

SQL – Structured Query Language

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Sumário

- **SQL – versões**
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

SQL – versões

- Linguagem padrão para SGBD relacionais comerciais
 - SQL-86 - padrão proposto por ANSI e ISO
 - SQL2 ou SQL-92 - padrão revisado e expandido
 - SQL3 ou SQL-99 – última versão
-

Sumário

- SQL – versões
 - **CREATE SCHEMA**
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

CREATE SCHEMA

- Especifica novo esquema
- Exemplo:

CREATE SCHEMA EMPRESA;

Sumário

- SQL – versões
 - CREATE SCHEMA
 - **CREATE TABLE, DROP TABLE, ALTER TABLE**
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

CREATE TABLE

- Comando CREATE TABLE especifica:
 - Chave primária
 - Chaves secundárias
 - Chaves estrangeiras
- Exemplo:

```
CREATE TABLE DEPT
(   DNAME      VARCHAR(10) NOT NULL,
    DNUMBER    INTEGER      NOT NULL,
    MGRSSN     CHAR(9),
    MGRSTARTDATE CHAR(9),
    PRIMARY KEY (DNUMBER),
    UNIQUE (DNAME),
    FOREIGN KEY (MGRSSN) REFERENCES EMP );
```

DROP TABLE

- Remove relação e esquema
- Exemplo:

DROP TABLE DEPENDENT;

ALTER TABLE

- Inclui atributos em relações
 - ❑ Valor do atributo será NULL para todas as tuplas na relação
 - ❑ Novo valor para o atributo pode ser alterado com comando UPDATE
- Exemplo:

```
ALTER TABLE EMPLOYEE ADD JOB  
VARCHAR(12);
```

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - **Opções de Integridade Referencial**
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Opções de Integridade Referencial

- No caso de remoção, atualização (ON DELETE, ON UPDATE)
 - Default: rejeitar a operação que viola integridade referencial
 - Outras opções especificadas explicitamente: CASCADE, SET NULL, SET DEFAULT

```
CREATE TABLE DEPT
( DNAME VARCHAR(10) NOT NULL,
  DNUMBER INTEGER NOT NULL,
  MGRSSN CHAR(9),
  MGRSTARTDATE CHAR(9),
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMP
    ON DELETE SET DEFAULT
    ON UPDATE CASCADE );
```

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - **Consultas**
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Consultas

- Comando SELECT
 - Diferença entre SQL e o modelo relacional:
 - uma tabela em SQL pode ter tuplas idênticas
 - Atributos PRIMARY KEY ou UNIQUE podem forçar relações a serem conjuntos
 - Consultas correspondem às operações de álgebra relacional: seleção, projeção e junção
-

Consultas

SELECT <attribute list>
FROM <table list>
WHERE <condition>

- <attribute list> lista de atributos a serem recuperados
 - <table list> lista de relações necessárias para processar a consulta
 - <condition> expressão condicional (Boolean) que identifica as tuplas a serem recuperadas
-

Consultas – outras cláusulas

SELECT <attribute list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]

- Avaliação da consulta:
 1. Cláusula WHERE,
 2. Cláusulas GROUP BY e HAVING,
 3. Cláusula SELECT
-

Consultas – exemplo

EMPLOYEE

| | | | | | | | | | |
|-------|-------|-------|------------|-------|---------|-----|--------|----------|-----|
| FNAME | MINIT | LNAME | <u>SSN</u> | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
|-------|-------|-------|------------|-------|---------|-----|--------|----------|-----|

DEPARTMENT

| | | | |
|-------|----------------|--------|--------------|
| DNAME | <u>DNUMBER</u> | MGRSSN | MGRSTARTDATE |
|-------|----------------|--------|--------------|

DEPT_LOCATIONS

| | |
|----------------|------------------|
| <u>DNUMBER</u> | <u>DLOCATION</u> |
|----------------|------------------|

PROJECT

| | | | |
|-------|----------------|-----------|------|
| PNAME | <u>PNUMBER</u> | PLOCATION | DNUM |
|-------|----------------|-----------|------|

WORKS_ON

| | | |
|-------------|------------|-------|
| <u>ESSN</u> | <u>PNO</u> | HOURS |
|-------------|------------|-------|

DEPENDENT

| | | | | |
|-------------|-----------------------|-----|-------|--------------|
| <u>ESSN</u> | <u>DEPENDENT_NAME</u> | SEX | BDATE | RELATIONSHIP |
|-------------|-----------------------|-----|-------|--------------|

Consultas – exemplo

| EMPLOYEE | FNAME | MINIT | LNAME | SSN | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
|----------|----------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| | John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 333445555 | 5 |
| | Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 |
| | Alicia | J | Zelaya | 999887777 | 1968-07-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| | Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 |
| | Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 |
| | Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| | Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| | James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | null | 1 |

| | | | | | DEPT_LOCATIONS | DNUMBER | DLOCATION |
|--|--|--|--|--|----------------|---------|-----------|
| | | | | | | 1 | Houston |
| | | | | | | 4 | Stafford |
| | | | | | | 5 | Bellaire |
| | | | | | | 5 | Sugarland |
| | | | | | | 5 | Houston |

| DEPARTMENT | DNAME | DNUMBER | MGRSSN | MGRSTARTDATE |
|------------|----------------|---------|-----------|--------------|
| | Research | 5 | 333445555 | 1988-05-22 |
| | Administration | 4 | 987654321 | 1995-01-01 |
| | Headquarters | 1 | 888665555 | 1981-06-19 |

| WORKS_ON | ESSN | PNO | HOURS |
|----------|-----------|-----|-------|
| | 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 |

| PROJECT | PNAME | PNUMBER | PLOCATION | DNUM |
|---------|-----------------|---------|-----------|------|
| | ProductX | 1 | Bellaire | 5 |
| | ProductY | 2 | Sugarland | 5 |
| | ProductZ | 3 | Houston | 5 |
| | Computerization | 10 | Stafford | 4 |
| | Reorganization | 20 | Houston | 1 |
| | Newbenefits | 30 | Stafford | 4 |

| DEPENDENT | ESSN | DEPENDENT_NAME | SEX | BDATE | RELATIONSHIP |
|-----------|-----------|----------------|-----|------------|--------------|
| | 333445555 | Alice | F | 1986-04-05 | DAUGHTER |
| | 333445555 | Theodore | M | 1983-10-25 | SON |
| | 333445555 | Joy | F | 1958-05-03 | SPOUSE |
| | 987654321 | Abner | M | 1942-02-28 | SPOUSE |
| | 123456789 | Michael | M | 1988-01-04 | SON |
| | 123456789 | Alice | F | 1988-12-30 | DAUGHTER |
| | 123456789 | Elizabeth | F | 1967-05-05 | SPOUSE |

Consultas – exemplo

- Query 0: recupera birthdate e address do empregado chamado 'John Smith'.

```
Q0: SELECT      BDATE, ADDRESS  
      FROM      EMPLOYEE  
      WHERE      FNAME='John' AND LNAME='Smith'
```

- Similar a seleção-projeção da álgebra relacional:
 1. cláusula WHERE especifica a condição de seleção
 2. cláusula SELECT especifica projeção
- Resultado pode conter tuplas duplicadas

Consultas – exemplo

- Query 1: recupera name e address dos empregados que trabalham no departamento 'Research'.

**Q1: SELECT FNAME, LNAME, ADDRESS
 FROM EMPLOYEE, DEPARTMENT
 WHERE DNAME='Research' AND
 DNUMBER=DNO**

- Similar a seleção-projeção-junção da álgebra relacional:
 - (DNAME='Research') é condição de seleção
 - (DNUMBER=DNO) é condição de junção

Uso de *

- Recupera todos os atributos
- Exemplos:

Q1C: SELECT *
FROM EMPLOYEE
WHERE DNO=5

Q1D: SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND
DNO=DNUMBER

Uso de DISTINCT

- Elimina tuplas duplicatas de uma consulta
- Resultado de Q11 pode ter valores duplicados de SALARY:

**Q11: SELECT SALARY
 FROM EMPLOYEE**

- Q11A não tem valores duplicados:

**Q11A: SELECT DISTINCT SALARY
 FROM EMPLOYEE**

NULLS

- **IS** ou **IS NOT** compara NULLs.
- Query 14: recupera nome de todos os empregados que não têm supervisores.

**Q14: SELECT FNAME, LNAME
 FROM EMPLOYEE
 WHERE SUPERSSN IS NULL**

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - **Operações de Conjunto**
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Operações de Conjunto

- **UNION**

- **MINUS**

- **INTERSECT**

- ❑ Tuplas duplicadas são eliminadas
 - ❑ As duas relações devem ter mesmos atributos na mesma ordem
-

Operações de Conjunto

- Query 4: lista os números dos projetos que têm um empregado de sobrenome 'Smith' como trabalhador ou regente do departamento que controla o projeto.

Q4: (SELECT PNAME
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
LNAME='Smith')
UNION

(SELECT PNAME
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND
LNAME='Smith')

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - **Consultas Aninhadas**
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Consultas Aninhadas

- Especificadas dentro da cláusula WHERE de outra consulta (*outer query*)
- Query 1: recupera name e address de todos os empregados que trabalham no departamento 'Research'.

```
Q1: SELECT      FNAME, LNAME, ADDRESS
      FROM      EMPLOYEE
      WHERE      DNO IN
                (SELECT DNUMBER
                  FROM    DEPARTMENT
                  WHERE    DNAME='Research' )
```

Consultas Aninhadas Correlacionadas

- Se uma condição na cláusula WHERE de uma consulta aninhada referencia um atributo de uma relação na consulta mais externa, as consultas são chamadas correlacionadas.
- Query 12: recupera o nome de cada empregado que tem um dependente com mesmo nome.

```
Q12: SELECT      E.FNAME, E.LNAME
      FROM        EMPLOYEE AS E
      WHERE       E.SSN IN
                  (SELECT      ESSN
                   FROM DEPENDENT
                   WHERE        ESSN=E.SSN AND
                               E.FNAME=DEPENDENT_NAME)
```

Consultas Aninhadas Correlacionadas

- Uma consulta com blocos aninhados SELECT... FROM... WHERE... usando IN pode ser expressa em um bloco único.
- Q12 pode ser escrita como Q12A

```
Q12A:  SELECT  E.FNAME, E.LNAME  
        FROM    EMPLOYEE E, DEPENDENT D  
        WHERE   E.SSN=D.ESSN AND  
                E.FNAME=D.DEPENDENT_NAME
```

Conjuntos Explícitos

- Ao invés de consulta aninhada, é possível enumerar um conjunto de valores explicitamente.
- Query 13: recupera SSN de todos os empregados que trabalham nos projetos de número 1, 2 ou 3.

Q13: **SELECT DISTINCT ESSN**
 FROM WORKS_ON
 WHERE PNO IN (1, 2, 3)

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - **Junção**
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Junção

- Especificada na cláusula FROM

**Q1: SELECT FNAME, LNAME, ADDRESS
 FROM EMPLOYEE, DEPARTMENT
 WHERE DNAME='Research' AND DNUMBER=DNO**

- Ou:

**Q1: SELECT FNAME, LNAME, ADDRESS
 FROM (EMPLOYEE JOIN DEPARTMENT
 ON DNUMBER=DNO)
 WHERE DNAME='Research'**

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - **Funções de Agregação**
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Funções de Agregação

- **COUNT, SUM, MAX, MIN, AVG**
- Query 15: encontra valor máximo, mínimo e médio de salário entre todos os empregados.

Q15: **SELECT** **MAX(SALARY),**
 MIN(SALARY),
 AVG(SALARY)
 FROM **EMPLOYEE**

- Queries 17: encontra número total de empregados na empresa.

Q17: **SELECT** **COUNT (*)**
 FROM **EMPLOYEE**

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - **Cláusula GROUP BY**
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Cláusula Group By

- Especifica agrupamento de atributos.
 - Cada subgrupo tem mesmo valor nos atributos de agrupamento.
 - A função de agregação é aplicada para cada subgrupo.
-

Cláusula Group By

- Query 20: para cada departamento, recupera o número do departamento, número de empregados no departamento e sua média de salário.

```
Q20:SELECT      DNO, COUNT (*), AVG (SALARY)
      FROM      EMPLOYEE
      GROUP BY   DNO
```

- Tuplas de EMPLOYEE são agrupadas levando-se em conta o atributo DNO – cada grupo tem o mesmo valor para DNO
 - Funções COUNT e AVG funções são aplicadas para cada grupo
 - Cláusula SELECT inclui atributo de agrupamento e funções de agregação
-

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - **Cláusula HAVING**
 - Cláusula ORDER BY
 - Atualizações
 - Exercícios
-

Cláusula HAVING

- Especifica uma condição de seleção nos grupos.
- As funções de agregação são aplicadas somente nos grupos que satisfaçam a condição.
- Query 20A: para cada departamento no qual mais de 2 empregados trabalham, recupera o número do departamento, número de empregados no departamento e sua média de salário.

Q20A: **SELECT DNO, COUNT (*), AVG (SALARY)**
 FROM EMPLOYEE
 GROUP BY DNO
 HAVING COUNT (*) > 2

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - **Cláusula ORDER BY**
 - Atualizações
 - Exercícios
-

Cláusula ORDER BY

- Classifica tuplas de acordo com valores de atributos
- Query 28: recupera lista de empregados classificados pelo sobrenome.

| | | |
|-------------|-----------------|---------------------|
| Q28: | SELECT | LNAME, FNAME |
| | FROM | EMPLOYEE |
| | ORDER BY | LNAME |

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - **Atualizações**
 - Exercícios
-

Atualizações

- INSERT, DELETE, UPDATE



INSERT

- Insere tuplas em uma relação
- Exemplo:

```
U1: INSERT INTO EMPLOYEE  
VALUES ('Richard','K','Marini', '653298653',  
      '30-DEC-52',  
      '98 Oak Forest,Katy,TX', 'M',  
      37000,'987654321', 4 )
```

DELETE

- Remove tuplas de uma relação
- Exemplo:

**U4A: DELETE FROM EMPLOYEE
 WHERE LNAME='Brown'**

UPDATE

- Modifica valores de atributos das tuplas selecionadas
- Integridade referencial deve ser preservada
- Exemplo: modifica a localização e o número do departamento que controla o projeto número 10 para 'Bellaire' e 5, respectivamente.

**U5: UPDATE
SET
WHERE**

**PROJECT
PLOCATION = 'Bellaire', DNUM = 5
PNUMBER=10**

Sumário

- SQL – versões
 - CREATE SCHEMA
 - CREATE TABLE, DROP TABLE, ALTER TABLE
 - Opções de Integridade Referencial
 - Consultas
 - Operações de Conjunto
 - Consultas Aninhadas
 - Junção
 - Funções de Agregação
 - Cláusula GROUP BY
 - Cláusula HAVING
 - Cláusula ORDER BY
 - Atualizações
 - **Exercícios**
-

Exercícios – Funcionários e Dependentes

- F(numfd,nomef)
- D(numfd,nomed,par)

| F | |
|-------|-------|
| numfd | nomef |
| 01 | F1 |
| 02 | F2 |
| 03 | F3 |
| 04 | F4 |

| D | | |
|-------|-------|--------|
| numfd | nomed | par |
| 01 | Alice | filha |
| 02 | Alice | esposa |
| 02 | Clara | filha |
| 03 | José | filho |

Exercícios

- Quais os nomes e parentescos de todos os dependentes?
 - $\pi_{\text{nomed,par}} (D)$
- Quais funcionários possuem dependentes filhas?
 - $\pi_{\text{numfd}} (\sigma_{\text{par}='filha'} (D))$
- Quais funcionários não possuem dependentes?
 - $\pi_{\text{numfd}} (F) - \pi_{\text{numfd}} (D)$
- Dê os nomes dos funcionários que possuem algum dependente.
 - $\pi_{\text{noref}} (F |X| D)$
- Dê o nome de cada funcionário que possui uma dependente chamada Alice.
 - $\pi_{\text{noref}} (F |X| (\sigma_{\text{nomed}='Alice'} (D)))$

Exercícios

- Quais funcionários possuem mais de um dependente?

- $\rho_{D_1}(\text{numfd}, \text{nomed1}, \text{par1}) (D)$

- $\rho_{D_2}(\text{numfd}, \text{nomed2}, \text{par2}) (D)$

- $\pi_{\text{numfd}} ((\sigma_{\text{nomed1} \neq \text{nomed2}} (D_1 \mid X \mid D_2)))$

- Quais funcionários não têm Alice como dependente?

- $\pi_{\text{numfd}} (D) - \pi_{\text{numfd}} (\sigma_{\text{nomed} = \text{'Alice'}} (D))$

- Para cada funcionário que tem uma dependente chamada Alice, dê o número do funcionário e o nome dos outros dependentes, se houver.

- $\pi_{\text{numfd}, \text{nomed2}} (\sigma_{\text{nomed2} \neq \text{'Alice'}} (\sigma_{\text{nomed1} = \text{'Alice'}} (D_1 \mid X \mid D_2)))$

SQL – Aspectos Avançados

Sumário

- Asserções
 - Visões
 - Programação de BD
 - SQL embutida
 - SQL dinâmica
 - Biblioteca de funções
 - Procedimentos/Funções armazenados
-

Sumário

- **Asserções**
 - Visões
 - Programação de BD
 - ❑ SQL embutida
 - ❑ SQL dinâmica
 - ❑ Biblioteca de funções
 - ❑ Procedimentos/Funções armazenados
-

Asserções

- Mecanismo para especificar restrições adicionais
 - `CREATE ASSERTION` inclui:
 - ❑ Nome de restrição,
 - ❑ `CHECK` e
 - ❑ Condição
-

Asserções

- “O salário de um empregado não deve ser maior que o salário do gerente do departamento no qual ele trabalha”

```
CREAT ASSERTION SALARY_CONSTRAINT  
CHECK (NOT EXISTS (  
    SELECT *  
    FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D  
    WHERE E.SALARY > M.SALARY AND  
          E.DNO=D.NUMBER AND D.MGRSSN=M.SSN) )
```

Gatilhos

- Tipo especial de asserção
 - Objetivo: monitorar o BD e realizar alguma ação quando uma condição ocorre
 - Sintaxe:
 - evento (por ex, uma atualização)
 - condição
 - ação
-

Gatilhos - Exemplo

- Gatilho para comparar salário de um empregado com o salário de seu gerente durante operações de insert ou update:

```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
    SALARY, SUPERVISOR_SSN ON EMPLOYEE
FOR EACH ROW WHEN (NEW.SALARY >
    (SELECT SALARY
     FROM EMPLOYEE
     WHERE SSN=NEW.SUPERVISOR_SSN) )
    INFORM_SUPERVISOR(NEW.SUPERVISOR_SSN, NEW.SSN) ;
```

Sumário

- Asserções
 - **Visões**
 - Programação de BD
 - ❑ SQL embutida
 - ❑ SQL dinâmica
 - ❑ Biblioteca de funções
 - ❑ Procedimentos/Funções armazenados
-

Visões

- Uma visão é uma tabela virtual derivada de outras tabelas
 - Operações de update são limitadas
 - Operações de consulta são as mesmas de tabelas
-

Visões

- Comando `CREATE VIEW` inclui:
 - ❑ Nome da tabela (visão)
 - ❑ Lista de atributos
 - ❑ Uma consulta que determina o conteúdo da tabela.
-

Visões

- Criando nova tabela WORKS_ON:

```
CREATE TABLE WORKS_ON_NEW AS  
  SELECT FNAME, LNAME, PNAME, HOURS  
  FROM EMPLOYEE, PROJECT, WORKS_ON  
  WHERE SSN=ESSN AND PNO=PNUMBER  
  GROUP BY PNAME;
```

- Exemplo de consulta sobre a tabela:

```
SELECT FNAME, LNAME  
FROM WORKS_ON_NEW  
WHERE PNAME='MuseuVirtual';
```

- Remoção de visão:

```
DROP WORKS_ON_NEW;
```

Sumário

- Asserções
 - Visões
 - **Programação de BD**
 - ❑ SQL embutida
 - ❑ SQL dinâmica
 - ❑ Biblioteca de funções
 - ❑ Procedimentos/Funções armazenados
-

Programação de BD

- Objetivo: acessar um BD de um programa de aplicação
 - Motivação:
 - Interface interativa não é suficiente
 - A maioria das operações sobre BDs são chamadas de programas de aplicação (aplicações Web)
-

Programação de BD - abordagens

- Comandos embutidos em linguagem de programação (*programação estática*)
 - SQL dinâmica (*programação dinâmica*)
 - Biblioteca de funções de BD (*programação dinâmica*)
 - Procedimentos/funções armazenadas
-

Programação de BD - passos

1. Programa cliente abre conexão com servidor de BD
 2. Programa cliente submete consulta/atualizações ao BD
 3. Programa cliente termina a conexão
-

Sumário

- Asserções
 - Visões
 - Programação de BD
 - ❑ **SQL embutida**
 - ❑ SQL dinâmica
 - ❑ Biblioteca de funções
 - ❑ Procedimentos/Funções armazenados
-

SQL Embutida

- Comando SQL embutido em linguagens de programação:
 - ❑ EXEC SQL
 - ❑ END-EXEC (ou ;)
 - ❑ Variáveis compartilhadas (em ambas as linguagens) têm o prefixo : quando aparecem em comandos SQL
-

Exemplo: Declaração de Variáveis na Linguagem C

```
int loop;  
EXEC SQL BEGIN DECLARE SECTION;  
    varchar dname[16], fname[16], ...;  
    char ssn[10], bdate[11], ...;  
    int dno, dnumber, ...;  
EXEC SQL END DECLARE SECTION;
```

Exemplo: Comando Embutido na Linguagem C

```
loop = 1;
while (loop) {
    prompt ("Enter SSN: ", ssn);
    EXEC SQL
        select FNAME, LNAME, ADDRESS, SALARY
        into :fname, :lname, :address, :salary
        from EMPLOYEE where SSN == :ssn;
    if (SQLCODE == 0) printf(fname, ...);
    else printf("SSN does not exist: ", ssn);
    prompt("More SSN? (1=yes, 0=no): ", loop);
    END-EXEC
}
```

SQL Embutida em Java

- SQLJ: padrão para embutir comandos SQL em Java
 - Tradutor SQLJ converte comandos SQL para Java (executados pela interface JDBC)
 - Java Database Connectivity (JDBC)
 - Fornece chamadas de função de conexão
-

Passos para acesso ao BD via JDBC

1. Importar biblioteca JDBC (`java.sql.*`)
 2. Carregar driver JDBC:
`Class.forName("oracle.jdbc.driver.OracleDriver")`
 3. Definir variáveis
 4. Criar objeto de conexão (via `getConnection`)
 5. Criar objeto comando da classe `Statement`:
 1. `PreparedStatement`
 2. `CallableStatement`
-

Passos para acesso ao BD via JDBC

6. Ligar parâmetros a variáveis de programa
 7. Executar comando SQL via JDBC's
`executeQuery`
 8. Processar resultados de consulta
(retornados em objeto do tipo `ResultSet`)
-

Exemplo de SQL embutida em Java

```
ssn = readEntry("Enter a SSN: ");
try {
    #sql{select FNAME< LNAME, ADDRESS, SALARY
    into :fname, :lname, :address, :salary
    from EMPLOYEE
    where SSN = :ssn};
}
catch (SQLException se) {
    System.out.println("SSN does not exist: "+ssn);
    return;
}
System.out.println(fname+" "+lname+... );
```

Sumário

- Asserções
 - Visões
 - Programação de BD
 - SQL embutida
 - **SQL dinâmica**
 - Biblioteca de funções
 - Procedimentos/Funções armazenados
-

SQL Dinâmica

- Objetivo: executar comandos SQL em tempo de execução
 - Comandos SQL são passados pelo teclado
- Exemplo:

```
EXEC SQL BEGIN DECLARE SECTION;  
varchar sqlupdatestring[256];  
EXEC SQL END DECLARE SECTION;  
...  
prompt ("Enter update command:", sqlupdatestring);  
EXEC SQL PREPARE sqlcommand FROM :sqlupdatestring;  
EXEC SQL EXECUTE sqlcommand;
```

Sumário

- Asserções
 - Visões
 - Programação de BD
 - SQL embutida
 - SQL dinâmica
 - **Biblioteca de funções**
 - Procedimentos/Funções armazenados
-

Biblioteca de Funções

■ API:

- ❑ vantagem: não requer pré-processamento
- ❑ desvantagem: verificação de sintaxe deve ser feita em tempo de execução

■ SQL Call Level Interface (SQL/CLI): padrão SQL

- ❑ Comandos SQL são criados dinamicamente e passados como parâmetros do tipo string nas chamadas
-

Sumário

- Asserções
 - Visões
 - Programação de BD
 - SQL embutida
 - SQL dinâmica
 - Biblioteca de funções
 - **Procedimentos/Funções armazenados**
-

Procedimentos/Funções Armazenados

- Procedimentos/funções armazenados são executados pelo servidor de BD
 - Vantagens:
 - Podem ser chamados por várias aplicações (reduzindo duplicações)
 - Execução pelo servidor reduz custo de comunicação
-

Procedimentos/Funções Armazenados

- Criando procedimento armazenado

```
CREATE PROCEDURE procedure-name (params)
local-declarations
procedure-body;
```

- Criando função armazenada

```
CREATE FUNCTION fun-name (params) RETURNS return-type
local-declarations
function-body;
```

- Chamando procedimento/função

```
CALL procedure-name/fun-name (arguments);
```

Procedimentos/Funções Armazenados - Padrão

- SQL Persistent Stored Modules (SQL/PSM)
 - parte do padrão SQL para escrita de módulos armazenados persistentes

Procedimentos/Funções Armazenados - Exemplo

```
CREATE FUNCTION DEPT_SIZE (IN deptno INTEGER)
RETURNS VARCHAR[7]
DECLARE TOT_EMPS INTEGER;

SELECT COUNT (*) INTO TOT_EMPS
FROM SELECT EMPLOYEE WHERE DNO = deptno;

IF TOT_EMPS > 100 THEN RETURN "HUGE"
ELSEIF TOT_EMPS > 50 THEN RETURN "LARGE"
ELSEIF TOT_EMPS > 30 THEN RETURN "MEDIUM"
ELSE RETURN "SMALL"
ENDIF;
```
