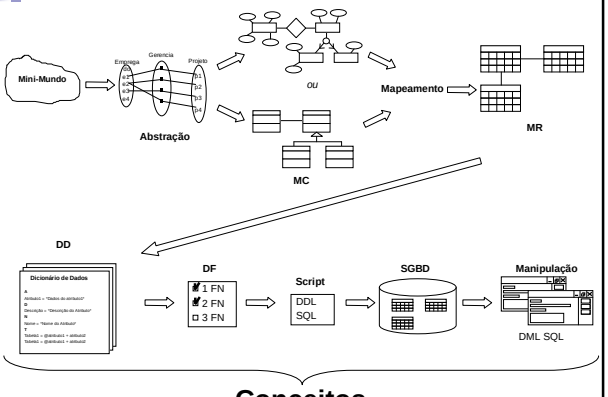
Banco de Dados I
DML

Osmar de Oliveira Braz Junior

Objetivos

- Conhecer a sintaxe do SQL DML.
- Utilizar um terminal para realizar consultas em um SGBD.



Conceitos

1. Linguagem de Manipulação de Dados

- Uma vez que o esquema esteja compilado e o banco de dados esteja populado, usa-se uma linguagem para fazer a manipulação dos dados, a **DML** (Data Manipulation Language - Linguagem de Manipulação de Dados).
- Por manipulação entendemos:
  - A recuperação das informações armazenadas no banco de dados;
  - Inserção de novas informações no banco de dados;
  - A remoção das informações no banco de dados;
  - A modificação das informações no banco de dados;

2. Estruturas Básicas(Esquema)

- Cidade = (nome_cidade, estado_cidade)
- Agencia = (nome_agencia, cidade_agencia, fundos)
- Cliente = (nome_cliente, rua_cliente, cidade_cliente,salario_cliente)
- Emprestimo = (numero_emprestimo, nome_agencia, total)
- Devedor = (nome_cliente, numero_emprestimo)
- Conta = (numero_conta, nome_agencia, saldo)
- Depositante = (nome_cliente, numero_conta)
- Departamento = (codigo_departamento, nome_departamento,total_salario)
- Funcionário = (rg_funcionario, nome_funcionario, cpf_funcionario, salario_funcionario, rg_supervisor, codigo_departamento)

2. Estruturas Básicas

- A estrutura básica consiste de três cláusulas: select, from e where.
- **SELECT** – Ela é usada para relacionar os atributos desejados no resultado de uma consulta.
- **FROM** – Ela associa as relações que serão pesquisadas durante a evolução de uma expressão.
- **WHERE** – Ela consiste em um predicado envolvendo atributos da relação que aparece na cláusula from.

2. Estruturas Básicas

- Uma consulta típica em SQL tem a seguinte forma:

```
SELECT A1, A2, .., An
FROM R1, R2, ..., Rn
WHERE P;
```

- Cada Ai representa um atributo em cara Ri uma relação. P é um predicado.

7

2. Estruturas Básicas
2.1. Cláusula Select

- O comando **select** permite a seleção de tuplas e atributos em uma ou mais tabelas.
- Exemplo: "*Encontre os nomes de todos os clientes*".

```
SELECT nome_cliente
FROM cliente;
```

8

2. Estruturas Básicas
2.1. Cláusula Select

- *Especificador Distinct*

  Eliminação da duplicidade

```
SELECT DISTINCT cidade_cliente
FROM cliente;
```

- *Operador* *****

  Seleciona todos os atributos de uma tabela

```
SELECT *
FROM cliente;
```

9

2. Estruturas Básicas
2.1. Cláusula Select

- *Expressões Aritméticas*

  A cláusula select pode conter expressões aritméticas envolvendo os operadores **+, -, /** e ***** e operandos constantes ou atributos das tuplas:
- Exemplo: "*Mostre o nome dos clientes e o seu salário reajustado em 10%*"

```
SELECT nome_cliente, salario_cliente * 1.1
FROM cliente;
```

10

2. Estruturas Básicas
2.2. Cláusula Where

- *Critérios*

  Considere a consulta "*encontre todos os nomes dos clientes que ganham mais de 2400*".
- Esta consulta pode ser escrita em SQL como:

```
SELECT nome_cliente
FROM CLIENTE
WHERE salario_cliente > 2400;
```

11

2. Estruturas Básicas
2.2. Cláusula Where

- *Conectores Lógicos*

  A SQL usa conectores lógicos **and**, **or** e **not** na cláusula where. Os operandos dos conectivos lógicos podem ser expressões envolvendo operadores de comparação **<, <=, >, >=, =** e **!=**.
- Pode ser utilizando ainda o <> no lugar do !=.

12

2. Estruturas Básicas
2.2. Cláusula Where

- **Claúsula BetWeen**
  - Operador de comparação **between** para simplificar a cláusula where;
  - Especifica que um valor pode ser menor ou igual a algum valor e maior ou igual a algum outro valor.
  - Se desejarmos encontrar os números de empréstimos cujos montantes estejam entre 90 mil dólares e 100 mil dólares, podemos usar a comparação **between** escrevendo:

```
SELECT numero_emprestimo
FROM emprestimo
WHERE total between 90000 and 100000;
```

  - em vez de

```
SELECT numero_emprestimo
FROM emprestimo
WHERE total >=90000 and total <= 100000;
```

- negação **not between.**

13

2. Estruturas Básicas
2.3. Cláusula From

- A cláusula **FROM** por si só define um produto cartesiano das relações da cláusula.
- Uma vez que a junção natural é definida em termos de **produto cartesiano**, uma seleção é uma projeção são um meio relativamente simples de escrever a expressão SQL para uma junção natural.

14

2. Estruturas Básicas
2.3. Cláusula From

- Escrevemos a expressão em álgebra relacional:

$$\Pi_{\text{nome\_cliente, numero\_emprestimo}}\left(\sigma_{\text{devedor.numero\_emprestimo = emprestimo.numero\_emprestimo}}(\text{devedor} \times \text{emprestimo})\right)$$

para a consulta "para todos os clientes que tenham um empréstimo em um banco, encontre seus nomes e números de empréstimos".

- Em SQL, essa consulta pode ser escrita como:

```
SELECT distinct nome_cliente, devedor.numero_emprestimo
FROM devedor, emprestimo
WHERE
      devedor.numero_emprestimo =
      emprestimo.numero_emprestimo;
```

15

2. Estruturas Básicas
2.3. Cláusula From

- Note que a SQL usa a notação **`nome_relação.nome_atributo`**, como na álgebra relacional para evitar ambiguidades nos casos em que um atributo aparecer no esquema mais de uma relação.

16

2. Estruturas Básicas
2.4. Operação Rename

- A SQL proporciona um mecanismo para rebatizar tanto relações quanto atributos, usando a cláusula **as**, da seguinte forma:

```
nome_antigo as nome_novo
```

- Considere novamente a consulta usada anteriormente:

```
SELECT distinct nome_cliente, devedor.numero_emprestimo
FROM devedor, emprestimo
WHERE devedor.numero_emprestimo =
            emprestimo.numero_emprestimo;
```

- Por exemplo se desejarmos que o nome do atributo nome_cliente seja substituido pelo nome nome_devedor, podemos reescrever a consulta como:

```
SELECT distinct nome_cliente as devedor,
            devedor.numero_emprestimo
FROM devedor, emprestimo
WHERE devedor.numero_emprestimo =
            emprestimo.numero_emprestimo;
```

17

2. Estruturas Básicas
2.5. Variáveis Tuplas

- Variáveis tuplas são definidas na cláusula **from** por meio do uso da cláusula **as**.
- Com isto é possível renomear uma relação
- Váriaveis tuplas são úteis para comparação de duas tuplas de mesma relação.

18

2. Estruturas Básicas
2.5. Variáveis Tuplas

- Exemplo

  Funcionário
  - (rg_funcionario, nome_funcionario, cpf_funcionario,salario_funcionario,rg_supervisor, codigo_departamento)

- Encontre o nome de todos funcionários e o nome do seu supervisor(Supervisor também é um funcionário:

```
SELECT distinct
        F.nome_funcionario, S.nome_funcionario
FROM Funcionario AS F, Funcionario AS S
WHERE F.RG_Funcionario = S.RG_Supervisor;
```

19

2. Estruturas Básicas
2.6. Operações em String

- As operações em strings mais usadas são as checagens para verificação de coincidências de pares, usando o operador **like**. Indicaremos esses pares por meio do uso de dois caracteres especiais:
  - Porcentagem (%) : o caracter % compara qualquer substring.
  - Sublinhado (_) : o caracter _ compara qualquer caracter.
- Para pesquisar diferenças em vez de coincidências, use o operador **not like**.

20

2. Estruturas Básicas
2.6. Operações em String

- Exemplos

  Comparações desse tipo são sensíveis ao tamanho das letras; isto é, minúsculas não são iguais a maiúsculas, e vice-versa.
- Para ilustrar considere os seguintes exemplos;
  - **"Pedro%"** corresponde a qualquer string que comece com "Pedro"
  - **"%inh%"** corresponde a qualquer string que possua uma substring "inh", por exemplo "huguinho", "zezinho" e "luizinho"
  - **"_ _ _"** corresponde a qualquer string com exatamente três caracteres
  - **"_ _ _ %"** corresponde a qualquer string com pelo menos três caracteres

21

2. Estruturas Básicas
2.6. Operações em String

- Consulta

  Considere a consulta "*encontre os nomes de todos os clientes cujos os nomes possuam a substring 'Silva'* ".
- Esta consulta pode ser escrita assim:

```
SELECT nome_cliente
FROM cliente
WHERE nome_cliente LIKE '%Silva%';
```

22

2. Estruturas Básicas
2.7. Precedência dos Operadores

Ordem de Avaliação	Operadores
1	Operadores Aritméticos
2	Operador de Concatenação
3	Condições de Comparação
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Condição lógica NOT
7	Condição lógica AND
8	Condição lógica OR

Para modificar a ordem de avaliação utilize parênteses.

23

2. Estruturas Básicas
2.8. Ordenação

- A SQL oferece ao usuário algum controle sobre a ordenação por meio da qual as tuplas de uma relação serão apresentadas. A cláusula **order by** faz com que as tuplas do resultado de uma consulta apareçam em uma determinada ordem.
- Para listar em ordem alfabética todos os clientes, escrevemos:

```
SELECT nome_cliente, salario_cliente
FROM cliente
ORDER BY nome_cliente;
```

- Por default, a cláusula **order by** é em ordem **ascendente**.
- Especificação
  - **desc** para ordem descendente
  - **asc** para ordem ascedente.
- A ordenação pode ser realizada por diversos atributos.

24

2. Estruturas Básicas
2.9. Ordenação

- Para listar em ordem alfabética as cidades dos clientes:

```
SELECT distinct cidade_cliente
FROM cliente
ORDER BY cidade_cliente asc;
```

- Para listar em ordem alfabética as cidades e o nome dos clientes em ordem alfabética descendentes:

```
SELECT cidade_cliente, nome_cliente
FROM cliente
ORDER BY cidade_cliente asc, nome_cliente desc;
```

25

2. Estruturas Básicas

- Resolver os exercícios de DML Select (ExercicioDML_Select.pdf)

26

3. Composição de Relações

- Utilizamos junção (join) de tabelas para extrair dados de mais de uma tabela.

```
SELECT   R1.A1, R2.A2
FROM     R1, R2
WHERE    R1.A1 = R2.A2;
```

- Uma condição deve estar presente na cláusula WHERE, justificando a relação entre as tabelas envolvidas.
- Deve ser usado um prefixo para as colunas que tiverem o mesmo nome em mais de uma tabela.

27

3. Composição de Relações

- Um produto cartesiano é formado nas seguintes condições:
  - Uma condição de junção(join) é omitida
  - Uma condição de junção(join) é inválida
  - Todas as linhas da primeira tabela são multiplicadas com todas as linhas da segunda tabela
- Para impedir a formação de um produto cartesiano, sempre inclua uma condição de restrição válida na cláusula **WHERE**.

28

3. Composição de Relações

EMP (14 linhas)

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT (4 linhas)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

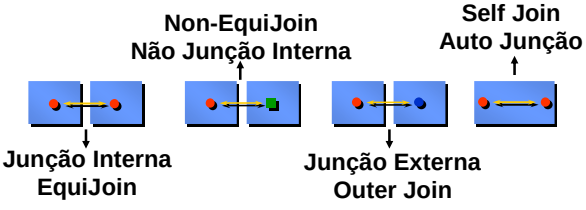
"Produto Cartesiano: 14*4=56 linhas"

ENAME	DNAME
KING	ACCOUNTING
BLAKE	ACCOUNTING
...	
KING	RESEARCH
BLAKE	RESEARCH
...	

56 rows selected.

29

3. Composição de Relações
3.1 Tipos de Junção



30

3. Composição de Relações
3.1 Tipos de Junção

- Claúsulas de Junção
  - `Inner join => join`
  - `Left outer join => left join`
  - `Right outer join => right join`
  - `Full outer join => full join`
- Condições de junção
  - `Natural`
  - `On <predicado>`
  - `Using (A1, A2, ..., An)`

31

3. Composição de Relações
3.2 Junção Interna

- **Junção Interna(Inner Join) é quando tuplas são incluídas no resultado de uma consulta somente se existir uma correspondente na outra relação.**
  - SELECT tabela1.atributo1, tabela2.atributo2
  - FROM tabela1 INNER JOIN tabela2
    ON tabela1.atributo1 = tabela2.atributo2;
- **Se o nome dos atributos for igual a cláusula ON e desnecessária, para isto usa-se o a Junção Natural com cláusula NATURAL.**
  - SELECT tabela1.atributo1, tabela2.atributo2
  - FROM tabela1 NATURAL INNER JOIN tabela2 ;
    OU
  - SELECT tabela1.atributo1, tabela2.atributo2
  - FROM tabela1 NATURAL JOIN tabela2 ;

32

3. Composição de Relações
3.2 Junção Interna

- **A condição de junção USING(A1, A2, ..., An) é similar à condição de junção natural exceto pelo fato de que seus atributos de junção são os A1, A2, .. An em vez de todos os atributos comuns a ambas as relações.**
- **Os atributos A1, A2, ..., An devem ser somente os atributos comuns a ambas as relações e eles aparecem apenas uma vez no resultado da junção**
  - SELECT tabela1.atributo1, tabela2.atributo2
  - FROM tabela1 NATURAL INNER JOIN tabela2 USING (atributo1);
    OU
  - SELECT tabela1.atributo1, tabela2.atributo2
  - FROM tabela1 NATURAL JOIN tabela2 USING (atributo1);

33

3. Composição de Relações
3.2 Junção Interna

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

... 14 rows selected.

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

... 14 rows selected.

Foreign key Primary key

34

3. Composição de Relações
3.3 Resultado da Junção Interna

```
SQL> SELECT   emp.empno, emp.ename, emp.deptno,
  2           dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ---------
 7839 KING       10     10 NEW YORK
 7698 BLAKE      30     30 CHICAGO
 7782 CLARK      10     10 NEW YORK
 7566 JONES      20     20 DALLAS
...
14 rows selected.
```

- **Usando as claúsulas FROM e WHERE para realizar o junção**

35

3. Composição de Relações
3.3 Resultado da Junção Interna

```
SQL>SELECT  emp.empno, emp.ename, emp.deptno,
  2         dept.deptno, dept.loc
  3 FROM    emp INNER JOIN dept ON
                emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ---------
 7839 KING       10     10 NEW YORK
 7698 BLAKE      30     30 CHICAGO
 7782 CLARK      10     10 NEW YORK
 7566 JONES      20     20 DALLAS
...
14 rows selected.
```

- **Use o INNER JOIN ou simplesmente JOIN para realizar junção sem where.**

36

3. Composição de Relações
3.3 Resultado da Junção Natural

```
SQL>SELECT  emp.empno, emp.ename, deptno, dept.loc
  2 FROM    emp NATURAL INNER JOIN dept;
```

```
EMPNO ENAME  DEPTNO LOC
----- ------ ------ ---------
 7839 KING       10 NEW YORK
 7698 BLAKE      30 CHICAGO
 7782 CLARK      10 NEW YORK
 7566 JONES      20 DALLAS
...
14 rows selected.
```

- Use o NATURAL INNER JOIN (Junção Natural) ou NATURAL JOIN quando os atributos da junção tiverem nomes iguais

37

3. Composição de Relações
3.3 Resultado da Junção Natural com Using

```
SQL>SELECT  emp.empno, emp.ename, deptno, dept.loc
  2 FROM    emp INNER JOIN dept USING(deptno);
```

```
EMPNO ENAME  DEPTNO LOC
----- ------ ------ ---------
 7839 KING       10 NEW YORK
 7698 BLAKE      30 CHICAGO
 7782 CLARK      10 NEW YORK
 7566 JONES      20 DALLAS
...
14 rows selected.
```

- Use a cláusula USING para especificar que atributos devem ser usados na junção

38

3. Composição de Relações
3.4 Ambigüidade

- Use prefixos para distinguir os nomes das colunas quando utilizar múltiplas tabelas.
- Utilize-se de alias quando existirem colunas com mesmo nome em mais de uma tabela.
- Utilize o operador rename (as)

39

3. Composição de Relações
3.5 Alias de Tabela

- Objetivo: Simplificar as consultas com produto cartesiano

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2         dept.deptno, dept.loc
  3  FROM   emp, dept
  4  WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e, dept d
  4  WHERE  e.deptno=d.deptno;
```

40

3. Composição de Relações
3.5 Alias de Tabela

- Objetivo: Simplificar as consultas com join

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2         dept.deptno, dept.loc
  3  FROM   emp INNER JOIN dept ON
                    emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e INNER JOIN dept d ON
                    e.deptno=d.deptno;
```

41

3. Composição de Relações
3.6 Junção de mais de duas Tabelas

CUSTOMER

```
NAME           CUSTID
-----------    ------
JOCKSPORTS        100
TKB SPORT SHOP    101
VOLLYRITE         102
JUST TENNIS       103
K+T SPORTS        105
SHAPE UP          106
WOMENS SPORTS     107
...               ...
9 rows selected.
```

ORD

```
 CUSTID  ORDID
 ------- -------
    101     610
    102     611
    104     612
    106     601
    102     602
    106
    106
...     ...
21 rows
```

ITEM

```
 ORDID  ITEMID
------- -------
    610       3
    611       1
    612       1
    601       1
    602       1
...
64 rows selected.
```

42

3. Composição de Relações
3.6 Junção de mais de duas Tabelas

```
SQL>  SELECT customer.name, customer.custid,
             ord.custid, ord.ordid,
             item.ordid, item.itemid
  2  FROM   customer, ord, item
  3  WHERE  customer.custid = ord.custid
  4           and ord.ordid = item.ordid;
```

ou

```
SQL>SELECT customer.name, customer.custid,
           ord.custid, ord.ordid,
           item.ordid, item.itemid
  2 FROM (customer INNER JOIN ord
  3         ON customer.custid = ord.custid)
  4                INNER JOIN item
  5                      ON ord.ordid = item.ordid;
```

43

3. Composição de Relações
3.6 Junção de mais de duas Tabelas

```
SQL>SELECT customer.name, custid, ordid ,item.itemid
  2 FROM (customer NATURAL INNER JOIN ord)
  3               NATURAL INNER JOIN item;
```

- Os atributos utilizados na junção natural podem se especificados somente uma vez na claúsula SELECT(custid e ordid).

44

3. Composição de Relações
3.7 Não Junção Interna

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950

...
14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"O salário na tabela EMP está entre o menor e o maior salário da tabela SALGRADE"

45

3. Composição de Relações
3.7 Resultado Não Junção Interna

```
SQL>  SELECT   e.ename, e.sal, s.grade
  2   FROM     emp e, salgrade s
  3   WHERE    e.sal
  4   BETWEEN  s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- ---------- ---------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

46

3. Composição de Relações
3.7 Resultado Não Junção Interna

```
SQL>  SELECT  e.ename, e.sal, s.grade
   2   FROM     emp e INNER JOIN salgrade s ON e.sal
                      BETWEEN s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- ---------- ---------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

- A diferença da junção interna tradicional é a não utilização do operador de igualdade.

47

3. Composição de Relações
3.8 Junção Externa

- **Junção externa é quando tuplas são incluídas no resultado sem que exista uma tupla correspondente na outra relação.**

EMP

ENAME	DEPTNO
KING	10
BLAKE	30
CLARK	10
JONES	20
...	

DEPT

DEPTNO	DNAME
10	ACCOUNTING
30	SALES
10	ACCOUNTING
20	RESEARCH
...	
40	OPERATIONS

Nenhum empregado no departamento OPERATIONS

48

3. Composição de Relações
3.8 Junção Externa

- Utiliza-se junção externa para listar tuplas que não usualmente se reúnem numa condição join.
- O operador de junção externa é o sinal de adição (+) ou OUTER JOIN tanto para a esquerda(LEFT) como a direita(RIGHT)

```
SELECT R1.A1, R2.A2
FROM   R1, R2
WHERE  R1.A1(+) = R2.A1;
```

```
SELECT R1.A1, R2.A2
FROM   R1, R2
WHERE  R1.A1 = R2.A2(+);
```

49

3. Composição de Relações
3.8 Junção Externa

```
SELECT R1.A1, R2.A2
FROM   R1 RIGHT OUTER JOIN R2 ON
           R1.A1= R2.A2;
```

```
SELECT R1.A1, R2.A2
FROM   R1 LEFT OUTER JOIN R2 ON
           R1.A1 = R2.A2;
```

- RIGHT OUTER JOIN = Junção Externa a Direita
- LEFT OUTER JOIN = Junção Externa a Esquerda
- FULL OUTER JOIN = Junção Externa Total
  - Junção Externa a Esquerda + Junção Externa a Direita

Recomendável pela Oracle usar OUTER do que (+)

50

3. Composição de Relações
3.9 Utilização da Junção Externa

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM     emp e, dept d
  3  WHERE    e.deptno(+) = d.deptno
  4  ORDER BY e.deptno;
```

```
ENAME         DEPTNO DNAME
---------- --------- --------------
KING              10 ACCOUNTING
CLARK             10 ACCOUNTING
...
                  40 OPERATIONS
15 rows selected.
```

51

3. Composição de Relações
3.9 Utilização da Junção Externa

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM     emp e RIGHT OUTER JOIN dept d ON
                   e.deptno = d.deptno
  3  ORDER BY e.deptno;
```

```
ENAME         DEPTNO DNAME
---------- --------- --------------
KING              10 ACCOUNTING
CLARK             10 ACCOUNTING
...
                  40 OPERATIONS
15 rows selected.
```

52

3. Composição de Relações
3.10 Auto Junção

EMP (EMPREGADO)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (GERENTE)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE

"MGR na tabela Empregado é idêntica a EMPNO na tabela Gerente"

53

3. Composição de Relações
3.10 Auto Junção

```
SQL> SELECT trab.ename||' trabalha para '||ger.ename
  2  FROM   emp trab, emp ger
  3  WHERE  trab.mgr = ger.empno;
```

```
TRAB.ENAME||TRABALHA PARA||GER
------------------------------
BLAKE trabalha para KING
CLARK trabalha para KING
JONES trabalha para KING
MARTIN trabalha para BLAKE
...
13 rows selected.
```

|| realiza a concatenação de Literais.

54

3. Composição de Relações
3.10 Auto Junção

```
SQL> SELECT trab.ename||' trabalha para '||ger.ename
  2  FROM   emp trab INNER JOIN emp ger
                ON trab.mgr = ger.empno;

TRAB.ENAME||TRABALHA PARA||GER
------------------------------
BLAKE trabalha para KING
CLARK trabalha para KING
JONES trabalha para KING
MARTIN trabalha para BLAKE
...
13 rows selected.
```

55

3. Composição de Relações

- Resolver os exercícios de DML Join(ExercicioDML_Join.pdf)

56

4. Modificações no BD
4.1. Inserção

- Para inserir dados em relação podemos especificar uma tupla a ser inserida ou escrever uma consulta cujo resultado é um conjunto de tuplas a inserir. A inserção é expressa por:

```
INSERT INTO r(A1, A2,..., AN)
VALUES (V1, V2,...,VN);
```

- Em que **r** é a relação **Ai** representa os atributos as serem inseridos e **Vi** os valores contidos nos atríbutos **Ai**.

57

4. Modificações no BD
4.2. Alteração

- Em determinadas situações, podemos querer modificar valores das tuplas sem, no entanto alterar todos os valores. Para esse fim, o comando update pode ser usado.

```
UPDATE r
SET A1 = V1, A2 = V2, ... NA = VN
WHERE p;
```

- em r é a relação, **Ai** representa o atributo a ser atualizado e **Vi** o valor a ser atualizado no atributo **Ai**, e **P** um predicado. A cláusula where pode ser omitida nos casos de atualização de todas as de r.

58

4. Modificações no BD
4.3. Remoção

- A remoção de dados é expresso muitas vezes do mesmo modo que uma consulta.
- Podemos remover somente tuplas inteiras; não podemos excluir valores de um atributo em particular.
- Em SQL, a remoção é expressa por:

```
DELETE FROM r
WHERE P;
```

  - P representa um predicado
  - r uma relação.
- O comando delete encontra primeiro todas as tuplas **t** em r para as quais **P(t)** é verdadeira e então removê-las de **r**.
- A cláusula where pode ser omitida nos casos de remoção de todas as de r.

59

5. Transações

- Transação é uma unidade **atômica** de trabalho que atua sobre um banco de dados.
- Uma transação pode ser constituída por **uma** ou **mais** operações de acesso à base de dados.
- Todas as operações devem ser bem-sucedidas, caso contrário os efeitos da transação devem ser **revertidos**.

60

5. Transações

- Inicia com o **primeiro** comando SQL emitido para a base de dados
- Finaliza com os seguintes eventos:
  - COMMIT ou ROLLBACK
  - Comandos DDL executam commit automático
  - Saída do Usuário
  - Queda do Sistema

61

5. Transações
5.1. COMMIT

- Uma transação bem-sucedida termina quando um comando **COMMIT** é executado.
- O comando **COMMIT** finaliza e efetiva todas as alterações feitas na base de dados durante a transação.
- **Grava** uma transação no banco de dados.
- Sintaxe:

```
COMMIT;
```

62

5. Transações
5.1. COMMIT

Alteração de Dados

```
SQL> UPDATE   funcionario
  2  SET      codigo_departamento = 3
  3  WHERE    rg_funcionario = '1112';
1 linha atualizada.
```

Commit nos Dados

```
SQL> COMMIT;
Commit concluído.
```

63

5. Transações
5.2. ROLLBACK

- Restaura uma transação.
- Recupera o banco de dados para a posição que esta após o último comando **COMMIT** ser executado.
- Sintaxe:

```
ROLLBACK;
```

64

5. Transações
5.2. ROLLBACK

- Descarta todas as mudanças da transação.
  - Os valores anteriores são recuperados.
  - Os bloqueios são desfeitos.

```
SQL> DELETE FROM funcionario;
14 linhas deletadas.

SQL> ROLLBACK;
Rollback concluído.
```

65

Conclusão

- Não precisa nem dizer que Banco de dados são essenciais no desenvolvimento de sistemas.
- Conhecer somente banco de dados ou programação não ajudar para criar bons sistemas.
- Pois o programa depende do banco de dados e o banco de dados depende do programa.
- Se um ou outro for mal projetado todos os dois terão problemas.

66

Bibliografia

- **Principal**
  - CHEN, P. **The Entity-Relationship Model - Toward a Unified View of Data**. TODS, 1976.
  - DATE, C. J. **Introdução a Sistemas de Banco de Dados.** 7. ed. Rio de Janeiro: Campus, 2000. 803 p.
  - ELMASRI, S. N.; NAVATHE, B.S.. **Sistemas de Banco de Dados:** Fundamentos e Aplicações. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 2002. 837 p.
  - SILBERSCHATZ, A. ; KORTH, H.F. ; SUDARSHAN, S. **Sistema de Banco de Dados.** 5. ed. Rio de Janeiro: Elsevier, 2006.
- **Complementar**
  - BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML.** 1. ed. Rio de Janeiro: Campus, 2002. 286 p.
  - CHIAVENATO, I. **Introdução à Teoria Geral da Administração.** 7. ed. Rio de Janeiro: Campus, 2004. 634 p.
  - DAUM, B. **Modelagem de Objetos de negócio com XML:** abordagem com base em XML Schema. Rio de Janeiro: Elsevier, 2004.
  - KROENKE, D.M. **Banco de Dados:** Fundamentos, Projeto e Implementações. 6. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1998. 382 p.
  - OZSU, M.T.; VALDURIEZ, P. **Princípios de sistemas de banco de dados distribuídos.** 1. ed. Rio de Janeiro: Campus, 2001.
  - YOURDON, E. **Análise Estrutura Moderna.** 3. ed. Rio de Janeiro: Campus, 1992. 836 p.

Fim