

SQL - Structured Query Language : Definição de Esquema, Restrições, Consultas e Visões

1

Introdução

- SQL (Structured Query Language);
- Usa uma combinação da álgebra relacional e construções de cálculo relacional;
- Foi desenvolvida pela IBM no início dos anos 70 e mais tarde se tornou um padrão ANSI;
- Se estabeleceu como a linguagem padrão para banco de dados relacional;
- Embora seja chamada de “linguagem de consulta” ela contém outras capacidades além de consultas a banco de dados;
- Inclui recursos para definição de estruturas, modificação e restrições de dados.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

2

2

Definição de Dados em SQL

- Usado para CRIAR, EXCLUIR, e ALTERAR a descrição das tabelas (relações) de um banco de dados

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

3

3

CREATE DATABASE

- Especifica um novo esquema de banco de dados, dando-lhe um nome
- Em PostgreSQL:

```
CREATE DATABASE <<nome da database>>  
    WITH OWNER = postgres  
    ENCODING = 'UTF8';
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

4

4

CREATE TABLE

- Especifica uma nova relação, dando-lhe um nome e especificando cada um de seus atributos e seus tipos de dados (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- Uma restrição NOT NULL pode ser especificada para um atributo que não aceita valor nulo

```
CREATE TABLE DEPARTAMENTO (  
    DNOME          VARCHAR(10)      NOT NULL,  
    DNUMERO         INTEGER          NOT NULL,  
    GERSSN          CHAR(9) ,  
    GERDATAINICIO  CHAR(9)  ) ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

5

5

CREATE TABLE

- Em SQL2, o comando CREATE TABLE pode ser usado para especificar a chave primária, a chave secundária e restrições de integridade referencial (chaves estrangeiras).
- Atributos chave podem ser especificados através dos comandos PRIMARY KEY e UNIQUE

```
CREATE TABLE DEPT (  
    DNOME          VARCHAR(10)      NOT NULL,  
    DNUMERO         INTEGER          NOT NULL,  
    GERSSN          CHAR(9) ,  
    GERDATAINICIO  CHAR(9) ,  
    PRIMARY KEY (DNUMERO) ,  
    UNIQUE (DNOME) ,  
    FOREIGN KEY (GERSSN) REFERENCES  
    EMP(SSN)  ) ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

6

6

DROP TABLE

- Usada para excluir elementos de um esquema, como relações (tabelas), domínios ou restrições
- A relação não poderá mais ser utilizada em consultas, atualizações ou qualquer outro comando visto que ela já não existe mais
- Exemplo:

DROP TABLE DEPENDENTE ;

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

7

7

ALTER TABLE

- Usado para alterar elementos do esquema.
- As ações de alteração podem ser: adicionar / eliminar um atributo, alterar um atributo ou adicionar / eliminar restrições
 - O novo atributo terá NULLs em todas as tuplas da relação logo após a adição; consequentemente, a restrição NOT NULL não é permitida
- Exemplo:
ALTER TABLE EMPREGADO ADD FUNCAO VARCHAR(12) ;
- O usuário deve informar um valor para o novo atributo FUNCAO para cada tupla EMPREGADO.
 - Isso pode ser feito usando o comando UPDATE

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

8

8

OPÇÕES BÁSICAS DE INTEGRIDADE

■ Restrições de Atributos

- Em SQL é permitido NULL como valor de atributos
- Uma restrição NOT NULL implica que o valor do atributo não pode ser vazio (NULL)

```
CREATE TABLE DEPT (  
    DNOME          VARCHAR(10)    NOT NULL,  
    DNUMERO         INTEGER        NOT NULL,  
    ...  
);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

9

9

OPÇÕES BÁSICAS DE INTEGRIDADE

■ Restrições de Padrão (*default*) de Atributos

- Em SQL é também possível definir um valor default para um atributo
- Adicionando a cláusula DEFAULT <valor> na definição do atributo

```
CREATE TABLE DEPT (  
    ...  
    GERSSN          CHAR(9)  DEFAULT 555,  
    GERDATAINICIO   CHAR(9)  DEFAULT '8/01/2009',  
    ...  
);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

10

10

OPÇÕES BÁSICAS DE INTEGRIDADE

■ Restrições de Verificação de Atributos

- Em SQL restrições podem limitar os valores de atributos ou de domínios pelo uso da cláusula `CHECK`
- Adicionando a cláusula `CHECK <restrição>` na definição do atributo

```
CREATE TABLE DEPT (  
    DNAME          VARCHAR(10)      NOT NULL,  
    DNUMERO        INTEGER          NOT NULL  
        CHECK (DNUMERO > 0) ,  
    ...  
);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

11

11

OPÇÕES BÁSICAS DE INTEGRIDADE

■ Restrições de Unicidade de Atributos

- A restrição de unicidade garante que os dados contidos na coluna é único em relação a todas as outras linhas da tabela

```
CREATE TABLE DEPT (  
    DNAME          VARCHAR(10)      NOT NULL,  
    DNUMERO        INTEGER          NOT NULL,  
    SSN            INTEGER          UNIQUE  
    ...  
);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

12

12

OPÇÕES DE INTEGRIDADE REFERENCIAL

- Pode-se especificar restrições de integridade referencial tais como: RESTRICT, CASCADE, SET NULL or SET DEFAULT

```
CREATE TABLE DEPT (  
    DNOME          VARCHAR(10)    NOT NULL,  
    DNUMERO        INTEGER        NOT NULL,  
    GERSSN         CHAR(9) ,  
    GERDATAINICIO CHAR(9) ,  
    PRIMARY KEY (DNUMERO) ,  
    UNIQUE (DNOME) ,  
    FOREIGN KEY (GERSSN) REFERENCES EMP  
    ON DELETE SET DEFAULT ON UPDATE  
    CASCADE) ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

13

13

OPÇÕES DE INTEGRIDADE REFERENCIAL (cont)

```
CREATE TABLE EMP (  
    ENOME          VARCHAR(30)    NOT NULL,  
    ESSN           CHAR(9) ,  
    BDATA          DATE ,  
    DNO            INTEGER DEFAULT 1 ,  
    SUPERSSN       CHAR(9) ,  
    PRIMARY KEY (ESSN) ,  
    FOREIGN KEY (DNO) REFERENCES DEPT  
    ON DELETE SET DEFAULT ON UPDATE  
    CASCADE ,  
    FOREIGN KEY (SUPERSSN) REFERENCES EMP  
    ON DELETE SET NULL ON UPDATE CASCADE) ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

14

14

Tipos de Dados do PostgreSQL - Numéricos

Nome	Tamanho de armazenamento	Descrição	Faixa de valores
smallint	2 bytes	inteiro com faixa pequena	-32768 a +32767
integer	4 bytes	escolha usual para inteiro	-2147483648 a +2147483647
bigint	8 bytes	inteiro com faixa larga	-9223372036854775808 a 9223372036854775807
decimal	variável	precisão especificada pelo usuário, exato	sem limite
numeric	variável	precisão especificada pelo usuário, exato	sem limite
real	4 bytes	precisão variável, inexato	precisão de 6 dígitos decimais
double precision	8 bytes	precisão variável, inexato	precisão de 15 dígitos decimais
serial	4 bytes	inteiro com auto-incremento	1 a 2147483647
bigserial	8 bytes	inteiro grande com auto-incremento	1 a 9223372036854775807

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

15

15

Tipos de Dados do PostgreSQL - Monetário

- O tipo monetário está em desuso. Em seu lugar deve ser utilizado o tipo `numeric` ou `decimal`

Nome	Tamanho de Armazenamento	Descrição	Faixa
money	4 bytes	quantia monetária	-21474836.48 a +21474836.47

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

16

16

Tipos de Dados do PostgreSQL – Cadeias de Caracteres

Nome	Descrição
<code>character varying(n)</code> , <code>varchar(n)</code>	comprimento variável com limite
<code>character(n)</code> , <code>char(n)</code>	comprimento fixo, completado com brancos
<code>text</code>	comprimento variável não limitado

- Onde `n` é um número inteiro positivo e indica que pode ser armazenada uma cadeia de caracteres com comprimento de até `n` caracteres.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

17

17

Tipos de Dados do PostgreSQL – Dado Binário

- O tipo de dado `bytea` permite o armazenamento de cadeias binárias, ou seja, sequências de octetos

Nome	Tamanho de Armazenamento	Descrição
<code>bytea</code>	4 bytes mais a cadeia binária	Cadeia binária de comprimento variável

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

18

18

Tipos de Dados do PostgreSQL – Data e Hora

Nome	Tamanho de Armazenamento	Descrição	Menor valor	Maior valor	Resolução
timestamp [(p)] [without time zone]	8 bytes	tanto data quanto hora	4713 AC	5874897 DC	1 microssegundo / 14 dígitos
timestamp [(p)] with time zone	8 bytes	tanto data quanto hora, com zona horária	4713 AC	5874897 DC	1 microssegundo / 14 dígitos
interval [(p)]	12 bytes	intervalo de tempo	-178000000 anos	178000000 anos	1 microssegundo / 14 dígitos
date	4 bytes	somente data	4713 AC	32767 DC	1 dia
time [(p)] [without time zone]	8 bytes	somente a hora do dia	00:00:00.00	23:59:59.99	1 microssegundo / 14 dígitos
time [(p)] with time zone	12 bytes	somente a hora do dia, com zona horária	00:00:00.00+12	23:59:59.99-12	1 microssegundo / 14 dígitos

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

19

19

Tipos de Dados do PostgreSQL – Formato de Data e Hora

- Utilizando o comando `set datastyle` o formato da saída para os tipos data e hora pode ser definido

Especificação de estilo	Descrição	Exemplo
ISO	ISO 8601/padrão SQL	2005-04-21 18:39:28.283566-03
SQL	estilo tradicional	04/21/2005 18:39:28.283566 BRT
POSTGRES	estilo original	Thu Apr 21 18:39:28.283566 2005 BRT
German	estilo regional	21.04.2005 18:39:28.283566 BRT

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

20

20

Tipos de Dados do PostgreSQL – Endereço de Rede

- O PostgreSQL disponibiliza tipos de dados para armazenar endereços IPv4, IPv6 e MAC
- É preferível utilizar estes tipos em vez dos tipos texto puro, porque possuem verificação de erro

Nome	Tamanho de Armazenamento	Descrição
<code>cidr</code>	12 ou 24 bytes	redes IPv4 e IPv6
<code>inet</code>	12 ou 24 bytes	hospedeiros e redes IPv4 e IPv6
<code>macaddr</code>	6 bytes	endereço MAC

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

21

21

Manipulação de Dados em SQL

- Existem três comandos SQL para modificar o banco de dados: **INSERT**, **DELETE**, e **UPDATE**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

22

22

INSERT

- Em sua forma simplificada, é utilizado para adicionar uma ou mais tuplas a uma relação
- Os valores dos atributos devem ser listados na mesma ordem como foram especificados os atributos no comando **CREATE TABLE**

23

INSERT (cont.)

- Exemplo:
U1: INSERT INTO EMPREGADO VALUES ('Richard', 'K', 'Marini', '653298653', '30-DEC-52', '98 Oak Forest, Katy, TX', 'M', 37000, '987654321', 4)
- Uma forma alternativa de INSERT especifica explicitamente os nomes dos atributos que correspondem aos valores na nova tupla
 - Atributos com valores NULL podem ser deixados de fora
- Exemplo: Insira uma tupla para um novo EMPREGADO para o qual só se conhece o PNAME, UNOME e SSN.

U1A: INSERT INTO EMPREGADO (PNAME, UNOME, SSN)
VALUES ('Richard', 'Marini', '653298653')

24

INSERT (cont.)

- Nota Importante: Somente as restrições especificadas em comandos DDL são automaticamente impostas pelo SGBD quando atualizações são aplicadas ao banco de dados
 - Outra variação de INSERT permite a inserção de múltiplas tuplas
 - Exemplo:

```
INSERT INTO EMPREGADO (PNAME, UNAME, SSN)
VALUES ('Richard', 'Marini', '653298653'),
('Fred', 'Packer', '893298646'),
('Mary', 'Smith', '948752045')
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

25

25

DELETE

- Exclui tuplas de uma relação
 - Contém uma cláusula WHERE para selecionar as tuplas a serem excluídas
 - A integridade referencial deve ser mantida
 - As tuplas são deletadas de apenas uma tabela por vez (a menos que um CASCADE seja especificado em uma restrição de integridade referencial)
 - A ausência da cláusula WHERE especifica que todas as tuplas da relação serão excluídas; a tabela então se tornará vazia
 - O número de tuplas excluídas depende do número de tuplas da relação que satisfaçam a cláusula WHERE

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

26

26

DELETE (contd.)

- Examples:

U4A:	DELETE FROM WHERE	EMPREGADO UNOME='Brown'
U4B:	DELETE FROM WHERE	EMPREGADO SSN='123456789'
U4C:	DELETE FROM WHERE	EMPREGADO DNO IN (SELECT DNUMERO FROM DEPARTAMENTO WHERE DNOME='Pesquisa')
U4D:	DELETE FROM	EMPREGADO

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

27

27

UPDATE

- Usada para modificar valores de atributos de uma ou mais tuplas selecionadas
- Uma cláusula WHERE seleciona as tuplas a serem modificadas
- Uma cláusula SET adicional especifica os atributos a serem modificados e seus novos valores
- Cada comando modifica tuplas da mesma relação
- A integridade referencial deve ser mantida

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

28

28

UPDATE (cont.)

- Exemplo: Mude a localização e o número do departamento de controle do projeto número 10 para 'Bellaire' e 5, respectivamente.

```
U5:      UPDATE PROJETO
          SET     PLOCALIZACAO = 'Bellaire',
                DNUM = 5
          WHERE   PNUMERO=10
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

29

29

UPDATE (cont.)

- Exemplo: Dê a todos os empregados do departamento de 'Pesquisa' um aumento de 10%.

```
U6: UPDATE EMPREGADO
      SET     SALARIO = SALARIO *1.1
      WHERE   DNO IN (SELECT DNUMERO
                       FROM   DEPARTAMENTO
                       WHERE  DNOME='Pesquisa')
```

- Aqui, a modificação do valor do SALARIO depende do valor original do SALARIO em cada tupla
 - A referência ao atributo SALARIO do lado direito do = se refere ao valor antigo do SALARIO **antes** da modificação
 - A referência ao atributo SALARIO do lado esquerda do = se refere ao valor do novo SALARIO **depois** da modificação

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

30

30

Consultas de Dados em SQL

- Serve para buscar dados armazenados no BD;
- É altamente flexível, o que facilita a filtragem dos dados recuperados em uma consulta;
- É baseada na álgebra relacional e no cálculo relacional de tuplas.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

31

31

Consultas em SQL

- Forma básica de um comando SQL SELECT é chamado um mapeamento ou um bloco SELECT-FROM-WHERE

SELECT <lista de atributos>
FROM <lista de tabelas>
WHERE <condição>

- <lista de atributos> é uma lista de nomes de atributos cujos valores serão recuperados pela consulta
- <lista de tabelas> é uma lista dos nomes das relações necessárias para o processamento da consulta
- <condição> é uma expressão condicional (booleana) que identifica as tuplas que serão recuperadas pela consulta

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

32

32

Esquema do Banco de Dados Relacional

EMPREGADO

PNAME	MINICIAL	UNOME	SSN	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	-----	----------	----------	------	---------	----------	-----

DEPARTAMENTO

DNOME	DNUMERO	GERSSN	GERDATAINICIO
-------	---------	--------	---------------

DEPTO_LOCALIZACOES

DNUMERO	DLOCALIZACAO
---------	--------------

PROJETO

PJNOME	PNUMERO	PLOCALIZACAO	DNUM
--------	---------	--------------	------

TRABALHA_EM

ESSN	PNO	HORAS
------	-----	-------

DEPENDENTE

ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
------	-----------------	------	----------	------------

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

33

33

Banco de Dados Populado

EMPREGADO	PNAME	MINICIAL	UNOME	SSN	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fordham, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alice	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellare, TX	F	43000	888665555	4	
Ramesh	K	Nanyan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5831 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jasbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Burg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1	

DEPT LOCALIZACOES

DNUMERO	DLOCALIZACAO
1	Houston
4	Stafford
5	Bellare
5	Sugarland
	Houston

DEPARTAMENTO	DNOME	DNUMERO	GERSSN	GERDATAINICIO
Pesquisa		5	333445555	1988-05-22
Administracao		4	987654321	1995-01-01
Sede administrativa		1	888665555	1981-06-19

TRABALHA_EM	ESSN	PNO	HORAS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJETO	PJNOME	PNUMERO	PLOCALIZACAO	DNUM
ProdutoX		1	Bellare	5
ProdutoY		2	Sugarland	5
ProdutoZ		3	Houston	5
Automatizacão		10	Stafford	4
Reorganizacao		20	Houston	1
Novos Beneficios		30	Stafford	4

DEPENDENTE	ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
	333445555	Alice	F	1986-04-05	FILHA
	333445555	Theodore	M	1983-10-25	FILHO
	333445555	Joy	F	1958-05-03	CÔNJUGE
	987654321	Atner	M	1942-02-28	CÔNJUGE
	123456789	Michael	M	1988-01-04	FILHO
	123456789	Alice	F	1988-12-30	FILHA
	123456789	Elizabeth	F	1967-05-05	CÔNJUGE

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

34

34

Consulta SQL Simples

- Consultas SQL básicas usam as seguintes operações da álgebra relacional:
 - SELEÇÃO
 - PROJEÇÃO
 - JUNÇÃO
- Todos os exemplos subsequentes usam o banco de dados EMPRESA

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

35

35

Consulta SQL Simples (cont.)

- Exemplo de uma consulta simples em uma relação
- Consulta 0: Recupere o aniversário e o endereço dos empregados cujo o nome seja 'John B. Smith'.

```
Q0:SELECT      DATANASC, ENDERECO
    FROM      EMPREGADO
    WHERE      PNAME='John' AND MINICIAL='B'
    AND        UNOME='Smith'
```
- Similar a um par de operações SELECT-PROJECT da álgebra relacional:
 - A cláusula SELECT especifica os atributos de projeção e a cláusula WHERE especifica a condição de seleção
- Entretanto, o resultado de uma consulta pode conter tuplas duplicadas

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

36

36

Consulta SQL Simples (cont.)

- Consulta 1: Recupere o nome e o endereço de todos os funcionários que trabalham no departamento de 'Pesquisa'.

```
Q1:SELECT      PNAME, UNOME, ENDERECO  
      FROM      EMPREGADO, DEPARTAMENTO  
      WHERE     DNAME='Pesquisa' AND DNUMERO=DNO
```

- Similar a uma sequência de operações SELECT-PROJECT-JOIN da álgebra relacional
- (DNAME='Pesquisa') é uma condição de seleção (corresponde a uma operação de SELECAO na álgebra relacional)
- (DNUMERO=DNO) é uma condição de junção (corresponde a uma operação de JUNCAO na álgebra relacional)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

37

37

Consulta SQL Simples (cont.)

- Consulta 2: Para cada projeto localizado em 'Stafford', liste o número do projeto, o número do departamento responsável e o último nome do gerente do departamento, seu endereço e data de nascimento.

```
Q2: SELECT      PNUMERO, DNUM, UNOME, DATANASC, ENDERECO  
      FROM      PROJETO, DEPARTAMENTO, EMPREGADO  
      WHERE     DNUM=DNUMERO AND GERSSN=SSN  
               AND PLOCALIZACAO='Stafford'
```

- Em Q2, existem duas condições de junção
- A condição de junção DNUM=DNUMERO relaciona um projeto a seu departamento de controle
- A condição de junção GERSSN=SSN relaciona o departamento de controle com o empregado que administra esse departamento

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

38

38

Aliases, * e DISTINCT, cláusula WHERE ausente

- Em SQL, pode-se usar o mesmo nome para dois ou mais atributos sendo que os atributos estejam em relações diferentes
- Uma consulta que se refere a dois um mais atributos com o mesmo nome deve qualificar o nome do atributo com o nome da relação através da prefixação do nome da relação ao nome do atributo
- Exemplo:
 - EMPREGADO. NOME, DEPARTAMENTO. NOME

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

39

39

ALIASES

- Algumas consultas precisam se referenciar a mesma relação mais de uma vez
 - Nesse caso, *aliases* (pseudônimos) são dados aos nomes das relações
- Consulta 8: Para cada empregado, recupere seu nome e o nome de seu superior imediato.

```
Q8: SELECT      E.PNOME, E.UNOME, S.PNOME, S.UNOME
      FROM      EMPREGADO E, EMPREGADO S
      WHERE     E.SUPERSSN=S.SSN
```

- Em Q8, os nomes alternativos E e S são chamados *aliases* ou *variáveis de tupla* para a relação EMPREGADO
- Pode-se pensar em E e S como duas cópias diferentes de EMPREGADO; E representa os empregados no papel de *supervisionados* e S representa os empregados no papel de *supervisores*

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

40

40

ALIASES (cont.)

- Aliases pode ser também utilizados em qualquer consulta SQL por conveniência
- Pode-se também utilizar a palavra chave AS para especificar aliases

```
Q8:SELECT E.PNOME AS PRIMEIRONOME,  
        S.PNOME, S.UNOME  
FROM    EMPREGADO AS E,  
        EMPREGADO AS S  
WHERE   E.SUPERSSN=S.SSN
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

41

41

Cláusula WHERE Ausente

- A ausência da cláusula *WHERE* indica que não há nenhuma condição; conseqüentemente, todas as tuplas das relações na cláusula FROM serão selecionadas
 - Isso é equivalente a condição *WHERE* ser verdadeira
- Consulta 9: Recupere o SSN para todos os empregados.
 - Q9: SELECT SSN
 FROM EMPREGADOS
- Se mais de uma relação for especificada na cláusula FROM e não existir condição de junção, então será obtido o PRODUTO CARTESIANO dessas relações

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

42

42

Cláusula WHERE Ausente (cont.)

■ Exemplo:

Q10: SELECT SSN, DNOME
 FROM EMPREGADO,
 DEPARTAMENTO

- É extremamente importante especificar todas as condições de seleção e de junção na cláusula WHERE; senão, podem ocorrer resultados incorretos ou muito grandes

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

43

43

Uso do * (Asterisco)

- Para recuperar os valores de todos os atributos das tuplas selecionadas, o * é usado, que significa selecionar todos os atributos

Exemplos:

Q1C: SELECT *
 FROM EMPREGADO
 WHERE DNO=5

Q1D: SELECT *
 FROM EMPREGADO, DEPARTAMENTO
 WHERE DNOME='Pesquisa' AND
 DNO=DNUMERO

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

44

44

Uso do DISTINCT

- A SQL não trata uma relação como um conjunto; assim, tuplas duplicadas podem aparecer
- Para eliminar tuplas duplicadas no resultado de uma consulta, a palavra chave **DISTINCT** é usada
- Por exemplo, o resultado de Q11 pode ter valores de SALARIO duplicados, enquanto que Q11A não tem nenhum valor duplicado

Q11: SELECT SALARIO
 FROM EMPREGADO

Q11A: SELECT **DISTINCT** SALARIO
 FROM EMPREGADO

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

45

45

Operações de Conjuntos

- A SQL tem diretamente incorporada algumas operações de conjunto
- Existe uma operação de união (UNION), e em algumas versões da SQL há operações de diferença de conjuntos (MINUS) e interseção (INTERSECT)
- As relações resultantes dessas operações de conjunto são conjuntos de tuplas; tuplas duplicadas são eliminadas no resultado
- Operações de conjuntos são aplicáveis apenas a relações união compatíveis; as duas relações devem ter os mesmos atributos e os atributos devem aparecer na mesma ordem

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

46

46

Operações de Conjuntos (cont.)

- Consulta 4: Faça uma lista de todos os números de projetos para projetos que envolvam um empregado cujo último nome seja 'Smith', como trabalhador ou como gerente do departamento que controla o projeto.

```
Q4:      (SELECT      PNUMERO
          FROM        PROJETO, DEPARTAMENTO,
                   EMPREGADO
          WHERE       DNUM=DNUMERO AND
                   GERSSN=SSN AND UNOME='Smith')

          UNION
          (SELECT      PNUMERO
          FROM        PROJETO, TRABALHA_EM,
                   EMPREGADO
          WHERE       PNUMERO=PNO AND
                   ESSN=SSN AND NOME='Smith')
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

47

47

Consultas Aninhadas

- Algumas consultas necessitam de valores presentes no BD para, então, usá-los na condição de comparação
 - Essas consultas pode ser formuladas por meio de consultas aninhadas
- Uma consulta SELECT completa, chamada consulta aninhada, pode ser especificada dentro da cláusula WHERE de outra consulta, chamada consulta externa
- Consulta 1: Recupere nome e endereço dos empregados que trabalhem no departamento de 'Pesquisa'.

```
Q1:SELECT      PNAME, UNOME, ENDereco
          FROM        EMPREGADO
          WHERE       DNO IN (SELECT DNUMERO
                              FROM        DEPARTAMENTO
                              WHERE       DNAME='Pesquisa' )
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

48

48

Consultas Aninhadas (cont.)

- A consulta aninhada seleciona o número do departamento de 'Pesquisa'
- A consulta externa seleciona uma tupla se seu valor de DNO estiver no resultado da consulta aninhada
- O operador de comparação IN compara um valor v com um conjunto (ou multiconjunto) V de valores, e avalia para verdade se v for um dos elementos em V
- Em geral, pode haver vários níveis de consultas aninhadas
- Uma referência a um atributo não qualificado é atribuída à relação declarada na consulta mais interna do aninhamento

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

49

49

Consultas Aninhadas Correlacionadas

- Se uma condição na cláusula WHERE de uma consulta aninhada se referir a algum atributo da relação declarada na consulta externa, as duas consultas são chamadas correlacionadas
 - O resultado de uma consulta aninhada correlacionada é diferente para cada tupla (ou combinação de tuplas) de relações da consulta externa
- Consulta 12: Recupere o nome de cada empregado que tenha um dependente com o mesmo primeiro nome como o empregado.

```
Q12: SELECT      E.PNOME, E.UNOME
      FROM        EMPREGADO AS E
      WHERE       E.SSN IN
                  (SELECT      ESSN
                   FROM        DEPENDENTE
                   WHERE       ESSN=E.SSN AND
                              E.PNOME=DEPENDENTE_NOME)
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

50

50

Consultas Aninhadas Correlacionadas (cont.)

- Em Q12, a consulta aninhada tem um resultado diferente na consulta externa
- Uma consulta escrita com blocos SELECT... FROM... WHERE... aninhados e usando o operador de comparação = ou IN pode sempre ser expressada como um bloco simples de consulta. Por exemplo, Q12 pode ser escrita como em Q12A

```
Q12A:  SELECT      E.PNOME, E.UNOME
        FROM        EMPREGADO E, DEPENDENTE D
        WHERE       E.SSN=D.ESSN AND
                   E.PNOME=D.DEPENDENTE_NOME
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

51

51

Consultas Aninhadas Correlacionadas (cont.)

- Consulta 3: Recupere o nome de cada empregado que trabalha em todos os projetos controlados pelo departamento número 5.

```
Q3:     SELECT      PNOME, UNOME
        FROM        EMPREGADO
        WHERE (      (SELECT      PNO
                        FROM        TRABALHA_EM
                        WHERE       SSN=ESSN)
                   CONTAINS
                   (SELECT      PNUMERO
                        FROM        PROJETO
                        WHERE       DNUM=5) )
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

52

52

Consultas Aninhadas Correlacionadas (cont.)

- EM Q3, a segunda consulta aninhada, que não é correlacionada com a consulta externa, recupera os números de todos os projetos controlados pelo departamento 5
- A primeira consulta aninhada, que é correlacionada, retorna os números dos projetos em que o empregado trabalha, que é diferente para cada tupla de empregado, por causa da correlação.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

53

53

A Função EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém tuplas) ou não
 - A consulta 12 pode ser formulada de forma alternativa usando EXISTS

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

54

54

A Função EXISTS (cont.)

- Consulta 12: Recupera o nome de cada empregado que tem um dependente com o mesmo primeiro nome com do empregado.

```
Q12B:  SELECT  PNAME, UNOME
        FROM    EMPREGADO
        WHERE   EXISTS (SELECT *
                        FROM    DEPENDENTE
                        WHERE   SSN=ESSN
                        AND
                        PNAME=DEPENDENTE_NAME)
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

55

55

A Função EXISTS (cont.)

- Consulta 6: Recupere os nomes dos empregados que não têm dependentes.

```
Q6:    SELECT  PNAME, UNOME
        FROM    EMPREGADO
        WHERE   NOT EXISTS (SELECT *
                        FROM    DEPENDENTE
                        WHERE   SSN=ESSN)
```

- EM Q6, a consulta aninhada correlacionada recupera todas as tuplas DEPENDENTE relacionadas a uma tupla EMPREGADO. Se não existir nenhuma, a tupla EMPREGADO será selecionada
 - EXISTS é necessária para a expressividade da SQL

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

56

56

Conjuntos Explícitos

- Também é possível usar um **conjunto explícito (enumerado) de valores** na cláusula WHERE ao invés de uma consulta aninhada
- Consulta 13: Recupere os números do seguro social de todos os empregados que trabalhem nos projetos número 1, 2, ou 3.

```
Q13:  SELECT  DISTINCT ESSN
      FROM    TRABALHA_EM
      WHERE   PNO IN (1, 2, 3)
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

57

57

NULLs em Consultas SQL

- A SQL permite que consultas verifiquem se um valor é **NULL** (desconhecido, omitido ou não aplicável)
- A SQL usa **IS** ou **IS NOT** para comparar NULLs porque cada NULL é considerado diferente de outros valores NULLs; portanto, comparação de igualdade não é apropriada.
- Consulta 14: Recupere os nomes de todos os empregados que não têm supervisor.

```
Q14:  SELECT  PNAME, UNOME
      FROM    EMPREGADO
      WHERE   SUPERSSN IS NULL
```

- Nota: Se uma condição de junção for especificada, tuplas com valores NULL para o atributo de junção não são incluídas no resultado

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

58

58

Características de Junção de Relações em SQL2

- Pode-se especificar uma "junção de relações" na cláusula FROM
 - Se parece com qualquer outra relação, mas é o resultado de uma junção
 - Permite ao usuário especificar diferentes tipos de junções (regular JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

59

59

Características de Junção de Relações em SQL2 (cont.)

- Exemplos:

Q8:SELECT	E.PNOME, E.UNOME, S.PNOME, S.UNOME
FROM	EMPREGADO E S
WHERE	E.SUPERSSN=S.SSN
- Pode ser escrito como:

Q8:SELECT	E.PNOME, E.UNOME, S.PNOME, S.UNOME
FROM	(EMPREGADO E LEFT OUTER JOIN
	EMPREGADO S ON E.SUPERSSN=S.SSN)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

60

60

Características de Junção de Relações em SQL2 (cont.)

- Exemplos:

```
Q1:SELECT      PNAME, UNOME, ENDereco
FROM      EMPREGADO, DEPARTAMENTO
WHERE      DNAME='Pesquisa' AND DNUMERO=DNO
```

- Pode ser escrito como:

```
Q1:SELECT      PNAME, UNOME, ENDereco
FROM      (EMPREGADO JOIN DEPARTAMENTO
           ON DNUMERO=DNO)
WHERE      DNAME='Pesquisa'
```

- ou como:

```
Q1:SELECT      PNAME, UNOME, ENDereco
FROM      (EMPREGADO NATURAL JOIN
           DEPARTAMENTO
           AS DEPT(DNAME, DNO, GERSSN, GERDATA))
WHERE      DNAME='Pesquisa'
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

61

61

Características de Junção de Relações em SQL2 (cont.)

- Outro exemplo: Q2 pode ser escrito como segue; Isso ilustra múltiplas junções

```
Q2:      SELECT      PNUMERO, DNUM, UNOME,
                   ENDereco, DATANASC
FROM      ((PROJETO JOIN DEPARTAMENTO
           ON DNUM=DNUMERO) JOIN
           EMPREGADO ON
           GERSSN=SSN)
WHERE      PLOCALIZACAO='Stafford'
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

62

62

Funções de Agregação

- Inclui **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**
- Consulta 15: Encontre o salário máximo, mínimo e a média de salário entre todos os empregados.

```
Q15:  SELECT  MAX(SALARIO),  
          MIN(SALARIO),  
          AVG(SALARIO)  
FROM    EMPREGADO
```

- Algumas implementações da SQL podem não permitir mais de uma função na cláusula **SELECT**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

63

63

Funções de Agregação (cont.)

- Consulta 16: Encontre o salário máximo, mínimo e a média de salário entre os empregados que trabalham para o departamento 'Pesquisa'

```
Q16:  SELECT  MAX(SALARIO),  
          MIN(SALARIO),  
          AVG(SALARIO)  
FROM    EMPREGADO,  
          DEPARTAMENTO  
WHERE    DNO=DNUMERO AND  
          DNAME='Pesquisa'
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

64

64

Funções de Agregação (cont.)

- Consulta 17 e 18: Recupere o número total de empregados da empresa (Q17), e o número de empregados do departamento 'Pesquisa' (Q18).

Q17: SELECT COUNT (*)
 FROM EMPREGADO

Q18: SELECT COUNT (*)
 FROM EMPREGADO, DEPARTAMENTO
 WHERE DNO=DNUMERO AND
 DNAME='Pesquisa'

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

65

65

Agrupamento

- Em muitos casos, precisa-se aplicar as funções de agregação a subgrupos de tuplas de uma relação
- Cada subgrupo de tuplas consistirá em tuplas que tenham o mesmo valor em algum de seus atributos, chamado atributo(s) de agrupamento
- A função de agregação é aplicada a cada subgrupo independentemente
- A SQL tem uma cláusula **GROUP BY** para especificar os atributos de agrupamento, que devem aparecer também na cláusula SELECT

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

66

66

Agrupamento (cont.)

- Consulta 20: Para cada departamento, recupere o número do departamento, o número de empregados do departamento e sua média salarial.

```
Q20:      SELECT      DNO, COUNT (*), AVG (SALARIO)
           FROM        EMPREGADO
           GROUP BY    DNO
```

- As tuplas de EMPREGADO são divididas em grupos
 - Cada grupo tendo o mesmo valor para o atributo de agrupamento DNO
- As funções COUNT e AVG são aplicadas a cada subgrupo de tuplas, separadamente
- A cláusula SELECT inclui somente os atributos de agrupamento e as funções a serem aplicadas a cada subgrupo de tuplas
- Uma condição de junção pode ser usada em conjunto com o agrupamento

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

67

67

Agrupamento (cont.)

- Consulta 21: Para cada projeto, recupere o número do projeto, o nome e o número de empregados que trabalham nesse projeto.

```
Q21:      SELECT      PNUMERO, PNUMERO, COUNT (*)
           FROM        PROJETO, TRABALHA_EM
           WHERE        PNUMERO=PNO
           GROUP BY    PNUMERO, PNUMERO
```

- Neste caso, o agrupamento e a função são aplicados após a junção das duas relações

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

68

68

A Cláusula HAVING

- As vezes, precisa-se recuperar valores de funções de agregação somente para os grupos que satisfazem certas condições
- A cláusula **HAVING** é usada para especificar uma condição de seleção em grupos (ao invés de em tuplas individuais)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

69

69

A Cláusula HAVING (cont.)

- Consulta 22: Para cada projeto em que mais de dois empregados trabalham, recupere o número do projeto, o nome e o número de empregados que trabalham nesse projeto.

```
Q22:  SELECT  PNUMERO, PNAME,
           COUNT(*)
        FROM    PROJETO, TRABALHA_EM
        WHERE   PNUMERO=PNO
        GROUP BY PNUMERO, PNAME
        HAVING  COUNT (*) > 2
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

70

70

Comparação de Substring

- O operador de comparação **LIKE** é usado para comparar strings parciais
- Dois caracteres reservados são usados: '%' (ou '*' em algumas implementações) substitui um número arbitrário de caracteres, e '_' substitui um único caractere

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

71

71

Comparação de Substring (cont.)

- Consulta 25: Recupere todos os empregados cujo endereço é Houston, Texas. Aqui, o valor do atributo ENDERECO deve conter a substring 'Houston,TX'.

```
Q25:  SELECT  PNAME, UNOME
        FROM    EMPREGADO
        WHERE
                ENDERECO LIKE '%Houston,TX%'
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

72

72

SUBSTRING COMPARISON (contd.)

- Consulta 26: Recupere todos os empregados que nasceram durante a década de 1950s.
 - Aqui, '5' deve ser o 8º caractere da string (de acordo com o formato da data), assim o valor de DATANASC deve ser '_____5_', com cada sublinhado como um lugar para um caractere.

Q26: SELECT PNOME, UNOME
 FROM EMPREGADO
 WHERE DATANASC LIKE '_____5_'

- A operação LIKE permite quebrar a impressão de que cada valor é atômico e indivisível
 - Consequentemente, em SQL, atributos string não são atômicos

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

73

73

Operações Aritméticas

- As operações aritméticas padrão '+', '-', '*', e '/' (para adição, subtração, multiplicação e divisão, respectivamente) podem ser aplicadas a valores numéricos em resultados de consultas SQL
- Consulta 27: Mostra o efeito de dar um aumento de 10% a todos os empregados que trabalham no projeto 'ProductX'.

Q27: SELECT PNOME, UNOME, 1.1*SALARIO
 FROM EMPREGADO, TRABALHA_EM,
 PROJECT
 WHERE SSN=ESSN AND PNO=PNUMERO
 AND PNOME='ProductX'

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

74

74

ORDER BY

- A cláusula **ORDER BY** é usada para ordenar as tuplas no resultado de uma consulta baseado nos valores de alguns atributos
- Consulta 28: Recupere a lista de empregados e o projeto em que trabalham, ordenado pelo departamento e, dentro do departamento, ordenado alfabeticamente pelo último nome do empregado.

```
Q28:      SELECT      DNAME, UNOME, PNAME, PNAME
          FROM        DEPARTAMENTO, EMPREGADO,
          TRABALHA_EM, PROJETO
          WHERE        DNUMERO=DNO AND SSN=ESSN
          AND PNO=PNUMERO
          ORDER BY     DNAME, UNOME
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

75

75

ORDER BY (cont.)

- A ordenação padrão (*default*) é em ordem ascendente de valores
- Pode-se especificar a palavra chave **DESC** se desejado ordenar de forma descendente; a palavra chave **ASC** pode ser usada para especificar explicitamente a ordenação ascendente, mesmo sendo *default*

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

76

76

Resumo de Consultas SQL

- Um consulta em SQL pode consistir de seis cláusulas, mas somente as duas primeiras, SELECT e FROM, são obrigatórias. As cláusulas são especificadas na seguinte ordem:

SELECT	<lista de atributos>
FROM	<lista de tabelas>
[WHERE	<condições>]
[GROUP BY	<atributo(s) de agrupamento>]
[HAVING	<condição de agrupamento>]
[ORDER BY	<lista de atributos>]

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

77

77

Resumo de Consultas SQL (cont.)

- A cláusula SELECT lista os atributos e funções a serem recuperadas
- A cláusula FROM especifica todas as relações (ou aliases) necessárias na consulta, mas não aquelas necessárias nas consultas aninhadas
- A cláusula WHERE especifica as condições para a seleção e junção das tuplas das relações especificadas na cláusula FROM
- A cláusula GROUP BY especifica os atributos de agrupamento
- A cláusula HAVING especifica uma condição para a seleção de grupos
- A cláusula ORDER BY especifica uma ordenação para mostrar o resultado de uma consulta
 - Uma consulta é avaliada, primeiramente, aplicando a cláusula WHERE, depois a GROUP BY e a HAVING, e, finalmente, a cláusula SELECT

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

78

78