MAC0439 – Laboratório de Bancos de Dados

Aula 13 - SQL

Consultas com União, Intersecção e Diferença
de Relações
e
Operações de Inserção, Remoção e Alteração

2 de outubro de 2015

Profa. Kelly Rosa Braghetto

(Adaptação dos slides do prof. Jeffrey Ullman, da *Stanford University*)

Exemplo para a aula

- Todas as nossas consultas SQL serão baseadas no seguinte esquema de BD:

  Refrigerante(nome, fabricante)
  Lanchonete(nome, endereco, cnpj)
  Cliente(nome, endereco, telefone)
  Apreciador(nome_cliente, nome_refri)
  Venda(nome_lanch, nome_refri, preco)
  Frequentador(nome_cliente, nome_lanch)

Exemplo para a aula

- Todas as nossas consultas SQL serão baseadas no seguinte esquema de BD:

  Refrigerante(nome, fabricante)
  Lanchonete(nome, endereco, cnpj)
  Cliente(nome, endereco, telefone)
  Apreciador(nome_cliente, nome_refri)
  Venda(nome_lanch, nome_refri, preco)
  Frequentador(nome_cliente, nome_lanch)

União, Intersecção e Diferença

- União, intersecção e diferença de relações são expressas nas seguintes formas, todas envolvendo subconsultas:
  - (<subconsulta>) **UNION** (<subconsulta>)
  - (<subconsulta>) **INTERSECT** (<subconsulta>)
  - (<subconsulta>) **EXCEPT** (<subconsulta>)

Exemplo: Intersecção

- Usando

  Apreciador(nome_cliente, nome_refri), Venda(nome_lanch, nome_refri, preco) e Frequentador(nome_cliente, nome_lanch), encontre os clientes e refris tais que:

  1. O cliente aprecia o refri e
  2. O cliente frequenta pelo menos uma lanchonete que vende o refri

“Truque”:
a subconsulta é uma tabela armazenada.

Solução

Refris vendidos nas lanchonetes que o cliente frequenta.

```
(SELECT * FROM Apreciador)
   INTERSECT
(SELECT nome_cliente, nome_refri
 FROM Venda, Frequentador
 WHERE Frequentador.nome_lanch
            = Venda.nome_lanch
);
```

Semântica de multiconjunto

- Os comandos SELECT-FROM-WHERE usem “semântica de multiconjunto”
  - Ou seja, a resposta deles podem conter tuplas repetidas
- Já para as operações de união, intersecção e diferença, o padrão é a **semântica de conjunto**
  - Ou seja, as **duplicações de tuplas são eliminadas quando a operação é aplicada**.

Motivação: eficiência

- É muito caro eliminar duplicações de uma relação.
- A operação de projeção considera somente uma tupla por vez (não requer a ordenação das tuplas) – por isso a consulta feita com o comando SELECT-FROM-WHERE fica mais eficiente quando não é preciso remover duplicações.
- Para intersecções e diferenças, é mais eficiente ordenar as relações antes.
  - Nesse caso, as duplicações já podem ser facilmente eliminadas.

Controlando a eliminação de duplicações

- É possível forçar que o resultado de uma consulta seja um conjunto (= sem repetições) usando a cláusula **DISTINCT**:

```
SELECT DISTINCT atrib1, atrib2 ...
```

- Para forçar que o resultado seja um multiconjunto (ou seja, que as duplicações não sejam eliminadas) use a cláusula **ALL**, como em:

```
(<subconsulta>) UNION ALL (<subconsulta>)
(<subconsulta>) INTERSECT ALL (<subconsulta>)
(<subconsulta>) EXCEPT ALL (<subconsulta>)
```

Exemplo: DISTINCT

- A partir de

  Venda(nome_lanch, nome_refri, preco), encontre todos os diferentes preços cobrados por refrigerantes:

  ```
  SELECT DISTINCT preco
  FROM Venda;
  ```

- Sem o DISTINCT, cada preço poderia ser listado tantas vezes quanto o número de pares (nome_lanch,nome_refri) associados a esse preço na tabela.

Exemplo: ALL

- Usando as relações Frequentador(nome_cliente, nome_lanch) e Apreciador(nome_cliente, nome_refri):

```
(SELECT nome_cliente FROM Frequentador)
EXCEPT ALL
(SELECT nome_cliente FROM Apreciador);
```

- Lista o nome de cada cliente que frequenta um número de lanchonetes que é maior que o número de refris que ele gosta (e cada nome aparece tantas vezes quanto for a diferença entre essas quantidades).

Modificações no banco de dados

- Um comando de modificação não devolve um (multi)conjunto de tuplas como resultado (como uma consulta faz); ele **modifica o estado do BD de alguma forma**
- Existem 3 tipos de modificações:

1. Inserção de tupla(s)
2. Remoção de tupla(s)
3. Modificação do(s) valor(es) dos componentes de tupla(s) existente(s)

Inserção

- Para inserir uma única tupla:

  INSERT INTO <relação>

  VALUES (<lista de valores>);

- Exemplo: adicione a Apreciador(nome_cliente, nome_refri) o fato de que Ana gosta de Fanfa.

```
INSERT INTO Apreciador
VALUES('Ana', 'Fanfa');
```

Especificando atributos no INSERT

- Nós podemos adicionar ao nome da relação uma lista de atributos.
- Há duas razões para se fazer isso:
  1. Quando esquecemos a ordem de criação dos atributos na relação.
  2. Quando não temos valores para todos os atributos e queremos que o SGBD preencha os componentes faltantes com NULL ou um valor *default*.

Exemplo: especificando atributos

- Outra forma de adicionar o fato de que Ana gosta de Fanfa a Apreciador(nome_cliente, nome_refri):

```
INSERT INTO
 Apreciador(nome_refri, nome_cliente)
VALUES('Fanfa', 'Ana');
```

Adicionando valores padrão

- Em um comando CREATE TABLE, podemos indicar um valor padrão para um atributo, por meio da cláusula DEFAULT.
- Quando uma tupla inserida não possui valor para esse atributo, o valor padrão será usado.

Exemplo: valores padrão

```
CREATE TABLE Cliente (
   nome CHAR(30) PRIMARY KEY,
   endereco CHAR(50)
        DEFAULT 'Av. Paulista, 123',
   telefone CHAR(16)
);
```

Exemplo: valores padrão

```
INSERT INTO Cliente(nome)
VALUES('Ana');
```

Tupla resultante:

nome	endereco	telefone
Ana	Av. Paulista 123	NULL

Inserção de várias tuplas

- Podemos inserir o resultado todo de uma consulta em uma relação usando a forma:

  INSERT INTO <relação>

  (<subconsulta>);

Exemplo: inserção de subconsultas

- Usando Frequentador(nome_cliente, nome_refri), insira em uma nova relação Colegas(nome) todos os colegas “em potencial” da Ana, i.e., os clientes que frequentam pelo menos uma lanchonete frequentada pela Ana.

Nome dos potenciais colegas

Solução

Pares de tuplas de Clientes onde a primeira é para Ana e a segunda é para um outro cliente qualquer, e as lanchonetes são a mesma.

```
INSERT INTO Colegas
(SELECT f2.nome_cliente
 FROM Frequentador f1, Frequentador f2
 WHERE f1.nome_cliente = 'Ana' AND
   f2.nome_cliente <> 'Ana' AND
   f1.nome_lanch = f2.nome_lanch
);
```

Remoção

- Para remover tuplas que satisfazem uma condição em uma relação:

  DELETE FROM <relação>

  WHERE <condição>;

Exemplo: remoção

- Remova de Apreciador(nome_cliente, nome_refri) o fato de que Ana gosta de Fanfa:

```
DELETE FROM Apreciador
WHERE nome_cliente = 'Ana'
   AND nome_refri = 'Fanfa';
```

Exemplo: remoção de todas as tuplas

- Esvazie a relação Apreciador:

```
DELETE FROM Apreciador;
```

- Observe que nenhuma cláusula WHERE é necessária.

Exemplo: remoção de algumas tuplas

- Remova de Refrigerante(nome, fabricante) todos os refris para os quais existe um outro refri feito pelo mesmo fabricante.

```
DELETE FROM Refrigerantes r
WHERE EXISTS (
    SELECT nome FROM Refrigerante
    WHERE fabricante = r.fabricante AND
        nome <> r.nome);
```

Refris com o mesmo Fabricante e um nome diferente do nome do refri representado pela tupla r.

Semântica do comando de remoção do slide anterior (1)

- Suponha que a Cola-Coca produza somente Fanfa e Fanfa Diet.
- Suponha que o processamento da remoção “passe” primeiro pela tupla da Fanfa.
  - A subconsulta é não vazia, por causa da tupla da Fanfa Diet, então remove-se a tupla da Fanfa.
- Agora, quando **r** é a tupla para Fanfa Diet, essa tupla será removida também?
  - Essa dúvida é válida já que, após a remoção da tupla de Fanfa, Fanfa Diet passaria a ser o único refri de seu fabricante.

Semântica do comando de remoção do slide anterior (2)

- Resposta: Fanfa Diet será removida também!
- A razão para isso é o fato de que a remoção acontece em dois estágios:
  1. Marcação de todas as tuplas para as quais a condição WHERE é satisfeita.
  2. Remoção das tuplas marcadas.

Alterações

- Para mudar alguns atributos em algumas tuplas de uma relação:

  UPDATE <relação>

  SET <lista de atribuições a atributos>

  WHERE <condição sobre as tuplas>;

Assim como no comando DELETE, a cláusula WHERE é opcional.

Exemplo: alteração

- Mude o número do telefone do cliente Mário para 555-1212:

```
UPDATE Cliente
SET telefone = ’555-1212’
WHERE nome = ’Mário’;
```

Exemplo: **modificação de várias tuplas**

- Faça com que R$4 seja o preço máximo para um refrigerante:

```
UPDATE Venda
SET preco = 4.00
WHERE preco > 4.00;
```

Referências Bibliográficas

- *Database Systems – The Complete Book*, Garcia-Molina, Ullman e Widom. 2002. Capítulo 6
- *Sistemas de Bancos de Dados* (6ª edição), Elmasri e Navathe. 2010. Capítulos 4 e 5