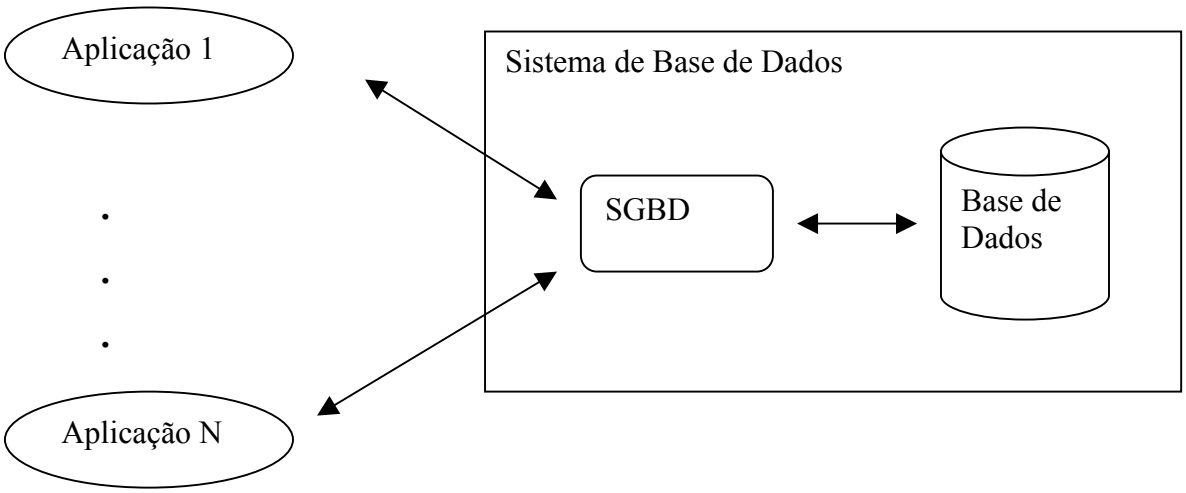
1.1.2. Sistemas de Bases de Dados

Um sistema de base de dados tenta baixar os custos de manutenção através da separação entre a forma como os dados são percebidos pelo programador e a forma como esses dados são armazenados fisicamente.

Se um programador altera uma estrutura de dados, essa nova estrutura é criada a partir da Base de Dados através do software de gestão da Base de Dados e não tem que reflectir-se nos outros programas.

=> Existem Registos Lógicos
Cada programa refere-se a registos lógicos de dados e não a registos físicos.

Os dados passam a estar integrados num único conjunto, sendo este administrado por uma aplicação específica: o Sistema de Gestão de Bases de Dados - SGBD.



SGBD – conjunto de programas que permite desempenhar as tarefas de armazenamento e manipulação de dados, fornecendo aos programadores e utilizadores finais os dados tal como eles são pedidos.

O acesso aos dados implica obrigatoriamente a comunicação com uma entidade (SGBD) que reserva para si os privilégios de acesso físico à base de dados e aos ficheiros que a compõem.

- Independência dos dados

Independência entre as aplicações e o formato em que é registada a informação. Uma vez que cada aplicação apenas tem que comunicar com o SGBD no processo de consulta e alteração de dados, pode abstrair-se da forma como estes são internamente mantidos.

Ao contrário dos sistemas de ficheiros, é possível que:

"Uma aplicação possa ser modificada, alterando a forma de utilização ou acesso à informação, sem que isso implique alterações nos restantes programas que partilham a utilização da mesma informação"

⇨ Cada aplicação tem a sua estrutura lógica de dados

- Níveis de abstracção

Consideram-se três níveis de abstracção:

- Nível Físico (ou nível interno)

Descrição do armazenamento físico da informação numa base de dados.

Definição das estruturas físicas que permitam obter um nível de desempenho, segurança e consistência satisfatório.

Definição das políticas de armazenamento de informação, de acordo com o número, exigência e necessidades de cada cliente específico.

⇨ Colecção de ficheiros, índices e outras estruturas de armazenamento

- Nível conceptual

Abstracção do mundo real, no que respeita aos utilizadores da base de dados.

O SGBD tem uma linguagem de definição de dados que permite ao utilizador descrever a implementação do esquema conceptual (esquemas de estrutura) pelo esquema físico.

⇨ A base de dados conceptual é tida como incluindo todos os dados da organização.

- Nível de visualização ("views" ou nível externo)

Uma "view" (subesquema) é uma porção da base de dados conceptual ou uma abstracção de parte da base de dados conceptual.

Pode ser apenas uma pequena base de dados ao mesmo nível de abstracção que a base de dados conceptual.

Pode estar a um nível de abstracção mais elevado i. é, os dados de uma "view" podem ser construídos a partir da base de dados conceptual mas não estarem presentes na base de dados *(exemplo: - idade)*.

⇨ **Como resultado destes três níveis de abstracção vamos ter dois níveis de independência de dados**

1. Independência física de dados

Devido a questões de optimização do desempenho ou de segurança, é possível alterar aspectos relativos à implementação física da base de dados, sem que se altere o seu esquema conceptual, isto é, manter os dados e as associações entre eles inalterado.

Qualquer alteração no modelo físico não implicará alterações ou ajustamentos no modelo conceptual.

Exemplos:

- Alteração das estruturas de armazenamento de informação (ficheiros)
- Criação de índices para optimizar o acesso à informação

2. Independência lógica de dados

Durante o período de vida da base de dados pode ser necessário alterar o modelo conceptual, por exemplo, adicionando informação a entidades já existentes ou criando novas entidades

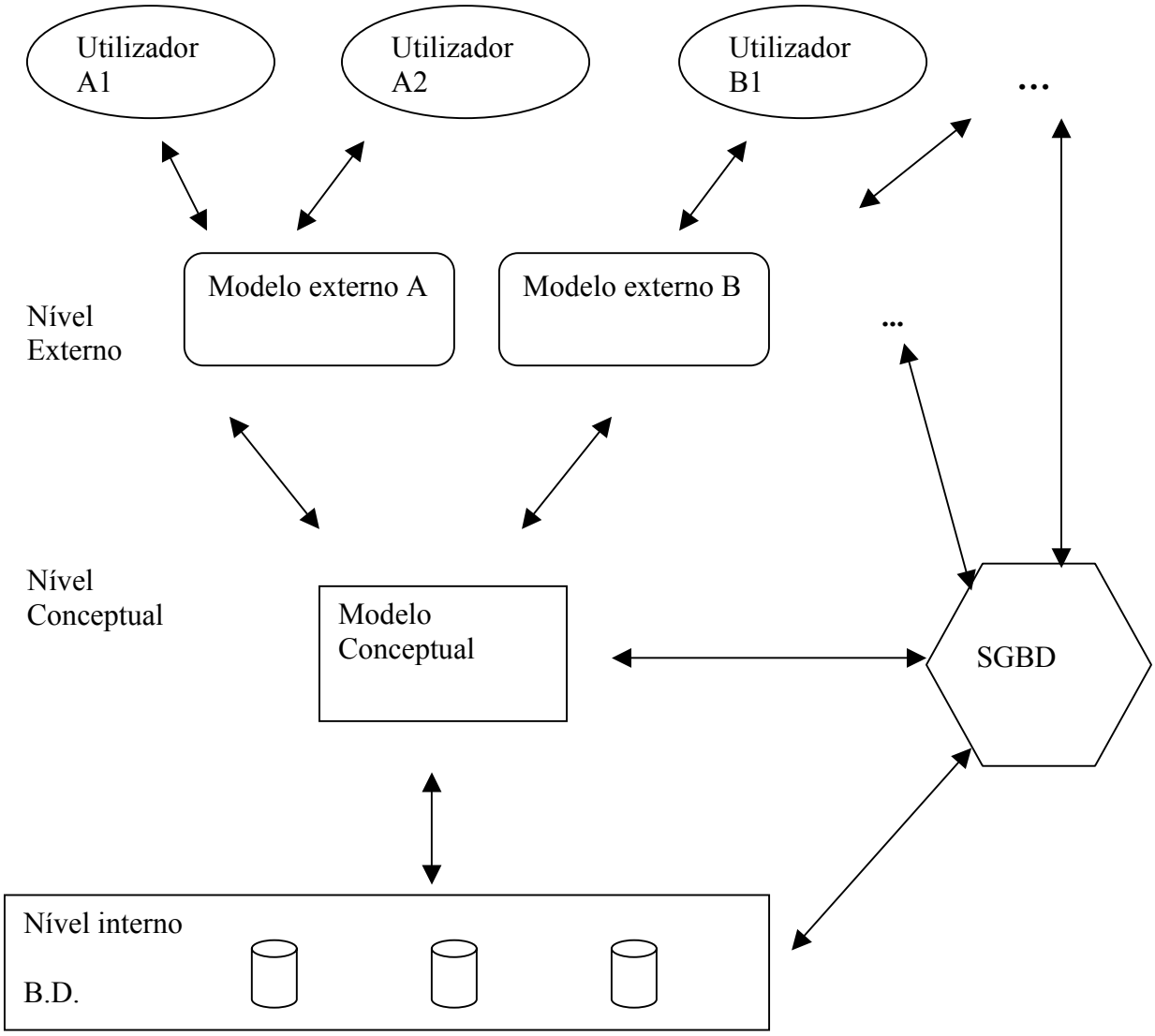
Muitas alterações podem ser feitas sem afectar vistas ("views") já existentes.

Exemplo:
Necessidade de registo de informação acerca de um novo atributo de uma qualquer entidade. (Registo do "BI" de todas as "Pessoas")

Eliminação de informação?

Arquitectura ANSI / SPARC para um sistema de base de dados

(Proposta em 1975)



ANSI - American National Standards Institute
SPARC - Standards Planning and Requirements Committee

1.2. Objectivos e Capacidades de um Sistema de Gestão de Bases de Dados

Um SGBD é um sistema de gestão e armazenamento de dados, capaz de aceder eficientemente a grandes quantidades de dados.

Serve de intermediário entre o nível aplicacional e a base de dados, evitando a manipulação directa por parte de cada aplicação cliente.

É a única entidade responsável pela segurança, integridade e validade dos dados armazenados.

A maioria dos SGBD's possui:

- Suporte para pelo menos um modelo de dados através do qual o utilizador possa visualizar os dados.

- Suporte para linguagens de alto nível que permitam ao utilizador definir a estrutura de dados e manipular os dados.

- Gestão de transacções - Possibilitar o acesso à base de dados de vários utilizadores em simultâneo (acesso concorrente)

- Controlo de acesso – Capacidade para impedir o acesso aos dados a utilizadores não autorizados e verificar a validade dos dados

- Capacidade de recuperação de falhas sem perda de dados.

Linguagens da base de dados

Nas linguagens de programação mais comuns, as instruções de declaração e execução fazem parte de um só conjunto, isto é, estão englobadas numa mesma linguagem.

Em sistemas de bases de dados as duas funções estão separadas em duas linguagens específicas:

- Uma linguagem para definir a base de dados

- *Data Definition Language (DDL)*

Utilizada para definir a estrutura da base de dados e da informação que deve armazenar.

Constitui uma notação para descrever a estrutura da informação.

Não possuindo instruções de execução, não pode ser utilizada para obter ou alterar os dados em si, para isso existe:

- Uma linguagem de interrogação da Base de Dados
("query language")

- *Data Manipulation Language (DML)*

É a linguagem disponibilizada para obter, armazenar, alterar ou eliminar informação da base de dados.

As instruções pertencentes a esta linguagem podem ser:

. Executadas de forma autónoma permitindo aos utilizadores finais usar directamente a Base de Dados sem que programas de aplicação tenham que ser escritos

ou

. Embutidas em linguagens hospedeiras (C, Pascal, Visual Basic, Fortran, Java, ...).

Neste caso, um pré-compilador traduz as instruções DML para chamadas de subrotinas com os parâmetros correspondentes.

Transacções

Numa transacção todas as instruções acabam ou nenhuma. Não há operações que sejam parcialmente completadas.

Exemplo típico: transferência de dinheiro entre duas entidades "A" e "B".

Suponha-se que o cliente "A" efectuou uma compra de 1000 euros à empresa "B".

É necessário debitar este valor na conta de "A" e creditar o mesmo valor na conta de "B".
Em primeiro lugar, é debitado o valor na conta de "A", passando esta a apresentar um saldo de X-1000.
Se uma situação excepcional interrompe a aplicação e a quantia não é creditada na conta de "B" a base de dados ficará corrompida.

Uma transacção é um conjunto de operações perfeitamente delimitado em que garantidamente são executadas todas as instruções ou então nenhuma o será.

```
Begin Transaction
        Operação 1
        Operação 2
        ...
        Operação n
End Transaction
```

Voltando ao nosso exemplo, as duas actualizações, deveriam ser agrupadas numa única transacção.

Por definição, uma transacção deve exibir quatro características fundamentais:

(Propriedades ACID – Atomicity, Consistency, Isolation, Durability)

Atomicidade *(Atomicity)*

O conjunto de instruções que compõem a transacção é indivisível, no sentido em que todas elas são executadas, ou então nenhuma o será.

Sempre que todas sejam executadas sem nenhuma situação excepcional diz-se que foi executado o COMMIT da transacção.

Na ocorrência de alguma situação excepcional que impossibilite a sua completa execução, deve ser anulado o efeito de todas as instruções que compõem a transacção e que ainda foram executadas (ROLLBACK).

Consistência (Consistency)

Uma transacção deve, após a sua completa execução, deixar a base de dados num estado consistente.

Pode acontecer que, entre a execução de alguma das operações que a compõem, a consistência não se verifique, no entanto após a execução de todas as operações, ela tem que ser verificada.

Isolamento (Isolation)

Apesar de ser possível a execução paralela e simultânea de diferentes transacções, o sistema deve dar a ilusão de que cada uma delas é a única a executar, estando por isso aparentemente *isolada*.

Sempre que existam várias transacções a aceder aos mesmos dados, o sistema deve evitar que existam interferências mútuas, e garantir que o estado final da base de dados seria o mesmo após uma eventual execução em série.

Persistência (Durability)

Deve ser assegurado que após a execução bem sucedida de uma transacção (COMMIT), os efeitos dela resultantes se tornam persistentes (não voláteis) na base de dados.

Qualquer transacção futura deve operar sobre o novo conjunto de dados, bem como qualquer eventual falha não deve anular as alterações entretanto produzidas.

Os efeitos de cada transacção podem apenas ser desfeitos ou alterados por outras transacções.

Controlo de acesso

Segurança

É um dos requisitos básicos exigidos a um SGBD. Consiste em proteger os dados armazenados dos acessos não autorizados e garantir que todas as operações executadas sobre a base de dados o são por utilizadores (aplicações) devidamente credenciados.

Integridade

Por definição, uma base de dados está num estado de integridade se todos os dados que contém são válidos, isto é, não contradizem a realidade que estão a representar nem se contradizem entre si.

O SGBD permite definir regras que garantem a integridade após cada processo de alteração de informação.

Deve ser possível definir restrições de integridade do tipo:

- um item de dados pertencer a um conjunto de valores
- formato (natureza, comprimento)
- coerência com outras informações
- ...

Tolerância a Falhas

Devido à potencial importância dos dados armazenados numa base de dados, é essencial a implementação de mecanismos de tolerância a falhas (*hardware / software*), que garantam a reposição da informação para um estado anterior válido.

Para tal são utilizados basicamente dois mecanismos:

• Implementação de cópias de segurança com estados válidos da base de dados (*Backups)*

• Registos de actividade (*Logging*). Registo de todas as operações efectuadas sobre a base de dados a partir do último ponto de cópia

Se ocorre uma anomalia, o procedimento de recuperação permite a partir da última cópia e do registo de actividades, restaurar a base de dados para um estado consistente e detectar a origem da anomalia.

Arquitectura cliente-servidor

Tipicamente as aplicações de bases de dado seguem um modelo cliente-servidor:

- Um servidor contém os dados e executa o SGBD
- Os Clientes ligados por uma rede de comunicações executam os programas de aplicação, que fazem a interface com o utilizador e o acesso remoto à base de dados.

Numa base de dados distribuída, esta surge ao utilizador como uma única base de dados, sendo na realidade constituída por diversas bases de dados (i. é, diversos SGBD's) distribuídos por diferentes computadores.

Modelos de Dados

Um modelo de dados é a colecção de, pelo menos, 3 componentes:

1) Um conjunto de tipos de estruturas de dados

Define o tipo de dados e como se interrelacionam

2) Um conjunto de operadores

Operações que permitem manipular as estruturas de dados definidas.

3) Um conjunto de regras de integridade

Regras que definem que dados são válidos

1ª Geração de SGBD's (*meio dos anos 60*):

Modelo Hierárquico

Modelo em Rede ou Codasyl

2ª Geração

Modelo Relacional *(proposto em 1970 por E. F. Codd)*

3º Geração

Modelo Object - Oriented

...

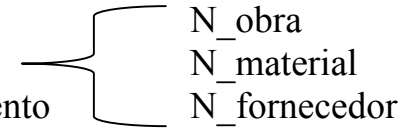
Suponhamos a Base de Dados exemplo:

Obra
N_obra
Nome_obra
Data_de_inicio
Data_de_fim
Orçamento

Fornecedor
N_fornecedor
Nome_fornecedor
Morada

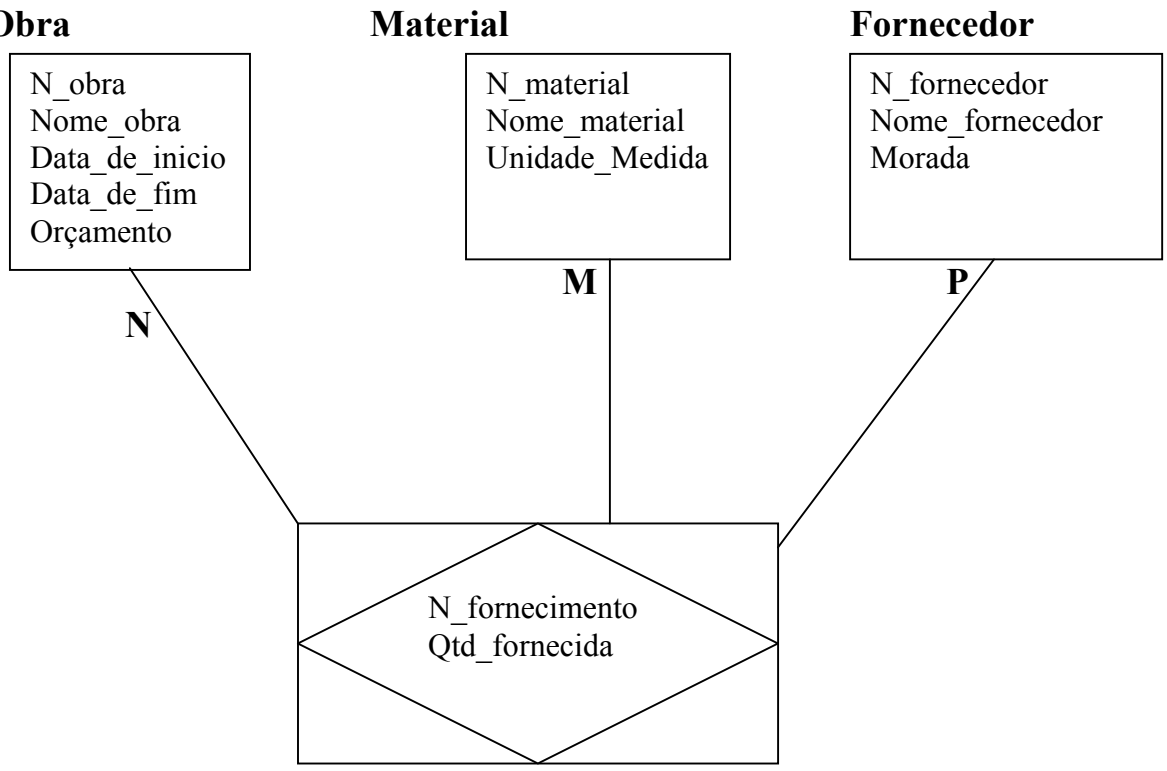
Material
N_material
Nome_material
Unidade_medida

Fornecimento



N_fornecimento
Qtd_fornecida

Modelo Conceptual



. Cada fornecedor pode fornecer vários materiais (M) para várias obras (N).

. Cada material pode ser fornecido por vários fornecedores (P) para várias obras (N).

. Cada obra pode receber vários materiais (M) de vários fornecedores (P).

Como realizar as operações que se seguem nos modelos Hierárquico, Rede e Relacional ➔

. **Interrogações**

I1: Quem forneceu o material M1 para a obra O1?

I2: Que materiais forneceu o fornecedor F2 e para que obras?

. Actualizações:

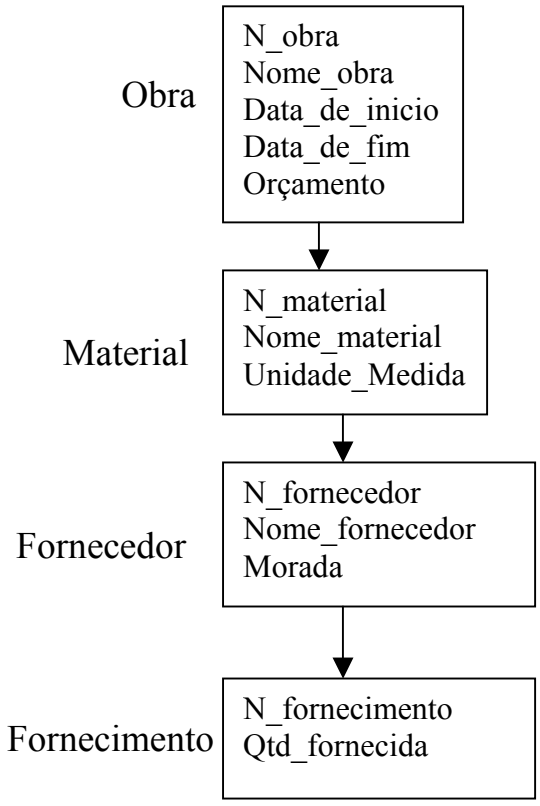
A1: Apagar todos os fornecimentos do fornecedor F1 para a obra O1.

A2: Juntar à base de dados um novo fornecedor F5.

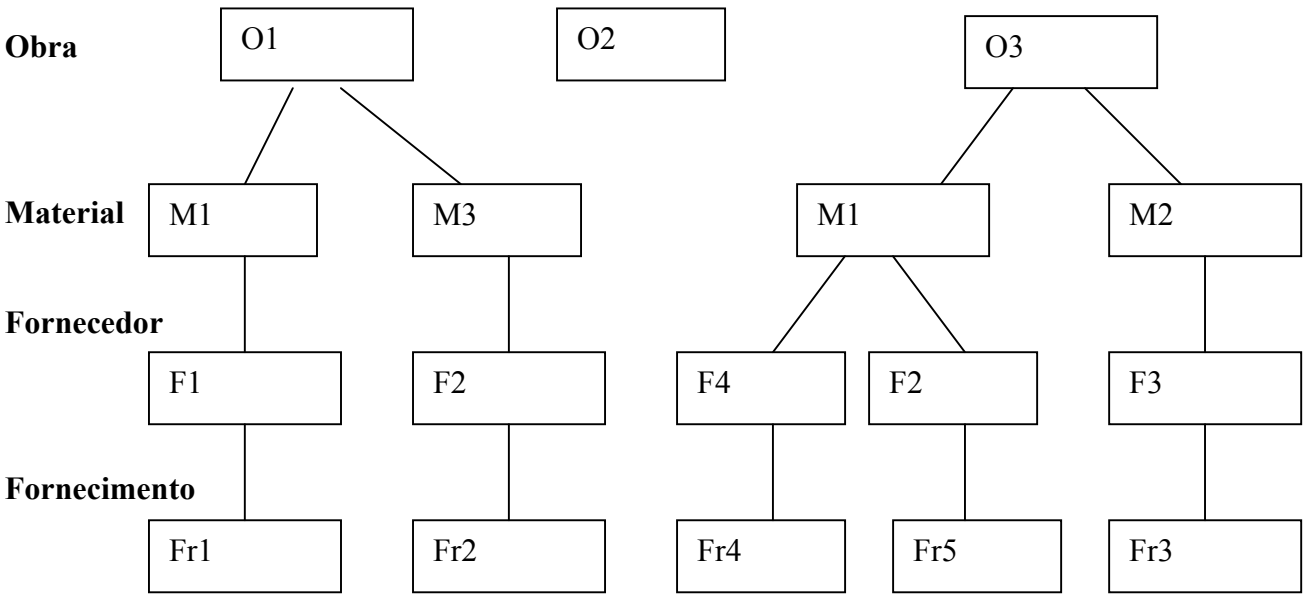
A3: Modificar a morada do fornecedor F3 para "Covilhã"

Modelo Hierárquico

Um esquema possível:



Conteúdo da Base de Dados num determinado instante:



Operação elementar de interrogação do modelo hierárquico:

```
Get [ Next | Superior ] <nome do registo 1>
[ For This <nome do registo 2> AND
     This <nome do registo 3> ...(*) ]
[ Where <condição lógica>  ]
```

(*) até ao nível anterior ao registo 1

Entre [ ] significa opcional.

I1: Quem forneceu o material M1 para a obra O1?

```
Get Obra Where N_obra = O1

Get Material For This Obra
     Where N_material = M1

Get Next Fornecedor
     For This Obra AND This Material

Do While not end-of-fornecedor
     Print N_fornecedor, Nome_fornecedor, Morada

     Get Next Fornecedor
          For This Obra AND This Material

End Do
```

I2: Que materiais forneceu o fornecedor F2 e para que obras?

Get Next **Obra**

```
Do While not end-of-obra

    Get Next Material
        For This Obra

    Do While not end-of-material
        Get Fornecedor
            For This Obra AND This Material
            Where N_fornecedor = F1
        If not end-of-fornecedor Then
            Print N_obra, Nome_obra, N_material, Nome_material
        End if
        Get Next Material
            For This Obra
    End Do
    Get Next Obra
End Do
```

Nota: o esquema é o pior possível para esta interrogação

Operações elementares de actualização do modelo hierárquico:

. Insert Into <nome do registo>
<lista de valores>
[**Linking** < chave (!) do 1º nível = valor1 ,
chave do 2º nível = valor2, *]

Nas operações seguintes é necessário em primeiro lugar encontrar o registo (com Get) e só depois apagá-lo ou modificá-lo.

. Delete <nome do registo>

. **Update** <nome do registo>
Setting <lista de modificações>

(*) até ao nível anterior ao do registo

A1: Apagar todos os fornecimentos do fornecedor F1 para a obra O1.

A1: Apagar todos os fornecimentos do fornecedor F1 para a obra O1.

```
Get Obra Where N_obra = O1
Get Next Material For This Obra
Do While not end-of-material
     Get Fornecedor
          For This Obra AND This Material
          Where N_fornecedor = F1
     If not end-of-fornecedor Then
          Get Next Fornecimento
               For This Obra AND This Material
                AND This Fornecedor
          Do While not end-of-fornecimento
               Delete Fornecimento
               Get Next Fornecimento
                    For This Obra AND This Material
                      AND This Fornecedor
          End Do
     End If
     Get Next Material For This Obra
End Do
```

A2: Juntar à base de dados um novo fornecedor F5.

A2: Juntar à base de dados um novo fornecedor F5.

Impossível.
Não podemos juntar à base de dados um novo fornecedor enquanto ele não efectuar algum fornecimento.

Falta a definição do contexto: Qual a obra e qual o material a que seria ligado?

Uma solução para este problema consiste em criar uma obra fictícia (por ex. N_obra = 0) e um material fictício para ligar os fornecedores sem fornecimentos.

Seria então:

```
Insert Into Fornecedor
     ( F5, Empresa X, Lisboa)
     Linking N_obra=0, N_material = 0
```

A3: Modificar a morada do fornecedor F3 para "Covilhã"

➔

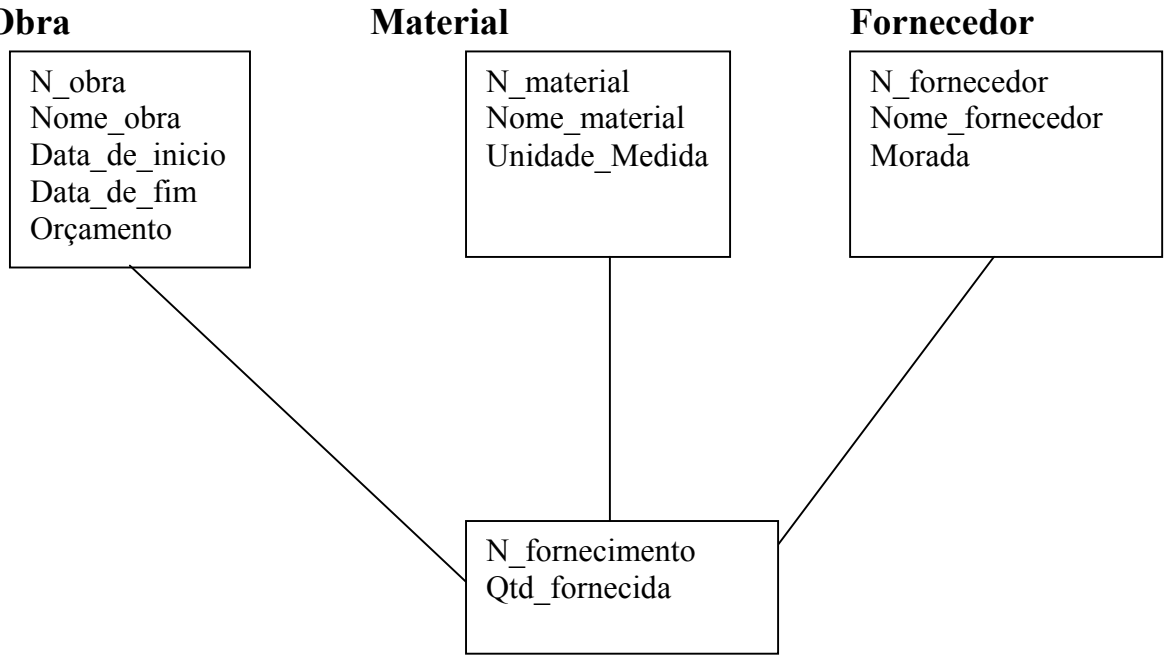
A3: Modificar a morada do fornecedor F3 para "Covilhã"

```
Get Next Obra
Do While not end-of-obra

    Get Next Material
        For This Obra

    Do While not end-of-material
        Get Fornecedor
            For This Obra AND This Material
            Where N_fornecedor = F3
        If not end-of-fornecedor Then
            Update Fornecedor
                Setting  Morada = "Covilhã"
        End If
        Get Next Material
        For This Obra
    End Do
    Get Next Obra
End Do
```

Modelo em Rede

*Definido pelo DBTG (Data Base Task Group) da CODASYL (**CO**nference on **Da**ta **Sy**stems **L**anguages)*

Esquema da base de dados:

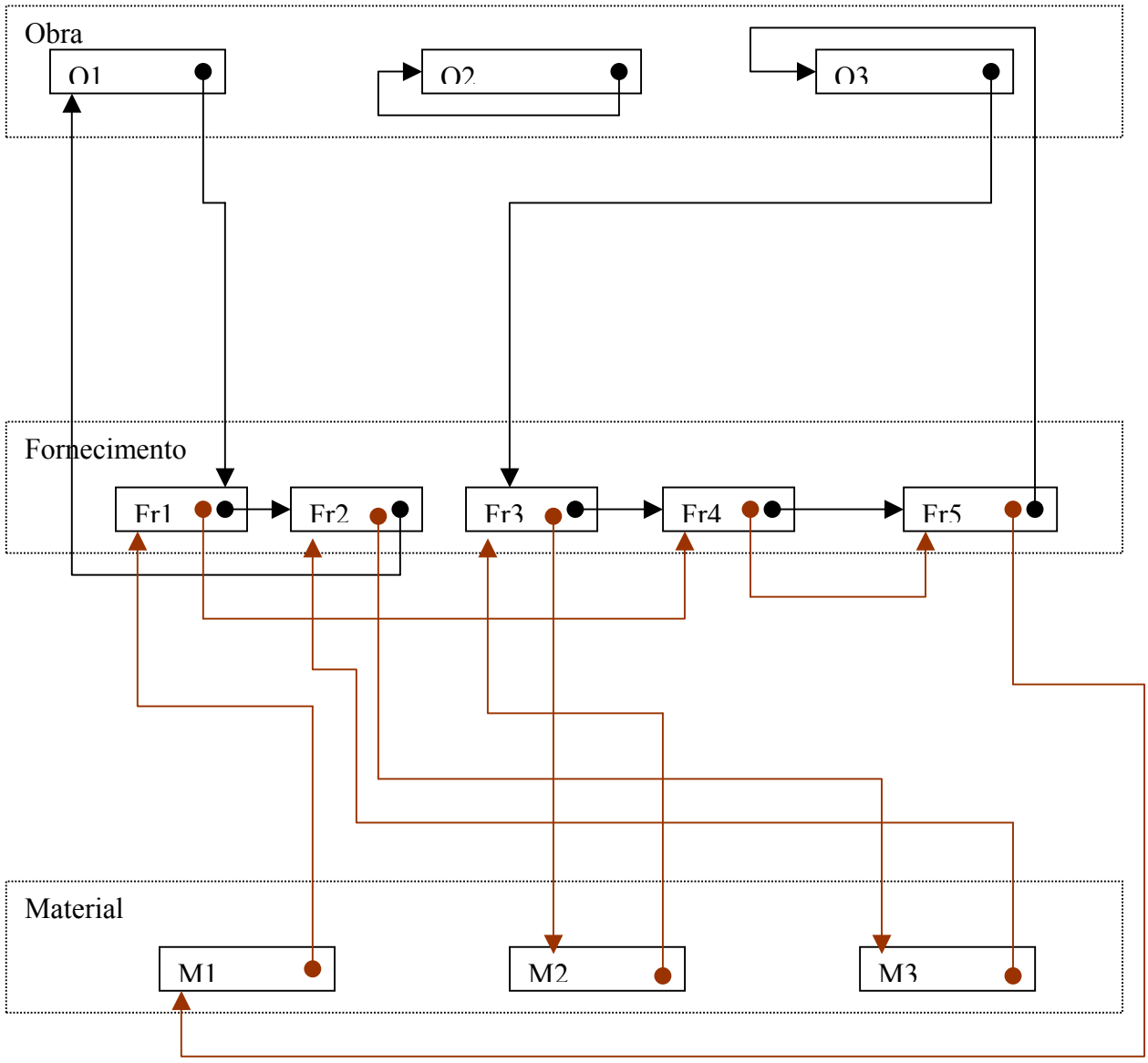


. Uma obra tem vários fornecimentos.

. Um material tem vários fornecimentos

. Um fornecedor faz vários fornecimentos.

Conteúdo da base de dados:



Completar para Fornecedor / Fornecimento ...