UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

CIÊNCIA DA COMPUTAÇÃO

BANCO DE DADOS I

RESUMO III

Gabriel Mace dos Santos Ferreira

Marcus Vinícius Souza Fernandes

Ouro Preto

2021

Álgebra Relacional

A álgebra relacional são as operações básicas definidas no modelo relacional, denominadas operações de recuperação que podem ser divididas em:

- Operações específicas de bancos de dados relacionais: seleção, projeção, junção, entre outras.
- Operações da teoria de conjuntos: união, interseção, diferença e produto cartesiano

Vale ressaltar que o resultado de uma dessas operações é uma nova relação, formada a partir de uma ou mais relações.

1. Operações específicas de bancos de dados relacionais

1.1. **Operação Seleção:** Essa é uma operação unária (seleciona uma única relação) utilizada para selecionar um conjunto de tuplas de uma relação. Além disso, é possível afirmar que a ordem dos operandos não altera o resultado (operação comutativa), logo se utilizarmos o resultado de uma operação de seleção para realizar outra operação de seleção, a ordem não importa.

- **Notação:** $\sigma_{<cond>}(<R>)$, onde <cond> é uma condição de seleção e <R> é o nome da seleção.
- **Notação em cascata:** $\sigma_{<cond1>}(\sigma_{<cond2>}(\sigma_{<cond3>}(<R>))) = \sigma_{<cond1>E<cond2>E<cond3>}(<R>)$

É importante destacar que por ser uma operação de recuperação ela gera uma nova relação, no entanto o número de atributos (grau) dessa nova relação é idêntico ao da relação original, o que pode haver mudanças é o número de tuplas da nova relação (podendo ser menor ou igual à original).

1.2. **Operação Projeção:** Essa é uma operação unária utilizada para selecionar um conjunto de atributos de uma relação. Esta relação não é comutativa.

- **Notação:** $\pi_{<atributos>}(<R>)$, onde <atributos> é uma lista de atributos dentre os da relação R e <R> é o nome da relação.

- **Notação em cascata:** $\pi_{<lista1>}(\pi_{<lista2>}(<R>))) = \pi_{<lista1>}(<R>)$

Dada a natureza do modelo relacional, em um exemplo de operação de projeção com os seguintes atributos: nome e sobrenome. Caso duas tuplas possuam os mesmos valores, por exemplo, dois indivíduos possuem o nome “João” e o sobrenome “Silva”, apenas uma tupla representando estes valores será evidente (tuplas duplicadas são removidas). Portanto a relação resultante possui um número menor ou igual de tuplas em relação à original.

2. Sequência de operações

Normalmente são aplicadas diversas operações da álgebra relacional sobre uma relação, normalmente essa tática é denotada ao escrever as operações como uma única expressão ou a aplicação dessas operações uma por vez, nessa última instância é necessária a nomeação das relações envolvidas, dado que são geradas relações intermediárias. A renomeação dos atributos das relações intermediárias e de resultado pode ser feita ao listar os nomes dos novos atributos entre parênteses juntamente com o nomes das novas relações, assim como no exemplo a seguir:

Dep5_Emp <- $\sigma_{NumDepto=5}$(Empregado)
Resultado(PNome, UNome,Sal) <-
$\pi_{PrimeiroNome,UltimoNome,Salario}$(Dep5_Emps)

Para as relações vistas anteriormente, caso não seja utilizada a renomeação os atributos da relação resultante estarão na mesma ordem e terão os mesmos nomes que a relação original.

3. Operação Renomeação

Essa é uma operação utilizada para renomear uma relação ou os atributos desta.

- **Notação:** Ps(b1,b2,...,bn)(<R>) ou Ps(<R>) ou $P_{(b1,b2,...,bn)}$(<R>), onde <S> é o novo nome para a relação, <R> é a relação original e $<b_1,b_2,...,b_n>$ são os novos nomes dos atributos. Na primeira notação temos que tanto atributos quanto relação são renomeados, na segunda temos que

apenas a relação é renomeada e a terceira renomeia apenas os atributos.

4. Operações Teóricas de Conjuntos

Na álgebra relacional existe um grupo padrão de operações matemáticas sobre conjuntos, que podem ser:

- **Binárias:** Envolvem duas relações.
- **Binárias compatíveis para união:** Caso duas relações $R(A_1,A_2,...,A_n)$ e $S(B_1,B_2,...,B_n)$ possuam o mesmo grau "n" e se $dom(A_i) = dom(B_i)$ para $i<=i<=n$ elas são compatíveis para união. De forma simplificada, as relações devem possuir o mesmo número de atributos e estes atributos devem ser compatíveis com o seu par equivalente (possuir o mesmo domínio). Essas operações são:

  i) União: Denotada por R ∪ S, gera uma relação que inclui todas as tuplas presentes em ambas.

  ii) Interseção: Denotada por R ∪ S, gera uma relação que inclui todas as tuplas em comum entre R e S.

  iii) Diferença: Denotada por R - S, gera uma relação que inclui todas as tuplas que não são comuns entre R e S.

  As operações de união e interseção são comutativas e associativas e a relação resultante destas operações possuirão os mesmos nomes de atributos da primeira relação envolvida.

  Obs: As tuplas duplicadas são desconsideradas independente da operação utilizada.

  Além das operações mais comuns descritas anteriormente também há outra não muito usual que pode ser realizada, ela é a operação de conjunto binária **Produto Cartesiano:**

  - Denotada por R x S, gera uma nova relação que soma os atributos da relação R com os da S.

- Esta relação gera tuplas relacionando uma a uma das tuplas da relação R com as **n** tuplas da relação S.

- Nesta operação não há necessidade de compatibilidade e geralmente as tuplas formandas não fazem muito sentido (tuplas espúrias).

- Uma prática de extrema importância nestes casos é eliminar estas tuplas espúrias, aplicando uma seleção com uma condição que refaça o sentido da relação.

4.1. Operação Junção

A operação de Junção é bem semelhante ao produto final da operação Produto Cartesiano, porém diferentemente do Produto Cartesiano que segue uma série de etapas para resultar em uma relação com a ausência de tuplas espúrias, ela já obtém esta relação final com uma única operação que possui a seguinte notação: $R \bowtie_{<cond>} S$, onde R e S são as relações envolvidas e <cond> é a condição de seleção desejada.

Uma operação de Junção em que não é utilizada nenhuma condição é denominada Equijunção, mas ela também pode ser uma Junção Natural, nesse caso a condição existe de forma implícita encontrando de forma automática os atributos de mesmo nome entre as relações e realizando a seleção com base neles. A junção natural possui a seguinte notação: R * S, onde R e S representam as relações envolvidas.

4.2. Operação Divisão

Outra operação binária é a de Divisão, normalmente utilizada para uma relação de consulta especial em aplicações de bancos de dados, no entanto só é aplicada quando os atributos de uma relação também fazem parte de outra ($Z \subseteq X$).
A relação resultante consiste do conjunto de atributos da primeira relação que não pertencem à segunda relação (Z-X). As tuplas que formam essa relação são compostas de forma semelhante, ou seja, Z-X.

- **Notação: R(Z) ÷ S(X)**, em que R e S são as duas relações envolvidas, enquanto Z e X são seus atributos. O símbolo ÷ representa a operação em si.

5. Funções de Agregação e Agrupamento

São **funções matemáticas de agregação** utilizadas na álgebra relacional para representar uma solicitação por meio de funções aplicadas sobre a coleção de valores numéricos, como: **Sum**(soma), **Average**(média), **Maximum**(máximo), **Minimum**(mínimo), **Count**(contador de tuplas).
Outra solicitação pode ser feita por meio do **agrupamento de tuplas** de uma relação, a partir dos valores de alguns atributos, sendo possível aplicar uma função de agregação posteriormente. Vale ressaltar que é necessário a especificação do atributo utilizado para realizar o agrupamento.

- **Notação Agrupamento:** $_{\text{<atributos de agrupamento>}}\mathcal{Z}_{\text{<funções de agregação>}}$**(<R>),** onde **R** é a relação que se deseja agrupar, **atributos de agrupamento** representam a lista de atributos utilizadas no agrupamento, **funções de agregação** é uma lista de pares no formato **<função><atributo>** em que são definidos uma função de agregação à ser utilizada e o atributo sobre o qual ela será aplicada.

Ao utilizar a notação acima, obtêm-se uma relação com os atributos de agrupamento, mas com os resultados obtidos com a aplicação das funções de agregação (uma tupla para cada grupo gerado pelos atributos dos agrupamentos). No entanto, caso não seja especificado um atributo de agrupamento às funções de agregação, serão aplicadas a todas as tuplas da relação.

6. Fechamento Recursivo

Tipo de operação aplicada a um auto-relacionamento entre tuplas de mesmo tipo. Por exemplo, existe uma relação denominada Empregado e se deseja listar todos os empregados que são supervisionados por outro empregado denominado “Fábio Lemos”.

Exemplo:

FabioNum<-($\sigma_{\text{PrimeiroNome='Fábio' E UltimoNome = 'Lemos'}}$(Empregado))
Supervisao(NumEmp, NumSup)<-$\pi_{\text{numEmpregado, NumSupervisor}}$(Empregado)
Resultado1(Num)<-π_{numEmp}(Supervisao $\bowtie_{\text{NumSup=NumEmpregado}}$FabioNum)
Resultado2(Num)<-π_{numEmp}(Supervisao $\bowtie_{\text{NumSup=Num}}$Resultado1)
Resultado<-Resultado1 $\cup$ Resultado2

7. Operações de Junção Externa

As operações de Junção Externa são simplesmente operações de Junção, no entanto podem manter as tuplas de uma relação R e/ou de uma relação S independentemente se estas possuem tuplas que se combinam nas relações.

- **Junção Externa à Esquerda:** Mantém todas as tuplas do lado da relação R, caso não existam tuplas em S que combinam, os atributos de S serão preenchidos com valores nulos. **Notação:** R **_|X|** S.

- **Junção Externa à Direita:** Mantém todas as tuplas da relação S, caso não existam tuplas em R que combinam, os atributos de R são preenchidos com valores nulos. **Notação:** R **|X|_** S**.**

- **Junção Externa Completa:** Mantém todas as tuplas das duas relações, caso uma das tuplas não combine ela é preenchida como nulo**. Notação:** R **_|X|_** S.

8. Operação de União Externa

Essa operação é utilizada para realizar a união de duas tuplas caso elas não sejam compatíveis para união, no entanto, devem ser no mínimo parcialmente compatíveis, ou seja, possuir alguns atributos compatíveis para união.
Nessa situação, os atributos que não são compatíveis para união são mantidos na relação resultante e caso em uma tupla não existam valores para tal atributos, será atribuído valor nulo para representação.

Structured Query Language (SQL)

É uma linguagem de consulta que permite ao usuário realizar consultas em um esquema relacional, definir estruturas de dados, definir restrições de integridade, modificações de dados no banco, especificação de restrições de segurança, controle de transações e utilização em linguagens hospedeiras.
Como dito anteriormente, as operações são realizadas em um esquema relacional, no entanto termos como tupla, atributos e relação são substituídos na nomenclatura da linguagem por linha, coluna e tabela respectivamente.

Outros termos que não estão presentes no esquema relacional são esquema que um agrupamento de tabelas e outros componentes da aplicação de banco de dados e catálogo que é uma coleção de esquemas.

1. Comandos em SQL

Seguem alguns dos comandos essenciais para a linguagem SQL:

1.1. CREATE TABLE

Comando utilizado para especificar uma nova relação, são fornecidos o nome para a tabela, seus atributos e restrições.

- **Atributos:** Durante o fornecimento dos atributos são especificados o nome, tipo de dados e restrições de cada um. Os tipos de dados que os atributos podem assumir são:

  i) Numéricos: Valores inteiros, tais como: integer, int e smallint. Valores reais, como: float, real, double, precision. E números formatados, como: os decimal(i,j), numeric(i,j) e dec(i,j), onde o número de dígitos à esquerda e direita são definidos.

  ii) Alfanuméricos: Cadeias de caracteres de tamanho fixo, tais como: char(n) e character(n) ou de tamanho variável, como: varchar(n), char varying(n) e varying(n) (**n** neste caso é o tamanho máximo de caracteres).

  iii) Datas: Datas que contêm o dia, mês e ano. Podendo ser no formato brasileiro ou americano.

  iv) Horas: Podem conter horas, minutos e segundos, nanosegundos, dependendo da formatação desejada.

- **Restrições:** As restrições (CONSTRAINT) de chave são definidas após os atributos, vale ressaltar que é necessária a nomeação de cada uma dessas restrições. São utilizadas as palavras:

i) PRIMARY KEY: Para chaves primárias.

ii) UNIQUE: Para chaves candidatas.

iii) FOREIGN KEY: Integridade referencial.

iv) NOT NULL: Não permitir que valores nulos sejam atribuídos.

v) DEFAULT<valor>: Atribuído em caso de não especificar um valor específico.

- **Restrição de integridade:** Em caso de violação de uma restrição de integridade (atualizações do valor de uma chave primária e/ou de exclusão de uma determinada tupla) existem as seguintes opções:

  i) Bloqueio: Default (geralmente utilizado em casos que não se define uma opção).

  ii) CASCADE: Propagação.

  iii) SET NULL: Substituição por nulo.

  iv) SET DEFAULT: Substituição por um valor default.

  Obs.: As opções devem ser especificadas com as cláusulas **ON DELETE** e **ON UPDATE**.

Exemplo de comando CREATE:

```
CREATE TABLE  Empregado
(     PrimeiroNome      VARCHAR(15)      NOT NULL,
      InicialMeio       CHAR,
      UltimoNome        VARCHAR(15)      NOT NULL,
      NumEmpregado      CHAR(9)          NOT NULL,
      DataNascimento    DATE,
      Endereco          VARCHAR(30),
      Sexo              CHAR,
      Salario           DECIMAL(10,2),
      NumSupervisor     CHAR(9),
      NumDepto          INT              NOT NULL,
   CONSTRAINT PK_Emp PRIMARY KEY (NumEmpregado),
   CONSTRAINT FK_NumSup FOREIGN KEY (NumSupervisor)
     REFERENCES  Empregado (NumEmpregado),
   CONSTRAINT FK_EmpDep FOREIGN KEY (NumDepto)
     REFERENCES  Departamento (NumDepto)
);
```

Imagem 1 - Exemplo simples de uso do comando CREATE (retirada dos slides do professor Guilherme Tavares de Assis).

Obs.: Sabemos que um banco de dados é composto por diversas tabelas, logo a criação do banco nada mais é do que um script em que é chamado diversas vezes o comando CREATE TABLE.

1.2. DROP SCHEMA

Comando utilizado para a remoção de um esquema do banco de dados.

- **Notação:** DROP SCHEMA<E><opção>; onde <E> é o nome do esquema a ser removido e <opção> pode ser RESTRICT (não elimina o esquema se houver algum elemento nele) ou CASCADE (elimina o esquema e todas as opções).

1.3. DROP TABLE

Comando utilizado para a remoção de uma relação do banco de dados.

- **Notação:** DROP TABLE<R><opção>; onde <R> é o nome da relação a ser removido e <opção> pode ser

RESTRICT (não elimina a relação se houver alguma restrição ou visão associada a ela) ou CASCADE (elimina a relação e as restrições e visões associadas a ela).

1.4. ALTER TABLE

Comando utilizado para adicionar ou remover atributos e/ou restrições de uma relação, além de alterar a definição de um atributo. As formas mais comuns de utilizá-lo são:

- **Notação Adição Atributo**: ALTER TABLE<R>ADD<A><D>, em que o atributo A de domínio D é adicionado à relação R.

- **Notação Remoção Atributo:** ALTER TABLE<R>DROP<A><opção>, onde A é o atributo a ser removido e R a relação do qual ele está sendo removido, além disso <opção> pode ser RESTRICT (não elimina o atributo se houver alguma restrição ou visão referenciado-o) ou CASCADE (elimina o atributo, visões e restrições que o referenciam).

- **Notação Remoção Cláusula:** ALTER TABLE<R>ALTER<A> DROP DEFAULT, onde é removida a cláusula default do atributo A que se encontra na relação R.

- **Notação Adição Cláusula:** ALTER TABLE<R>ALTER<A> SET DEFAULT<V>, onde é adicionada uma cláusula default com valor V ao atributo A que se encontra na relação R.

Dado que é usual a criação das tabelas sem restrições de chaves estrangeiras, os comandos mais utilizados se encontram a seguir:

- **Notação Remoção Restrição:** ALTER TABLE<R>DROP CONTRAINT<C>, onde a restrição C é removida da relação existente R.

- **Notação Adição Restrição:** ALTER TABLE<R>ADD CONTRAINT<C>, onde a restrição C é adicionada à relação existente R.

2. Consultas em SQL

As consultas em SQL seguem são equivalente à seguinte expressão em álgebra relacional $\pi_{A1,A2,...,An}(\sigma_{cond}(R_1 * R_2 * R_m))$, no entanto em SQL a consulta segue a notação:

SELECT <A_1>,<A_2>,...,<A_n> {projeção}
FROM <R_1>,<R_2>,...,<R_n> {produto cartesiano}
WHERE <cond>; {seleção}, onde:

- cada <A_i>, para 1<=i<=n, representa um atributo;
- cada <R_j>, para 1<=j<=m, representa uma relação;
- <cond> é a condição de seleção (expressão lógica);

Obs1.: Se não houver uma condição de seleção não será necessário o uso do **WHERE**, além disso se for necessário consultar todos os atributos de uma relação pode-se substituir a lista de A_i por *.

Obs2.: É importante destacar que ao utilizar diversas relações no **FROM**, torna-se necessária a associação correta destas no **WHERE**, caso contrário valores inválidos podem ser colhidos.

2.1. Palavras-chave DISTINCT e ALL

Ao contrário do modelo relacional, o SQL permite a existência de tuplas idênticas em uma mesma relação. Vale ressaltar que em SQL uma relação não é um conjunto de tuplas. Para fazer o controle dessa tuplas idênticas são utilizadas as palavras chaves a seguir:

- **DISTINCT:** Utilizado quando é necessária a remoção de tuplas idênticas em conjunto com a cláusula SELECT. Exemplo: **SELECT DISTINCT** UltimoNome **FROM** Empregado.

- **ALL:** Utilizado quando o objetivo é manter as tuplas duplicadas, usualmente é a cláusula default. Exemplo: **SELECT ALL** UltimoNome **FROM** Empregado.

2.2. Operador LIKE

O operador LIKE é utilizado para comparações em partes de cadeia de caracteres. Sabendo disso, pode-se fazer a comparação das seguintes formas:

- %: Utilizado para substituir um número arbitrário de caracteres. **SELECT** * **FROM** Empregado WHERE PrimeiroNome LIKE ‘A %’, nesse caso serão listados todos os empregados com nome inicial A, dado que o % desconsidera os demais caracteres do nome.

- _: Utilizado para substituir um único caractere. **SELECT** * **FROM** Empregado **WHERE** DataNascimento LIKE ‘_ _ 5 %’, nesse caso há uma combinação entre o ‘%’ e ‘_’, de forma que serão listados apenas os indivíduos que nasceram na década de 50.

2.3. Operador BETWEEN

Operador utilizado para especificar o intervalo de valores de um atributo.

- **Notação:**

  SELECT *
  FROM <R>
  WHERE (<Atributo> **BETWEEN** value1 **AND** value2) **AND** <xtra_atribute>

  O <Atributo> representa o atributo sobre o qual o intervalo será analisado, value1 e value2 representam os valores do intervalo. Além disso, pode-se utilizar o AND ao final com o intuito de filtrar os valores recebidos especificando outra condição.

2.4. Valor NULL

A linguagem SQL permite a verificação se o valor de um atributo é NULL, por meio de consultas, para tal são utilizados os operadores IS ou IS NOT.

- **Notação:**

  SELECT <atribute1>
  FROM <R>
  WHERE <atribute2> **IS NULL**

  Onde estão sendo listados o <atribute1> de uma relação R, quando seu <atribute2> é nulo.

Obs.: O uso dos operados IS ou IS NOT é bem intuitivo, basta realizar a tradução do inglês para o português.

2.5. Renomeando Relações

Na linguagem SQL é possível “renomear” nomes de relações, ao utilizar o operador AS, no entanto não é obrigatório o seu uso em ferramentas como o Postman. Vale ressaltar que uma mesma relação pode possuir dois nomes, além disso também é possível renomear nomes de atributos. Seguem exemplos

- **Notação Renomeação:**

  SELECT E.PrimeiroNome, E.UltimoNome, E.Endereço
  FROM Empregado **AS** E, Departamento **AS** M
  WHERE (E.NumDepto = M.NumDepto)
  AND (D.NomeDepto = ‘Pesquisa’)

  As relações Empregado e Departamento estão sendo nomeadas como E e M respectivamente, além de estarem sendo selecionadas o nome, sobrenome e endereço dos funcionários que atendem os requerimentos.

- **Notação Renomeação Relação:**

  SELECT E.UltimoNome, M.UltimoNome
  FROM Empregado **AS** E, Empregado **AS** M
  WHERE (E.NumSupervisor = M.NumEmpregado)

  A relação Empregado está sendo renomeada como E e M respectivamente, dado que se deseja listar um grupo de empregados comuns, assim como seus supervisores.

- **Notação Renomeação Atributo:**

  ....
  FROM Empregado **AS** E(PN,IM,UN,Num,DN,End,Sexo,Sal,NumS,NumD)
  ....

2.6. Operadores Aritméticos

Na linguagem SQL é permitido o uso de operadores aritméticos, como '+', '-', '*', '/' e etc nas consultas.

- **Notação:**
  SELECT <atrib1>, 1.1*<atrib2>
  FROM <R>
  WHERE <X>

  Onde estão sendo apresentados os <atrib2> com um aumento de 10% de instâncias de uma relação <R> que atendem a condição <X>.

- **Exemplo:**

  SELECT PrimeiroNome, UltimoNome, 1.1 * Salario
  FROM Empregado, Trabalha_em, Projeto
  WHERE (Empregado.NumEmpregado = Trabalha_em.NumEmpregado) **AND** (Projeto.NumProj = Trabalha_em.NumProj) **AND** (NomeProj = 'Produto Y')

  Onde estão sendo apresentados os salários de empregados que trabalham no 'Produto Y' com um aumento de 10%.

2.7. Cláusula ORDER BY

Essa cláusula é utilizada para ordenar os valores obtidos na consulta de acordo com os valores de um ou mais atributos.

- **Notação:**

  SELECT <atrib1>, <atrib2>, <atrib3>
  FROM <R>
  WHERE <condition>

ORDER BY <atrib1>, <atrib3>

Onde as tuplas obtidas com a consulta dos atributos da relação R, serão retornados ordenadamente de acordo com <atrib1> e <atrib3>.

- **Exemplo:**

SELECT NomeDepto, PrimeiroNome, UltimoNome, NomeProj
FROM Departamento, Empregado, Trabalha_em, Projeto
WHERE
(Empregado.NumDepto=Departamento.NumDepto)
AND
(Empregado.NumEmpregado=Trabalha_em.NumEmpregado) **AND** (Projeto.NumProj = Trabalha_em.NumProj)
ORDER BY NomeDepto, PrimeiroNome

Onde as tuplas obtidas serão ordenadas de acordo com os atributos NomeDepto e PrimeiroNome.

2.8. Operações de Conjunto

Como dito anteriormente, grande parte das operações se baseiam no modelo relacional. Dessa forma, o modelo SQL contém operações de conjunto, no entanto só podem ser realizadas entre relações compatíveis.

- União (UNION): Une tuplas e durante a operação de união são removidas as tuplas duplicadas. Caso seja necessário manter as tuplas duplicadas usa-se UNION ALL. Notação:

SELECT DISTINCT NumProj
FROM Projeto, Departamento, Empregado
WHERE
(Projeto.NumDepto=Departamento.NumDepto)
AND (NumGerente=NumEmpregado) **AND** (UltimoNome= 'Silva')
UNION
SELECT DISTINCT NumProj
FROM Trabalha_em, Empregado

WHERE
(Empregado.NumEmpregado=Trabalha_em.NumEmpregado) **AND** (UltimoNome= 'Silva');

- Interseção (INTERSECT): Realiza a interseção de tuplas entre duas tabelas.
- Diferença (EXCEPT): Realiza a diferença de tuplas entre duas tabelas.

2.9. Junção entre Tabelas

Especifica a condição da junção na cláusula **FROM** por meio das cláusulas **JOIN (INNER JOIN)** e **ON**.

- **Notação:**

  SELECT PrimeiroNome, UltimoNome, Endereco
  FROM (Empregado **JOIN** Departamento **ON** Empregado.NumDepto=Departamento.NumDepto)
  WHERE NomeDepto= 'Pesquisa'

  Onde está sendo feita a junção sobre as relações Empregado e Departamento, no entanto apenas quando o atributo NumDepto coincidem.

3. Subconsultas

Como visto anteriormente uma consulta possui uma cláusula **WHERE**, um bloco completo (SELECT… FROM… WHERE) que existe dentro dessa cláusula é denominado subconsulta. Usualmente são utilizadas para auxiliar a consulta "externa".

- **Exemplo:**

  SELECT DISTINCT NumProj
  FROM Projeto
  WHERE NumProj IN (**SELECT** NumProj
  FROM Projeto, Departamento, Empregado 32
  WHERE
  (NumGerente=NumEmpregado)
  AND
  (Projeto.NumDepto=Departamen

```
                    to.NumDepto)          AND
                    (UltimoNome= 'Silva'))
          OR
          NumProj IN ( SELECT T.NumProj
                    FROM Trabalha_em AS T,
                    Empregado AS E
                    WHERE
                    (E.NumEmpregado=T.NumEmpr
                    egado) AND (E.UltimoNome=
                    'Silva'))
```

Obs.: O operador **IN** representa o operador pertence da teoria dos conjuntos.

3.1. Operadores SOME e ALL

Como visto anteriormente o operador **IN** pode ser utilizado para comparar um valor de um atributo com um conjunto, além dele é possível utilizar outros, no entanto esses necessitam que um operador lógico (=, <>, >, >=, <, <=) seja escolhido, como:

- **<operador lógico> SOME (ou ANY):** Retorna verdadeiro se o valor do atributo é <operador lógico> que algum valor do conjunto determinado.

  i) Exemplo:

```
SELECT E.PrimeiroNome, E.UltimoNome
FROM Empregado AS E
WHERE E.Salario > SOME (SELECT M.Salario
FROM Empregado AS M, Departamento AS D
WHERE (M.NumDepto = D.NumDepto) AND
(D.NomeDepto = ‘Pesquisa’))
```

- **<operador lógico> ALL:** Retorna verdadeiro se o valor do atributo é <operador lógico> que todos os valores do conjunto determinado.

  i) Exemplo:

```
SELECT PrimeiroNome, UltimoNome
FROM Empregado
```

WHERE Salario > **ALL** (**SELECT** Salario **FROM** Empregado **WHERE** NumDepto = 5)

Obs.: O operador ‘=SOME’ é equivalente operador **IN**.

4. Conjuntos Explícitos de Valores

Alternativamente ao uso de subconsultas é possível utilizar um conjunto explícito de valores na cláusula **WHERE.**

- **Exemplo:**

  SELECT DISTINCT NumEmpregado
  FROM Trabalha_em
  WHERE NumProj **IN** (1, 2, 3)

5. Subconsultas Correlacionadas

Quando uma condição presente na cláusula **WHERE** de uma subconsulta refere-se a um atributo de uma relação declarada na consulta externa é dito que as consultas estão correlacionadas. Nessa situação, a subconsulta é avaliada uma vez para cada tupla ou combinação de tuplas da consulta externa.

- **Exemplo:**

  SELECT
  FROM
  WHERE E.NumEmpregado **IN** (**SELECT** D.NumEmpregado
  FROM Dependente **AS** D)
  WHERE E.PrimeiroNome = NomeDependente

5.1. Função EXISTS

A função **EXISTS** é utilizada para verificar se o resultado de uma subconsulta é vazio ou não.

- **Exemplo:**

  SELECT E.PrimeiroNome, E.UltimoNome
  FROM Empregado **AS** E

WHERE NOT EXISTS (**SELECT** * **FROM** Dependente **AS** D **WHERE** E.NumEmpregado = D.NumEmpregado)

Oonde estão sendo retornados todos os atributos que não apresentam dependentes. Dado que a cláusula **NOT** nega os valores obtidos por **EXISTS.**

6. Agrupamentos e Funções de Agregação

A linguagem SQL incorpora conceitos da álgebra relacional, como os conceitos de agrupamento e funções de agregação.

6.1. Agrupamento

Para agrupar tuplas geralmente são utilizadas as cláusulas:

- **GROUP BY:** Utilizada para agrupar tuplas que possuem o mesmo valor para os atributos relacionados em tal cláusula.

- **HAVING:** Utilizada em conjunto com uma cláusula **GROUP BY** com o intuito de especificar uma condição de seleção sobre o grupo de tuplas recuperadas em uma consulta. Nesse caso, apenas grupos que satisfazem essa condição serão retornados.

6.2. Funções de Agregação

As funções de agregação são utilizadas sobre um agrupamento e funcionam em conjunto com uma cláusula **SELECT** ou uma cláusula **HAVING**. As principais funções de agregação são:

- **COUNT:** Número de tuplas recuperadas em uma consulta;

- **SUM:** Soma dos valores de um atributo em uma consulta;

- **MAX:** Valor máximo de um atributo em uma consulta;

- **MIN:** Valor mínimo de um atributo em uma consulta;

- **AVG:** Média dos valores de um atributo em uma consulta.

- **Exemplo SUM, MAX, MIN, AVG:**

  SELECT SUM(Salario), **MAX**(Salario), **MIN**(Salario), **AVG**(Salario)
  FROM (Empregado **JOIN** Departamento **ON** Empregado.NumDepto = Departamento.NumDepto)
  WHERE NomeDepto = ‘Pesquisa’;

  Serão retornados a soma de todos os salários, o maior salário, o menor salário e o salário médio.

- **Exemplo COUNT:**

  SELECT COUNT(*)
  FROM Empregado **AS** E, Departamento **AS** D
  WHERE (E.NumDepto = D.NumDepto) **AND** (D.NomeDepto = ‘Pesquisa’);

  Obs.: No **Exemplo COUNT** está sendo realizada uma comparação semelhante ao **Exemplo SUM, MAX, MIN, AVG**.

7. Comando INSERT

Em sua forma mais simples, este comando adiciona uma única tupla a uma relação. Para isso, devem ser especificados o nome da relação e a lista de valores da tupla (devem estar na mesma ordem em que foram definidos no CREATE TABLE).

- **Exemplo Usual:**

  INSERT INTO Empregado
  VALUES ('Rosana', 'P', 'Souza', '653298653', '1962-12-30', 'R.Alagoas, 1230', 'F', 3000, '987654321', 4)

  Onde está sendo adicionada uma nova tupla na relação Empregado, caso um dos valores viole uma restrição estabelecida durante a criação da relação, não será possível sua inserção na mesma.

- **Exemplo Especificando Atributos:**

  INSERT INTO Empregado (PrimeiroNome, UltimoNome, NumEmpregado, NumDepto)
  VALUES ('Rosana', 'P', 'Souza', '653298653', 4)

  Onde estão sendo especificados os atributos que receberão os valores inseridos, normalmente utilizado quando não se tem conhecimento ou não existem os valores dos demais atributos e não existe uma restrição quanto à valores nulos.

- **Exemplo Várias Tuplas:**

  CREATE TABLE Info_Deptos
  (Nome_Depto **VARCHAR(15),**
  Num_de_Emps **INTEGER,**
  Total_Sal **INTEGER**)
  INSERT INTO Info_Deptos
  SELECT NomeDepto, **COUNT**(*), **SUM**(Salario)
  FROM (Departamento **JOIN** Empregado **ON** Departamento, NumDepto=Empregado.NumDepto)
  GROUP BY NomeDepto

  Onde está sendo criada a relação Info_Depto e sendo adicionadas diversas tuplas na relação Info_Depto após uma pesquisa.

8. Comando DELETE

Este comando remove tuplas de uma relação específica, pode ser utilizada a cláusula **WHERE** para filtrar as tuplas a serem excluídas, caso esta cláusula não seja utilizada, todas as tuplas serão excluídas e a tabela permanece no banco, porém vazia.
Vale ressaltar que normalmente a exclusão se limita apenas a relação selecionada, no entanto dependendo das restrições de integridade pode se propagar.

- **Exemplo Exclusão Total:**

  DELETE FROM Empregado

  Onde estão sendo excluídas todas as tuplas da relação empregado.

- **Exemplo Exclusão com WHERE:**

  DELETE FROM Empregado
  WHERE Salario < 800

  Onde estão sendo excluídas todas tuplas cujo atributo Salario possua valor inferior a 800.

9. Comando UPDATE

Este comando é utilizado para alterar valores de determinados atributos em uma ou mais tuplas de uma única relação selecionada. De forma semelhante ao comando **DELETE**, pode ser utilizada a cláusula **WHERE** para filtrar as tuplas a serem modificadas, ademais pode ser utilizada a cláusula **SET** para especificar atributos a serem modificados e seus novos valores.

- **Exemplo:**

  UPDATE Projeto
  SET Localização = ‘Anchieta’, NumDepto = 1
  WHERE NumProj = 10

  Onde estão sendo alterados os valores dos atributos Localização e NumDepto de uma tupla da relação Projeto, cujo NumProj seja igual a 10.

Obs.: A alteração de uma chave primária pode se propagar para outras relações, visto que ela pode ser uma chave estrangeira com restrição de integridade referencial de propagação.

10. Visões em SQL

Em SQL uma relação única derivada de outras relações ou de outras visões definidas anteriormente é denominada uma visão, no entanto ela é considerada uma relação virtual, ou seja, não necessariamente existe na forma física e portanto as operações de atualização que podem ser aplicadas sobre ela são limitadas, por outro lado não há limitação nas consultas.
Uma das principais vantagens da utilização de visões é simplificar a especialização de algumas consultas e também promover uma maior segurança e autorização.

10.1. CREATE VIEW

Esse comando especifica uma nova visão, para tal devem ser especificados o nome da visão, uma lista de nomes de atributos e uma consulta. Toda visão está associada a uma consulta, para especificar o conteúdo da visão.

- **Notação:**

  CREATE VIEW V_Trabalha_em
  AS SELECT PrimeiroNome, UltimoNome, NomeProj, Horas
  FROM Empregado **AS** E, Projeto **AS** P, Trabalha_em **AS** T
  WHERE (E.NumEmpregado = T.NumEmpregado) **AND** (P.NumProj = T.NumProj),

  Onde está sendo criada uma visão V_Trabalha_em.

Obs.: Em uma situação em que não são realizadas operações aritméticas ou aplicação de funções sobre os atributos da visão, não é necessário especificar o nome dos atributos, dado que seriam os mesmos que os das relações definidoras da visão.

10.2. Consultas

Após a criação de uma visão, pode-se realizar consultas utilizando a ela.

- **Exemplo:**

  SELECT PrimeiroNome, UltimoNome
  FROM V_Trabalha_em
  WHERE NomeProj = ‘Projeto X’

  Onde está sendo realizada uma consulta sobre a visão V_Trabalha_em definida anteriormente que retorna o nome e sobrenome de um funcionário que trabalha no Projeto X.

10.3. Atualização

Uma visão está sempre atualizada, visto que não efetua nenhuma ação quando é definida, e sim no momento em que é especificada uma consulta que a utiliza. Quanto a atualização de atributos das relações que compõem uma visão, é necessária a verificação de algumas condições:

- **Visão definida por uma relação:** Se uma visão for definida por uma única relação e que a chave primária ou alguma chave candidata componham os atributos dessa visão, permite o mapeamento de cada tupla da visão para uma tupla única da relação definidora, ou seja, é possível realizar a atualização dos valores das tuplas.

- **Visão definida por múltiplas relações com junções:** Se uma visão é definida pela junção de múltiplas relações ela geralmente não é atualizável, dado que os valores alterados deverão ser mapeados entre as relações definidoras.

- **Visão definida com o uso de agrupamento e funções agregadas:** Se uma visão é definida com o uso de agrupamento e funções agregadas ela não é atualizável, dado que alguns atributos não existem nas relações definidoras.

10.4. Remoção

O comando **DROP VIEW** remove uma visão.

- **Exemplo:**

  DROP VIEW V_Trabalha_em.