[MAC0313] Introdução aos Sistemas de Bancos de Dados

Aula 17
Linguagem SQL (Parte 3)

Consultas Envolvendo Operações de Teoria dos Conjuntos e (prévia de) Agregações

10 de outubro de 2017

Profª Kelly Rosa Braghetto

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

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

- Embora os comandos SELECT-FROM-WHERE usem a “semântica de multiconjunto”, o padrão para a união, intersecção e diferença 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 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 (ou seja, não tenha repetições) usando a cláusula **DISTINCT**:

  SELECT DISTINCT . . .

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

  . . . UNION ALL . . .

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 os clientes que frequentam mais lanchonetes do que o número de refris que eles gostam (e faz isso tantas vezes quanto for a diferença entre essas contagens).

Agregações

- **SUM**, **AVG**, **COUNT**, **MIN** e **MAX** podem ser aplicados a uma coluna na cláusula SELECT para produzir a agregação da referida coluna.
- Além disso, **COUNT(*)** conta o número de tuplas.

Exemplo: Agregação

- A partir de
  Venda(nome_lanch, nome_refri, preço),
  encontre o preço médio de Fanfa:

```
SELECT AVG(preço)
FROM Venda
WHERE nome_refri = 'Fanfa';
```

Eliminando duplicações em uma agregação

- Pode-se usar o DISTINCT dentro de uma agregação
- Exemplo: encontre o número de preços *diferentes* cobrados pela Fanfa:

```
SELECT COUNT(DISTINCT preço)
FROM Venda
WHERE nome_refri = 'Fanfa';
```

Valores NULL são ignorados na agregação

- Um NULL nunca contribui para uma soma, média ou contagem, e nunca pode ser nem o mínimo, nem o máximo de uma coluna
- Mas se não existir valores não nulos em uma coluna, então o resultado da agregação é NULL
  - Exceção: COUNT de um conjunto vazio é 0

Exemplo: efeito de NULLs

```
SELECT count(*)
FROM Venda WHERE
nome_refri = ’Fanfa’;
```

O número de lanchonetes que vendem Fanfa.

```
SELECT count(preço)
FROM Venda WHERE
nome_refri = ’Fanfa’;
```

O número de lanchonetes que vendem Fanfa a um preço conhecido.

O que vem na sequência:

- Agregações com agrupamentos
- Cláusula HAVING
- Junções
  - Natural X Theta
  - Interna X Externa
    - Esquerda, direita, completa

Referências bibliográficas

- *A First Course in Database Systems,*
  Ullman e Widom. 1997.
  Capítulo 5
- *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