# Modelagem Semântica e Gerenciadores de Banco de Dados

Regina Lúcia de Oliveira Moraes
Instituto de Computação – Universidade Estadual de Campinas UNICAMP
e-mail: regina@ceset.unicamp.br

**Resumo**

*Modelos Semânticos surgiram devido à necessidade de se captar, com um nível de detalhamento maior, as propriedades de uma aplicação. Apesar de não terem atingido o nível de aceitação esperado por parte da indústria, modelos semânticos são muito utilizados como ferramentas complementares no desenvolvimento de sistemas. Esse trabalho apresenta alguns dos modelos semânticos existentes na literatura, ressaltando suas particularidades e a sua tradução para os esquemas de gerenciadores de base de dados.*

**Abstract**

*Semantic Models were come about due to the necessity to capture, at a more detailed level, the properties of an application. Despite their inability to reach the acceptance levels expected by the industry, semantic models are very used as complementary tools in systems' development. This work presents some of the semantic models documented by the literature, highlighting their idiosyncrasies and translations for database management systems design.*

## 1. INTRODUÇÃO

A motivação que impulsionou o desenvolvimento de sistemas de banco de dados foi a busca da flexibilidade e da eficiência no compartilhamento, gerenciamento, recuperação e armazenamento de grandes massas de dados, permitindo o compartilhamento e utilização concorrente, a redução da redundância e inconsistência dos dados, provimento de segurança e confiabilidade, proteção dos dados contra acessos perniciosos, fornecimento de mecanismos de recuperação em caso de falhas, permissão para incorporar e manter algumas restrições pelo sistema de banco de dados.

Os primeiros modelos que surgiram guardavam uma relação bastante estreita com a estruturação física dos dados. Com o modelo relacional, surge uma linguagem de manipulação e o conceito de independência física dos dados. Embasado no conceito de relações que descrevem entidades e relacionamentos, utilizam os valores de atributos e não mais ponteiros para definir os relacionamentos existentes.

As propriedades de uma aplicação de banco de dados podem ser agrupadas em dois aspectos: estrutural e comportamental. O aspecto estrutural compreende os tipos de dados, relacionamentos e restrições que podem ser necessárias sobre os dados; o aspecto comportamental representa as operações e transações que interagem sobre as entidades e relacionamentos. Os aspectos comportamentais, na maioria dos modelos existentes, são relegados às aplicações. Esses modelos têm um conhecimento muito restrito a respeito do significado dos dados. Compreendem certos valores atômicos simples, certos relacionamentos, mas pouco mais além disso, seria interessante se os sistemas compreendessem um pouco mais, de maneira que pudessem responder, com inteligência, as interações dos usuários.

Embora tenha se firmado como um modelo de referência na área de banco de dados, não faltam críticas ao modelo relacional, no que se refere à representação adequada dos diversos

relacionamentos que existem numa aplicação. Numa tentativa de suprir essa deficiência surgem os modelos semânticos [1] [2], que propiciaram uma melhor captação das propriedades de uma aplicação. Infelizmente, esses modelos ainda não obtiveram o reconhecimento da indústria, que ainda hoje se apóia, em sua grande maioria, nos modelos relacionais.

Apesar disso, diversas pesquisas têm sido desenvolvidas nesse campo objetivando o desenvolvimento de conceitos que possam melhor representar as propriedades das aplicações e dessa forma, a modelagem semântica tem sido difundida.

O objetivo desse trabalho é apresentar alguns modelos semânticos que são encontrados na literatura e a maneira como é feito o mapeamento desses modelos para o sistema de banco de dados visando a persistência desses dados juntamente com seus aspectos comportamentais.

No capítulo 2 são comentados alguns trabalhos relacionados que embasaram esse trabalho. Alguns modelos semânticos foram destacados no capítulo 3, enquanto que no capítulo 4 foram expostos alguns aspectos da tradução dos modelos apresentados para gerenciadores de banco de dados. Uma breve conclusão é apresentada no capítulo 5.

## 2. TRABALHOS RELACIONADOS

Esse trabalho foi fortemente influenciado pela dissertação de mestrado apresentada no Instituto de Computação [10], que propôs uma interface que estende a capacidade de captação semântica dos gerenciadores de banco de dados relacionais. Alguns dos modelos apresentados na dissertação suscitou o interesse para a pesquisa desse trabalho. Outros modelos foram acrescentados, como o modelo MEASUR [17] que é utilizado por alunos e pesquisadores na área de inteligência artificial e que também foi pesquisado a partir de outra dissertação de mestrado do Instituto de Computação[16]. Além desses, foi acrescentado o modelo UML uma vez que esse modelo tem apresentado um crescimento acentuado para a modelagem de banco de dados[6] [7] [9] [13].

## 3. MODELOS SEMÂNTICOS

Algumas definições ressaltam a importância do significado da informação para os seres humanos que irão utilizá-las. Esses seres humanos interagem com os objetos existentes no mundo real, têm deles uma noção de contexto onde se inserem e que comportamento têm nesse contexto.

Uma aplicação de negócios representa aspectos infinitamente ricos e complexos, o que faz da extração de requisitos uma tarefa árdua. Modelos gráficos auxiliam o entendimento e diminui a ambigüidade existente na linguagem natural, sendo um forte aliado para a validação dos requisitos junto aos clientes. Os modelos relacionais, devido a sua estrutura baseada em registro, têm dificuldades de representar algumas características que normalmente estão presentes no comportamento dos objetos que representa. Dificuldades são apresentadas quando há variações nos atributos, restrição de domínio do conjunto de objetos e a dificuldade de representar os diversos aspectos semânticos que envolvem os objetos no mundo real. Devido a essas dificuldades, surgiram os modelos semânticos.

Modelos semânticos têm como objetivo a representação de determinada parte do mundo real, sendo assim, o que se busca é que o modelo produzido traduza de maneira mais próxima possível os objetos que ele representa. Segundo Date [3], a modelagem semântica é uma classificação apropriada para a atividade de representar o sentido e são caracterizadas por: (i) identificação de um conjunto de conceitos semânticos que parecem úteis ao se falar do mundo real; (ii) determinação de um conjunto de objetos simbólicos (formais) correspondentes para

representarem aqueles conceitos semânticos; (iii) determinação de um conjunto de regras de integridade ao lado dos objetos simbólicos; (iv) desenvolvimento de um conjunto de operadores para a manipulação daqueles objetos simbólicos.

Por ter como base um contexto e a semântica que cada objeto tem para um determinado grupo de usuário dentro desse contexto, um determinado objeto no mundo real pode muito bem ser considerado uma entidade por algumas pessoas e propriedades por outras e ainda associação por outras. Uma das metas da modelagem semântica é suportar tal flexibilidade de interpretação.

### *3.1. MODELO ENTIDADE-RELACIONAMENTO*

Uma das principais propostas da área de modelagem semântica e certamente uma das propostas de maior influência foi o modelo de entidade-relacionamento proposto por Chen[1]. Um dos modelos com maior capacidade semântica, o modelo tem por base a percepção de que o mundo real é formado por um conjunto de objetos chamado *entidades* e pelo conjunto de *relacionamentos* entre esses objetos. Um *conjunto de entidades* é um conjunto de entidades de um mesmo tipo que compartilham as mesmas propriedades, ou seja, têm os mesmos *atributos.*

Atributos são propriedades descritivas de cada entidade. Formalmente, um atributo de um conjunto de entidades é uma função que relaciona o conjunto de entidades a seu domínio [15]. Da maneira como é usado no modelo entidade-relacionamento um atributo pode ser caracterizado como *simples ou compostos* (podem ser divididos em outros atributos), *monovalorados ou multivalorados* (possuem um conjunto de valores para uma única entidade), *nulos* (ausência de valor) e *derivados* (valor é derivado de outros atributos ou entidades).

Uma superchave é um conjunto de um ou mais atributos que, tomados coletivamente, nos permitem identificar de maneira única uma entidade em um conjunto de entidades e são chamadas *chaves candidatas* quando nenhum subconjunto dela própria é uma superchave. A chave candidata que melhor caracteriza um conjunto de entidades é escolhida pelo projetista como a *chave primária*. Quaisquer duas entidades de um conjunto de entidades não podem ter, simultaneamente, os mesmos valores em sua chave primária. Quando não é possível identificar uma chave candidata para um dado conjunto de entidades, pode-se gerar um número para desempenhar essa função e nesse caso dizemos que foi criado um *surrogate*.

Um *conjunto relacionamento* é um conjunto de relacionamentos de mesmo tipo. De maneira formal, é a relação matemática com $n \geq 2$ conjuntos de entidades, podendo ser ou não distintos [15]. Chama-se *participação*, a associação entre os conjuntos de entidades. Uma *instância* de relacionamento em um esquema entidade-relacionamento representa a existência de uma associação entre essa entidade e o mundo real. *Papel* é o nome que se dá à função desempenhada por uma entidade em um relacionamento. O número de entidades participantes do relacionamento define o *grau* do relacionamento. Um relacionamento também pode apresentar *atributos descritivos***.**

Restrições de negócio fazem parte de qualquer problema real. Uma restrição importante é o mapeamento das cardinalidades que expressa o número de entidades às quais outra entidade pode estar associada através de um conjunto de relacionamentos. Para conjuntos de relacionamentos binários podemos ter entre os conjuntos de entidades A e B, cardinalidade *Um para Um***,** *Um para Muitos*, *Muitos para Um* e *Muitos para Muitos*. Essas cardinalidades dependem da semântica das situações reais que está sendo modelada.

Em muitas situações reais, a existência da entidade A depende da existência da entidade B, então A é dita dependente da existência de B, implicando na destruição de A quando B é destruído. A entidade B, nesse caso, é dita *entidade dominante* (ou forte) e a entidade A é dita *entidade subordinada* (ou fraca) e esta deve fazer parte de um conjunto de relacionamentos um para muitos com a entidade forte que a define.

Toda estrutura lógica do banco de dados pode ser expressa graficamente através de um diagrama, o *Diagrama Entidade-Relacionamento(DER)*. Esse diagrama é composto por retângulos, que representam os conjuntos de entidades, elipses, que representam os atributos, losangos, que representam os conjuntos de relacionamentos, linhas, que unem os atributos aos conjuntos de entidades e os conjuntos de entidades aos conjuntos de relacionamentos, elipses duplas, que representam atributos multivalorados e retângulos em linhas duplas que representam entidades fracas. A notação do DER difere de autor para autor, assim, o exemplo da Figura 3.1, apresenta o diagrama utilizando a notação apresentada por [5] e representa parte de um modelo para o controle de táxis conforme a especificação descrita no anexo A.

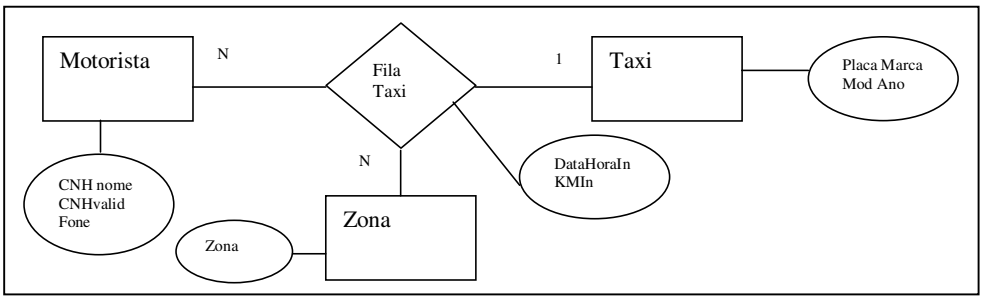



Figura 3.1: Diagrama Entidade-Relacionamento

Apesar de ser possível representar a maioria dos bancos de dados apenas com os conceitos básicos do modelo entidade-relacionamento, foi criado mais tarde por Teorey[18] e seus co-autores, uma extensão desse modelo. No modelo estendido, abstrações como generalização, especialização, agregação, restrições de projetos, como participação parcial ou total, representação para atributos multivalorados, compostos e derivados ganharam representação. A Figura 3.2 mostra uma abstração de generalização/especialização e uma agregação (MT) segundo a notação apresentada em [4].

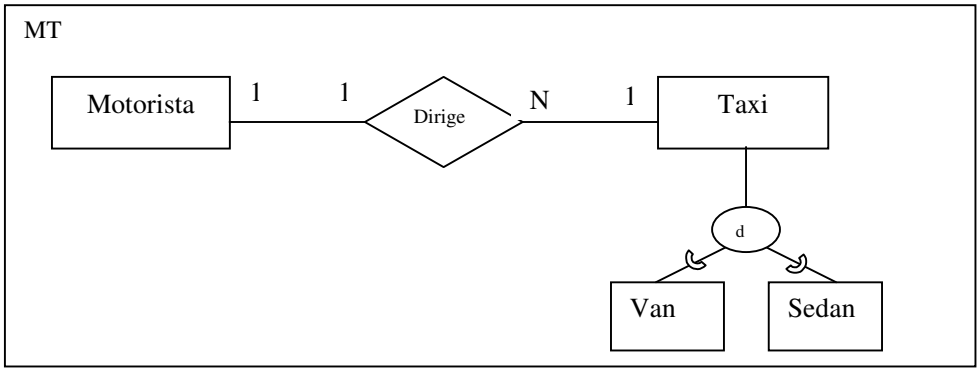



Figura 3.2: Modelo entidade-relacionamento estendido

O modelo entidade-relacionamento estendido agregou bastante significado de representação à modelagem de dados, tornando-se um modelo semântico bastante completo e representativo para esse tipo de modelagem.

### *3.2. RM/T*

O modelo relacional ampliado RM/T foi primeiramente definido por Codd [2]. Uma diferença entre RM/T e o modelo entidade-relacionamento, é que o RM/T não faz distinções desnecessárias entre entidades e relacionamentos que são considerados um tipo especial de entidade. Outras diferenças observadas são que os aspectos estruturais e de integridade do modelo são mais ampliados e definidos de forma mais precisa no RM/T. O modelo inclui seus próprios operadores especiais, além dos operadores do modelo relacional básico.

O funcionamento do modelo implica [3]:

- *entidades* são representadas por *relações E*, que registram a existência das entidades e *relações P* que registram certas *propriedades dessas entidades*.
- pode existir uma variedade de *relacionamentos* entre as entidades (associações, subtipos, etc.). O RM/T inclui uma estrutura de *catálogo formal* através do qual o sistema toma conhecimento dos relacionamentos (restrições de integridade).
- existem vários *operadores* de alto nível para facilitarem a manipulação dos vários objetos RM/T (relações E, relações P, relações de catálogo, etc.).

São três as categorias de entidades representadas no sistema:

- *entidades-semente*, são as entidades que têm existência independente.
- *entidades-características*, descrevem ou caracterizam uma outra entidade. São entidades dependentes da existência da entidade que descrevem.
- *entidades-associativas*, representam o relacionamento de muitos para muitos entre duas ou mais entidades.

As entidades podem ter propriedades e qualquer entidade pode ter uma propriedade cuja função seja identificar ou atribuir outra entidade relacionada. No RM/T todas as chaves primárias e externas são substitutos, que é gerado sempre que um novo representante de entidade é criado. Esse valor gerado é único com relação a todos os valores substitutos que existem ou já existiram no banco de dados e tem a garantia que nunca vai mudar (*surrogate*). A *relação E* de cada entidade contém os *surrogates* de todas as suas instâncias. Esse *surrogate* é a base de todas as referências dentro do sistema. Cada entidade pode ter quantas *relações P* forem necessárias para representar suas propriedades, sendo todas elas ligadas à relação E da mesma entidade através do *surrogate.* O modelo, através de regra de integridade, proíbe que exista uma tupla em qualquer relação P sem que haja um *surrogate* relativo a essa tupla na relação E da entidade. A figura 3.3 exemplifica as relações E e P para o exemplo do Táxi.

TAXI

| |
|---|
| 325 |
| 368 |
| ..... |
| ..... |

TAXI_MOTORISTA

| | |
|---|---|
| 325 | Mário |
| 368 | Luís |
| .... | ..... |
| .... | .... |

Figura 3.3 Representação RM/T da relação E de Táxi e seu relacionamento com Motorista

O relacionamento de subtipo /supertipo também pode ser representado no RM/T. Todas as propriedades do supertipo se aplicam automaticamente ao subtipo, mas não o oposto.

O modelo RM/T inclui diversas regras de integridade novas e preservam as regras existentes no modelo relacional básico. São elas[3]: (i) *integridade de entidade* – as relações E aceitam inserções e eliminações, mas não atualizações; (ii) *integridade de propriedade* – se uma

tupla t aparecer na relação P, então o valor da chave primária de t deve aparecer na relação E correspondente a P; (iii) *integridade de característica* – a entidade característica não pode existir, a menos que a entidade que a descreve também exista; (iv) *integridade de associação* – uma entidade de associação não pode existir, a menos que cada entidade participante da associação também exista; (v) integridade de atribuição – só pode existir se a entidade que atribui também existir no banco de dados; (vi) integridade de subtipo – sempre que um substituto aparecer na relação E para uma entidade do tipo A, esse substituto também deve aparecer na relação E para qualquer tipo de entidade A', para a qual A represente um subtipo.

Além do conjunto de objetivos e regras descritos, o que distingue o RM/T da maioria de outras propostas na área de modelagem semântica é o conjunto de operadores que inclui. Estes operadores permitem, entre outras coisas, a definição de visões do usuário bastante ampla sobre o banco de dados básico comum. O RM/T proporciona um operador único, o operador *Propriedade***,** cujo efeito é reunir todas as propriedades imediatas para um tipo de entidade específica numa única relação n-ária.

Um modelo ampliado como o RM/T pode ser útil como auxílio ao projeto de banco de dados, contudo sua complexidade ultrapassa em muito a complexidade do modelo relacional. Parte das extensões semânticas do RM/T são feitas no dicionário de dados do modelo relacional, através de relações que descrevem os inter-relacionamentos existentes junto a novos operadores.

### *3.3. SDM*

SDM é uma descrição de alto nível para banco de dados baseado em semântica, estruturado formalmente [5]. É um modelo que objetiva a captura de mais significado do ambiente de uma aplicação, através de um conjunto de primitivas. Uma especificação em SDM descreve um banco de dados em termos do tipo de entidades que existe no ambiente da aplicação, a classificação e agrupamento dessas entidades e a interconexão estrutural entre elas. Por acomodar informações derivadas na especificação da estrutura do banco de dados, SDM permite que a mesma informação possa ser vista de diferentes maneiras, fazendo com que seja possível representar diretamente uma variedade de necessidades e requisitos de processamento típicos de aplicações de banco de dados. Uma descrição do banco de dados utilizando SDM poderá ser utilizada como uma especificação formal e uma ferramenta de documentação para um banco de dados, podendo prover a base de apoio a uma variedade de interfaces de usuários e como modelo conceitual para banco de dados na fase de projeto.

Assim, um esquema SDM é uma coleção de entidades organizadas em *classes* e *conexões* entre as classes e atributos derivados. Nas classes são especificados os atributos dos membros e das classes. Existem dois tipos de conexões entre as classes, uma que representa o mecanismo de agrupamento e outro que representa o mecanismo de generalização/ especialização.

Para a conexão de uma associação, estão previstas três formas de especificação [5]: (i) controlada pelo próprio usuário, (ii) baseada no valor comum de um determinado atributo da classe base, (iii) a partir de um conjunto de subclasses derivadas de uma mesma classe base. Um exemplo de uma especificação de associações em SDM para o sistema do táxi poderia ser a apresentada na Figura 3.4.

A conexão que define subclasses é determinada a partir de um predicado que a conecta às superclasses envolvidas. Estão disponíveis quatro formas de especificação do predicado[5]: (i) baseada nos valores dos atributos da superclasse, (ii) baseado no atributo de uma classe que tem a superclasse como contra-domínio, (iii) definidas por operações de conjunto (união, intersecção, etc..) entre superclasses derivadas a partir de uma mesma classe base, (iv) controlada pelo usuário (inclusões e retiradas sob responsabilidade do usuário).

**MOTORISTAS_TAXI**
interclass conection: **grouping** of motoristas on common value of taxi
description: conjunto dos motoristas agrupados por táxi
member attributes:
datahorain
value class: DATES.
Kmin
value class: INTEGER.

Figura 3.4: Representação de especificação de associação em SDM

Um exemplo de uma especificação de subclasses em SDM para o sistema do táxi poderia ser o apresentado na Figura 3.5.

**TAXI_VAN**
interclass conection: **subclass** of TAXI where tipo = 'Van'
description: todos os táxis que são Vans
member attributes:
lugares
value class: INTEGER

Figura 3.5: Representação de especificação de subclasse em SDM

O SDM pode ser considerado uma linguagem de especificação de esquema devido à estrutura natural de sua sintaxe e a grande variedade de primitivas, mas a complexidade envolvida na sua especificação inibe a possibilidade de que seja implementado por um SGBD.

### *3.4. TAXIS*

TAXIS [8] é uma linguagem para projetos de sistemas de informação interativos, que integra a captação dos aspectos estruturais e comportamentais de uma aplicação através de mecanismos de abstração. TAXIS oferece as facilidades do gerenciamento de banco de dados relacionais, um significado para a especificação semântica de restrições de integridade e um mecanismo de tratamento de exceções, integrado numa única linguagem, através da qual os conceitos de classe, propriedade e relacionamento de generalização são providos.

Para descrever classes, os seguintes grupos de categorias estão disponíveis[8]: (i) *chaves* – identifica uma instância, (ii) *características* – agrupam as propriedades invariantes ao longo do tempo, (iii) *atributos* – abrigam as propriedades que variam ao longo do tempo. TAXIS permite a definição de meta-classes para representar seus atributos gerais. Classes são sempre relacionadas através de mecanismos de especialização para facilitar a representação semântica. Oferece um conjunto pré-definido de classes a partir das quais novas classes podem ser especializadas. As operações de banco de dados atuam sobre hierarquias de objetos.

Existem três objetos em TAXIS: *tokens*, representam constantes, *classes*, descrevem conjuntos de tokens que compartilham as mesmas propriedades e *metaclasses*, descrevem coleções de classes e podem fornecer valores totalizados das classes que pertencem à coleção.

Na Figura 3.6, vemos um exemplo de especificação para a classe táxi. A classe variable-class é uma classe pré-definida que suporta um objeto do tipo relação.

```
VARIABLE-CLASS TAXI with
keys
        Taxi_id: Placa
characteristics
        Placa: integer;
        Marca: string;
        Modelo: string;
        AnoFab: date;
Attribute-properties:
        AnoFab: OVER 1960;
end
```

Figura 3.6: Especificação da classe Táxi no modelo TAXIS

No modelo TAXIS, transações são consideradas classes e podem ser especificadas a partir da especialização de outras transações, através da especialização de parâmetros que modela a parte estática da aplicação e da redefinição dos procedimentos. Através de pré-requisitos, ações e resultados, o usuário poderá fatorar o corpo de uma transação em restrições de validação (*constraint checks*) semi-independentes e ações que podem estar associadas com uma transação durante a definição da mesma ou indiretamente através de herança.

Exceções também podem ser vistas como classes e assim, podem ser especializadas. Toda exceção estará relacionada a um procedimento de resultado ou de pré-requisito e será ativada sempre que os procedimentos resultem num valor falso (não *true*). O procedimento que chama a transação deverá indicar qual a transação que deverá ser invocada caso a exceção seja ativada. Por exemplo, a figura 3.7 define uma exceção *táxi_não_encontrado*, associada ao pré-requisito *existente?*, e definir a transação *apresenta_lista* para tratar a exceção.

```
TRANSACTION-CLASS Requisita_táxi
…..
action Atende_Chamada: Obtem_táxi(zona, datahorain) exec-handler Valida_táxi for
                        táxi_não_encontrado is apresenta_lista
…..
```

Figura 3.7: Especificação de uma transação no modelo TAXIS

Em resumo, TAXIS é fortemente baseada na abstração de herança (IS-A) utilizando-a para estruturar dados e procedimentos de uma aplicação, incluindo as expressões, transações e exceções. Provê uma metodologia para tratar restrições semânticas de integridade, auxilia a organização de projetos através das especializações sucessivas, propondo um formalismo para todos os aspectos envolvidos em uma aplicação.

### *3.5. S-SQL*

S-SQL é uma interface que estende a capacidade de captação semântica dos gerenciadores de banco de dados relacionais que ofereçam a linguagem SQL [10]. A interface implementa características semânticas, tais como, classificação, generalização, agregação,

derivação de classes, *surrogates* e atributos multivalorados. A idéia da interface, segundo o autor[10], surgiu devido a rejeição do mercado aos modelos semânticos. Criando uma interface portável, buscou-se prover os SGBD relacionais com mais de inteligência para responder as interações dos usuários preservando o investimento feito pelas empresas em SGBD relacionais.

Ao se propor a interface foram propostas as linguagens de definição de dados (LDD) associada ao modelo. Para construir o esquema da aplicação, a LDD provê instruções para criar classes (Create Class), alterar classes (Alter Class), excluir classes (Drop Class), definir chaves (Key), definir subclasses (Subclass of), indicar a pertinência da subclasse (Total / Partial), indicar o tipo de generalização que a subclasse assume (Covering, Overllaping, Disjoint, Partitioning), declarar uma subclasse derivada (Derived subclass of), entre outras. Alguns exemplos são apresentados abaixo:

*Create Class* Motorista (CNH integer, Nome char(30), CNHValid char(10),
Fone{char(15)}) *Key* (CNH)
*Covering Subclasses of* Taxi *are* Taxi_Van, Taxi_Sedan
*Alter Class* Taxi *add* (cor char(10))
*Alter Class* Motorista *drop* (fone)

Para facilitar a visualização do modelo, a interface S-SQL provê uma representação gráfica que pode ser visualizada na Figura 3.8, representando a generalização de classes.

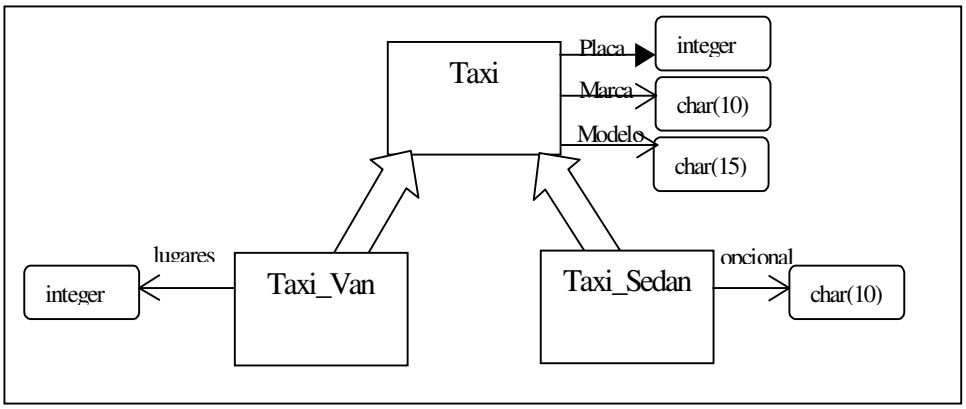



Figura 3.8: Representação gráfica da interface S-SQL

Além da LDD, uma linguagem de manipulação de dados (LMD) foi proposta, buscando dar ao usuário uma linguagem de interação mais fácil e intuitiva do que a linguagem SQL padrão. A LMD do S-SQL é traduzida posteriormente para a SQL padrão pela própria interface. Através da LMD da S-SQL é possível a qualificação das instâncias, baseado nos valores dos atributos das instâncias componentes de uma agregação e efetuar consultas através de junções relacionais. Um *surrogate* é criado automaticamente através da interface no momento de se definir uma relação. Consultas que utilizem comparações entre *surrogates* podem ser efetuadas desde que essas comparações sejam feitas entre *surrogates* de uma mesma classe ou entre *surrogates* de classes especializadas a partir da mesma classe base. O valor de um surrogate é disponibilizado pela interface S-SQL através do atributo identificador padrão, **<***Nome-da-Tabela***>#.** Para a inclusão de objetos em uma agregação a S-SQL aplica predicados que identifiquem essas instâncias nas suas respectivas classes e só incluirá a instância se o predicado for válido para uma única instância e se for válido para todas as instâncias da classe componente. A retirada de objetos de uma agregação pode ser feita pela qualificação dos valores dos atributos das instâncias componentes. O acesso aos atributos de uma superclasse, a

partir de uma subclasse é transparente para o usuário. A interface S-SQL provê o operador *IS-A* quando for necessário especificar a qual subclasse um determinado *surrogate* está associado, como também, para especificar instâncias envolvidas em uma agregação. Para tratar valores de atributos multivalorados, o operador IN é provido pela interface, assim como operadores de conjuntos, igualdade(=), contém(=>) e contido(<=) utilizados para comparações entre atributos multivalorados de um mesmo tipo base. Funções como Min e Max podem ser utilizados com atributos multivalorados do tipo inteiro e a função Count para atributos do tipo Char. O exemplo a seguir recupera as instâncias da classe táxi que tenham ano de fabricação a partir de 1990.

*Select* Taxi.Placa, Taxi.Marca, Taxi.Modelo
*From* Taxi
*Where* Taxi.AnoFab > 1990

A especificação de elementos de um atributo multivalorado é feita através da inclusão dos valores dos elementos separados por vírgulas e entre chaves. Por exemplo, a sentença da S-SQL para a inclusão de um motorista que tivesse dois telefones seria

*Insert into* Motorista (CNH, nome, CNHvalid, fone)
*Values* ("02660934590", "Regina Moraes", "26/12/2007",
{"19-3441.6645", "19-8121.6511"})

A inclusão e exclusão de elementos de um atributo multivalorado são feitas através do comando Update. Os sinais + e – antes da chave de valores dos elementos indicam respectivamente a inclusão e a exclusão de elementos do conjunto. Por exemplo, a sentença a seguir adiciona um novo número de telefone a uma instância já existente.

*Update* Motorista
*Set* fone = +{"19-3404.7105"}
*Where* Motorista.CNH = "02660934590"

Uma interface que implementa um modelo semântico deve ser a única responsável pela coerência do esquema tanto na criação quanto na efetivação de modificações ao longo do tempo. A S-SQL restringiu a sete operações básicas, categorizadas em três grupos[10]:

1. Modificação do conteúdo de uma Classe (Inclusão/Exclusão de um atributo, Inclusão/Exclusão de uma chave)
2. Modificações no conjunto de Classes (Inclusão/Exclusão de uma Classe)
3. Modificações nos Relacionamentos (Conexão de uma classe como subclasse em uma categoria existente)

A interface S-SQL foi implementada como uma "casca" para um SGBD relacional que tem como base a linguagem SQL[10]. Essa decisão de implementação trouxe algumas vantagens, como a similaridade das operações da interface com a linguagem SQL, facilitando o entendimento dos usuários que já estavam acostumados com a linguagem.

### *3.6. MEASUR*

Semiótica pode ser definida, baseado em Peirce[11], como a ciência dos signos e dos processos significativos (semiose) na natureza e na cultura. Tem por objeto de investigação, todas as linguagens possíveis. A investigação semiótica abrange virtualmente todas as áreas do

conhecimento envolvidas com as linguagens, tais como a lingüística (verbal), a matemática (dos números), a biologia (da vida), as artes (estética) etc. A divisão da semiótica é sintaxe, semântica e pragmática que trata de estruturas, significados e utilização de sinais.

Semiótica Organizacional (SO) é um estudo que utiliza os conceitos da semiótica e tem por base que todo comportamento organizado é afetado pela comunicação e interpretação de sinais pelas pessoas, individualmente ou em conjunto, no grupo onde estão inseridas[11].

MEASUR (*Methods for Eliciting, Analysing and Specifying User's Requirements*) foi introduzido por Stamper [17]. MEASUR é um conjunto de métodos orientado a normas (*norm-oriented methods*), que tem como objetivo lidar com todos os aspectos dos projetos de sistemas de informação que se relacionam com o uso de sinais, suas funções no significado das comunicações e intenções e suas conseqüências sociais. Com a utilização de MEASUR, estaremos tratando Métodos de Articulação de Problemas (*Problem Articulation Methods* – PAM – utilizado nos estágios iniciais do projeto quando se encontra um problema vago e complexo), Métodos de Análise Semântica (*Semantic Analysis Methods* – SAM – auxilia os usuários a extrair e representar seus requisitos de uma maneira formal) e Métodos de Análise de Normas (*Norm Analysis Methods* – NAM – apresenta um significado para especificar padrões gerais de comportamentos dos agentes num sistema de negócios). Além desses métodos principais, MEASUR ainda conta com Análise de Controle e Comunicação (*Control and Communication Analysis*) e Análise de Meta-Sistemas (*Meta-Systems Analysis*). Esse conjunto de métodos permite que se inicie com um problema vago e desestruturado e gradualmente se refine esse modelo até obter um problema preciso o suficiente para derivar um conjunto de soluções técnicas. MEASUR auxilia a solução de uma gama variada de problemas, que requeiram intervenção organizacional ou social para serem solucionados[16].

No Método de Análise Semântica, após a extração de requisitos feita no PAM, é representado o contexto do problema num modelo formal, as funções requeridas serão especificadas num modelo ontológico que irá descrever uma visão dos agentes responsáveis no domínio do negócio. O significado do sinal usado no modelo semântico, para representar o mundo do negócio, é tratado como um relacionamento entre o sinal e as ações apropriadas. No anexo C, a tabela C1 apresenta os conceitos envolvidos nesse método de análise. O modelo ontológico delineia um contexto que envolve os conceitos e as terminologias usados no domínio particular do problema. Permite uma semântica contextual porque toda palavra ou expressão é ligada com seus antecedentes.

Uma dependência ontológica pode ser definida da seguinte forma: dado dois objetos X e Y, se a existência de Y depende da existência de X e Y só existe enquanto X existir, então o relacionamento de dependência entre X e Y é chamado de dependência ontológica. O objeto X é chamado de antecedente e Y é seu dependente. A dependência ontológica será representada por linhas que tem antecedentes do lado esquerdo, e dependentes do lado direito.

Essa modelagem exige que seja analisada cada palavra do texto que especifica o problema a ser modelado, procurando-se identificar todos os aspectos semânticos que o envolve, sejam eles explícitos ou implícitos. Devido a isso, a análise é bastante trabalhosa. Para efeito de exemplo, foi modelada apenas uma parte da especificação do sistema do táxi (ref. ao anexo A) cujo modelo ontológico é apresentado na figura 3.9. A separação das características do modelo pela análise da especificação do sistema de táxis é a que segue:

- Identificação dos agentes (substantivos que detêm responsabilidades; definir agente raiz):
  Raiz –sociedade; Agentes – Empresa, Pessoas, Motorista,Usuário, Operador
- Identificação dos *affordances* (demais substantivos, verbos)
  Affordances: Táxi, fila de táxi, chamada, gerenciar, dirigir, atualizar, examinar, fazer.
- Identificar Determinantes:
  Pessoa: #id, #local; Táxi: #id, #km.

- Agrupamento: (papel, todo/parte, genérico/específico, relações)
  Papel: Supervisor – papel assumido por um operador;
  Genérico/Específico: Pessoa (genérico) e Usuário, Motorista, Operador (específicos).

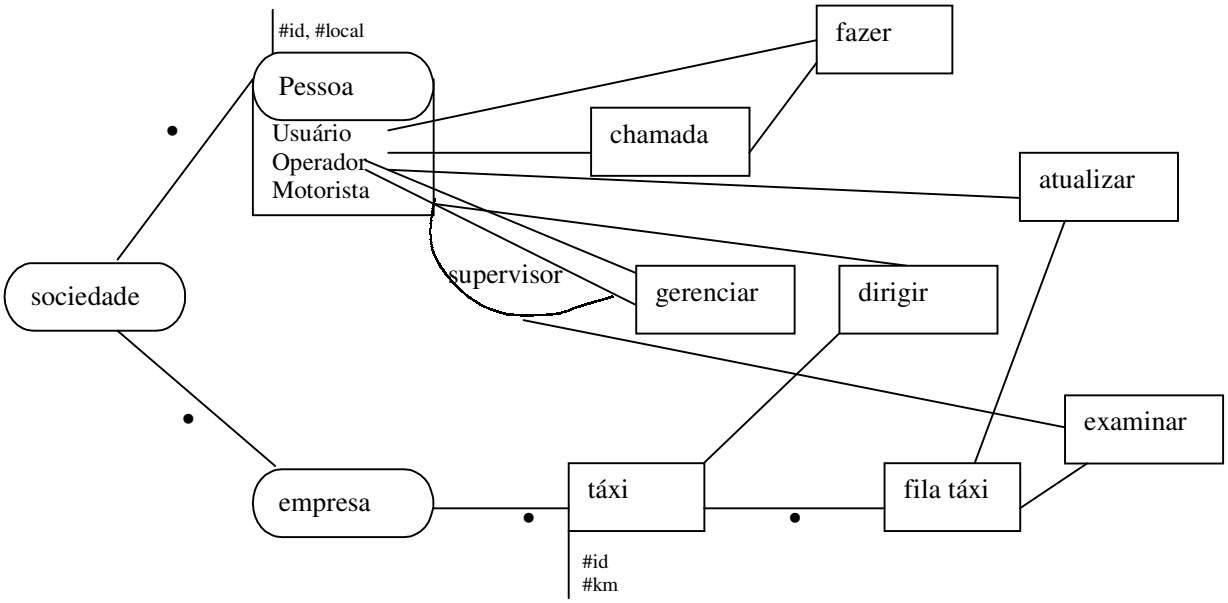



Figura 3.9: Modelo Ontológico para trecho da empresa de táxi

### *3.7. UML – UNIFIED MODELING LANGUAGE*

Nos anos 90, os primeiros movimentos do mundo OO começaram a sugerir meios de utilizar metodologias OO no projeto de banco de dados. Os desenvolvedores da OML – *Object Modeling Language* publicaram um estudo que descrevia como se poderia utilizar OML para projetar bancos de dados [14]. O uso da modelagem do projeto de banco de dados é muito utilizado, excedendo a utilização da modelagem de aplicativos mas, normalmente, se restringe à modelagem de tabelas, colunas e relacionamentos. Através do diagrama de banco de dados da UML poderemos representar aspectos complementares a esse, tais como, tablespaces, visões, restrições (*constraints*), gatilhos (*triggers*), índices, procedimentos armazenados (*stored procedures*) e domínios. O *profile* (extensão da UML que conserva seus princípios básicos) para projeto de banco de dados da UML adiciona os estereótipos necessários para a representação desse tipo de modelagem. Alguns ícones também podem ser utilizados para auxiliar a visualização dos aspectos da modelagem. No anexo D a figura D.2 apresenta esses ícones.

Dessa forma, uma representação para o relacionamento entre Táxi e Motorista através da fila de táxi poderia ser a apresentada na Figura 3.10.


Figura 3.10: Relacionamento entre entidades

A representação de um auto-relacionamento é mostrada na Figura 3.11.

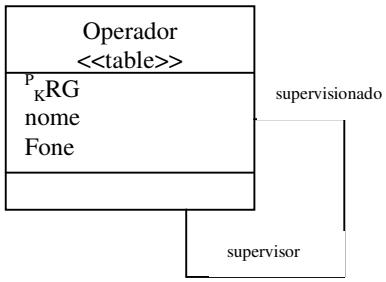



Figura 3.11: Auto-relacionamento entre entidades

Uma visão é definida na UML como uma classe, com o estereótipo <<view>>. Uma visão pode ser derivada de uma ou mais tabelas. O relacionamento da visão com suas tabelas "pais" é modelado como uma dependência com o estereót ipo <<derived>>. As abstrações de generalização/especialização e todo/parte têm representações bastante similares às respectivas representações no modelo entidade-relacionamento e podem ser verificadas na Figura 3.12.

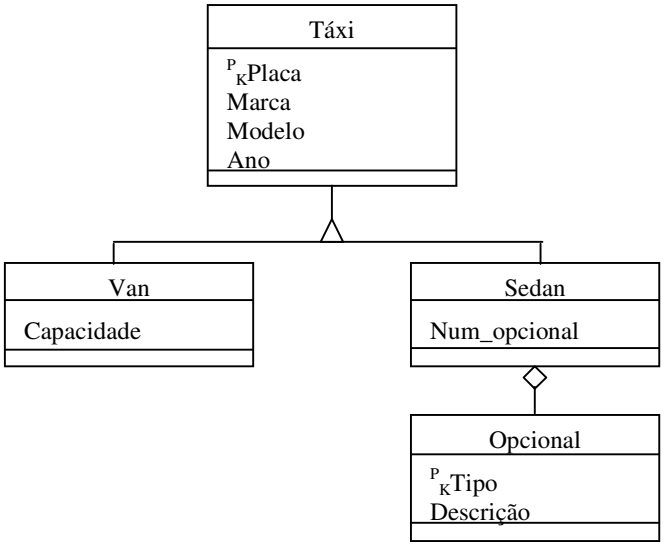



Figura 3.12: Representação de generalização/especialização e todo/parte

Na UML, entidades são modeladas com o padrão de classes da UML. No diagrama de banco de dados, na primeira divisão teremos o nome de entidades e seu respectivo estereótipo, na segunda divisão teremos os atributos com as representações de chave primária, estrangeira e na terceira divisão chaves adicionais <<PK>> ,<<FK>> ou <<Unique>>, restrições de *check* <<check>>, gatilhos <<trigger>>, índices <<index>>, procedimentos armazenados <<SP>>. A Figura 3.13 apresenta esses conceitos representados na classe Táxi. Na divisão onde se encontram os atributos, o projetista pode representar o domínio do atributo se assim o desejar. Atributos compostos são modelados como uma estrutura de domínio, onde é mostrado cada um dos atributos simples que o formam (por identação). Um atributo multivalorado será modelado como uma classe separada.

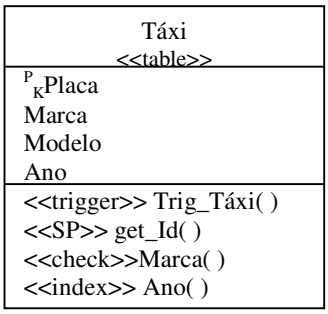



Figura 3.13: Representação de restrições, gatilhos, índices na UML

Relacionamentos são chamados associações na UML e suas instâncias são os links. Uma associação binária é representada por uma linha que conecta as entidades participantes do relacionamento e pode opcionalmente ter um nome. Um atributo de relacionamento é modelado como uma caixa que é conectada à linha de relacionamento por uma linha tracejada. As multiplicidades dos relacionamentos são modeladas na forma de participações mínimas e máximas de cada entidade participante de um relacionamento. Agregação deve ser modelada com um diamante vazio junto à entidade que representa o todo. Composição também representa o relacionamento entre um objeto todo e suas partes componentes que são destruídos quando o todo é destruído e deve ser modelado com um diamante cheio junto à entidade que representa o todo. Entidades fracas podem ser modeladas utilizando uma construção de associação qualificada. Um triângulo branco representa uma especialização/generalização disjunta, enquanto que um triângulo preenchido representa a sobreposição das entidades especializadas[4] (veja Figura D.1 no anexo D). A representação das abstrações utilizadas na modelagem de dados baseados na UML são bastante similares às representações utilizadas no modelo entidade-relacionamento estendido.

Os diagramas da UML conduzem ao contexto do banco de dados. Assim, devemos estar preparados para pensar num diagrama entidade-relacionamento quando terminarmos o diagrama de seqüência e fizermos os ajustes no diagrama de classes respectivos com visibilidade, coleções, atributos ou propriedades específicas da nossa necessidade.

Uma importante questão a ser considerada é a navegação, ela é bastante importante para definirmos os relacionamentos entre as entidades. Quanto aos atributos, se tivermos um atributo de uma classe que está mapeado para um campo de uma entidade no banco de dados, devemos marcá-lo com o modificador *persistente*. Esse modificador pode ser usado tanto para uma classe como para um atributo específico, ou mesmo vários atributos.

## 4. MAPEAMENTO PARA GERENCIADORES DE BANCO DE DADOS

Nessa seção iremos apresentar como é feito o mapeamento dos modelos semânticos apresentados, para o esquema dos gerenciadores de base de dados. Mapear abstrações oferecidas por um modelo semântico em um SGBD, envolve uma série de decisões de implementação, tais como, desempenho, espaço de armazenamento, manutenção de integridades entre outros. Quando se trata de persistência de dados, nem sempre temos padrões bem estabelecidos. Assim, quando se vai definir um esquema para traduzir uma modelagem, tenha sempre em mente que: (i) é preciso compreender sua tecnologia antes de decidir aplicá-la, a escolha da tecnologia pode significar a diferença entre um projeto bem-sucedido e um fracasso ilimitado; (ii) seja flexível em sua abordagem de projeto, não tome decisões de projeto definitivas antes de ter desenvolvido protótipos de suas implicações em aplicações de produção; (iii) se você tem

preocupação com portabilidade, preste atenção que tipo de gerenciador escolher, normalmente gerenciadores objeto-relacional e orientados a objetos ainda não atingiram uma condição que possibilite uma portabilidade tranqüila [7].

### *4.1. MODELO ENTIDADE-RELACIONAMENTO*

O modelo entidade-relacionamento é mais utilizado para a modelagem de sistemas que utilizam gerenciadores de banco de dados relacionais. Por esse motivo e também pela limitação de espaço desse trabalho iremos nos restringir a esse enfoque.

Toda entidade regular que é apresentada no modelo entidade-relacionamento dá origem a uma tabela com os mesmos nomes e os mesmos atributos. Atributo determinante dá origem a chave primária e chaves estrangeiras devem ser especificadas. Uma entidade de ligação dá origem a uma tabela com chave primária, atributos, mais a(s) chave(s) externa(s) e possíveis atributos do(s) relacionamento(s) absorvido(s). A forma de traduzir uma entidade fraca é unir essa entidade e o relacionamento que a associa à entidade regular em uma tabela, contendo a chave externa da entidade regular e os atributos próprios da entidade fraca.

A estrutura da chave primária de um conjunto relacionamento depende do mapeamento das cardinalidades do conjunto de relacionamentos. Se tivermos uma cardinalidade de relacionamentos de muitos para muitos, a chave primária do relacionamento será a união das chaves primárias dos conjuntos de entidades participantes do relacionamento. Para relacionamentos muitos para um ou um para muitos, a chave primária pode ser a chave primária do conjunto de entidades que participa do "lado muitos", já se o relacionamento é um para um, qualquer uma das chaves primárias pode ser usada como chave primária do relacionamento.

Relacionamentos nem sempre exigem um depósito de dados (tabela) na sua implementação. Quando o relacionamento for implementado como uma tupla de uma tabela, conterá tantas referências (chaves externas) quantas forem as ocorrências de entidades participantes, mais os eventuais atributos próprios do relacionamento. Um auto-relacionamento sempre gera tabela contendo chaves externas que apontam para ocorrências da entidade e, eventualmente, atributos próprios do auto-relacionamento. Relacionamentos cuja existência não é obrigatória (0,1: 1 ou 0,1:N) impõem a criação de um depósito de dados.

Generalização / Especialização pode ser transformada para o modelo relacional de duas maneiras distintas. Na primeira delas, cria-se uma tabela para entidade do nível mais alto e uma tabela para cada entidade do nível mais baixo com seus atributos, mais a chave primária da entidade do nível superior. Na outra maneira cria-se uma tabela para cada entidade do nível mais baixo com os atributos próprios, mais todos os atributos herdados do nível mais alto. Esse método só pode ser usado quando todas as entidades do nível mais alto são representadas no nível mais baixo. No anexo B pode-se encontrar um resumo das regras de mapeamento que foram descritas nessa sessão.

### *4.2. RM/T*

No modelo RM/T, toda entidade é realizada como uma tabela no banco de dados relacional. Entidade-semente por ser independente, entidade-característica por definir o lado múltiplo de uma relação da qual depende (por exemplo, as linhas de uma nota fiscal ou os itens de um pedido) e entidade-associativa, pois define a relação muitos-para-muitos de uma associação entre duas entidades. Toda entidade é passível de distinção e o RM/T cria um substituto (surrogate) que é controlado pelo sistema. Esses substitutos estão na relação E de cada entidade e serão mapeadas como chaves primárias da relação no modelo relacional. As

relações P que possuem o mesmo substituto podem ser agrupadas numa mesma tabela relacional. O mapeamento de subtipos e supertipos pode ser feito segundo as mesmas regras já descritas para o modelo entidade-relacionamento [3].

### *4.3. SDM*

Como já dito na seção 3.3, o objetivo do SDM é definir uma maneira de especificar o esquema de banco de dados e não propriamente a preocupação da implementação em um SGBD. Desta forma, o SDM pode utilizar os conceitos já descritos no modelo entidade-relacionamento para traduzir o modelo para tabelas relacionais. Cada <CLASS_NAME> deve ser traduzida numa tabela relacional cujos campos estão especificados na cláusula *member atributes:* do modelo SDM e inclui seus respectivos tipos de dados e restrições de campos (tais como a cláusula not null). As chaves primárias da tabela estão igualmente especificadas no modelo SDM sob a *cláusula identifiers:*.Tipos de usuários estão definidos em classes de tipos no modelo SDM e podem ser utilizados como domínio dos campos de uma tabela relacional tal como indicado no modelo SDM. Através da cláusula *interclass connection:* do modelo SDM pode-se identificar os relacionamentos por agrupamento ou generalização/especialização. Similarmente ao modelo entidade-relacionamento, pode ser feita a tradução para esses relacionamentos [5].

### *4.4. TAXIS*

Pouco material foi encontrado a respeito da maneira como um sistema modelado com o TAXIS deve ser traduzido para um modelo relacional. O que é apresentado a seguir é baseado em deduções feitas pelo autor baseadas em [8]. O modelo TAXIS é fortemente baseado no conceito de generalização/especialização. Uma VARIABLE-CLASS é claramente uma representação de uma entidade generalizada, a partir da qual as especializações devem seguir. Nesse caso, toda VARIABLE-CLASS do modelo TAXIS irá gerar uma tabela no banco de dados relacional, suas *characteristics* serão os campos dessa tabela, a unicidade da tabela será definida pela *keys* e as restrições de domínio dos campos definidas pelo *attribute-properties*. Uma AGGREGATE CLASS define um relacionamento entre tabelas no modelo relacional. Os componentes comportamentais do TAXIS irão direcionar algumas restrições de domínio, inserções de cláusulas check nas tabelas criadas, definições de triggers e stored procedures no modelo relacional.

### *4.5. S-SQL*

Na S-SQL, basicamente toda classe irá corresponder a uma tabela cuja interface acrescentará um atributo destinado a conter o *surrogate* da tupla. Classes que possuem atributos multivalorados irão gerar tantas tabelas quantos atributos multivalorados tiver, além da tabela base e implicam em uma junção entre a tabela base da classe e a tabela que armazena os elementos do atributo e só é possível uma correta implementação se o SGBD utilizado prouver o *outer join*. Consultas que possuem mais de um atributo multivalorado numa cláusula Select acaba gerando um resultado que associa cada tupla de um dos atributos com cada uma das tuplas do outro atributo e vice-versa. Cada chave declarada irá gerar automaticamente um índice. Na tradução de transações, cada transação S-SQL poderá gerar várias transações SQL. Ao traduzir uma transação todos os desmembramentos necessários são processados pela interface, que poderá criar identificadores automáticos que se façam necessários. Na inclusão de uma instância, para cada *surrogate*, a interface gera uma consulta baseada no predicado dado, a fim de recuperar o *surrogate* associado e completar a cadeia de cada instância componente que

é definida pelos predicados. As transações de exclusão são transformadas em transações de consultas para que a interface possa efetuar a exclusão requisitada e todas as exclusões que devem ser propagadas para as tabelas relacionadas. Para as transações que recuperam instâncias de classes participantes de uma hierarquia de generalização, o processo é bastante próximo ao processo de tradução de junções implícitas, mas a interface precisará recorrer ao dicionário de dados quando for preciso relacionar os atributos às superclasses onde estão definidos. O operador de pertinência IS-A e o operador IN são traduzidos para o operador de igualdade (=). Para as transações de inclusão numa rede de generalização, a interface deve determinar através do dicionário de dados, a qual superclasse está associada cada atributo. Para a inclusão em classes derivadas com base em predicados, a interface irá gerar para cada inclusão na classe base uma inclusão na classe derivada. É quase obrigatória a utilização de cláusulas Group by e Having para solucionar consultas que envolvam agrupamentos.

De uma forma geral, as traduções que envolveram as abstrações de agregação e generalização foram efetuadas de uma maneira simples, o que não ocorreu com as traduções que envolvem atributos multivalorados que requisitaram traduções mais complexas. Foi apresentado um resumo das traduções da interface S-SQL para o modelo relacional, uma abordagem mais completa pode ser obtida em [10].

### *4.6. MEASUR*

Para a tradução do modelo MEASUR todo agente e *affordance* do tipo entidade deve ser tratada como entidade e conseqüentemente gerar uma tabela no modelo relacional. *Affordances* do tipo ação, irão gerar os relacionamentos e devem seguir as regras do modelo entidade-relacionamento para a sua tradução em tabelas relacionais. Papéis representados no modelo devem gerar entidades que se relacionam numa cardinalidade 1x1. Relação parte-todo e genérido-específico, geram entidades com relacionamentos 1xN. Determinantes geram atributos nas suas respectivas tabelas. Após a tradução, a cardinalidade dos relacionamentos deve ser analisada e as normas associadas com o modelo semântico devem ser satisfeitas.

Para a tradução desse modelo para um modelo orientado a objeto ou objeto-relacional as seguintes regras devem ser utilizadas: agentes e *affordances* do tipo entidade geram objetos, affordances do tipo ação, geram comunicação entre atributos ou serviços, papéis geram restrições de atributos e conjuntos, relação parte-todo geram objetos aninhados ou separados, relação genérico-específico geram herança de classes e finalmente determinantes geram atributos em suas respectivas classes [16].

### *4.7. UML – UNIFIED MODELING LANGUAGE*

A transformação de um modelo de dados UML em um esquema de banco de dados relacional usa as técnicas para transformar um modelo entidade-relacionamento e adiciona outras para aproveitar alguns dos recursos expandidos desses modelos. Durante a modelagem, classificadores estereotipados com <<type>> e os tipos compostos são transformados em tipos de usuários no esquema de banco de dados e utilizados para a tipagem de atributos. Cada classe UML do modelo é transformada em uma tabela e cada atributo dentro da classe se torna uma coluna dessa tabela. Os atributos que foram marcados com PK são definidos como uma chave primária (*Primary Key*). Classes que participam de relacionamentos vários-para-vários e ternários, classes de ligação e classes de um relacionamento de herança devem ser mapeadas, utilizando as mesmas regras existentes para o modelo entidade-relacionamento. Quando houver uma indicação na modelagem estereotipada como <<identity>> isso deverá gerar um surrogate (SEQUENCE no Oracle ou IDENTITY no SQL Server). Toda marca com estereótipo <<unique>> deve receber uma restrição UNIQUE como restrição de coluna. Para os atributos

que não receberem marcadores {nullable} é preciso que se adicione a palavra chave NOT NULL à coluna da tabela [6]. Procedimentos armazenados e gatilhos indicados na modelagem devem gerar correspondentes procedimentos e gatilhos no seu esquema.

Da mesma forma que com esquemas relacionais, você cria um esquema objeto-relacional em dois passos básicos: a criação de tipos e tabelas (a partir de tipos persistentes e para associações muitos-para-muitos e ternárias). A transformação de um modelo de dados UML em um esquema de banco de dados objeto-relacional admite a representação de classes e interfaces para tipos estruturados através de Create Type. O Oracle8, Informix e DB2 suportam essas espécies de tipo, embora com sintaxe e características diferentes. O tipo define atributos, mas não define suas conexões a outros tipos, nem define suas chaves primárias ou regras do negócio (restrições check), você terá que colocá-las diretamente nas tabelas que criar utilizando a declaração Create Table e que utilize um determinado tipo. No Oracle 8 e no DB2 é possível criar tipos estendidos e no Informix além de tipos estendidos é possível criar a generalização de tipos em estruturas de herança. Embora no Oracle8, você não possa se referir diretamente a um tipo de objeto para um atributo, pode definir um REF para tal tipo.

Um atributo-objeto é um atributo de um tipo de objeto que é ele mesmo um objeto. Nesse caso, estaríamos criando a possibilidade de um objeto embutido em uma tabela e assim, cada linha dessa tabela constitui mais um objeto com identidade. Em um diagrama UML, essa é uma associação com multiplicidade 1..1 ou 0..1 para a classe que representa o objeto embutido. Uma alternativa a isso é usar uma referência, que é um ponteiro para um objeto armazenado em algum lugar fora da tabela atual. O "ponteiro" , nesse caso, é o identificador do objeto, seu OID. O SGBDOR constrói um OID para cada objeto de um tipo estruturado que você cria e então se pode referir ao objeto colocando o OID em um valor de coluna na tabela referente. Assim, em um SGBDOR você pode representar associações como chaves estrangeiras ou como referências. Se você utilizar chaves estrangeiras terá que realizar a junção das tabelas em consultas, se usar referências, poderá navegar até o objeto referenciado por intermédio de expressões SQL, evitando a junção, mas dificultando a integridade referencial, uma vez que se você remover um objeto, quaisquer referências àquele objeto permanecem exatamente como estão, sendo necessário utilizar gatilhos ou um esquema de encapsulamento de procedimentos armazenados para garantir essa integridade. Uma associação com multiplicidade "vários" pode ser representado pelo tipo coleção, que pode ser um vetor, um conjunto ou uma tabela. O Oracle8 oferece o tipo VARYING ARRAY e tabela aninhada, o Informix oferece SET, MULTISET e LIST. O objetivo de todos esses tipos é o de coletar vários objetos em um único valor de coluna.

Para o comportamento, os bancos de dados objetos-relacionais variam ligeiramente a abordagem dos relacionais. O Oracle8 acrescenta métodos aos tipos de objeto. No que diz respeito a regras de negócios há muito pouca diferença entre um sistema relacional e um sistema objeto-relacional. Uma diferença é a presença do tipo de objeto ou estruturado que não tem restrições, apenas as tabela as tem.

O mapeamento de um modelo de dados UML a um esquema Orientado a Objetos (OO) é direto, embora haja algumas questões a considerar. Muitas questões surgem da natureza inerente do projeto de objetos persistentes, aparecendo em todos os produtos SGBDOO e no próprio padrão ODMG. O padrão ODMG 2.0 é uma avaliação de desempenho abrangente para as capacidades de um SGBDOO, mas ainda não são oferecidas facilidades de definição de esquemas em conformidade com o padrão. Alguns, como POET e Versant, oferecem variações sobre o padrão, outros como ObjectDesign e Objectivity têm uma maneira completamente diferente de definir seus esquemas. Transações, caching cliente/servidor, associações complexas, chaves e extensões são conceitos externos às linguagens de programação OO, mas são essenciais para bancos de dados. Não podendo ignorar essas questões, todos os produtos OO oferecem mapeamento para elas, mas de formas diferentes.

Quase todos os produtos SGBDOO começaram utilizando linguagens de programação OO como suas linguagens de definição de esquemas. Acrescentando elementos de linguagem ou bibliotecas de classes para suportar a persistência, esses produtos transformaram as definições de interface em declarações de esquemas. Embora o processo de mapeamento para SGBDOO seja mais simples do que no caso de banco de dados relacionais, a promessa de transparência ainda não foi cumprida.

Os valores nulos são um tanto problemáticos em produtos SGBDOO, pois a maioria das linguagens de programação OO não tem nenhum conceito de um valor nulo. Bindings C++, Smalltalk e Java não suportam os tipos literais nullable, nem nulos para objetos.

A associação é onde os produtos SGBDOO dão mais trabalho na geração do esquema. Pode-se representar um vínculo entre dois objetos de maneiras diferentes, as alternativas básicas são: (i) Atributo – colocar um único objeto relacionado diretamente como membro do objeto relacionador, que é proprietário do objeto (isso se traduz em um objeto sem OID, e consiste em valores embutidos no objeto relacionador); (ii) Referência – embutir um único objeto relacionado como membro de referência do objeto relacionador utilizando um manipulador ou referência persistente . Nesse caso, a integridade referencial deve ser gerenciada pelo aplicativo; (iii) Coleção de Objetos – coloca-se um template de coleção para objetos como membro do objeto relacionador. A integridade referencial é gerenciada pela coleção como propriedade dos objetos parte; (iv) Coleção de Referência – coloca-se um template de coleção para referência de objetos como membro do objeto relacionador. A integridade referencial deve ser mantida pela aplicação; (v) Relacionamento – coloca-se um template especial de coleção de referências de objetos como membro do objeto relacionador. O template de coleção mantém integridade referencial de ambos os lados do relacionamento. Em alguns produtos SGBDOO, as referências também podem permitir ativação adiada (*lazy activation*) de objetos, permitindo que se recupere o objeto do servidor apenas quando precisar usá-lo. Outros sistemas ativam todos os objetos que pode alcançar a partir de um objeto ativado (por exemplo o SGBDOO Ozone).

Com algumas raras exceções, produtos SGBDOO localizam o comportamento no cliente, tornando o SGBDOO um pouco menos flexível. A tarefa do SGBDOO é materializar objetos persistentes em memória do cliente e então deixar aqueles objetos desempenharem sua tarefa, coordenando através de transações, caches compartilhados e assim por diante. A vantagem dessa abordagem é a drástica simplificação da interface comportamental.

Comportamento limitador é simples no esquema OO. Você cria um programa para fazê-lo. Não há restrição ou gatilho de banco de dados no esquema OO; é apenas mais um comportamento que você anexa a uma classe. Assim o modelo OO transforma restrições complexas em problemas de programação em vez de problemas de bancos de dados.

## 5. CONCLUSÕES

O projeto de bancos de dados e a sua crescente utilização pelas empresas estão relacionados à necessidade do domínio de informações para garantir qualidade de aplicações, respostas consistentes, ações aderentes e rápidas às necessidades do negócio, assegurando a competitividade de um mercado altamente mutável.

Apesar de vários modelos semânticos terem sido propostos, a maioria esmagadora das empresas utiliza o modelo entidade-relacionamento, na sua forma estendida (com a extensão foi provido alguns mecanismos típicos para a reutilização) que vagarosamente tem sido substituído pelo modelo UML nos últimos anos. Mesmo assim, essa substituição tem se dado devido a utilização de ferramentas integradas baseadas na UML, que inicialmente são utilizadas para as demais fases de projeto e acaba levando o usuário a aderir a UML também para o modelo de dados.

Os mecanismos do Banco de Dados Relacional são insatisfatórios para implementação da hierarquia de generalização, e os Bancos de Dados Orientados a Objeto ainda não conquistaram espaço junto às empresas para implementação de bancos de dados corporativos. É possível que quando isso acontecer (se acontecer) a substituição mais rápida do modelo entidade-relacionamento se dê, devido a menor impedância do modelo de dados baseado na UML com a representação de objetos.

Na verdade, apesar de algumas críticas ao modelo entidade-relacionamento, as representações providas pelo modelo estendido atendem quase a totalidade das necessidades da modelagem necessária para os sistemas empresariais, devendo-se a isso a sua grande aceitação. Os modelos que foram propostos ao longo desses anos, pouco complemento trouxeram às representações já providas pelo modelo entidade-relacionamento e não justificaram o esforço necessário para que as empresas investissem na substituição de um modelo conhecido e tão bem aceito pelos profissionais envolvidos em projetos de sistemas.

**REFERÊNCIAS**


[1] CHEN, P. **The entity-relationship model – Toward o unified view of data**, ACM Transactions on Database Systems, pp. 9-36, Março 1976.

[2] CODD, E. F. **Extending the database relational model to capture more meaning**, ACM Transactions on Database Systems, pp. 397-494, Dezembro 1979.

[3] DATE, C. J. **Introdução a Sistemas de Banco de Dados**, Tradução da 4ª Edição Americana, p. 674, Editora Campus, 1996.

[4] ELMASRI, R.; NAVATHE, S. B. **Fundamental of Database Systems**, 3th Edition, p. 955, Addison Wesley, 2000.

[5] HAMMER, M.; MCLEOD, D. **Database Description with SDM: a semantic database model**, ACM Transactions on Database Systems 6(3), pp. 351-386, Setembro 1981.

[6] MEDEIROS, E. **Desenvolvendo Software com UML**, Pearson Makron Books, p. 264, 2004.

[7] MULLER, R. J. **Projeto de Banco de Dados usando UML para modelagem de dados**, Tradução do título original Database design for smarties using UML for data modeling, Morgan Kaufmann – Berkeley, p. 495, 2002

[8] MYLOPOULOS, J.; BERNSTEIN, P. A.; WONG, H. K. T. **A language facility for design database-intensive applications**, ACM Transactions on Database Systems 5(2), pp. 185-207, Junho 1980.

[9] NAIBURG, E. J.; MAKSIMCHUCK, R. A. **UML For Database Design**, Object Technology Series, Addison-Wesley, p. 300, 2001,

[10] OLIVEIRA, R. C. N., **S-SQL: Uma Interface Semântica**, Dissertação de Mestrado , Instituto de Matemática Estatística e Ciência da Computação, UNICAMP, Agosto 1989.

[11] PEIRCE, C.S. **Collected Papers**, Cambridge, Mass: Harvard University Press, 1958.

[12] RAMAKRISHMAN, R.;GEHRKE, J. Database Management System, 3th Edition, p., 2003.

[13] REED, P.R. Jr **Desenvolvendo Aplicativos com Visual Basic e UML**, Makron Books, p. 462, 2000.

[14] RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. Object-Oriented Modeling and Design, Englewood Cliffs, NJ, Prentice Hall, 1992.

[15] SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de Banco de Dados**, p. 778, Makron Books, 1999.

[16] SIMONI, C. A. C. **A Prática de Desenvolvimento de Software e a Abordagem da Semiótica Organizacional.,** Dissertação de Mestrado, Instituto de Computação, Unicamp, Brasil, 2003.

[17] STAMPER, R. K. **Information in Business and Administrative Systems**, John Wiley and Sons, New York, 1973.

[18] TEOREY, T.;DONGQING Y.; JAMES, P. **A logical design methodology for relational databases using the extended entity-relationship model**, Computer Surveys 18 (2), pp. 198-222.

## ANEXO A

## *ESPECIFICAÇÃO DO SISTEMA DE TÁXI*

O sistema de táxi foi utilizado nesse trabalho para exemplificar o modelo. Uma especificação do sistema resumida é apresentada nesse anexo, para que o leitor possa ter a idéia do problema exemplificado. Essa especificação do problema foi disponibilizada como estudo de caso para a disciplina de Banco de Dados do Instituto de Computação no ano de 2004.

**Despacho e controle de Táxis via terminais móveis ligados on-line com um sistema multi-usuário.**

Uma empresa de agenciamento de corridas de táxis está utilizando um sistema de rádio digital para gerenciar a frota de táxis associados. A empresa tem cerca de 500 táxis associados. Cada associado tem instalado no seu táxi um equipamento da empresa que funciona como um terminal de computador bastante simplificado. Este equipamento tem um teclado simplificado e uma tela de cristal líquido para visualizar mensagens. Cada terminal só recebe mensagens destinadas a todos os táxis ou destinadas a ele próprio. Quando um taxista envia uma mensagem de volta ao sistema o terminal, automaticamente, inclui a identificação do táxi, a data, a hora e a kilometragem atual na mensagem enviada. O sistema deve poder atender a uma demanda de 10.000 transações por dia, e até 1500 despachos por hora durante os períodos de pico. Do ponto de vista do motorista do táxi, o sistema funciona da seguinte maneira:

Quando o motorista começa a trabalhar, ou termina com uma corrida, ele envia uma mensagem para o sistema dizendo qual o número de "espera" (zona geográfica) que deseja e aperta o botão desta transação. Sua posição na fila para aquela espera é mostrada na tela LCD do terminal móvel;

Quando o escritório recebe uma chamada para um táxi, ao primeiro táxi na fila para aquela área é oferecida a corrida pelo computador, o qual envia um sinal (alarme sonoro) para o terminal do táxi. Informações sobre a corrida são também mostradas na tela; Se o motorista deseja a corrida, ele aceita apertando um botão. Se ele estiver fora do veículo ou ignorar o sinal por mais de 60 segundos, o sistema retira a corrida dele e deixa uma mensagem a respeito. O motorista pode também escolher que rejeita uma corrida e o sistema o leva para o fim da fila;

Quando o motorista chega num endereço ele deve enviar uma transação de início de corrida. Se nenhum cliente estiver lá, ele pode apertar um botão que o colocará de volta no topo

da fila (o tempo transcorrido e a kilometragem percorrida desde a aceitação da corrida até o momento de receber o passageiro é controlado). Quando ele termina uma corrida o motorista deve enviar uma transação de fim de corrida.

Operadores no escritório de despacho fazem a inicialização das requisições de táxis. Cada operador está sentado à frente de um terminal de vídeo, no qual um formato padrão de pedido está esperando entradas. À medida que o operador vai digitando o nome, endereço, etc. o sistema vai movendo de campo em campo no formato padrão. Um diretório completo de ruas deve estar disponível no disco, e o sistema automaticamente verifica o endereço entrado para verificar se o mesmo é verdadeiro ou se já não houve problemas com este cliente.

A corrida mais comum é aquela que o cliente pede com antecedência e determina um data/hora de início e local de apanhar. Há também corridas que são pedidas a qualquer momento com o cliente esperando num determinado endereço e corridas que são oferecidas para os associados a partir de um determinado evento, como por exemplo, o fim de um show.

A empresa vive das mensalidades dos taxistas associados e de convênios com empresas que utilizam de "*vouchers*" . Estes "*vouchers*" são créditos que as empresas fornecem a seus clientes para utilizarem o sistema de táxis.

Quando a hora de iniciar uma corrida estiver próxima, o sistema determina qual a zona do endereço, e automaticamente avisa o táxi no topo desta fila para oferecer a corrida.

Um supervisor no escritório pode reservar, suspender, reiniciar, ou dar prioridade a corridas para qualquer unidade. Ele também pode enviar mensagens confidenciais, cancelar chamadas, criar corridas que são feitas numa forma repetitiva, monitorar a carga em qualquer zona, e examinar filas de táxis e corridas em tempo real.

## ANEXO B

### *MAPEAMENTO DO MODELO ENTIDADE-RELACIONAMENTO*

Nesse anexo é apresentado um resumo do que foi descrito na sessão 4.1 a respeito do mapeamento do modelo entidade-relacionamento para esquemas de banco de dados relacionais. Dessa forma,

*Entidades Fortes* **-** cada conjunto de entidades fortes no MER gera uma tabela.

*Entidades Fracas* **-** gera tabelas com seus atributos, mais chave primária das entidades fortes dos quais elas dependem.

*Entidades De Ligação* **-** gera tabelas com chave primária, atributos mais as chaves externas e possíveis atributos dos relacionamentos absorvidos.

*Relacionamentos (N:N)* **-** gera tabela que tem as chaves das entidades que ele associa mais os atributos próprios do relacionamento.

*Relacionamentos (1:N) Ou (1:1)* **-** podem ser representados por novas tabelas ou através de chaves estrangeiras.

*Auto-Relacionamentos* **-** sempre geram tabelas.

*Relacionamentos De Grau Maior Que Dois* **-** sempre geram tabelas.

*Relacionamentos De Existencia Não Obrigatória* **-** sempre geram tabelas.

*Relacionamentos Diversos Com Mesmas Entidades* **-** apenas um relacionamento pode ser absorvido, gerando tabelas dos outros relacionamentos.

*Generalização E Especialização* **–** (duas alternativas)

1- Criar uma tabela para entidade do nível mais alto e uma tabela para cada entidade do nível mais baixo com seus atributos mais a chave primária da entidade do nível superior.

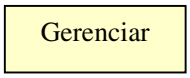
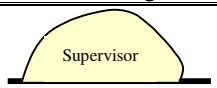
2- Criar uma tabela para cada entidade do nível mais baixo com os atributos próprios, mais todos os atributos herdados do nível mais alto. Esse método só pode ser usado quando todas as entidades do nível mais alto são representadas no nível mais baixo.

- *Agregação* – relacionamentos de agregação são mapeados como tabelas com seus atributos descritivos e chave primária composta da chave primária do relacionamento, mais a chave primária do relacionamento dentro da agregação.

## ANEXO C

## *REPRESENTAÇÃO DOS CONCEITOS DO MÉTODO MEASUR*

Tabela C.1: Representação dos Conceitos do Método MEASUR

| Conceito | Significado | Representação |
|---|---|---|
| Agente | Ator que constrói e interage com a realidade. É indispensável que se defina um agente raiz (por exemplo, sociedade, nação, etc.). |   |
| *Affordance* | Primitivas semânticas que representam padrões possíveis de ações ou comportamentos do agente. |   |
| Relação Ontológica | Define o limite ou o período relativo da existência do *affordance* que se relaciona ao agente que o possui. |  |
| Determinante | Agentes e *affordances* têm propriedades que são invariantes e diferenciam um do outro. | #kilometragem |
| Papel | Um agente pode ter um papel para executar quando está envolvido nas relações e nas ações. |   |
| Todo/Parte | Define uma subdivisão possível do agente ou representa uma hierarquia. |  |
| Genérico/Específico | Se possuírem propriedades não compartilhadas ou diferentes. |   |

## ANEXO D

### *UML – UNIFIED MODELING LANGUAGE*

Nesse anexo são apresentados o diagrama de entidade-relacionamento do sistema de táxi utilizando a UML, os ícones utilizados pela UML para representação de objetos de modelagem semântica e as regras resumidas de mapeamento para gerenciadores de banco de dados objeto-relacional e orientado a objetos.

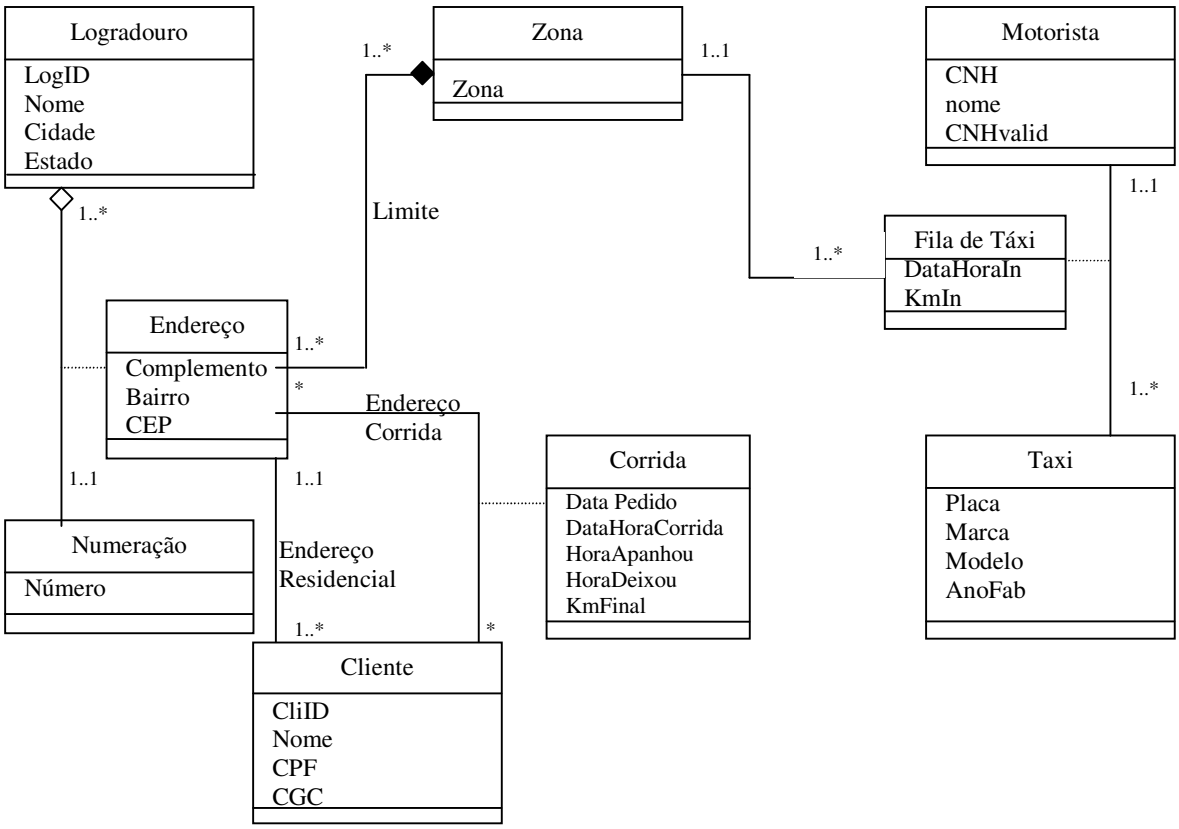



Figura D.1: Esquema conceitual UML para o sistema de Táxi

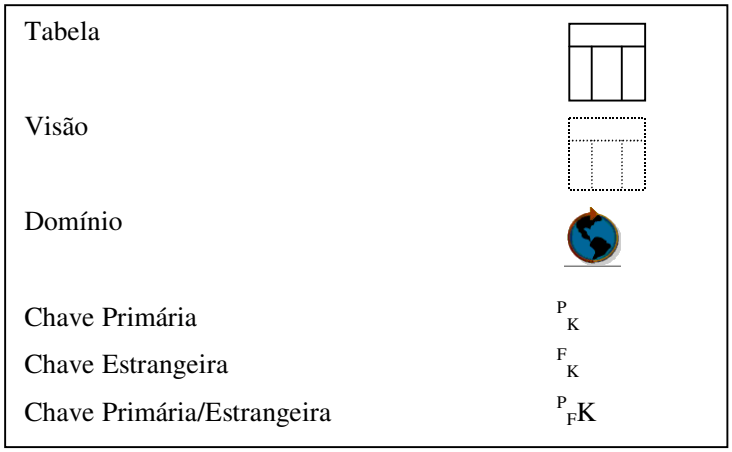



Figura D.2: Ícones utilizados pela UML

Mapeamento do diagrama UML para Gerenciadores de Banco de Dados Objeto-Relacional (SGBDOR):

Criar quaisquer tipos, tabelas ou objetos auxiliares de que necessita para criar e utilizar extensões SGBDOR reutilizáveis, como cartuchos do Oracle8, o DataBlades do Informix ou o UDB do DB2.
A classe UML torna-se o tipo de objeto (estruturado), geralmente com uma tabela correspondente àquele tipo, para conter as linhas de objetos, mas possivelmente com mais do que uma só tabela dessa natureza.
O tipo UML torna-se um tipo distinto se baseado em tipo embutido ou tipo de objeto, se possuir atributos.
O atributo UML na classe torna-se atributo do tipo de objeto.
O tipo de atributo na classe torna-se o tipo de atributo no tipo de objeto através da tabela de transformação de tipos e/ou objetos e tipos distintos.
Se o marcador de atributo UML for {nullable}, o atributo possuirá restrição Null, caso contrário Not Null.
Se o atributo UML possiuir inicializador, acrescente cláusula DEFAULT à coluna.
Para classes sem generalização (raiz ou independente) e identidade implícita, crie chave primária de número inteiro; para {oid} acrescente colunas com marcadores {oid} à restrição de Primary Key; ignore agregações compostas e classes de associação.
Para subclasses, acrescente a chave de cada classe pai a uma restrição Primary Key e a uma restrição Foreign Key; alternativamente para produtos SGBDOR totalmente em conformidade com SQL3, você poderá usar uma cláusula UNDER para representar o relacionamento, mas para todos os demais terá que colocar a restrição de chave estrangeira nas tabelas que definir com seu tipo de objeto.
Para classes de associação, crie um tipo de objeto e acrescente uma chave primária de cada tabela de desempenho de papéis à restrição Primary Key e à restrição Foreign Key.
Se o tag for {alternate oid=<n>}, acrescentar colunas à restrição Unique.
Acrescente Check para cada restrição explícita.
Crie colunas Foreign Key na tabela de referenciação para cada papel 0..1, 1..1 na associação; alternativamente, use uma referência a um tipo de objeto para declarar o objeto como atributo em si para objetos singelos ou para uma coleção como um vetor de referências para objetos múltiplos relacionados.

Crie Primary Key para agregação composta com Foreign Key para tabela de agregação (com opção Cascade); acrescente uma coluna adicional para Primary Key; alternativamente, use um tipo de objeto para armazenar o agregado na própria tabela, quer através de um atributo de objeto (para um objeto simples) quer através de uma coleção, como um vetor ou uma tabela aninhada.
Otimize classes de associação binária, mudando para tabela do lado para-muitos, onde apropriado.
Crie tabelas para associações muitos-para-muitos e ternárias sem classes de associação através de chaves estrangeiras.
Crie restrições Primary Key e Foreign Key a partir de chaves de tabelas de desempenho de papéis em associações muitos-para-muitos e ternárias.
Crie métodos para tipos de objetos para operações nas classes UML correspondentes; utilize os formatos 'purity level' (Oracle8) apropriados para operações com marcador {readonly} ou {query}[6].

Mapeamento do diagrama UML para Gerenciadores de Banco de Dados Orientado a Objeto (SGBDOO):

A classe UML <<persistent>> torna-se uma classe persistentes no SGBDOO.
A interface UML realizada por uma classe <<persistent>> torna-se uma interface SGBDOO para linguagens que suportam interfaces (Java, ODL) ou uma classe de base abstrata com apenas membros virtuais puros e nenhum membro de dados para aquelas linguagens que não têm interfaces (C++, Smalltalk).
Um classificador de tipo UML torna-se um enum ou typedef conforme indicado.
O atributo UML na classe torna-se atributo na classe SGBDOO com transformações e restrições de tipos apropriados.
Use tipo de literal nullable (nullable_short, por exemplo) se aparecer o marcador {nullable} e o binding que estiver usando suporta valores nulos (ODL); caso contrário ignore-o (C++ ou Java).
Se o atributo UML tiver um incializador, acrescente o código inicializador ao construtor, ou como parte do método construtor ou em uma lista de inicialização de membro C++.
Para subclasses, inclua a especificação de superclasse na declaração da classe.
Para classes de associações, crie uma classe com os atributos da classe de associação.
Para identidade específica de objeto {oid} ou chave candidata {alternate oid}, especifique uma declaração de chave se o seu binding a suportar, caso contrário, forneça métodos apropriados para verificação de restrições de singularidade sobre conjuntos de objetos (C++).
Acrescente um método à classe apropriada para cada restrição explícita e assegure-se de que o sistema chame aquele método sempre que o sistema exija que a restrição seja satisfeita.
Crie um relacionamento para cada associação binária que não tenha classe de associação com o objeto ou tipo de coleção apropriados derivando das multiplicidades da associação. Use relacionamentos inversos a não ser que haja setas explícitas na associação.
Crie um relacionamento em classes de associação para cada vínculo de papel a uma outra classe.
Crie um código ou use recursos do SGBDOO para propagar exclusões de quaisquer associações de agregação composta, conforme seu SGBDOO específico assim o exigir.
Crie uma classe de associação para associações ternárias e relacionamentos da classe de associação para as classes associadas com os tipos de dados apropriados, dadas as multiplicidades [6].