

# **Structured Query Language ( S Q L )**

# SQL

A linguagem SQL foi desenvolvida originalmente no início dos anos 70 nos laboratórios da IBM em San Jose, dentro do projeto System R, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional proposto por E. F. Codd. O nome original da linguagem era **SEQUEL**, acrônimo para "**Structured English Query Language**", vindo daí o fato de, até hoje, a sigla, em inglês, ser comumente pronunciada "síquel" ao invés de "és-kiú-él", letra a letra. No entanto, em português, a pronúncia mais corrente é a letra a letra: "ésse-quê-éle".

# SQL

*Embora a SQL tenha sido originalmente criada pela IBM, rapidamente surgiram vários "dialetos" desenvolvidos por outros produtores. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela American National Standards Institute (ANSI) em 1986 e ISO em 1987.*

*A SQL foi revista em 1992 e a esta versão foi dado o nome de SQL-92. Foi revista novamente em 1999 e 2003 para se tornar SQL:1999 (SQL3) e SQL:2003, respectivamente.*

# SQL

A SQL:1999 usa expressões regulares de emparelhamento, queries recursivas e gatilhos (triggers). Também foi feita uma adição controversa de algumas características de orientação a objeto. A SQL:2003 introduz características relacionadas à XML, sequências padronizadas e colunas com valores de auto-generalização (inclusive colunas-identidade).

# SQL

*Tal como dito anteriormente, a SQL, embora padronizada pela ANSI e ISO, possui muitas variações e extensões produzidas pelos diferentes fabricantes de sistemas gerenciadores de bases de dados. Tipicamente, a linguagem pode ser migrada de plataforma para plataforma sem mudanças estruturais principais.*

# SQL

- SQL - Linguagem de Consulta Estruturada
- Origem em Sequel - desenvolvida pela IBM para o System R (DB2)
- Tornou-se a linguagem de consulta padrão para sistemas relacionais
- Possui muitas implementações (nem todas são compatíveis)

# SQL

Além de ser uma “Query Language”, SQL também possui recursos de:

- **Linguagem de Definição de Dados (DDL)**
  - **definição de esquemas de relações**
  - **exclusão de relações**
  - **criação de índices**
  - **modificação nos esquemas das relações**

# SQL

Além de ser uma “Query Language”, SQL também possui recursos de:

## ➤ **Linguagem de Manipulação de Dados (DML)**

- **consulta;**
- **inserção;**
- **remoção; e**
- **modificação de tuplas**

# SQL

Além de ser uma “Query Language”, SQL também possui recursos de:

## ➤ **Embedded DML**

- **Forma de comandos SQL que podem ser incorporados às linguagens de propósito geral (C, Fortran, Delphi)**

# SQL

Além de ser uma “Query Language”, SQL também possui recursos para:

- **Definição de Visões**
- **Definição de Autorizações**
  - Direitos de acesso às relações e às visões

# SQL

Além de ser uma “Query Language”, SQL também possui recursos para:

- **Definição de Restrições de Integridade**
  - Atualizações que violam as restrições são desprezadas
- **Controle de Transações**
  - Especificação do início e do fim das transações (algumas implementações permitem explicitar o tipo de bloqueio de dados para controle de concorrência)

# SQL - Estrutura Básica

A estrutura básica de uma expressão SQL consiste em 3 cláusulas: **select**, **from** e **where**.

- **Select:** Implementa a operação de projeção da Álgebra Relacional (  $\pi$  ).
- **From:** Implementa a operação de produto cartesiano da Álgebra Relacional (  $\times$  ).
- **Where:** Implementa a operação de seleção da Álgebra Relacional (  $\sigma$  ).

# SQL - Estrutura Básica

Exemplo:

```
Select distinct A1, A2, ..., An
From r1, r2, r3
Where P
```


$$\pi_{A_1, A_2, \dots, A_n}(\sigma_P(r_1 \times r_2 \times r_3))$$

# SQL - Estrutura Básica

- se a cláusula **Where** for omitida, então **P** é considerado **verdadeiro**.

Exemplo: Selecione todas as tuplas da relação **r**, mantendo-se apenas o atributo **A<sub>1</sub>**.

```
Select A1
From r
```

# SQL - Estrutura Básica

- ao contrário da Álgebra Relacional, o resultado de uma consulta em SQL pode apresentar tuplas repetidas; para eliminar duplicidades, deve-se utilizar o qualificador **distinct**.

**Exemplo:** Selecione todas as tuplas da relação **r**, mantendo-se apenas o atributo **A<sub>1</sub>**, e eliminando-se as tuplas repetidas.

```
Select distinct A1
From r
```

# SQL - Estrutura Básica

- o qualificador **all** especifica explicitamente que as repetições devem ser mantidas na seleção; este é o qualificador default da seleção;

**Exemplo:** Selecione todas as tuplas da relação **r**, inclusive as repetidas, mantendo-se apenas o atributo **A<sub>1</sub>**.

```
Select all A1
From r
```

# SQL - Estrutura Básica

- o símbolo \* pode ser utilizado na cláusula **Select** para denotar **todos** os atributos.

**Exemplo:** Selecione todas as tuplas da relação r, mantendo-se **todos** os seus atributos.

```
Select *
From r
```

**conta**

CtaNAg	<u>CtaNum</u>	Saldo
Down	A - 101	500
Mianus	A - 215	700
Perry	A -102	400
Round	A - 305	350
Brig	A - 201	900
Red	A - 222	700
Brig	A - 217	750

**devedor**

DevNome	<u>EmpNum</u>
John	L-17
Smith	L-23
Hayes	L-15
Jack	L-14
Curry	L-93
Smith	L-11
Will	L-17
Adams	L-16

**empréstimo**

ANome	<u>EmpNum</u>	Total
Down	L-17	1000
Red	L-23	2000
Perry	L-15	1500
Down	L-14	1500
Mianus	L-93	500
Round	L-11	900
Perry	L-16	1300

# SQL - Estrutura Básica

- A cláusula **Select** pode conter expressões aritméticas envolvendo os operadores **+**, **-**, **\*** e **/**.

**Exemplo:**

```
Select ANome, Total * 100  
From empréstimo
```

# SQL - Estrutura Básica

- O predicado da cláusula **Where** pode conter expressões lógicas envolvendo os operadores **and**, **or** e **not**.

**Exemplo:**

```
Select EmpNum  
From empréstimo  
Where (ANome = 'Perry') and (Total > 1200)
```

# SQL - Estrutura Básica

- O predicado da cláusula **Where** pode conter o operador de comparação **between - and**.

**Exemplo:**

```
Select EmpNum  
From empréstimo  
Where Total between 9.000 and 10.000
```

equivale a:

```
Select EmpNum  
From empréstimo  
Where Total >= 9.000 and Total <=10.000
```

# SQL - Estrutura Básica

- A **junção natural** pode ser obtida com o comando **select - from - where**, da seguinte forma:

**Exemplo:**

```
Select distinct DevNome , ANome  
From empréstimo , devedor  
Where empréstimo.EmpNum = devedor.EmpNum
```

# SQL - Estrutura Básica

- Predicado da junção natural combinado com outro predicado:

**Exemplo:**

```
Select distinct DevNome , ANome  
From empréstimo , devedor  
Where empréstimo.EmpNum = devedor.EmpNum  
and Total > 1.000
```

# SQL - Exercícios

Considere a Base de Dados “Empresa”. Escreva expressões SQL para as seguintes consultas:

- 1 - Retorne o nome e o endereço dos empregados que trabalham para o departamento “Research”.

# SQL - Exercícios

Considere a Base de Dados “Empresa”. Escreva expressões SQL para as seguintes consultas:

- 1 - Retorne o nome e o endereço dos empregados que trabalham para o departamento “Research”.

Select Fname, Lname, Address

From **employee, department**

**Where Dname = ‘Research’ and Dnumber = Dno**

# SQL - Exercícios

2 - Para todo projeto localizado em “Stafford”, liste o número do projeto, o número do departamento que o controla e o último nome, o endereço e a data de aniversário do gerente do departamento.

# SQL - Exercícios

2 - Para todo projeto localizado em “Stafford”, liste o número do projeto, o número do departamento que o controla e o último nome, o endereço e a data de aniversário do gerente do departamento.

```
Select Pnumber, Dnum, Lname, Address, Bdate  
From project , department , employee  
Where Plocation = 'Stafford' and Dnum = Dnumber  
and MGRSSN = SSN
```

# SQL - Exercícios

3 - Retorne os nomes de todas as empregadas que têm dependentes e os nomes destes.

# SQL - Exercícios

3 - Retorne os nomes de todas as empregadas que têm dependentes e os nomes destes.

Select **Fname, Lname, Dependent\_name**

From **employee , dependent**

Where **SSN = ESSN and employee.Sex = 'F'**

# SQL - Exercícios

4 - Para todo empregado que trabalha no projeto “ProductZ”, retorne o nome do departamento ao qual ele pertence.

# SQL - Exercícios

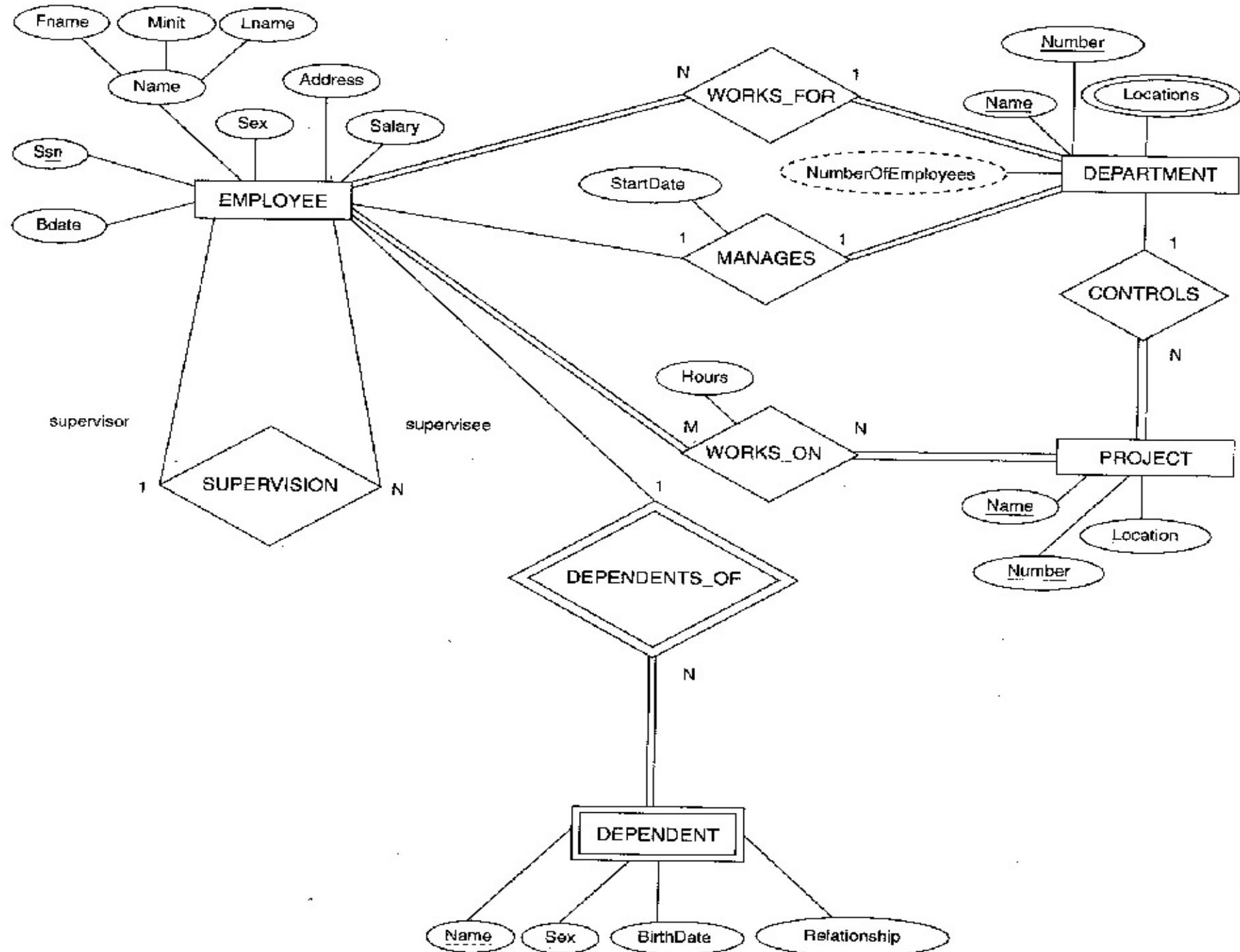
4 - Para todo empregado que trabalha no projeto “ProductZ”, retorne o nome do departamento ao qual ele pertence.

Select Dname

From project , works\_on , department , employee

Where Pname = ‘ProductZ’ and Pnumber = Pno

and ESSN = SSN and Dno = Dnumber



# SQL - Estrutura Básica

- SQL permite renomear tanto as relações quanto os seus atributos.

**Exemplo:**

```
Select distinct DevNome,  
       devedor.EmpNum as idemp  
From devedor,empréstimo  
Where devedor.EmpNum = empréstimo.EmpNum  
      and ANome = 'Perry'
```

# SQL - Estrutura Básica

- SQL permite a definição de variáveis tuplas, que precisam estar associadas a relações específicas.

## Exemplo 1:

```
Select distinct DevNome, T.EmpNum  
From devedor T , empréstimo S  
Where T.EmpNum = S.EmpNum  
and ANome = 'Perry'
```

# SQL - Estrutura Básica

- SQL permite a definição de variáveis tuplas, que precisam estar associadas a relações específicas.

## Exemplo 2:

```
Select E.LNAME as Employee_Name,  
      S.LNAME as Supervisor_Name  
  From employee E , employee S  
 Where E.SUPERSSN = S.SSN
```

# SQL - Estrutura Básica

- SQL permite a definição de variáveis tuplas, que precisam estar associadas a relações específicas.

## Exemplo 3:

```
Select E.FNAME , E.LNAME  
From employee E , dependent D  
Where E.SSN = D.ESSN and E.SEX = D.SEX  
      and E.FNAME = D.DEPENDENT_NAME
```

# SQL - Estrutura Básica

- SQL permite algumas operações com strings:
  - %: compara qualquer substring.
  - \_ : compara qualquer caractere.
  - Like: operador de comparação para pesquisar semelhanças.

## Exemplos:

- “Perry%” corresponde a qualquer string que comece com “Perry”;
- “%idge%” corresponde a qualquer string que possui “idge” como substring;
- “\_\_\_” corresponde a qualquer string com exatamente 3 caracteres;

```
select *  
from employee  
where ADDRESS like '%BAURU%'
```

# SQL - Estrutura Básica

- **escape:** para comparações que envolvem caracteres especiais.
- **not like:** operador de comparação para pesquisar diferenças.

## Exemplos:

**like 'ab\%cd%' escape '\'** corresponde a qualquer string que comece com 'ab%cd';

```
select *  
from employee  
where ADDRESS not like '%Bauru%'
```

# SQL - Estrutura Básica

- SQL permite **ordenar** as **tuplas** selecionadas, através da cláusula **order by**:

```
select distinct LNAME  
from employee  
where ADDRESS like '%Bauru%'  
order by LNAME asc
```

- A ordenação default é a crescente (asc).
- Os parâmetros de order by precisam ser parâmetros especificados na cláusula select.

# SQL - Estrutura Básica

- A ordenação pode ser de diversos atributos:

**Exemplo :**

```
select distinct LNAME,SALARY  
from employee  
where SEX = 'M'  
order by LNAME desc , SALARY
```

# SQL-Operações de Conjuntos

## ➤ Operação União:

**Exemplo :**

```
(select DepNome  
from depositante)  
union  
(select DevNome  
from devedor)
```

- A operação **União** elimina duplicações de tuplas automaticamente
- Se desejarmos manter todas as tuplas devemos utilizar o qualificador **all**

## Exemplo :

```
(select DepNome  
from depositante)  
union all  
(select DevNome  
from devedor)
```

➤ Operação Intersecção:

**Exemplo :**

```
(select DepNome  
from depositante)  
intersect  
(select DevNome  
from devedor)
```

- A operação Intersecção elimina duplicações de tuplas automaticamente
- Se desejarmos manter todas as tuplas devemos utilizar o qualificador **all**

## Exemplo :

```
(select DepNome  
from depositante)  
intersect all  
(select DevNome  
from devedor)
```

➤ Operação Minus:

**Exemplo :**

```
(select distinct DepNome  
from depositante)
```

minus

```
(select distinct DevNome  
from devedor)
```

- A operação Minus elimina duplicações de tuplas automaticamente
- Se desejarmos manter todas as tuplas devemos utilizar o qualificador **all**

## Exemplo :

```
(select DepNome  
from depositante)  
Minus all  
(select DevNome  
from devedor)
```

# SQL - Funções Agregadas

- Funções agregadas tomam uma coleção de valores como entrada e retornam um valor simples.
- SQL oferece 5 funções agregadas:
  - Média: **avg**
  - Mínimo: **min**
  - Máximo: **max**
  - Soma: **sum**
  - Contagem: **count**

# SQL - Funções Agregadas

## Exemplos:

Select **avg** (saldo)

From conta

Where CtaNag = 'Perry'

Select **count** (DevNome)

From devedor

Select **count** (**distinct** DevNome)

From devedor

# SQL - Agrupamentos

- Funções agregadas podem ser aplicadas em subconjuntos de tuplas, agrupadas através da cláusula **group by**.

**Exemplo:** Encontre o saldo médio das contas de cada agência.

```
Select CtaNag , avg(saldo)  
From conta  
Group by CtaNag
```

# S Q L - Agrupamentos

**Exemplo:** Encontre o número de depositantes de cada agência.

```
Select CtaNag , count(distinct DepName)
From depositante D , conta C
Where D.CtaNum = C.CtaNum
Group by CtaNag
```

# SQL - Predicado de Agrupamento

- **having:** cláusula que define o predicado que deve ser satisfeito pelo agrupamento de tuplas.

**Exemplo:**

```
Select CtaNag , avg(saldo)  
From conta  
Group by CtaNag  
having avg(saldo) > 1200
```

# SQL - Predicado de Agrupamento

- **OBS:** se as cláusulas `where` e `having` aparecem na mesma consulta, a cláusula `where` é aplicada primeiro.

**Exemplo:**

```
Select CtaNag , avg(saldo)  
From conta  
Where CtaNag like '%osp%'  
Group by CtaNag  
having avg(saldo) > 1200
```

# SQL - Valores Nulos

- SQL permite o uso do valor nulo (*null*) para indicar a ausência de informações sobre o valor de um atributo.
- SQL possui a palavra-chave especial **null** que pode ser utilizada para testar a existência de valores nulos.

## Exemplo:

```
Select EmpNum  
From empréstimo  
Where Total is null
```

```
Select EmpNum  
From empréstimo  
Where Total is not null
```

# SQL - Valores Nulos

- O uso do operador **is** é necessário, pois SQL considera os **valores nulos distintos entre si**, não sendo, portanto, válida a utilização do operador **=** para comparar valores nulos.

# **SQL - Aninhamento**

- **SQL proporciona mecanismos para o aninhamento de sub-consultas:**
  - **pertinência a conjuntos**
  - **comparação de conjuntos**
  - **verificação de relações vazias**
  - **teste para ausência de tuplas repetidas**

# SQL - Aninhamento

## Pertinência a Conjuntos

- Operador **in**

```
Select distinct DevNome  
From devedor  
Where DevNome in ( Select DepNome  
From depositante )
```

Consulta: *retorne os nomes de todos os clientes do banco que tenham conta e empréstimo.*

# SQL - Aninhamento

```
Select distinct D.DevNome  
From devedor D , empréstimo E  
Where D.EmpNum = E.EmpNum  
    and ANome = 'Perry'  
    and (ANome, D.DevNome) in  
        (Select C.CtaNAG, Dep.DepNome  
        From depositante Dep , conta C  
        Where Dep.CtaNum = C.CtaNum)
```

**Consulta:** *retorne os nomes de todos os clientes do banco que tenham conta e empréstimo na agência "Perry".*

# SQL - Aninhamento

## Pertinência a Conjuntos

- Operador **not in**

```
Select distinct DevNome  
From devedor  
Where DevNome not in (Select DepNome  
From depositante)
```

**Consulta:** *retorne os nomes de todos os clientes do banco que tenham empréstimo, mas que não tenham conta.*

# SQL - Aninhamento

- Operadores **in** e **not in** com conjuntos enumerados.

**Select distinct DevNome**

**From devedor**

**Where DevNome **not in** ('Smith','Jones')**

**Consulta:** *retorne os nomes de todos os clientes do banco que tenham empréstimo, e que não sejam nem o Smith e nem o Jones.*

# SQL - Aninhamento

## Comparação de Conjuntos

- Operador **some**

```
Select distinct T.ANome  
From agência as T, agência as S  
Where T.Fundos > S.Fundos  
and S.ACidade = 'Brook'
```

# SQL - Aninhamento

## Comparação de Conjuntos

- Operador **some** (ou **any**)

**Select ANome**

**From agência**

**Where Fundos > some (Select Fundos**

**From agencia**

**Where Acidade = 'Brook')**

**Consulta:** *retorne os nomes das agências do banco que tenham fundos maiores do que alguma agência localizada em "Brook".*

# SQL - Aninhamento

## Comparação de Conjuntos

### ➤ Operador **all**

Select ANome

From agência

Where Fundos > **all** (Select Fundos

From agencia

Where Acidade = 'Brook')

**Consulta:** *retorne os nomes das agências do banco que tenham fundos maiores do que todas as agências localizada em "Brook".*

# SQL - Aninhamento

## Comparação de Conjuntos

```
Select CtaNAg  
From conta  
Group by CtaNAg  
Having avg(Saldo) >= all ( Select avg (Saldo)  
                           From conta  
                           Group by CtaNAg )
```

Consulta: *retorne o nome da agência do banco que possui maior saldo médio.*

**SQL não permite agregação de funções agregadas, ou seja, max(avg(...)) não é permitido.**

# SQL - Aninhamento

## Verificação de Relações Vazias

- Operador **exists** (permite verificar se uma consulta possui tuplas ou se é vazia).

Select DevNome

From devedor

Where **exists** (

Select \*

From depositante

Where DepNome = DevNome )

Consulta: *retorne os nomes de todos os clientes do banco que tenham conta e empréstimo.*

# SQL - Aninhamento

## Verificação de Relações Vazias

- Operador **not exists**.

Select DevNome

From devedor

Where **not exists** (Select \*

From depositante

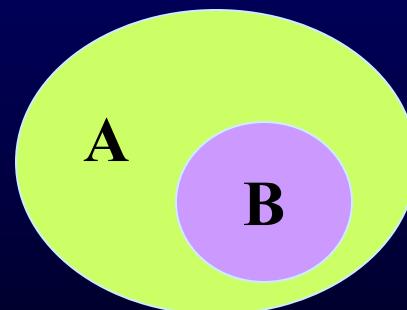
Where DepNome = DevNome )

Consulta: *retorne os nomes de todos os clientes do banco que tenham empréstimos, mas que não tenham conta.*

# SQL - Aninhamento

- Simulação do operador **contains** com os operadores **not exists** e **minus**.

O predicado “a relação A contém a relação B” é verdadeiro se **not exists (B minus A)**.



# SQL - Aninhamento

Select distinct S.DepNome

From depositante as S

Where not exists ( ( Select Anome

From agência

Where Acidade = 'Brook' )

minus

( Select R.CtaNAg

From conta as R

Where S.DNum = R.CtaNum

)

) and exists ( Select Anome

From agência Where Acidade = 'Brook')

# SQL - Aninhamento

```
Select distinct S.DepNome  
From depositante as S  
Where not exists ( ( Select Anome  
                      From agência  
                      Where Acidade = 'Brook' )  
                    minus  
                    ( Select R.CtaNAg  
                      From depositante as T, conta as R  
                      Where T.CtaNum = R.CtaNum  
                        and S.DepNome = T.DepNome )  
                )
```

Consulta: *Encontre os nomes dos clientes que tenham conta em todas as agências localizadas em "Brook" ⇒ Operação ÷ da Álgebra Relacional.*

**Encontrar todos os clientes que tenham conta em todas as agências localizadas em Brook.**

$$\pi_{\text{NomeCli}, \text{NomeAg}}(\text{depositante} \bowtie \text{conta}) \div$$
$$\pi_{\text{NomeAg}}(\sigma_{\text{CidadeAg} = \text{"Brook"}}(\text{agência}))$$

NomeCli	NomeAg
John	Down
Smith	Mianus
Hayes	Perry
Turner	Round
John	Brig
Jones	Brig
Lindsay	Red



# SQL - Aninhamento

## Teste para Ausência de Tuplas Repetidas

- Operador **unique**.

Select DepNome

From depositante as T

Where **unique** (Select R.DepNome

From conta, depositante as R

Where T.DepNome = R.DepNome

and R.CtaNum = conta.CtaNum

and conta.CtaNAg = 'Perry' )

**Consulta:** retorne os nomes dos clientes que têm apenas uma conta na agência “Perry”.

# SQL - Relações Derivadas

## Sintaxe:

```
( select a1,...,an
  from r1
  where P ) as r2 (b1,...,bn)
```

- Este comando cria uma relação temporária  $r_2$ , com atributos  $b_1, \dots, b_n$ , a partir dos atributos  $a_1, \dots, a_n$ , das tuplas de  $r_1$ , que satisfazem o predicado  $P$ .
- Possibilita o uso de sub-consultas na cláusula from.
- SQL-92

# SQL - Relações Derivadas

## Exemplo:

```
select NomeAgencia, SaldoMedio  
from ( select ANome, avg(Saldo)  
      from depositante  
     group by ANome )  
   as resultado (NomeAgencia,SaldoMedio)  
where SaldoMedio > 1200
```

# SQL - Visões

- Comando **create view**

**Sintaxe:**

**create view v as <expressão de consulta>**

# SQL - Visões

```
create view todos_clientes as
(      Select ANome, DepNome as CliNome
      From depositante as D, conta as C
     Where D.CtaNum = C.CtaNum )
union
(      Select ANome, DevNome as CliNome
      From devedor as D, empréstimo as E
     Where D.EmpNum = E.EmpNum )
Select CliNome
From todos_clientes
Where ANome = 'Perry'
```

# SQL - Visões

create view

**emp\_total\_agência (AgNome,EmpTotal)**

as

Select ANome, sum(Total)

From empréstimo

Group by ANome

# SQL - Visões

- O nome de uma visão pode aparecer em qualquer lugar onde permite-se o nome de uma relação;
- Consultas complexas são mais fáceis de escrever e entender quando são quebradas em visões menores e depois combinadas;
- a cláusula `create view` cria uma definição de visão em um banco de dados que fica armazenada até que um comando `drop view` seja executado.

# SQL - Modificação do BD

## Remoção de Tuplas

```
delete  
from r  
where P
```

- Remove as tuplas de **r** que satisfazem o predicado **P**.

# **SQL - Modificação do BD**

## **Remoção de Tuplas - Exemplos**

- 1) delete from depositante  
where DepNome = 'Smith'**
  
- 2) delete from empréstimo  
where Total between 1300 and 1500**
  
- 3 ) delete from conta  
where CtaNAg in (select Nome  
from agencia  
where Acidade = 'Perry' )**

# SQL - Modificação do BD

## Remoção de Tuplas - Exemplos

4) **delete from conta**

```
where Saldo < ( select avg (Saldo)
                  from conta )
```

# SQL - Modificação do BD

## Inserção de Tuplas

```
insert  
into r  
values (v1,v2,...,vn)
```

- Insere a tupla (v<sub>1</sub>,v<sub>2</sub>,...,v<sub>n</sub>) na relação r.

# SQL - Modificação do BD

## Inserção de Tuplas

```
insert into r1
select a1,...,an
from r2
where P
```

- Insere na relação  $r_1$  as tuplas formadas por valores dos atributos  $a_1, \dots, a_n$  das tuplas da relação  $r_2$  que satisfazem o predicado  $P$ .

# SQL - Modificação do BD

## Inserção de Tuplas - Exemplos

- 1) insert into conta  
values ('Perry','A-9732',1200)**
  
- 2) insert into conta  
select EmpNAg, EmpNum, 200  
from empréstimo  
where EmpNAg = 'Perry'**

# SQL - Modificação do BD

## Atualização de Tuplas

```
update r  
set ai = vi  
where P
```

- Modifica o atributo  $a_i$  (que recebe o novo valor  $v_i$ ) das tuplas de  $r$  que satisfazem o predicado  $P$ .

# SQL - Modificação do BD

## Atualização de Tuplas - Exemplos

1) **update conta**

**set Saldo = Saldo \* 1.05**

**where Saldo > 1000**

2) **update conta**

**set Saldo = Saldo \* 1.05**

**where Saldo > select avg (Saldo)**

**from conta**

# SQL - Modificação do BD

## Modificação de Visões

- Visões podem ser modificadas de forma idêntica às relações.
- O **efeito colateral** destas modificações é a alteração das relações sobre as quais as visões foram definidas. Modificações em visões, definidas em termos de duas ou mais relações, podem não ser toleradas.

# **SQL - DDL**

**SQL- DDL permite definir:**

- o esquema das relações;
- o domínio dos atributos;
- regras de integridade;
- índices;
- segurança e autorização;
- estrutura de armazenamento físico.

# SQL - DDL

## Tipos de Domínios:

- **char(n) e varchar(n)**
- **int (ou integer) e smallint**
- **numeric(p,d)**
- **real, double precision e float(n)**
- **date**
- **time**

### **Observações:**

- **O valor nulo pertence a todos os domínios;**
- **Para eliminar o valor nulo do domínio de um atributo deve-se utilizar a declaração **not null**.**

# SQL - DDL

## Criação de Domínios

### Sintaxe:

```
create domain NomePessoa char(20)
```

# SQL - DDL

## Criação de Tabelas

### Sintaxe:

```
create table r ( A1 D1 , A2 D2, . . . , An Dn ,
    <regra de integridade1>,
    ...,
    <regra de integridadek> )
```

A<sub>i</sub> ⇒ i-ésimo atributo da nova relação r

D<sub>i</sub> ⇒ domínio do i-ésimo atributo

# SQL - DDL

## Criação de Tabelas

### Exemplo:

```
1) create table cliente  
    ( Nome char(20) not null ,  
      Rua   char(30) ,  
      Cidade char(20) ,  
      primary key (Nome)  
    )
```

# SQL - DDL

## Criação de Tabelas

### Exemplo:

```
2) create table agencia  
    ( Nome char(15) not null ,  
      Cidade char(20) ,  
      Fundos integer,  
      primary key (Nome) ,  
      check (Fundos>=0)  
    )
```

# SQL - DDL

## Criação de Tabelas

### Exemplo:

#### 3) create table devedor

```
( NomeCliente char(20) not null ,  
  NumeroEmprestimo char(10) not null ,  
  primary key (NomeCliente,NumeroConta)  
)
```

# SQL - DDL

## Criação de Tabelas

### Exemplo:

#### 4) create table estudante

```
( Nome char(20) not null ,  
  NumeroId char(10) not null ,  
  Nivel char(11) not null,  
  primary key (NumeroId),  
  check (Nivel in ('Bacharelado','Mestrado',  
                    'Doutorado'))  
)
```

# SQL - DDL

## Criação de Tabelas

### Exemplo:

5) **create table conta**

```
( NomeAgencia char(20) ,  
  Numero char(10) not null ,  
  Saldo integer,  
  primary key (Numero),  
  foreign key (NomeAgencia) references agencia,  
  check (Saldo > 0 )  
)
```

# SQL - DDL

## Remoção de Tabelas

Sintaxe:

```
drop table r
```

# SQL - DDL

## Modificação de Tabelas

### Sintaxe:

**alter table r add A D**

**alter table r drop A**

# SQL - DDL

## Modificação de Tabelas

Exemplo:

**alter table agencia add NomeGerente char(30)**

**alter table agencia drop Cidade**