Linguagem SQL

1. Introdução

Histórico

- ✓ Desenvolvida pela IBM - 1970 para Banco de Dados Relacionais
- ✓ Na IBM, se chamava Sequel – Structured English Query Language
- ✓ SQL (Structured Query Language)
- ✓ Operações não procedurais de manipulação de conjuntos de dados
- ✓ Padrão ANSI
- ✓ SQL-86 ou SQL-1
- ✓ SQL-89
- ✓ SQL-92 ou SQL-2
- ✓ Suporte a linguagens de programação
- ✓ Atualmente, os produtos caminham em direção ao padrão SQL-3.

2. SQL

Apesar de ser denominada uma linguagem de consulta, a SQL possui embutidas outras funcionalidades:

- ✓ Linguagem para definição de dados (DDL)
- ✓ Linguagem para manipulação de dados (DML)
- ✓ Definição de Visões
- ✓ Controle de acesso
- ✓ Restrições de Integridade
- ✓ Controle de Transações

2.1. Definição do BD (DDL)

A linguagem de definição de dados da SQL permite a criação, modificação e exclusão de:

- ✓ Tabelas
- ✓ Índices
- ✓ Domínios
- ✓ Chaves Estrangeiras
- ✓ Chaves Primárias
- ✓ Visões

Tipos de Domínios em SQL

Para criação de objetos no banco de dados, é necessário entender os tipos de dados que podem ser armazenados no banco. O padrão SQL aceita uma variedade de tipos de domínios embutidos, incluindo os seguintes:

char(n): é uma cadeia de caracteres de tamanho fixo, com o tamanho n definido pelo usuário. A forma completa, **character**, também pode ser usada.

varchar(n): (SQL-92) é uma cadeia de caracteres de tamanho variável, com o tamanho n máximo definido pelo usuário. A forma completa, **character varying**, é equivalente.

int: é um inteiro. A forma completa, **integer**, também pode ser usada.

smallint: é um inteiro pequeno.

numeric(p, d): é um número de ponto fixo cuja precisão é definida pelo usuário. O número consiste de *p* dígitos(mais o sinal), sendo que *d* dos *p* dígitos estão à direita do ponto decimal. Assim, numeric(3,1) permite que 44,5 seja armazenado de modo exato, mas nem 444,5, nem 0,32 podem ser armazenados corretamente em um campo desse tipo.

real, double precision: são números de pontos flutuantes e ponto flutuante de precisão dupla, cuja precisão depende do equipamento.

float(n): é um número de ponto flutuante com a precisão definida pelo usuário em pelo menos *n* dígitos.

date: (SQL-92) armazena datas, contendo ano (com quatro dígitos), mês e dia do mês.

time: (SQL-92) representa horário, em horas, minutos e segundos.

Esquemas

O conceito de um esquema em SQL foi incorporado à SQL2 no sentido de agrupar tabelas e componentes que pertencem à mesma aplicação de banco de dados. Um esquema é identificado através de um nome, e inclui um identificador de autorização para indicar o usuário ou a conta que possui o esquema, bem como descritores para cada elementos do esquema. Esses elementos incluem tabelas, restrições, visões, domínios e outros componentes, como concessões de autorização, que descrevem o esquema.

Criação do esquema / banco de dados

```
create schema controle_bancário;

create database controle_bancário;
```

Domínios

```
create domain dcidade char(30)
```

Tabelas

Criação de Tabelas

```
create table cliente ( cliente_numero int not null,
                       cliente_nome char(20),
                       rua char(30),
                       cidade dcidade,
                       primary key (cliente_numero))

create table deposito (conta_numero char(10) not null,
                       cliente_numero char(20) not null,
                       saldo integer,
                       primary key (conta_numero, cliente_numero),
                       foreign key (cliente_numero) references
cliente)
```

Alteração de sua estrutura

```
alter table cliente add pais char(20)

alter table cliente modify rua char(40)

alter table cliente drop primary key
```

Exclusão

```
drop table deposito
```

2.2 Manipulando registros e dados nas tabelas

Inserção

```
insert into cliente(cliente_numero, cliente_nome, rua, cidade)
values (1, “João”, “A”, “Porto Alegre”)
```

Alteração

```
update cliente set cidade = “Belém”
where cliente_nome = “João”
```

Exclusão

```
delete depósito where cliente_nome = “João”

delete cliente
```

Empregado

Matricula	**Nome**	**DataNasc**	**Endereco**	**Sexo**	**Salario**	***MatSupervisor***	***CodigoDepto***
1	Maria	22/12/75	Rua A, 9	F	2000	null	1
2	Marcus	12/11/70	Rua B, 2	M	1500	1	1
3	Abel	13/11/60	Rua Z, 6	M	4000	1	1
4	Carlos	05/03/66	Rua J, 8	M	3000	2	3
5	Édson	16/04/70	Rua M, 9	M	1000	3	2
6	Flávio	19/09/73	Rua R, 3	M	4000	3	2
7	Gilda	20/10/74	Rua S, 4	F	1000	3	2
8	Hilton	01/11/65	Rua Z, 7	M	2000	2	3
9	Irene	05/02/50	Rua B, 3	M	3000	2	3
10	José	07/02/57	Rua M, 10	F	4000	3	3
11	Kátia	12/06/70	Rua W, 19	F	1000	2	2
12	Noel	13/07/68	Rua Y, 78	M	2000	3	3
13	Otto	15/03/75	Rua B, 10	M	2000	3	3
14	Ana	07/12/73	Rua D, 12	F	2000	3	1

Dependente

MatriculaEmp	**CodigoDependente**	**Nome**	**Idade**
1	1	Lucas	7
2	1	Clara	1
2	2	Patrick	5
3	1	Alan	10

Departamento

Codigo	**Nome**	***MatriculaGerente***
1	Direção Geral	1
2	Produção	2
3	Desenvolvimento	3

Projeto

Codigo	**Nome**	**Localizacao**	***CodigoDepto***
1	Tamar	Itaúna, ES	2
2	Jubarte	Abrolhos, BA	2
3	Pantanal	Bonito, MS	3

TrabalhaNo

MatriculaEmp	***CodigoProj***	**Horas**
4	3	40
5	1	40
6	1	40
7	2	40
8	3	40
9	3	20
10	3	20
11	2	40
12	3	20
13	3	20
9	1	20
10	1	20
12	2	20
13	2	20
14	3	40

Elaborando Consultas em SQL

1) Especificação de Consultas

- **Todas as consultas são formuladas através do comando** select**.**
- **O comando** select **tem nada a ver com a operação de** seleção **da álgebra relacional.**
- **O comando** select **implementa todas as operações da álgebra relacional.**
- **Uma tabela SQL é um multiset de tuplas, isto é, pode possuir tuplas duplicatas.**

2) O Comando SELECT

```
SELECT [ ALL | DISTINCT ] lista-de-seleção
FROM tabela [ , tabela, ... ]
[ WHERE condição ]
[ GROUP BY atributo(s) ]
[ HAVING condição ]
[ ORDER BY atributo(s) ]
```

1. Recuperação simples de tabelas:

```
SELECT Nome, DataNasc
FROM Empregado
```

Nome	DataNasc
Maria	13/07/55
João	23/12/65
Abel	31/10/73
Carlos	11/09/72
Edson	21/02/44
Flávio	30/09/56
Gilda	01/08/66
Hilton	15/05/64
Irene	16/04/63
José	25/05/59
Kátia	22/03/60
Noel	03/01/61
Otto	01/11/62

2. Listar todos os atributos para tuplas obedecendo a uma determinada condição:

```
SELECT *
FROM Empregado
WHERE Sexo = ' F '
```

Nome	Snome	Matrícula	DataNasc	Endereço	Sexo	Salário	SuperVisor	Depto
Maria	Silva	001	13/07/55		F	1000	NULL	DG
Gilda	Gomes	007	01/08/66		F	2000	003	PROD
Irene	Santos	009	16/04/63		F	3000	002	DES
Kátia	Moura	011	22/03/60		F	3000	002	PROD

3. Recuperação com a eliminação de tuplas duplicadas:

```
SELECT DISTINCT Salário
FROM Empregado
```

Salário
1000
2000
3000

4. Recuperação com valores computados de colunas:

```
SELECT DISTINCT Salário,
Salário * 1,10 AS "Novo
Salário"
FROM Empregado
```

Salário	Novo Salário
1000	1100
2000	2200
3000	3300

5. Ordenando o resultado de uma consulta:

```
SELECT Nome, Salário
FROM Empregado
ORDER BY Nome
```

Nome	Salário
Abel	1000
Carlos	2000
Edson	1000
Flávio	2000
Gilda	3000
Hilton	1000
Irene	2000
João	3000
José	3000
Kátia	1000
Maria	3000
Noel	2000
Otto	2000

6. Recuperação com condições envolvendo conjuntos e intervalos:

Usando IN:

```
SELECT DISTINCT Matrícula
FROM Empregado
WHERE SuperVisor IN ( 001, 003 )
```

Matrícula
002
003
005
006
007
010
012
013

Usando BETWEEN:

```
SELECT Nome, Salário
FROM Empregado
WHERE Salário BETWEEN 1000
AND 2000
```

Nome	Salário
Maria	1000
João	2000
Abel	1000
Carlos	2000
Flávio	1000
Gilda	2000
José	1000
Noel	2000
Otto	2000

7. Recuperação por aproximação usando LIKE:

"_" - uma ocorrência de qualquer caractere,

"%" - nenhuma ou várias ocorrências de quaisquer caracteres.

```
SELECT Nome
FROM Empregado
WHERE Snome LIKE 'J%'
```

Nome
João
José

8. Usando NULL em comparações:

```
SELECT Nome
FROM Empregado
WHERE SuperVisor IS NULL
```

Nome
Maria

9. Uso de funções de agregação:

```
SELECT AVG(Salário)
FROM Empregado
```

AVG(Salário)
2000

```
SELECT MAX(Salário)
FROM Empregado
```

MAX(Salário)
3000

```
SELECT MIN(Salário)
FROM Empregado
```

MIN(Salário)
1000

```
SELECT SUM(Salário)
FROM Empregado
```

SUM(Salário)
26000

```
SELECT COUNT(*)
FROM Empregado
```

COUNT(*)
13

10. Agrupando o resultado:

```
SELECT Depto, AVG(Salário)
FROM Empregado
GROUP BY Depto
```

Depto	AVG(Salário)
DG	1333
PROD	2250
DES	2167

```
SELECT SuperVisor, COUNT(*)
FROM Empregado
GROUP BY SuperVisor
```

Supervisor	COUNT(*)
001	2
002	4
003	6
	1

```
SELECT Sexo, SUM(Salário)
FROM Empregado
GROUP BY Sexo
```

Sexo	SUM(Salário)
F	9000
M	17000

```
SELECT Depto, AVG(Salário)
FROM Empregado
GROUP BY Depto
HAVING AVG(Salário) > 2000
```

Depto	AVG(Salário)
PROD	2250
DES	2167

```
SELECT SuperVisor, COUNT(*)
FROM Empregado
GROUP BY SuperVisor
HAVING COUNT(*) > 3
```

Supervisor	COUNT(*)
002	4
003	6

```
SELECT Sexo, SUM(Salário)
FROM Empregado
GROUP BY Sexo
HAVING SUM(Salário) <> 0
```

Sexo	SUM(Salário)
F	9000
M	17000

11. Observando os valores nulos:

- **Indica um dado não preenchido, de valor ignorado, ou um campo que não tem valor naquele registro.**
- **Não participa nas computações.**

```
média (2,4,0 ) = 2 ( 6 / 3 )
média (2,4,NULL) = 3 ( 6 / 2 )
```

```
SELECT COUNT(*)
AS "Total de empregados",
COUNT(SuperVisor)
AS "Total de empregados supervisionados"
FROM Empregado
```

Total de Empregados	Total de Empregados Supervisionados
13	12

4) Observações

- **Função_de_agregação([DISTINCT] expressão)**

```
SELECT AVG(DISTINCT Salário)
FROM Empregado → 2000
```

- **Sumariar em mais de um nível:**

```
SELECT Depto, AVG(Salário), SuperVisor, COUNT(Nome)
FROM Empregado
GROUP BY Depto, SuperVisor
```

- **GROUP BY pode ser usado sem funções de agregação.**

Funciona como se fosse um DISTINCT.

- **GROUP BY não necessariamente ordena.**

5) Having X Where

- **WHERE seleciona as tuplas antes da agregação.**
- **HAVING seleciona as tuplas depois da agregação.**

```
SELECT SuperVisor, COUNT(*)
FROM Empregado
WHERE Salário > 2000
GROUP BY SuperVisor
```

Supervisor	COUNT(*)
002	3
003	1

```
SELECT SuperVisor, COUNT(*)
FROM Empregado
GROUP BY SuperVisor
HAVING Salário > 2000
```

Supervisor	COUNT(*)
002	4
003	6

6) Funções

Cadeias de Caracteres:

- **UPPER: Converte toda a cadeia de caracteres para letras maiúsculas.**
- **LOWER: Converte toda a cadeia de caracteres para letras minúsculas.**
- **TRIM: Retira os caracteres de espaço contidos nas margens da cadeia de caracteres.**
- **LENGTH: Retorna o tamanho da cadeia de caracteres.**
- **SUBSTR: Retorna um trecho da cadeia de caracteres. Deve ser informada a posição de início do subtrecho e seu comprimento.**

Datas:

- **DAY: Retorna o dia do mês da data fornecida.**
- **MONTH: Retorna o mês da data fornecida.**
- **YEAR: Retorna o ano da data fornecida.**

Números:

- **ABS: Retorna o valor absolute do parâmetro passado.**
- **CEILING ou CEIL: Retorna o menor valor inteiro que seja maior ou igual ao parâmetro.**
- **EXP: Retorna o valor exponencial do parâmetro passado.**
- **FLOOR: Retorna o maior valor inteiro que seja menor ou igual ao parâmetro passado.**
- **LN: Retorna o logaritmo natural do parâmetro passado.**
- **MOD: Aceita dois parâmetros. Retorna o resto da divisão inteira do primeiro parâmetro pelo segundo.**
- **POWER: Aceita dois parâmetros. Retorna o valor da potência do primeiro parâmetro pelo segundo.**
- **SQRT: Retorna a raiz quadrada do parâmetro.**

7) Junções

■ **Em várias consultas será necessário correlacionar os dados de várias tabelas.**

■ **Por exemplo:**
- Qual o nome do depto de cada empregado ?
- Qual a lista de empregados de um projeto?
- Qual o nome dos gerentes de cada departamento?

■ **Produto Cartesiano:**

```
SELECT * FROM Departamento, Projeto
```

■ **Junção:**

```
SELECT * FROM Departamento, Projeto
WHERE Projeto.Depto = Departamento.DeptoSigla
```

■ **Usando apelidos para as tabelas:**

```
SELECT * FROM Departamento D, Projeto P
WHERE P.Depto = D.DeptoSigla
```

■ **Selecionando somente as colunas que interessam:**

```
SELECT D.DeptoNome, P.PNome
FROM Departamento D, Projeto P
WHERE P.Depto = D.DeptoSigla
```

■ **Junção com 3 tabelas**

Listar empregados e os projetos em que trabalham 20 horas.

```
SELECT E.Nome, E.SNome, P.Pnome
FROM Empregado E, Trabalha_Em T, Projeto P
WHERE E.Matricula = T.MatriculaEmp
AND T.Pcódigo = P.Pcódigo
AND T.Horas = 20;
```

8) Subconsultas

- **Selecionar os empregados que não possuem dependentes:**

```
SELECT E.SNome
FROM Empregado E
WHERE Not Exists ( SELECT *
                   FROM Dependente D
                   WHERE D.matricula = E.matricula)

SELECT E.SNome
FROM Empregado E
WHERE E.Matricula Not In ( SELECT D.Matricula
                           FROM Dependente D
                           WHERE D.matricula = E.matricula)
```

- **Selecionar os empregados que ganham o maior salário da empresa:**

```
SELECT E.SNome
FROM Empregado E
WHERE E.Salario >= ( SELECT Max(E.Salario) FROM Empregado E )
```

- **Selecione os empregados que tenham dependentes com seu nome:**

```
SELECT E.SNome
FROM Empregado E
WHERE Exists ( SELECT *
               FROM Dependente D
               WHERE D.matricula = E.matricula
               AND E.Nome = D.Nome)
```

➔ Esta consulta pode ser substituída por uma consulta com junções.

9) União

- **Podemos unir o resultado de várias consultas, desde que os tipos dos resultados sejam os mesmos.**
- **Departamentos gerenciados por mulheres e departamentos que atuam em projetos localizados no Espírito Santo:**

```
SELECT Nome_Dep
FROM Departamento D, Empregado E
WHERE D.Gerente = E.#Emp AND
E.Sexo = ' F '
UNION
SELECT Nome_Dep
FROM Departamento D, Projeto P
WHERE D.Nome_Dep = P.Nome_Dep AND
P.Plocalizacao LIKE "*,ES*"
```

10) SQL 2

```
SELECT E.Nome, E.SNome, P.Pnome
FROM Empregado E, Trabalha_Em T,
Projeto P
WHERE E.Matricula = T.MatriculaEmp
AND T.Pcódigo = P.Pcódigo
AND T.Horas = 20;
```

```
Select E.Nome, E.SNome, P.Pnome
From Empregado E Join Trabalha_Em T
On E.Matricula = T.MatriculaEmp
Join Projeto P
On T.Pcódigo = P.Pcódigo
Where T.Horas = 20;
```

11) Junção Externa

- **Especificado na SQL2.**

- **Exemplo: Listar os Empregados e seus dependentes, incluindo os empregados que não possuem dependentes.**

```
Select E.Nome
From Empregado E Left Outer Join Dependente D
On D.matricula = E.matricula;
```

- **Tipos de Junção Externa:**

  - ✓ **Left Outer Join:** Inclui todas as linhas da tabela à esquerda.
  - ✓ **Right Outer Join:** Inclui todas as linhas da tabela à direita.
  - ✓ **Full Outer Join:** Inclui todas as linhas de ambas as tabelas.

12) Atualizações com consultas

Consultas também podem ser utilizadas com comandos de atualização.
Exemplos:

Inserção:

```
Create Table Info_Deptos(Nome Varchar(200),
                         No_De_Emps, Integer,
                         Total_Sal Integer)

Insert into Info_Deptos
      Select D.DeptNome, Count(*), Sum(Salario)
      From Departamento D, Empregado E
      Where D.DeptoSigla = E.Depto
      Group By D.DeptoNome
```

Exclusão:

```
Delete From Empregado
Where Depto In (Select DeptoSigla
                From Departamento
                Where DeptNome = 'Contabilidade')
```

Atualização:

```
Update Empregado E
Set E.Salario = E.Salario * 1.1
Where Depto In (Select DeptoSigla
                From Departamento
                Where DeptNome = 'Vendas')
```

13) Visão

É uma tabela 'virtual' derivada de outras tabelas ou visões. Uma visão possui um nome, uma lista de nomes de atributos (opcional) e uma consulta para especificar o conteúdo da visão.

Exemplo:

```
Create View TrabalhaEm1 As
Select E.SNome, P.PNome, T.Horas
From Empregado E, Trabalha_Em T, Projeto P
Where E.Matricula = T.MatriculaEmp
And T.PCodigo = P.PCodigo

Create View Info_Depto (NomeDepto, NoDeEmps, TotalSal) As
Select D.DeptNome, Count(*), Sum(Salario)
From Departamento D, Empregado E
Where D.DeptoSigla = E.Depto
Group By D.DeptoNome

Drop View NomeView;
```

Uma visão está sempre atualizada. Se modificarmos as tuplas nas tabelas de base nas quais a visão é definida, a visão reflete automaticamente estas atualizações.

É possível atualizar uma visão. Contudo, nem sempre isso é permitido pelas regras de formação da visão ou das tabelas que formam a visão.

14) Atualizações e Segurança

Tipos de Segurança:

- ✓ **Questões legais e éticas com relação ao direito de acessar certas informações.**
- ✓ **Questões Políticas de nível governamental, institucional ou coorporativo.**
- ✓ **Questões relacionadas ao sistema, como níveis do sistema nos quais devem ser impostas funções de segurança.**

Um SGBD geralmente inclui um subsistema de autorização e segurança que é responsável por garantir a segurança de partes de um BD em relação a acessos desautorizados.

Mecanismos de Segurança de BD

→ Mecanismos flexíveis de segurança: Utilizados para conceder privilégios a usuários.

→ Mecanismos obrigatórios de segurança: Utilizados para impor a segurança multinível através de classificação de dados e usuários em várias classes ou níveis de segurança. A seguir, é implementada a política apropriada de segurança na organização.

Outras Questões relacionadas a segurança:

- ✓ **Acesso ao sistema**
- ✓ **Criptografia de dados**

Proteção de Acesso, contas e auditoria

No SGBD, contas e senhas são armazenadas em uma tabela ou arquivo criptografado.

O sistema de BD deve controlar todas as operações no BD que sejam aplicadas por um determinado usuário em cada sessão. Cada operação é armazenada no arquivo de LOG do sistema. Assim, o DBA pode descobrir qual usuário realizou uma adulteração. Assim, quando houver alguma suspeita de qualquer problema, é feita uma auditoria no BD.

Tipos de Privilégios Flexíveis:

- ✓ **Nível de conta:**
  - **Create Schema / Database**
  - **Create Table**
  - **Create View**
  - **Alter**
  - **Drop**
  - **Modify**
  - **Select**
- ✓ **Nível de Tabela:**
  - **Especifica para cada usuário as tabelas nas quais cada tipo de comando pode ser aplicado.**

Exemplos de comandos:

Create User Usuario1 Identified by Senha;

Create Database Exemplo Authorization Usuario1;

Create Role Papel1;

Grant Insert, Delete On Empregado to Papel1;

Grant Insert, Delete On Empregado to Usuario2;

Grant Select on Empregado, Departamento to Usuario3 With Grant Option;

Revoke Select on Empregado From Usuario4;