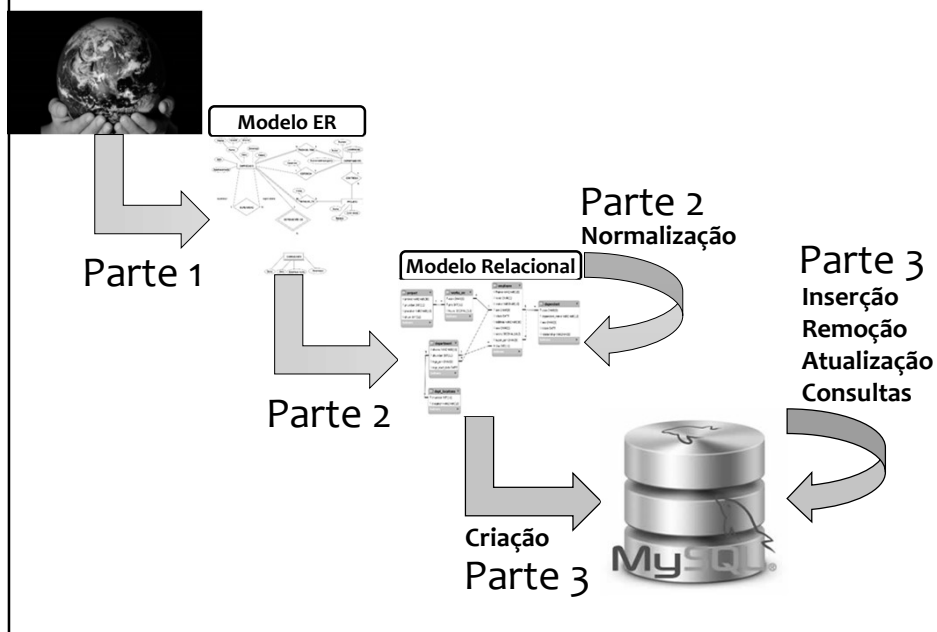


Banco de Dados – Parte 03

SQL
Programação com SQL
Transações
Ferramentas Administrativas

1

Até agora...



2

Estrutura da Aula

- SQL
- Técnicas de Programação SQL
- Transações
- Ferramentas Administrativas
- Laboratório:
 - SQL
- Projeto
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas

3

SQL



QUE
CURIO
SIDA
DES

- O que significa?
 - Structured Query Language
- Projetada e implementada pela IBM research (Anos 70)
 - Prova de conceito de implementação do modelo relacional
 - Inicialmente SEQUEL (“*Structured English Query Language*”)
 - "síquel" ao invés de "és-kiú-él"
 - Porém, em português "ésse-quê-éle".

4

SQL

- Versões padronizadas pela ANSI e ISO
 - SQL-86 (ANSI) ou SQL-87 (ISO)
 - SQL-89
 - SQL-92 ou SQL2
 - SQL:1999 ou SQL3
 - SQL:2003
 - SQL:2006
 - SQL:2008: *SML Query Language*
 - SQL:2011
 - SQL:2016
 - SQL:2019

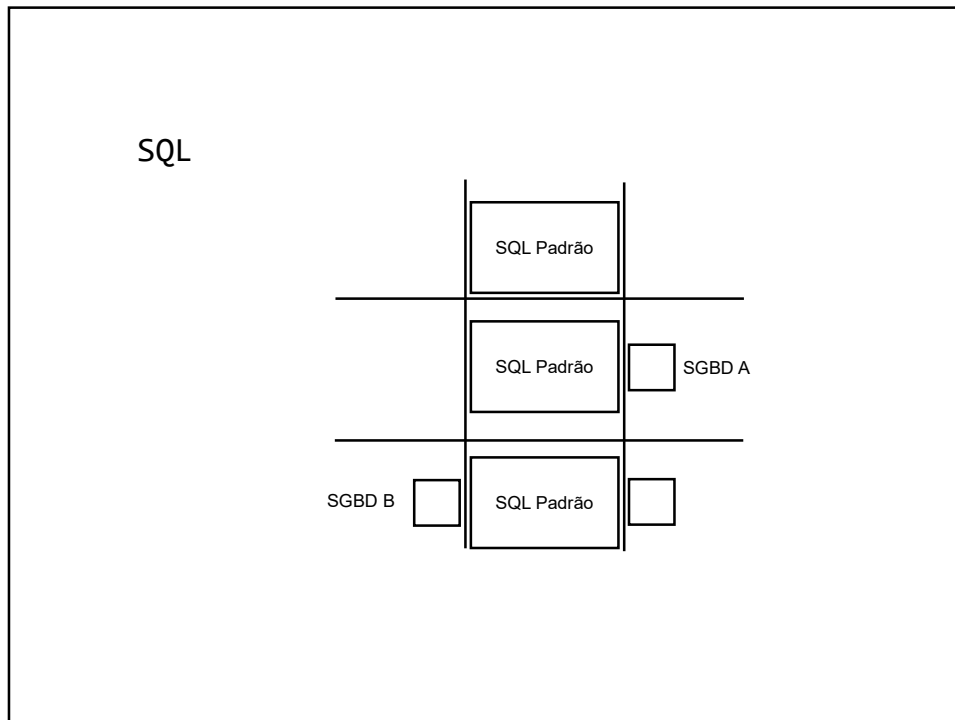
Fonte: <https://en.wikipedia.org/wiki/SQL>

5

SQL

- Uma das principais razões do sucesso do modelo relacional
- Possui sintaxe amigável
- Tornou-se padrão
 - Migração dos modelos hierárquicos e de rede foi estimulada
 - Migração entre SGBDs
- Na prática...

6



7

SQL

- Vantagens de manter-se no padrão
 - Fazer migrações fáceis de SGBD
 - Uma mesma aplicação pode acessar mais de um BD em SGBDs diferentes
- Vantagens de usar linguagem especializada
 - Simplicidade em algumas operações
 - Otimização

8

A Linguagem

- DDL: Definições de dados
- DML: Consultas e atualizações
- VDL: Definição de visões
- Outros
 - Segurança
 - Autorizações de acesso
 - Restrições de integridade
 - Controle de transações

9

A Linguagem

- Linguagem central (core language)
 - Suportado pela maioria dos SGBDs
- Pacotes especializados opcionais
 - Mineração de dados, dados temporais, dados multimídia, etc
 - Adquiridos independentemente

10

Mas, para quê SQL?

“Aprender um framework de MOR é muito importante pois facilita principalmente o trabalho de CRUD e consultas básicas (filtros simples, JOINS, etc).

No entanto, há uma série de situações onde é preciso usar SQL diretamente, sendo a principal delas para relatórios que envolvam funções de agregação e agrupamento. É quase regra que quando se trata de um relatório de cálculo ou já com um pouco de complexidade usar SQL direto é bem mais indicado.

Já tive casos aqui de termos profissionais que trabalharam somente com frameworks MOR e não sabiam SQL. Qualquer consulta um pouquinho mais complicada ele tentava usar o framework e incorriam em um código com diversos $N + 1$ Select, anti-performático e difícil de entender. Este motivo, inclusive, resultou em seu desligamento por essa insuficiência técnica que ele não conseguiu sanar. Ou seja, é requisito básico :)”



Gleydson Lima
Sócio-Fundador ESig

11

Mas, para quê SQL?

- Melhoria de desempenho de consultas
- Tornar-se um DBA
- Business intelligence: extração dos dados e importação em Pentaho, Qlickview, etc será feita usando SQL



Prof. Itamir Filho
Diretor de Tecnologia da Informação - IMD

12

Nesta Aula

- Criação de esquemas e tabelas
- Tipos de dados básicos
- Restrições básicas
 - Integridade referencial e de chave
- Modificações de esquemas, tabelas, e restrições
- Consultas básicas
- Consultas complexas
 - Funções agregadas e agrupamento

13

Nesta Aula

- Comandos de inserção, remoção e atualização de dados
- Restrições mais gerais sobre o banco (Asserções)
- Triggers
- Visões
- Resumo de outras características de SQL



MySQL 8.0 Reference Manual
Including MySQL NDB Cluster 8.0

1 / 6128

14

Definição de Dados e Tipos de Dados

- Termos
 - Relação → TABLE
 - Tupla → TUPLE
 - Atributo → COLUMN
- Comando CREATE
 - Esquemas
 - Tabelas
 - Domínios
 - Visões
 - Asserções
 - Triggers

15

Esquemas x Catálogo

- Esquema
 - Identificado por nome
 - Inclui identificador de autorização para indicar o usuário que possui o esquema
 - Descrições dos **elementos**
 - Tabelas
 - Restrições
 - Visões
 - Domínios
 - Outros construtores (autorizações)

16

Esquemas x Catálogo

•Catálogo

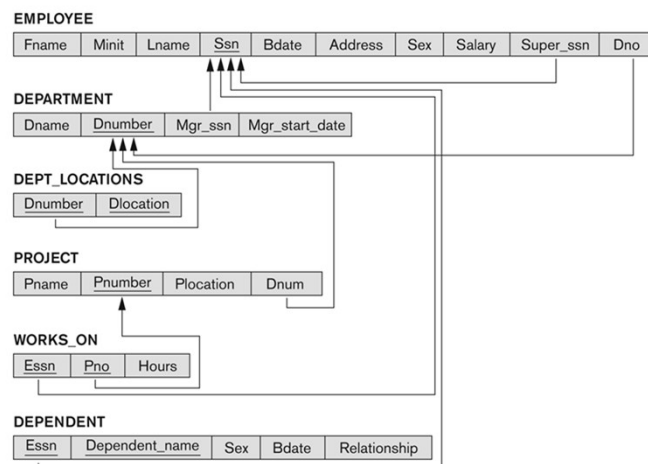
- Uma coleção de esquemas em um ambiente SQL
- Sempre possui um esquema especial INFORMATION_SCHEMA
 - Informações de todos os esquemas do catálogo e suas descrições
- Restrições de integridade apenas entre relações de esquemas do mesmo catálogo
- Esquemas do mesmo catálogo podem compartilhar alguns elementos como definição de domínio

17

Nosso Exemplo

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



18

Criando Esquemas

(create_db.sql)

- Criação

CREATE DATABASE COMPANY;

- Exibição

SHOW DATABASES;

- Remoção

DROP DATABASE COMPANY;

- Seleção

USE COMPANY;

19

Criando Tabelas

(create_tables_1.sql)

- CREATE TABLE

- Cria uma nova tabela

- Nome

- Atributos

- Nome

- Tipo

- Restrições do atributo (NOT NULL)

- Restrições iniciais

- Chaves

- Chaves estrangeiras

- Restrições de integridade

- CREATE TABLE ou ALTER TABLE

20

Comandos em Geral

```
CREATE TABLE COMPANY.EMPLOYEE
...
X
CREATE TABLE EMPLOYEE...
```

21

Criando Tabelas

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	-----	-------	---------	-----	--------	-----------	-----

```
CREATE TABLE employee(
  fname    VARCHAR(15)    NOT NULL,
  minit    CHAR,
  lname    VARCHAR(15)    NOT NULL,
  ssn      CHAR(9)        NOT NULL,
  bdate    DATE,
  address  VARCHAR(30),
  sex      CHAR,
  salary   DECIMAL(10,2),
  super_ssn CHAR(9),
  dno      INT            NOT NULL,
  PRIMARY KEY(ssn),
  FOREIGN KEY(super_ssn) REFERENCES employee(ssn)
);
```

22

Criando Tabelas

- Outras tabelas

- department
- dept_locations
- project
- works_on
- Dependent

- Ainda falta uma chave estrangeira

- dno de EMPLOYEE

23

Tipos de Dados

- Numéricos

- INTEGER, INT [4 bytes]
- SMALLINT [2 bytes]
- FLOAT [4 bytes]
- REAL [8 bytes]
- DECIMAL (i, j) [Varia]
 - i – precisão: total de números decimais
 - j – escala: número de dígitos depois do ponto
- Vários outros

24

Tipos de Dados

•Caracter/String

- CHAR (n) : tamanho fixo de n
 - Preenchimento com branco à esquerda
 - Brancos geralmente ignorados em comparações
- VARCHAR (n) : tamanho variável até n
- Concatenação: 'banco' || ' de dados'

25

Tipos de Dados

•Bit-string

- BIT (n) : tamanho fixo de n
 - B'0101'
- BIT VARYING (n) : tamanho variável até n
- BLOB : Objetos binários grandes

26

Tipos de Dados

- **BOOLEAN**
 - `true`
 - `false`
 - **NULL** (UNKNOWN – 3-Values Logic)

27

Tipos de Dados

- **TEMPO**
 - **DATE**: YYYY-MM-DD
 - `DATE '2008-10-16'`
 - **TIME**: HH:MM:SS
 - `TIME '18:57:54'`
 - **TIME (i)**
 - Tempo com mais precisões
 - **TIME WITH TIME ZONE**
 - 6 posições adicionais: +13:00 a -12:59

28

Tipos de Dados

•TEMPO

`TIMESTAMP`

`'2008-10-16 18:57:54 423443'`

•Domínios

`CREATE DOMAIN SSN_TYPE as CHAR(9);`

Não disponível em várias implementações como MySQL ;-)

29

Restrições

- Integridade referencial e de chave
- Domínios de atributos
- NULLs
- Restrições sobre tuplas individuais

30

Restrições

- NOT NULL
- DEFAULT <valor>

```
(create_tables_2.sql)
CREATE TABLE employee(
...
    dno INT NOT NULL DEFAULT 1,
...
);
```

31

Restrições

- CHECK <condição>

```
CREATE TABLE department(
...
    dnumber INT
        NOT NULL
        CHECK (dnumber > 0 AND dnumber < 21),
...
);
```

32

Restrições

- PRIMARY KEY (<atributo>)

```
CREATE TABLE employee(
...
    CONSTRAINT EMPPK
        PRIMARY KEY(ssn),
...
);
```

33

Restrições

- FOREIGN KEY (<atributo>)
REFERENCES <tabela>(<atributo>)

```
CREATE TABLE employee(
...
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY(super_ssn)
            REFERENCES employee(ssn)
...
);
```

34

Restrições

- `UNIQUE (<atributo>)`
 - Chaves secundárias

```
CREATE TABLE department (
...
    CONSTRAINT DEPTSK UNIQUE (dname) ,
...
);
```

35

Restrições

- Comportamento default à violações:
`reject`
- Qualificações
 - `ON DELETE` ou `ON UPDATE`
- Opções
 - `SET NULL`
 - `CASCADE`
 - `SET DEFAULT`

36

Restrições

```
CREATE TABLE employee (
    ...
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (super_ssn) REFERENCES
            employee (ssn)
            ON DELETE SET NULL
            ON UPDATE CASCADE
);
```

37

Restrições

```
ON UPDATE CASCADE
ON UPDATE SET NULL
```

“If ON UPDATE CASCADE or ON UPDATE SET NULL recurses to update the same table it has previously updated during the cascade, it acts like RESTRICT.

This means that you cannot use self-referential ON UPDATE CASCADE or ON UPDATE SET NULL operations.

This is to prevent infinite loops resulting from cascaded updates. A self-referential ON DELETE SET NULL, on the other hand, is possible, as is a self-referential ON DELETE CASCADE. Cascading operations may not be nested more than 15 levels deep. “

FONTE: <http://dev.mysql.com/doc/refman/5.0/en/innodb-foreign-key-constraints.html>

38

Restrições

- Outros CHECK
 - Departamento com dept_create_date

```
CREATE TABLE department(  
...  
CONSTRAINT DATE CHECK  
(dept_create_date <= mgr_start_date)  
);
```

39

Mudanças de Esquema

- DROP
 - Remoção de elementos de esquema
 - Tabelas
 - Domínios
 - Restrições
 - Remoção de esquemas
- Duas opções de comportamento (não em MySQL)
 - CASCADE
 - RESTRICT

40

Mudanças de Esquema

(change_tables.sql)

```
DROP TABLE dependent CASCADE;
```

- **CASCADE:** Todas as restrições e visões que referenciam a tabela removida são removidas do esquema
- **RESTRICT:** Só funciona se tabela não for referenciada no esquema
- **DROP TABLE** remove toda a tabela. Para remover tuplas veremos **DELETE** mais adiante

41

Mudanças de Esquema



42

Mudanças de Esquema

- ALTER
 - Alteração de elementos de esquema
 - Tabelas
 - Adição/Remoção de colunas
 - Mudança de definição de coluna
 - Adição/Remoção de restrições

43

Mudanças de Esquema

```
ALTER TABLE employee  
ADD COLUMN job VARCHAR(12);
```

- Para tuplas já existentes
 - DEFAULT
 - NULL caso contrário

44

Mudanças de Esquema

```
ALTER TABLE employee  
DROP COLUMN address CASCADE;
```

- **CASCADE:** Todas as restrições e visões que referenciam a coluna removida são removidas do esquema
- **RESTRICT:** Só funciona se coluna não for referenciada no esquema

45

Mudanças de Esquemas

```
ALTER TABLE department  
  ALTER COLUMN mgr_ssn  
  DROP DEFAULT;  
ALTER TABLE department  
  ALTER COLUMN mgr_ssn  
  SET DEFAULT '33344555';
```

46

Mudanças de Esquema

```
ALTER TABLE employee  
  ADD CONSTRAINT  
    FOREIGN KEY (dno)  
  REFERENCES  DEPARTMENT (dnumber) ;
```

47

Consultas Básicas

- Criando e Populando o BD

- (create_db_and_populate.sql)

- Modelo

- company_model.docx

- Dados

- company_data.xlsx

48

Consultas Básicas

- `SELECT`
 - Sem relação com o select da álgebra relacional
- Diferenças entre SQL e modelo relacional
 - Tuplas podem ter mesmos valores
 - Tabela é um *bag* de tuplas
 - Restrições para conjuntos são feitas através de chaves e/ou cláusula `DISTINCT`

49

`SELECT-FROM-WHERE`

`SELECT <lista de atributos>`

`FROM <lista de tabelas>`

`WHERE <condição>`

• Comparadores básicos

- `=, <, <=, >, >=, e <>`

`(basic_queries.sql)`

50

SELECT-FROM-WHERE

- Q0: recupere a data de nascimento e endereço do empregado 'John B. Smith'

```
SELECT BDATE, ADDRESS
FROM EMPLOYEE
WHERE FNAME='John'
      AND MINIT='B'
      AND LNAME='Smith';
```

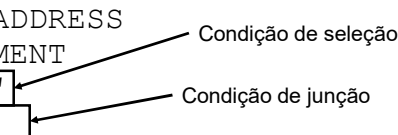
- Resultado SQL pode ter tuplas repetidas

51

SELECT-FROM-WHERE

- Q1: Recupere o nome e endereço de todos os empregados do departamento 'Research'

```
SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research'
      AND DNUMBER=DNO;
```



- Similar à sequência de operações SELECT-PROJECT-JOIN da álgebra relacional
- Mais de uma condição de junção...

52

SELECT-FROM-WHERE

- Q2: para todo projeto localizado em 'Stafford', liste o número do projeto, o número do departamento que o controla, e o último nome, endereço e data de nascimento do gerente do departamento

```
SELECT PNUMBER, DNUM, LNAME,
       BDATE, ADDRESS
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER
       AND MGR_SSN=SSN
       AND PLOCATION='Stafford';
```

Relaciona o projeto com o departamento que o controla

Relaciona o departamento com o seu gerente

53

Nomes de atributos ambíguos, codinome e variáveis de tupla

•Acontece quando

- temos mesmo nome de atributo para relações diferentes
 - SELECT NAME FROM...
 - Qualificação do atributos
 - EMPLOYEE.fname
- Temos duas referências (ou mais) para a mesma relação
 - Codinomes

54

Nomes de atributos ambíguos, codinome e variáveis de tupla

- Q8: Para cada empregado, recupera o nome do empregado e o nome de seu supervisor

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.SUPER_SSN=S.SSN;
```

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN=S.SSN;
```

55

Nomes de atributos ambíguos, codinome e variáveis de tupla

- A prática do uso de codinomes para as relações deveria ser adotada independente da existência de ambiguidade

Q1A:

```
SELECT E.FNAME, E.LNAME, E.ADDRESS
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNAME='Research'
      AND E.DNUMBER=D.DNO;
```

56

Nomes de atributos ambíguos, codinome e variáveis de tupla

- Codinomes para atributos

```
EMPLOYEE AS
```

```
E(Fn, Mi, Ln, Ssn, Bd, Sex, Sal, Sssn, Dno)
```

57

Cláusula WHERE e *

- WHERE true pode ser omitida
- Se mais de uma relação faz parte do SELECT temos um produto cartesiano
- Q9: Recupere os SSN dos empregados
- Q10: Todas as combinações de SSN com nome de departamento

```
SELECT SSN FROM EMPLOYEE;
```

```
SELECT SSN, DNAME
```

```
FROM EMPLOYEE, DEPARTMENT;
```

58

Cláusula WHERE e *

- Recuperação de todos os atributos
 - *
- Q1C: Recupere todos os atributos dos empregados do departamento 5

```
SELECT * FROM EMPLOYEE WHERE DNO=5;
```

- Q10: Todas os atributos do empregados do departamento 'Research'

```
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE dname='Research'
      AND dno=dnumber;
```

59

Cláusula WHERE e *

- Cuidado: o esquecimento de condição de junção e seleção pode gerar relações incorretas

Q10A:

```
SELECT *
FROM EMPLOYEE, DEPARTMENT;
```

60

Tabelas como Conjuntos

- SQL não elimina tuplas repetidas
 - Alto custo: ordenação e eliminação
 - Duplicatas podem ser desejadas
- Eliminação explícita de tuplas
 - DISTINCT
- Q11: Retorne os salários dos empregados

```
SELECT ALL SALARY FROM EMPLOYEE;  
SELECT DISTINCT SALARY  
FROM EMPLOYEE;
```

61

Tabelas como Conjuntos

- Operações sobre conjuntos
 - UNION, EXCEPT, INTERSECT
- Resultados são conjuntos
- Aplicáveis apenas a relações união-compatíveis
 - Mesmos atributos
 - Mesma ordem de atributos

62

Tabelas como Conjuntos

- Q4: Lista de todos os números dos projetos que envolvem um empregado cujo último nome é 'Smith' como trabalhador ou gerente do departamento que controla o projeto

```
(SELECT PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGR_SSN=SSN AND LNAME='Smith')
UNION
(SELECT PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith');
```

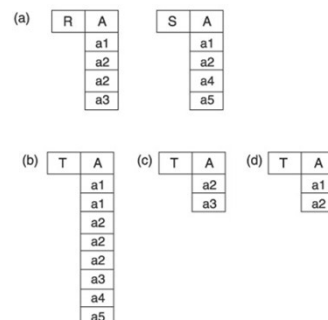
'Smith' como funcionário

'Smith' como gerente

63

Tabelas como Conjuntos

- Operações sobre bags
 - UNION ALL, EXCEPT ALL, INTERSECT ALL
- Resultados são bags



64

Casamento de Padrões

- LIKE

- `_` : Exatamente um caracter
- `%` : Qualquer número de caracteres
- `'AB_CD\%EF'` `ESCAPE '\'` = `'AB_CD%EF'`
- `"` representa um `'` como caracter

- Q12: Todos os empregados que moram em Houston, TX.

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE address LIKE '%Houston,%TX%';
```

65

Casamento de Padrões

- Q12A: Todos os empregados nascidos nos anos 50.

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE bdate LIKE
'__5_____';
```

66

Aritmética em Consultas

- +, -, *, e /
- Q13: Mostre os salários resultantes se todo empregado que trabalha no projeto 'ProductX' receber 10% de aumento

```
SELECT fname, lname, salary AS old_salary,
       1.1*salary AS increased_salary
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE ssn=essn AND pno=pnumber
      AND pname='ProductX';
```

67

Outras operações em Consultas

- Concatenação de string: ||
- Incremento (+) e decremento (-)
 - DATE, TIME, TIMESTAMP, INTERVAL
- BETWEEN
 - Q14: Recupere os empregados do departamento 5 com salário entre 30000 e 40000.

```
SELECT fname, lname FROM EMPLOYEE
WHERE
  (salary BETWEEN 30000 AND 40000)
  AND dno=5;
```

68

Ordenação de Resultados

- ORDER BY
- ASC e DESC (padrão é ascendente)
- Q15: lista dos empregados e dos projetos que eles trabalham ordenados por departamento, e dentro do departamento, ordenado por último nome, primeiro nome

```
SELECT dname, lname, fname, pname
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE dnumber=dno AND ssn=essn AND pno=pnumber
ORDER BY dname, lname, fname;
```

69

Ordenação de Resultados

- ORDER BY
- ASC e DESC (padrão é ascendente)
- Q15: lista dos empregados e dos projetos que eles trabalham ordenados por departamento, e dentro do departamento, ordenado por último nome, primeiro nome

```
SELECT dname, lname, fname, pname
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE dnumber=dno AND ssn=essn AND pno=pnumber
ORDER BY dname DESC, lname ASC, fname ASC;
```

70

Comparações com NULL e lógica de três valores

- Valores NULL
 - Valor desconhecido
 - Valor indisponível
 - Atributo não se aplica
- SQL não faz distinção
- Em geral, dois valores NULL são diferentes entre si
- Quando NULL está envolvido em uma operação de comparação o resultado é desconhecido
- SQL usa lógica de três valores

71

Operações na Lógica de Três Valores

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN
OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN
NOT			
TRUE		FALSE	
FALSE		TRUE	
UNKNOWN		UNKNOWN	

72

Comparações com NULL e lógica de três valores

- Comparações usando = e <> podem não ser apropriadas visto que valores NULL são diferentes entre si
- Tuplas com valores NULL para os atributos de junção não são incluídas
- IS e IS NOT
- Q18: nome dos empregados sem supervisores

```
SELECT fname, lname  
FROM EMPLOYEE  
WHERE super_ssn IS NULL;
```

73

Consultas Entrelaçadas

- Uma consulta entrelaçada pode ser feita dentro da cláusula WHERE de outra consulta

74

Consultas Entrelaçadas

- Q4: Lista de todos os números dos projetos que envolvem um empregado cujo último nome é 'Smith' como trabalhador ou gerente do departamento que controla o projeto

```
(SELECT PNAME
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGR_SSN=SSN AND LNAME='Smith')
UNION
(SELECT PNAME
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith');
```

75

Consultas Entrelaçadas

- Q4A: Lista de todos os números dos projetos que envolvem um empregado cujo último nome é 'Smith' como trabalhador ou gerente do departamento que controla o projeto

```
SELECT DISTINCT pnumber
FROM PROJECT
WHERE pnumber IN
  (SELECT pnumber
   FROM PROJECT, DEPARTMENT, EMPLOYEE
   WHERE DNUM=DNUMBER AND MGR_SSN=SSN AND LNAME='Smith')
OR pnumber IN
  (SELECT pno
   FROM WORKS_ON, EMPLOYEE
   WHERE ESSN=SSN AND LNAME='Smith');
```

'Smith' como funcionário

'Smith' como gerente

76

Consultas Entrelaçadas

- Tuplas de valores em comparações
- QX1: Recupere os `ssns` de todos os empregados que trabalham a mesma combinação (projeto, horas) em algum projeto que o funcionário '123456789' trabalha

```
SELECT DISTINCT essn
FROM WORKS_ON
WHERE (pno, hours)
IN (SELECT pno, hours
    FROM WORKS_ON
    WHERE essn='123456789');
```

77

Consultas Entrelaçadas

- Outros operadores
 - `v IN S`
 - `v = ANY S`
 - Operadores `>`, `<`, `<=`, `>=`, `<>`
 - `V op ALL S (v > ALL S)`
- QX2: Recupere os nomes dos empregados cujo salário é maior do que os salários de todos os empregados do departamento 5

```
SELECT lname, fname
FROM EMPLOYEE
WHERE salary > ALL (SELECT salary
                    FROM EMPLOYEE
                    WHERE dno='5');
```

78

Consultas Entrelaçadas

- Podemos ter vários níveis de entrelaçamento
- Ambiguidade
 - Referências a atributos sem qualificação se referem a relação declaradas no nível mais interno
 - Q4A

```
SELECT DISTINCT pnumber
FROM PROJECT
WHERE pnumber IN
    (SELECT pnumber
     FROM PROJECT, DEPARTMENT, EMPLOYEE
     WHERE DNUM=DNUMBER AND MGR_SSN=SSN AND LNAME='Smith')
```

79

Consultas Entrelaçadas

- Q16: recupere o nome de cada empregado que tem um dependente com o mesmo primeiro nome e sexo que o empregado

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE E.SSN IN
    (SELECT essn
     FROM DEPENDENT
     WHERE E.ssn = essn
           AND E.FNAME=DEPENDENT_NAME
           AND E.sex=sex);
```

80

Consultas Entrelaçadas Correlatas

- Se uma condição na cláusula `WHERE` de uma consulta entrelaçada referencia um atributo de uma relação declarada na consulta mais externa, as duas consultas são correlatas
- O resultado de uma consulta entrelaçada correlata é diferente para cada tupla (ou combinação de tuplas) da relação mais externa

81

Consultas Entrelaçadas Correlatas

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE E.SSN IN
    (SELECT essn
     FROM DEPENDENT
     WHERE E.ssn = essn
      AND E.FNAME=DEPENDENT_NAME
      AND E.sex=sex);
```

- Para cada tupla de `EMPLOYEE`, calcule a consulta entrelaçada. Se o `ssn` da tupla de empregado estiver no resultado, selecione a tupla

82

Consultas Entrelaçadas Correlatas

- Podemos reescrever algumas consultas entrelaçadas que utilizem = ou IN

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E, DEPENDENT AS D
WHERE E.SSN=D.essn
      AND E.FNAME=D.DEPENDENT_NAME
      AND E.sex=D.sex;
```

83

EXISTS

- Retorna TRUE se existe a resposta de uma consulta entrelaçada correlata não é vazia

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE EXISTS
  (SELECT *
   FROM DEPENDENT
   WHERE E.SSN=essn AND E.FNAME=DEPENDENT_NAME
        AND E.sex=sex);
```

84

NOT EXISTS

- O contrário de EXISTS
- Q6: Recupere os nomes dos funcionários que não têm dependentes

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS
```

```
(SELECT *
FROM DEPENDENT
WHERE SSN=ESSN) ;
```

Seleciona tuplas relacionadas para cada um dos funcionários

85

EXISTS

- Q7: Recupere os nomes dos gerentes que têm pelo menos um dependente

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE
```

```
EXISTS
(SELECT * FROM DEPENDENT WHERE SSN=ESSN)
```

```
AND
```

```
EXISTS
(SELECT * FROM DEPARTMENT WHERE SSN=MGR_SSN) ;
```

Tem pelo menos um dependente

Gerenciam pelo menos um departamento

86

EXISTS

- Q3: Recupere o nome de cada empregado que trabalha em todos os projetos controlados pelo departamento '5'

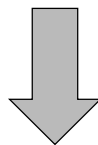
```
SELECT FNAME, LNAME FROM EMPLOYEE
WHERE
    ( (SELECT pno FROM WORKS_ON
      WHERE ssn=essn)
  CONTAINS
    (SELECT pnumber FROM PROJECT
      WHERE dnum='5') );
```

Não é parte de SQL
Precisamos de outra
técnica

87

EXISTS

Empregado **E** trabalha em todos os projetos controlados pelo departamento '5'



$$\forall x. P(x) \Leftrightarrow \neg \exists x. \neg P(x)$$

Não existe um projeto controlado pelo departamento '5' que o funcionário **E** não trabalhe

88

EXISTS

- Q3: Recupere o nome de cada empregado que trabalha em todos os projetos controlados pelo departamento '5'

```
SELECT FNAME, LNAME FROM EMPLOYEE
WHERE NOT EXISTS
```

```
    ( (SELECT * FROM WORKS_ON W1
```

```
        WHERE (W1.pno IN
```

```
            (SELECT pnumber FROM PROJECT
              WHERE dnum=5) )
```

Projeto do
Departamento
'5'

Funcionário não
trabalha em algum
projeto do
departamento '5'

```
        AND
```

```
        NOT EXISTS (SELECT *FROM WORKS_ON W2
                     WHERE W2.essn=ssn AND W1.pno=W2.pno)
```

```
    ) );
```

Não existe um projeto controlado pelo departamento '5' que o funcionário não trabalhe

89

UNIQUE

- UNIQUE (C)

- Retorna TRUE se não existirem tuplas repetidas no resultado da consulta C
- Conjunto ou Bag?

90

Conjuntos Explícitos

Q17: Recupere os SSN dos empregados que trabalham nos projetos 1, 2, ou 3

```
SELECT DISTINCT ESSN  
FROM WORKS_ON  
WHERE PNO IN (1, 2, 3);
```

91

Renomeação

- Q8A: recupere o último nome dos empregados e seus supervisores

```
SELECT  
  E.lname AS Employee_name,  
  S.lname AS Supervisor_name  
FROM EMPLOYEE AS E, EMPLOYEE AS S  
WHERE E.super_ssn=S.ssn;
```

92

Junção de Tabelas

- Podemos definir junção de tabelas na cláusula `FROM`
- É uma relação, mas resultante de uma junção
- Diferentes junções
 - `JOIN`
 - `NATURAL JOIN`
 - `LEFT OUTER JOIN`
 - `RIGHT OUTER JOIN`
 - `CROSS JOIN`
 - Etc
- Facilita o entendimento da consulta ao retirar junções da cláusula `WHERE`

93

Junção de Tabelas

- Q1: Recupere o nome e endereço de todos os empregados do departamento 'Research'

```
SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND DNUMBER=DNO;
```

Q1A:

```
SELECT fname, lname, address
FROM EMPLOYEE JOIN DEPARTMENT ON dno=dnumber
WHERE dname='Research';
```

94

Analisado Eficiência da Consulta (MySQL)

- Q1A Explain
 - EXPLAIN
- Q3 STATS
 - Query Stats
- Execution Plan



95

Junção de Tabelas

- NATURAL JOIN
- Q1: Recupere o nome e endereço de todos os empregados do departamento 'Research'

Q1B:

```
SELECT fname, lname, address
FROM
    EMPLOYEE NATURAL JOIN
    (SELECT dname, dnumber AS dno,
           mgr_ssn AS mssn, mgr_start_date AS msdate
     FROM DEPARTMENT) AS DEPTO
WHERE dname='Research';
```

96

Junção de Tabelas

- NATURAL JOIN
- Caso não existam colunas com mesmo nome, o resultado será um produto cartesiano

```
SELECT dno, dnumber  
FROM EMPLOYEE NATURAL JOIN DEPARTMENT;
```

97

Junção de Tabelas

- Junção padrão é INNER JOIN
 - Apenas tuplas que casam são incluídas
- OUTER JOIN
 - Inclui todas as tuplas

98

Junção de Tabelas

- Q8: Recupere o último e primeiro nome do funcionário e de seu supervisor (NULL se inexistente)

Q8A:

```
SELECT E.lname AS Employee_name,
       S.lname AS Supervisor_name
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.super_ssn=s.ssn;
```

Q8B:

```
SELECT E.lname AS Employee_name,
       S.lname AS Supervisor_name
FROM (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S
      ON E.super_ssn=S.ssn);
```

99

Junção Entrelaçada

- Q2: para todo projeto localizado em 'Stafford', liste o número do projeto, o número do departamento que o controla, e o último nome, endereço e data de nascimento do gerente do departamento

```
SELECT PNUMBER,DNUM,LNAME, BDATE, ADDRESS
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER
      AND MGR_SSN=SSN
      AND PLOCATION='Stafford';
```

100

Junção Entrelaçada

- Q2A: para todo projeto localizado em 'Stafford', liste o número do projeto, o número do departamento que o controla, e o último nome, endereço e data de nascimento do gerente do departamento

```
SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS  
FROM ((PROJECT JOIN DEPARTMENT ON DNUM=DNUMBER)  
      JOIN EMPLOYEE ON MGR_SSN=SSN)  
WHERE PLOCATION='Stafford';
```

101

Funções Agregadas

- COUNT, SUM, MAX, MIN, AVG
- Podem ser usados na cláusula SELECT ou HAVING
- MAX e MIN exigem apenas uma ordem total entre os elementos do tipo

102

Funções Agregadas Exemplos

- Q19: Retorne a soma, o salário máximo, o salário mínimo, e o salário médio do salário de todos os empregados

```
SELECT SUM(salary),  
        MAX(salary),  
        MIN(salary),  
        AVG(salary)  
FROM EMPLOYEE;
```

103

Funções Agregadas Exemplos

- Q20: Q19 mas apenas para os funcionários do departamento 'Research'

```
SELECT SUM(salary), MAX(salary),  
        MIN(salary), AVG(salary)  
FROM (EMPLOYEE JOIN DEPARTMENT  
      ON dno=dnumber)  
WHERE dname='Research';
```

104

Funções Agregadas Exemplos

- Q21 e 22: Número total de empregados da companhia (Q21) e do departamento 'Research' (Q22)

Q21:

```
SELECT COUNT(*) FROM EMPLOYEE;
```

Q22:

```
SELECT COUNT(*)  
FROM EMPLOYEE, DEPARTMENT  
WHERE dno=dnumber AND dname='Research';
```

105

Funções Agregadas Exemplos

- Q23: Conte o número de salários diferentes na empresa

```
SELECT COUNT(DISTINCT salary)  
FROM EMPLOYEE;
```

106

Funções Agregadas Exemplos

- `COUNT(salary)` vs `COUNT(DISTINCT salary)`?
- Em geral, tuplas com salário `NULL` são descartadas e não contadas
- Funções agregadas também podem ser usadas em condições de seleção

107

Funções Agregadas Exemplos

- Q5: Nome de todos os empregados que têm mais de um dependente

```
SELECT lname, fname FROM EMPLOYEE
WHERE
  (SELECT COUNT(*) FROM DEPENDENT
   WHERE ssn=essn) > 1;
```

108

Agrupamento

- Permite a aplicação de funções agregadas a subgrupos de tuplas agrupados de acordo com os valores de alguns atributos
 - Média de salários de cada departamento
- Definição de partições (grupos)
 - GROUP BY
 - Faz parte da cláusula SELECT

109

Agrupamento

- Q24: Para cada departamento, recupere o código do departamento, o número de empregados, e o salário médio deles

```
SELECT dno,  
       COUNT(*) AS num_employee,  
       AVG(salary) as avg_salary  
FROM EMPLOYEE  
GROUP BY dno;
```

110

Agrupamento

PNAME	MINICIAL	LNOME	SSN	• • •	SALARIO	SUPERSSN	DNO
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	• • •	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	null	1

DNO	COUNT (*)	AVG (SALARIO)
5	4	33250
4	3	31000
1	1	55000

Resultado da Q24

111

Agrupamento

- Um grupo novo é criado para tuplas com valor NULL para o atributo de agrupamento
- Podemos ter uma condição de junção junto com um agrupamento

112

Agrupamento

- Q25: Para cada projeto, recupere o número do projeto, o nome do projeto, e o número de empregados que trabalham naquele projeto

```
SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME;
```

- Agrupamento e contagem é feito apenas após a junção

113

Agrupamento

- E se quisermos incluir apenas projetos com mais de dois empregados?
- Cláusula HAVING

Q26:

```
SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT (*) > 2;
```

114

Agrupamento

PNome	PNumero		ESSN	PNO	HORAS
ProdutoX	1	...	123456789	1	32,5
ProdutoX	1		453453453	1	20,0
ProdutoY	2		123456789	2	7,5
ProdutoY	2		453453453	2	20,0
ProdutoY	2		333445555	2	10,0
ProdutoZ	3		666884444	3	40,0
ProdutoZ	3		333445555	3	10,0
Automacao	10		333445555	10	10,0
Automacao	10		999887777	10	10,0
Automacao	10		987987987	10	35,0
Reorganizacao	20	...	333445555	20	10,0
Reorganizacao	20		987654321	20	15,0
Reorganizacao	20		888665555	20	null
NovosBeneficios	30		987987987	30	5,0
NovosBeneficios	30		987654321	30	20,0
NovosBeneficios	30		999887777	30	30,0

Esses grupos não são selecionados por HAVING, condição da Q26

115

Agrupamento Mais exemplos

- Restringindo as tuplas a serem contadas...
- Q27: Para cada projeto, recupere o número do projeto, o nome do projeto, e o número de empregados do departamento '5' que trabalham no projeto

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND SSN=ESSN AND DNO=5
GROUP BY PNUMBER, PNAME;
```

116

Agrupamento

- Permite uma melhor maneira de fazer a divisão da álgebra relacional

T1	A	B
	a1	b1
	a1	b2
	a1	b3
	a2	b1
	a2	b3
	a3	b2
	a3	b3
	a3	b4
	a4	b1

T2	B
	b2
	b3

T3	A
	a1
	a3

pode ser substituído por *
se B não se repete em T1

```
SELECT A
FROM T1
WHERE B IN ( SELECT B FROM T2)
GROUP BY A
HAVING COUNT(DISTINCT B) =
      (SELECT COUNT(*) FROM T2);
```

A Simpler (and Better) SQL Approach to Relational Division. Victor M. Matos. *Journal of Information Systems Education*, Vol. 13(2)

117

EXISTS

- Q3Op: Recupere o nome de cada empregado que trabalha em todos os projetos controlados pelo departamento '5'

```
SELECT fname, lname
FROM (works_on JOIN employee ON essn=ssn)
WHERE pno IN
      (SELECT pnumber FROM project WHERE dnum=5)
GROUP BY essn
HAVING COUNT(DISTINCT pno) =
      (SELECT COUNT(*) FROM project WHERE dnum=5);
```

Neste caso
DISTINCT pno
pode ser substituído por *

118

Agrupamento

- Devemos tomar cuidado quando queremos satisfazer duas condições diferentes (uma na cláusula `SELECT` e outra na `HAVING`)

119

Agrupamento

- Q28: Retorne o número total, em cada departamento, de empregados com o salário maior que 40000, mas apenas para departamentos com mais de 5 empregados

<code>SELECT DNAME, COUNT (*)</code>	Apenas departamentos que têm mais de 5 empregados que ganham mais de 40000 serão selecionados
<code>FROM DEPARTMENT, EMPLOYEE</code>	
<code>WHERE dnumber=dno AND salary>40000</code>	
<code>GROUP BY dname</code>	WHERE é executado antes para selecionar tuplas;
<code>HAVING COUNT (*) > 5;</code>	HAVING é executado depois...

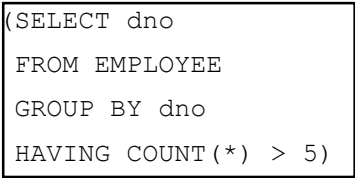
120

Agrupamento

- Q28: Retorne o número total, em cada departamento, de empregados com o salário maior que 40000, mas apenas para departamentos com mais de 5 empregados

```
SELECT DNAME, COUNT(*)
FROM DEPARTMENT, EMPLOYEE
WHERE dnumber=dno AND salary>40000
      AND dno IN (SELECT dno
                  FROM EMPLOYEE
                  GROUP BY dno
                  HAVING COUNT(*) > 5)
GROUP BY dnumber;
```

Departamento com mais de 5 funcionários



121

Resumo de Consultas SQL

•Estrutura geral da consulta

```
SELECT <lista de atributos e funções>
FROM <lista de tabelas>
[ WHERE <condição> ]
[ GROUP BY <atributos de agrupamento> ]
[ HAVING <condição de agrupamento> ]
[ ORDER BY <lista de atributos> ]
```

122

Resumo de Consultas SQL

- Ordem conceitual de valoração da consulta

SELECT <lista de atributos e funções>

FROM <lista de tabelas>

[**WHERE** <condição>]

[**GROUP BY** <atributos de agrupamento>]

[**HAVING** <condição de agrupamento>]

[**ORDER BY** <lista de atributos>]

123

Resumo de Consultas SQL

- Ordem conceitual de valoração da consulta

SELECT <lista de atributos e funções>

FROM <lista de tabelas>

WHERE <condição>

- Para cada combinação de tuplas das tabelas do **FROM**, verifique se a condição do **WHERE** é **TRUE**; neste caso, selecione os atributos do **SELECT** e coloque-os na resposta da consulta
- Cada SGBD implementa isso de maneira mais otimizada (Capítulos 19 e 20)

124

Considerações

- Existem várias maneiras de se fazer uma mesma consulta
 - Ex: Condições de junção na cláusula `WHERE`, ou junção de relação na cláusula `FROM`, ou consultadas entrelaçadas e operador `IN`
- Usuário pode escolher que técnica utilizar
 - Preferível utilizar o menor número possível de entrelaçamento
- Várias possibilidades pode confundir o usuário

125

Considerações

- Existem várias maneiras de se fazer uma mesma consulta
 - Usuário pode utilizar técnica menos eficiente
 - O SGBD deveria tornar isso transparente ao usuário
 - Na prática é bom saber os atalhos do SGBD

126

INSERT

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

- Adiciona tuplas a uma relação
- Especificamos nome da relação e lista de valores da tupla (ordem deve ser respeitada)

(modify_queries.sql)

U1:

```
INSERT INTO EMPLOYEE
```

```
VALUES ('Richard','K','Marini', '653298653',
        '1952-12-30', '98 Oak Forest, Katy, TX',
        'M', 37000, '987654321', '4');
```

127

INSERT

- Especificamos explicitamente que atributos estão na lista de valores passada
 - Valores NULL para os não presentes

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

U1A:

```
INSERT INTO EMPLOYEE(fname,lname,dno,ssn)
```

```
VALUES ('Richard','Marini','4','653298653');
```

128

INSERT

- SGBD deveriam garantir todas as restrições de integridade especificadas na DDL
 - Ex: Integridade referencial
- Alguns SGBD não o fazem por eficiência

U2:

```
INSERT INTO EMPLOYEE (fname, lname, ssn, dno)
VALUES ('Robert', 'Hatcher', '980760540', '2');
```

U2A:

```
INSERT INTO EMPLOYEE (fname, lname, dno)
VALUES ('Robert', 'Hatcher', '5'); (ssn=NULL)
```

129

INSERT

- Podemos inserir várias tuplas ao mesmo tempo em uma tabela
- U3A e U3B: Tabela contendo nome, número de empregados e salário total dos empregados de cada departamento

```
CREATE TABLE DEPTS_INFO
(dept_name VARCHAR(15), no_of_emps INTEGER,
total_sal INTEGER);

INSERT INTO DEPTS_INFO(dept_name, no_of_emps, total_sal)
(SELECT dname, COUNT(*), SUM(salary)
FROM (DEPARTMENT JOIN EMPLOYEE ON dnumber=dno)
GROUP BY DNAME);
```

130

INSERT

- Alterações em DEPARTMENT ou EMPLOYEE não alteram DEPTS_INFO
- DEPTS_INFO pode ficar desatualizada
- Visões devem ser utilizadas para que isso aconteça

131

DELETE

- Remove tuplas (zero, uma, ou mais) de uma relação
- Possui uma cláusula WHERE similar ao SELECT
- Tuplas são removidas explicitamente de apenas uma tabela

132

DELETE

- Remoções podem ser propagadas para outras relações de ações de propagação foram definidas nas restrições de integridade referencial da DDL

```
CREATE TABLE employee (
...
  FOREIGN KEY (super_ssn)
  REFERENCES employee (ssn)
  ON DELETE <SET NULL | CASCADE>
  ON UPDATE CASCADE
);
```

133

DELETE

- Sem a cláusula WHERE, removemos todas as tuplas
 - Tabela é mantida no BD
 - DROP TABLE <nome>

U4A: Tenta remover 'Brown', mas ele não existe

```
DELETE FROM EMPLOYEE
WHERE lname='Brown';
```

134

DELETE

U4B: Remove 'John Smith', e consequentemente altera as tabelas DEPENDENT e WORKS_ON

```
DELETE FROM EMPLOYEE  
WHERE SSN='123456789';
```

135

DELETE

U4C: Remove todos os funcionários do departamento de pesquisa, e consequentemente altera as tabelas DEPENDENT, WORKS_ON, DEPARTMENT, e EMPLOYEE

```
DELETE FROM EMPLOYEE  
WHERE dno IN  
    (SELECT DNUMBER  
     FROM DEPARTMENT  
     WHERE DNAME='Research');
```

136

DELETE

U4D: Remove todos os funcionários e consequentemente altera as tabelas `DEPENDENT`, `WORKS_ON`, `DEPARTMENT`, e `EMPLOYEE`

```
DELETE FROM EMPLOYEE;
```

137

UPDATE

- Modifica valores dos atributos de uma ou mais tuplas de uma relação
- Alteram apenas uma relação
- A cláusula `WHERE` seleciona que tuplas devem ter os valores alterados
- Alteração nas chaves primárias podem ser propagadas para as chaves estrangeiras de outras relações
- Uma cláusula adicional, `SET`, estabelece o novo valor dos atributos

138

UPDATE

- U5: Altere a localização e o departamento que controla o projeto '10' para 'Bellaire' e '5'.

```
UPDATE PROJECT
SET PLOCATION = 'Bellaire', DNUM = 5
WHERE PNUMBER=10;
```

139

UPDATE

- Várias tuplas podem ser alteradas ao mesmo tempo
- U6: Aumente o salário dos funcionários do departamento 'Research' em 10%.

```
UPDATE EMPLOYEE
SET salary = salary*1.1
WHERE dno IN
    (SELECT dnumber
     FROM DEPARTMENT
     WHERE dname='Research');
```

140

Asserções

- Restrições gerais podem ser descritas usando asserções

```
CREATE_ASSERTION <nome> CHECK <condição>
```

- Não suportada por MySQL

- Exemplo

- Salário do empregado deve ser menor que o salário do gerente do departamento que 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) ) ;
```

141

Asserções

- CHECK vs CREATE ASSERTION

- CHECK é verificada apenas quando tuplas são inseridas ou alteradas
 - Por eficiência, recomenda-se seu uso se tivermos certeza que a restrição só pode ser quebrada por essas operações

142

Gatilhos

- `CREATE ASSERTION` aborta a operação caso a condição seja violada
- Gatilhos permitem especificar o tipo de ação a tomar quando um evento ocorre e alguma condição é satisfeita
 - Informar usuário da violação
 - Executar procedimentos armazenados
 - Executar atualizações

143

Gatilhos

- Especificamos um evento, uma condição, e uma ação

```

CREATE TRIGGER
  <nome do gatilho>
  <instante do gatilho>
  <evento do gatilho>
ON <nome da tabela>
[FOR EACH ROW]
[WHEN <condição>]
  <comando do gatilho>
  
```

BEFORE ou AFTER

INSERT, DELETE, ou UPDATE

144

Gatilhos

- Não podemos ter dois triggers para o mesmo instante e evento em uma mesma tabela
- Comando
 - BEGIN ... END
 - OLD.atributo e NEW.atributo
 - Sintaxe varia bastante entre os SGBDs
 - Em MySQL...

145

Gatilhos

(assertion_views.sql)

```
DELIMITER $$
CREATE TRIGGER SALARY_T
  BEFORE INSERT ON EMPLOYEE
  FOR EACH ROW
  BEGIN
    IF NEW.salary > (SELECT salary FROM EMPLOYEE
                     WHERE ssn=NEW.super_ssn)
    THEN SET NEW.salary = 0;
    END IF;
  END$$
DELIMITER ;
```

146

Visões

- Tabelas simples derivadas de outras tabelas (ou visões)
- Tabela virtual
 - Não necessariamente existe fisicamente
 - Limita as atualizações
- SQL

```
CREATE VIEW <nome da visão>
AS <consulta SQL>
```

147

Visões

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

```
WORKS_ON_NEW
  fname   lname   pname   hours
```

148

Visões

```
CREATE VIEW
  DEPT_INFO(dept_name, no_of_emps, total_sal)
AS SELECT dname, COUNT(*), SUM(salary)
  FROM DEPARTMENT, EMPLOYEE
  WHERE dnumber=dno
  GROUP BY dname;

DEPT_INFO
  dept_name  no_of_emps  total_sal
```

149

Visões

- Consultas podem ser feitas naturalmente sobre visões

```
QV1:
SELECT FNAME, LNAME
FROM WORKS_ON_NEW
WHERE PNAME='ProductX';
```

150

Visões

- Também podem ser usadas como mecanismo de acesso e segurança
- Sempre estão atualizadas
 - Alterações nas tabelas implicam em alterações nas visões
 - Visões são executadas não na criação, mas nas consultas

151

Implementando Visões

- Duas abordagens principais
 - Modificação de consulta
 - Materialização da visão

152

Implementando Visões

- Modificação de consulta
 - Altera a consulta da visão em uma consulta nas tabelas base

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



```
SELECT FNAME, LNAME
  FROM EMPLOYEE, PROJECT, WORKS_ON
   WHERE SSN=ESSN AND PNO=PNUMBER AND NAME='ProductX'
   GROUP BY PNAME;
```

153

Implementando Visões

- Modificação de consulta
 - Ineficiente para consultas complexas
 - Ineficiente se várias consultas são feitas à visão

154

Implementando Visões

- Materialização da visão
 - Cria uma tabela temporária na primeira consulta e a mantém
 - Precisa de mecanismos eficientes de atualização das tabelas materializadas
 - Técnicas de atualização incremental
 - Mantém as tabelas enquanto elas estão sendo consultadas
 - Não existente em MySQL

155

Atualizando Visões

- Complicado e Ambíguo
 - Não é o caso para visões sobre uma tabela sem funções agregadas
- Visões envolvendo junções podem ser mapeadas em várias atualizações de tabelas base. Cuidado!!!

156

Atualizando Visões

- Visões definidas usando grupos e funções agregadas não são atualizáveis
- Visões definidas sobre várias tabelas usando junção geralmente não são atualizáveis
- **WITH CHECK OPTION:** deve ser adicionado à definição da visão se ela for ser atualizada

157

Outras Características de SQL

- Linguagens de programação e SQL
- SGBDs têm seus próprios comandos para especificar parâmetros de projeto físico de BD (SDL)
 - Ex: Criação de índices

158

Outras Características de SQL

- Comandos de controle de transações
 - Permitem especificar unidades de processamento de BD para fins de concorrência e recuperação
- Construções para especificar privilégios de usuários
 - GRANT e REVOKE
 - Tabelas recebem proprietários
 - Usuários recebem o direito de usar certos comandos SQL sobre a relação
 - Usuários recebem o direito de criar esquemas, tabelas, visões

159

160

Estrutura da Aula

- SQL
- Técnicas de Programação SQL
- Transações
- Ferramentas Administrativas
- Laboratório
 - SQL
- Projeto
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas

161

Até agora...

- Aspectos da linguagem SQL
 - Padrão para os BDs relacionais
 - Definição de dados, modificação de esquemas, consultas, visões, atualizações
 - Restrições
- Como acessar um BD a partir de um programa?

162

Programação para BD

- Assunto bastante vasto
- Várias técnicas e livros dedicados a cada uma delas
- Criação de novas técnicas acontecem constantemente

163

Nesta Aula

- Introdução básica a algumas técnicas de programação para BD
- Exemplo prático de duas delas

164

Programação para BD

- A maioria dos SGBDs têm uma interface interativa
 - Digitação de comandos
 - Carregamento de arquivo com comandos
- Conveniente para
 - Criação do esquema
 - Criação de restrições
 - Consultas ad-hoc

165

Programação para BD

- Na prática, a maioria das interações são feitas através de programas *“cuidadosamente” projetados e testados*
 - Aplicações de BD
 - Interfaces Web
 - Servlets
 - PHP
- Várias abordagens para fazer isso...

166

Abordagens

- Comandos embutidos
 - Comandos de BD são embutidos em uma linguagem de programação de propósito geral
- Biblioteca de funções de BD
 - Disponível para a linguagem hospedeira para chamadas ao BD (API)
- Uma linguagem completamente nova
 - Minimiza a impedância de correspondência

167

Abordagens

- Mais comuns
 - Comandos embutidos
 - Biblioteca de funções de BD
- Ideal para aplicações com interação intensiva com o BD
 - Uma linguagem completamente nova

168

Impedância de Correspondência

- Incompatibilidade entre o modelo da linguagem hospedeira e o modelo do BD
 - Tipos de dados
 - Ligações de dados
 - Resultados de consultas são *bags* de tuplas (sequência de valores de atributos)
 - Ligações de estrutura de dados
 - Uso de cursor para fazer o *loop*

169

Interação com o BD

1. Programa cliente abre uma conexão com o servidor de BD
2. Programa cliente submete consultas/atualizações para o BD
3. Quando o acesso ao BD não é mais necessário o programa cliente fecha a conexão

170

Comandos Embutidos

- A maioria dos comandos SQL podem ser embutidos em uma linguagem de programação
 - Definições de dados, restrições, consultas, atualizações, visões
- Distinguimos os comandos SQL dos comandos da linguagem hospedeira
 - Sintaxe varia com a linguagem
 - `EXEC SQL` ou `EXEC SQL BEGIN`
 - `END-EXEC` ou `EXEC SQL END`
- Variáveis compartilhadas
 - Geralmente prefixadas com `(:)` em SQL
- Pré-processador pode separar os comandos SQL embutidos do programa na linguagem hospedeira

171

SQLJ

- Padrão para embutir SQL em Java
- O tradutor SQLJ transforma os comandos SQL em Java usando a interface JDBC
 - Executados via a interface *JDBC*

172

SQLJ

- Importando pacotes necessários
- Abrindo conexão
- Estabelecendo o contexto padrão

```

1) import java.sql.* ;
2) import java.io.*;
3) import sqlj.runtime.* ;
4) import sqlj.runtime.ref.* ;
5) import oracle.sqlj.runtime.* ;
...
6) DefaultContext cntxt =
7)   oracle.getConnection("<nome url>", "<nome usuario>", "<senha>", true) ;
8) DefaultContext.setDefaultContext(cntxt) ;
...

```

173

SQLJ Consultas

```

1) String dnome, ssn , pnome, fn, unome, ln, datanasc, endereco ;
2) Char sexo, minicial, mi ;
3) double salario, sal ;
4) Integer dno, dnumero ;

//Segmento de Programa J1:
1) ssn = readEntry("Entre com o Numero do Seguro Social: ") ;
2) try {
3)   #sql {select PNAME, MINICIAL, UNOME, ENDERECO, SALARIO
4)         into :pnome, :minicial, :unome, :endereco, :salario
5)         from EMPREGADO where SSN = :ssn} ;
6) } catch (SQLException se) {
7)   System.out.println("Numero do Seguro Social Inexistente: " + ssn) ;
8)   Return ;
9) }
10) System.out.println(pnome + " " + minicial + " " + unome + " " + endereco + " " +
    salario)

```

174

Bibliotecas de Acesso JDBC

- Java Database Connectivity
- Biblioteca de funções de BD
 - Disponível para a linguagem hospedeira para chamadas ao BD (API)
- Um programa Java com funções JDBC pode acessar qualquer SGBD relacional que tem um driver JDBC
- JDBC permite que um programa se conecte a vários BDs (fontes de dados)

175

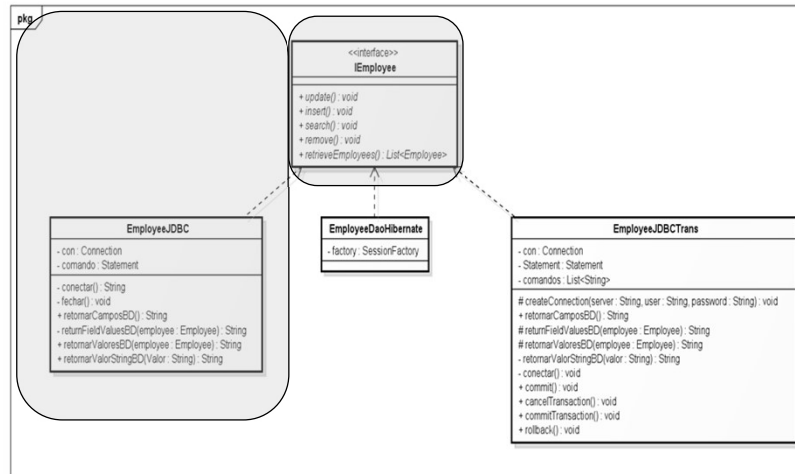
JDBC Passos do Acesso ao BD

1. Importar a biblioteca JDBC (`java.sql.*`)
2. Definir variáveis apropriadas
3. Criar uma conexão (`Connection`)
4. Criar um comando SQL (`Statement`)
6. Identificar parâmetros do comando SQL (designados por ?)
7. Ligar parâmetros a variáveis do programa
8. Executar comando SQL via JDBC (`executeQuery`)
9. Processar o resultado (`ResultSet`)

ResultSet é uma tabela bi-dimensional

176

JDBC Exemplo



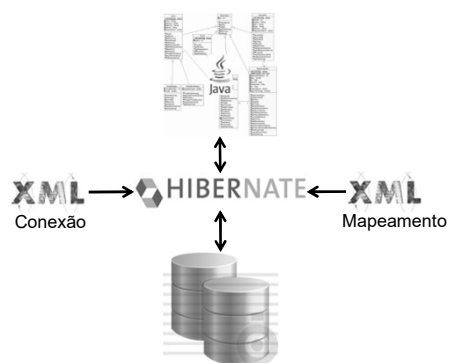
177

Hibernate

- Facilita o mapeamento OO-Relacional
- Transforma (vice-versa)
 - Classes Java em tabelas de dados
 - Tipos de dados
- Gera chamadas SQL e libera o desenvolvedor do trabalho manual de conversão dos dados resultantes
 - Portabilidade entre SGBDs

178

Hibernate



179

Hibernate

• Passos para uso

- Baixar Hibernate (www.hibernate.org) e importar os .jar da pasta hibernate\lib para a lib do seu projeto
- Criar a tabela no seu banco de dados onde os objetos vão persistir;
- Criar a classe cujo estado vai ser persistido;
- Criar um XML, que relaciona as propriedades do objeto aos campos na tabela

180

Hibernate

- Passos para uso

- Criar arquivos contendo as propriedades para que o *Hibernate* se conecte ao banco de dados
- Criar a classe DAO que vai persistir seu objeto;

181

Hibernate

- Exemplo do projeto

- Criação da classe da regra de negócio (`Employee`)
- Criar um XML, que relaciona as propriedades do objeto aos campos na tabela (`Employee.hbm.xml`)
- Criar arquivos contendo as propriedades para que o *Hibernate* se conecte ao banco de dados (`hibernate.properties`)
- `log4j.properties`
- Criar a classe DAO que vai persistir seu objeto (`EmployeeDaoHibernate.java`)

182

Programação BD com Chamadas de Funções

- SQL embutido provê programação BD estática
- API: programação BD dinâmica com uma biblioteca de funções
 - Vantagem: não precisa de pré-processamento (mais flexível)
 - Desvantagem: checagem do SQL é feita em tempo de execução

183

Procedimentos Armazenados

- Funções e procedimentos (módulos) são armazenados localmente e executados pelo SGBD
 - Ao contrário da execução no cliente
- Vantagens
 - Se o procedimento é necessário para muitas aplicações, ele pode ser invocado por quaisquer uma delas (evitando duplicações)
 - Execução pelo servidor reduz custos de comunicação
 - Aumenta o poder de definição de visões e gatilhos
- Desvantagens:
 - Problemas de portabilidade
 - Todo SGBD tem sua própria sintaxe
 - Alguns SGBDs permitem a definição de procedimentos em linguagens de programação de propósito geral

184

SQL Injection



- Ameaça à segurança e integridade de sistemas que interagem com bases de dados via SQL
- Uso de *strings* maliciosas

REMOVE EMPLOYEE
 SSN:



```
String sql = "DELETE FROM EMPLOYEE  
WHERE ssn=" + ssn;
```

185

SQL Injection



REMOVE EMPLOYEE
 SSN:



```
String sql = "DELETE FROM EMPLOYEE  
WHERE ssn=" + ssn;
```



```
String sql = "DELETE FROM EMPLOYEE  
WHERE ssn=111111100 OR TRUE";
```

186



SQL Injection

- Soluções

- Utilização de PreparedStatement
 - Ex: `EmployeeJDBC.unsafeRemove(String ssn)`
- Remover ou substituir palavras-chaves de SQL e caracteres especiais

```
String sql = "DELETE FROM EMPLOYEE  
             WHERE ssn=111111100 OR TRUE";
```

- Verificação de tipos de argumentos de consulta
- Permissões de acesso ao BD
 - Ex: Aplicações não deveriam ser capazes de apagar tabelas

[Clique aqui para mais sobre Statement vs PreparedStatement](#)

187

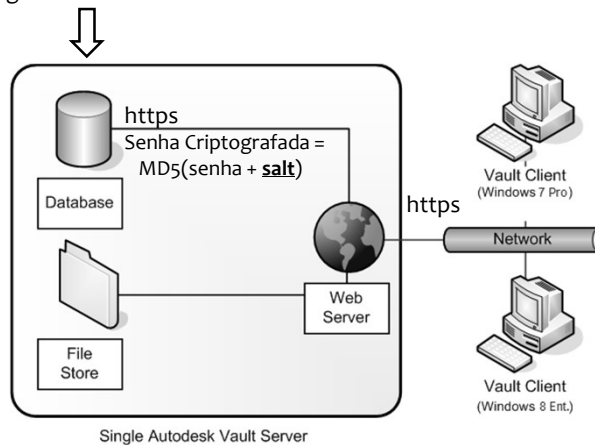
Dados Criptografados

- Como armazenar informações sigilosas (senhas) no BD?
- Como enviar/receber estas informações?

188

Dados Criptografados

Restringir acesso a IP do servidor



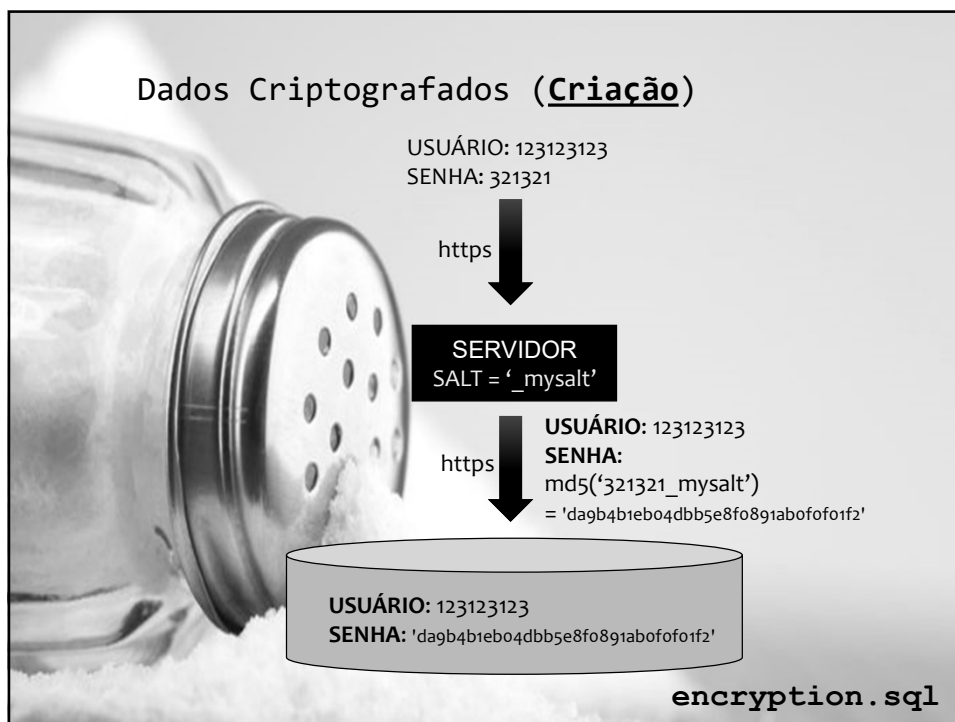
189

Dados Criptografados

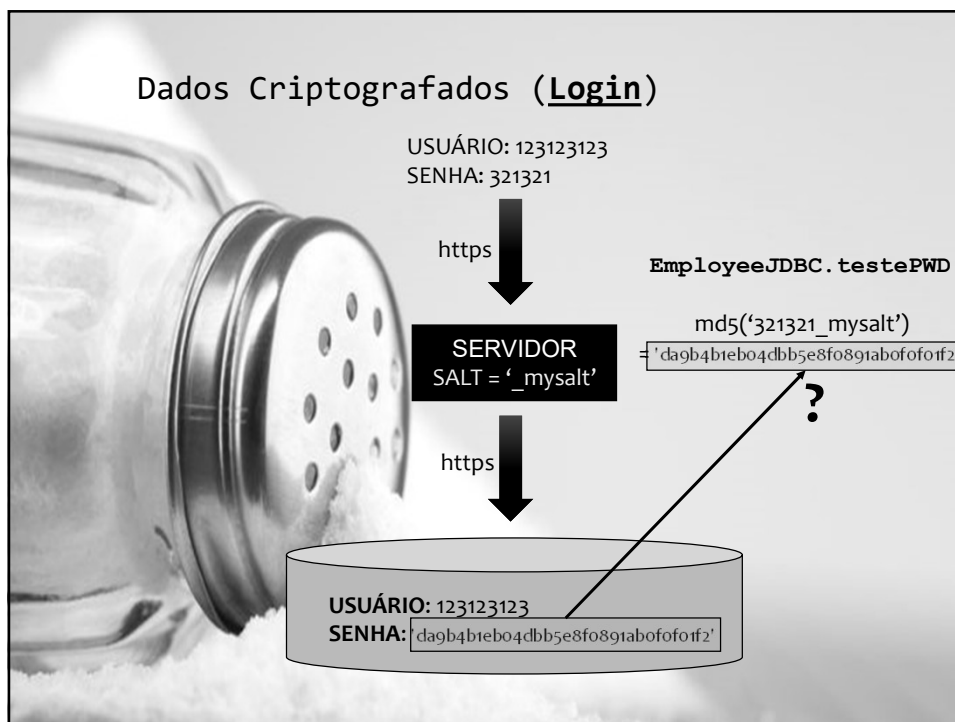


- Escolha do “sal”
 - Um por usuário
 - Um para todos guardado localmente
 - Combinação

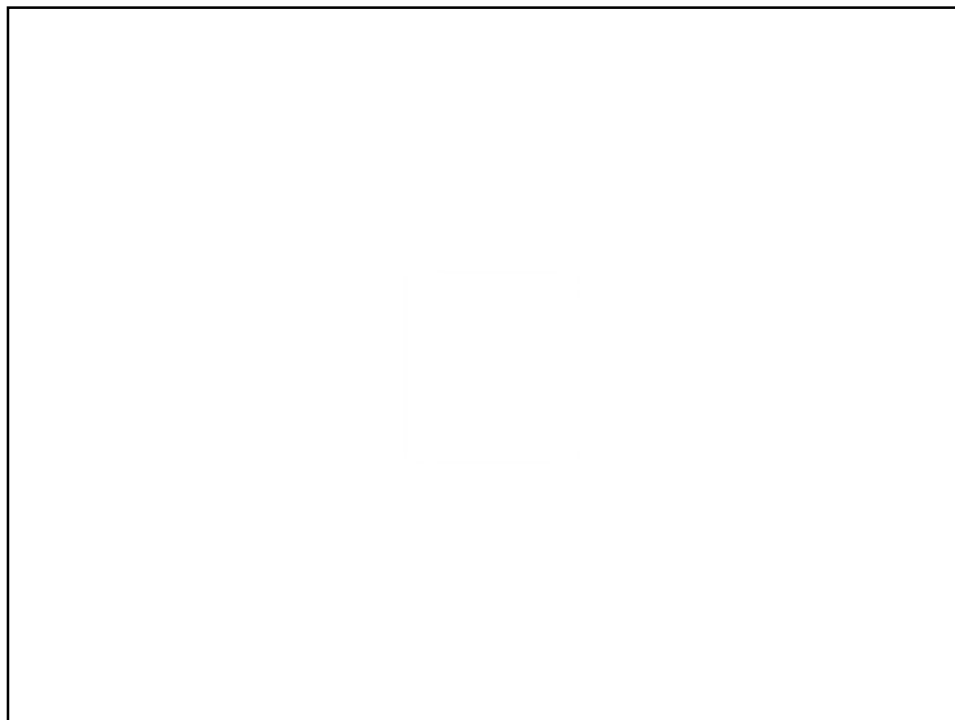
190



191



192



193

Estrutura da Aula

- SQL
- Técnicas de Programação SQL
- Transações
- Ferramentas Administrativas
- Laboratório
 - SQL
- Projeto
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas

194

Transação

- Conceito que fornece mecanismos para descrição de unidades lógicas de processamento de dados
- Sistemas de processamento de transações
 - Reserva de passagens
 - Bancos
 - Cartões de crédito
 - Ações
 - Compras on-line

195

Transação

- Sistemas de processamento de transações
 - Alta disponibilidade
 - Baixo tempo de resposta
 - Usuários concorrentes
- Extrema utilidade

196

Nesta Parte da Aula...

- Introdução ao processamento de transações
- Conceitos de transação e sistemas BD
- Propriedades desejáveis de transações
- Caracterização de plano de execução baseado na seriabilidade
- Suporte para transações em SQL

197

Transações

- Unidade lógica de processamento de BD que inclui um ou mais acessos
 - Leitura
 - Escrita (inserção ou atualização)
 - Remoção
- Uma transação (conjunto de operações) pode ser especificada em uma linguagem de alto-nível como SQL e submetida interativamente, ou pode fazer parte de um programa
- Delimitadores da transação
- Uma aplicação pode conter várias transações separadas pelos delimitadores de transação

198

Modelo Simples de BD

- Nesta aula, utilizaremos o seguinte modelo (simplificado) de BD
- BD é uma coleção de itens de dados com nomes
- Granularidade do dado
 - Um campo, um registro, bloco de disco
 - Conceitos independem da granularidade
- Operações básicas
 - `read_item(X)`: lê um item do BD chamado X em uma variável de programa (por simplificação, também chamada de X)
 - `write_item(X)`: escreve o valor de uma variável de programa X no item do BD chamado X

199

Exemplos de Transação

(a) T_1

```

read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);

```

Conjunto de leitura = { X, Y }
 Conjunto de escrita = { X, Y }

(b) T_2

```

read_item (X);
X:=X+M;
write_item (X);

```

Conjunto de leitura = { X }
 Conjunto de escrita = { X }

200

Por que Controle de Concorrência?

- Duas transações como T_1 e T_2 podem ser executadas por diferentes usuários concorrentemente
 - Acesso ao mesmo item do BD
 - Descontrole pode levar à inconsistência

201

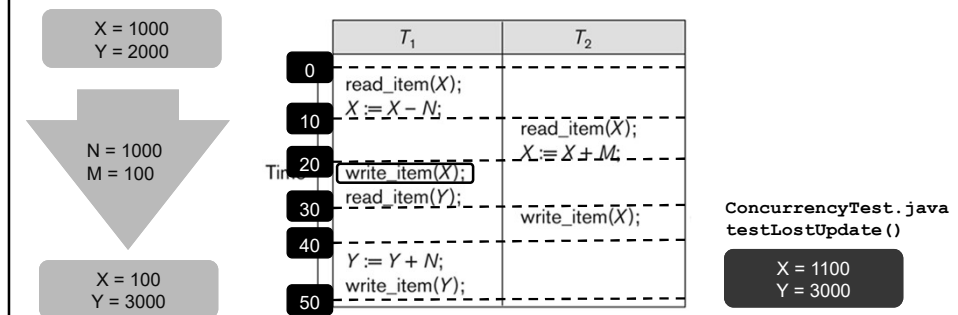
Por que Controle de Concorrência?

- Problemas
 - Atualização perdida
 - Atualização temporária (leitura suja)
 - Sumário incorreto

202

Problema da Atualização Perdida

- Ocorre quando duas transações que acessam os mesmos itens de BD têm suas operações intercaladas de uma maneira que uma sobrescreve a escrita da outra



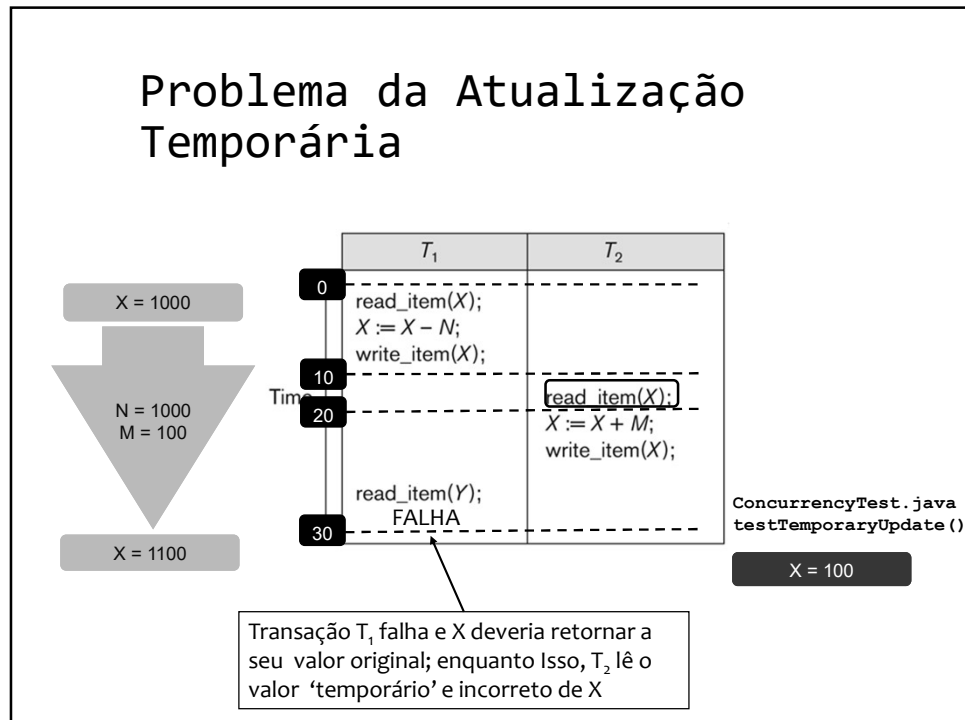
203

Problema da Atualização Temporária

- Ocorre quando uma transação atualiza um item do BD e então falha por alguma razão
- O item atualizado é acessado por outra transação antes que o seu valor retorne ao valor original

204

Problema da Atualização Temporária



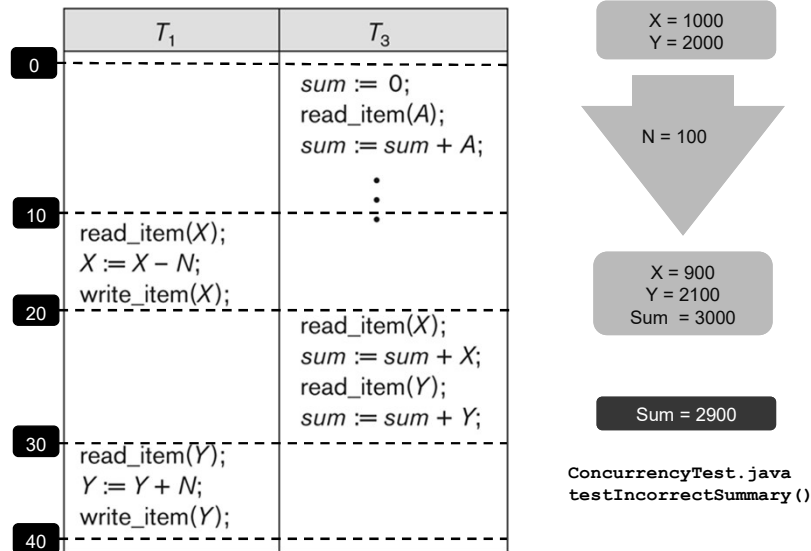
205

Problema do Sumário Incorreto

- Se uma transação está calculando uma função agregada sobre um número de itens enquanto outras transações estão atualizando alguns destes itens
- A função agregada pode calcular alguns valores antes deles serem atualizados e outros depois deles serem atualizados

206

Problema do Sumário Incorreto



207

Por que a Restauração é Necessária?

- Na execução de uma transação o SGBD deve garantir
 - Todas as operações foram bem sucedidas e o resultado será gravado no BD
- Caso contrário
 - A transação não terá nenhum efeito no BD
- Falhas de transação...

208

Falhas de Transação

- Falhas de transação

- Falha de computador

- Problema de hardware ou software durante a execução da transação

- Erro de transação ou de sistema

- Operações da transação podem levá-la a falhas (overflow, divisão por zero, parâmetros errados, lógica de programa)
 - Usuário cancela a execução

209

Falhas de Transação

- Erros locais ou condições de exceção detectadas pela transação

- Certas condições necessitam do cancelamento da transação

- Falta de dados
 - Saldo insuficiente
 - Programa aborta

- Imposição do controle de concorrência

- O método de controle de concorrência pode decidir abortar a transação, para ser reiniciada mais tarde, porque ela viola a serialização ou porque muitas transações estão em um estado de deadlock

210

Falhas de Transação

- Falha de disco
 - Mal funcionamento de leitura ou escrita, ou um *crash* no cabeçote de leitura e escrita podem levar à perda dos dados de blocos do disco
- Problemas físicos e catastróficos
 - Lista sem fim de problemas
- Essas duas falhas são mais “raras” que as outras

211

Conceitos de Transação

- Uma transação é uma unidade atômica de trabalho que ou é completada por inteiro ou não é realizada
 - A fim de poder restaurar, o sistema precisa manter o controle de quando a transação inicia (*start*), termina (*terminate*), comita (*commit*), e aborta (*abort*)

212

Conceitos de Transação

- Gerente de restauração mantém as seguintes informações

- ***begin_transaction***: Marca o início da execução de uma transação
- ***read*** ou ***write***: Marcam operações de leitura e escrita em itens do BD que são executadas como parte de uma transação
- ***end_transaction***: Marca o fim das operações de leitura e escrita e, conseqüentemente, o limite final da execução da transação
 - Verificar se as alterações podem ser efetivamente feitas no BD ou se ela deve ser abortada por violar controle de concorrência ou outros motivos

213

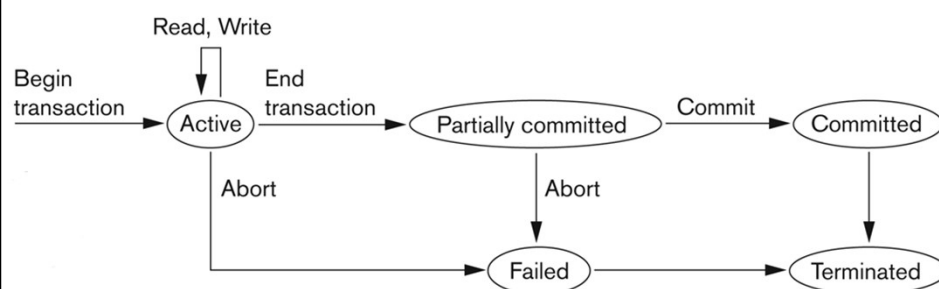
Conceitos de Transação

- Gerente de restauração mantém as seguintes informações

- ***commit_transaction***: indica término com sucesso de uma transação de modo que todas as suas alterações persistiram no BD
- ***rollback*** (ou ***abort***): indica que a transação terminou sem sucesso, e que quaisquer alterações feitas por esta transação no BD devem ser desfeitas

214

Estados de uma Transação



215

Propriedades ACID

- **Atomicidade**
 - Uma transação é uma unidade atômica de processamento (ou tudo ou nada)
- **Preservação de Consistência**
 - Uma execução correta de uma transação leva o BD de um estado consistente para um outro estado também consistente
- **Isolamento**
 - As atualizações de uma transação não deveriam ser percebidas por outras transações antes de seu *commit*
 - Resolve o problema da atualização temporária
- **Durabilidade**
 - Após o *commit*, nenhuma alteração feita por uma transação deve ser perdida devido a uma falha

216

Plano de Execução

- A ordem de execução de operações de transações concorrentes é chamada de **Plano de Execução (Schedule)**
 - T_1, \dots, T_n - Transações
 - S - Plano de Execução
 - Ordem em que as operações das diversas transações acontecem
 - As operações de cada transação T_i devem acontecer em S na mesma ordem em que elas acontecem em T_i
 - Operações de outras transações podem ser intercaladas

217

Plano de Execução

1	2	3	4
1	2	3	4
1	2	3	4

1	1	2	1	3	2	3	4	2	3	4	4
---	---	---	---	---	---	---	---	---	---	---	---

218

Plano de Execução

• Operações

- r – read
- w – write
- c – commit
- a – abort

• Exemplo

- $r_1(X)$ – leitura do item X pela transação T_1

219

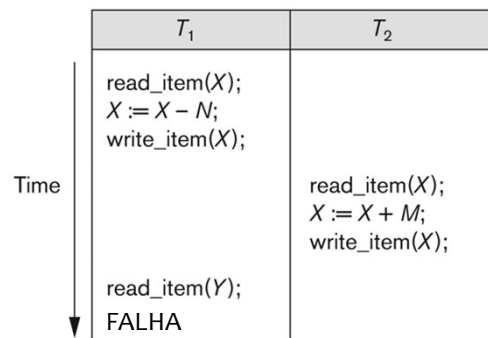
Plano de Execução

T_1	T_2
$\text{read_item}(X);$ $X := X - N;$ $\text{write_item}(X);$ $\text{read_item}(Y);$ $Y := Y + N;$ $\text{write_item}(Y);$	$\text{read_item}(X);$ $X := X + M;$ $\text{write_item}(X);$

$S_a: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

220

Plano de Execução



$$S_b: r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); a;$$

221

Plano de Execução

• **Conflito de operações**

- Pertencem a transações diferentes
- Acessam o mesmo item X
- Pelo menos uma das operações é um $\text{write_item}(X)$

$$S_a: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_2(Y);$$

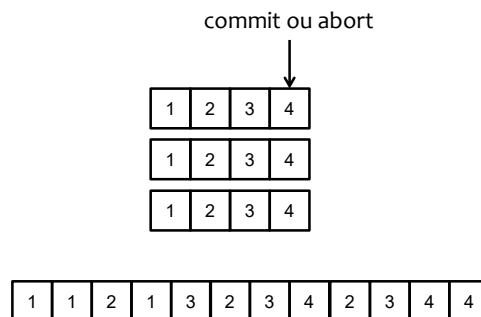

222

Plano de Execução

- **Plano completo** S de transações T_1, \dots, T_n
 - Operações em S são exatamente as de T_1, \dots, T_n tendo um commit ou um abort como última operação de T_1, \dots, T_n no plano
 - A ordem das operações de cada transação T_i no plano respeita a ordem das operações em T_i

223

Plano de Execução



224

Caracterizando Planos de Execução

- Em alguns planos a restauração é fácil e em outros bastante complicada
- É importante caracterizar os tipos de planos nos quais a restauração é possível (e/ou simples)

225

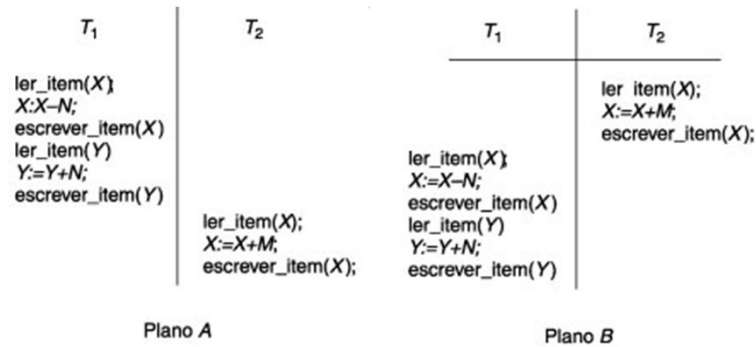
Caracterizando Planos

- Restaurabilidade (Capítulo 21)
- **Seriabilidade**

226

Caracterizando Planos baseados na Seriabilidade

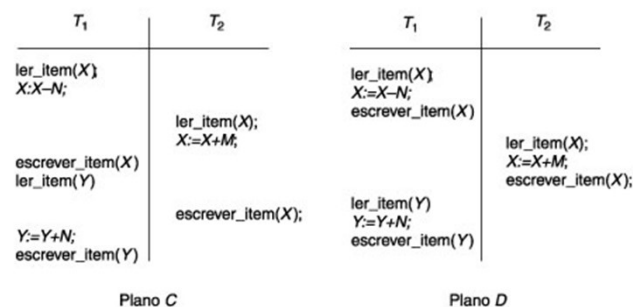
- Sem intercalação de transações temos duas possibilidades



227

Caracterizando Planos baseados na Seriabilidade

- Com intercalação de transações temos várias possibilidades

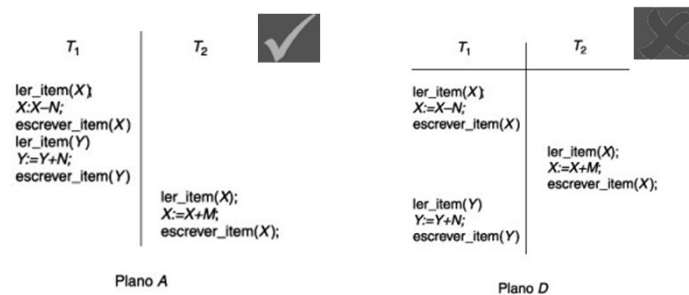


Quais delas são corretas??? Para isso usamos o conceito de seriabilidade de planos de execução

228

Caracterizando Planos baseados na Seriabilidade

- Plano de execução serial
 - Para toda transação T que participa do plano de execução, todas as operações de T são executadas consecutivamente no plano



229

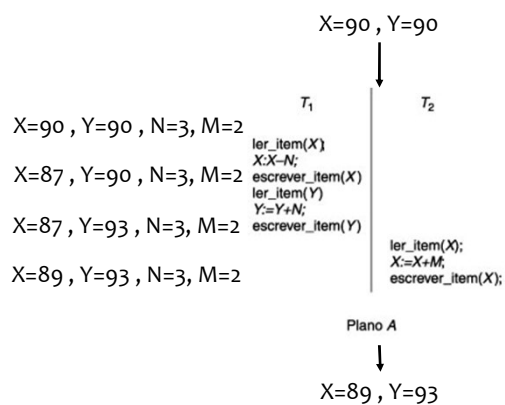
Caracterizando Planos baseados na Seriabilidade

- Plano de execução serial
 - Apenas uma transação está ativa por vez
 - Não há intercalação
 - Cada plano de execução é considerado correto (baseado no princípio da conservação da consistência)
 - Limitam a concorrência
 - Geralmente, inaceitáveis na prática

230

Caracterizando Planos baseados na Seriabilidade

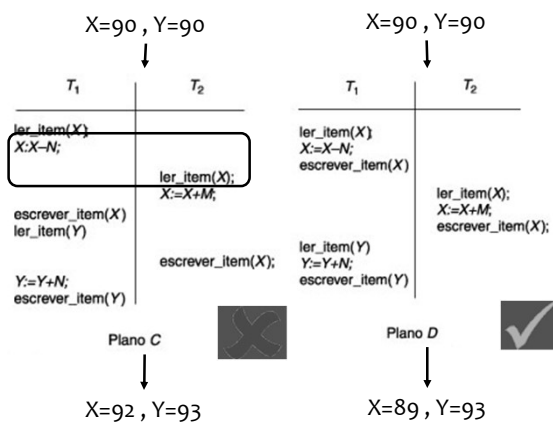
- Analisemos os planos de execução



231

Caracterizando Planos baseados na Seriabilidade

- Analisemos os planos de execução



232

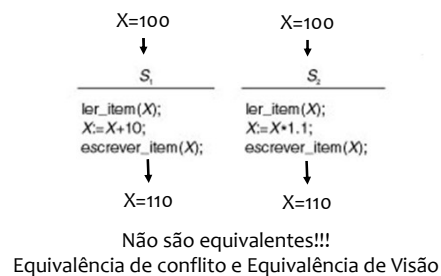
Caracterizando Planos baseados na Seriabilidade

- Que planos não-seriais sempre fornecem resultados corretos? Que planos podem fornecer resultados errôneos?
- Seriabilidade
 - Um plano de execução é serializável se ele é equivalente a algum plano de execução serial nas mesmas transações

233

Caracterizando Planos baseados na Seriabilidade

- Equivalência de planos
 - Resultados equivalentes?



234

Caracterizando Planos baseados na Seriabilidade

- Equivalência de conflito
 - Dois planos são conflito equivalentes se a ordem de quaisquer duas operações conflitantes é a mesma em ambos os planos
 - Conflito de Operações
 - Exemplo
 - $S_1: \dots; r_1(X); \dots; w_2(X); \dots$
 - $S_2: \dots; w_2(X); \dots; r_1(X); \dots$



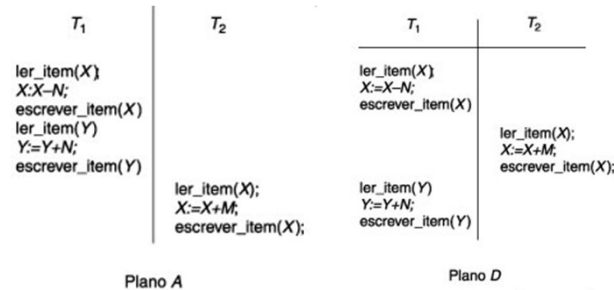
235

Caracterizando Planos baseados na Seriabilidade

- Plano de execução conflito serializável
 - Se ele for conflito equivalente a algum plano de execução serial
 - Podemos reordenar as operações não conflitantes em S até formar um plano de execução serial equivalente

236

Caracterizando Planos baseados na Seriabilidade



$P_A: r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X);$

$P_D: r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); w_1(Y);$

T_1 é a última a alterar Y

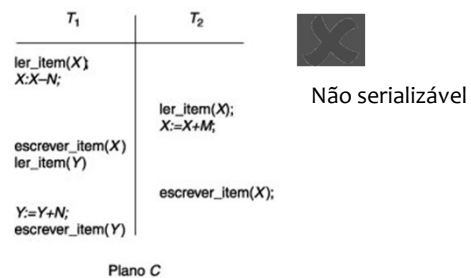
T_2 é a última a alterar X

↑ Não conflitantes

$P_D: r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X);$

237

Caracterizando Planos baseados na Seriabilidade



$P_A: r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X);$

$P_C: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

$P_B: r_2(X); w_2(X); r_1(X); w_1(X); r_1(Y); w_1(Y);$

$P_C: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

238

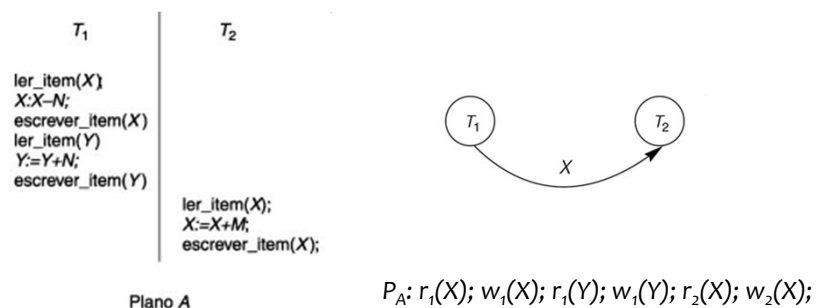
Caracterizando Planos baseados na Seriabilidade

- Testando o conflito serialidade de um plano
 - Olha apenas para operações de leitura e escrita
 - Constrói um grafo de precedência (de serialização) com arestas direcionadas
 - Cada transação é um nó
 - Uma aresta é criada de T_i para T_j se uma das operações de T_i aparece antes de alguma operação conflitante de T_j
 - O plano de execução é serializável se, e somente se, o grafo não tiver ciclos

239

Caracterizando Planos baseados na Seriabilidade

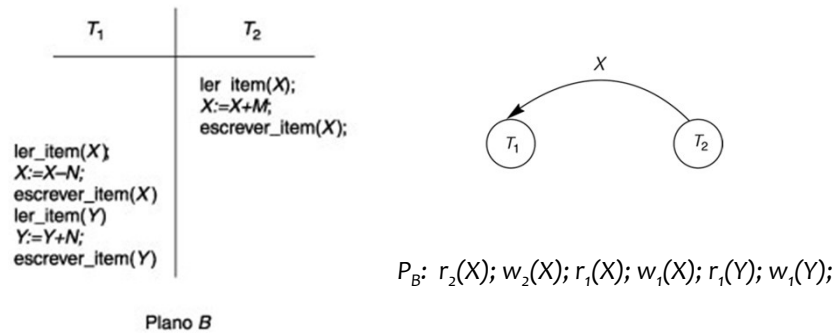
- Testando o conflito serialidade de um plano



240

Caracterizando Planos baseados na Seriabilidade

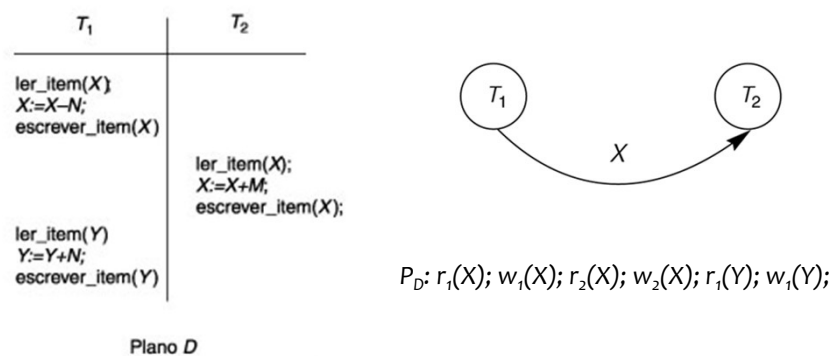
- Testando o conflito serialidade de um plano



241

Caracterizando Planos baseados na Seriabilidade

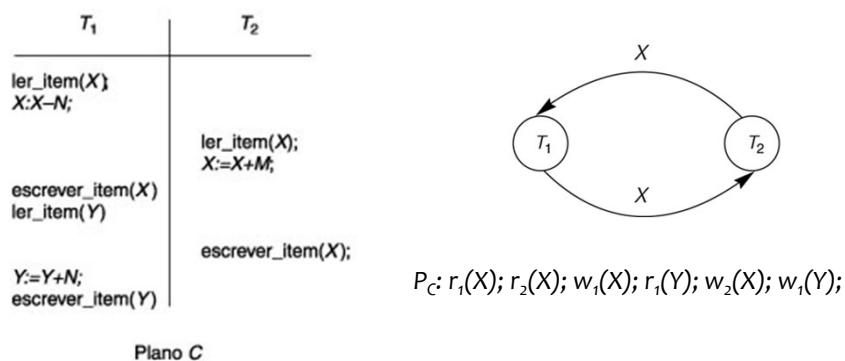
- Testando o conflito serialidade de um plano



242

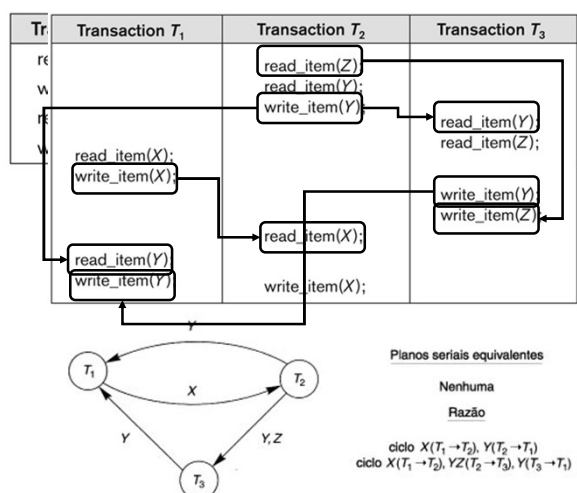
Caracterizando Planos baseados na Seriabilidade

- Testando o conflito serialidade de um plano



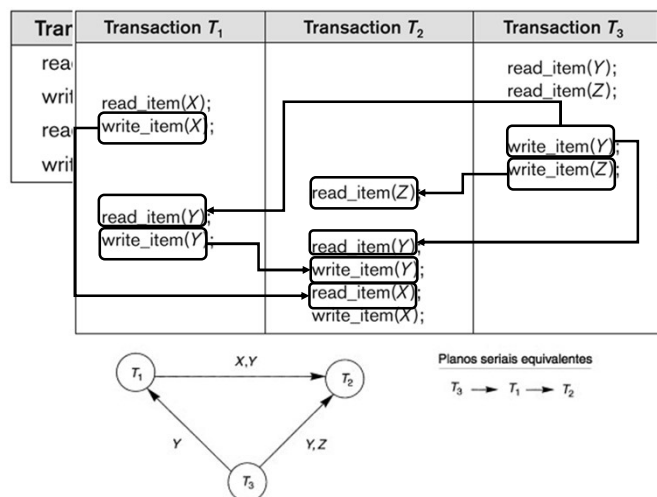
243

Caracterizando Planos baseados na Seriabilidade



244

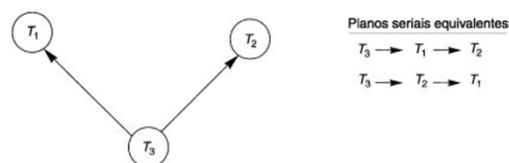
Caracterizando Planos baseados na Seriabilidade



245

Caracterizando Planos baseados na Seriabilidade

- Pode haver mais de um plano serial equivalente



246

Caracterizando Planos baseados na Seriabilidade

- Ser serializável é diferente de ser serial
- Ser serializável implica que o plano de execução é correto
 - Deixará o BD em um estado consistente
 - A intercalação é apropriada e resultará em um estado como se as transações tivessem sido executadas serialmente
 - Foram, porém, executadas eficientemente (com concorrência)

247

Caracterizando Planos baseados na Seriabilidade

- Na prática, seriabilidade é difícil de verificar
 - Operações intercaladas ocorrem em um sistema operacional através de um agendamento
 - Difícil de determinar diante mão como as operações serão intercaladas
 - Impraticável cancelar o efeito de um plano caso verificarmos que ele não é serializável

248

Caracterizando Planos baseados na Seriabilidade

- Abordagem prática
 - Uso de **protocolos** que garantem a seriabilidade de todas as suas transações
 - Não é possível determinar quando um agendamento começa e quando ele termina
 - Checamos apenas as operações de transações efetivadas

249

Caracterizando Planos baseados na Seriabilidade

- Abordagem atual na maioria dos SGBDs
 - Uso de bloqueios em duas fases (*two-phase locking*)
- Outros protocolos
 - Multi-versões (PostgreSQL)
 - Ordenação por marca de tempo (*timestamp*)
 - Otimistas

250

Caracterizando Planos baseados na Seriabilidade

- Equivalência de visão (Capítulo 21)

251

Transações em SQL

- Conceitos similares aos vistos aqui
- Uma declaração SQL é considerada atômica
- Início da transação
 - Implícito: `SET AUTOCOMMIT=0 ;`
 - Explícito: `START TRANSACTION`
- Fim da transação
 - Explícito: `COMMIT` ou `ROLLBACK`

252

Transações em SQL

- Outras características podem ser especificadas usando (antes do início da transação)

SET TRANSACTION

- Modo de acesso
 - **Padrão:** READ WRITE
 - READ ONLY
- Escopo
 - **Padrão:** aplica-se apenas à próxima transação
 - GLOBAL
 - SESSION

253

Transações em SQL

- Outras características podem ser especificadas usando (antes do início da transação)

SET TRANSACTION

- Nível de isolamento
 - **Padrão:** REPEATABLE READ
 - READ UNCOMMITTED
 - READ COMMITTED
 - SERIALIZABLE

254

Transações em SQL

• Sintaxe

• Modo de acesso

```
SET [GLOBAL | SESSION] TRANSACTION
{ READ ONLY | READ WRITE }
```

• Nível de isolamento

```
SET [GLOBAL | SESSION] TRANSACTION
ISOLATION LEVEL
{ READ UNCOMMITTED
  | READ COMMITTED
  | REPEATABLE READ
  | SERIALIZABLE }
```

255

Transações em SQL

- Com **SERIALIZABLE** a execução intercalada de transações respeitará nossa noção de seriabilidade
- Possíveis violações da seriabilidade nos outros níveis de isolamento
 - **Leitura de sujeira:** Leitura de um valor que foi escrito por uma transação que falhou (valor incorreto)
 - **Leitura não-repetível:** Permitindo que uma outra transação escreva um novo valor entre múltiplas leituras de uma transação (valores possivelmente diferentes para um mesmo item)
 - **Fantasmas:** Novas linhas sendo lidas usando a mesma condição de leitura
 - Uma transação T_1 faz uma consulta ao BD (**SELECT**) que retornará uma tabela contendo as tuplas que satisfazem a busca. Se depois T_2 inserir uma nova tupla no BD e essa tupla satisfizer a condição, e T_1 for repetida, teremos um fantasma

256

Transações em SQL

Possíveis violações baseadas em isolamento Níveis como definidos em SQL			
Nível de Isolamento	Leitura Suja	Não-repetível	Fantasma
READ UNCOMMITTED	Sim	Sim	Sim
READ COMMITED	Não	Sim	Sim
REPEATABLE READ	Não	Não	Sim
SERIALIZABLE	Não	Não	Não

257

Transações em MySQL

- (transactions.sql)

258

Transações em JDBC

```
conn.setAutoCommit(false);
```

```
stat.executeUpdate(...);
```

```
stat.executeUpdate(...);
```

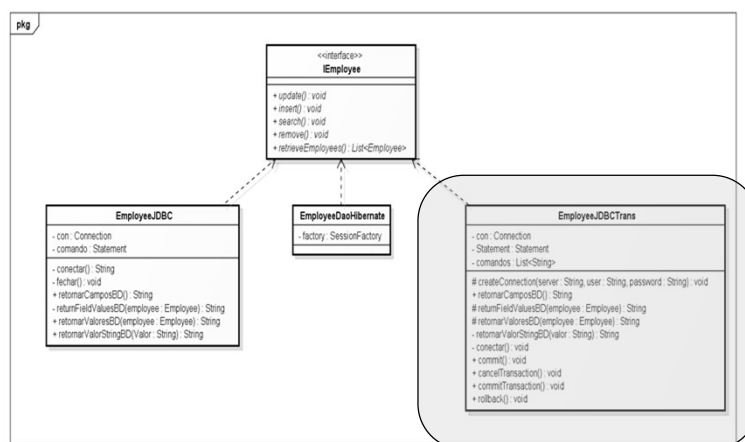
```
stat.executeUpdate(...);
```

```
conn.commit();
```

```
conn.rollback();
```

259

Transações em JDBC



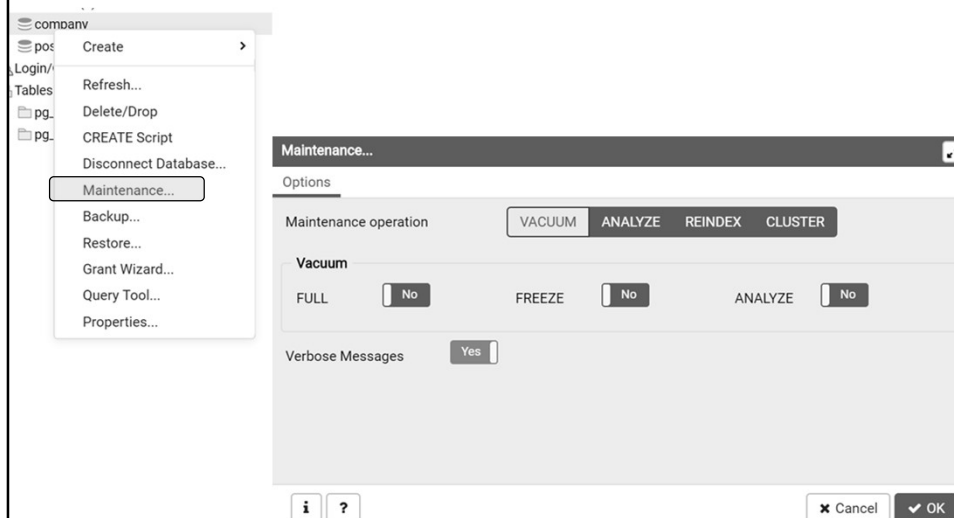
260

Estrutura da Aula

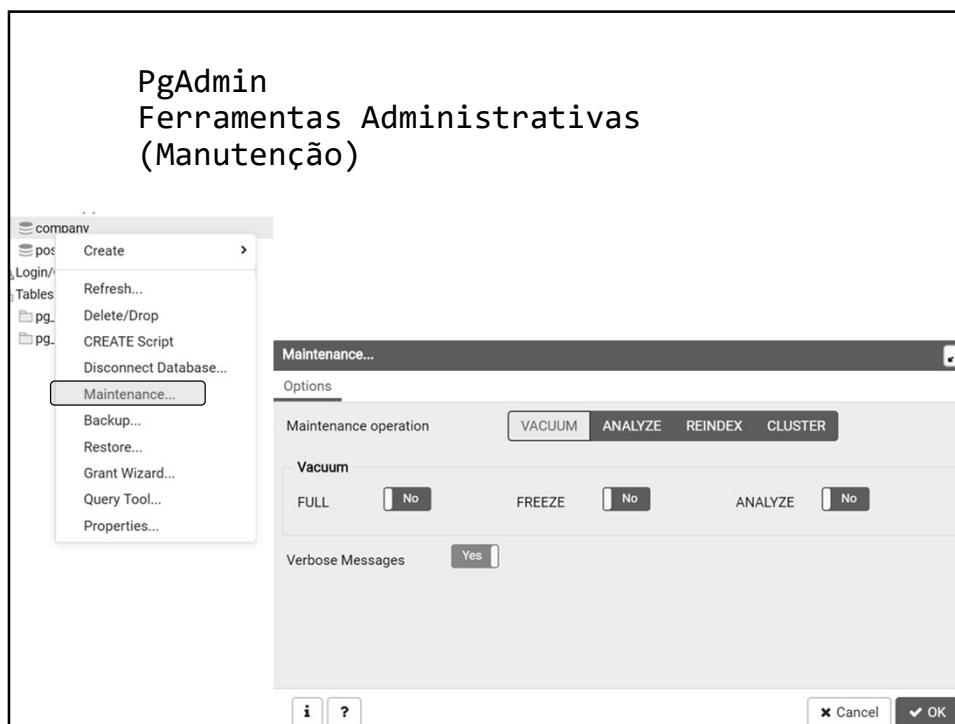
- SQL
- Técnicas de Programação SQL
- Transações
- Ferramentas Administrativas
- Laboratório
 - SQL
- Projeto
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas

261

PgAdmin Ferramentas Administrativas



262



263

PgAdmin: VACUUM

- MVCC (Multiversion Concurrency Control)
 - Remoção de registro não apaga o registro mas coloca uma marca nele
 - Atualização de um registro cria uma nova versão e marca o anterior
 - Mais eficiência
- VACUUM
 - Remoção dos registros marcados
 - Reorganização em disco dos registro que permanecem
 - Atualização das estatísticas do otimizador de consultas (opcional)

264

PgAdmin: VACUUM



265

PgAdmin: VACUUM

```
VACUUM [ ( { FULL | FREEZE | VERBOSE | ANALYZE } [, ...] ) ] [ table_name [ (column_name [, ...]) ] ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table_name ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table_name [ (column_name [, ...]) ] ]
```

- FULL
 - Limpeza completa em todas as tabelas e coluna
 - Reorganização em disco dos registro que permanecem
 - Bloqueia a utilização do banco durante a sua execução (FREEZE)
 - Deve ser utilizado com muito cuidado e em raros casos
 - Sem este parâmetro, apenas a limpeza é feita sem bloqueio
 - `autovacuum (SHOW autovacuum;)`
- VERBOSE
 - Relatório detalhado do que está sendo feito
- ANALYZE
 - Atualiza as estatísticas do otimizador de consultas
- tabela/coluna
 - Especifica onde o VACUUM será feito case seja desejado

266

PgAdmin: ANALYZE

```
ANALYZE [ VERBOSE ] [ table [ ( column [, ...] ) ] ]
```

- Coleta estatísticas sobre o conteúdo de tabelas no banco de dados e armazena os resultados no catálogo do sistema `pg_statistic`.
- Posteriormente, o planejador de consulta usa essas estatísticas para ajudar a determinar os planos de execução mais eficientes para as consultas.
- Recomendável fazer com frequência via *autovacuum*

267

PgAdmin: REINDEX

```
REINDEX [ ( VERBOSE ) ] { INDEX | TABLE | SCHEMA | DATABASE | SYSTEM } name
```

- Reconstrói um índice usando os dados armazenados na tabela do índice, substituindo a cópia antiga do índice.
- Cenários para usar o REINDEX
 - Recuperação de índice
 - Um índice foi corrompido e não contém mais dados válidos.
 - Raro, mas possível
 - Reconstrução do índice que contém muitas páginas vazias ou quase vazias.
 - Alteração de parâmetros de armazenamento (ex. fator de preenchimento) para um índice e deseja garantir que a alteração tenha efeito total.
 - Uma construção de índice com a opção `CONCURRENTLY` falhou, deixando um índice "inválido".

268

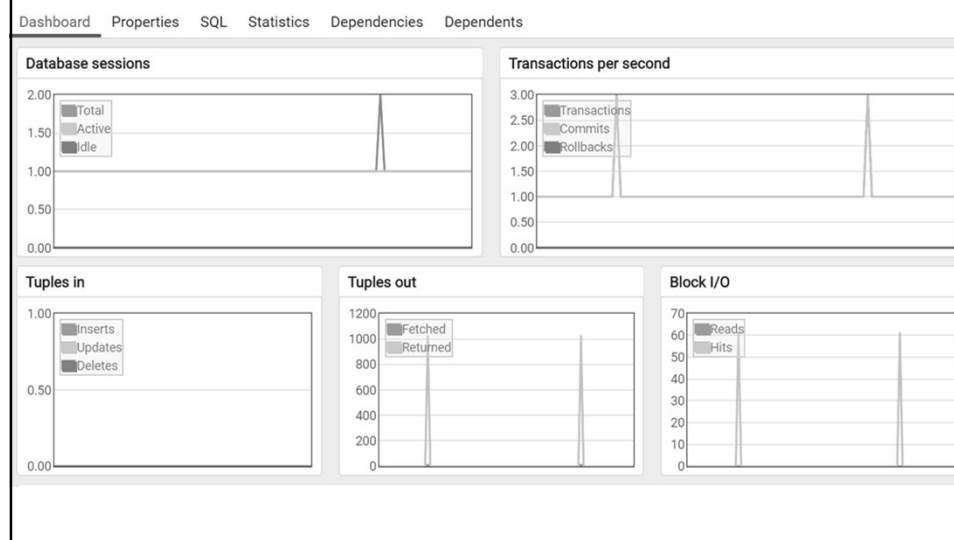
PgAdmin: CLUSTER

```
CLUSTER [VERBOSE] table_name [ USING index_name ]
CLUSTER [VERBOSE]
```

- Agrupar a tabela especificada por `table_name` com base no índice especificado por `index_name`.
- O índice já deve ter sido definido em `table_name`.
- Sem nenhum parâmetro reagrupa todas as tabelas já agrupadas anteriormente
- Bloqueia a tabela durante o reagrupamento

269

PgAdmin: Outras ferramentas (Dashboard)



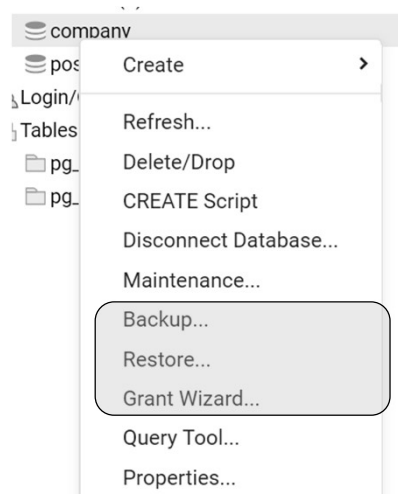
270

PgAdmin: Outras ferramentas (Statistics)

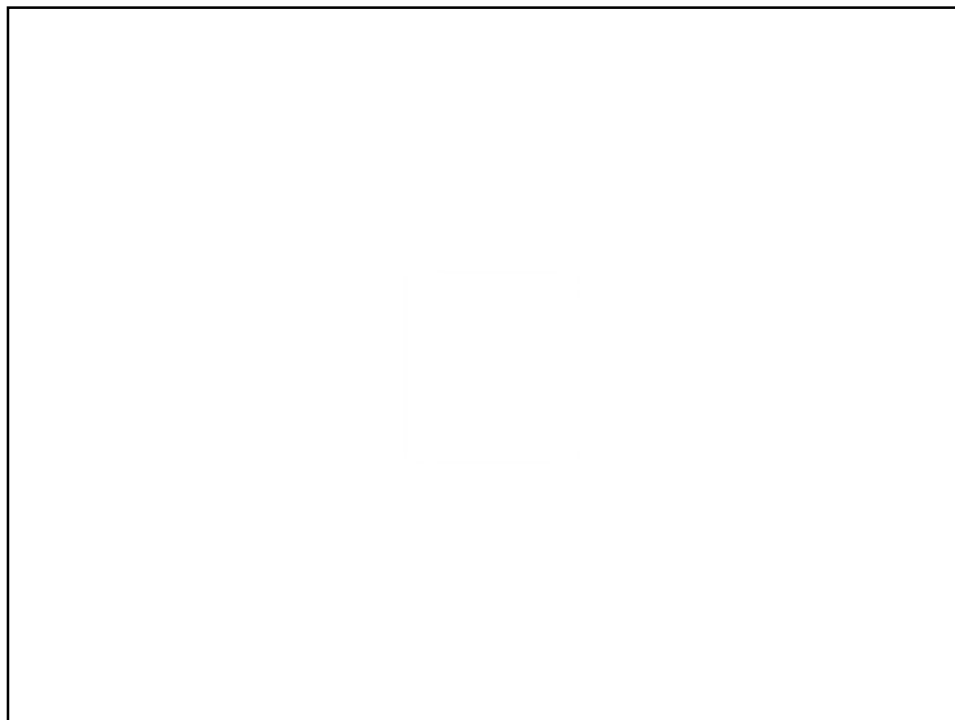
Dashboard Properties SQL Statistics Dependencies Dependents	
Statistics	Value
Backends	1
Xact committed	4800
Xact rolled back	5
Blocks read	1562
Blocks hit	79359
Tuples returned	378123
Tuples fetched	33900
Tuples inserted	1995
Tuples updated	1396
Tuples deleted	1733
Last statistics reset	2019-08-23 09:20:02.983618-03
Tablespace conflicts	0
Lock conflicts	0
Snapshot conflicts	0
Bufferpin conflicts	0
Deadlock conflicts	0

271

PgAdmin: Outras ferramentas



272



273

Estrutura da Aula

- SQL
- Técnicas de Programação SQL
- Transações
- Ferramentas Administrativas
- Laboratório
 - SQL
- Projeto
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas

274



- **Laboratório SQL Workbench.pdf** (disponível no SIGAA)
 - Exercício
- **Projeto**
 - Construção do BD Relacional
 - Inserção de Dados no BD
 - Criação de Consultas