacidade de gerenciar índices;
- Capacidade de construção de visões, que são formas de visualizarmos os dados na forma de listagens independentes das tabelas e organização lógica dos dados.
- Capacidade de cancelar uma série de atualizações ou de as gravarmos, depois de iniciarmos uma seqüência de atualizações. Os comandos Commit e Rollback são responsáveis por estas facilidades.

Linguagem SQL

- SQL apresenta uma série de comandos que permitem a definição dos dados, chamada de DDL (Data Definition Language), composta entre outros pelos comandos Create, que é destinado a criação do Banco de Dados, das Tabelas que o compõe, além das relações existentes entre as tabelas.
- Como exemplo de comandos da classe DDL temos os comandos Create, Alter e Drop.

Linguagem SQL

- Os comandos da série DML (Data Manipulation Language), destinados a consultas, inserções, exclusões e alterações em um ou mais registros de uma ou mais tabelas de maneira simultânea.
- Como exemplo de comandos da classe DML temos os comandos Select, Insert, Update e Delete.
- Uma subclasse de comandos DML, a DCL (Data Control Language), dispõe de comandos de controle como Grant e Revoke.

Linguagem SQL

- Devemos notar que a linguagem SQL consegue implementar estas soluções, somente pelo fato de estar baseada em Banco de Dados, que garantem por si mesmo a integridade das relações existentes entre as tabelas e seus índices.

Tipos de dados da linguagem SQL

Lista e descrição dos diferentes tipos de dados

- Para cada campo de cada uma tabela, é necessário determinar o tipo de dados que contem, para poder ajustar a estrutura da base de dados, e conseguir um armazenamento com a menor utilização de espaço.
- Os tipos de dados que pode ter um campo, podem-se agrupar em três grandes grupos:
  - Tipos numéricos
  - Tipos de Data
  - Tipos de Cadeia de caracteres

Tipos numéricos

- Existem tipos de dados numéricos, que se podem dividir em dois grandes grupos, os que estão em ponto flutuante (com decimais) e os que não.

Tipos de dados

Tipos de dados inteiros

- **binary:** armazena informações binárias em pares de dois bytes.
- **bit:** Dados inteiros com um valor de 1 ou 0. O tamanho de armazenamento é 1 bit.

Tipos de dados

Tipos de dados inteiros

- **tinyint:** suporta inteiros entre 0 e 255.
- **smallint:** valor inteiro compreendido entre -32768 e 32767.
- **int:** valor inteiro compreendido entre -2.147.483.648 e 2.147.483.647.
- **bigint:** valor inteiro compreendido entre - 9.223.372.036.854.775.808 e 9.223.372.036.854.775.807

Tipo de dados	Intervalo	Armazenamento
bigint	-2^63 (-9.223.372.036.854.775.808) a 2^63-1 (9.223.372.036.854.775.807)	8 bytes
int	-2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647)	4 bytes
smallint	-2^15 (-32,768) a 2^15-1 (32,767)	2 bytes
tinyint	0 a 255	1 byte

Tipos de dados

Tipos de dados reais

- **real:** Dados de número de precisão flutuantes de– 3.40E+38 a 3.40E+38. O tamanho de armazenamento é de 4 bytes.
- **float:** Dados de número de ponto flutuante de –1.79E +308 a 1.79E+308. O tamanho de armazenamento é 8 bytes.
- **numeric(p,s):** Dados de precisão e numéricos de escala fixos de $-10^{38}+1$ a $10^{38}-1$. A variável *p* especifica a precisão e pode variar entre 1 e 38. A variável *s* especifica a escala e pode variar entre 0 e *p*. Sinônimo: **decimal(p,s)**

Tipos de dados

Tipos de dados reais

- **decimal[(*p*[,*s*])] e numeric[(*p*[,*s*])]**
  Números de precisão e escala fixos. Quando a precisão máxima for usada, os valores válidos serão de - 10^38 +1 a 10^38 - 1.

  numeric é funcionalmente equivalente a **decimal**.
- p (precisão)
  O número máximo total de dígitos decimais que poderão ser armazenados, à esquerda e à direita do ponto decimal. A precisão deve ser um valor de 1 até a precisão máxima de 38. A precisão padrão é 18.
- *s* (escala)
  O número máximo de dígitos decimais que poderão ser armazenados à direita do ponto decimal. Esse número é subtraído de *p* para determinar o número máximo de dígitos à esquerda da casa decimal. O número máximo de dígitos decimais que podem ser armazenados à direita do ponto decimal.

Tipos de dados

Tipos de dados texto

- char(n): tamanho fixo, suporta até 8000 caracteres.
- nchar(n): suporta até 4000 caracteres.
- varchar(n): tamanho variável, suporta até 8000 caracteres
- nvarchar(n): suporta até 4000 caractere
- text: suporta até 2.147.483.647 caracteres (2GB)

Tipos de dados

Tipos de dados para data

- smalldatetime: data compreendida entre 01/01/1900 e 06/06/2079 com precisão de horário de 1 minuto.
- datetime: data compreendida entre 01/01/1753 e 31/12/9999 com precisão de horário de 3,33 milissegundos.

Tipos de dados

Outros tipos de dados

- **money:** Valores de dados monetários de (–2^63/10000) (–922.337.203.685.477,5808) a 2^63–1 (922.337.203.685.477,5807), com precisão de um décimo milionésimo de uma unidade monetária. O tamanho de armazenamento é 8 bytes.
- **timestamp:** Este é um número binário exclusivo gerado automaticamente. O tamanho de armazenamento é 8 bytes.
- **image:** armazenamento de imagens (até 2 GB)

Comandos de Modificações do Esquema e Criação de Banco de Dados

- **Comando Create**
- Este comando permite a criação de tabelas no banco de dados ou mesmo de sua criação.

Sintaxe:

```
CREATE DATABASE < nome_db >;
```

onde:

nome_db - indica o nome do Banco de Dados a ser criado.

Comandos de Modificações do Esquema e Criação de Banco de Dados

Sintaxe:

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

onde:

nome_table - indica o nome da tabela a ser criada.

nome_atributo - indica o nome do campo a ser criado na tabela.

tipo - 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

Comandos de Modificações do Esquema e Criação de Banco de Dados

- Agora vamos criar um BANCO DE DADOS:
- **CREATE DATABASE trabalho;**
- O comando acima criou um Banco de Dados, porém este na verdade não passa de uma abertura no diretório, pois não conta com nenhuma tabela.

Comandos de Modificações do Esquema e Criação de Banco de Dados

- Agora criaremos as tabelas que estarão contidas no Banco de Dados TRABALHO.
- A primeira Tabela será a Departamento.
- A segunda tabela será a Empregado.
- Não devemos esquecer de primeiramente abrirmos o Banco de Dados. Diferentemente do que ocorre em alguns aplicativos, em SQL o fato de criarmos um Banco de Dados, não significa que o banco recém criado já está preparado para utilização.
- A instrução a seguir, providencia a abertura do Banco de Dados criado.
- **OPEN DATABASE trabalho;**

Comandos de Modificações do Esquema e Criação de Banco de Dados

- Agora estamos prontos para criarmos as tabelas necessárias.
- A seguir código necessário a criação da tabela Departamento e seu índice:

```
CREATE TABLE departamento
(DepNume integer(4) not null,
DepNome char(20) not null,
DepLoca char(20) not null,
DepOrca integer(12,2),
primary key (DepNume));
```

- Note-se que a chave primária já está definida juntamente com o registro da tabela.

Comandos de Modificações do Esquema e Criação de Banco de Dados

- Vamos analisar o código necessário para a criação da tabela de empregados, apresentado a seguir:

```
CREATE TABLE empregado
(EmpNume integer(5) not null,
EmpNome char(30) not null,
EmpGere integer(5),
EmpServ char(20),
DepNume integer(4) not null,
EmpAdmi date not null,
EmpSala integer(10,2),
EmpComi integer(10,2),
Primary key(EmpNume));
```

Comandos de Modificações do Esquema e Criação de Banco de Dados

- Comando Drop
- Este comando elimina a definição da tabela, seus dados e referências.

Sintaxe:

DROP TABLE < nome_tabela > ;

Ex:

DROP TABLE EMP;

Inserção de Registros

Inserção de registros no Banco de Dados

- INSERÇÃO DE REGISTROS
- Para efeitos didáticos, criamos as tabelas de forma que sua população, em outras palavras os dados, sejam facilmente referenciáveis pelos estudantes.
- Assim sendo, na tabela de departamentos, contamos com 5 departamentos, cada um deles tendo seu gerente. Todos os "gerentes" tem nomes de cantoras brasileiras (Gal Costa, Marina Lima, etc), todos os "operários" tem nomes de jogadores de futebol, todas as vendedoras tem nomes de jogadoras de volei, todas as balconistas tem nome de jogadoras de basquete e o presidente da empresa exemplo, tem o mesmo nome do presidente do Brasil.
- Desta forma os testes devem resultar em grupos bastante definidos. Assim se você estiver listando Gerentes e aparecer um homônimo da Ana Paula (jogadora de volei), verifique sua consulta atentamente, pois muito provavelmente a mesma estará errada.

Inserção de registros no Banco de Dados

- Comando Insert
- Este comando insere dados em uma tabela.

Sintaxe:

INSERT INTO < nome_tabela > VALUES <valores>;

Inserção de registros no Banco de Dados

- INSERT INTO empregado VALUES (1, “Gal Costa”, “Luciano Huck”, “faxineira”, 1, 23, 200.00)
- Inserir ao menos 6 registros na tabela empregado e na tabela departamento