



CEFET/RJ



SQL

Eduardo Ogasawara
eogasawara@ieee.org
<https://eic.cefet-rj.br/~eogasawara>

Structured Query Language (SQL)

- A Linguagem de Consulta Estruturada (SQL) é a linguagem de consulta declarativa padrão para banco de dados relacional
- Muitas das características originais do SQL foram inspiradas na álgebra relacional
- A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efetuar sobre um banco de dados, tais como:
 - DDL - Linguagem de Definição de Dados
 - DDL é um subconjunto da linguagem SQL que permite definir tabelas e elementos associados
 - DML - Linguagem de Manipulação de Dados
 - DML é um subconjunto da linguagem SQL que é utilizado para realizar inclusões, consultas, alterações e exclusões de dados nas tabelas

Linguagem de definição de dados (DDL)

DDL: Linguagem de definição de dados

- Permite a especificação dos conjuntos de relações e informações associadas, incluindo:
 - O esquema para cada relação
 - O domínio dos valores associados a cada atributo
 - As restrições de integridade
 - O conjunto dos índices a serem mantidos para cada relação
 - As informações de segurança e autorização para cada relação
 - A estrutura de armazenamento físico de cada relação no disco

Tipos básicos de domínio na SQL

- char(n)
 - String de caracteres de tamanho fixo com tamanho n especificado pelo usuário
- varchar(n)
 - String de caracteres de tamanho variável com tamanho n máximo especificado pelo usuário
- Int
 - Inteiro (um subconjunto finito de inteiros que é dependente da máquina)
- float, double
 - Números de ponto flutuante e ponto flutuante de precisão dupla com precisão dependente da máquina
- Existem domínios específicos para campos longos textuais e binários que variam de banco para banco
 - Dia-hora: timestamp, date, time
 - Textuais: Text, Clob
 - Binários: Image, Blob

Esquema de projetos usado nos exemplos

- departamento(*dnome, dnumero, gercpf**, gerdatainicio)
- dependente(*ecpf*, nome_dependente, sexo, datanasc, parentesco*)
- dept_localizacoes(*dnumero*, dlocalizacao*)
- empregado(*pnome, minicial, unome, cpf, datanasc, endereco, sexo, salario, gerente_cpf**, dno*)
- projeto(*pjnome, pnumero, plocalizacao, dnum**)
- trabalha_em(*ecpf*, pno*, horas*)
- (*) chaves estrangeiras

Linguagem de definição de consulta (DML)

Cláusulas *select* e *from*

- A cláusula *select* lista os atributos desejados no resultado de uma consulta
 - corresponde à operação projeção da álgebra relacional
- A cláusula *from* indica as tabelas desejadas
- Encontre o primeiro nome dos empregados
 - *select* E.pnome
from Empregado E
- Na álgebra relacional, a consulta seria:
 - $\pi_{E.pnome}(\text{Empregado } E)$

Projeção generalizada

- A cláusula *select* pode conter expressões aritméticas envolvendo os operadores $+$, $-$, $*$, $/$ e funções operando em constantes ou atributos de tuplas
- Encontre o primeiro e último nome com um aumento de 5% do salario dos empregados
 - *select* E.pnome, E.unome, 1.05*e.salario
from Empregado E
- Em álgebra relacional
 - $\pi_{pnome,unome,1.05*salario}(Empregado E)$

A cláusula *where* com filtragem de tuplas

- A cláusula *where* especifica condições que o resultado precisa satisfazer
 - Corresponde ao predicado de seleção da álgebra relacional
 - Os resultados da comparação podem ser combinados usando os conectivos lógicos *AND*, *OR*, e *NOT*
 - As comparações podem ser aplicadas aos resultados das expressões aritméticas
- Encontre todos os funcionários com salário superior a US\$1200:
 - *select* E.pnome, E.salario
from Empregado E
where E.salario > 1200
- Na álgebra relacional:
 - $\pi_{E.pnome, E.salario}(\sigma_{E.salario > 1200}(Empregado\ E))$

Seleção de todos os atributos de uma tabela

- O asterisco representa todos os atributos da tabela
 - Evite usar *select ** numa aplicação
 - Traga apenas os campos necessários
- Encontre todos os atributos do departamento de pesquisa
 - *select **
from Departamento D
where D.Dnome = 'Pesquisa'
- Em álgebra relacional:
 - $\sigma_{D.NOME=Pesquisa}(Departamento\ D)$

A cláusula where com filtragem between

- A SQL inclui um operador de comparação *between*
- Encontre o nome dos empregados com salário entre US\$ 1.000 e US\$ 2.000
 - *select E.pnome*
from Empregado E
where E.salario between 1000 and 2000
- Em algebra relacional:
 - $\pi_{E.pnome}(\sigma_{E.salario \geq 1000 \wedge E.salario \leq 2000}(Empregado E))$

Uso de NULL

- SQL permite que a consulta verifique se o valor de um atributo é NULL (ausente ou indefinido ou não se aplica)
- SQL usa *is* ou *is not* para comparar NULL pois considera que cada valor NULL é distinto de outros valores NULL, assim, comparação via igualdade não é apropriado
- Nota: Quando há atributos, numa condição de junção, que possuem valor NULL, as tuplas desses valores não são incluídas no resultado da junção
- Encontre os nomes de todos os empregados quem não possuem supervisores
 - *select* pnome, unome
from Empregado E
where gercpf *is* NULL
- Em álgebra relacional
 - $\pi_{E.pnome, E.unome}(\sigma_{E.gercpf \text{ is NULL}}(\mathbf{Empregado E}))$

Consulta com like

- O operador like pode ser usado para avaliar substrings
- Obter o nome dos empregados que tenham no nome a letra 'r'
- *select* e.pnome, e.unome, e.salario
from empregado e
where e.unome *like* '%r%'

Operações de Ordenação

- A cláusula *order by* ordem as tuplas do resultado da consulta
- Obtenha o nome dos empregados ordenados pelo salário
 - *select* e.pnome, e.unome, e.salario
from empregado e
order by e.salario

A cláusula FROM com mais de uma tabela

- A cláusula *from* lista as relações envolvidas na consulta
 - Corresponde à operação de produto cartesiano da álgebra relacional
- Encontre o produto cartesiano empregado e dependente
 - *select **
from Empregado E, Dependente D
- Em álgebra relacional
 - *Empregado E* \times *Dependente D*

A cláusula *where* com junção implícita

- Junção implícita
 - As tabelas comumente devem ser interligadas
 - Se há n tabelas, deve-se ter $n-1$ junções (implícitas ou explícitas)
- Encontre o nome dos empregados com os seus respectivos dependentes
 - *select* E.pnome, D.nome_dependente
from Empregado E, Dependente D
where E.Cpf = D.Ecpf
- Álgebra relacional
 - $\pi_{E.pnome, D.nome_dependente} (\sigma_{E.cpf=D.ecpf} (Empregado E \times Dependente D))$

Junção implícita

A cláusula where combinando filtragem com junção implícita

- Encontre o nome dos empregados do departamento de pesquisa usando junção implícita
 - *select* pnome
from Empregado E, Departamento D
where D.dnome = 'Pesquisa' *and* D.dnumero = E.dno
- Em álgebra relacional
 - $\pi_{E.pnome}(\sigma_{E.dno=D.dnumero \wedge dnome="Pesquisa"}(Empregado E \times Departamento D))$

A cláusula *join* para junção explícita

- A junção explícita combina tabelas diretamente com a cláusula *join*, especificando a condição de junção na cláusula *on*, o que torna a intenção da junção mais clara e o comando SQL mais legível
- Encontre o nome dos projetos e dos seus respectivos departamentos usando junção explícita
 - *select* P.pjnome, D.dnome
from Projeto P *join* Departamento D *on* (P.dnum = D.dnumero)
- Em álgebra relacional
 - $\pi_{P.pjnome, D.dnome}(\text{Projeto } P \bowtie_{P.dnum = D.dnumero} \text{Departamento } D)$

Junção implícita versus junção explícita

- Selecionar os gerentes dos departamentos com seus respectivos números de projetos de projetos localizados em Stafford
 - Usando junção implícita
 - `SELECT E.UNOME, P.PNUMERO,
FROM PROJETO P, DEPARTAMENTO D, EMPREGADO E
WHERE P.DNUM = D.DNUMERO
AND D.GERCPF = E.CPF
AND P.PLOCALIZACAO = 'Stafford'`
 - Usando junção explícita
 - `SELECT E.UNOME, P.PNUMERO,
FROM PROJETO P JOIN DEPARTAMENTO D ON P.DNUM = D.DNUMERO
JOIN EMPREGADO E ON D.GERCPF = E.CPF
WHERE P.PLOCALIZACAO = 'Stafford'`
- Como fica em álgebra relacional

Operações de conjuntos

- SQL apresenta algumas operações de conjuntos:
 - A operação de união (*union*), e em algumas versões da SQL há também as operações de diferença (*minus*) and interseção (*intersect*)
 - As relações resultantes dessas operações de conjuntos são de fato conjuntos de tuplas
 - Tuplas duplicadas são eliminadas do resultado
 - As operações de conjuntos se aplicam apenas a relações união compatíveis
 - As duas relações tem que ter os mesmos atributos que precisam aparecer na mesma ordem
- Obtenha a lista do nome de todos os projetos que envolvem algum empregado cujo sobrenome é 'Smith' como trabalhador ou como gerente do departamento que controla o projeto
 - *select p.pjnome*
from projeto p, departamento d, empregado e
where p.dnum = d.dnumero and d.gercpf = e.cpf and e.unome='smith'
union
select p.pjnome
from projeto p, trabalha_em t, empregado e
where p.pnumero=t.pno and t.ecpf=e.cpf and e.unome = 'smith'

Funções agregadas

- Essas funções operam no multiconjunto dos valores de uma coluna de uma relação e retornam um valor
 - *avg*: valor médio
 - *min*: valor mínimo
 - *max*: valor máximo
 - *sum*: soma dos valores
 - *count*: número de valores
- Obtenha a quantidade de empregados
 - *select count(*)*
from Empregado E
- Obtenha o salário médio dos empregados por departamento
 - *select count(*)*
from Empregado E

Funções agregadas agrupadas

- As funções de agregação são computadas para cada multiconjunto agrupado pelos atributos presentes na cláusula *group by*
- Obtenha o salário médio dos empregados por departamento
 - *select* E.dno, avg (E.salario)
from Empregado E
group by E.dno

Filtragem pós-agrupamento

- Filtros nos resultados do agrupamento devem ser realizados com a cláusula *having*
- Os predicados na cláusula *having* são aplicados após a formação de grupos, enquanto os predicados na cláusula *where* são aplicados antes da formação de grupos
- Encontre todos os departamentos cuja média salarial dos empregados seja superior a 32000
 - *select* E.dno, avg (E.salario) as salario
from Empregado E
group by E.dno
having e.salario > 32000

Consultas aninhadas

- Uma consulta com comandos *select* embutidos ou aninhados é chamada de consulta aninhada
- Esse tipo de consulta pode ser especificado dentro da cláusula *where* de uma outra consulta, chamada de consulta externa
- Diversas das consultas anteriores podem ser especificadas de modo alternativo usando aninhamento
- Em geral, é possível haver vários níveis de consultas aninhadas

Consultas aninhadas usando cláusula IN para registros de outra tabela

- Obtenha o CPF de todos os empregados que trabalham em projetos de números 1, 2, ou 3
 - *select* E.cpf
from Empregado E
where dno In (
 select D.dnumero
 from Departamento D
 where D.dnome='Pesquisa'
)

Consultas aninhadas usando cláusula IN para conjuntos explícitos

- É também possível usar um conjunto de valores explícito (enumerado) na cláusula WHERE, ao invés de uma consulta aninhada
- Obtenha o CPF de todos os empregados que trabalham em projetos de números 1, 2, ou 3
 - *select* distinct T.ecpf
from Trabalha_em T
where T.pno In (1, 2, 3)

Consultas aninhadas correlacionadas

- Caso a condição da cláusula WHERE da consulta interna referencie um atributo de uma relação declarada na consulta externa, as duas consultas são ditas correlacionadas
- O resultado de uma consulta aninhada correlacionada é diferente para cada tupla da relação da consulta externa

Uso do EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada (contém uma ou mais tupla associadas)
- Encontre o nome dos empregados com dependentes
 - *select* E.pnome
from Empregado E
where exists (
 select *
 from Dependente D
 where D.ecpf=E.cpf
 and E.pnome=D.nome_dependente
)

Uso do NOT EXISTS

- NOT EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém tupla associada)
- Encontre o nome dos empregados sem dependentes
 - *select e.pnome, e.unome*
from empregado e
where not exists (
*select **
from dependente d
where d.ecpf=e.cpf
)

Referências

