IMPLEMENTAÇÃO DE UM BANCO DE DADOS TEMPORAL UTILIZANDO O MODELO ORIENTADO A OBJETO TF-ORM

Fernando Tonial
Curso de Ciência da Computação
Universidade do Vale do Itajaí - UNIVALI
tonial@inf.univali.br

Rogério Gonçalves Bittencourt
Curso de Ciência da Computação
Universidade do Vale do Itajaí – UNIVALI
rgb@inf.univali.br

RESUMO

Um banco de dados temporal armazena vários estágios dos dados, assim como os instantes em que estes diferentes estágios são válidos. Os bancos de dados tradicionais armazenam somente o estado corrente dos dados. Muitas aplicações têm necessidades de informações sobre os dados passados, como Sistemas de Informações e Sistemas de Engenharia. Essas aplicações necessitam representar a duração de eventos, e gerenciar o tempo de vida dos objetos ou operações, requerendo a implementação de um banco de dados temporal para alcançar uma efetiva representação da realidade. Neste trabalho é apresentada uma abordagem para a implementação de bancos de dados temporal, usando um Banco de Dados Objeto-Relacional (SGBDOR), tendo como base o modelo TF-ORM (Temporal Functionality in Objects with Roles Model), um modelo temporal orientado a objetos. Como forma de ilustrar, apresenta-se um exemplo de implementação utilizando o sistema Oracle8i.

Palavras-Chave: Banco de Dados Objeto-Relacional, Banco de Dados Temporal, ModeloTemporal.

Classificação: Banco de Dados

1. INTRODUÇÃO

A modelagem de banco de dados tradicional armazena apenas dados estáticos, não registrando a sua evolução. Já a modelagem de banco de dados temporal, permite armazenar e recuperar todos os estados de um objeto, registrando sua evolução ao longo do tempo, especificando tanto aspectos estáticos, como os aspectos dinâmicos de uma aplicação. Informações temporais são associadas implicitamente aos dados, correspondendo ao tempo de validade (tempo que a informação será válida no banco de dados) e/ou ao tempo de transação (tempo que a informação foi inserida no banco de dados). Hoje, cada vez mais as aplicações necessitam de informações sobre dados passados, como sistemas de informações gerenciais. Quando se almeja estes tipos de aplicações, necessita-se da implantação de um banco de dados temporal.A utilização de um modelo de dados temporal para especificação de uma aplicação não implica, necessariamente, na utilização de um SGBD específico para o modelo. Bancos de dados comerciais podem ser utilizados se existir um mapeamento adequado entre o modelo temporal e o banco de dados utilizado. Um banco de dados (BD) temporal pode ser implementado sobre um BD relacional, orientado a objetos, objeto-relacional e outros. Em cada um deles devem ser preservadas as características individuais, bem como, as regras que regem cada BD.

Este trabalho visa analisar a implementação de um BD temporal sobre um Sistema de Gerência de Banco de Dados (SGBD) objeto-relacional, especialmente no que concerne ao mapeamento das classes representadas no modelo TF-ORM. O modelo temporal orientado a objetos utilizado é o TF-ORM, mapeado para o SGBD Oracle 8.

2. TEMPORALIDADE EM BANCO DE DADOS

Conforme [CAV1995], apesar de muitos esforços, ainda não existem SGBDs temporais comercialmente disponíveis. A maioria dos SGBDs atuais não possui capacidade de armazenar e processar aspectos temporais de forma efetiva. Além disso, os modelos de dados mais difundidos e utilizados para a modelagem de aplicações, como os modelos relacional e entidade-relacionamento (ER), não oferecem recursos para a modelagem dos aspectos temporais.Segundo [EDE1994], a representação dos aspectos temporais na especificação de um sistema de informação é importante por mais de um motivo: o sistema pode apresentar informações temporais a serem introduzidas no banco de dados, sob forma de informação propriamente dita; os processos a serem executados podem apresentar interações temporais; determinadas tarefas podem apresentar pré-condições à sua execução, as quais podem ser representadas através de restrições temporais; condições de integridade temporal do banco de dados podem ser representadas.

2.1. TIPOS DE BANCO DE DADOS TEMPORAIS

Segundo [EDE1994], banco de dados temporal é aquele que, de alguma forma, representa informações temporais. Os banco de dados temporais podem ser classificados em quatro tipos diferentes, conforme a forma utilizada para armazenar os dados temporais: banco de dados instantâneos; banco de dados de tempo de transação; banco de dados de tempo de validade; banco de dados bitemporais.

Banco de Dados Instantâneos

Conforme [CAV1995], banco de dados instantâneos, são bancos de dados (BD) tradicionais, onde os valores disponíveis são apenas os atuais. Em qualquer operação de atualização, o valor anteriormente armazenado é perdido, e somente o novo valor fica disponível. Cada instância do banco de dados pode ser comparada a um "instantâneo" (*snapshot*) do estado atual da aplicação. A manipulação de dados temporais, somente pode ser feita explicitamente, através da inclusão de

atributos definidos sobre um domínio de tempo, e pela sua manipulação através de programas de aplicação. A figura 1 apresenta algumas atualizações feitas em momentos diferentes em um banco de dados instantâneo.

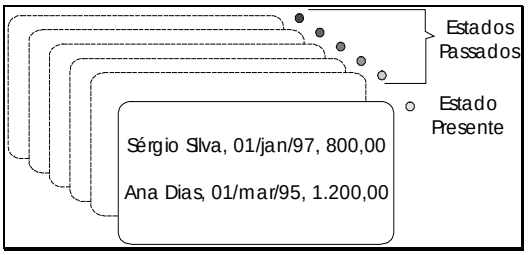

Figura 1 – Banco de Dados Instantâneo
Fonte: [EDE1994]

Banco de Dados de Tempo de Transação

Segundo [CAV1995], os banco de dados de tempo de transação, são BDs que suportam modelos de dados temporais, que possuem somente o tempo de validade definido.

De acordo com [EDE1994], as informações temporais são associadas a cada valor definido, o instante temporal em que foi realizada a transação, sob forma de um rótulo temporal. Este tempo é fornecido automaticamente pelo SGBD. Neste caso, a alteração do valor de uma propriedade, não destrói o valor definido anteriormente, ficando assim, todos os valores armazenados no banco de dados. A identificação dos diferentes estados do banco de dados é feita através do tempo de transação associado a cada informação. A cada atualização, é associado o dia em que esta foi realizada, como mostra a figura 2, sendo este valor válido até o momento em que é realizada uma nova atualização.

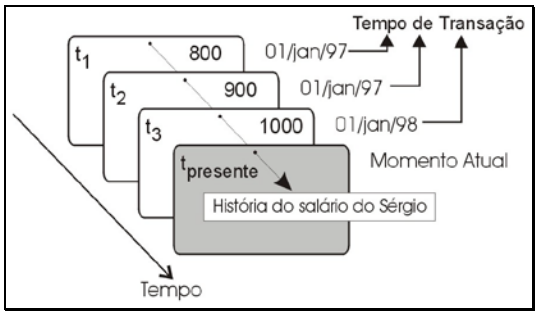

Figura 2 – Banco de Dados de Tempo de Transação
Fonte: [EDE1994]

Ainda para [EDE1994], quando a atualização de um atributo não coincidir com a data em que começa a sua validade, esta data de início de validade pode ser armazenada como um atributo explícito, como demonstrado na figura 3.

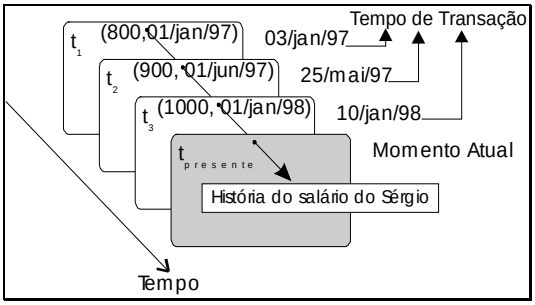

Figura 3 – Banco de Dados de Tempo de Transação
Fonte: [EDE1994]

Banco de Dados de Tempo de Validade

De acordo com [CAV1995], os bancos de dados de tempo de validade, são BDs que suportam modelos de dados temporais, que possuem somente o tempo de transação definido. Neste tipo de BDs, o tempo de validade deve ser armazenado como um atributo explícito, ficando a gerência (atualizações e inclusões) deste, sob responsabilidade dos programas de aplicação.

Conforme [EDE1994], a associação do tempo de transação às informações, muitas vezes não é suficiente para representar a realidade de forma correta. O tempo de validade corresponde ao tempo em que uma informação é verdadeira no mundo real, podendo ser igual ou diferente do tempo de transação. A cada informação é associado não o tempo transação, mas o seu tempo de validade. O tempo de validade deve sempre ser fornecido pelo usuário. Neste tipo de banco de dados, não se tem acesso ao tempo em que a informação foi definida, mas somente o tempo em que ela é valida, como mostra a figura 4.

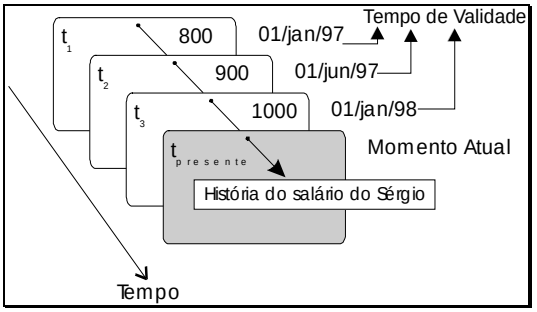

Figura 4 – Banco de Dados de Tempo de Validação
Fonte: [EDE1994]

Banco de Dados Bitemporais

Conforme [CAV1995], os banco de dados bitemporais, são BDs que suportam modelos de dados, que possuem os conceitos de tempo de validade e tempo de transação. Neste tipo de BD, é possível o acesso a todas as instâncias passadas do banco de dados, tanto em relação à história das transações, através do tempo de transação, como à validade dos dados. O estado atual do banco de dados é constituído somente pelos valores válidos no momento.

Segundo [EDE1994], a forma mais completa de armazenar informações temporais, é associando a cada informação, tanto o tempo de transação, como o tempo de validade. É possível ter acesso a todos os estados passados do banco de dados. Tanto a história de transações realizadas, como a história da validade dos dados. Sendo assim, é possível saber não somente o valor atual dos dados, como o valor que era válido em qualquer data passada e também valores futuros, sendo definidos através do tempo de validade. A figura 5 mostra um exemplo deste tipo de banco de dados, demonstrando sua história de atualização do salário de um funcionário.

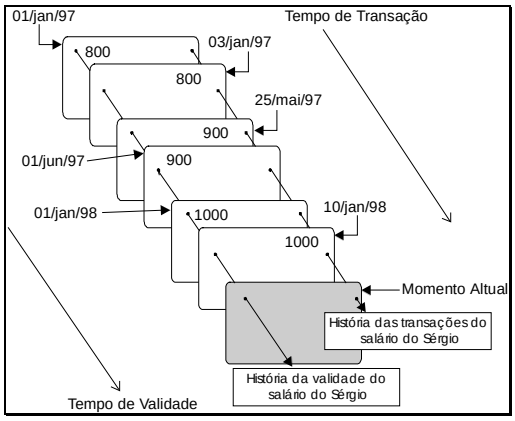

Figura 5 – Banco de Dados Bitemporal
Fonte: [EDE1994]

3. MODELO DE DADOS TEMPORAL ORIENTADO A OBJETOS TF-ORM

Conforme [EDE1994], o TF-ORM (Temporal Functionality in Objects With Roles Model), permite a modelagem dos aspectos estáticos e dinâmicos da aplicação, pois considera todos os estados do objeto, associando informações temporais às propriedades que podem mudar de valor ao longo do tempo. É um modelo bitemporal, ou seja, suporta tanto tempo de transação, quanto tempo de validade. De acordo com [EDE1999], o TF-ORM é um modelo de dados orientado a objetos, que utiliza o conceito de papéis para representar os diferentes comportamentos dos objetos.

O TF-ORM é uma extensão do modelo F-ORM (*Functionality in Objects with Roles Model),* cujo objetivo é a representação da funcionalidade dos sistemas de informação de escritórios, através da modelagem dos recursos (geralmente documentos) e processos (atividades sobre os documentos) por meio de classes. Com o objetivo de permitir a modelagem dos aspectos temporais de uma aplicação, o modelo TF-ORM foi proposto contendo as seguintes extensões ao modelo F-ORM [CAV1995]: um conjunto de tipos de dados temporais e as funções associadas; *timestamps* associados às instâncias e às propriedades dinâmicas; um valor especial *null* para valores de propriedades fora do período de validade; condições temporais adicionadas às regras, escritas em uma linguagem de lógica temporal. Dessa forma, o modelo possui as facilidades necessárias para o suporte à modelagem dos aspectos temporais de uma aplicação. (Ibidem)

De acordo com [CAR1997], o modelo apresenta a definição de dois tipos de propriedades: propriedades estáticas: o valor ao longo do tempo não sofre alterações; propriedades dinâmicas: o valor da propriedade pode sofrer alterações, com o passar do tempo, sendo que todos os valores ficam armazenados, permitindo assim, que todo o histórico de uma instância possa ser recuperado, em qualquer momento.

3.1. CONCEITO DE PAPÉIS

Segundo [EDE1999], em modelos de dados orientados a objetos convencionais, um objeto é criado como uma instância de uma classe, apresentando as propriedades e comportamento desta classe. Este objeto possui um identificador único que o distingue dos demais objetos. Uma vez criado, o objeto pertence a esta classe durante todo o seu tempo de vida, mesmo se, com o passar do tempo, suas características comportamentais evoluam de modo a ser identificado com as características de outra classe. Por exemplo: neste tipo de evolução, considere-se a classe pessoa com duas subclasses, Aluno e Funcionário, como mostra a figura 6. Um objeto pode ser instanciado em somente uma das subclasses, o que significa que ele não pode ser Aluno e Funcionário, e nem passar de Aluno a Funcionário ou vice-versa.

Conforme [EDE1994], outra característica dos modelos orientados a objetos tradicionais, é não permitir a instanciação múltipla de um mesmo objeto.

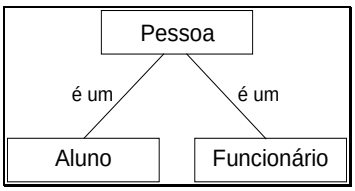

Figura 6 – Herança entre Classes
Fonte: [EDE1999]

De acordo com [EDE1999], a utilização do conceito de papéis resolve estas limitações dos modelos orientados a objetos tradicionais, permitindo a representação da evolução do comportamento de um objeto. Através deste conceito, uma classe pode apresentar diversos papéis, os quais podem ser instanciados dinamicamente. Um objeto continua sendo instância de uma só

classe, mas pode desempenhar diferentes papéis (comportamentos) durante sua existência. Os papéis desempenhados por um objeto, são representados por instâncias destes papéis, as quais podem ser dinamicamente criadas e destruídas durante a existência do objeto.

Um objeto pode apresentar simultaneamente instâncias de mais de um papel, assim como pode apresentar mais de uma instância de um mesmo papel no mesmo instante de tempo. No exemplo anterior, ao invés de utilizar subclasses para representar os comportamentos de estudante e funcionário, seriam utilizados papéis contidos na definição da classe pessoa. A abstração de generalização/classificação continua existindo, estendendo-se o conceito de herança também para os papéis. Um objeto pessoa que fosse somente estudante apresentaria uma instância do papel estudante. A evolução do objeto trocando de papel pode ser representada através da criação da instância de outro papel, e da destruição da instância existente, como mostra a figura 7. O fato de um estudante ser também funcionário seria representado por uma instância de estudante e uma instância de funcionário, as duas existindo simultaneamente. Para a pessoa que apresentar mais de um emprego, cada um destes seria representado por uma instância independente deste papel. (Ibidem)

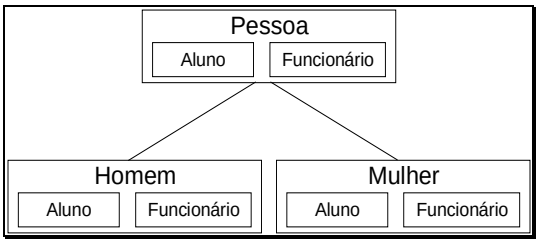

Figura 7 – Herança de Classes com os Papéis
Fonte: [EDE1999]

Conforme [CAR1997], o conceito de papéis tem como principal objetivo separar os aspectos estáticos dos aspectos dinâmicos.

3.2. CLASSES NO MODELO TF-ORM

Segundo [EDE1994], no modelo TF-ORM, as classes são decompostas em três tipos distintos, pré-definidos.

Conforme [CAV1995], o objetivo da distinção entre classes, é promover uma melhor representação para aplicações de automação de escritórios.

Para [EDE1999], os três tipos de classes são:

- Classe de agente (*agent class*): representam pessoas atuando no sistema que está sendo modelado, e cada papel representa um comportamento que esta pessoa pode apresentar na aplicação considerada.
- Classe de recurso (*resource class*): representam as estruturas dos recursos (dados, documentos), com os papéis que os recursos podem desempenhar durante a sua existência.
- Classe de processo (*process class*): integram os agentes e os recursos, descrevendo a organização do trabalho desenvolvida na aplicação, e a cooperação entre os agentes. Os papéis nas classes de processo representam diferentes tarefas a serem executadas no processo.

De acordo com [CAV1995], todas as instâncias de uma classe possuem informações temporais associadas, que podem ser o tempo de criação da instância, o tempo em que esta deixou de existir, e os tempos de suspensão e retomadas de atividades da instância. Essas informações são armazenadas em propriedades especiais pré-definidas.

4. MAPEAMENTO DO MODELO TF-ORM PARA O BANCO DE DADOS OBJETO-RELACIONAL

O mapeamento de um esquema TF-ORM para o *Oracle 8i* é realizado mapeando cada classe, subclasse, papel, propriedades dinâmicas e estado (*state*) para um objeto complexo. Os objetos complexos provenientes das classes, subclasses e papéis, darão origem as tabelas de objetos. Uma característica importante deste mapeamento é a utilização de intervalos no tempo para as propriedades dinâmicas. O modelo TF-ORM utiliza pontos no tempo, associado ao valor da propriedade apenas o tempo de validade inicial e o tempo em que a transação de definição deste valor foi realizada.

Com a utilização de pontos no tempo, para descobrir se uma instância do objeto representa uma informação válida no momento considerado, seria necessário pesquisar a data de validade inicial da próxima instância do objeto. Já com a utilização de intervalos no tempo, cada atributo dinâmico possuirá a data de validade inicial e final associada. Sendo assim, a pesquisa de validade da instância fica restrita à verificação das datas nos atributos *V_TIMEI* (tempo de validade inicial) e *V_TIMEF* (tempo de validade final) da mesma instância do objeto. Portanto, para cada propriedade dinâmica são associadas quatro informações temporais:

- Tempo de validade inicial (*V_TIMEI*): indica o instante de validade da informação.
- Tempo de validade final (*V_TIMEF*): indica o instante em que o dado deixou de ser válido.
- Tempo de transação inicial (*T_TIMEI*): corresponde ao instante de tempo em que a informação foi inserida no banco.
- Tempo de transação final (*T_TIMEF*): corresponde ao instante de tempo da próxima transação realizada sobre o dado.

O detalhamento da etapa de mapeamento pode ser encontrado em [TON2000].

4.1. MAPEAMENTO DAS PROPRIEDADES ESTÁTICAS

As propriedades estáticas (pe_i) são mapeadas como um atributo do objeto complexo da classe a qual esta pertence. Sua forma geral é apresentada na figura 8.

$NOMEDACLASSE\_OT\ (pe_1, pe_2, ..., pe_n);$

Figura 8 – Mapeamento das Propriedades Estáticas

4.2. MAPEAMENTO DAS PROPRIEDADES DINÂMICAS

As propriedades dinâmicas (pd_i), que variam com o tempo, são mapeadas criando-se objetos complexos que possuem, além do próprio atributo, mais quatro informações temporais associadas a este atributo, como visto anteriormente. Já o nome destes objetos complexos são formados pelo nome da classe a qual ele está associado, o nome do atributo e mais as letras "*OT*". A forma geral desta descrição é apresentada na figura 9.

$NOMECLASSE\_NOMEATRUBUTO\_OT\ (pd_1, V\_TIMEI, V\_TIMEF, T\_TIMEI, T\_TIMEF)$

Figura 9 – Mapeamento das Propriedades Dinâmicas

4.3. MAPEAMENTO DAS CLASSES

Para as classes, a criação dos objetos complexos é necessária para podermos fazer a relação entre as outras classes, pois quando um dado faz referência a outra classe, não faz diretamente a ela, mas sim a seu objeto complexo criado. A partir deste objeto é criada a tabela de objetos. O nome do objeto será igual ao da classe, acrescido "*OT*", para podermos distinguir os objetos complexos das tabelas. A criação destes objetos complexos será mostrada mais adiante. Os atributos desse objeto serão as propriedades estáticas e as dinâmicas do papel base (*ROLE_BASE*) e a propriedade dinâmica pré-definida (*dynamic*). A forma geral é apresentada na figura 10.

NOMECLASSE_OT (pe_1, pe_2, ..., pe_n, pd_1, pd_2, ..., pd_n, *dynamic*)

Figura 10 – Mapeamento das Classes

4.4. MAPEAMENTO DAS SUBCLASSES

O mapeamento das subclasses é análogo ao das classes, acrescentando apenas o atributo que será do tipo da superclasse (*atsc*). A forma geral é apresentada na figura 11.

NOMESUBCLASSE_OT (*atsc*, pe_1, pe_2, ..., pe_n, pd_1, pd_2, ..., pd_n, *dynamic*)

Figura 11 – Mapeamento das Subclasses

4.5. MAPEAMENTO DOS PAPÉIS

Com exceção ao papel base (*ROLE_BASE*), onde suas propriedades são incorporadas as classes, todos os papéis são mapeados de forma idêntica às subclasses, pois os papéis tornam-se subclasses de sua classe, apenas acrescentado uma propriedade dinâmica pré-definida (*state*), criando assim os seus próprios objetos complexos. A forma geral é apresentada na figura 12.

NOMEPAPEL_OT (*atsc*, pe_1, pe_2, ..., pe_n, pd_1, pd_2, ..., pd_n, *dynamic*, *state*)

Figura 12 – Mapeamento dos Papéis

A propriedade dinâmica gera um pequeno problema, que é a quantidade que esta poderá ser alterada, fazendo assim, com que este não possa ser um atributo simples em uma tabela, pois devemos guardar todas as alterações destes atributos.

Para resolver este problema, pode-se utilizar dois novos conceitos que o *Oracle 8i* possui: um é o *Varying Arrays* ou *Varray* e o outro é o *Nested Tables*. Os dois conceitos trabalham como um conjunto de registros, do mesmo tipo, dentro de um único atributo, ou seja, uma relação todo-parte entre dois objetos. A diferença fundamental entre eles é que no primeiro, o número de elementos deste conjunto é limitado, como mostra a figura 13(a). Já no segundo, o número de elementos é ilimitado, como mostra a figura 13(b), sendo este último conceito utilizado neste trabalho, pois não se pode definir o número máximo que uma propriedade dinâmica poderá ser alterada.

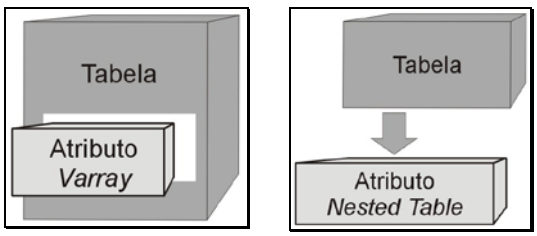

Figura 13 – (a) Atributo *Varray*; (b) Atributo N*ested Table*
Fonte: [EDE1999]

A seqüência SQL para gerar um tipo *DEPTO_OT* utilizando *Nested Table* é demonstrado na figura 14.

```
/* O_DYNAMIC_OT */
CREATE OR REPLACE TYPE O_DYNAMIC_OT AS OBJECT
        (OBJECT_INSTANCE        DATE,
        V_TIMEI                 DATE,
        V_TIMEF                 DATE,
        T_TIMEI                 DATE,
        T_TIMEF                 DATE,
        END_OBJECT              DATE);
```

```
/* DEPTO_DYNAMIC_NT */
CREATE OR REPLACE TYPE DEPTO_DYNAMIC_NT
AS TABLE OF O_DYNAMIC_OT;


CREATE OR REPLACE TYPE DEPTO_OT  AS OBJECT
        (NOMED          VARCHAR2(40),
        DYNAMIC         DEPTO_DYNAMIC_NT);
```

Figura 14 – Cria o tipo *DEPTO_OT* utilizando *Nested Table*

Estes procedimentos criam um tipo que utiliza o conceito de *Nested Table*. Porém, ainda existe uma particularidade para este conceito, que pode ser vista quando cria-se a tabela que irá herdar as características do tipo, como é apresentado na figura 15.

```
CREATE TABLE DEPTO OF DEPTO_OT NESTED TABLE DYMANIC STORE AS DEPTO_DYNAMIC_NT_TAB;
```

Figura 15 – Criação do *Nested Table* da Tabela *DEPTO*

Pode-se observar que, além de criar uma tabela normal, uma outra tabela será criada, *DEPTO_DYNAMIC_NT_TAB*, onde serão armazenados os dados dinâmicos referenciados na tabela *DEPTO*.

Conforme [BAR1999] apud [KOC1997], dentro dos conceitos deste SGBD, existem dois tipos de objetos: os objetos de colunas e os objetos de linhas. Os objetos de colunas são extensões das características já existentes em um banco de dados relacional. Uma tabela pode possuir um tipo de dado criado pelo usuário, ou um *Varray*, ou ainda um *Nested Tables* dentro de uma coluna. Isto para o *Oracle 8i* representa um objeto de coluna. Os objetos de linhas são formados pelo conjunto de objetos de uma tabela de objetos, ou seja, uma tabela onde serão armazenados os objetos referentes a um tipo específico, utilizando-se o comando *CREATE TABLE*. Isto pode ser facilmente compreendido porque, como citado anteriormente, o *Oracle 8i* implementa seus objetos através dos mecanismos de um banco de dados relacional. O comando para a criação de uma tabela de objetos do tipo *PESSOA_OT*, é apresentado na figura 16.

```
CREATE TABLE PESSOA OF PESSOA_OT;
```

Figura 16 – Ciração da Tabela *PESSOA*

Quanto a criação de objetos complexos, o *Oracle 8i* pode criar objetos utilizando o conceito de referência, ou o conceito de todo-parte. Foi mostrada anteriormente, com o exemplo da criação de um tipo chamado *PESSOA_OT*, uma implementação do todo-parte. Para representar as referências, será utilizada a criação do tipo *CLIENTE_PLANO_OT*. A cláusula SQL de criação deste tipo é demonstrada na figura 17.

```
/* CLIENTE_PLANO_OT */
CREATE OR REPLACE TYPE CLIENTE_PLANO_OT AS OBJECT
        (PLANO          REF PLANOSAUDE_OT,
        V_TIMEI         DATE,
        V_TIMEF         DATE,
        T_TIMEI         DATE,
        T_TIMEF         DATE);
```

Figura 17 – Criação do Tipo *CLIENTE_PLANO_OT*

Pode-se observar a utilização da palavra chave *REF* dentro desta cláusula. Ela serve para representar que o atributo *PLANO* faz uma referência ao *Object ID* ou *OID* do tipo *PLANOSAUDE_OT*. Observa-se que não são mencionadas, até o momento, as chaves primárias dos bancos de dados relacionais. Nenhuma das cláusulas SQL para criação de tipos ou de tabelas apresenta qualquer comando que possa criá-las. O *Oracle 8i* possui a característica de atribuir *OIDs* únicos aos objetos que ele armazena, sem que exista a necessidade de um controle sobre este atributo. A *OID* é um dispositivo automático e transparente ao usuário, a menos que este resolva controlar esta propriedade.

5. MANIPULAÇÃO

5.1. CONSULTA

Segundo [BUS1999], na utilização da SQL em um banco relacional, informam-se as tabelas desejadas, quais os campos (colunas) dentro destas tabelas e quais os critérios de seleção, e o SGBD encaminha apenas os valores solicitados. Ao escolher um SGBD Objeto-Relacional (como o *Oracle 8i*), o uso da SQL continua sendo permitido como no SGBD Relacional, mas pode haver algumas alterações. Ainda são informadas as tabelas desejadas e quais os critérios de seleção, podendo ou não fazer referência aos campos (propriedades) dentro destas tabelas, porém, isto iria ferir a propriedade de encapsulamento.

De acordo com [BUS1999], na propriedade de encapsulamento, apenas os métodos do objeto podem ter acesso às suas propriedades. Portanto, ao invés de indicar quais os campos desejados, deve-se escrever um método nesta classe que informe os valores destes campos e, na SQL, informar quais os métodos desejados de cada objeto que obedeça aos critérios de seleção. A mesma observação vale para o processo de seleção, ou seja, a própria execução das cláusulas SQL. Em bancos relacionais, é comum indicar quais os campos que cada registro deve obedecer a certos critérios, o que em bancos orientados a objetos, vai contra a propriedade de encapsulamento, pois com isso, permite-se a outra estrutura, que não o próprio objeto ter acesso a suas propriedades.

Portanto, um SGBD Objeto-Relacional deve permitir que se executem operações via SQL baseadas em métodos de objetos, e não em propriedades. Essa é uma das grandes dificuldades do desenvolvimento orientado a objetos: respeitar a propriedade de encapsulamento. No caso específico da utilização de bancos objetos-relacionais, esta dificuldade aumenta, pois a maioria destes bancos, a exemplo do *Oracle 8i*, não prevêem controle de acesso as propriedades dos objetos, ficando, a cargo do desenvolvedor, implementar esta segurança. Na figura 18, são mostrados dois exemplos de declarações SQL que estão de acordo com o que foi exposto.

```
01 - Select M.ListaCRM() from Medico M
02 - Select M.PesqEspec() from Medico M where M. ListaCRM() = 'Numero do CRM'
```

Figura 18 – Declarações em SQL com os Métodos da Tabela *MEDICO*

O primeiro é um exemplo simples. Sua função é retornar, para todos os objetos selecionados, o resultado do método *LISTACRM()*, porém, pode-se notar algumas peculiaridades:

- Na cláusula *Select* nota-se a indicação de que o valor a ser retornado não será um campo, mas sim o resultado de um método;
- Na cláusula *From* é indicada a tabela (MEDICO) e um "apelido" para os objetos retornados (M).

O segundo é um exemplo um pouco mais complexo. Sua função é retornar, para todos os objetos selecionados, o resultado do método *PESQESPEC()*, e devem ser selecionados apenas os objetos onde o resultado do método *LISTACRM()* seja igual ao número do *CRM* passado.

Para a criação de métodos de objetos utiliza-se o comando *CREATE TYPE BODY*. A sua sintaxe é demonstrada na figura 19.

```
CREATE OR REPLACE TYPE BODY NOME_DO_TIPO IS
MEMBER PROCEDURE/FUNCTION NOME_PROC [RETURN TIPO] IS
        BEGIN
                /* DECLARAÇÕES */
        END NOME_PROC;
```

Figura 19 – Forma Geral do Comando *CREATE TYPE BODY*

Além dos métodos, o *ORACLE* permite as consultas através das visões de objetos, ou *Object Views*. Segundo [FAN2000], visões são tabelas lógicas que não ocupam lugar no banco de dados e podem ser compostas por colunas e agrupamentos de uma ou mais colunas. Sendo assim, visão é o resultado de uma consulta ou seleção. O comando, para a criação de uma visão, é demonstrado na figura 20.

```
CREATE VIEW NOME_DA_VIEW AS
SELECT NOME_COLUNA [,NOME_COLUNA]
FROM NOME_TABELA
[WHERE <CONDIÇÕES>]
[ORDER BY NOME_COLUNA]
[GROUP BY NOME_COLUNA]
```

Figura 20 – Forma Geral para Criação de *Views*
Fonte: [FAN2000]

5.2. INSERÇÃO

Para efetuar a inserção de dados utiliza-se o comando SQL, *INSERT INTO*. Deve-se levar em conta a particularidade que a tabela *DEPTO* é do tipo *DEPTO_OT* e também possui outros objetos complexos, por isto, para efetuar a inserção de dados, deve-se referenciar os tipos. O comando é demonstrado na figura 21.

```
INSERT INTO DEPTO VALUES
( 'FINANCEIRO',
DEPTO_DYNAMIC_NT (O_DYNAMIC_OT('01-01-98', '01-01-98', '', '31-12-97', '', '' ) ),
    DEPTO_NUM_FUNC_NT (DEPTO_NUM_FUNC_OT (4, '31-12-97', '', '31-12-97', '') )
);
```

Figura 21 – Inserção de Dados na Tabela *DEPTO*

6. CONCLUSÃO

O surgimento de aplicações ditas "não convencionais" tais como: Sistemas de Informações Geográficas, Sistemas de Engenharia, etc., tornou obrigatório a evolução dos bancos de dados para melhor representar a realidade destas aplicações. Surgiu assim, os bancos de dados objeto-relacional temporais.

O banco de dados objeto-relacional temporal é um tema bastante interessante, sendo alvo de estudos da comunidade acadêmica do mundo todo, inclusive com a formação de grupos de especialistas com o objetivo de aprimorar os modelos temporais existentes, e estudar novos conceitos em temporalidade.

O modelo orientado a objetos estudado, TF-ORM, possui vários aspectos temporais imcorporados, mas a sua implementação completa torna-se difícil, pois não existem muitos trabalhos publicados referentes a este tema. A abordagem com o SGBD OR *Oracle* 8*i* é um trabalho inédito, pois os trabalhos consultados abordam os SGBDs relacionais.

7. BIBLIOGRAFIA

[BAR1999] BARAMARCHI, Luciano André. **Estudo sobre os principais modelos de Banco de Dados e os Sistemas Gerenciadores para estes modelos**. Itajaí, 1999. Monografia (Graduação em Ciência da Computação) – Centro de Educação Superior de Ciências Tecnológicas da Terra e do Mar, Universidade do vale do Itajaí.

[BUS1999] BUSARELLO, Elton Carlos. **Desenvolvimento de Sistemas Orientados a Objetos**. Itajaí, 1999. Monografia (Graduação em Ciência da Computação) – Centro de Educação Superior de Ciências Tecnológicas da Terra e do Mar, Universidade do vale do Itajaí.

[CAR1997] CARVALHO, Tanisi Pereira de. **Implementação de Consultas para um Modelo de Dados Temporal Orientado a Objetos**. Porto Alegre, 1997. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul.

[CAV1995] CAVALCANTI, João Marcos Bastos. **Implementação de Banco de Dados Temporais usando SGBDs Relacionais.** Belo Horizonte, 1995. Dissertação (Mestrado em Ciência da Computação) – Instituto de Ciências Exatas, Universidade Federal de Minas Gerais.

[EDE1994] EDELWEISS, Nina Krahe; OLIVEIRA, José Palazzo Moreira de. **Modelagem de aspectos temporais de sistemas de informação.** IX ESCOLA DE COMPUTAÇÃO. Recife. 1994.

[EDE1999] _____. "O modelo TF-ORM". [http://www.inf.ufrgs.br/~nina/indice.htm]. (23 ago. 1999 13:30)

[FAN2000] FANDERUFF, Damaris. **Oracle 8*i* – Utilizando SQL*PLUS e PL/SQL.** São Paulo: Makron Books, 2000.

[KOC1997] KOCH, Henry F.; SILBERSCHATZ, Abrajam. **Sistemas de Banco de Dados.** São Paulo : Makron Books, 1992.

[MCC1998] MCCULLOUGH-DIETER, Carol. **Oracle8 - Bible**. Foster City : IDG Books WorldWide Inc, 1998.

[SAL1999] SALGADO, Ana Carolina. **In Simpósio Brasileiro de Banco de Dados, SGBD Objeto-Relacional.** Florianópolis. 1999.

[TON2000] TONIAL, Fernando. **Implementação de um banco de dados temporal utilizando o modelo orientado a objeto TF-ORM**. Itajaí, 2000. Monografia (Graduação em Ciência da Computação) – Centro de Educação Superior de Ciências Tecnológicas da Terra e do Mar, Universidade do vale do Itajaí.

[WIN1993] WINBLAD, Ann L.; EDWARDS, Samuel D.; KING, David R.. **Software orientado ao objeto**, São Paulo: Makron Books, 1993.