Banco de Dados I

05 - Modelo Físico / SQL

Arthur Porto - IFNMG Campus Salinas

arthur.porto@ifnmg.edu.br
arthurporto.com.br

Sumário I

Sumário II

Sumário III

Introdução

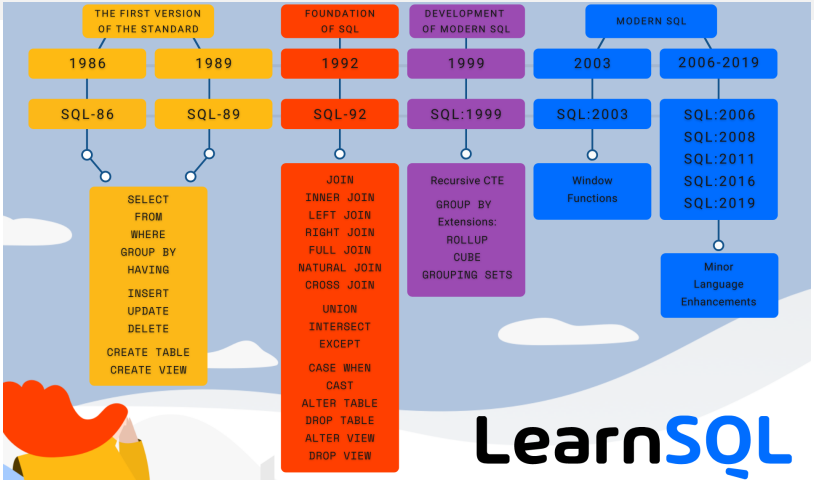


Figura 1: https://learnsql.com/blog/history-of-sql-standards/

Introdução

- SQL
  - *Structured Query Language* - Linguagem de Consulta Estruturada.
  - Já foi chamada de SEQUEL (*Structured English QUEry Language*).
  - Linguagem declarativa em alto nível.
  - Implementada pela IBM para ser a interface do SYSTEM R.
  - É uma DDL e DML
    - *Data Definition Language*
    - *Data Manipulation Language*

Figura 2: **How I use SQL as a Data Analyst**

Introdução

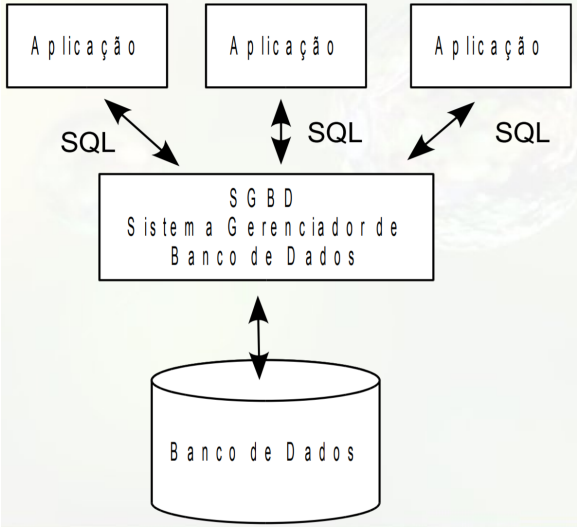


Figura 3: Fonte https://goo.gl/4zbCu3

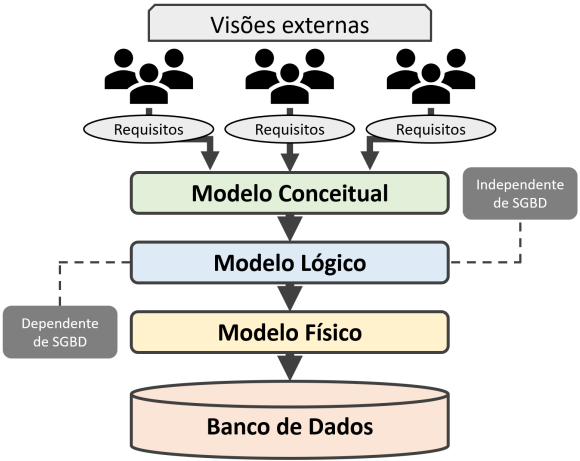


Figura 4: Etapas projeto de BDs.

Ferramentas

- **PostgreSQL**
  - pgAdmin: **Instalando o PostgreSQL e Criando o Primeiro Banco de Dados**
- **bit.io: Postgres in 5 seconds**
- **How to Set Up a PostgreSQL Database with Docker**

Esquema SQL

- Tabela, linha e coluna = relação, tupla e atributo.
- O esquema SQL é identificado por um nome e o seu proprietário (identificador de autorização).
- Não são todos os usuário que estão autorizados a criar esquemas.
- Criado pela instrução *CREATE SCHEMA*.
- Catálogo
  - Uma coleção de esquemas.
  - Possuí um esquema especial *INFORMATION_SCHEMA* com as informações dos outros esquemas.

Criar esquema

```
1 -- CREATE SCHEMA <NOMESCHEMA> AUTHORIZATION <IdAutorizado>;
2 CREATE SCHEMA EMPRESA AUTHORIZATION 'Jsilva';
```

Esquema SQL

Exemplo Postgres

Criar Banco

```
1 -- No prompt de comando
2 psql -U nome_do_usuario -c 'create database nome_do_banco;'
```

Acessando Banco

```
1 -- Após acessar o banco
2 \c nome_do_banco
```

Criando *Schema*

```
1 -- Após acessar o banco
2 \c nome_do_banco
```

- **DOC: Creating a Database**
- **DOC: Accessing a Database**
- **DOC: Creating a Schema**

Esquema SQL - CREATE TABLE

- Especificar uma nova relação.
  - Atributos + Tipos
  - Restrições de chave
  - Restrições referenciais

CREATE TABLE

```
1  CREATE TABLE FUNCIONARIO
2  (
3    Pnome      VARCHAR(16) NOT NULL,
4    Nome       VARCHAR(64) NOT NULL,
5    Cpf        CHAR(11)  NOT NULL,
6    Datanasc   DATE,
7    Endereco   VARCHAR(32),
8    Sexo       CHAR  DEFAULT 'm',
9    Salario    DECIMAL(10,2) CHECK(Salario>0 AND Salario<1000),
10   Cpf_sup    CHAR(11)  NOT NULL,
11   Dnr        INT NOT NULL CHECK(Dnr>0 AND Dnr<20),
12   PRIMARY KEY(Cpf),
13   FOREIGN KEY(Cpf_sup) REFERENCES FUNCIONARIO(Cpf),
14   FOREIGN KEY(Dnr) REFERENCES DEPARTAMENTO(Dnumero)
15 );
```

- **DOC: Creating a New Table**
- **DOC: CREATE TABLE**

Esquema SQL - Tipos de dados

- Numéricos.
  - Inteiros: INTEGER ou INT.
  - Ponto flutuante: FLOAT ou REAL.
  - Decimal: DECIMAL(i,j) ou DEC(i,j) ou NUMERIC(i,j).
- Cadeia de caracteres.
  - Tamanho fixo: CHAR(n) ou CHARACTER(n).
  - Tamanho variável: VARCHAR(n)
- Cadeia de bits: BIT(n).
- Booleano: TRUE, FALSE ou UNKNOWN

Esquema SQL - Tipos de dados

- Data
  - DATE: DD-MM-YYYY
  - TIME: HH:MM:SS
  - TIMESTAMP: DD-MM-YYY HH:MM:SS
- Criar um domínio

Criando domínio

```
1 -- CREATE DOMAIN NOME_DOMINIO TIPO_DOMINIO;
2 CREATE DOMAIN TIPO_CPF CHAR(11);
```

- **DOC: Data Types**
- **DOC: CREATE DOMAIN**

Esquema SQL - Restrições: Atributo

- *NOT NULL*: Define um atributo que não pode ser nulo.
- *DEFAULT*: Estabelece um valor padrão para o atributo.
- *CHECK*: Limites de valores para um atributo.

CREATE TABLE

```
1  CREATE TABLE FUNCIONARIO
2  (
3    Pnome      VARCHAR(16) NOT NULL,
4    Nome       VARCHAR(64) NOT NULL,
5    Cpf        CHAR(11)  NOT NULL,
6    Datanasc   DATE,
7    Endereco   VARCHAR(32),
8    Sexo       CHAR  DEFAULT 'm',
9    Salario    DECIMAL(10,2) CHECK(Salario>0 AND Salario<1000),
10   Cpf_sup    CHAR(11)  NOT NULL,
11   Dnr        INT NOT NULL CHECK(Dnr>0 AND Dnr<20),
12   PRIMARY KEY(Cpf),
13   FOREIGN KEY(Cpf_sup) REFERENCES FUNCIONARIO(Cpf),
14   FOREIGN KEY(Dnr) REFERENCES DEPARTAMENTO(Dnumero)
15 );
```

- **DOC: Constraints**

Esquema SQL - Restrições: Chave

- *PRIMARY KEY* (chave primária): especifica os atributos que compõe a chave.

Especificação de chave

```
1  -- Primera forma de especificação
2  CREATE TABLE NOME_RELACAO
3  (
4    Nome_atributo_PK TIPO_ATRIBUTO,
5    ...
6    PRIMARY KEY (Nome_atributo_PK),
7    ...
8  );
9  -- Segunda forma de especificação
10 CREATE TABLE NOME_RELACAO
11 (
12   Nome_atributo TIPO_ATRIBUTO PRIMARY KEY,
13   ...
14 );
```

Esquema SQL - Restrições: Chave alternativa

- *UNIQUE*: especifica as chaves secundárias.

Especificação de chave alternativa

```
1  -- Primera forma de especificação
2  CREATE TABLE NOME_RELACAO
3  (
4    Nome_atributo TIPO_ATRIBUTO,
5    ...
6    UNIQUE (Nome_atributo),
7    ...
8  );
9  -- Segunda forma de especificação
10 CREATE TABLE NOME_RELACAO
11 (
12   Nome_atributo TIPO_ATRIBUTO UNIQUE,
13   ...
14 );
```

Esquema SQL - Restrições: Integridade referencial

- *FOREIGN KEY* (chave estrangeira).
  - Referencia a outra relação com base na chave primária da mesma relação.

Especificação de chave estrangeira

```
1  CREATE TABLE RELACAO_1
2  (
3    R1_Atributo1 Dominio1,
4    ...
5    PRIMARY KEY (R1_Atributo1),
6    ...
7  );
8  CREATE TABLE RELACAO_2
9  (
10   R2_Atributo1 Dominio2,
11   R2_Atributo2 Dominio1,
12   ...
13   PRIMARY KEY (R2_Atributo1),
14   FOREIGN KEY (R2_Atributo2) REFERENCES RELACAO_1(R1_Atributo1)
15   ...
16 );
```

Esquema SQL - Restrições: Integridade referencial

- Restrições contra possíveis violações de integridade.
  - Atualizações
    - ON DELETE
    - ON UPDATE
  - Ações
    - NO ACTION: Impede ação na tabela referência.
    - SET NULL: Valores de referência serão alterados para nulo.
    - CASCADE: Propaga a ação da tabela referência.
    - SET DEFAULT: Valores de referência alterados para o padrão.
  - **DOC: CREATE TABLE**

Esquema SQL - Integridade referencial: Restrições

Relação FUNCIONARIO

```
1  CREATE TABLE FUNCIONARIO
2  (
3    Nome       VARCHAR(64) NOT NULL,
4    Cpf        CHAR(11)  NOT NULL,
5    Datanasc   DATE,
6    Endereco   VARCHAR(32),
7    Sexo       CHAR  DEFAULT 'm',
8    Salario    DECIMAL(10,2) CHECK(Salario>0 AND Salario<1000,
9    Cpf_sup    CHAR(11)  NOT NULL,
10   Dnr        INT NOT NULL CHECK(Dnr>0 AND Dnr<20),
11   PRIMARY KEY(Cpf),
12   FOREIGN KEY(Cpf_sup) REFERENCES FUNCIONARIO(Cpf) ON DELETE SET NULL ON UPDATE CASCADE,
13   FOREIGN KEY(Dnr) REFERENCES DEPARTAMENTO(Dnumero) ON DELETE SET DEFAULT ON UPDATE CASCADE
14 );
```

Relação DEPARTAMENTO

```
1  CREATE TABLE DEPARTAMENTO
2  (
3    Dnumero      INT            NOT NULL,
4    Dnome        VARCHAR(16)    NOT NULL,
5    Cpf_gerente  CHAR(11)       NOT NULL,
6    Data_inicio_gerente DATE,
7    PRIMARY KEY (Dnumero),
8    UNIQUE (Dnome),
9    FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf) ON DELETE SET DEFAULT ON UPDATE CASCADE
10 );
```

Esquema SQL - DROP TABLE

- Excluir uma tabela

DROP TABLE

```
1 | DROP TABLE EMPLOYEE;
```

- **DOC: DROP TABLE**

Esquema SQL - ALTER TABLE

- Alterar a estrutura de uma tabela

ALTER TABLE

```
1 ALTER TABLE table_name
2 -- opções
3 ADD column_name datatype;
4 DROP COLUMN column_name;
5 ALTER COLUMN column_name datatype;
```

- **DOC: ALTER TABLE**
- PostgreSQL permite uma variedade de ações
  - Adicionar coluna, Excluir coluna, Mudar o tipo da coluna, Renomear a coluna, Definir um valor padrão, Adicionar uma restrição, renomear a tabela, etc.

Esquema SQL - Exercício 01

- Escreva o código SQL para criar o esquema:
  - PESSOA (nome, nome_da_mae, ano_nascimento, nome_cidade_natal)
    - nome_cidade_natal : Chave Estrangeira de CIDADE.
  - CIDADE (nome_cidade, sigla_estado)

Recuperação

- Recuperação de informações, ou seja, consultas nas tabelas (relações).
- Instrução básica **SELECT**.
- Estrutura básica

Estrutura SQL

```
1 SELECT <lista atributos>
2 FROM <lista de tabelas>
3 WHERE <condição>;
```

- **DOC: QUERIES**

Recuperação

- Consulta
  - *Recuperar a data de nascimento e o endereço do(s) funcionário(s) cujo nome seja 'John B. Smith'.*
    - Relação utilizada

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Select

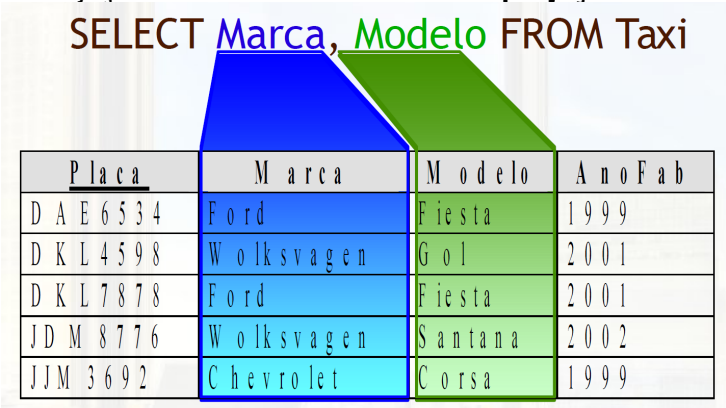
```
1  SELECT  Bdate, Address
2  FROM    EMPLOYEE
3  WHERE   Fname='John'
4          AND Minit='B'
5          AND Lname='Smith';
```

- Resultado

Bdate	Address
1965-01-09	731Fondren, Houston, TX

Recuperação - Projeção

- O SELECT realiza uma **Projeção** e uma **Seleção**.
- Projeção: Através dos **atributos de projeção**.



Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Figura 5: Fonte https://goo.gl/4zbCu3

- Resultado

Marca	Modelo
Ford	Fiesta
Wolksvagen	Gol
Ford	Fiesta
Wolksvagen	Santana
Chevrolet	Corsa

Recuperação - Seleção

- Seleção: Através das **condições de seleção**.

SELECT * FROM Taxi WHERE AnoFab > 2000

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Figura 6: Fonte https://goo.gl/4zbCu3

- Resultado

Placa	Marca	Modelo	AnoFab
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002

Recuperação - Junção

- Combinação de tuplas.
- Consulta
  - *Recuperar o nome e endereço de todos os funcionários que trabalham para o departamento Pesquisa*.
    - Relações utilizadas

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Condição de junção

```
1 SELECT  Fname, Lname, Address
2 FROM    EMPLOYEE, DEPARTMENT
3 WHERE   Dname=`Research` AND Dnumber=Dno;
```

- *Dnumber* = *Dno* é uma **condição de junção** [**?**].

- Resultado

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

Recuperação - Seleção-Projeção-Junção

- Consulta que envolve apenas condições de seleção e junção, mais atributos de projeção.
  - *Para cada projeto localizado em 'Stafford', liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento*.
    - Relações utilizadas

PROJECT

Pname	Pnumber	Plocation	Dnum

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Condição de junção

```
1  SELECT   Pnumber, Dnum, Lname, Address, Bdate
2  FROM     PROJECT, DEPARTMENT, EMPLOYEE
3  WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn  AND Plocation='Stafford';
```

- Resultado

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Recuperação - Exercício 02

- Para a tabelas que você montou no exercício, escreva um comando SQL que retorne:
  - nomes de todas as mães.
  - nomes de todas as mães com filhos maiores de 12 anos.
- Esquemas
  - PESSOA (nome, nome_da_mae, ano_nascimento, nome_cidade_natal)
    - nome_cidade_natal : Chave Estrangeira de CIDADE.
  - CIDADE (nome_cidade, sigla_estado)

Recuperação - SELECT ALL

- Recuperando todos os valores de uma relação

Omitindo o WHERE

```
1 SELECT  *
2 FROM    <lista de tabelas>
```

- Recuperando todos os atributos das tuplas selecionadas

SELECT ALL - *

```
1 SELECT  *
2 FROM    <lista de tabelas>
3 WHERE   <condição>;
```

SELECT * FROM Taxi WHERE AnoFab > 2000

Placa	Marca	Modelo	AnoFab
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002

Figura 7: Fonte https://goo.gl/4zbCu3

Recuperação - Ambiguidade

- Ambiguidade de nomes de atributos
- Exemplo
  - *Employee (name, ssn, bdate, address, sex, salary, super_snn, dnumber)*
  - *Department (name, dnumber, mgr_ssn, mgr_start_date)*

Ambiguidade

```
1 SELECT  EMPLOYEE.Name, Address
2 FROM    EMPLOYEE, DEPARTMENT
3 WHERE   DEPARTMENT.Name='Research'
4         AND DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

- Atributos com nomes semelhantes em relações diferentes.
- Basta especificar o nome da relação junto ao atributo.

- Ou pode-se utilizar “*apelidos*” (AS)

Apelido - AS

```
1 SELECT  Fname, E.Name, Address
2 FROM    EMPLOYEE AS E, DEPARTMENT AS D
3 WHERE   D.Name='Research'
4         AND D.Dnumber=E.Dnumber;
```

- Renomeando os atributos

Apelido dos atributos

```
1 EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)
```

Recuperação - Duplicatas

- A SQL não elimina automaticamente as duplicadas.
- Para eliminar duplicatas usa-se o **DISTINCT**
  - Consulta
    - *Recuperar o salário de cada funcionário.*
  - Relações

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

(a)

Salary
30000
40000
25000
43000
38000
25000
25000
55000

(b)

Salary
30000
40000
25000
43000
38000
55000

Removendo duplicatas

```
1  -- a
2  SELECT   ALL Salary
3  FROM     EMPLOYEE;
4  -- b
5  SELECT   DISTINCT Salary
6  FROM     EMPLOYEE;
```

- **DOC: DISTINCT**

Recuperação - Operação de comparação

- Usado para combinação de padrão de cadeia.
  - % Qualquer cadeia com 0 ou n caracteres
  - _ exatamente um caractere (qualquer)
    - Relação utilizada.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

- Consulta: Recuperar todos os funcionários cujo endereço esteja em Houston, Texas.

Comparação LIKE

```
1 SELECT  Fname, Lname
2 FROM    EMPLOYEE
3 WHERE   Address LIKE '%Houston TX%';
```

- **DOC: LIKE**

- Consulta: Encontrar todos os funcionários o nome começa com a letra 'a' e tem no mínimo 3 caracteres.

Comparação LIKE

```
1 SELECT  Fname, Lname
2 FROM    EMPLOYEE
3 WHERE   Bdate LIKE 'A__%';
```

Recuperação - Operação de comparação

Relação

TAXI			
Placa	Marca	Modelo	AnoFab

- *Placas que comecem com DK*

SQL

```
1 SELECT  *
2 FROM    TAXI
3 WHERE   Placa LIKE 'DK%';
```

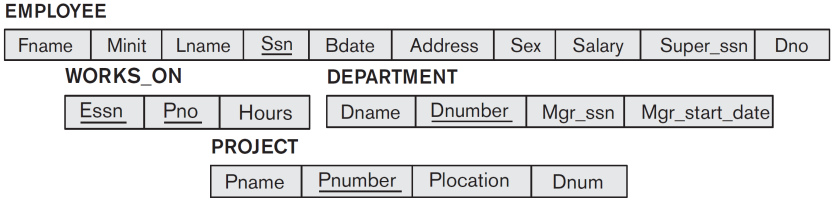
- *Placas com 7 na penúltima posição*

SQL

```
1 SELECT  *
2 FROM    TAXI
3 WHERE   Placa LIKE '%7_';
```

Recuperação - Ordenação

- Permite que o usuário ordena as tuplas.
- Relações



- Consulta
  - Recupere uma lista dos funcionários e dos projetos em que estão trabalhando, ordenada por departamente e, dentro de cada departamento, ordenada alfabeticamente pelo sobrenome, depois pelo nome.

Ordenação

```
1  SELECT    D.Dname, E.Lname, E.Fname, P.Pname
2  FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W, PROJECT P
3  WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND W.Pno= P.Pnumber
4  ORDER BY  D.Dname, E.Lname, E.Fname;
```

INSERT

- Usado para acrescentar tupla à relação.
  - Deve respeitar a ordem dos atributos.

INSERT

```
1 | INSERT INTO <RELAÇÃO>
2 | VALUES      (V1, V2, ..., Vn);
```

- Relação

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Inserindo tuplas

```
1 | INSERT INTO EMPLOYEE
2 | VALUES      ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '
  |      653298653', 4);
```

- **DOC: INSERT**

INSERT

- Usado para acrescentar **apenas** alguns atributos da tupla à relação.
  - Respeitar a ordem fornecida.
  - Os não definidos recebem o **DEFAULT** ou **NULL**.
  - Deve-se tomar cuidado com as restrições de integridade.

INSERT

```
1 | INSERT INTO <RELAÇÃO> (A1, A2, ..., An)
2 | VALUES      (V1, V2, ..., Vn);
```

- Relação

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Inserindo tuplas

```
1 | INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
2 | VALUES      ('Richard', 'Marini', 4, '653298653');
```

INSERT - INSERT + SELECT

CREATE TABLE

```
1 CREATE TABLE WORKS_ON_INFO
2 (
3   Emp_name VARCHAR(15),
4   Proj_name VARCHAR(15),
5   Hours_per_week DECIMAL(3,1)
6 );
```

INSERT + SELECT

```
1 INSERT INTO WORKS_ON_INFO (Emp_name, Proj_name, Hours_per_week)
2 SELECT      E.Lname, P.Pname, W.Hours
3 FROM        PROJECT P, WORKS_ON W, EMPLOYEE E
4 WHERE       P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

- Geralmente utilizada para tabelas temporárias [**?**].

DELETE

- Remove tuplas de uma relação.
- A exclusão de uma tupla pode disparar uma ação em outra relação devido as restrições referenciais.

DELETE

```
1  -- 1
2  DELETE FROM EMPLOYEE
3  WHERE Lname='Brown';
4  -- 2
5  DELETE FROM EMPLOYEE
6  WHERE Ssn='123456789';
7  -- 3
8  DELETE FROM EMPLOYEE
9  WHERE Dno=5;
```

- Exclusão da relação: DROP
- **DOC: DELETE**

UPDATE

- Modificar valores nas tuplas.
- A atualização de uma tupla pode disparar uma ação em outra relação devido as restrições referenciais.
- Relações

PROJECT

Pname	Pnumber	Plocation	Dnum

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

UPDATE - Única tupla

```
1  UPDATE   PROJECT
2  SET      Plocation = 'Bellaire', Dnum = 5
3  WHERE    Pnumber=10;
```

- **DOC: UPDATE**

UPDATE - Múltiplas tuplas

```
1  UPDATE   EMPLOYEE
2  SET      Salary = Salary * 1.1
3  WHERE    Dno = 5;
```

Junção - JOIN - Produto Cartesiano

- Uma consulta que envolve duas tabelas sem condições (sem WHERE) e sem associação entre atributos das tabelas.

Produto Cartesiano

```
1 SELECT  ...
2 FROM    <tabela1>, <tabela2>
```

- **DOC: JOIN**

Junção - JOIN - Produto Cartesiano

Produto Cartesiano

```
1 | SELECT   Cliente.CliId, Cliente.Nome, Corrida.CliId, Corrida.Placa, Corrida.DataPedido
2 | FROM     Cliente, Corrida
```

- Cliente

CliId	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02

- Corrida

CliId	Placa	DataPedido
1532	JDM8776	2003-02-18
1755	DAE6534	2003-02-15

- Produto Cartesiano

CliId	Nome	CliId	Placa	DataPedido
1532	Asdrúbal	1532	JDM8776	2003-02-18
1532	Asdrúbal	1755	DAE6534	2003-02-15
1755	Doriana	1532	JDM8776	2003-02-18
1755	Doriana	1755	DAE6534	2003-02-15
1780	Quincas	1532	JDM8776	2003-02-18
1780	Quincas	1755	DAE6534	2003-02-15

Junção - JOIN - Definição

- Uma tabela resultante de uma operação de junção (JOIN).
- Junção implícita é realizada através das condições de junção.
- Junção explícita é o tipo clássico
  - Também conhecido como INNER JOIN
- Existem outros tipos
  - NATURAL JOIN
  - OUTER JOIN

Junção - JOIN - INNER JOIN

Junção implícita

```
1 | SELECT   Cliente.CliId, Cliente.Nome, Corrida.CliId, Corrida.Placa,
2 |          Corrida.DataPedido
3 | FROM     Cliente, Corrida
4 | WHERE    Cliente.CliId = Corrida.CliId
```

- Cliente

CliId	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02

- Corrida

CliId	Placa	DataPedido
1532	JDM8776	2003-02-18
1755	DAE6534	2003-02-15

- JOIN

CliId	Nome	CliId	Placa	DataPedido
1532	Asdrúbal	1532	JDM8776	2003-02-18
1755	Doriana	1755	DAE6534	2003-02-15

Junção - JOIN - INNER JOIN

- Relações

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

- Consulta: Recupere o nome e endereço de todos os funcionários que trabalham para o departamento ‘Research’

Junção explícita

```
1  SELECT  Fname, Lname, Address
2  FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
3  WHERE   Dname='Research';
```

Junção - JOIN - INNER JOIN: Exercício

- Com base no esquema
  - MEDICAMENTO(nomeVenda, compostoAtivo)
  - VIRUS(nomeCientifico, nomePopular, incubacao, **nomeVendaMed**)
    - Chave estrangeira: nomeVendaMed para medicamento
- Recupere as seguintes informações
  - Nome popular do vírus tratado pelo medicamento com o nome de venda ‘W’.
  - Nome popular do vírus tratado pelo medicamento com o composto ativo ‘X’.

Junção - JOIN - INNER JOIN: Exercício

- Com base no esquema abaixo:
  - PESSOA (nome, nome_da_mae, ano_nascimento, nome_cidade_natal)
    - nome_cidade_natal : Chave Estrangeira de CIDADE.
  - CIDADE (nome_cidade, sigla_estado)
- Recupere as seguinte informações
  - Nomes de pessoas que nasceram no mesmo estado que você

Junção - JOIN - INNER JOIN: Exercício

- Com base no esquema abaixo:
  - CLIENTE (CliId, Nome, CPF)
  - TAXI (Placa, Marca, Modelo, AnoFab)
  - CORRIDA (CliId, Placa, DataPedido)
    - CliId: *FK* de CLIENTE
    - Placa: *FK* de TAXI
- Recupere as seguinte informações
  - Qual o modelo de taxi para cada corrida?
  - Quais os modelos de taxi tomados por cada cliente?

Junção - JOIN - NATURAL JOIN

- No NATURAL JOIN as condições não são especificadas.
- É criada a condição *EQUIJOIN* implícita
  - Condição que junta cada par de atributos com o mesmo nome nas relações R e S.
- Se os nomes não forem iguais, basta renomeá-los.
- Cada atributo é adicionado apena uma vez na relação resultante.

Junção - JOIN - NATURAL JOIN

JOIN Padrão

```
1 | SELECT  *
2 | FROM    cliente JOIN corrida on cliente.CliId = corrida.CliId;
```

NATURAL JOIN

```
1 | SELECT  *
2 | FROM    cliente NATURAL JOIN corrida
```

- Cliente

CliId	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02

- Corrida

CliId	Placa	DataPedido
1532	JDM8776	2003-02-18
1755	DAE6534	2003-02-15

- JOIN Padrão

CliId	Nome	CPF	CliId	Placa	DataPedido
1532	Asdrúbal	448.754.253-65	1532	JDM8776	2003-02-18
1755	Doriana	567.387.387-44	1755	DAE6534	2003-02-15

- NATURAL JOIN

CliId	Nome	CPF	Placa	DataPedido
1532	Asdrúbal	448.754.253-65	JDM8776	2003-02-18
1755	Doriana	567.387.387-44	DAE6534	2003-02-15

Junção - JOIN - OUTER JOIN

- Junção externa
  - Mantém todas as tuplas de uma das relações, presentes na junção, independentemente de possuírem tuplas correspondentes na realação. beginitemize
  - Relações

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

  - Para todos os funcionário, recupere o último nome do funcionário e o último nome de seu supervisor imediato.

Junção externa - OUTER JOIN

```
1 SELECT  E.Lname AS Employee_name, S.Lname AS Supervisor_name
2 FROM    EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S ON E.Super_ssn=S.Ssn;
```

Junção - JOIN

- Opções de junção
  - Junção Interna: INNER JOIN
    - Apenas pares de tuplas que combinam com a condição de junção.
  - Junção Externa: OUTER JOIN
    - LEFT OUTER JOIN: Todas as tuplas da relação à esquerda devem aparecer
    - RIGHT OUTER JOIN: Todas as tuplas da relação à esquerda devem aparecer
    - FULL OUTER JOIN: Todas as tuplas aparecem
  - NATURAL JOIN O
    - Pode existir junções externas naturais: Ex.: NATURAL LEFT OUTER JOIN
  - Produto cartesiano
    - CROSS JOIN

Funções de agregação

- **DOC: Funções Agregadas**

- COUNT
  - Retorna o número de tuplas ou valores, conforme especificado na consulta.
  - Relações

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

- Recuperar o número total de funcionários da empresa

COUNT

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

- Recuperar o número total de funcionários no departamento ‘Research’

COUNT

```
1 SELECT  COUNT (*)
2 FROM    EMPLOYEE, DEPARTMENT
3 WHERE   Dno = Dnumber AND Dname = 'Research';
```

Funções de agregação

- SUM, MAX, MIN, e AVG

- Relações

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

- Recuperar a soma, o máximo, o mínimo e a média dos salários dos funcionários.

Agregação em uma relação

```
1 SELECT  SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
2 FROM    EMPLOYEE;
```

- ... do departamento ‘Research’

Agregação com junção

```
1 SELECT  SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
2 FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
3 WHERE   Dname = 'Research';
```

Agrupamento

- As funções de agrupamento servem para particionar a relação em subgrupos de tuplas.
- Os grupos são formados pelas tuplas com o mesmo valor no(s) **atributos de agrupamento**.
- **DOC: GROUP BY**

Agrupamento - GROUP BY

- Essa cláusula especifica os atributos de agrupamento.
  - que também devem estar no SELECT
- Relação

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

- Para cada departamento, recupere o número do departamento, o número de funcionários e o salário médio.

Agrupamento

```
1 SELECT    Dno, COUNT (*), AVG (Salary)
2 FROM      EMPLOYEE
3 GROUP BY  Dno;
```

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Figura 8: Fonte: [?]

Agrupamento - GROUP BY

- Recuperar os modelos dos taxis e a sua quantidade.

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Agrupamento

```
1 SELECT   Modelo, COUNT(*) AS Qtd
2 FROM     Taxi
3 GROUP BY Modelo
```

Modelo	Qtd
Corsa	1
Fiesta	2
Gol	1
Santana	1

Agrupamento - HAVING

- Estabelece algumas condições para o agrupamento
  - Limita após o agrupamento
  - WHERE - Antes do agrupamento
- Relação

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

- Para cada projeto em que mais de dois funcionários trabalham, recupere o nome e o número de funcionários que trabalham no projeto.

Agrupamento

```
1  SELECT    Pname, COUNT (*)
2  FROM      PROJECT, WORKS_ON
3  WHERE     Pnumber=Pno
4  GROUP BY  Pname
5  HAVING    COUNT (*) > 2;
```

- **DOC: HAVING**

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Figura 9: Fonte: [?]

Outros Tópicos

- Views
- Triggers
- Consultas aninhadas
- UNIQUE e EXISTS

Outros Tópicos - : Exercício

- Com base no esquema
  - MEDICAMENTO(nomeVenda, compostoAtivo)
  - VIRUS(nomeCientifico, nomePopular, incubacao, **nomeVendaMed**)
    - Chave estrangeira: nomeVendaMed para medicamento
- Recupere as seguintes informações
  - Quantos vírus são tratados por cada medicamento?
  - Quantos vírus são tratados por cada composto ativo?
  - Quais os compostos ativos que tratam um vírus com período de incubação maior que 5 dias?
  - Quais os compostos ativos que tratam mais que 5 vírus?

Referências