Álgebra Relacional

- Conjunto de operações que usa uma ou duas relações como entrada e gera uma relação de saída
  - *operação* (REL_1) $\rightarrow$ REL_2
  - *operação* (REL_1,REL_2) $\rightarrow$ REL_3

- Operações básicas:
  - seleção
  - projeção
  - união
  - diferença
  - produto cartesiano

Operadores da Álgebra Relacional

- Seleção:
  - seleciona tuplas que satisfazem um certo predicado ou condição

Pessoas

Nome	***Registro***
João	1
Maria	2
José	3

  - a) selecionar tuplas cujo nome = João

$\sigma_{nome=\text{“João”}}$ (Pessoa)

Nome	***Registro***
João	1

Operadores da Álgebra Relacional

b) selecionar as tuplas de *Pessoas* cujo registro > 1

$\sigma_{registro>1}$ (Pessoa)

Nome	***Registro***
Maria	2
José	3

c) selecionar as tuplas de *Pessoas* com registro > 1 e registro < 3

$\sigma_{registro>1 \wedge registro < 3}$ (Pessoa)

Nome	***Registro***
Maria	2

Operadores da Álgebra Relacional

- Projeção:
  - gera novas relações excluindo alguns atributos
  - exemplo: projete o atributo *Nome* sobre a relação *Pessoa*

Pessoa

Nome	***Registro***
João	1
Maria	2
José	3

Π_{nome} (Pessoa)

Nome
João
Maria
José

Operadores da Álgebra Relacional

- União:
  - união de atributos do mesmo domínio que estão em relações diferentes
  - as relações devem possuir o mesmo número de atributos
  - exemplo: encontre todos os clientes da agência que possuem conta corrente ou empréstimo.
    - Relações existentes para a agência:
      - CONTACORRENTE e EMPRÉSTIMOS

Operadores da Álgebra Relacional

- União: clientes com conta corrente ou empréstimo

CONTACORRENTE

Nome	***Conta***
João	1
Maria	2
José	3

EMPRÉSTIMO

Nome	***Empréstimo***
Paulo	100
Maria	200
Carlos	300

=

UNIÃO

Nome
João
Maria
José
Paulo
Carlos

Operadores da Álgebra Relacional

- Diferença:
  - tuplas que se encontram em uma relação, mas não em outra (ex: todos clientes sem empréstimo)

Nome	***Conta***
João	1
Maria	2
José	3

-

Nome	***Empréstimo***
Paulo	100
Maria	200
Carlos	300

=

DIFERENÇA

Nome
João
José

Operadores da Álgebra Relacional

- Produto Cartesiano
  - combina operações entre duas relações
  - união de atributos forma a nova relação

  (todos clientes com conta corrente X empréstimo de Maria)

Nome$_{cc}$	***Conta***	***Nome$_{emp}$***	***Empréstimo***
João	1	Maria	200
Maria	2	Maria	200
José	3	Maria	200

Operadores da Álgebra Relacional

- Operadores derivados:
  - intersecção
    - tudo que está em ambas relações

    (todos os clientes que possuem empréstimo)

CONTACORRENTE

Nome	***Conta***
João	1
Maria	2
José	3

EMPRÉSTIMO

Nome	***Empréstimo***
Paulo	100
Maria	200
Carlos	300

INTERS.

Nome
Maria

Operadores da Álgebra Relacional

- Operadores derivados
  - junção: inclui um produto cartesiano, seguido de uma seleção (pode ter projeção ao final)

    (nomes dos clientes com conta corrente e número de empréstimo)
    - prod. cartesiano: CONTACORRENTE X EMPRÉSTIMO
    - seleção: $\text{Nome}_{\text{contacorrente}} = \text{Nome}_{\text{empréstimo}}$
    - projeção: $\text{Nome}_{\text{contacorrente}}$, $\text{Empréstimo}_{\text{empréstimo}}$

Operadores da Álgebra Relacional

- Junção natural
  - nomes dos atributos nas relações da seleção são iguais
- Junção externa (“outer join”): evita perda de informação

(nome,endereço,cidade)	(nome,banco,salário)
{ Coyote,Toon,Hollywood }	{ Coyote,Mesa,1500 }
{ Coelho,Túnel,Cenoura }	{ Coelho,Mesa,1300 }
{ Smith,Revolver,Vale Morte}	{ Gates, Msm, 5300 }

Junção: (nome,endereço,cidade,banco,salário)

{ Coyote, Toon, Hollywood, Mesa, 1500 }

{ Coelho, Túnel, Cenoura, Mesa, 1300 }

Álgebra Relacional

- Funções de agregação
  - retorna um valor único de resultado a partir de uma coleção de valores

  sum: soma dos valores { sum $_{\text{salário}}$ (inpe) }

  avg: média dos valores

  count: total de ítens na coleção

  min e max: mínimo e máximo valores da coleção

  count-distinct: elimina duplicação primeiro e conta o total

SQL - Structured Query Language

- Linguagem de banco de dados relacional
  - linguagem de definição de dados (DDL)
  - linguagem de manipulação de dados (DML)
  - definição de vistas
  - integridade
  - controle de transação

SQL - Structured Query Language

- Linguagem de Definição de Dados (DDL)
  - é responsável por criar, alterar e excluir entidades, colunas, relacionamentos etc.
- Linguagem de Manipulação de Dados (DML)
  - permite ao usuário manipular os dados. Incluir, alterar e excluir dados de uma entidade.

SQL - Structured Query Language

- Linguagem de definição de dados (DDL)
  - esquema para cada relação
  - domínio de valores associados a cada atributo
  - restrições de integridade
  - índices para cada relação
  - segurança e autorização de acesso para cada relação
  - estrutura de dados em disco para cada relação

SQL - Structured Query Language

- Definição de esquema:
  - comando create table - cria uma nova tabela

create table r (A_1D_1, A_2D_2, ..., A_nD_n,
<restrição de integridade$_1$>,
....,
<restrição de integridade$_k$>)

Restrições de integridade:
primary key (A_{j1}, A_{j2}, ..., A_{jm}) - define chave
check (P) - verifica predicado

SQL - Structured Query Language

```
create table cliente
    (nome       char(20) not null,
     endereço   char(30),
     cidade     char(30),
     primary key (nome))

create table contacorrente
    (número     char(10) not null,
     banco      char(30),
     saldo      integer,
     primary key (número),
     check (saldo >= 0))
```

SQL - Structured Query Language

- Definição de esquema:
  - comando drop table - elimina tabela do banco

    drop table cliente

  - comando delete from - elimina todos os registros

    delete from cliente

  - comando alter table - altera tabela do banco

    alter table cliente add A D : adiciona atributo A com domínio D

    alter table cliente drop A : elimina atributo A

SQL - Linguagem de Consulta

- Sintaxe:
  - SELECT <atributos> FROM <relações> WHERE<expressão>
- SELECT: corresponde ao operador *projeção* da álgebra relacional
- FROM: corresponde ao operador *produto cartesiano* da álgebra relacional
- WHERE: corresponde ao operador *seleção* da álgebra relacional

Exemplos de Consultas

- ALUNO

Nome	***Id***
João	1
Maria	2
José	3

CADEIRA

Aluno	***Tipo***
1	escolar
2	normal
10	poltrona

- SELECT *nome* FROM *aluno* WHERE *id* = 1
  - FROM: todos as tuplas da relação *aluno*
  - WHERE: selecione as tuplas cujo *id* = 1
  - SELECT: projete o atributo *nome*

- SELECT *nome*, *id*, *tipo* FROM *aluno*, *cadeira* WHERE *id* = *aluno*

  – FROM: produto cartesiano (ALUNO x CADEIRA)

Nome	***Id***	***Aluno***	***Tipo***
João	1	1	escolar
Maria	2	1	escolar
José	3	1	escolar
João	1	2	normal
Maria	2	2	normal
José	3	2	normal
João	1	10	poltrona
Maria	2	10	poltrona
José	3	10	poltrona

- SELECT *nome*, *id*, *tipo* FROM *aluno*, *cadeira* WHERE *id* = *aluno*

  – WHERE: selecione tuplas cujo *id* = *aluno*

Nome	***Id***	***Aluno***	***Tipo***
João	1	1	escolar
Maria	2	2	normal

  – SELECT: projete os atributos *nome*, *id*, *tipo*

Nome	***Id***	***Tipo***
João	1	escolar
Maria	2	normal

SQL - Structured Query Language

- Operações em conjunto de caracteres (”strings”) :
  - O caracter % representa qualquer sub-string
  - O caracter _ representa qualquer caracter
  - O termo like é utilizado para comparar padrões

“Carl%” : qualquer nome que comece com “Carl”
“%ulo%” : qualquer nome que possui “ulo”
“_ _ _” : qualquer nome com 3 caracteres

```
select nome
from cliente
where endereco like “%Ademar%”
```

SQL - Structured Query Language

- Funções de agregação

  – Média: avg

```
select avg (saldo) from conta_corrente
where banco_nome = “Brasil”
```

  – Cláusula group by :
    - junta as tuplas com atributos em group by de mesmo valor

```
select banco_nome, avg (saldo)
from conta_corrente
group by banco_nome
```

SQL - Structured Query Language

- Funções de agregação
  - Contador : count
    - recupere o número de tuplas ou registros na relação cliente

```
select count (*) from cliente
```

  - Mínimo : min
  - Máximo : max
  - Soma : sum

SQL - Structured Query Language

- Junção natural : (“inner join”)

empréstimo (banco,numero,saldo)

{ Centro, L-170, 3000 }

{ Satelite, L-230, 4000 }

{ Inpe, L-260, 300 }

cliente (nome,emp_numero)

{ João, L-170 }

{ Sandra, L-230 }

{ Paulo, L-155 }

```
select *
from emprestimo inner join cliente on
                    emprestimo.numero = cliente.emp_numero
```

{ Centro, L-170, 3000, João, L-170}

{ Satelite, L-230, 4000, Sandra, L-230 }

SQL - Structured Query Language

- Junção externa : (“outer join”)

empréstimo (banco,numero,saldo)

{ Centro, L-170, 3000 }

{ Satelite, L-230, 4000 }

{ Inpe, L-260, 300 }

cliente (nome,emp_numero)

{ João, L-170 }

{ Sandra, L-230 }

{ Paulo, L-155 }

```
select *
from emprestimo left outer join cliente on
                        emprestimo.numero = cliente.emp_numero
```

{ Centro, L-170, 3000, João, L-170}

{ Satelite, L-230, 4000, Sandra, L-230 }

{ Inpe, L-260, 1700, null, null, }

Passos na Modelagem de BD

- Requisitos:
  - identificação dos dados
- Modelagem conceitual:
  - mapear visão do usuário em um conjunto de dados
  - descreve entidades, atributos e relacionamentos
- Implementação:
  - esquema de banco de dados
- Projeto físico:
  - estruturas de dados, métodos de acesso, segurança