A04 SQL Básica

Prof. Dr. George H. G. Fonseca

CDD003 Fundamentos de Banco de Dados
Pós Graduação em Ciência dos Dados
Universidade Federal de Ouro Preto

Março de 2020

Sumário

Introdução

- *Structured Query Language* (SQL) é a linguagem padrão para manipular bancos de dados relacionais.
- A SQL foi desenvolvida nos anos 70 pela IBM e foi adotada por várias outras companhias que desenvolvem SGBDs.

Definição e Tipos de Dados SQL

- SQL usa os termos **tabela**, **linha** e **coluna** para os termos relacionais relação, tupla e atributo, respectivamente.
- O principal comando SQL para definição de dados é o CREATE, que pode ser usado para criar esquemas, tabelas, tipos dentre outros.

Definição e Tipos de Dados SQL

```
CREATE SCHEMA UNIVERSIDADE;
```

```
CREATE TABLE UNIVERSIDADE.ALUNO
(
    Matricula          INT             NOT NULL,
    Nome               VARCHAR(45)     NOT NULL,
    DataNascimento     DATE            NOT NULL,
    Endereco           VARCHAR(45)     NOT NULL,
    CCodigo            INT             NOT NULL,
    PRIMARY KEY (Matricula),
    FOREIGN KEY (CCodigo) REFERENCES CURSO(Codigo)
);
```

Definição e Tipos de Dados SQL

- Os comandos DROP e ALTER são responsáveis por excluir e alterar uma tabela respectivamente.
- Esses comandos não serão abordados em detalhes pois ferramentas CASE apresentam a funcionalidade de gerar e executar esses comandos de forma intuitiva via interfaces gráficas.

```
DROP TABLE ALUNO;

ALTER TABLE    PROFESSOR
  ADD COLUMN 'Endereco' VARCHAR(45) NOT NULL AFTER 'Salario';
```

Consultas SQL

- Consultas mais simples serão apresentadas e gradualmente consultas mais complexas serão introduzidas.

- A forma básica da declaração SELECT é formada pelas três cláusulas SELECT, FROM, WHERE e tem a seguinte forma:

```
SELECT  <lista de atributos>
FROM    <lista de tabelas>
WHERE   <condicoes>;
```

- Em SQL, os operadores lógicos básicos de comparação são $=$, $<$, $<=$, $>$, $>=$ e $<>$. Eles correspondem respectivamente aos operadores de álgebra relacional $=$, $<$, $\leq$, $>$, $\geq$ e $\neq$.

Consultas SQL

Consulta 1. Recuperar a data de nascimento e salário dos professores cujo nome é 'George Fonseca'.

```
SELECT   DataNascimento, Salario
FROM     PROFESSOR
WHERE    Nome='George Fonseca';
```

Consultas SQL

Consulta 2. Recuperar o CPF e o salário de todos os professores que trabalham nos departamento cujo campus é 'Ipatinga'.

```
SELECT  CPF, Salario
FROM    PROFESSOR, DEPARTAMENTO
WHERE   Campus='Ipatinga' AND Codigo=DCodigo;
```

- Na cláusula WHERE da Consulta 2 a condição Campus='Ipatinga' é uma **condição de seleção** que seleciona a tupla particular de interesse da tabela DEPARTAMENTO, pois Campus é um atributo de DEPARTAMENTO.
- A condição Codigo=DCodigo é chamada **condição de junção**, pois combina duas tuplas: uma de DEPARTAMENTO e outra de PROFESSOR sempre que o valor de Codigo em DEPARTAMENTO é igual ao valor de DCodigo em PROFESSOR.

Nomes de Atributos Ambíguos, Pseudônimo e Variáveis de Tupla

- Na SQL o mesmo nome pode ser usado para mais de um atributo em diferentes tabelas.
- Se esse for o caso e uma consulta multi-tabela se refere a dois ou mais atributos com o mesmo nome, devemos qualificar o nome do atributo de acordo com o nome da tabela para prevenir ambiguidade.
- Isso é feito prefixando o nome da tabela ao nome do atributo, separando-os por ponto.

```
SELECT   PROFESSOR.CPF, PROFESSOR.Salario
FROM     PROFESSOR, DEPARTAMENTO
WHERE    DEPARTAMENTO.Campus='Ipatinga' AND
         DEPRATAMENTO.Codigo=PROFESSOR.DCodigo;
```

Nomes de Atributos Ambíguos, Pseudônimo e Variáveis de Tupla

- É possível ainda renomear nomes de tabelas para nomes mais curtos criando um **pseudônimo**, como no exemplo da Consulta 3.

- De fato essa prática é recomendável para facilitar a **legibilidade** das consultas SQL.

Consulta 3. Recupere o código e o nome das disciplinas que pertencem ao curso de 'Medicina'.

```
SELECT  D.Codigo, D.Nome
FROM    DISCIPLINA AS D, CONTEM AS C, CURSO AS CS
WHERE   D.Codigo=C.DCodigo AND CS.Codigo=C.CCodigo
        AND C.Nome='Medicina';
```

Cláusura WHERE Não Especificada e Asterisco

- Uma cláusula WHERE faltando significa que não há condição na seleção das tuplas, assim, todas as tuplas especificadas na cláusula FROM serão selecionadas.

Consulta 4. Selecione os CPFs de todos os professores.

```
SELECT  CPF
FROM    PROFESSOR;
```

Consulta 5. Selecione os CPFs de todos os empregados e todas as combinações de CPFs de professores e nomes de departamentos.

```
SELECT  E.CPF, D.Nome
FROM    EMPREGADO AS E, DEPARTAMENTO AS D;
```

Cláusura WHERE Não Especificada e Asterisco

- Para recuperar os valores de todos os atributos das tuplas selecionadas não é necessário listar todos os atributos na SQL, é possível especificar apenas um asterisco (*).

Consulta 6. Recuperar todos os atributos dos professores que trabalham no DEPARTAMENTO cujo código é 'DECOM'.

```
SELECT   *
FROM     PROFESSOR
WHERE    DCodigo='DECOM'
```

Cláusura WHERE Não Especificada e Asterisco

Consulta 7. Recuperar todos os atributos dos professores e os atributos do DEPARTAMENTO em que eles trabalham para cada empregado do departamento 'Computacao'.

```
SELECT   P.*
FROM     PROFESSOR AS P, DEPARTAMENTO AS D
WHERE    P.DCodigo=D.Codigo AND D.Nome='Computacao';
```

Tabelas como Conjuntos em SQL

- SQL não trata uma tabela como um conjunto, e sim como um multi-conjunto, assim tuplas duplicadas podem aparecer como resultado de uma consulta.

- Se deseja-se eliminar duplicatas dos resultados de uma consulta SQL a palavra-chave DISTINCT deve ser usada na cláusula SELECT, especificando que apenas tuplas distintas devem permanecer no resultado.

Consulta 8. Recuperar todos os salários distintos de todos os professores.

```
SELECT   DISTINCT Salario
FROM     PROFESSOR;
```

Tabelas como Conjuntos em SQL

- A SQL incorporou diretamente algumas das operações da teoria dos conjuntos, como operações de união (UNION), diferença (DIFFERENCE) e interseção (INTERSECT).
- Os resultados dessas operações são conjuntos de tuplas, assim, duplicatas são eliminadas do resultado.
- As operações sobre conjuntos se aplicam apenas a tabelas com tipos compatíveis, ou seja, as tabelas para as quais a operação está sendo processada têm que ter os mesmos atributos e na mesma ordem.

Tabelas como Conjuntos em SQL

Consulta 9. Listar todos o nomes de todas as pessoas (professores e alunos) que participaram da oferta de disciplinas no ano de 2020.

```
(SELECT   P.Nome
 FROM     PROFESSOR AS P, OFERTA AS O
 WHERE    P.CPF=O.PCPF AND O.Ano=2020)
UNION
(SELECT   A.Nome
 FROM     ALUNO AS A, OFERTA AS O
 WHERE    A.Matricula=O.AMatricula AND O.Ano=2020);
```

Correspondência de Padrão de Substring

- Uma funcionalidade da SQL permite a comparação de apenas partes de caracteres de uma string, usando o operador de comparação LIKE.
- Isso pode ser usado para **correspondência de padrões de strings**.
- Strings parciais são especificadas usando dois caracteres especiais: % substitui um número arbitrário de zero ou mais caracteres e sublinhado (_) substitui um único caractere.

Correspondência de Padrão de Substring

Consulta 10. Recuperar todos alunos cujo endereço contém 'Ipatinga'.

```
SELECT   Nome
FROM     ALUNO
WHERE    Endereco LIKE '%Ipatinga%';
```

Consulta 11. Recuperar todos os professores nascidos nos anos 80.[1]

```
SELECT   Nome
FROM     PROFESSOR
WHERE    DataNascimento LIKE '__8_______';
```

[1] O formato de data empregado na SQL é aaaa-mm-dd

Operações Aritméticas

- Outra característica da SQL permite o uso de operações aritméticas nas consultas.
- Os operadores aritméticos padrão para adição (+), subtração (-), multiplicação (*) e divisão (/) podem ser aplicados a valores numéricos ou atributos com domínios numéricos.

Consulta 12. Mostrar os salários resultantes se cada professor que do deprtamento 'Computacao' recebesse um aumento de 10%.

```
SELECT  P.Nome, P.Salario * 1.1 AS SalAumentado
FROM    PROFESSOR AS P, DEPARTAMENTO AS D
WHERE   D.Codigo=P.DCodigo AND D.Nome='Computacao';
```

Ordenando os Resultados de uma Consulta

- A SQL permite ao usuário ordenar as tuplas resultantes de uma consulta pelos valores de um ou mais dos atributos que aparecem no resultado da consulta usando a cláusula ORDER BY.

Consulta 13. Recuperar a lista dos professores e dos departamentos nos quais eles estão trabalham ordenados pelo nome do departamento e, dentro de cada departamento, ordenar alfabeticamente pelo nome.

```
SELECT    D.Nome, P.Nome
FROM      DEPARTAMENTO AS D, PROFESSOR AS P
WHERE     D.Codigo=P.DCodigo
ORDER BY  D.Nome, P.Nome;
```

Ordenando os Resultados de uma Consulta

- A ordem padrão é a ordem ascendente dos valores.
- Pode-se especificar a palavra-chave DESC se deseja-se ver os resultados na ordem decrescente de valores.
- É possível ainda limitar o número de tuplas retornadas com a cláusula LIMIT para qualquer tipo de consulta.
- Por exemplo, caso se desejasse os nomes dos departamentos em ordem decrescente a cláusula ORDER BY ficaria:

```
ORDER BY D.Nome DESC, P.Nome
LIMIT 5;
```

Funções de Agregação

- **Funções de agregação** são usadas para sumarizar informações de múltiplas tuplas em uma única-tupla sumário.
- Várias funções de agregação existem como a contagem de elementos (COUNT), o mínimo (MIN) e máximo (MAX) de uma série de valores, a soma (SUM) e a média (AVG) de um conjunto de dados.
- Há ainda funções de agregação para cálculos estatísticos mais avançados, porém estas não serão abordadas aqui.
- Essas funções podem ser usadas em uma cláusula SELECT.

Funções de Agregação

Consulta 14. Recuperar a soma dos salários de todos os professores, o salário máximo, o salário mínimo e o salário médio.

```
SELECT    SUM (Salario), MIN (Salario),
          MAX (Salario), AVG (Salario)
FROM      PROFESSOR;
```

- Pode-se usar a palavra-chave AS para criar renomear os nomes das colunas resultantes da consulta:

```
SELECT   SUM (Salario) AS TotalSal,
         MIN (Salario) AS MinSal,
         MAX (Salario) AS MaxSal,
         AVG (Salario) AS MedSal
FROM     PROFESSOR;
```

Funções de Agregação

Consulta 15. Recuperar o número total de professores que trabalham no departamento 'Computacao'.

```
SELECT  COUNT (*)
FROM    PROFESSOR AS P, DEPARTAMENTO AS D
WHERE   D.Codigo=P.DCodigo AND D.Nome='Computacao';
```

Agrupamento: Cláusulas GROUP BY e HAVING

- Muitas vezes deseja-se aplicar funções de agregação a subgrupos de tuplas em uma relação, onde esses subgrupos são baseados em alguns valores de atributos.
- Por exemplo, se desejarmos encontrar o salário médio dos empregados em cada departamento.
- Nesses casos precisa-se particionar as tabelas em conjuntos sem sobreposição (ou grupos) de tuplas.
- Cada grupo consistirá das tuplas que têm o mesmo valor para um determinado **atributo de agrupamento**.
- A cláusula GROUP BY tem sessa finalidade. A cláusula GROUP BY especifica os atributos de agrupamento, que devem também aparecer na cláusula SELECT.

Agrupamento: Cláusulas GROUP BY e HAVING

Consulta 16. Para cada departamento, recuperar o código do departamento, o número de professores no departamento e seu salário médio.

```
SELECT    DNumero, COUNT (*), AVG (Salario)
FROM      PROFESSOR
GROUP BY  DNo;
```

Agrupamento: Cláusulas GROUP BY e HAVING

- Algumas vezes deseja-se recuperar valores para funções de agregação apenas para grupos que satisfazem determinada condição. Isso pode ser feito pela cláusula HAVING.

Consulta 17. Para cada departamento, no qual mais de dois professores trabalham, recuperar o código do departamento, o nome do departamento e o número de professores que nele trabalham.

```
SELECT    D.Codigo, D.Nome, COUNT (*)
FROM      PROFESSOR, DEPRTAMENTO
WHERE     D.Codigo=P.DCodigo
GROUP BY  D.Codigo, D.Nome
HAVING    COUNT (*) > 2;
```

Estrutura Geral de uma Consulta SQL

- Em qualquer consulta SQL as cláusulas SELECT e FROM são **obrigatórias**. As cláusulas WHERE, GROUP BY, ORDER BY, HAVING e LIMIT são **opcionais**, porém, caso presentes, devem ocorrer nessa ordem.

```
SELECT   <lista de atributos>
FROM     <lista de tabelas>
 |WHERE    <condições>
 |GROUP BY <atributos>
 |ORDER BY <atributos>
 |HAVING   <condições>
 |LIMIT    <número de linhas>;
```

Declarações INSERT, DELETE e UPDATE

- O comando INSERT adiciona uma tupla a uma tabela.
- Deve-se especificar o nome da relação e um lista de valores para a tupla.
- Esses valores devem estar na mesma ordem em que os atributos foram especificados no comando CREATE TABLE.

```
INSERT INTO ALUNO VALUES
(
    '2018005',
    'Mariana Gomes',
    '2001-02-28',
    'Rua Pedro A. Cabral, Belo Horizonte - MG',
    '1'
);
```

Declarações INSERT, DELETE e UPDATE

- O comando DELETE remove uma tupla de uma tabela.
- Ele inclui uma cláusula WHERE, de forma similar às consultas, para selecionar as tuplas a serem deletadas.

```
DELETE FROM    ALUNO
WHERE          Nome='Caio Nunes';
```

Declarações INSERT, DELETE e UPDATE

- O comando UPDATE é usado para modificar os dados de uma ou mais tuplas de uma tabela.
- Assim como no comando DELETE, uma cláusula WHERE deve ser especificada para definir quais tuplas sofrerão atualização.

```
UPDATE    ALUNO
SET       Endereco='Av. Ipanema, Itabira - MG'
WHERE     Nome='Ana Pedrosa';
```

Declarações INSERT, DELETE e UPDATE

- Múltiplas tuplas podem ser atualizadas ao mesmo tempo em um único comando UPDATE, como no exemplo abaixo, que aumenta o salário dos professores do departamento de código '3' em 5%.

```
UPDATE    PROFESSOR
SET       Salario=Salario*1.05
WHERE     DCodigo='3';
```

Referências