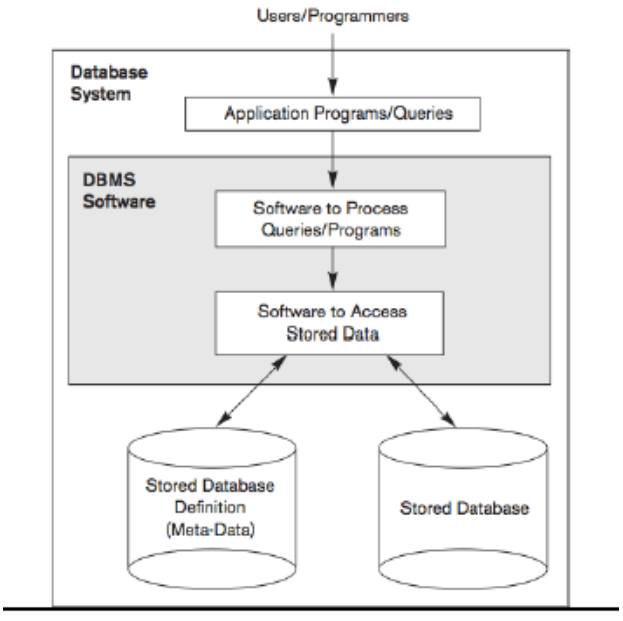
BASE DE DADOS – Mariana Pinto, nmec 84792

Introdução

- ✓ **Base de dados ->** coleção organizada de dados que estão relacionados e que podem ser partilhados por múltiplas aplicações.
- ✓ **Processamento isolado de dados:**
  - **Dados isolados –** cada aplicação gere os seus próprios dados;
  - Os mesmos dados podem estar replicados;
  - Diferentes organizações e formatos de dados;
  - Problemas de "sincronismo" -> traz incoerências.
- ✓ **Sistema de Gestão de Ficheiros:**
  - Dados organizados e armazenados em ficheiros partilhados por várias aplicações.
  - Cada aplicação acede diretamente aos ficheiros;
  - Cada aplicação usa uma interface proprietária;
  - Problemas de acesso concorrente aos dados;
  - Problemas de integridade;
  - Problemas de segurança.
- ✓ **Sistema gestão de base de dados (SGBD)/Database Management System (DBMS):** software de propósito geral que facilita os processos de definir, construir, manipular e partilhar bases de dados entre vários utilizadores e aplicações.



- **Definição (defining):** especificação do tipo de dados, estruturas de dados e restrições – database catalog ou dicionário
- **Construção (constructing):** processo de armazenamento de dados
- **Manipulação:** envolve operações como a pesquisa e obtenção de dados
- **Partilha (Sharing):** acesso simultâneo aos dados por parte de vários utilizadores e programas
- Entidade única que opera com a BD – o acesso à BD é sempre mediado pelo SGBD
- Existe uma interface de acesso que esconde os detalhes de armazenamento físico dos dados
- Elevada abstração ao nível aplicacional
- Os dados estão integrados (nível lógico) numa mesma unidade de armazenamento
- Suporta uma ou mais BD
- **Keyword** -> Data independence
- **Vantagens SGBD**: (muitas destas vantagens são requisitos funcionais de um SGBD)
    - Independência entre programas e dados
    - Integridade dos dados -> controlo de alteração 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 -> especialmente em cenários de manipulação de grandes quantidades de dados, por um ou mais utilizadores
    - Isolamento utilizadores -> cada utilizador tem a “sensação” de ser único
    - Melhor gestão do acesso concorrencial
    - Serviços de segurança -> controlo de acesso/permissões e codificação de dados
    - Mecanismos de backup e recuperação de dados

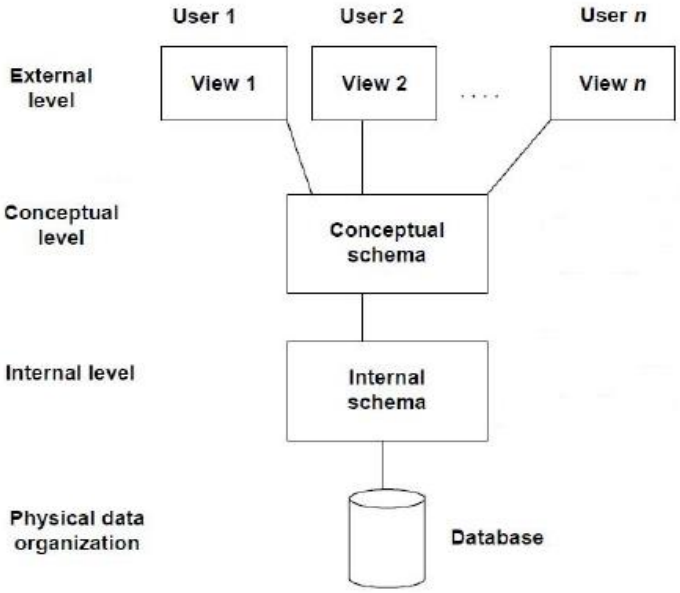
        - Administração de dados -> disponibilidade de ferramentas desenvolvidas pelo fabricante e/ou terceiras entidades
        - Linguagem de desenho e manipulação de dados
    - Desvantagens:
        - Maiores custos e complexidade na instalação e manutenção -> especial em soluções empresariais
        - Não respondem aos requisitos de alguns cenários aplicacionais como, por exemplo, pesquisa de texto
        - Centralização dos dados mais suscetível a problemas de tolerância a falhas (Software e Hardware) e de escalabilidade
    - **Utilizadores finais:** utilizadores finais, programadores de aplicações e administradores da BD
    - Dicionário de dados:
        - O SGBD contém BD mas também informação relativa à definição da própria estrutura da BD, incluindo as restrições - Metadados (dados sobre dados)
        - Um dicionário contém:
            - Descritores de objetos da BD (tabelas, utilizadores, regras, vistas, indexes, etc)
            - Informação sobre dados em uso e por quem (locks)
            - Schemas e mappings

✓ **Interfaces:**

- **Web-based**
- **Form-based (desktop)**
- **GUI (graphical user interface**) -> Manipulação visual de esquemas de BD com recurso a diagramas. Possibilidade de construção e execução de queries.
- **Natural Query Language**
- **DBMS Command Line**
    - Criar contas de utilizadores, parametrizar os sistemas, definir permissões e privilégios, definir/alterar estruturas de dados, definir tipos de dados, etc.

- Utilizando uma linguagem própria – SQL

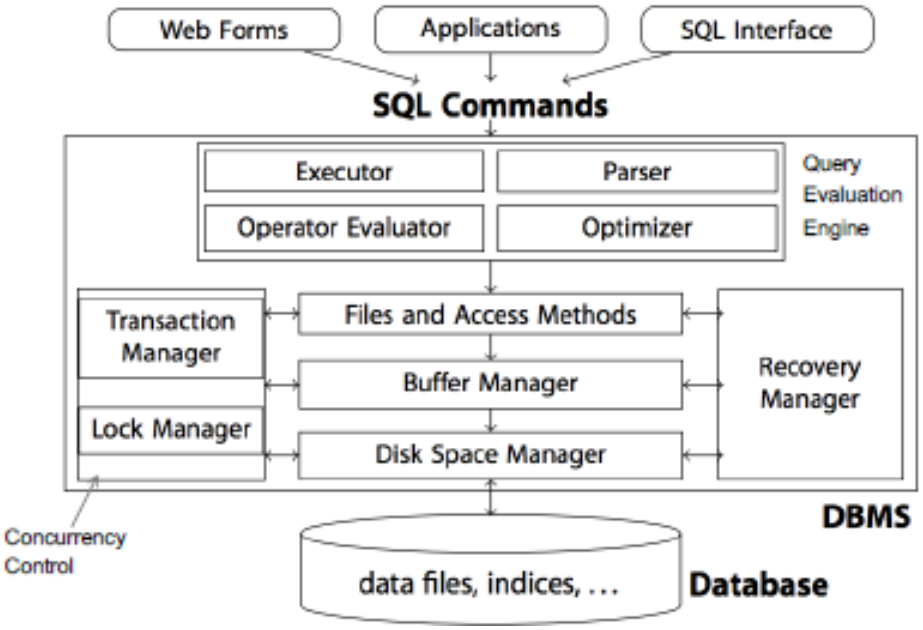
✓ SGBD – Arquitetura ANSI/SPARC : Three-level architecture



- External level -> database users:
  - Oferece vistas da BD adaptadas a cada utilizador:
    - Apresentação dos dados pode ser trabalhada, parte dos dados pode ser ocultada, etc.
  - Domínio: Utilizadores finais e prog. de aplicações
- Conceptual level -> database designers and administrators:
  - Descreve a estrutura da base de dados para os utilizadores
    - Descreve entidades, tipos de dados relações, operações, restrições, etc.
    - Utiliza (tipicamente) um modelo de dados para descrição do esquema conceptual
  - Oculta detalhes de implementação física (Abstração)
  - Domínio: Administrador BD e prog. de aplicações
- Internal level -> systems designers
  - Lida com a implementação física da BD
    - Estrutura dos registos em disco
    - Indexes e ordenação dos registos
  - Domínio: Programadores de sistemas de BD

- Alteração do esquema (schema) de um nível não tem impacto no esquema do nível acima -> Dois níveis de independência:
  - Nível físico:
    - Alterações do nível físico não devem ter impacto no esquema conceptual – ex: podemos alterar a forma como armazenamos os dados no sistema de ficheiros por razões de desempenho.
  - Nível lógico:
    - Alterações no esquema conceptual (modelo de dados) não devem repercutir-se nos esquemas externos ou aplicações já desenvolvidas

✓ SGBD – Arquitetura Típica:

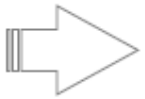


✓ **Modelo base de dados:** coleção de conceitos para descrição lógica de dados (Modelo lógico)

- **Esquema (Schema):** a descrição de um conjunto particular de dados com recurso a um determinado modelo
- Um bom modelo de dados é fundamental para garantir a independência dos dados
- O modelo relacional é um dos mais utilizados nos dias de hoje.

✓ Modelos de BD:

- 1ª Geração (Pré-relacional)
  - Hieráquico
  - Rede
- 2ª Geração
  - Relacional
- 3ª Geração (Pós-relacional)
  - Object-relational
  - Object-oriented
  - Key-value store
  - Document-oriented
  - Column-oriented
  - Graph database

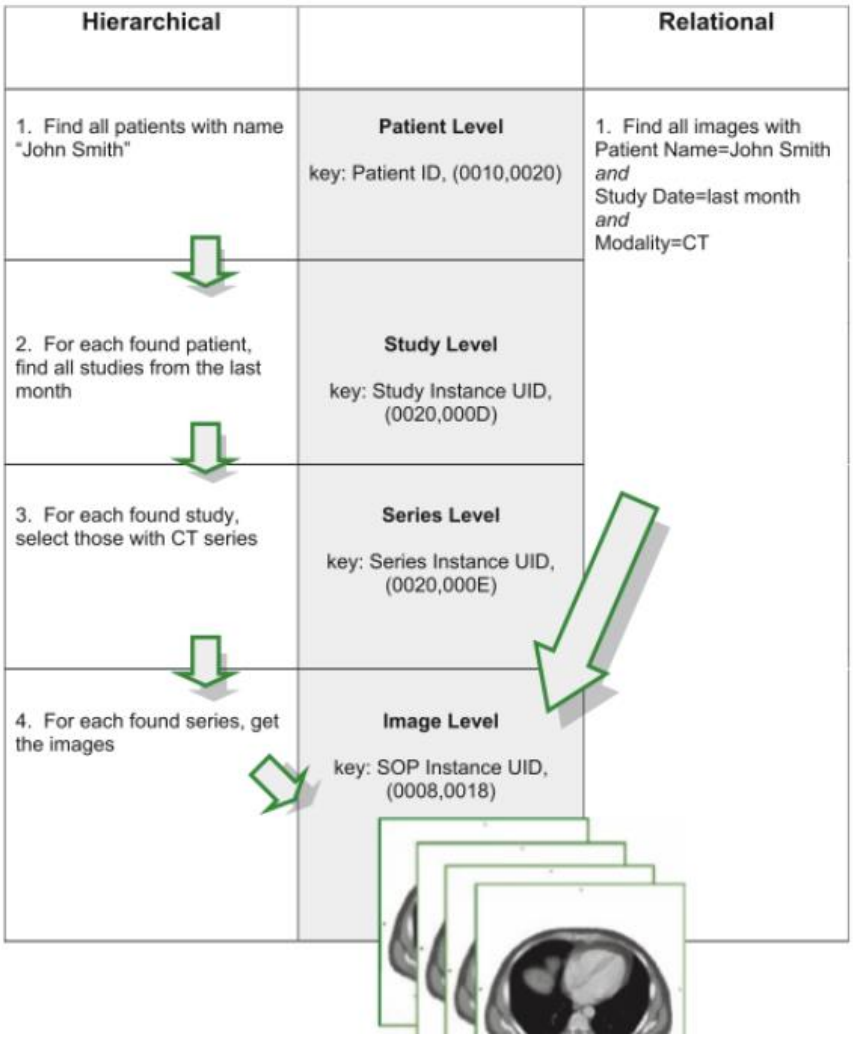


Disciplina de Base de Dados

- Modelo hierárquico:
  - Dados estão armazenados numa estrutura hierárquica (árvore)
  - Os nós da árvore designam-se como registos que estão ligados por ponteiros (links)
  - Um registo é composto por um conjunto de atributos
  - Um link é uma associação entre dois registos do tipo pai-filho
  - Um registo pai encontra-se associado a N registos filhos (1:N)
  - Desvantagens:
    - Adaptado a cenários de acesso sequencial aos dados:
      - Qualquer acesso aos dados passa sempre pelo segmento de raiz
      - A maior parte das necessidades atuais requer acesso aleatório
    - Redundância de informação
      - Desperdício de espaço e inconsistência de dados
    - Restrição de integridade
      - Exemplo: a eliminição 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 -> visão de rede
  - Melhorias na capacidade de navegação na estrutura de dados
  - Relações representadas através de grafos
  - Um conjunto (SET) suporta associação entre registos do mesmo tipo
    - Tipicamente implementados com listas ligadas circulares
  - Relacionamento 1:N ente dois tipos de registo
- **Modelo relacional vs. Modelo Hierárquico**



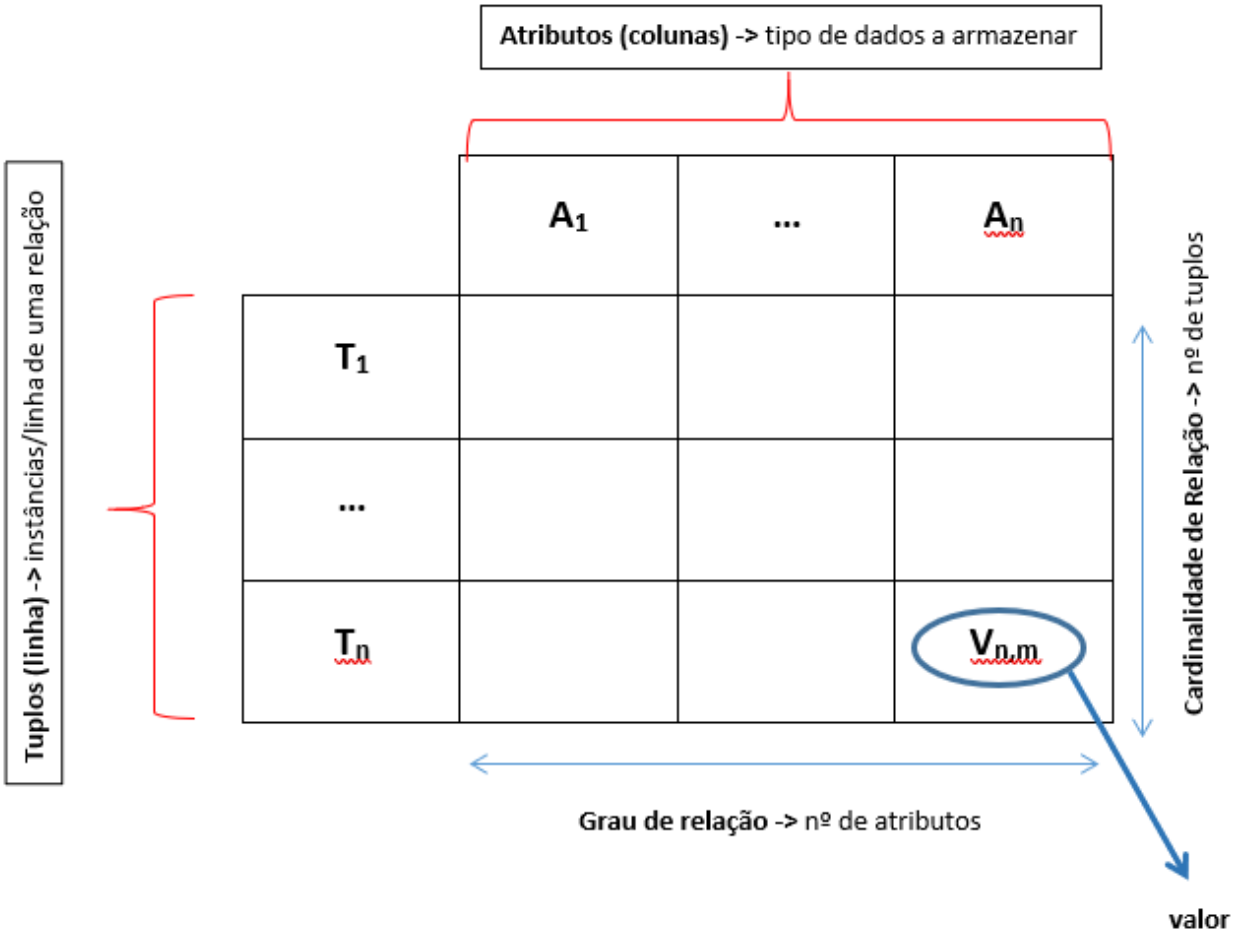
BD2 – Desenho Conceptual

- ✓ **Análise de Requisitos:**

Modelo Relacional

- ✓ **Modelo Relacional ->** baseado na noção matemática de "Relação", representadas por tabelas.

Conceitos:

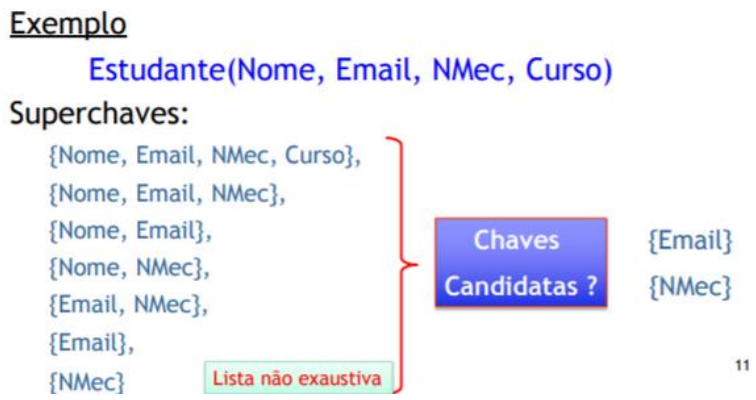


Base do Modelo Relacional – Relação (Tabela)

- ✓ **Domínio ->** tipo de dados, gama de valores possíveis para determinado atributo
- ✓ **Esquema de Relação ->** $R(A_1, \ldots, A_n)$: nome do esquema e lista de atributos
- ✓ **Relação ->** r(R): estrutura bidimensional com determinado esquema e zero ou mais instâncias (tuplos)
- ✓ **Atomacidade ->** valor de um atributo num tuplo é atómico

Relação – Chaves

- ✓ **Superchave (superkey) ->** conjunto de atributos que identificam de forma única os tuplos da relação (Cada relação tem pelo menos 1)



- ✓ **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 as chaves candidatas, valor que raramente é alterado
- ✓ **Chave única ->** chave candidata não eleita como primária
- ✓ **Chave estrangeira ->** conjunto de um ou mais atributos que é chave primária noutra relação

Restrições de Integridade -> regras que visam garantir a integridade dos dados

Tipos:

- ✓ **Domínio –** dos **atributos**. Forma mais elementar de integridade. 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**.
- ✓ **Referencia –** o **valor** de uma chave estrangeira ou é **null** ou contém um valor que é **chave primária** na relação de onde foi importada.

Regras para ver se é ou não relacional (Regras de Codd) – 12 regras

1. **Representação da informação –** numa BD relacional, todos os dados, incluindo o próprio dicionário de 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 respetiva coluna (atributo)
3. **Suporte sistemático de valores nulos (NULL) –** valores NULL são suportados para representar informação não disponível ou não aplicável, independentemente do domínio dos respetivos atributos.
4. **Catálogo ativo e disponível –** os meta dados são representados e acedidos da mesma forma que os próprios dados
5. **Linguagem completa –** apesar de um sistema relacional poder suportar várias linguagens, deverá existir pelo menos uma linguagem com as seguintes caraterísticas:
    a. Manipulação de dados, c/ possibilidade de utilização interativa/programas de aplicação;
    b. Definição de dados, views, restrições de integridade e acessos (autorizações)
    c. Manipulação de transações
6. **Regra da atualização de vistas (view) –** numa vista, todos os dados modificados (em atributos actualizáevis) devem ver essas modificações traduzidas nas tabelas base.
7. **Operações de alto nível –** capacidade de tratar uma tabela como se fosse um simples operando – utilização de linguagem set-oriented – tanto em operações de consulta como de atualização ou eliminação
8. **Independência física dos dados –** alterações na organização física dos ficheiros da base ou nos métodos de acesso a esses ficheiros (nível interno) não devem afetar o nível lógico
9. **Independência lógica dos dados –** alterações no esquema da base de dados (nível lógico), que não envolvam remoção de elementos, não devem afetar o nível externo.

10. **Restrições de integridade –** restrições de integridade devem poder ser especificadas numa linguagem relacional, independentemente dos programas de aplicação, e armazenadas no dicionário de dados.
11. **Independência da localização –** facto de uma base de dados estar centralizada numa máquina, ou distribuída por várias máquinas, não deve repercutir-se ao nível da manipulação dos dados
12. **Não subversão –** se existir no sistema uma linguagem de mais baixo-nível (tipo record-oriented), ela não deverá permitir ultrapassar as restrições de integridade e segurança

Conversão do DER em Modelo Relacional

Passo 1: entidade regular

- ✓ Para cada entidade regular E do esquema ER, criar uma relação (tabela) R que inclui todos os atributos de E
- ✓ Incluir os atributos compostos como elementos singulares

- ✓ Selecionar uma das chaves de E para chave primária de R

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

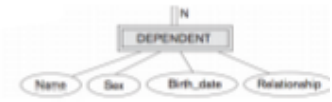
Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Passo 2: entidade fraca

- ✓ Cada entidade fraca W do esquema ER é representada por uma relação (tabela) R que inclui os seus atributos, assim como a chave primária da entidade dominante E que passará a ser chave estrangeira em R
- ✓ Incluir os atributos compostos de W, caso existam, como elementos singulares
- ✓ A chave primária de R é a combinação da chave primária de E e da chave parcial de W.

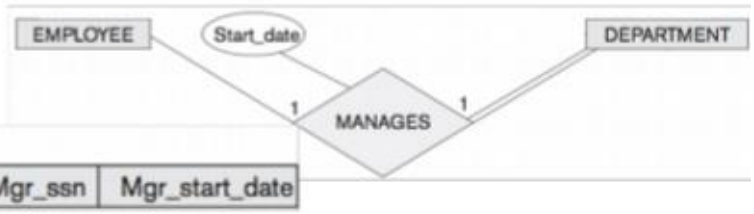


DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Passo 3: Relacionamentos 1:1

- ✓ Escolher uma das relações, digamos S, e incluir como chave estrangeira, a chave primária da outra relação
- ✓ Incluir em S eventuais atributos do relacionamento
- ✓ Devemos escolher como S uma relação com participação total



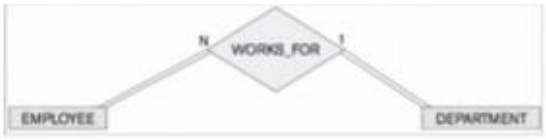
DEPARTMENT

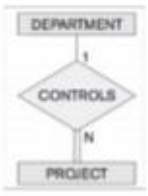
Dname	Dnumber	Mgr_ssn	Mgr_start_date

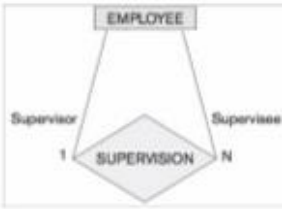
Passo 4: Relacionamento 1:N

- ✓ Escolher como S a relação que representa a entidade do lado N e como T a entidade do lado 1
- ✓ Incluir em S, como chave estrageira, a chave primária da Relação T

- ✓ Incluir os atributos do relacionamento em S







PROJECT

Pname	Pnumber	Plocation	Dnum

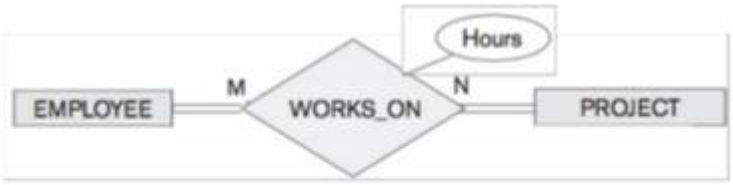
EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

25

Passo 5: Relacionamento N:M

- ✓ Criar uma nova relação (tabela R):
  - Incluir como chave estrageira as chaves primárias das relações que participam em R. Estas chaves combinadas formarão a chave primária da relação R.
  - Incluir os atributos do relacionamento em R.

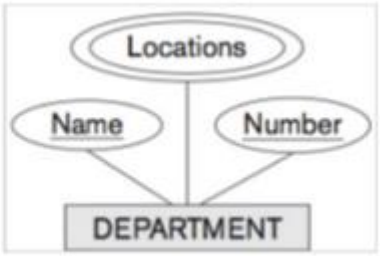


WORKS_ON

Essn	Pno	Hours

Passo 6: Atributo multi-valor

- ✓ Criar uma nova relação (tabela) R:
  - Incluir um atributo correspondendo 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

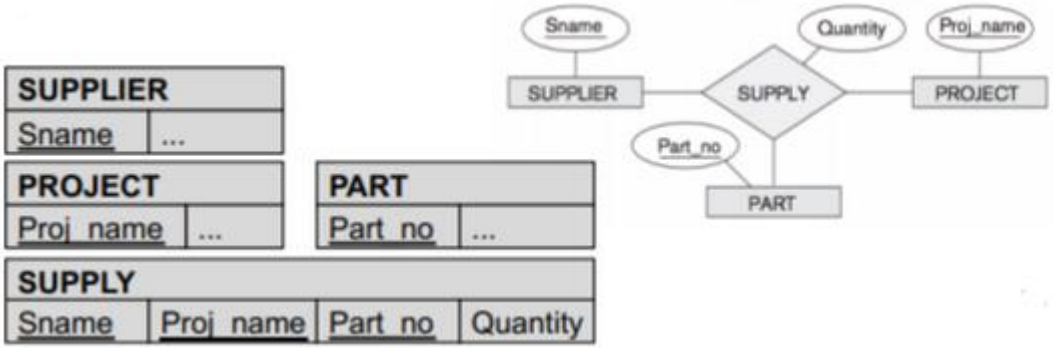


DEPT_LOCATIONS

Dnumber	Dlocation

Passo 7: Relacionamento n-ário (n>2)

- ✓ Criar uma nova relação (tabela) R;
- ✓ Incluir, como chaves estrangeiras, as chaves primárias das relações que representam as entidades participantes
- ✓ Incluir os eventuais atributos do relacionamento;
- ✓ A chave primária de R é normalmente a combinação das chaves estrangeiras



DER para Relacional - Resumo

ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

SQL – DDL

SQL -> Structured Query Language (SQL)

- ✓ Linguagem para definir, manipular e questionar uma base de dados relacional. É uma linguagem orientada ao processamento de conjuntos.
- ✓ 2 sub-linguagens:
  - DDL – Data Definition Language
  - DML – Data Manipulation Language
- ✓ 1 sub-linguagem de controlo BD
  - DCL – Data Control Language
- ✓ Hierarquia de objetos:
  - Catalog -> Schema -> Table -> Column
  - Há mais elementos (triggers, vistas, índices, stored procedures, funções, etc)

Notas Introdutórias - SQL:

- ✓ SQL utiliza tabela, linha e coluna para designar relação, tuplo e atributo do modelo relacional.
- ✓ Cada instrução termina com ponto e vírgula, “;”.
- ✓ Comentários linha a linha: “--”
- ✓ Comentários em bloco: “/* … */”

SQL – DDL – Data Definition Language

Utilizada para especificar a informação acerca de cada relação: esquema de cada relação, o domínio de valores associados com cada atributo, restrições de integridade, o conjunto de índices a manter para cada relação.

Há comandos não disponíveis em alguns SGBD – consultar os manuais para uma sintaxe mais completa.

Linguagem:

- ✓ Base de dados:
  - Criar uma base de dados:

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

CREATE DATABASE COMPANY;
```

- Eliminar uma base de dados:

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

DROP DATABASE COMPANY;
```

✓ Schema: é um "namespace" que agrupa tabelas e outros elementos pertencentes à mesma aplicação.

- Criar um Schema

```
CREATE SCHEMA schemaname [AUTHORIZATION username];

CREATE SCHEMA COMPANY AUTHORIZATION 'CCosta';
```

- Eliminar em Schema

```
DROP SCHEMA schemaname;

DROP SCHEMA COMPANY;
```

✓ Tipos de dados:

- Numbers;
- Characters, strings;
- Date e Time;
- Binary objects

✓ Tipos de dados podem variar de acordo com o SGDB

✓ Utilizar, na medida do possível, tipos de dados compatíveis com o standard -> aumenta a portabilidade da solução.

- Numeric
  - NUMERIC(p,s) e.g. 300.00
  - DECIMAL(p,s)
  - INTEGER (alias: INT) e.g. 32767
  - SMALLINT small integers
  - FLOAT(p) e.g. -1E+03
  - REAL (for short floats) DOUBLE (for long floats)

- String
  - CHARACTER(n) (fixed length)
  - CHARACTER (variable lenght)
  - CHARACTER VARYING(n) (alias: VARCHAR(n))
  - CLOB (Character Large Object, e.g., for large text)

- Date
  - DATE e.g. '1993-01-02'
  - TIME e.g. '13:14:15'
  - TIMESTAMP e.g. '1993-01-02 13:14:15.000001'

- Binary
  - BIT[(n)] e.g. B'01000100'
  - BLOB[(n)] e.g. X'49FE' (Binary Large Objects, e.g., for multimedia)

- Boolean
  - Boolean

Listagem não exaustiva...

✓ Tipos de dados mais utilizados:

Numeric Data Types

Data Type	Description	Length
`int`	Stores integer values ranging from -2,147,483,648 to 2,147,483,647	4 bytes
`tinyint`	Stores integer values ranging from 0 to 255	1 byte
`smallint`	Stores integer values ranging from -32,768 to 32,767	2 bytes
`bigint`	Stores integer values ranging from -2^{53} to $2^{53}-1$	8 bytes
`money`	Stores monetary values ranging from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
`smallmoney`	Stores monetary values ranging from -214,748.3648 to 214,748.3647	4 bytes
`decimal(p,s)`	Stores decimal values of precision *p* and scale *s*. The maximum precision is 38 digits	5–17 bytes
`numeric(p,s)`	Functionally equivalent to decimal	5–17 bytes
`float(n)`	Stores floating point values with precision of 7 digits (when *n*=24) or 15 digits (when *n*=53)	4 bytes (when *n*=24) or 8 bytes (when *n*=53)
`real`	Functionally equivalent to float(24)	4 bytes

Character String Data Types

Data Type	Description	Length
`char(n)`	Stores *n* characters	*n* bytes (where *n* is in the range of 1–8,000)
`nchar(n)`	Stores *n* Unicode characters	2*n* bytes (where *n* is in the range of 1–4,000)
`varchar(n)`	Stores approximately *n* characters	Actual string length +2 bytes (where *n* is in the range of 1–8,000)
`varchar(max)`	Stores up to $2^{31}-1$ characters	Actual string length +2 bytes
`nvarchar(n)`	Stores approximately *n* characters	2*n*(actual string length) +2 bytes (where *n* is in the range of 1–4,000)
`nvarchar(max)`	Stores up to $((2^{31}-1)/2)-2$ characters	2*n*(actual string length) +2 bytes

Binary Data Types

Data Type	Description	Length
`bit`	Stores a single bit of data	1 byte per 8 bit columns in a table
`binary(n)`	Stores *n* bytes of binary data	*n* bytes (where *n* is in the range of 1–8,000)
`varbinary(n)`	Stores approximately *n* bytes of binary data	Actual length +2 bytes (where *n* is in the range of 1–8,000)
`varbinary(max)`	Stores up to $2^{31}-1$ bytes of binary data	Actual length +2 bytes

Date and Time Data Types

Data Type	Description	Length	Example
`date`	Stores dates between January 1, 0001, and December 31, 9999	3 bytes	2008-01-15
`datetime`	Stores dates and times between January 1, 1753, and December 31, 9999, with an accuracy of 3.33 milliseconds	8 bytes	2008-01-15 09:42:16.142
`datetime2`	Stores date and times between January 1, 0001, and December 31, 9999, with an accuracy of 100 nanoseconds	6–8 bytes	2008-01-15 09:42:16.1420221
`datetimeoffset`	Stores date and times with the same precision as `datetime2` and also includes an offset from Universal Time Coordinated (UTC) (also known as Greenwich Mean Time)	8-10 bytes	2008-01-15 09:42:16.1420221 +05:00
`smalldatetime`	Stores dates and times between January 1, 1900, and June 6, 2079, with an accuracy of 1 minute (the seconds are always listed as ":00")	4 bytes	2008-01-15 09:42:00
`time`	Stores times with an accuracy of 100 nanoseconds	3–5 bytes	09:42:16.1420221

Definição de domínio:

- ✓ Create domain: permite definir novos tipos de dados
  - Domain -> pode conter um valor de defeito (default) e restrições do tipo not null e check

CREATE DOMAIN domainname

```
Criação...
CREATE DOMAIN compsalary INTEGER
       NOT NULL CHECK (compsalary > 475);

Utilização...
CREATE TABLE EMPLOYEE (
   ...
   Salary              compsalary,
   ...);
```

- ✓ Alternativa ao domain -> criar um novo tipo (alias) com o comando Create type (mas 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,
   ...);
```

- ✓ Atributos -> podem ser definidos valores por omissão para cada coluna pelo tema default
- ✓ Restrições de Integridade:
  - Check (P) -> impor uma regra a um atributo
  -