BANCO DE DADOS II

Professor Alex Lemos

BASES TECNOLÓGICAS

- Revisão das regras de conversão do modelo conceitual para o modelo relacional(MER) / lógico.
- Apresentação da linguagem SQL: histórico, ANSI SQL.
- Introdução ao SGBD Mysql: histórico e visão geral.
- Criação de banco de dados: create database e drop database.
- Interface de comando.
- Comandos da ferramenta x comandos SQL.
- Subconjuntos da linguagem SQL: DDL, DML.

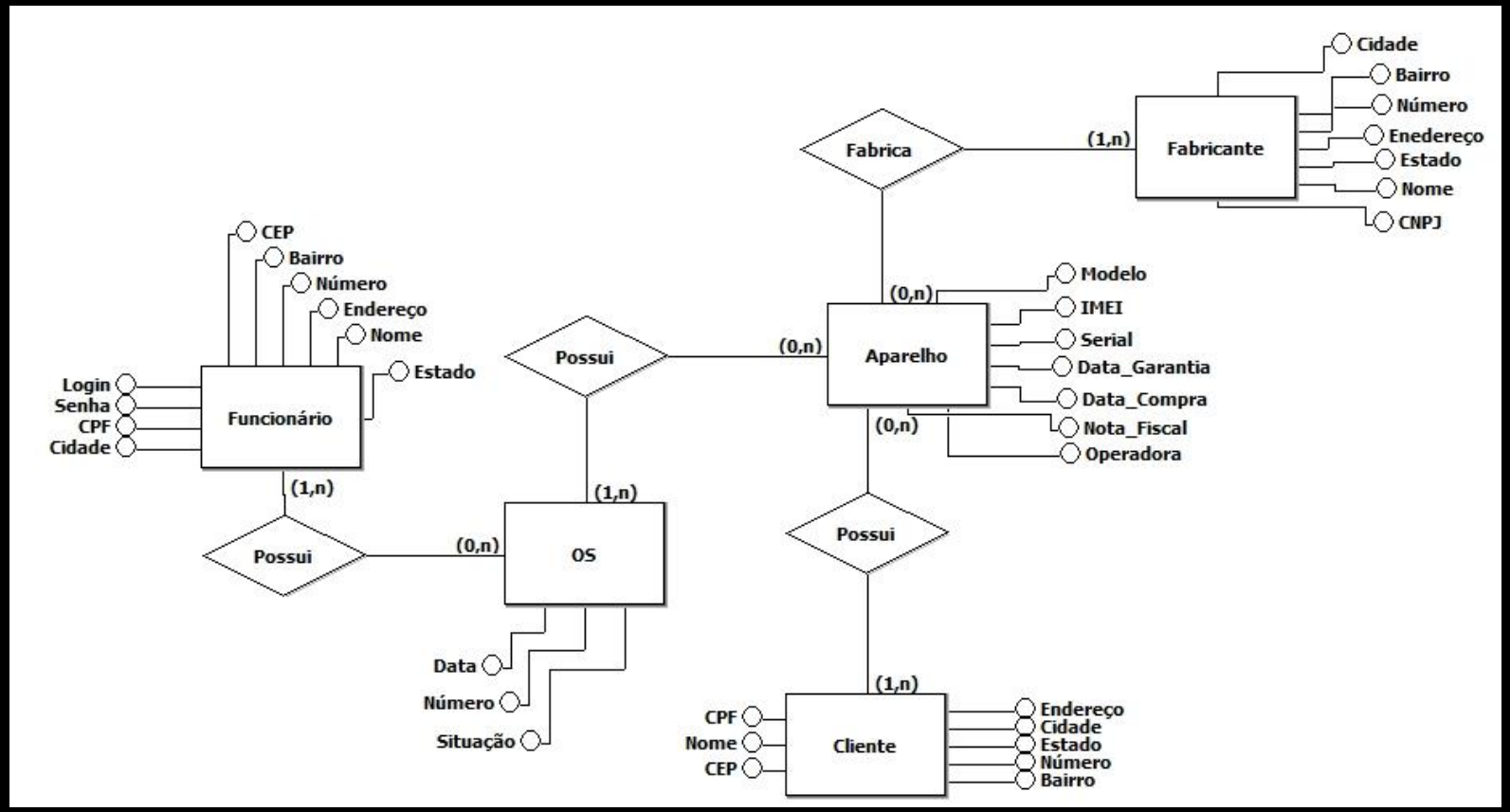
BASES TECNOLÓGICAS

- DDL - Linguagem de definição de dados: tipos de dados; criar, apagar e alterar tabelas:
- constraints: conceitos de chave primária (UK, CK e NN); conceitos de integridade referencial – constraint de foreign key.
- DML - Linguagem de manipulação de dados: inserir, apagar e alterar dados; comandos de transação:
- Funções (agrupamento, numéricas, caracteres, datas) funções de agregação:
  - max, min, sum, count, avg
- Inner join, outer join e self join
- Banco de Dados de Apoio: MySQL
- Ferramenta de Gráfica

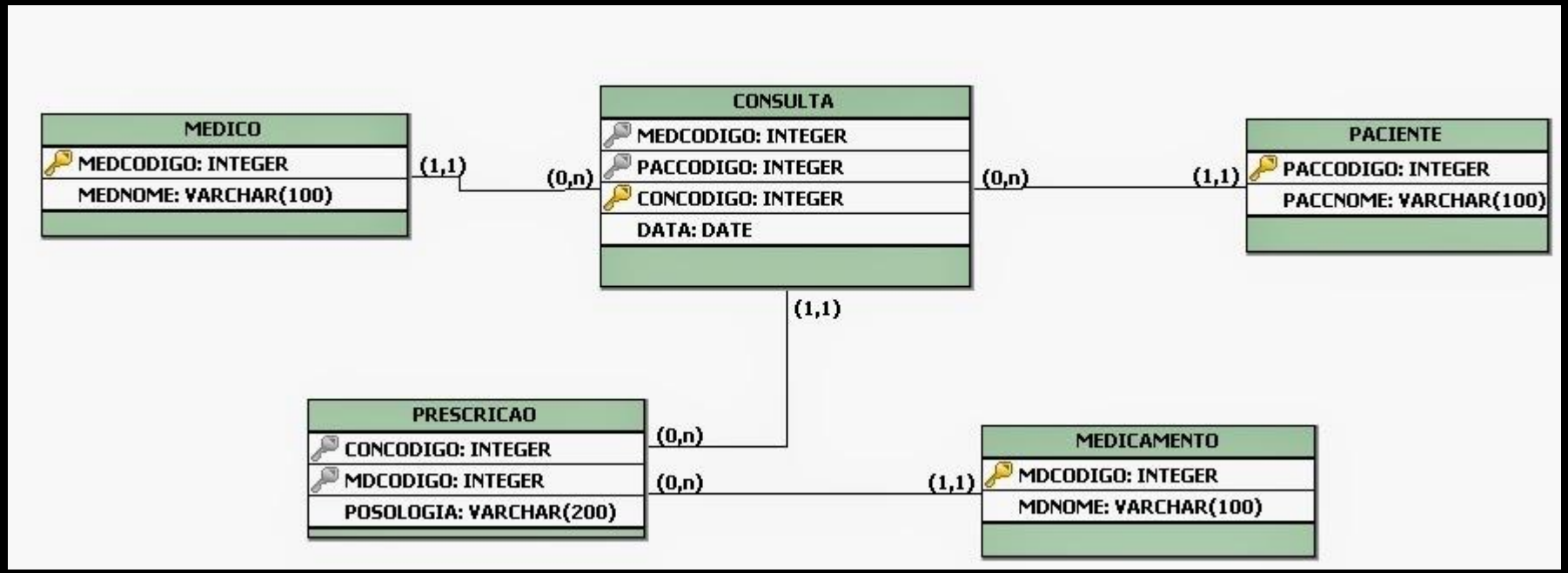
BANCO DE DADOS II

Professor Alex Lemos

MODELO CONCEITUAL



MODELO LÓGICO



MODELO FÍSICO

```
CREATE TABLE my_contacts
(
contact_id INT IDENTITY NOT NULL,
last_name varchar(30) default NULL,
first_name varchar(20) default NULL,
email varchar(50) default NULL,
gender char(1) default NULL,
birthday smalldatetime default NULL,
profession varchar(50) default NULL,
location varchar(50) default NULL,
status varchar(20) default NULL,
interests varchar(100) default NULL,
seeking varchar(100) default NULL,
PRIMARY KEY (contact_id)
)
```

O QUE É SQL?

- **Structured Query Language**, ou **Linguagem de Consulta Estruturada** ou **SQL**, é uma linguagem de pesquisa declarativa para banco de dados relacional.
- A linguagem de consulta estruturada SQL foi desenvolvida primeiramente pela empresa IBM (Internacional Business Machine) e apresentada em sua primeira versão em 1974, com o nome Structured English QUEry Language(SEQUEL). A linguagem de consulta SEQUEL foi desenvolvida pelo Ph.D. Donald D. Chamberlin, e outros profissionais da IBM.

SQL (PRINCIPAIS CARACTERÍSTICAS)

- A linguagem SQL é um grande padrão de banco de dados.
- Por ser uma linguagem declarativa (não procedural), uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele.
- Apesar de ser originalmente criada pela IBM, muitos desenvolvedores foram criando "dialetos" para ela. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem.
- Em 1986/87 a linguagem SQL foi padronizada pela ANSI e ISO sendo revisada nos anos de 1992, 1999 e 2003.
- Normalmente a linguagem pode ser aportada de plataforma para plataforma sem mudanças significativas em sua estrutura.

ESTRUTURA DA LINGUAGEM SQL

- A linguagem SQL é dividida em subconjuntos de acordo com as operações que se deseja efetuar sobre um banco de dados.
- Os principais subconjuntos são:
- **DDL** - Data Definition Language (Linguagem de Definição de Dados)
- Principais comandos: CREATE, ALTER e DROP
- **DML** - Data Manipulation Language (Linguagem de Manipulação de Dados)
- Principais comandos: SELECT, INSERT, UPDATE, DELETE, TRUNCATE e outros.
- **DCL** - Data Control Language (Linguagem de Controle de Dados)
- Principais comandos: GRANT, REVOKE e SET.

DDL LINGUAGEM DE DEFINIÇÃO DE DADOS

- O conjunto de comandos da linguagem DDL é usado para a definição das estruturas de dados, fornecendo as instruções que permitem a criação, modificação e remoção de objetos de banco de dados (base de dados, esquemas, tabelas, índices etc.).
- A maioria dos bancos de dados comerciais tem extensões proprietárias no DDL.
- Os comandos básicos da DDL são:
- CREATE: cria um objeto (uma Tabela, por exemplo) dentro da base de dados.
- DROP: apaga um objeto do banco de dados.
- ALTER: permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente.

DML LINGUAGEM DE MANIPULAÇÃO DE DADOS

- É o grupo de comandos dentro da linguagem SQL utilizado para a recuperação, inclusão, remoção e modificação de informações em bancos de dados.
- Os comandos básicos da DML são:
- **SELECT**: permite ao usuário especificar uma consulta ("query") como uma descrição do resultado desejado.
- **INSERT**: é usada para inserir um registro (formalmente uma tupla) a uma tabela existente.
- **UPDATE**: para mudar os valores de dados em uma ou mais linhas da tabela existente.
- **DELETE**: permite remover linhas existentes de uma tabela.
- **TRUNCATE**: remove rapidamente todas as linhas da tabela, esvaziando-a.
- **COMMIT**: efetiva a transação atualmente executada.
- **ROLLBACK**: desfaz a transação corrente, fazendo com que todas as modificações realizadas pela transação sejam rejeitadas.

DCL LINGUAGEM DE CONTROLE DE DADOS

- É o grupo de comandos que permitem ao administrador de banco de dados gerenciar os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.
- Alguns exemplos de comandos DCL são:
- **GRANT**: concede privilégios a um ou mais usuários para acessar ou realizar determinadas operações em um objetos de dados.
- **REVOKE**: revoga (remove) ou restringe a capacidade de um usuário de executar operações.
- **SET**: Define parâmetros em tempo de execução, como por exemplo, o tipo de codificação do cliente e o estilo de representação de data e hora.
- **LOCK**: Bloqueia explicitamente uma tabela fazendo o controle de acessos concorrente.

MYSQL

- O programa MySQL é um sistema gerenciador de banco de dados relacional que utiliza a linguagem de consulta estruturada SQL como interface de acesso e extração de informações do banco de dados em uso. O MySQL é um dos sistema gerenciador de banco de dados mais populares e usados no mundo. É rápido, multitarefas e multiusuários.

HISTÓRIA

- O MySQL originou-se na Suécia em e as ideias para seu desenvolvimento surgiram em 1979, de dois suecos David Axmark, Allan Larsson e um finlandês Michael Widenius.
- Em 1979 uma pequena empresa chamada TcX, que desenvolveu uma ferramenta de banco de dados (não SQL) para o gerenciamento de grandes tabelas denominada Unireg, utilizada para a geração de relatórios.

HISTÓRIA

- Em 1994, foi iniciado o desenvolvimento de um sistema gerenciador de banco de dados de código aberto baseado nos programas Unireg e mSQL.
- Em 1995, foi desenvolvido o MySQL pela empresa MySQL AB e o lançamento da primeira versão oficial ocorreu em 1996.
- Em janeiro de 2008 o programa MySQL foi comprado pela SUN Microsystems pelo valor de 1 bilhões de dólares.
- Em 20/04/2009 a Oracle comprou a SUN Microsystems pelo valor de 7,4 bilhões de dólares.
- Quando a Oracle comprou a Sun, o Michael Widenius, fez um fork do MySQL, chamado de MariaDB.

BANCO DE DADOS II

Professor Alex Lemos

DML LINGUAGEM DE MANIPULAÇÃO DE DADOS

- É o grupo de comandos dentro da linguagem SQL utilizado para a recuperação, inclusão, remoção e modificação de informações em bancos de dados.
- Os comandos básicos da DML são:
- **SELECT**: permite ao usuário especificar uma consulta ("query") como uma descrição do resultado desejado.
- **INSERT**: é usada para inserir um registro em uma tabela existente.
- **UPDATE**: para mudar os valores de dados em uma ou mais linhas da tabela existente.
- **DELETE**: permite remover linhas existentes em uma tabela.
- **TRUNCATE**: remove rapidamente todas as linhas da tabela, esvaziando-a.
- **COMMIT**: efetiva a transação atualmente executada.
- **ROLLBACK**: desfaz a transação corrente, fazendo com que todas as modificações realizadas pela transação sejam rejeitadas.

CRUD

- Comandos básicos para a manipulação de dados:
- Create – INSERT: comando utilizado para inserir dado na tabela.
- Read – SELECT: comando utilizado para pesquisar dados na tabela.
- Update – UPDATE: comando utilizado para alterar dados na tabela.
- Delete – DELETE: comando utilizado para apagar dados na tabela.

INSERT – INSERÇÃO DE REGISTROS

- A partir do momento em que uma tabela está pronta (criada fisicamente no banco de dados), ela já pode receber entrada de dados que podem ser enviadas de duas formas: uma direta e outra indireta. A forma direta usa instrução INSERT INTO e a indireta via instrução LOAD DATA.

- Para uma entrada de dados direta, a instrução INSERT INTO possui a seguinte sintaxe:

```
INSERT INTO <tabela> ( campo1, campo2, campo3) VALUES (valor1, valor2, valor3)
```

EXEMPLO TABELA FUNCIONÁRIO

```
CREATE TABLE FUNCIONARIO (
      COD_FUN    INT(11)     NOT NULL,
      NOME_FUN   VARCHAR(80) NOT NULL,
      COD_DEPTO  INT(11)     NOT NULL,
      COD_FUNCAO INT(11)     NOT NULL,
      PRIMARY KEY(COD_FUN),
      FOREIGN KEY(COD_DEPTO)  REFERENCES DEPARTAMENTO(COD_DEPTO),
      FOREIGN KEY(COD_FUNCAO) REFERENCES FUNCAO(COD_FUNCAO)
);
```

EXEMPLO DE COMANDO INSERT

```
INSERT INTO FUNCIONARIO (
      COD_FUN,
      NOME_FUN,
      COD_DEPTO,
      COD_FUNCAO
) VALUES (
      100,
      ‘CARLOS ALBERTO SANTOS’,
      1001,
      2500
);
```

INSERT LOAD DATA

- Uma forma de agilizar um pouco mais esse processo de entrada de registros é usar o recurso da forma indireta de entrada de dados com a instrução **LOAD DATA**.
- Para usar esse recurso é necessário criar um arquivo em formato de texto puro (.txt), Esse arquivo deve ser formado pelos dados a serem inseridos na mesma ordem dos campos da tabela. É necessário também que os dados da mesma linha estejam separados por uma marca de tabulação, a tecla Tab por exemplo.
- Os dados devem ser gravados no arquivo de nome dados.txt, separados por Tab.

DADOS DO ARQUIVO

1	Maria Ana Sousa	10	100
2	João Santos Silva	10	200
3	José Gomes	10	100
4	Marco Costa Amorim	10	200

O arquivo dados.txt deve ser gravado na pasta BIN dentro da instalação do MySQL, esta pasta pode variar de acordo com a versão do MySQL.

LOAD DATA INFILE "dados.txt" **INTO TABLE** FUNCIONARIO;

SELECT – CONSULTA DE REGISTROS

- A obtenção de uma listagem de registros armazenados em um tabela é uma das tarefas mais requisitadas e executadas nos programas de gerenciamento de banco de dados.
- Para extrair listagens de registros de uma tabela, a linguagem SQL disponibiliza o comando **SELECT** que anexado a alguns parâmetros fornece a mais importante ferramenta de instrução a um determinado banco de dados. A sintaxe básica para obter uma instrução de consulta tem a seguinte estrutura:

```
SELECT<campos> FROM<tabela>
SELECT<campos> FROM<tabela> WHERE <condição>
```

EXEMPLOS DE COMANDO SELECT

```
SELECT COD_FUN, NOME_FUN, COD_DEPTO, COD_FUNCAO FROM FUNCIONARIO;
```

Pesquisa a listagem de dados, com todas as colunas(campos) da tabela funcionario.

```
SELECT * FROM FUNCIONARIO;
```

Pesquisa a listagem de dados, com todas as colunas(campos) da tabela funcionario.

```
SELECT COD_FUN, NOME_FUN FROM FUNCIONARIO;
```

Pesquisa a listagem de dados, com as colunas código e nome da tabela funcionario.

```
SELECT COD_FUN, NOME_FUN, COD_DEPTO, COD_FUNCAO FROM FUNCIONARIO WHERE
COD_FUN = 1;
```

Pesquisa a listagem de dados, com as colunas todas as colunas com uma condição, um filtro de pesquisa onde Código do funcionário é igual a 1.

UPDATE – ALTERAÇÃO DE REGISTROS

- No processo de manutenção de registros em banco de dados é necessário periodicamente atualizar dados e informações existentes. O processo de alteração de registros é feito pela instrução **UPDATE** em conjunto com o comando **SET** e o comando **WHERE**.

```
UPDATE <tabela> SET <campo> = <novo valor> WHERE <condição>;
```

DELETE – REMOÇÃO DE REGISTROS

- Dentre as várias atividades de manutenção de registros, existe a possibilidade de remover registros que não sejam mais necessários. É preciso utilizar a instrução **DELETE FROM** que possui a seguinte sintaxe:

- `DELETE FROM funcionário WHERE cod_fun = 1;`

BANCO DE DADOS II

Professor Alex Lemos

INTEGRIDADE REFERENCIAL

- Chave Primária / Primary Key [PK]:
  - A chave primária de uma tabela é o atributo que identifica cada registro de uma tabela.

  - Os critérios para escolher a chave primária são:
    - O conteúdo da coluna ou do conjunto de colunas deve sempre existir, em outras palavras deve possuir a constraint semântica NOT NULL.
    - O valor da chave não deve ser alterado.

  - A chave primária tem as seguintes características:
    - Apenas uma chave primária por tabela.
    - Pode ser composta por mais de uma coluna.
    - Obrigatoriamente ser NOT NULL (não nulo).

INTEGRIDADE REFERENCIAL

- Chave Estrangeira / Foreign Key [FK]:
  - A chave estrangeira se refere ou é relacionada com alguma chave primaria de outra tabela, podendo ser inclusive da mesma tabela. É a concretização do relacionamento proposto no modelo entidade-relacionamento (MER).

- A chave estrangeira tem as seguintes características:
  - Uma tabela pode possuir zero, uma ou diversas chaves estrangeiras.
  - Pode ser composta de uma ou mais colunas de acordo com a chave primaria que está relacionada a ela (tipo de dado e tamanho).
  - Pode conter a clausula NULL.

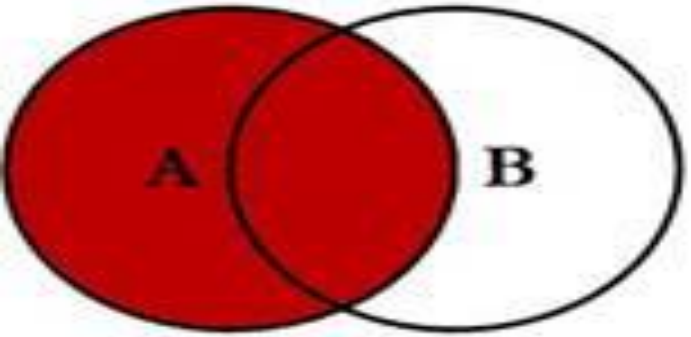
INTEGRIDADE REFERENCIAL

- Integridade referencial se da por valor:

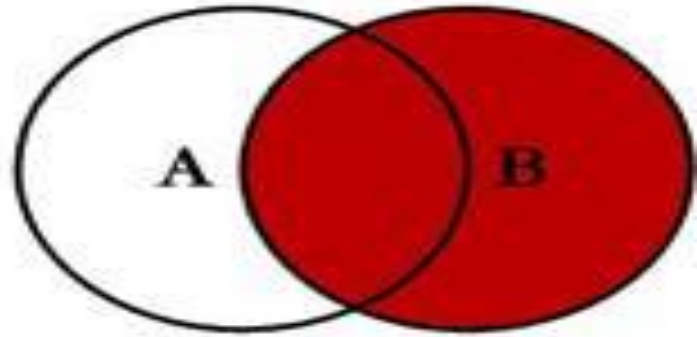
CLIENTE POSSUI O ID_CLIENTE = 1
PEDIDO POSSUI O ID_CLIENTE = 1

CLIENTE		PEDIDO		
ID	NOME	ID	VALOR	ID_CLIENTE
1	Maria	100	50,00	1
2	José	200	150,00	2
3	Ana	300	200,00	3

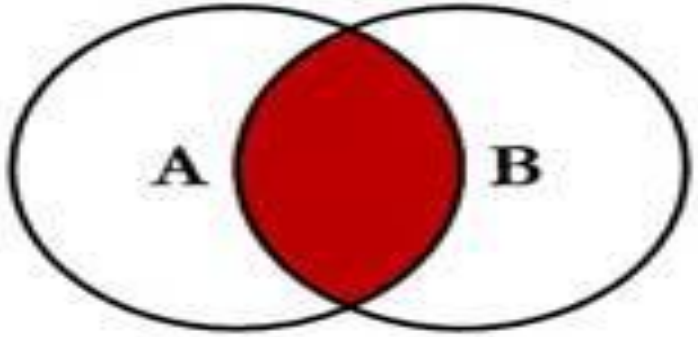
SQL JOINS



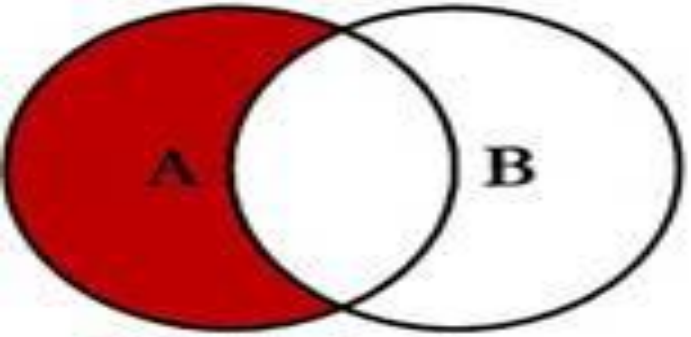

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



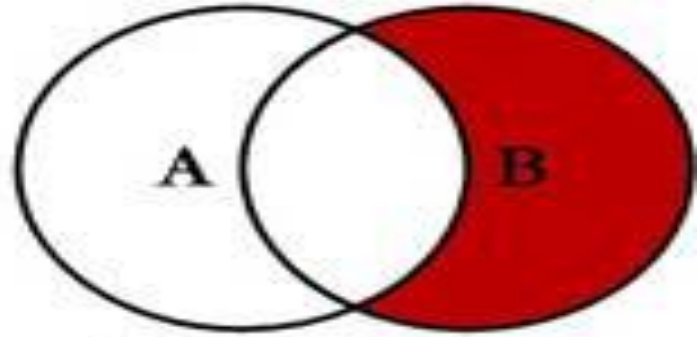

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```




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

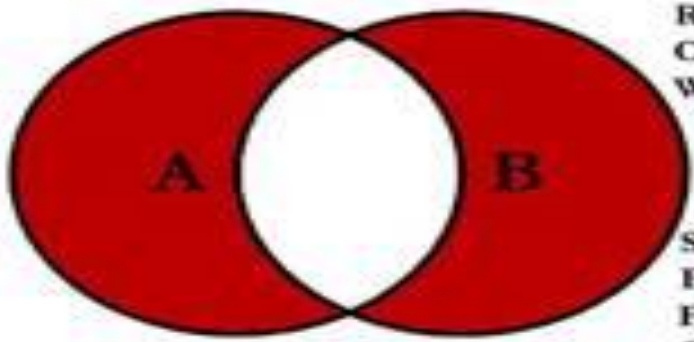



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
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
```

```
FROM nome_da_primeira_tabela tipo_de_associação nome_da_segunda_tabela [ON (condição_de_associação) ]
```

JOIN ENTRE TABELAS

- JOIN
- A cláusula JOIN permite que os dados de várias tabelas sejam combinados com base na relação existente entre elas. Por meio dessa cláusula, os dados de uma tabela são usados para selecionar os dados pertencentes à outra tabela.
- Com a cláusula JOIN, podemos especificar quais colunas das tabelas serão associadas. Para isso, será preciso definir uma chave estrangeira de uma tabela e a chave relacionada em outra tabela.
- Os valores pertencentes às colunas das tabelas associadas podem ser comparados entre si por meio de um operador lógico definido pela cláusula JOIN e usada pelo operador ON, como o sinal de igual (=).

INNER JOIN ENTRE TABELAS

- Exemplo
- FROM nome_da_primeira_tabela tipo_de_associação nome_da_segunda_tabela

  [ON (condição_de_associação)]

Onde:

  - tipo_de_associação: Permite identificar uma das seguintes associações: outer join, inner join e cross join (que serão descritas em detalhes mais adiante);
  - condição_de_associação: Define um critério para avaliar duas linhas de dados que já estão associadas.
- A forma mais indicada para a especificação de associações é a cláusula FROM, que permite que as condições JOIN sejam identificadas em relação às condições de busca referenciadas na cláusula WHERE.
- Vejamos um exemplo simples da associação entre tabelas:

```
SELECT * FROM CLIENTE
        INNER JOIN PEDIDO ON CLIENTE.ID_CLIENTE = PEDIDO.ID_CLIENTE;
```

INNER JOIN ENTRE TABELAS

- Exemplo usando a cláusula WHERE para especificar uma condição de associação:

```
SELECT CLIENTE.ID_CLIENTE, CLIENTE.NOME,
       PEDIDO.DESCRICAO, PEDIDO.VALOR
  FROM CLIENTE, PEDIDO
 WHERE CLIENTE.ID_CLIENTE = PEDIDO.ID_PEDIDO
   AND PEDIDO.VALOR > 50.00
```

LEFT OUTER JOIN ENTRE TABELAS

- LEFT OUTER
- A cláusula LEFT JOIN ou LEFT OUTER JOIN permite obter não apenas os dados relacionados de duas tabelas, mas também os dados não relacionados encontrados na tabela à esquerda da cláusula JOIN. Caso não existam dados relacionados entre as tabelas à esquerda e a direita do JOIN, os valores resultantes de todas as colunas da lista de seleção da tabela à direita serão nulos.

```
SELECT C.NOME_CARGO, F.NOME_FUNCIONARIO, F.SALARIO_FUNCIONARIO
  FROM CARGO AS C LEFT OUTER JOIN FUNCIONARIO AS F
       ON C.ID_CARGO = F.ID_CARGO
```

RIGHT OUTER JOIN ENTRE TABELAS

- RIGHT JOIN
- Ao contrário do LEFT JOIN, a cláusula RIGHT JOIN ou RIGHT OUTER JOIN retorna todos os dados encontrados na tabela à direita de JOIN. Caso não existam dados associados entre as tabelas à esquerda e à direita de JOIN, serão retornados valores nulos.
- Suponhamos que a posição das tabelas usadas nos exemplos anteriores foi trocada. Se mesmo assim desejamos obter o mesmo resultado obtido anteriormente, podemos usar a cláusula RIGHT JOIN, assim iremos conseguir tanto os dados relacionados como os não relacionados disponíveis na tabela à direita da cláusula JOIN.

RIGHT OUTER JOIN ENTRE TABELAS

- RIGHT JOIN
- Exemplo

```
SELECT C.NOME_CARGO, F.NOME_FUNCIONARIO, F.SALARIO_FUNCIONARIO
  FROM FUNCIONARIO AS F
       RIGHT OUTER JOIN CARGO AS C ON F.ID_CARGO = C.ID_CARGO
```

CROSS JOIN ENTRE TABELAS

- CROSS JOIN
- Todos os dados da tabela à esquerda de JOIN são cruzados com os dados da tabela à direita de JOIN por meio do CROSS JOIN, também conhecido como produto cartesiano. É possível cruzarmos informações de duas ou mais tabelas.
- Para facilitar a compreensão a respeito desse tipo de associação, vamos usar as tabelas de cargo e funcionario como exemplo. Caso a intenção seja exibir os dados de modo que todos os funcionários tenham todos os cargos e vice-versa. Para isso, devemos usar o CROSS JOIN, como no exemplo a seguir:

```
SELECT C.NOME_CARGO, F.NOME_FUNCIONARIO, F.SALARIO_FUNCIONARIO
  FROM CARGO AS C
       CROSS JOIN FUNCIONARIO AS F;
```