#SomosTodos#Trier

Banco de Dados

Banco de dados é uma coleção lógica coerente de dados com um significado inerente. Assim, uma disposição desordenada dos dados não pode ser referenciada como um banco de dados

Apresentação

- Quem sou eu?

- Quem é você?

História

- Anos 60
  - Banco de dados em arquivos;
  - Modelo hierárquico e Modelo em redes;
- Anos 70
  - Edgar frank Codd (Banco de dados relacional);
  - IBM e System R (1º Sistema);
  - Dr. Peter Chen (Modelo ER)

História

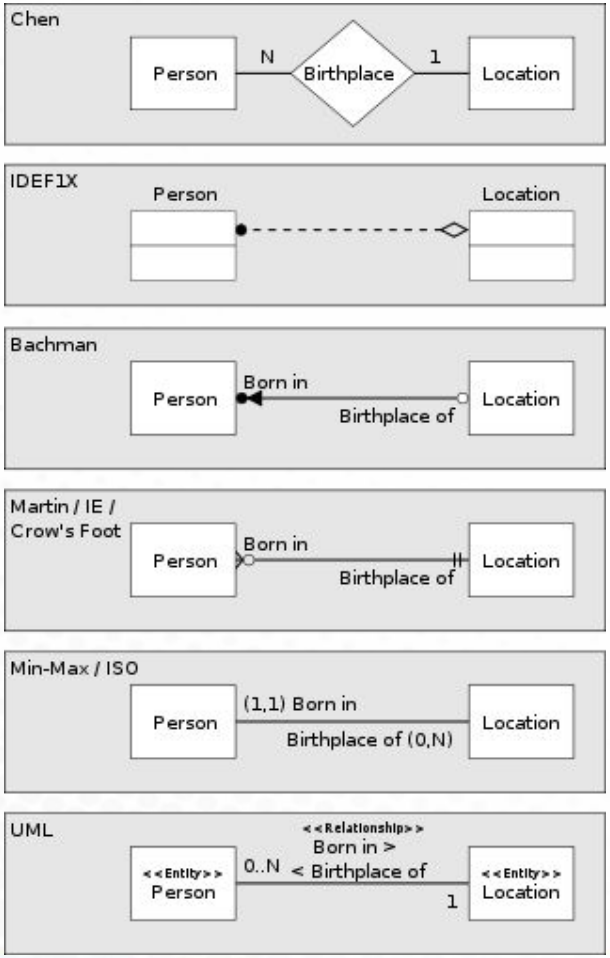
- Anos 80
  - Oracle (1º Sistema SQL);
- Anos 90
  - SGDBs;
  - Data Mining, Data Warehouse

Sistema de Banco de Dados

- Conceito
- Características
  - Abstração
  - Rapidez
  - Compartilhamento
  - Segurança
  - Facilidade do backup
  - Comunicação direta com o servidor
  - Garantir integridade
  - Tolerância a falhas

Modelo ER

- Inicialmente concebido por Peter Chen
- Várias notações diferentes da tradicional
- https://www.lucidchart.com



Objetos de um Banco de Dados

- Tabelas
- Visões
- Sequências
- Índices
- Functions/Procedures
- Esquemas

Comandos DML

- Data Manipulation Language
- Manipula dados (incluir, recuperar, remover, alterar)
  - Select
  - Insert
  - Update
  - Delete

Comandos DML

```sql
SELECT id, nome, salario, cpf FROM funcionario;

INSERT INTO funcionario(id, nome, salario, cpf)
VALUES (1, 'Tiago', 1000.00, '11111111111');

UPDATE funcionario SET nome = 'Tiago João' WHERE id = 1;

DELETE from funcionario WHERE id = 1;
```

Prática DML

- Insert / update / select / delete
- Operadores
- Ordenação (random)
- Limit e offset
- Agrupamento (max, min, sum, avg, count, having)

Criando e Restaurando

➔ Criar novo database

psql -U postgres -d postgres -c "CREATE DATABASE **${NOME_DATABASE}** WITH ENCODING='UTF8' OWNER=postgres CONNECTION LIMIT=-1;"

➔ Restaurar backup

pg_restore -U postgres --role postgres -d **$NOME_DATABASE** -v **$CAMINHO_ARQUIVO/$NOME_ARQUIVO_BACKUP**

- actor – stores actors data including first name and last name.
- film – stores film data such as title, release year, length, rating, etc.
- film_actor – stores the relationships between films and actors.
- category – stores film's categories data.
- film_category- stores the relationships between films and categories.
- store – contains the store data including manager staff and address.
- inventory – stores inventory data.
- rental – stores rental data.
- payment – stores customer's payments.
- staff – stores staff data.
- customer – stores customer data.
- address – stores address data for staff and customers
- city – stores city names.
- country – stores country names.

Atividades

1. Realizar Select (id, first_name, last_name e last_update) na tabela “actor” ordenando por “first name”
2. Realizar Insert de mais 2 atores na tabela “actor”
3. Excluir um Ator que foi recém inserido
4. Atualizar o “first name” do ator recém inserido que restou

Desafio

Realizar um select na tabela Payment (payment_id, customer_id, staff_id, renta_id, amount, payment_date) adicionado select mais uma coluna onde o amount recebe 10% de desconto

Atividades

1. Realizar Select na tabela “payment” somando a coluna “amount” e agrupando pela coluna customer_id, ordene de forma a mostrar as maiores somas primeiro
2. Realizar Select na tabela “rental” contando quanta locações foram feitas e agrupando pela coluna customer_id, ordene de forma a mostrar as menores contagens primeiro
3. Realizar Select na tabela “rental” de forma a descobrir qual cliente fez a última locação;
4. Realizar Select na tabela “rental” de forma a descobrir qual cliente fez a primeira locação;

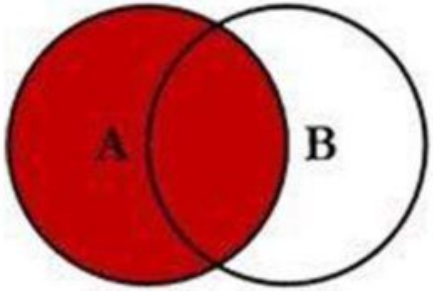
Atividades

1. Realizar Select na tabela “payment” calculando a média da coluna “amount” e agrupando pela coluna customer_id, ordene mostrando as maiores médias primeiro
2. Realizar Select na tabela “payment” somando a coluna “amount” e agrupando pela coluna customer_id, mostre somente os clientes que possuem somatório maior de 200

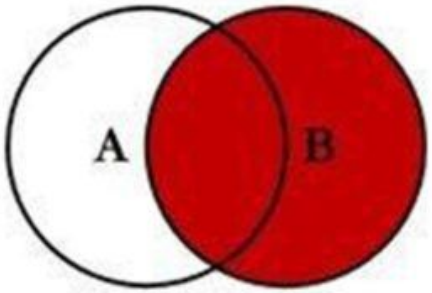
Prática DML

- JOIN (INNER, LEFT, RIGHT, OUTER)
- Função string_agg
- União, intersecção, exclusão

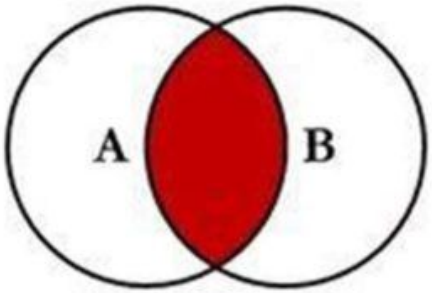
SQL JOINS



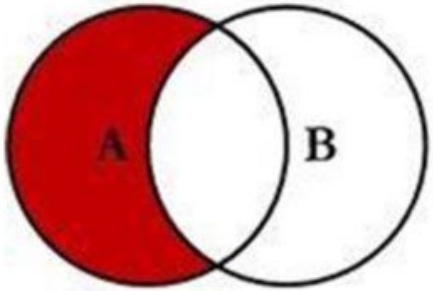

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



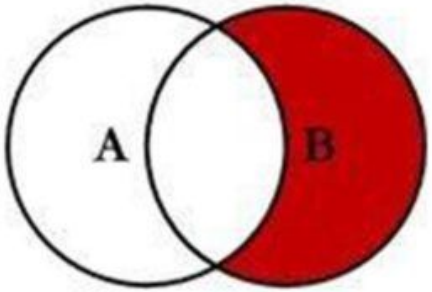

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



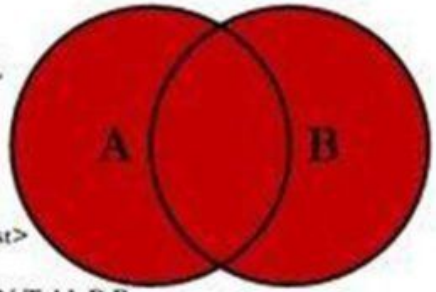

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



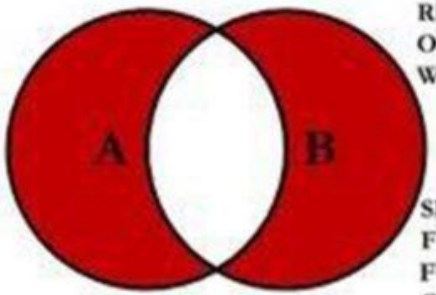

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

INNER JOIN (JOIN)

```
select r.rental_date, s.first_name || ' ' || s.last_name
from rental r
join staff s on r.staff_id = s.staff_id;
```

LEFT JOIN

```
select a.last_name, fa.film_id
from actor a
left join film_actor fa on a.actor_id = fa.actor_id
where a.actor_id = 201;
```

RIGHT JOIN

```
select l.language_id, f.film_id, f.title
 from film f
right join language l on l.language_id = f.language_id;
```

Atividades

1. Realizar Select na tabela “inventory” fazendo união com a tabela “film” e mostre o inventory_id e o título do filme
2. Realizar Select na tabela “Rental” mostrando a data do aluguel, o nome completo do customer, do staff e o título do filme
3. Realizar Select na tabela “Filme” mostrando a lista de nome dos atores, separados por “;”

Comandos DDL

- Data Definition Language
- Define estruturas
  - CREATE
  - ALTER
  - DROP

Comandos DDL

```sql
CREATE DATABASE jovemdev;

CREATE TABLE aluno(
    id INTEGER,
    nome varchar,
    PRIMARY KEY (id)
);

ALTER TABLE aluno ADD COLUMN dat_nascimento DATE;

DROP TABLE aluno;
```

Modelagem física e restrições

- Primary Key (unique + not null)
- Foreign Key
- Unique
- Check
- Not Null

Modelagem física e restrições

```
CREATE TABLE funcionario(
    id INTEGER,
    nome varchar NOT NULL,
    salario NUMERIC(8,2) CHECK (salario > 0),
    cpf varchar(11) UNIQUE,
    id_departamento INTEGER references departamento(id),
    PRIMARY KEY (id)
);
```

Modelagem física e tipos de dados numéricos

- Inteiros: Integer, smallint, bigint
- Auto-incrementáveis: Serial, bigserial
- Decimais exatos: Decimal, numeric
- Decimais inexatos: Real, Double

Modelagem física e tipos de dados de cadeia de caracteres

- Varchar ou character varying (tamanho variável com limite)
- Text (tamanho variável sem limite)
- Character ou char (tamanho fixo, preenchido com trailing spaces)

Modelagem física e tipos de dados temporais

- Timestamp
- Date
- Interval
- Timezone (deslocamento de fuso-horário, ex.: ‘-6:00’; GMT, UTC)
- Formato (YYYY-MM-DD HH24:MI:SS)

Atividades DDL

```sql
CREATE TABLE aluno_jovemdev(
    id_aluno_jovemdev serial,
    nom_aluno_jovemdev varchar(200),
    num_cpf varchar(11) NOT NULL UNIQUE,
    id_turma_jovemdev integer REFERENCES turma_jovemdev(id_turma_jovemdev),
    PRIMARY KEY (id_aluno_jovemdev)
);

DROP TABLE turma_jovemdev;
CREATE TABLE turma_jovemdev(
    id_turma_jovemdev serial,
    nom_turma_jovemdev varchar(100) NOT NULL UNIQUE,
    PRIMARY KEY (id_turma_jovemdev)
);
```

Atividades DDL

```
ALTER TABLE aluno_jovemdev ALTER COLUMN nom_aluno_jovemdev SET NOT NULL;
ALTER TABLE aluno_jovemdev ADD CONSTRAINT
    nom_aluno_jovemdev_check_size CHECK (char_length(nom_aluno_jovemdev) > 5);

ALTER TABLE aluno_jovemdev ADD COLUMN dat_nascimento date;
```

Atividades DDL

Criar uma tabela, usando as restrições:

- ➔ Primary Key (unique + not null)
- ➔ Foreign Key (será necessário criar uma segunda tabela)
- ➔ Unique
- ➔ Check
- ➔ Not Null

Comandos DTL

- Data Transaction Language
- Controla o ciclo de vida das transações
  - Begin
  - Commit
  - Rollback

Atividades DTL

Crie uma transação de delete, execute, dê rollback.

Faça uma instrução update, execute, dê commit.

Princípio ACID e as Transações

- Controle transacional e de concorrência (linhas obsoletas)
- **A**tomicidade (tudo é feito ou nada é feito)
- **C**onsistência (regras são inquebráveis)
- **I**solamento (não existem dados intermediários)
- **D**urabilidade (dados são persistidos em definitivo)

PL/SQL

A PL/SQL é uma linguagem processual projetada especificamente para incluir instruções SQL em sua sintaxe. As unidades de programa PL/SQL são compiladas pelo servidor do Oracle Database e armazenadas no banco de dados. E, em tempo de execução, tanto a PL/SQL quanto o SQL são executados no mesmo processo do servidor, o que proporciona eficiência ideal.

PL/SQL - Exemplo

```
DO $$
BEGIN
    FOR count IN 1..10 LOOP
        RAISE NOTICE 'Número %', count;
    END LOOP;
END;
$$ LANGUAGE plpgsql
```

PL/SQL - Function

```
-- DROP FUNCTION funcao_zeroUm(integer); //Como Realizar o DROP
CREATE FUNCTION funcao_zeroUm(num_inicial_p integer) RETURNS integer AS $$
DECLARE
    resultado integer := 0;
BEGIN
    FOR count IN 1..10 LOOP
        resultado := resultado + (num_inicial_p + count);
        RAISE NOTICE 'Número: %', resultado;
    END LOOP;

    RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT funcao_zeroUm(10); -- Como realizar a chamada
```

PL/SQL - Procedure

```
-- DROP FUNCTION funcao_zeroDois(integer); //Como Realizar o DROP
CREATE FUNCTION funcao_zeroDois(num_inicial_p integer) RETURNS void AS $$
DECLARE
    resultado integer := 0;
BEGIN
    FOR count IN 1..10 LOOP
        resultado := resultado + (num_inicial_p + count);
        RAISE NOTICE 'Número: %', resultado;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT funcao_zeroDois(10); -- Como realizar a chamada
```

PL/SQL - Trigger

```
-- DROP TRIGGER tg_atualiza_data ON actor;
-- DROP FUNCTION atualiza_data();
CREATE OR REPLACE FUNCTION atualiza_data() RETURNS TRIGGER AS $$
    BEGIN
        IF (TG_OP = 'UPDATE') THEN
            NEW.last_update = now();
            RETURN NEW;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_atualiza_data AFTER UPDATE
    ON actor FOR EACH ROW
        EXECUTE PROCEDURE atualiza_data();
```

Atividades PL/SQL

1 - Crie uma function para retornar o código do produto concatenado com seu nome, no seguinte formato: “(cod) nome_produto”, ao receber por parâmetro o cod_reduzido deste produto;

2 - Crie uma procedure que cria uma coluna na tabela cadprodu do tipo varchar, com o nome de cod_nom_produto. Depois preencha essa coluna com o código do produto, concatenado com seu nome, no seguinte formato: “(cod) nome_produto”;

3 - Crie uma trigger para que sempre que for inserido um produto novo, preencha a coluna criada na tarefa anterior com o código do produto concatenado com seu nome, no seguinte formato: “(cod) nome_produto”;

Dúvidas?