

Resumo

A large, handwritten signature in black ink, appearing to read "Ricardo", is written across three horizontal lines. The lines are evenly spaced and extend from left to right across the page.

Introdução aos Sistemas de Base de Dados

→ **Base de Dados:** uma coleção organizada de dados que estão relacionados e que podem ser partilhados por múltiplas aplicações.



Processamento Isolado de Dados

- **Dados isolados** - cada aplicação gera os seus próprios dados.
- Os mesmos dados podem estar replicados.
- Diferentes organizações e formatos de dados.
- Problemas de "sincronismo" → incoerências

Sistemas de Gestão de Ficheiros

- Dados organizados e armazenados em vários ficheiros partilhados por várias aplicações.
- Cada aplicação usa uma interface proprietária.

Problemas:
• Acesso concorrente aos dados
• Integridade • Segurança

Sistemas de Gestão de Base de Dados (SGBD)

Database Management System (DBMS): "is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications"

- **Definição:** Especificação do tipo de dados, estruturas de dados e restrições (database catalog or dictionary)
- **Construção:** Processo de armazenamento de dados
- **Manipulação:** Envolve operações como a pesquisa e obtenção de dados
- **Partilha:** Acesso simultâneo aos dados por parte de vários utilizadores e programas

Características Gerais

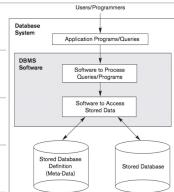
- Existe apenas uma entidade que opera com a BD / o acesso à BD é sempre mediado pelo SG-BD.
- Existe uma interface de acesso que esconde os detalhes de armazenamento físico dos dados.
- **Elevada abstracção** ao nível aplicacional
- Os dados estão integrados numa mesma unidade de armazenamento
- Suporta uma ou mais BD. **Keyword** - Data Independence

Vantagens

- Independência entre programas e dados
- Integridade dos dados → Controlo de alterações de dados de acordo com as regras de integridade definidas
- Consistência dos dados → Nos processos de transações e mesmo em falhas de software/hardware
- Eficiência no acesso aos dados
- Isolamento utilizadores → Um utilizador pensa que é o único.
- Melhor nível de acesso concorrential
- Serviços de segurança
- Mecanismos de Backup e recuperação de dados
- Administração de dados
- Linguagem de desenho e manipulação de dados

Desvantagens

- Maiores custos e complexidade na instalação e manutenção
- Não respondem aos requisitos de alguns cenários aplicacionais
- Centralização dos dados mais suscetíveis a problemas de tolerância a falhas e de escalabilidade



Utilizadores

- **Utilizadores Finais** → aqueles que usam o sistema com determinada finalidade com recurso a ferramentas disponibilizadas pelo fabricante do sistema ou aplicação de terceiros entidades
- **Programmadores de aplicação**: desenvolvem aplicações que permitem interagir com a base de dados
- **Administradores**: Tratam dos processos de gestão da BD.

Metadados (dados sobre dados)

- O SG-BD armazena uma descrição da própria estrutura da base de dados, restrições de integridade e condições de acesso

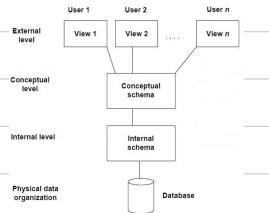
Interfaces

- Web-based
- Form-Based (desktop)
- GUI (Graphic user interface)
- Natural query language
- DBMS Command Line

Arquiteturas ANSI/SPARC

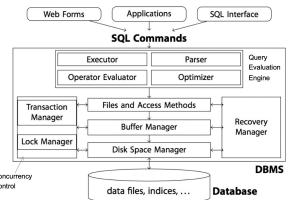
→ 3 níveis de arquitetura:

- Externa:
 - Oferece vistas da BD adaptadas a cada utilizador
 - **Dominio**: Utilizadores finais e prog. de aplicações
- Conceptual:
 - Descreve a estrutura da BD para os utilizadores (entidades, tipos de dados, relações, operações,...)
 - Oculta detalhes da implementação física (abstração)
 - **Dominio**: Administrador BD e prog. de aplicações
- Interno:
 - Lidam com a implementação física da base de dados
 - **Dominio**: programadores de sistemas de BD



Dois níveis de independência

- **Nível Físico:** Alterações do nível físico não devem ter impacto no esquema conceptual (modelo de dados)
- **Nível Lógico:** Alterações no esquema conceptual não devem repercutir-se nos esquemas externos ou aplicações já desenvolvidas

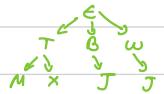


Modelo de Base de Dados → coleção de conceitos para descrição lógica de dados

Esquema (schema) → a descrição de um conjunto particular de dados com recurso a um determinado modelo

Modelo Hierárquico

- Dados armazenados numa estrutura hierárquica em "árvore".
 composta por um conjunto de
 atributos
- Os nós da árvore designam-se como registos que estão ligados por ponteiros (links).
- Um link é uma associação entre dois registos pai-filho.
- Um registo pai encontra-se associado a N registos filhos (1:N).



Modelo Hierárquico - (Des)vantagens

- Adaptado a cenários de acesso sequencial aos dados.
 - Qualquer acesso aos dados passa sempre pelo segmento raiz.
 - A maior parte das necessidades atuais requer acesso aleatório.
- Redundância de Informação
 - Descendência de espécie e inconsistências de dados.
- Restrições de Integridade, exemplo:
 - A eliminação de um segmento pai implica a remoção de todos os segmentos filhos associados.
- Não permite estabelecer associações N:M

Modelo de Rede ⇒ Extensão do modelo hierárquico

- Permite que um mesmo registo esteja envolvido em várias associações
- Relações representadas através de um grafos.

Desenho de Base de Dados - Diagramas E/R

Desenho de Base de Dados → • Análise de Requisitos

• Desenho Conceptual

• Desenho do Esquema Lógico ↑ SGBD - independente
↓ SGBD - dependente

• " " " Físico

• Administração

→ Análise de Requisitos: obriga a um processo de comunicação com o cliente

1. Levantamento detalhado de toda a informação associada ao "problema" do mundo real: entidades, relações, restrições, etc;
2. Filtragem da informação: remoção de redundâncias e "ruído" (informação pouco importante);
3. Discussão para clarificar aspectos ambíguos e eventuais falhas no ponto 1.
4. Distinção entre dados e operações

Modelo Entidade/Relacionamento (E/R)

Elementos principais:

- Entidades algo que existe; representar-se por retângulos

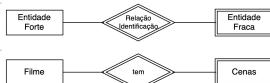
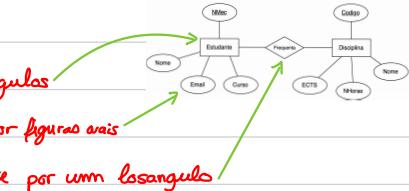
- Atributos propriedades da entidade; representar-se por figuras ovais

- Relacionamentos relação entre duas ou mais entidades; representar-se por um losango

- As entidades têm um ou mais atributos chave que as identificam. O nome destes atributos deve aparecer sublinhado nos diagramas E/R.

Entidades → Fortes → Não dependem de outras entidades

Entidades → Fracos → Dependem de outras entidades



- Os atributos podem ser: Derivados; Multivalor; Compostos;

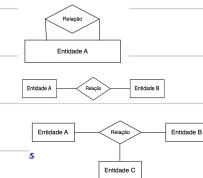
Relacionamentos: Interações entre 2 ou + entidades. Podem ter atributos

Classificação de Relacionamentos:

→ **Grau:** nº de entidades envolvidas na relação → • **Unária**

• **Binária** (mais comum)

• **Ternária** (podem ser convertidos em binárias)



→ **Obrigatoriedade:** da participação da entidade na relação

→ **Participação Total** (obrigatório): cada instância da entidade participa em pelo menos uma relação do conjunto de relações (linha dupla)

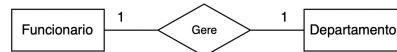


→ **Participação Parcial** (opcional): algumas(s) instância(s) da entidade podem não participar em qualquer relação do conjunto de relações

→ **Cardinalidade:** Relação entre o número de ocorrências numa entidade com as respectivas ocorrências na outra, com qual têm um relacionamento.

→ **Relação 1:1**: (um-para-um) → **Relação 1:N**: (um-para-muitos)

→ **Relação N:M**: (muitos-para-muitos)



Um funcionário gera um departamento. Um departamento só tem um gestor.



Um funcionário trabalham para um departamento. Um departamento tem vários funcionários.

Obrigatoriedade - Notação E/R (min,max)

• Existe uma notação alternativa com (min,max) para impor restrições à participação de cada entidade na relação



minímo: "0" → é opcional a participação da entidade na relação

máximo: "1" → cada instância da entidade está, no máximo, associada a uma única instância da relação

"1" → é obrigatória a participação da entidade na relação

"N" → cada instância da entidade está associada a várias instâncias da relação.

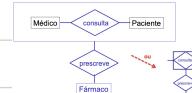
Agregação

- As vezes é necessário modelar uma relação entre uma entidade e outra relação envolvendo outras entidades

Exemplo: Como associar fármacos prescritos numa consulta médica?

Solução: Tornar uma relação numa entidade associativa

Entidade Associativa: Permite associar entidades a relacionamentos

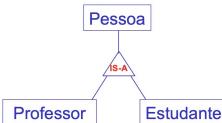


Generalização vs Especialização

- Classificação de entidades em hierarquia de classes

As sub-entidades herdam os atributos das super-entidades.

- Restrições: Sobreposição (overlapping)



- Disjunto: uma entidade só pode pertencer, no máximo, a uma subclasse de especialização

Sobrepõe: uma ocorrência de uma entidade genérica pode ter mais de uma especialização

Compleatude (covering)

- Total: uma entidade de nível superior tem de pertencer a pelo menos uma subclasse de especialização

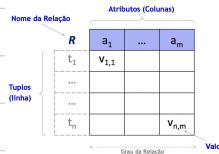
Parcial: pode não pertencer a nenhuma.



Modelo Relacional

- Baseado na noção matemática de "Relação", representadas por **Tabelas**.
- Dispõem de um sistema formal de manipulação das relações - **Álgebra Relacional**

Conceitos: • Base do Modelo Relacional - **Relação (Tabela)**



Atributo (A₁, A₂, ..., A_n): Representam o tipo de dados a armazenar

- O número de atributos de uma relação define o **grafo da relação**
- Nomes distintos

Dominio (D₁, D₂, ..., D_n): • Tipos de dados

- Gamma de valores possíveis para um determinado **atributo** Ex.: São f. "M", "F"
- Valores desconhecidos ou não existentes **NULL**

Esquema da Relação - R (A₁, A₂, ..., A_n): • Nome esquema e lista de atributos **Pessoa (nome, bi, idade)**

- Optionalmente inclui o tipo de dados **Pessoa (nome: string, bi: integer, idade: integer)**

Relação - r (R): • Estrutura bidimensional com determinado **esquema** e zero ou mais **instâncias (tuplos)**. r = {t₁, t₂, ..., t_n}

Tuplo: • Linha de uma relação $t = \langle v_1, v_2, \dots, v_n \rangle$

- Deverão ser distintas

- O nº de tuplos define a **cardinalidade da relação**

Atomicidade: • O valor de um atributo num tuplo é **atómico** (não é composto/multi-valor).

Esquema da BD: • conjunto de todos os esquemas da relação da BD.



Relações - Chaves

- **Superchave:** conjunto de atributos que identificam de forma única os tuplos da relação
- **Chave Candidata:** subconjunto de atributos de uma superchave que não pode ser reduzido sem perder essa qualidade de Superchave

- **Chave Primária:** chave principal selecionada entre os candidatos
- **Chave única:** chave candidata não eleita como primária.
- **Chave Estrangeira:** conjunto de um ou + atributos que é chave primária numa relação.

Superchaves e Chave Candidatas

- Cada relação tem pelo menos uma superchave (conjunto de todos os atributos)
- (Ver Exemplo Slide 11/3)

Restrições de Integridade

Regras que visam garantir a integridade dos dados.

Tipos.

- **Domínio** - dos atributos. Os campos devem obedecer ao tipo de dados e às restrições de valores admitidos para um atributo

- **Entidade** - cada tuplo deve ser identificado de forma única com recurso a uma **chave primária** que não se repete e não pode ser null.
- **Referencial** - o valor de uma **chave estrangeira** ou é null ou contém um valor que é **chave primária** na relação de onde foi importada.

Regras de Codd

Definem/avaliam um sistema de modo relacional

1. Representação da interligação: todos os dados são representados de uma só forma, em tabelas bidimensionais.
2. Acesso garantido: cada elemento de dados fica bem determinado pela combinação do nome da tabela onde está armazenado, valor da chave primária e respectiva coluna (atributo).
3. Suporte sistemático de valores nulos: valores nulos são aceitados para representar interligações não disponíveis ou não aplicáveis.
4. Catálogo ativo e disponível: os metadados são representados e acedidos da mesma forma que os próprios dados.
5. Linguagem completa
6. Regra da atualizações de vistos
7. Operações de alto-nível: capacidade de tratar a tabela como se fosse um simples operando

8. Independência física dos dados: alterações físicas nos ficheiros não deve afetar o nível lógico
9. Independência lógica de dados
10. Restrição de Integridade
11. Independência da localização
12. Não subversão

Conversão do DFD em Modelo Relacional

- Um desenho conceptual de umas BD, utilizando DFD, pode ser representado intermédio de um conjunto de relações (tabelas)

DFD para Relacional

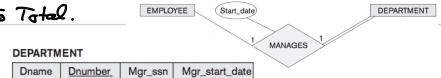
- Passo 1:**
- Para cada entidade regular E do esquema ER, criar uma Relação (tabela) R e incluir todos os atributos de E.
 - Incluir os atributos compostos como elementos singulares.
 - Selecionar uma chave de E para chave primária de R.

- Passo 2:**
- Cada entidade fraca W do esquema ER é representada por uma relação R que inclui os seus atributos, assim como as chaves primárias da entidade dominante E que passará a ser chave estrangeira em R.

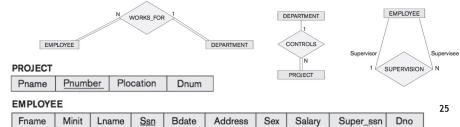
- Incluir os atributos compostos como singulares.
- A chave primária de R é a combinação da chave primária de E e da chave parcial de W.



- Passo 3:**
- Para cada relacionamento 1:1 do esquema ER, envolvendo as relações S e T:
 - escolher uma das relações, digamos S, e incluir como chave estrangeira, a chave primária da outra relação.
 - incluir em S eventuais atributos do relacionamento
 - deveremos escolher como S uma relação com participação Total.



Escolhemos com S a relação DEPARTMENT e incluímos a chave primária de EMPLOYEE como chave estrangeira.



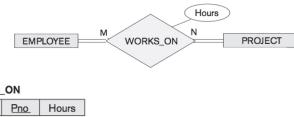
25

Passo 4: Relacionamentos 1:N:

- Escolher como S a relação que representa a entidade lado N e como T a do lado 1.
- Incluir em S, como chave estrangeira, a chave primária de T.
- Incluir os atributos da relação em S.

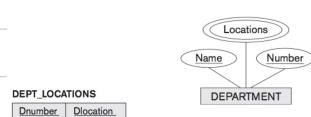
Passo 5: Relacionamento N:M - criar uma nova relação R:

- Incluir como chave estrangeira as chaves primárias das relações que participam em R. Estas chaves combinadas formarão a chave primária da relação R.
- Incluir os atributos da relação R.



Passo 6: Atributo multi-valor A - criar uma nova relação R:

- incluir um atributo correspondente a A.
- incluir a chave primária K da relação que tem A como atributo
- a chave primária de R é a combinação de A e K



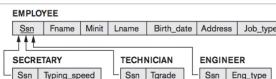
Passo 7: Relacionamento n-ário (n>2):

- Criar uma nova relação R
- incluir, como chaves estrangeira, as chaves primárias das relações que representam as entidades participantes
- incluir os atributos da relação
- a chave primária de R é a combinação das chaves estrangeiras



Especialização

- Formar uma relação L para a entidade de maior nível (C)
- Criar uma relação L_i para cada entidade de nível inferior. Incluir em cada uma destas relações a chave primária de C e os atributos locais.



Funciona com qualquer tipo de especialização: Total/Parcial, Disjunta/Sobreposta.

- Método 2:** • Criar uma relação L_i para cada entidade de nível anterior. Incluir os atributos da superclasse e os atributos locais

$$\text{Attrs}(L_i) = \{\text{atributes of } S_i\} \cup \{k, a_1, \dots, a_n\} \text{ e } \text{PK}(L_i) = k$$

CAR
Vehicle_id
License_plate_no
Price
Max_speed
No_of_passengers

TRUCK
Vehicle_id
License_plate_no
Price
No_of_axles
Tonnage

Só funciona com especialização total.

Só se recomenda em especializações disjuntas pois nas sobreposições há duplicação de informação da mesma entidade por várias relações (tabelas).

Linguagem SQL

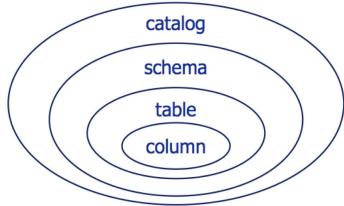
- Linguagem para definir, manipular e questionar uma Base de Dados Relacional

→ 2 sublinguagens principais: **DDL - Data Definition Language**

DML - Data Manipulation Language

→ 1 sublinguagem de controle BD: **DCL - Data Control Language**

Hierarquia de Objetos



- SQL utiliza tabela, linha e coluna para designar os termos formais: relação, tuplo e atributo do modelo relacional

Data Definition Language - DDL

- Utilizada para especificar a informação acerca de cada relação:

- O esquema de cada relação
- O domínio de valores associados com cada atributo.
- Restrições de integridade
- O conjunto de índices a manter para cada relação.

Criar e eliminar uma Base de Dados

```
CREATE DATABASE dbname;  
dbname - nome da base de dados a criar  
CREATE DATABASE COMPANY;
```

```
DROP DATABASE dbname;  
dbname - nome da base de dados a eliminar  
DROP DATABASE COMPANY;
```

Schema

- Namespace que agrupa tabelas e outros elementos pertencentes à mesma aplicação.

```
CREATE SCHEMA schemaname [AUTHORIZATION username];  
CREATE SCHEMA COMPANY AUTHORIZATION 'CCosta';
```

```
DROP SCHEMA schemaname;  
DROP SCHEMA COMPANY;
```

Tipo de Dados

- Os tipos de dados podem variar de acordo com SQL Server.
- Dados mais usados:
 - `char(n)` - cadeia de caracteres de tamanho fixo n
 - `varchar(n)` - cadeia de caracteres com tamanho máximo n
 - `int` - números inteiros (4 bytes)
 - `numeric(precisão, escala)` - números reais "sem limite" de tamanho
 - `date / time` - data e hora
 - `boolean` - valores booleanos

⚠ não existe em
SQL Server

Definição de Domínio

- `create domain` permite definir novos tipos de dados.
- Um domain pode conter um valor de defalut e restrição do tipo `not null` e `check` (não disponível em SQL Server)

```
CREATE DOMAIN domainname  
Criação...  
CREATE DOMAIN compsalary INTEGER  
    NOT NULL CHECK (compsalary > 475);  
  
Utilização...  
CREATE TABLE EMPLOYEE  
(...  
Salary           compsalary,  
...);
```

Definição de Novo Tipo

- Como alternativa ao domain, podemos criar só um novo tipo (alias) com o comando `create type`.

Nota: Em geral, é mais limitado que o `create domain`.

```
CREATE Type... em SQL SERVER  
Criação...  
CREATE TYPE SSN FROM varchar(9) NOT NULL;  
  
Utilização...  
CREATE TABLE EMPLOYEE  
(...  
SSN           SSN,  
...);
```

Criar uma Tabela

```
CREATE TABLE tbname ( A1 D1, A2 D2, ..., An Dn,
                      (integrity-constraint1),
                      ...
                      (integrity-constraintK) );
```

tbname - nome da relação (tabela)

```
CREATE TABLE COMPANY.EMPLOYEE (...)
```

```
CREATE TABLE EMPLOYEE (...)
```

COMPANY - nome do schema

A1 D1, A2 D2, ..., An Dn
A1...An - Atributos da relação
D1...Dn - Domínios dos atributos

Restrições de Integridade
integrity-constraint1,
...,
integrity-constraintN

CREATE TABLE...
definindo atributos e respectivo domínio.

```
CREATE TABLE EMPLOYEE (
    Fname          VARCHAR(15),
    Minit          CHAR,
    Lname          VARCHAR(15),
    Ssn            CHAR(9),
    Bdate          DATE,
    Address        VARCHAR(30),
    Sex             CHAR,
    Salary          DECIMAL(10,2),
    Super_ssn      CHAR(9),
    Dno             INT);
```

- Podem ser definidas valores por omissão para cada coluna, utilizando o termo "default".

```
CREATE com default ...
```

```
CREATE TABLE EMPLOYEE (
    Fname          VARCHAR(15),
    ...
    Salary          DECIMAL(10,2)  DEFAULT 0,
    ...
    Dno             INT);
```

Restrições de Integridade

- check(p) → impor uma regra a um atributo
- primary Key (A₁, ..., A_n) → definir chave primária
- foreign Key → definir chave estrangeira

• not null → atributo não pode ser null

• unique (A₁, ..., A_n) → chaves candidatas não primárias

As restrições podem ser de:
→ **coluna**: referem-se a apenas uma coluna e são descritas em frente à coluna;
→ **tabela**: referem-se a mais do que uma coluna e ficam separadas da definição das colunas.

Restrição check → restrição aplicada a cada atributo referenciado sempre que um tuplo é introduzido ou modificado

Restrição CHECK na coluna...

```
CREATE TABLE EMPLOYEE (
    ...
    Salary          DECIMAL(10,2)  CHECK (Salary > 12),
    ...);
```

Restrição CHECK na tabela...

```
CREATE TABLE DEPARTMENT (
    ...
    Dept_create_date   DATE           NOT NULL,
    Mgr_start_date     DATE,
    ...
    CHECK (Dept_create_date <= Mgr_start_date);
```

Restrição Primary Key → Só podemos definir uma primary key na tabela.

```
Restrição PRIMARY KEY na coluna...
CREATE TABLE EMPLOYEE (
    ...
    Ssn      CHAR(9)          PRIMARY KEY,
    ...);
```

```
Restrição PRIMARY KEY na tabela...
(obrigatório se PK for composta por mais do que um atributo)
CREATE TABLE EMPLOYEE (
    ...
    Ssn      CHAR(9),
    ...
    PRIMARY KEY (Ssn));
```

Restrição Unique → Utilizada para as chaves candidatas alternativas - não pode ter valores repetidos mas pode conter valores null.

```
Restrição UNIQUE na coluna...
CREATE TABLE DEPARTMENT (
    Dname    VARCHAR(15)  UNIQUE NOT NULL,
    Dnumber   INT          NOT NULL,
    PRIMARY KEY (Dnumber),
    ...);
```

```
Restrição UNIQUE na tabela...
CREATE TABLE DEPARTMENT (
    Dname    VARCHAR(15)  NOT NULL,
    Dnumber   INT          NOT NULL,
    PRIMARY KEY (Dnumber),
    UNIQUE (Dname, ...);
```

Restrição Foreign Key → Declara chaves estrangeiras

Uma chave estrangeira deve referenciar numa chave única ou primária

```
Restrição FOREIGN KEY na coluna...
CREATE TABLE EMPLOYEE (
    ...
    Super_ssn  CHAR(9)  REFERENCES EMPLOYEE(Ssn),
    Dno        INT       REFERENCES DEPARTMENT(Dnumber) NOT NULL,
    ...);
```

```
Restrição FOREIGN KEY na tabela...
CREATE TABLE EMPLOYEE (
    ...
    Ssn      CHAR(9),
    Dno      INT          NOT NULL,
    ...
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber));
```

• Podemos definir as seguintes ações alternativas: "on delete" e "on update", com as seguintes opções:

• **restrict** → não deixa efectuar a operação

• **cascade** → apaga os registas associados (delete) ou altera a chave estrangeira (update)

• **set null** → a chave estrangeira passa ser null

• **set default** → a chave estrangeira passa a ter o valor por omissão.

```
Restrição FOREIGN KEY
CREATE TABLE EMPLOYEE (
    ...
    Ssn      CHAR(9),
    Dno      INT          NOT NULL,
    ...
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Se o tuplo do supervisor é eliminado, a coluna Super_ssn dos supervisionados passa automaticamente a Null.

Se o Ssn do supervisor é actualizado, a coluna Super_ssn dos supervisionados é actualizada em cascata.

Restrição - atribuição de nome → Como referenciar uma restrição? "Babitar" a restrição com nome próprio.

Restrições com nome...

```
CREATE TABLE EMPLOYEE (
    ...
    ...
    CONSTRAINT EMPPK
        PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (Supер_ssn) REFERENCES EMPLOYEE(Ssn)
            ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
            ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Tabela - Drop

- O comando **drop Table** remove da base de dados toda a informação sobre a tabela e os dados

• Eliminar a tabela EMPLOYEE
não disponível
em SQL Server

```
DROP TABLE EMPLOYEE;
```

- A opção **CASCADE** permite eliminar a tabela e os elementos referenciados na restrição

• Eliminar a tabela EMPLOYEE com opção CASCADE
DROP TABLE EMPLOYEE CASCADE;

Tabela - Alter

- O comando **alter Table** é utilizado para modificar o esquema da tabela ou restrição existentes.

- Adicionar atributos à tabela:

```
ALTER TABLE tablename ADD Attribute Domain
```

```
ALTER TABLE EMPLOYEE ADD nofiscal INT;
```

- Todos os tuplos existentes ficam com valor null no novo atributo.

- Adicionar restrições à tabela:

```
ALTER TABLE tablename ADD CONSTRAINT name theconstraint
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT salarymin CHECK (Salary >475);
```

- Eliminar atributos da tabela:

```
ALTER TABLE tablename DROP COLUMN attributename
```

```
ALTER TABLE EMPLOYEE DROP COLUMN nofiscal;
```

- Eliminar restrições da tabela:

```
ALTER TABLE tablename DROP CONSTRAINT name
```

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT salarymin;
```

- Alterar um atributo de uma tabela:

```
ALTER TABLE tablename ALTER Attribute Domain
```

```
ALTER TABLE EMPLOYEE ALTER COLUMN noFiscal CHAR(9);
```

Algebra Relacional

- Linguagem de Consulta/Interrogação de BD
- Permite formular pedidas básicas de recuperação de informação (retrieval requests) sobre uma ou mais relações

Seleção



Expressão Booleana

- Utilizada para selecionar um subconjunto de tuplas da relação que satisfazem os critérios.

- Resultado é uma nova relação que tem um esquema relacional igual à original.

- Notação: $\sigma_{\text{selection condition}}(R)$

Operadores de Comparação

Permitem comparar dois atributos ou um atributo com um valor $=, \neq, >, \geq, <, \leq$

Operando: Nomes dos atributos e constantes

$\sigma_{Dno} = 4 (\text{EMPLOYEE})$

Condições Booleanas AND, OR, e NOT

Projeção



- O resultado é uma nova relação só com os K atributos selecionados

- São removidas as linhas duplicadas do resultado.

- Notação: $\Pi_{\text{attribute list}}(R)$

- $\text{attribute list} = A_1, A_2, \dots, A_K \hookrightarrow \text{Nome dos atributos}$

Renomeação

- Notação: $\rho_{R2(B1, B2, \dots, Bn)}(R1)$ ou $\rho_{R2}(R1)$ ou $\rho_{(B1, B2, \dots, Bn)}(R1)$
 - No primeiro caso o resultado é uma nova relação R2 com os atributos renomeados
 - No 2º Caso só renomearmos a relação
 - No 3º Caso só renomearmos os atributos.

União

- As tabelas têm de ser compatíveis • Mesmo nº de atributos e atributos com domínios compatíveis
- O resultado é uma relação que inclui todas as tuplas de R e S *Tuplos duplicados são eliminados*



STUDENT

Fname	Lastname
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

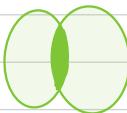
Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

U →

Fname	Lastname
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

Intersecção

- As tabelas têm de ser compatíveis • Mesmo nº de atributos e atributos com domínios compatíveis
- O resultado é uma relação que inclui os tuplos que existem simultaneamente em R e S *Tuplos duplicados são eliminados*



STUDENT

Fname	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

∩ →

Fname	Ln
Susan	Yao
Ramesh	Shah

Diferença

- As tabelas têm de ser compatíveis • Mesmo nº de atributos e atributos com domínios compatíveis
- O resultado é uma relação que inclui os tuplos de R que não existem em S



STUDENT

Fname	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

-

→

Fname	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

Em SQL existem os seguintes comandos: `UNION (ALL)` `INTERSECT (ALL)` `EXCEPT (ALL)`

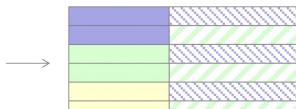
Propriedades:

- União e Intersecção são operações comutativas:
 - $R \cup S = S \cup R$ e $R \cap S = S \cap R$
- A diferença não é comutativa:
 - $R - S \neq S - R$
- União e Intersecção são operações associativas:
 - $R \cup (S \cup T) = (R \cup S) \cup T$ e $(R \cap S) \cap T = R \cap (S \cap T)$

Produto Cartesiano



RXS



- Permite-nos combinar tuplos de relações diferentes.
- O resultado é uma nova relação (Q) que combina cada elemento (tuplo) de uma relação (R) com um elemento de outra relação (S):

Notação: RXS

Junção θ (Theta JOIN) $R \bowtie_c S$

- Pode ser visto como a combinação do produto cartesiano com uma seleção.

▪ Pode ser visto como o resultado das seguintes operações:

$$R3 \leftarrow R1 \times R2 \quad (\text{produto cartesiano})$$

$$\sigma_c(R3) \quad (\text{seleção com condição } c)$$

▪ C é \langle join condition \rangle que pode tomar a seguinte forma:
 \langle condition \rangle AND \langle condition \rangle AND ... AND \langle condition \rangle

▪ Em cada \langle condition \rangle podemos aplicar operadores de comparação:
 $=, <, \leq, >, \geq, \neq$

• Pretendemos saber os nomes dos funcionários gestores de departamentos

Name	Mgr	Locate	Rep	Dept	Address	Sal	Salary	Super_id	Obc
John	B. Smith	Research	464-101-1234	Research	Houston, TX	30000	30000	0	A
Albert	C. Thomas	Administration	464-101-1235	Administration	Houston, TX	30000	30000	0	B
Allen	J. Zane	Research	464-101-1236	Research	Seattle, WA	30000	30000	0	C
Renata	K. Nancy	Marketing	464-101-1237	Marketing	Seattle, WA	30000	30000	0	D
Anita	A. Lynn	Marketing	464-101-1238	Marketing	Houston, TX	30000	30000	0	E
Patricia	G. Berg	Marketing	464-101-1239	Marketing	Houston, TX	30000	30000	0	F
James	E. Frank	Marketing	464-101-1240	Marketing	Houston, TX	30000	30000	0	G

Para obter o nome dos gestores temos de combinar cada tuplo do departamento (Department) com um tuplo dos funcionários (Employee) cujo Ssn é igual ao Mgr_ssn.

$$\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr_ssn}} \text{EMPLOYEE}$$

Deptname	Deptno	Locname	Mgr_ssn	Deptname	Mgr_ssn	Locname	Mgr_ssn
Research	5	Houston	0	Research	0	Houston	0
Administration	4	Houston	0	Administration	0	Houston	0
Headquarters	1	Houston	0	Headquarters	0	Houston	0

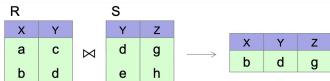
Depois só temos de utilizar projeção para obter os atributos desejados:
 $\text{RESULT} \leftarrow \Pi_{\text{Dname}, \text{Lname}, \text{Fname}}(\text{DEPT_MGR})$

19

Junção Natural (Natural Join): $R \bowtie S$

→ **Condicion Implícito:** igualdade das atributos com o mesmo nome

→ Atributos repetidos são removidos



PROJECT	Deptname	Deptno	Locname	Projectname	Deptname	Deptno	Locname	Projectname
ProductX	1	Baltimore	1	ProductX	1	Baltimore	1	ProductX
ProductY	2	Edinburgh	2	ProductY	2	Edinburgh	2	ProductY
ProductZ	3	Houston	3	ProductZ	3	Houston	3	ProductZ
Computerization	4	London	4	Computerization	4	London	4	Computerization
Reorganization	20	Stafford	1	Reorganization	20	Stafford	1	Reorganization
Newbenefits	30	Stafford	4	Newbenefits	30	Stafford	4	Newbenefits

PROJECT	Deptname	Deptno	Locname	Deptname	Deptno	Locname	Deptname	Deptno
ProductX	1	Baltimore	1	ProductX	1	Baltimore	1	ProductX
ProductY	2	Edinburgh	2	ProductY	2	Edinburgh	2	ProductY
ProductZ	3	Houston	3	ProductZ	3	Houston	3	ProductZ
Computerization	4	London	4	Computerization	4	London	4	Computerization
Reorganization	20	Stafford	1	Reorganization	20	Stafford	1	Reorganization
Newbenefits	30	Stafford	4	Newbenefits	30	Stafford	4	Newbenefits

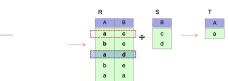
21

Divisão: $R \div S$

- Dadas as relações $R(A_1, \dots, A_r, B_1, \dots, B_k)$ e $S(B_1, \dots, B_k)$

→ O resultado incluirá todas as tuplas da $R(A_1, \dots, A_r)$ que tenham correspondências com todos os tuplos de S em $R(B_1, \dots, B_k)$.

- Em SQL não existe um operador que implemente a divisão temos de recorrer a operadores básicos.



Department	Dept_name	Location	Location	Dept_name	Name
1	Research	Houston			
2	Commercial	Houston			
3	Administration	LA			
2	Commercial	Houston			
4	Headquarters	Seattle			
3	Commercial	LA			

Departamentos que existem em todas as localizações?

OPERATION	PURPOSE	NOTATION
RESTRICT	Retain the tuples which satisfy certain conditions.	$R _{P_i}$
PROJECT	Produce a relation with only certain attributes.	$\Pi_{A_1, \dots, A_r}(R)$
SELECT	Selects a relation with only certain tuples.	$\sigma_{P_i}(R)$
JOIN	Joins two relations together based on common attributes.	$R \bowtie S$
EXISTS JOIN	Joins two relations together based on common attributes, and the result will have at least one tuple.	$R \bowtie_{\exists} S$
NESTED JOIN	Joins two relations together based on common attributes, and the result will have exactly one tuple.	$R \bowtie_{\forall} S$
MINUS JOIN	Joins two relations together but the join attributes of R must have the same values as the join attributes of S.	$R \bowtie_{\text{minus}} S$
UNION	Combines two relations into one.	$R \cup S$
INTERSECTION	Produces a relation that includes only the tuples that are common to both R and S.	$R \cap S$
DIFFERENCE	Produces a relation that includes only the tuples that are in R but not in S.	$R - S$
CROSS PRODUCT	Produces a relation that has all the attributes of R and S.	$R \times S$
DIVISION	Produces a relation that includes all tuples of R that have a common value for every tuple from S, where $R \geq S$.	$R \div S$

Operações Estendidas

Semi Join

- Left Semi Join: $R \ltimes S = \Pi_R(R \bowtie S)$

Projeção dos atributos de R na junção natural de R com S

R		S		
X	Y	Y	Z	
a	c	d	g	
b	d	e	h	

× →

X	Y
a	c
b	d

- Right Semi Join: $R \rtimes S = \Pi_S(R \bowtie S)$

Projeção dos atributos de S na junção natural de R com S

R		S		
X	Y	Y	Z	
a	c	d	g	
b	d	e	h	

× →

Y	Z
d	g
e	h

Inner Join vs Outer Join

- As operações de junção anteriores combinam dados de duas tabelas para que entes possam ser apresentados na forma de uma única tabela.
- Os tuplos que não estão relacionados são descartados.

Outer Join

- Incluímos no resultado todos os tuplos de uma (ou de ambas) das relações componentes.
- Os atributos que não fizerem matching são preenchidos com Null

- Left Outer Join: $R \bowtie S$

R		S		
A1	A2	B1	B2	
a	c	d	g	
b	d	e	h	

$\bowtie_{A_2 > B_1} \rightarrow$

A1	A2	B1	B2
a	c	null	null
b	d	d	g

- Right Outer Join: $R \bowtie S$

R		S		
A1	A2	B1	B2	
a	c	d	g	
b	d	e	h	

$\bowtie_{A_2 < B_1} \rightarrow$

A1	A2	B1	B2
b	d	d	g
null	null	e	h

- Full Outer Join: $R \bowtie S$

R		S		
A1	A2	B1	B2	
a	c	d	g	
b	d	e	h	

$\bowtie_{A_2 = B_1} \rightarrow$

A1	A2	B1	B2
a	c	null	null
b	d	d	g
null	null	e	h

Join - Quadro Resumo

- Natural | Left Outer | Right Outer | Full Outer

R		S		
X	Y	Y	Z	
1				
2		a		
3		b		
4		c		
5		d		

→

S	
Y	Z

All Joins			
X	Y	Z	
1		null	
2	a		
3		null	
4	b		
4	c		
5		null	
null	d		

R ... S			
⋈	⋈	⋈	⋈
x	v	x	v
v	v	v	v
x	v	x	v
v	v	v	v
v	v	v	v
x	v	x	v
x	x	v	v

Agregação

\exists $\langle \text{grouping attributes} \rangle \exists \langle \text{function list} \rangle (R)$

• Operações sobre vários tuplos da relação

- Lista de Funções de Agregação:
 - **avg:** média dos valores
 - **min:** mínimo dos valores
 - **count:** número dos valores
 - **max:** máximo dos valores
 - **sum:** soma dos valores

- Também podem ser usados em projeções
 - criar atributos agregados
 - os atributos não agregados são agrupados de forma a não haver valores repetidos.

EMPLOYEE

First	Minit	Lastname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelby	999887777	1968-01-19	3321 Castle Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-09-20	291 Berry, Bellare, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmed	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

$\exists \text{ count}(Ssn), \text{ avg}(Salary)(EMPLOYEE)$

Count_ssn

Average_salary

8

35125

Dno $\exists \text{ count}(Ssn), \text{ avg}(Salary)(EMPLOYEE)$

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

$P_R(Dno, \text{No_of_employees}, \text{Average_sal}) (Dno \exists \text{ count}(Ssn), \text{ avg}(Salary)(EMPLOYEE))$

R
Dno
No_of_employees
Average_sal

5 4 33250

4 3 31000

1 1 55000