**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**3D (Definição Dicionário de Dados)**

**Entidades**

*Objectivo: Criar uma estrutura que permite especificar entidades a utilizar na solução e que, para cada uma delas sejam especificados os seus campos, relações e herança.*

**.: Entidade :.**

*Descrição: Conjunto de elementos representativos das entidades que intervirão na solução.*

**Estratégia**

As entidades são um dos pontos principais na definição do dicionário de dados. Com elas surge a problemática das relações entre entidades, herança, campos que as constituem e respectivos atributos.

Inicialmente pensou-se em criar um elemento que permitisse definir a estrutura de uma entidade e, posteriormente, podiam-se caracterizar relações entre entidades num elemento adicional (e.g. *relations*). Apesar de se estar a esboçar uma solução prática, do ponto de vista do dicionário de dados, o mesmo não se verificava do ponto de vista do gerador, uma vez que iria ser necessário processamento adicional para agrupar toda a informação de uma entidade nela própria. Assim, optou-se por agregar em elementos *entity* toda a informação referente a uma entidade, nomeadamente:

- Identificação da Entidade

- Herança

- Campos

- Relações entre entidades

O elemento *entity* irá conter atributos disponibilizando informação relativa à identificação da entidade e dados relativos a herança.

Neste elemento irão existir dois tipos de elementos: *fields* e *relations*. O primeiro será o responsável por agrupar todos os campos que constituem a entidade e que não referenciem outras entidades (chaves estrangeiras), o segundo será o responsável por especificar as chaves estrangeiras.

**Identificação da Entidade**

Uma entidade deverá ser identificada inequívocamente na solução pelo seu nome. Este será o atributo utilizado para nomear a classe, produto da geração de código, bem como, a correspondente tabela, caso se opte por persistência.

O nome da entidade passa a ter um papel importante, do ponto de vista do *schema* do dicionário, uma vez que permitirá impor a unicidade de nomes, bem como, integridade referencial.

**Herança**

Devido à necessidade de especificar relações de herança no dicionário de dados, incluiu-se um atributo que indicasse qual o papel da entidade na hierarquia, podendo o mesmo tomar os valores *abstract*, *base* ou *dependent*.

As relações de herança em *object model* são representadas através de uma hierarquia de classes. Em modelo relacional permitem associações do tipo *IS-A*.

**Base**

As entidade que sejam deste tipo serão consideradas, em *object model*, classes base, podendo servir de ponto de extensibilidade para outras classes. Em modelo relacional constituirão a entidade base de uma associação *IS-A*.

**Abstract**

**QUAL O SIGNIFICADO DE ABSTRACT EM MODELO RELACIONAL?**

Entidades deste tipo serão representadas como classes abstractas, impedindo assim a sua instanciação.

**Dependent**

Estas serão as entidades que representam classes derivadas em *object model* e tabelas do modelo relacional que representam o mesmo papel na hierarquia.

**Campos**

Conforme mencionado, cada entidade será responsável por enumerar os seus campos que, deverão estar contidos no elemento *fields*. O elemento *field* é assim o descritor de um campo que não referencia outra entidade e o seu *schema* é o seguinte:

<xs:complexType name="t\_field">

<xs:attribute name="name" type="xs:string" use="required"/>

<xs:attribute name="type" type="xs:string" use="required"/>

<xs:attribute name="unique" type="xs:boolean" use="optional" default="false"/>

<xs:attribute name="visible" type="xs:boolean" use="optional" default="false"/>

<xs:attribute name="nillable" type="xs:boolean" use="optional" default="false"/>

<xs:attribute name="sequence" type="xs:boolean" use="optional" default="false"/>

<xs:attribute name="searchable" type="xs:boolean" use="optional" default="false"/>

</xs:complexType>

O nome do campo é o atributo que identifica inequívocamente o campo de uma entidade e, como tal, é garantida a unicidade do mesmo no âmbito de cada entidade. Detalham-se de seguida os restantes atributos que compõem um campo.

**Type**

Este atributo irá indicar qual o tipo de campo, sendo que terá que ser referenciado o atributo nome de um elemento filho de *userTypes*.

Assim, o tipo de um campo só pode ser o nome de um tipo definido no âmbito da solução.

**Unique**

Este atributo indicará quais os campos que pertencem à chave primária da entidade. Assim, a sua principal utilidade será ao nível do modelo relacional.

**Visible**

Indicador de visibilidade do campo, a ser utilizado ao nível do *object model*.

**Nillable**

Indicador da possibilidade de serem usados valores *null* para afectar o campo. Esta característica terá utilização no modelo relacional.

**Sequence**

**O QUE É O SEQUENCE?**

**Searchable**

Este atributo irá permitir a definição de índices no modelo relacional.

**Relações Entre Entidades**

As relações possíveis entre entidades são *um para um*, *um para n* ou *n para n* e, a representação destas relações é diferente em *object model* e modelo relacional.

Em *object model* uma relação é representada por uma referência para a entidade alvo, sendo que, e.g. numa relação *um para um* essa referência será uma instância da entidade alvo e, no modelo relacional, a relação será representada pela agregação da chave primária da entidade alvo na tabela da entidade de origem.

Assim, o atributo *Unique* associado aos campos de uma entidade assume um papel importante, na medida em que permite determinar qual o conjunto de atributos que define a sua chave primária e, consequentemente, que deverá ser incluido na tabela de origem.

As relações entre entidades deverão ser especificadas nas entidades de origem dentro do elemento *relations*. No sentido de permitir uma maior versatilidade na representação de relações, decidiu-se pela criação de elementos distintos para representar cada tipo de relação.

Para cada um dos tipos de relação é obrigatório indicar o nome do campo que a representará e o nome da entidade referenciada, bem como, a cardinalidade dessa relação. É neste último que surgem as diferenças entre os tipos de relações.

A fim de melhor ilustrar a questão da cardinalidade sugere-se o seguimento de um exemplo.

DESCREVER EXEMPLO QUE SERVIRÁ DE BASE À EXPLICAÇÃO DO SEGUINTES ELEMENTOS

***Squema* de Relações 1-1**

<xs:complexType name="t\_oneToOne">

<xs:attribute name="name" type="xs:string" use="required"/>

<xs:attribute name="entity" type="xs:string" use="required"/>

<xs:attribute name="minOccurs" type="t\_one" use="required"/>

<xs:attribute name="maxOccurs" type="t\_zeroOrOne" use="required"/>

</xs:complexType>

Os valores possíveis para o número máximo e mínimo de ocorrências são 0 ou 1. Assim, esta entidade pode ter 0 ou 1 referências para a entidade indicada por *entity*.

***Squema* de Relações 1-n**

<xs:complexType name="t\_oneToMany">

<xs:attribute name="name" type="xs:string" use="required"/>

<xs:attribute name="entity" type="xs:string" use="required"/>

<xs:attribute name="minOccurs" type="t\_maxOccurs" use="required"/>

<xs:attribute name="maxOccurs" type="t\_maxOccurs" use="required"/>

</xs:complexType>

Os valores possíveis para o número mínimo de ocorrências são inteiros positivos e, para o número máximo são os inteiros positivos excluindo o 0 e incluindo o *infinito* (materilizado pela keyword *unbounded*).

VERIFICAR SE O MINIMO NÃO DEVERIA SER 0 OU 1

***Squema* de Relações n-n**

<xs:complexType name="t\_manyToMany">

<xs:attribute name="name" type="xs:string" use="required"/>

<xs:attribute name="entity" type="xs:string" use="required"/>

<xs:attribute name="minOccurs" type="t\_maxOccurs" use="required"/>

<xs:attribute name="maxOccurs" type="t\_maxOccurs" use="required"/>

</xs:complexType>

Os valores possíveis para o número mínimo de ocorrências são inteiros positivos e, para o número máximo são os inteiros positivos excluindo o 0 e incluindo o *infinito*.

VERIFICAR SE O MÍNIMO NÃO DEVERIA SER 0 A UNBOUNDED