Sistemas de Gerenciamento de Banco de Dados

Diogo Bortolini

Senac

Estruturas (Conjuntos) de Linguagens SQL

DDL – Linguagem de Definição de Dados: criar novas tabelas e alterar ou excluir elementos associados. Comandos: CREATE – ALTER - DROP

DML – Linguagem de Manipulação de Dados: alterações dos dados no registro. Comandos: INSERT - UPDATE - DELETE

DCL – Linguagem de Controle de Dados: autorização de acesso ou manipulação dos dados. Comandos: GRANT - REVOKE

DTL – Linguagem de Transação de Dados: controlam as transações. Comandos: BEGIN WORK (START TRANSACTION) – COMMIT - ROLLBACK

DQL – Linguagem de Consulta de Dados: consulta dos dados. Comando: SELECT

DQL

DDL – Linguagem de Definição de Dados: criar novas tabelas e alterar ou excluir elementos associados. Comandos: CREATE – ALTER - DROP

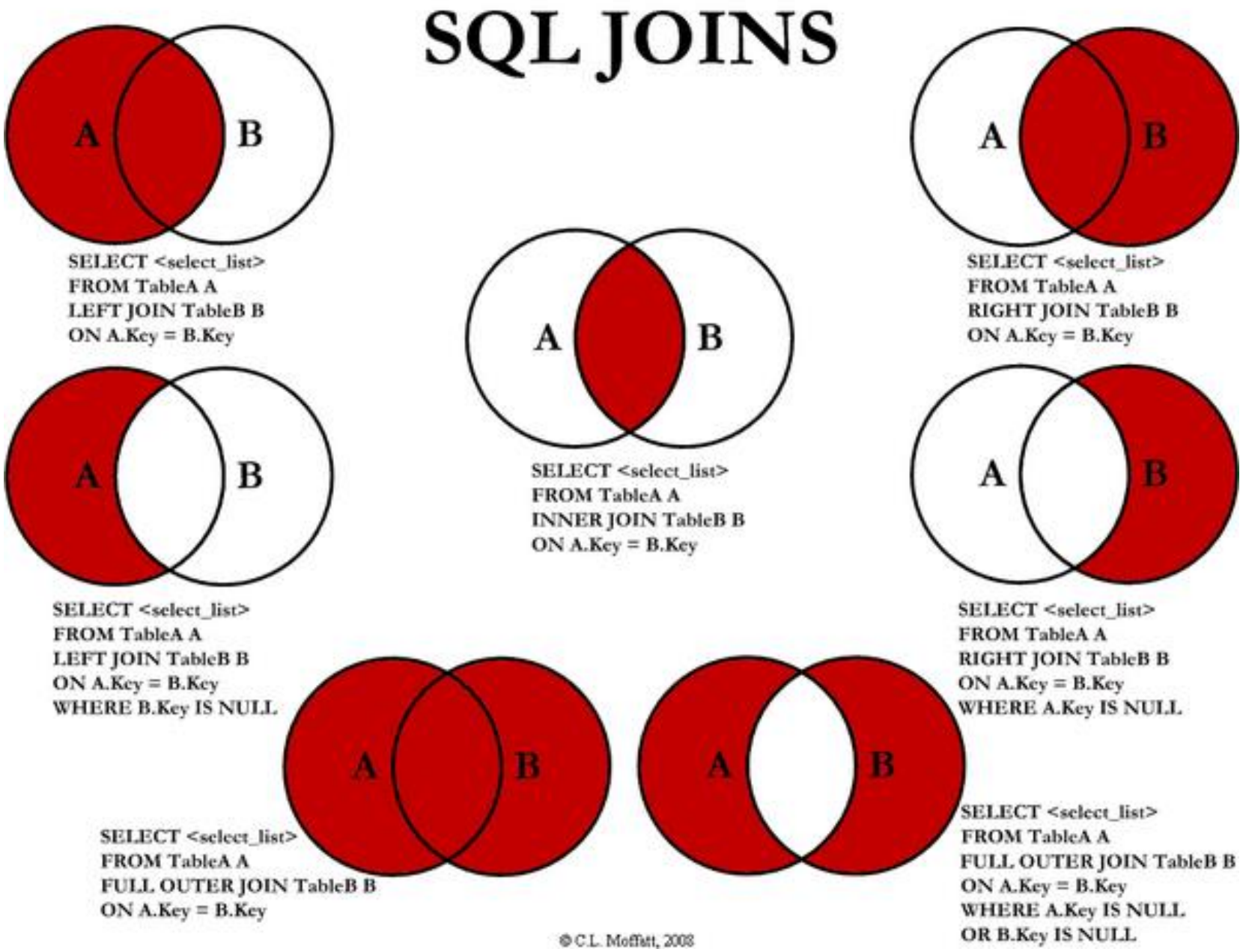
DML – Linguagem de Manipulação de Dados: alterações dos dados no registro. Comandos: INSERT - UPDATE - DELETE

DCL – Linguagem de Controle de Dados: autorização de acesso ou manipulação dos dados. Comandos: GRANT - REVOKE

DTL – Linguagem de Transação de Dados: controlam as transações. Comandos: BEGIN WORK (START TRANSACTION) – COMMIT - ROLLBACK

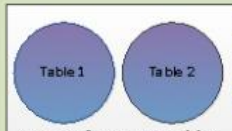
DQL – Linguagem de Consulta de Dados: consulta dos dados.
Comando: SELECT

Junções


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT 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
RIGHT 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
© C.L. Moffatt, 2008

MySQL JOIN Types

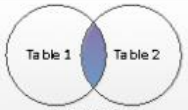
Created by Steve Stedman




```
SELECT *
  FROM Table1;

SELECT *
  FROM Table2;
```

SELECT from two tables




```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id;
```

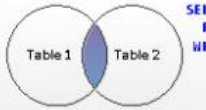
INNER JOIN


```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id;
```

LEFT OUTER JOIN

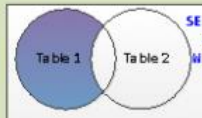

```
SELECT *
  FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
  ON t1.fk = t2.id;
```

RIGHT OUTER JOIN



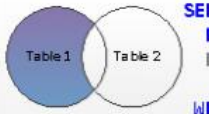

```
SELECT *
  FROM Table1 t1
WHERE EXISTS (SELECT 1
                FROM Table2 t2
               WHERE t1.fk = t2.id
              );
```

SEMI JOIN – Similar to INNER JOIN, with less duplication.



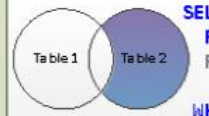

```
SELECT *
  FROM Table1 t1
WHERE NOT EXISTS (SELECT 1
                    FROM Table2 t2
                   WHERE t1.fk = t2.id
                  );
```

ANTI SEMI JOIN



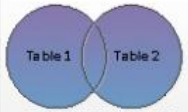

```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
WHERE t2.id is null;
```

LEFT OUTER JOIN with exclusion



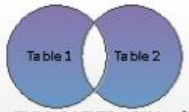

```
SELECT *
  FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
WHERE t1.fk is null;
```

RIGHT OUTER JOIN with exclusion



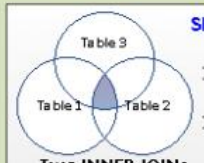

```
SELECT * FROM Table1 t1
  LEFT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
UNION
SELECT * FROM Table1 t1
  RIGHT OUTER JOIN Table2 t2
    ON t1.fk = t2.id;
```

FULL OUTER JOIN



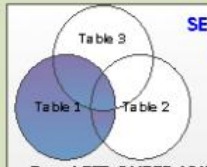

```
SELECT * FROM Table1 t1
  LEFT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t2.id IS NOT NULL
UNION
SELECT * FROM Table1 t1
 RIGHT OUTER JOIN Table2 t2
    ON t1.fk = t2.id
 WHERE t1.ID IS NOT NULL;
```

FULL OUTER JOIN with exclusion



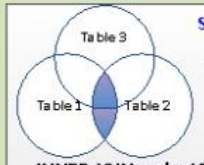

```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id
INNER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

Two INNER JOINs




```
SELECT *
  FROM Table1 t1
LEFT OUTER JOIN Table2 t2
  ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

Two LEFT OUTER JOINS




```
SELECT *
  FROM Table1 t1
INNER JOIN Table2 t2
  ON t1.fk = t2.id
LEFT OUTER JOIN Table3 t3
  ON t1.fk_table3 = t3.id;
```

INNER JOIN and a LEFT OUTER JOIN

Created By Steve Stedman http://SteveStedman.com Twitter @SqlEmt

Junções

INNER JOIN ou JOIN ou CROSS JOIN – Junção que exibe a intersecção das tabelas envolvidas, ou seja, exibe os dados que estão presentes em todas as tabelas envolvidas.

LEFT OUTER JOIN ou LEFT JOIN – Junção que exibe todos os registros da tabela mencionada à esquerda e os existentes na tabela da direita.

RIGHT OUTER JOIN ou RIGHT JOIN – Junção que exibe todos os registros da tabela mencionada à direita e os existentes na tabela da esquerda.

Junções

SELECT colunas **FROM** tabela1 **INNER JOIN** tabela2 **ON** tabela1.coluna=tabela2.coluna;

```
SELECT cidade, pais FROM pais JOIN cidade ON pais.pais_id=cidade.pais_id;

SELECT cidade, pais FROM pais JOIN cidade using(pais_id);
```

Junções

SELECT colunas **FROM** tabela1 **LEFT JOIN** tabela2 **ON** tabela1.coluna=tabela2.coluna;
SELECT colunas **FROM** tabela1 **RIGHT JOIN** tabela2 **ON** tabela1.coluna=tabela2.coluna;

```
SELECT c.cliente_id, c.primeiro_nome, c.ultimo_nome, a.ator_id, a.primeiro_nome, a.ultimo_nome
FROM cliente c LEFT JOIN ator a ON c.ultimo_nome = a.ultimo_nome
ORDER BY c.ultimo_nome;
```

```
SELECT c.cliente_id, c.primeiro_nome, c.ultimo_nome, a.ator_id, a.primeiro_nome, a.ultimo_nome
FROM cliente c RIGHT JOIN ator a ON c.ultimo_nome = a.ultimo_nome
ORDER BY c.ultimo nome;
```

ON vs USING

SELECT colunas **FROM** tabela1 **INNER JOIN** tabela2 **ON** tabela1.coluna=tabela2.coluna;

De uso geral, permite descrever quais colunas fazem parte da junção

~~Error Code: 1052. Column 'country_id' in on clause is ambiguous~~

SELECT colunas **FROM** tabela1 **INNER JOIN** tabela2 **USING**(coluna) ;

Utilizado quando a coluna que faz parte da junção tem o mesmo nome em todas as tabelas envolvidas

USANDO ALIASES (Apelidos)

```sql
SELECT cidade, pais FROM pais p JOIN cidade c ON p.pais_id=c.pais_id;
```

USANDO GROUP BY

```sql
SELECT pais, COUNT(cidade) FROM pais JOIN cidade ON pais.pais_id=cidade.pais_id GROUP BY pais;
```

```sql
SELECT pais, COUNT(cidade) FROM pais JOIN cidade ON pais.pais_id=cidade.pais_id GROUP BY pais HAVING COUNT(cidade) > 2;
```

Referências

- RODRIGUEZ, F. T. **Um pouco sobre a história dos bancos de dados**. Save Point 2017.
- GOMES, Daniella. **Modelagem de dados: 1:N ou N:N?** Disponível

  em: https://www.devmedia.com.br/modelagem-1-n-ou-n-n/38894
- FERREIRA, Nickerson Fonseca. **Normalização SQL.** IFRN, 2019
- DATE, C. J. **Introdução a sistemas de bancos de dados.** Rio de Janeiro: Campus, 2004.
- RAMAKRISHNAN, Raghu. **Sistemas de gerenciamento de banco de dados.** São Paulo: McGraw Hill, 2009.

Referências

- MILANI, ANDRÉ. C**onstruindo Aplicações Web com PHP e Mysql.** Novatec, 2010.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados.** Rio de Janeiro: Elsevier, 2006.
- MACHADO, Felipe Nery R.; ABREU, M. **Projeto de Banco de Dados: Uma visão prática.** 15ª ed. São Paulo: Érica, 2008.
- https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html