Linguagem SQL

BCD29008 – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Campus São José

Licenciamento

Sumário

Linguagem de Consulta Estruturada

***Structured Query Language* – SQL**

- Baseada em álgebra relacional, permite a definição, manipulação e controle de acesso aos em um banco de dados relacional
  - Desenvolvida dentro da IBM na década de 70, sendo hoje a mais usada em SGBD relacional
  - A linguagem não é sensível a caixa (alta ou baixa)
- Padronizada pela ANSI e ISO, porém um código SQL geralmente não é portável
  - SQL-86 foi o primeiro padrão e o SQL:2016 é o último publicado

Nesses slides serão apresentados exemplos da linguagem para os SGBD: SQLite e MySQL

Linguagem SQL

- **DDL – Data Definition Language**
  - Especificação do esquema relacional, indicando restrições
  - Cria, altera, exclui tabelas
- **DML – Data Manipulation Language**
  - Consulta, insere, modifica e exclui tuplas das tabelas
- **DCL – Data Control Language**
  - Controle de acesso e manipulação sobre os dados
- **DTL – Data Transaction Language**
  - Para especificar início e término de uma transação

SQLite

O que é o SQLite

SGBD transacional autocontido e projetado para ser embarcado em outras aplicações sem a necessidade de configuração

- Biblioteca tem menos que 750KiB
- Todo o banco de dados em um único arquivo
- Não possui processo servidor separado
- Disponível para diferentes sistemas operacionais
- Possui tabelas, índices, gatilhos, visões, funções e subconsultas
- Tabela pode ter até 32mil colunas e número ilimitado de linhas

Onde usar ou não usar o SQLite?

"Não é um substituto para o Oracle, mas sim para o `fopen()`"[1]

- **Adequado para**
  - Dispositivos embarcados e IoT
  - Formato de arquivo para aplicações (alternativa ao CSV, XML, etc)
  - Website com pouco ou médio tráfego (400k pedidos / dia)
  - Análise de dados
  - Ensino e treinamento
- **Não seria muito adequado para**
  - Aplicações cliente/servidor
    - Acesso concorrente ao sistema de arquivos poderia gerar problemas
  - Grande conjunto de dados
    - Tamanho máximo 140TB, mas o limite do sistema de arquivos pode ser mais restritivo

[1] https://www.sqlite.org/about.html

Ferramentas

- Biblioteca SQLite no Linux

```
1 sudo apt install sqlite3
2
3 sqlite3 lab01.db
```

- DB Browser for SQLite[2]

```
1 sudo apt install sqlitebrowser
```

- Instalador para Windows no site oficial[3]

[2] https://sqlitebrowser.org/
[3] https://www.sqlite.org

Tipos de dados das colunas

https://www.sqlite.org/datatype3.html

Tipo	Descrição
`INTEGER`	Inteiro com sinal
`REAL`	Número real
`TEXT`	Cadeia de caracteres codificada em UTF-8
`BLOB`	Para armazenar binários (i.e. imagens, PDF) – *binary large object*

Afinidade das colunas[4]

- Diferente de outros SGBD, o SQLite adota tipo dinâmico e colunas podem armazenar dados de qualquer classe (NULL, inteiro, texto, real, blob)
  - `STRICT tables` foi introduzido a partir da versão 3.37 e garante o tipo estático das colunas

[4] https://www.sqlite.org/datatype3.html#type_affinity

Tipos de dados da coluna para data e hora

- Não existe um tipo de dado específico para data e hora, mas é feito uso de funções[5] que permite guardar esse tipo de informação como

Tipo	Descrição
`TEXT`	Cadeia de caracteres (`"YYYY-MM-DD HH:MM:SS.SSS"`)
`INTEGER`	Número de segundos desde a era Unix – 1970-01-01 00:00:00 UTC
`REAL`	Número de dias desde o meio-dia em Greenwich em 24 de novembro de 4714 a.C. de acordo com o calendário gregoriano proléptico

[5] https://www.sqlite.org/lang_datefunc.html

Operadores

https://www.sqlite.org/lang_expr.html#operators_and_parse_affecting_attributes

Operador	Descrição
`AND`	E lógico
`OR`	OU lógico
`NOT`	Negação
`!=, <>`	Diferente
`>, >=`	Maior, maior ou igual
`<, <=`	Menor, menor ou igual
`==, =`	Igualdade
`=`	Atribuição

Alguns comandos do sqlite3

- Criando ou conectando em um banco

```
1 sqlite3 banco.db
```

- Listando os bancos conectados na sessão atual

```
1 sqlite> .databases
```

- Listando as tabelas e a instrução usada para criar uma tabela

```
1 sqlite> .tables
2 sqlite> .schema NomeDaTabela
```

Criando uma tabela com o comando CREATE TABLE

https://www.sqlite.org/lang_createtable.html

```
1 CREATE TABLE Disciplina(
2    codigo INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
3    nome TEXT NOT NULL,
4    cargaHoraria INTEGER NOT NULL);
```

- `AUTOINCREMENT` somente em coluna com `PRIMARY KEY`

Relacionamentos entre tabelas

https://www.sqlite.org/foreignkeys.html

```
-- habilitando a restrição de integridade referencial
sqlite> PRAGMA foreign_keys = ON;

CREATE TABLE Pessoa(idPessoa INTEGER NOT NULL PRIMARY KEY, nome TEXT);

CREATE TABLE Telefone(
    idTel   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    rotulo  TEXT,
    numero  TEXT,
    pessoa  INTEGER,
    FOREIGN KEY(pessoa) REFERENCES Pessoa(idPessoa)
);

-- Inserindo linhas nas tabelas
INSERT INTO Pessoa VALUES (1,'Juca');
INSERT INTO Telefone (rotulo, numero, pessoa) VALUES ('celular', '9876', 1);
```

Listando conteúdo das tabelas

Configurações do interpretador de instruções sqlite3

```
sqlite> SELECT * FROM Telefone;
1|celular|9876|1

-- Mudando a forma de apresentação dos resultados
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM Telefone;

idTel  rotulo   numero  pessoa
-----  -------  ------  ------
1      celular  9876    1

sqlite> SELECT P.nome, T.rotulo, T.numero
        FROM Telefone T NATURAL JOIN Pessoa P;

nome  rotulo   numero
----  -------  ------
Juca  celular  9876
```

Listando conteúdo das tabelas

Configurações do interpretador de instruções sqlite3

- Exportando o resultado de uma consulta para um arquivo CSV

```
sqlite> .mode csv
sqlite> .header off
sqlite> .output arquivo.csv
sqlite> SELECT * FROM Telefone
```

- Importando conteúdo de arquivo CSV para tabela

```
sqlite> .mode csv
sqlite> .import 'arquivo.csv' Telefone
```

ALTER TABLE

Funcionamento no SQLite

- Apenas para alterar o nome da tabela ou adicionar novas colunas
- Se precisar modificar, renomear ou excluir uma coluna existente, então deverá
  - Renomear a tabela atual
  - Criar uma nova tabela com as colunas desejadas
  - Copiar dados para nova tabela e apagar a tabela anterior

```
1 -- Desabilitando a restrição de chave estrangeira
2 PRAGMA foreign_keys=off;
3 -- Iniciando uma transação
4 BEGIN TRANSACTION;
5 ALTER TABLE Funcionario RENAME TO _funcionario_antigo;
6 CREATE TABLE ....
7 INSERT INTO Funcionario (...) SELECT (...) from _funcionario_antigo;
8 COMMIT;
9 PRAGMA foreign_keys=on;
```

Demais funções do SQLite

- https://www.sqlite.org/lang_corefunc.html
- https://www.sqlite.org/lang_datefunc.html

```sql
SELECT date('now');
-- 2018-08-03

SELECT time('now');
-- 07:05:51

SELECT datetime('now');
-- 2018-08-03 07:05:51

SELECT strftime('%Y-%m-%d','now');
-- 2018-08-03
```

MySQL

Alguns tipos de dados de colunas

https://dev.mysql.com/doc/refman/8.1/en/data-types.html

Tipo	Descrição
`CHAR(M)`	Cadeia de caracteres de tamanho fixo. $0 \geq M \leq 255$
`VARCHAR(M)`	Cadeia de caracteres de tamanho variável. $0 \geq M \leq 65.535$. UTF-8 requer 3 bytes por caractere, então tamanho máximo de 21.844
`SMALLINT(M)`	Inteiro de -32.768 a 32.767
`INT(M)`	Inteiro de $-2.147.483.648$ a $2.147.483.648$
`BIGINT(M)`	Inteiro de $-9.223.372.036.854.775.808$ a $9.223.372.036.854.775.807$
`FLOAT(M,D)`	Real sendo M o total de dígitos para a parte inteira e D o total de digital para a parte decimal
`BOOLEAN`	Tipo booleano (TRUE ou FALSE)
`DATE`	Data `'0000-00-00'`
`TIME`	Hora `'00:00:00'`
`DATETIME`	Data e hora `'0000-00-00 00:00:00'` (`TIMESTAMP` é outra opção)
`JSON`	Para guardar documentos JSON

Criando tabelas

https://dev.mysql.com/doc/refman/5.7/en/create-table.html

```
CREATE TABLE Aluno(matricula INT, nome VARCHAR(80), email VARCHAR(80));
```

- Campo matrícula como chave primária e seu valor incrementado automaticamente para cada nova tupla inserida na tabela

```
CREATE TABLE Aluno(
    matricula INT NOT NULL AUTO_INCREMENT,
    nome VARCHAR(80),
    email VARCHAR(80),
    PRIMARY KEY (matricula));
```

Criando tabelas

- Nome das colunas não precisam fazer referência ao nome da tabela
  - Ex: nomeAluno, emailAluno
- Para a chave primária é desejado que faça referência ao nome da tabela, pois esse campo poderá ser chave estrangeira em outra tabela
  - Ex: matriculaAluno

Como alterar ou excluir tabelas

- **Para excluir**

```
DROP TABLE Aluno;
```

- **Para alterar**
  - Diferentemente do SQLite, o MySQL permite alterar, apagar ou incluir colunas com o `ALTER TABLE`

```
ALTER TABLE Aluno ADD COLUMN tel INT;

ALTER TABLE Aluno CHANGE COLUMN tel telefone VARCHAR(40);

ALTER TABLE Aluno MODIFY COLUMN telefone VARCHAR(25);

ALTER TABLE Aluno DROP COLUMN telefone;
```

Data Manipulation Language – DML

- **Inserir linhas**

```
INSERT INTO Aluno (nome, email, telefone)
    VALUES ('Joao', 'j@email.co.br', '48-1234');
```

- **Excluir todas as linhas da tabela**

```
DELETE FROM Aluno;
```

- **Atualizar valores de todas as linhas da tabela**

```
UPDATE Aluno SET curso = 'Telecomunicações';
```

- **Listar todos os alunos**

```
SELECT * FROM Aluno
```

Data Control Language – DCL

- Garantir privilégio de consulta sobre a tabela `NOTAS` do esquema `academico` ao usuário `appwebuser`, apenas quando conectado a partir da máquina `localhost` e com a senha `supersenha`

```
GRANT SELECT ON academico.NOTAS TO 'appwebuser'@'localhost' identified by '
    supersenha';
```

- Revogar todos privilégios concedidos ao usuário , com conexão oriunda de qualquer IP (%) sobre todas as tabelas do esquema `academico`

```
REVOKE ALL PRIVILEGES ON academico.* FROM 'bibliotecauser'@'%';
```

Data Transaction Language – DTL

- `START TRANSACTION` – inicia uma nova transação
- `COMMIT` – efetiva uma transação
- `ROLLBACK` – desfaz a transação atual, cancelando qualquer mudança feita
- `SET autocommit` – habilitar COMMIT automático

```
START TRANSACTION;
  -- guarde na variável A o resultado da soma de todas as linhas da coluna
     salario e cujo cargo tem valor igual a 1
  SELECT @A:=SUM(salario) FROM Salarios WHERE cargo=1;

  -- Atualize o valor da coluna folha para o valor da variável A
  UPDATE Financeiro SET folha=@A WHERE cargo=1;
COMMIT;
```

- Garantirá que se houver atualização da tabela Salarios, isso não irá influenciar a atualização da tabela Financeiro

Operadores

Operador	Descrição
`AND , &&`	E lógico
`OR , ||`	OU lógico
`NOT , !`	Negação
`!= , <>`	Diferente
`>, >=`	Maior, maior ou igual
`<, <=`	Menor, menor ou igual
`:=`	Atribuição
`=`	Atribuição para as instruções `SET` e `UPDATE`; igualdade para os demais contextos
`LIKE`	Busca por padrão
`IS [NOT] NULL`	Se [não] é NULO
`BETWEEN ...AND ...`	Valor dentro de uma faixa

Data Manipulation Language – DML

- **Cláusulas**
  - `FROM` – para especificar tabela
  - `WHERE` – para especificar condições
  - `GROUP BY` – para agrupar linhas
  - `HAVING` – condição por grupo
  - `ORDER BY` – para ordenar linhas
  - `DISTINCT` – selecionar dados sem repetição
  - `UNION` – para combinar duas consultas
- **Funções de agregação**
  - `AVG` – calcular média
  - `COUNT` – contar número de linhas
  - `SUM` – somar todos valore de um campo
  - `MAX` – maior valor em um campo

SELECT – consulta SQL (query)

- Resultado de uma consulta SQL é uma tabela

```
SELECT A1, A2, ..., An
FROM T1, T2, ..., Tn
WHERE P
```

- **A** – Atributo
- **T** – Tabela
- **P** – predicado da consulta

- Recuperando todas as colunas e linhas uma tabela

```
SELECT * FROM Aluno;
```

- Recuperando as colunas *nome* e `email` de todas as linhas

```
SELECT nome, email FROM Aluno;
```

- Recuperando todas disciplinas cursadas por um aluno e removendo duplicatas

```
SELECT DISTINCT disciplina FROM Aluno;
```

SELECT

- Recuperando todos os dados de todos os alunos do curso de Telecomunicações e que moram em São José

```
SELECT *  FROM Aluno
WHERE curso = 'Telecomunicações' AND cidade = 'São José';
```

- Todos funcionários com salário maior que R$ 1.000,00

```
SELECT *  FROM Funcionarios
WHERE salario > 1000;
```

- Usando operadores aritméticos

```
SELECT 5 * 2 AS resultado;

SELECT horaExtra * valorHora * 2 as valorHE FROM funcionario;
```

SELECT

- Ordenando o resultado

```
SELECT *  FROM Aluno
ORDER BY nome, matricula;
```

- Buscando por padrões em cadeias de caracteres
  - % – qualquer substring
  - _ – qualquer caracter

```
SELECT *  FROM Aluno  WHERE nome LIKE 'João%';

SELECT *  FROM Aluno  WHERE nome LIKE '%Silva%';

SELECT *  FROM Aluno  WHERE nome LIKE 'Jo_o %';
```

SELECT: Funções de agregação

- Com exceção do `COUNT`, todos os demais ignoram tuplas que tenham o valor NULL nos atributos agregados

```
SELECT COUNT(*) AS totalDeAlunos FROM Aluno;

SELECT AVG(salario) FROM Funcionario;
```

- `GROUP BY` – Obtendo salário médio dos funcionários por departamento

```
SELECT departamento, AVG(salario)
  FROM Funcionario GROUP BY departamento;
```

  - Atributos na instrução `SELECT` que estejam fora das funções de agregação devem obrigatoriamente aparecer na lista do `GROUP BY`

SELECT

- Obtendo nomes dos departamentos cujo salário médio de seus funcionários seja maior que 1000

```
SELECT departamento, AVG(salario) FROM Funcionario
  GROUP BY departamento
    HAVING AVG(salario) > 1000;
```

- Retornando no máximo as 100 primeiras linhas

```
SELECT nome FROM Aluno LIMIT 100;
SELECT nome FROM Aluno LIMIT 0,100;
```

- Retornando no máximo 20 linhas a partir da linha 100

```
SELECT nome FROM Aluno LIMIT 100,20;
```

Valores nullos (NULL)

- Listar os empréstimos que tenham nulo no atributo valor

```
SELECT idEmprestimo FROM Emprestimo WHERE valor IS NULL
```

- Resultado de operações aritméticas com nulo sempre será nulo

```
SELECT 5 + NULL; -- retorna NULL
```

- Qualquer comparação relacional com valores nulos sempre resultará em nulo

```
SELECT (123 > NULL);
SELECT (NULL = 456);
SELECT (NULL <> NULL);
```

Valores nullos (NULL)

Operadores lógicos

- **Operador lógico OR**

```
SELECT (NULL OR TRUE); -- TRUE
SELECT (NULL OR FALSE); -- NULL
SELECT (NULL OR NULL); --  NULL
```

- **Operador lógico AND**

```
SELECT (NULL AND TRUE); -- NULL
SELECT (NULL AND FALSE); -- FALSE
SELECT (NULL AND NULL); --  NULL
```

- **Operador lógico NOT**

```
SELECT (NOT NULL); -- NULL
```

Valores nullos (NULL)

- Operações de agregação (i.e. SUM, COUNT) ignoram nulos, com exceção do COUNT(*)

```
-- total de linhas cujo valor no atributo cidade não seja NULO
SELECT COUNT(cidade) FROM Agencia;

-- total de linhas da relação
SELECT COUNT(*) FROM Agencia;
```

Funções DATE_FORMAT e TIME_FORMAT

Máscara	Descrição	Exemplo
%m	mês	01, ..., 12
%c	mês	1, ..., 12
%M	mês por extenso	'janeiro', ..., 'dezembro'
%d	dia do mês	01, ..., 31
%e	dia do mês	1, ..., 31
%Y	ano - 4 dígitos	2012, 2013, ...
%y	ano - 2 dígitos	12, 13, ...
%h	hora - 12h	01, ..., 12
%H	hora - 24h	00, ..., 23
%i	minuto	0, ..., 59
%s	segundo	0, ..., 59
%W	dia da semana	'domingo', ..., 'sábado'

```
mysql> select @@lc_time_names; -- verificando atual localização
mysql> set lc_time_names = 'pt_BR'; -- definindo localização para pt_BR
mysql> charset utf8; -- definindo codificação para utf8
```

Funções DATE_FORMAT e TIME_FORMAT

```sql
SELECT DATE_FORMAT('2017-03-31', '%e %M %Y'); --  31 março 2017

SELECT DATE_FORMAT('2017-03-29', '%d/%m/%Y'); -- 29/03/2017

SELECT TIME_FORMAT('15:40:00', '%Hh %im %ss'); -- 15h 40m 00s

SELECT CURRENT_DATE(); -- 2017-03-29

SELECT CURRENT_TIME(); -- 15:40:00

SELECT NOW(); -- 2017-03-29 15:40:00

SELECT DATEDIFF(CURRENT_DATE(), '2017-02-08'); -- valor em dias
```

Funções DATE_ADD e DATE_SUB

- Função para operações de soma e subtração com datas

```
DATE_ADD( data, INTERVAL expr unidade) ou DATE_SUB(...)
-- unidades: MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, YEAR
```

- Exemplos

```
SELECT DATE_ADD('2017-03-31 15:40', INTERVAL 2 HOUR); -- ... 17:40:00

SELECT DATE_SUB('2017-03-31 15:40', INTERVAL 1 YEAR); -- 2016-03-31 ...

SELECT DATE_ADD('2017-03-31 15:40:00', INTERVAL '1 2' DAY_HOUR); -- 2017-04-01
    17:40:00

SELECT Nome FROM Aluno
WHERE CURRENT_DATE() > (DATE_ADD(DIngresso, INTERVAL 9 YEAR));
```

Manipulação de documentos JSON

https://dev.mysql.com/doc/refman/8.1/en/json.html

```
CREATE TABLE cidade (idCidade INT, info JSON);

INSERT INTO cidade VALUES (JSON_OBJECT("populacao", "10000"));

INSERT INTO cidade VALUES ('{"populacao" : "2000"}');

SELECT info FROM cidade;

SELECT info->>"$.populacao" FROM cidade;
```