UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

FERRAMENTA DE APOIO AO MAPEAMENTO DE MODELO TEMPORAL DE DADOS PARA UM SGBD RELACIONAL

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

MARCOS LUIS KRETZCHMER

BLUMENAU, DEZEMBRO/2000

2000/2-38

FERRAMENTA DE APOIO AO MAPEAMENTO DE MODELO TEMPORAL DE DADOS PARA UM SGBD RELACIONAL

MARCOS LUIS KRETZCHMER

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Everaldo Artur Grahl — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Everaldo Artur Grahl

_____

Prof. Luiz Bianchi

_____

Prof. Marcel Hugo

DEDICATÓRIA

Dedico este trabalho a minha esposa pela sua compreensão.

AGRADECIMENTOS

Gostaria de agradecer aos professores do curso de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau, pelo conhecimento e experiência adquiridos no decorrer do curso. Agradeço especialmente ao professor Everaldo Artur Grahl, pela orientação e incentivo na condução deste trabalho.

Gostaria também de agradecer a todos que de forma direta ou indireta ajudaram na confecção deste trabalho.

SUMÁRIO

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

MER	ModeloEntidade Relacionamento
SGBD	Sistema Gerenciador de Banco de Dados

RESUMO

O presente trabalho traz um estudo sobre banco de dados temporal e apresenta um protótipo de uma ferramenta que auxilia na criação de base de dados que incorporam aspectos temporais através do mapeamento para o gerenciador de banco de dados relacional ORACLE.

ABSTRACT

The present work brings a study on secular data base and presents an prototype of a tool that assists in the creation data-base that they incorporate temporal aspects through the mapping for the gerenciador of relationary data base ORACLE.

1 INTRODUÇÃO

Este capítulo trata da origem, objetivos e da organização do trabalho.

1.1 ORIGEM

A maior parte das aplicações atuais tem a necessidade de manipular, de alguma maneira, informações históricas – dados relativos a estados passados da aplicação. Os SGBDs (Sistemas gerenciadores de banco de dados) convencionais, no entanto, não proporcionam suporte a estas informações. A necessidade de suprir esta lacuna fez com que nos últimos 20 anos muitas pesquisas tenham sido realizadas na área de Banco de Dados Temporais, com o objetivo de definir conceitos e estratégias para tratar de informações históricas [EDE1998].

Os modelos de dados tradicionais apresentam duas dimensões, representando (1) as instâncias dos dados (linhas de uma tabela) e (2) os atributos de cada instância (colunas desta tabela). Cada atributo de uma instância apresenta um só valor. Se for feita uma alteração deste valor, o anterior é perdido [EDE1998].

Os modelos temporais acrescentam mais uma dimensão aos modelos tradicionais – a dimensão temporal. Esta dimensão associa alguma informação temporal a cada valor. Caso o valor do atributo seja alterado, o valor anterior não é removido do banco de dados – o novo valor é acrescentado, associado a alguma informação que define a sua validade. Todos os valores definidos ficam armazenados no banco de dados. Deste modo é possível acessar toda a história dos atributos, sendo possível analisar sua evolução temporal [EDE1998].

Com o objetivo de dotar o modelo ER (Entidade Relacionamento) desta capacidade, algumas extensões temporais têm sido propostas, entre as quais : a abordagem ERT (*Entity Relationship Time Model*) [LOU1991], a abordagem TER (*Temporal Entity-Relationship Model*) [TAU1991], e a abordagem TEER (*Temporal enhanced Entity-Relationship Model*) [ELM1992],[ANT1999].

A utilização de um modelo de dados temporal para especificação de uma aplicação não implica necessariamente, na utilização de SGBD específico para o modelo. Banco 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 temporal pode ser implementado sobre um banco de dados relacional, orientado a objetos, objeto-relacional e outros. Em cada um deles

devem ser preservadas as características individuais, bem como, as regras que regem o BD [HUB1999].

A implementação de banco de dados temporais em SGBD convencionais deve-se à inexistência de um SGBD totalmente temporal. Algumas implementações de banco de dados temporal em SGBD relacionais e orientados a objetos já foram realizadas. Estas implementações geralmente estão baseadas em modelos já publicados na literatura. As implementações possuem características distintas e buscam armazenar as informações em seus diferentes estados: passado, presente e, em alguns casos, futuro [HUB1999].

1.2 OBJETIVOS

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta que auxilie o usuário na criação de uma base de dados temporal através do mapeamento para um sistema gerenciador de banco de dados relacional.

1.3 ORAGANIZAÇÃO DO TEXTO

Este trabalho está dividido em 5 capítulos. No primeiro capítulo é apresentado a introdução, os objetivos e a organização do trabalho.

No segundo capítulo é apresentado o que é um banco temporal, com as suas características, os seus elementos e o que difere dos modelos tradicionais.

No terceiro capítulo é apresentado o modelo temporal de dados TempER com as suas principais características.

No quarto capítulo é apresentado a especificação do protótipo, alguns detalhes sobre a entrada dos dados.

No quinto capítulo é apresentado o funcionamento do protótipo com um pequeno estudo de caso.

O sexto capitulo apresenta as conclusões finais do protótipo, as limitações, dificuldades encontradas e as sugestões que poderão contribuir para a futura continuação deste trabalho.

2 BANCO DE DADOS TEMPORAL

Este capítulo tem como objetivo dar uma visão geral sobre banco de dados temporal, sobre dimensão temporal, sobre ordem de tempo, seus tipos, seus elementos e formas de consulta.

2.1 INTRODUÇÃO

O tempo é um aspecto importante de todos os fenômenos do mundo real. Eventos ocorrem em pontos específicos de tempo; objetos e seus relacionamentos têm existência dependente de dimensões temporais; porém banco de dados convencionais (não temporais) representam o estado do mundo em um único momento do tempo. Embora o conteúdo do banco de dados continue a mudar porque novas informações são atualizadas, estas mudanças são modificações de estado: dados velhos, desatualizados, são apagados do banco de dados. Este conteúdo, corrente, pode ser visto como uma imagem instantânea (*snapshot*). Em tais sistemas, eventuais atributos envolvendo tempo são manipulados unicamente por programas de aplicação e o sistema de gerenciamento de banco de dados interpreta datas como valores de tipos de dados básicos[TON2000].

Dados em um banco de dados não temporal são temporariamente inconsistentes porque se tornam correntes em pontos de tempo diferentes e desconhecidos. Em contraste, um banco de dados temporal modela o mundo na sua dinâmica, rastreando pontos de mudança e retendo todos os dados.

Várias aplicações podem se beneficiar do suporte a tempo em SGBD: sistemas financeiros, planos de tratamento médico, monitoração ecológica, gerenciamento de dados de vídeo, etc. Estas aplicações requerem suporte a dados incompletos e consultas temporais muito complexas [TON2000].

2.2 DIMENSÃO TEMPORAL

Os modelos de dados tradicionais apresentam duas dimensões representado (1) as instâncias dos dados (linhas da tabela), e (2) os atributos de cada instância (colunas desta tabela). Cada atributo de uma instância apresenta um só valor. Se for feita uma alteração deste

valor, o anterior é perdido. Por exemplo, se o atributo representa o salário de um funcionário, o banco de dados somente armazena o ultimo valor.([EDE1998]).

Os modelos temporais acrescentam mais uma dimensão aos modelos tradicionais - a dimensão temporal. Esta dimensão associa alguma informação temporal a cada valor. Caso o valor de um atributo seja alterado, o valor anterior não é removido do banco de dados - o novo valor é acrescentado, associado a alguma informação que o define, por exemplo, seu tempo inicial de validade. Todos os valores definidos ficam armazenados no banco de dados. No exemplo anterior, todos os valores do salário do funcionário ficam armazenados no banco de dados. Deste modo é possível acessar toda da história dos atributos, sendo possível analisar sua evolução temporal [EDE1998].

2.3 ORDEM NO TEMPO

Segundo [EDE1998] a dimensão temporal é composta por uma seqüência de pontos consecutivos no tempo, que recebe o nome de eixo temporal. A definição de uma ordem a ser seguida no tempo é fundamental quando utilizada alguma representação temporal. O mais comum é que se assuma que o tempo flui linearmente; isto implica em uma total ordenação entre quaisquer dois pontos no tempo. Em alguns casos pode ser considerado tempo ramificado ("*branching time*"). Para estes a restrição linear é abandonada permitindo a possibilidade de dois pontos diferentes serem sucessores (ramificação no futuro) ou antecessores (ramificação no passado) imediatos de um mesmo ponto. Uma ramificação no futuro implica que podem ser considerados múltiplos possíveis desenvolvimentos futuros do domínio em questão. A combinação "passado linear, futuro ramificado" trabalha com uma só história passada e admite múltiplas histórias futuras, representando desta maneira a realidade atual de uma forma bastante fiel. Uma última opção de ordenação temporal é considerar o tempo circular. Esta forma pode ser utilizada para modelar eventos e processos recorrentes.

2.4 TEMPO ABSOLUTO E TEMPO RELATIVO

Outro conceito importante é o que diferencia tempo absoluto de relativo. Tempo absoluto consiste de uma informação temporal que define um tempo específico, definido com uma granularidade determinada, associado a um fato. Exemplo: Flávio nasceu no dia 30/08/73.

Um tempo é relativo quando sua validade está relacionada à validade de um outro fato, ou ao momento atual. Exemplo: o salário aumentou ontem; a loja abriu dois meses depois da abertura do Shopping[EDE1998].

2.5 VARIAÇÃO TEMPORAL

Duas formas basicamente diferentes de variação temporal podem ser consideradas: tempo contínuo e tempo discreto. Supõe-se que o tempo é contínuo por natureza. Entretanto, sem grande perda de generalidade, o tempo pode ser considerado como discreto. Esta segunda forma de representação simplifica consideravelmente a implementação de modelos de dados.

Modelos de dados que suportam uma noção discreta de variação temporal são baseados em uma linha de tempo composta de uma seqüência de intervalos temporais consecutivos, que não podem ser decompostos, de idêntica duração. Estes intervalos são denominados *chronons*. A duração particular de um *chronon* não é necessariamente fixada no modelo de dados, podem ser definida em implementações particulares do modelo de dados.

Considerando variação temporal discreta, a definição de informações ao longo do tempo, sob o ponto de vista de sua validade, pode ser feita das seguintes formas:

a. Variação ponto a ponto – o valor definido vale somente no ponto temporal onde foi definido. Não existe valor válido nos pontos para os quais não foram definidos valores;

b. Variação por escala – o valor fica constante desde o ponto em que foi definido até o instante em que outro valor seja definido. Corresponde, geralmente, à definição de valores em conseqüência da ocorrência de eventos(variação por eventos);

c. Variação temporal definida por uma função – existe uma função que define os valores e que permite a interpolação para obter os valores nos pontos não definidos. Esta função de interpolação pode ser definida pelo usuário ou incluída na modelagem conceitual.

2.6 ELEMENTOS PRIMITIVOS DA REPRESENTAÇÃO TEMPORAL

2.6.1 INSTANTE DE TEMPO

Segundo [EDE1998] o conceito de instante de tempo é representado por um ponto particular no tempo, dependendo da forma de variação considerada. Os tipos de instante de tempo são:

- Tempo Contínuo - É considerado tempo contínuo, um instante de um ponto de duração infinitesimal. Neste caso os instantes de tempo são isomórficos com os números reais, o que significa que entre dois pontos sempre existe um outro ponto no tempo.

- Variação discreta - É considerado variação discreta, um instante que é representando por um dos *chronons* da linha de tempo suportada pelo modelo. Na variação discreta, os instantes são isomórficos aos números inteiros ou ao subconjunto destes. Assim, entre dois pontos no tempo consecutivos, não existe outro ponto no tempo. Diz-se que um evento ocorre no tempo t se ocorrer em qualquer tempo durante o *chronon* representado por "t". Um *chronon*, que é a menor duração de tempo suportada por um SGBD temporal, pertence à representação discreta do tempo.

- Variação temporal linear - Se considerarmos a ordem de variação temporal linear, temos a existência de um instante especial, correspondente ao instante atual (*now*), o qual se move constantemente ao longo do eixo do tempo. Este ponto define o que é considerado como passado e como futuro.

2.6.2 INTERVALO TEMPORAL

Segundo [EDE1998] um intervalo temporal é caracterizado pelo tempo decorrido entre dois instantes – um subconjunto de pontos do eixo temporal. Existem 3 tipos de intervalo temporal que são:

- Aberto – Os limites não pertencem ao intervalo temporal

- Semi-aberto – Um dos limites não pertence ao intervalo de tempo

- Fechado – Ambos os limites pertencem ao intervalo temporal.

2.6.3 ELEMENTO TEMPORAL

Conforme [EDE1998], elemento temporal é uma união finita de intervalos de tempo. Este intervalos podem ser disjuntos, o que realmente o diferencia dos demais e enriquece o seu poder de expressão. O elemento temporal é fechado para as operações de união, interseção e complemento da teoria dos conjuntos, isto é, qualquer destas operações sobre o elemento temporal produz um novo elemento temporal. Como estas operações encontram contrapartida nos operadores booleanos *or*, *and* e *not*, isto produz uma substancial simplificação na habilidade do usuário de expressar consultas temporais.

Segundo [EDE1998], em termos de modelagem, o elemento temporal se mostra superior ao uso da primitiva intervalo de tempo pois, quando os intervalos são usados como rótulos temporais, os objetos são fragmentados em várias tuplas, uma para cada intervalo.

2.7 TIPOS DE BANCO DE DADOS TEMPORAL

Segundo [EDE1998], banco de dados temporal é aquele que, de alguma forma, representa informações temporais. Os bancos 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;

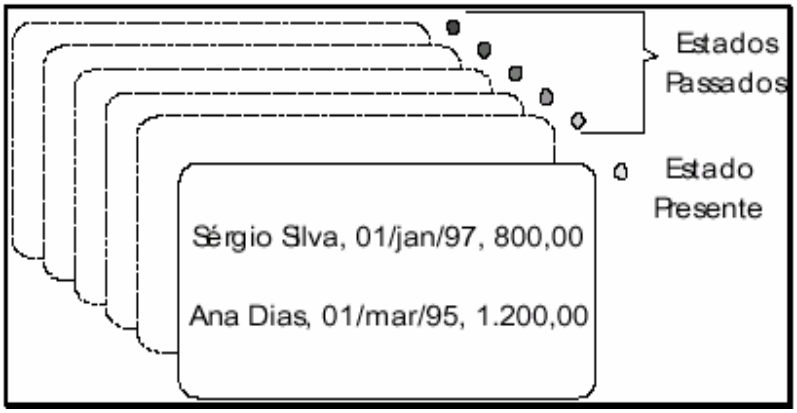
- Banco de dados bitemporais.

2.7.1 BANCO DE DADOS INSTANTÂNEOS

Conforme [CAV1995], banco de dados instantâneos, são banco 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" *(snapsho*t) 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 INSTANTÂNEOS



FONTE: [TON2000]

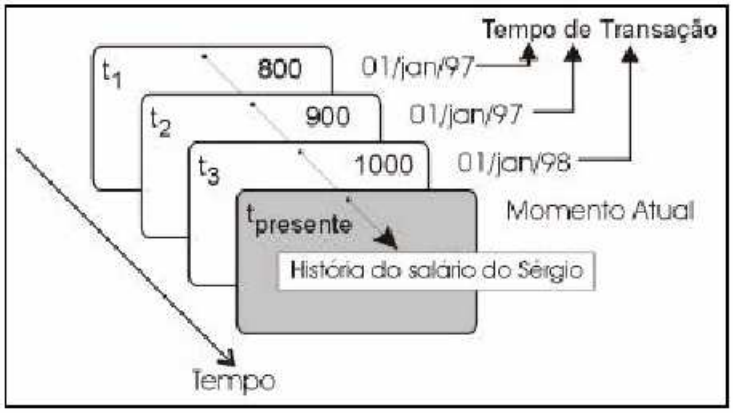
2.7.2 BANCO DE DADOS DE TEMPO DE TRANSAÇÃO

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

De acordo com [EDE1998], as informações temporais são associadas a cada valor definido, o instante temporal em que foi realizada a transação, sob forma de um rótulo temporal *(timestamp*). 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.

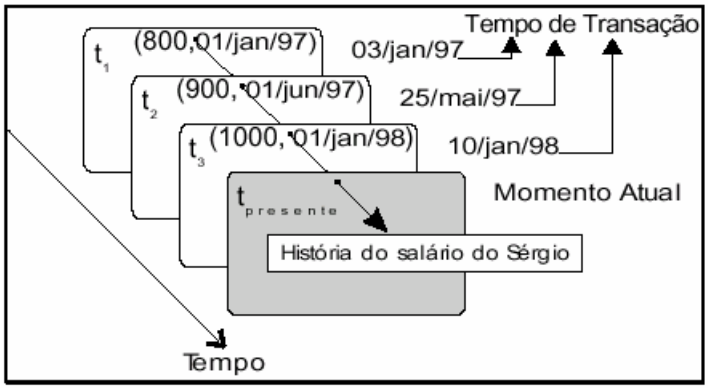
FIGURA 2 - BANCO DE DADOS DE TEMPO DE TRANSAÇÃO



FONTE: [TON2000]

Ainda para [EDE1998], 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 COM ATRIBUTO EXPLÍCITO



FONTE: [TON2000]

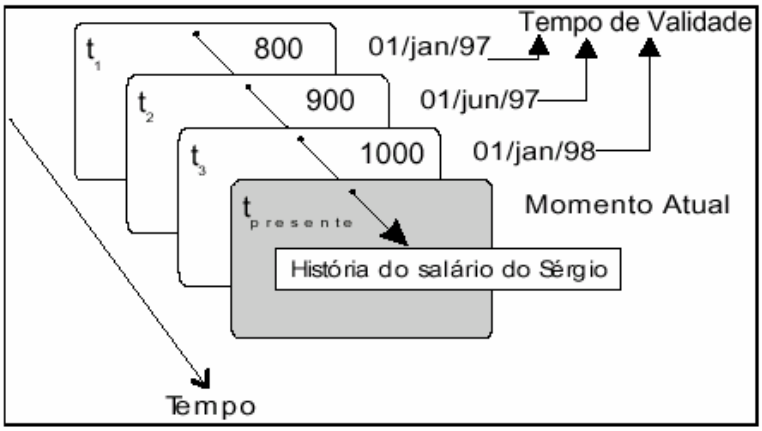
2.7.3 BANCO DE DADOS DE TEMPO DE VALIDADE

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 validade 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 [EDE1998], 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 VALIDADE



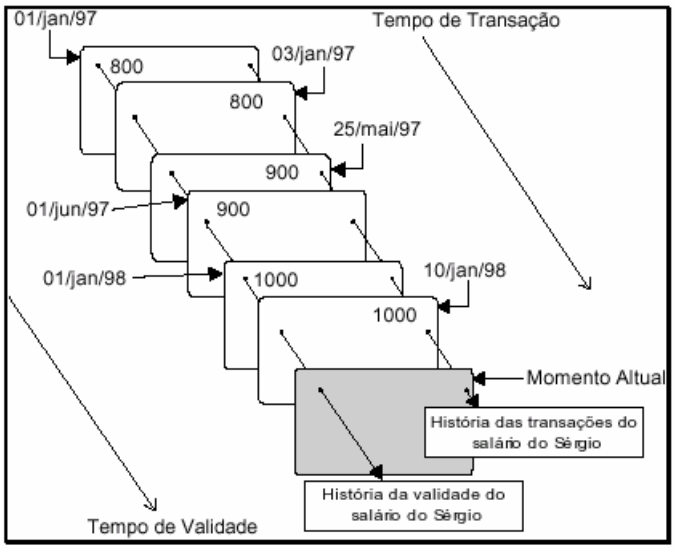
FONTE: [TON2000]

2.7.4 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 [EDE1998], 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 BITEMPORAIS



FONTE: [TON2000]

2.8 CONSULTA A BANCO DE DADOS TEMPORAIS

Segundo [EDE1998], quando é utilizado um banco de dados temporal, é importante também que esteja disponível uma linguagem de consulta temporal. Esta linguagem deve possibilitar a recuperação de todas as informações armazenadas no banco de dados (temporal ou não), de modo a que seja tirado real proveito do acréscimo da dimensão temporal. As consultas temporais permitem:

- Fornecer valores de propriedades cujo valor é temporal;
- Referir-se a um determinado instante ou intervalo temporal;

- Recuperar valores com base em restrições temporais;

- Fornecer informações temporais (datas, intervalos).

3 MODELOS DE DADOS TEMPORAIS

Este capítulo tem como objetivo dar uma visão geral sobre o modelo de dados existentes e em especifico o modelo TempER com as suas características , o identificador OID, as perspectivas temporal e intemporal, os componentes e o mapeamento para o SGBD relacional.

O modelo ERT (*Entity-Relationship-Time*) oferece uma série de dispositivos que permite modelar aplicações complexas de banco de dados. Ele possibilita a modelagem explícita do tempo e uma taxinomia para gerar hierarquias e objetos complexos.

O modelo TER (*Temporal Entity-Relationship*) possui como característica principal o desdobramento da noção de cardinalidade em *snapshot* e *lifetime*. Diferente do modelo ERT, o modelo TER não diferencia, ao nivel de notação gráfica, objetos temporais de objetos não temporais.

O modelo TF-ORM é um modelo orientado à objetos que permite a modelagem de aspectos estáticos e dinâmicos e a sua representação é através de papeis.

Já o modelo TempER é um modelo Entidade-relacionamento que contempla os aspectos temporais e que pode ser facilmente modelado em uma ferramenta CASE com pequenas adaptações.

3.1 MODELOS DE DADOS TEMPORAL TEMPER

Antes de iniciar a descrição das características do modelo TempER é importante ter em mente o que significa existência – ou validade temporal – no âmbito do presente trabalho. Existência de uma entidade nada mais é que o conjunto de pontos do tempo em que esta entidade é considerada como válida para efeito de interferências temporais no banco de dados. Não se deve confundir existência com a presença física da entidade no banco de dados, ou seja, uma entidade pode estar presente no banco de dados, e em relação a um determinado ponto do tempo “não existir”.

O modelo TempER é um modelo de dados tipo Entidade-Relacionamento que incorpora dispositivos que permitem referenciar os objetos (entidades, relacionamentos ou valores de atributos) à dimensão temporal.

No modelo TempER, a dimensão temporal é assumida como sendo um eixo de pontos discretos, isomórfico ao conjunto dos números inteiros, como já foi definido anteriormente.

Assim como o modelo ER convencional, o modelo TempER apresenta também os seguintes elementos básicos: entidade, relacionamento e atributos.

Um grupo de entidades da mesma natureza e com a mesma estrutura de atributos, é denominado de conjunto-entidade e é este conjunto que se simboliza graficamente no diagrama. Um exemplo é o conjunto-entidade Empregado da figura 6, que se representa todos os empregados de uma certa empresa. Da mesma forma, um grupo de relacionamentos com as mesmas características (mesmos conjuntos-entidade associados e mesma finalidade) é denominado de conjunto-relacionamento. Por exemplo, todas as lotações existentes de empregados em departamentos são simbolizadas pelo conjunto-relacionamento Lotação da figura 6. Em resumo, entidade é uma instância de um conjunto relacionamento[ANT1999].

Os atributos são propriedades das entidades e relacionamentos. A associação de um atributo com um valor, do domínio de valores deste atributo, é chamada, no contexto deste trabalho, de valoração de um atributo ou atribuição de valor a uma propriedade. Os domínios são conjuntos predeterminados e constantes de valores primitivos.

Quando um atributo de uma entidade é especificado como sendo temporal, está se referindo apenas à valoração deste atributo, ou seja ao seu conteúdo.

Aos objetos de uma aplicação que forem especificados como temporalizados, sejam eles entidades, relacionamentos ou valoração de atributos, são implicitamente anexados rótulos temporais (*timestamps*), que conterão o conjunto de pontos do tempo nos quais este objetos são considerados como existentes no contexto da realidade modelada. O rótulo temporal utilizado no modelo TempER tem o formato do elemento temporal explicado anteriormente.

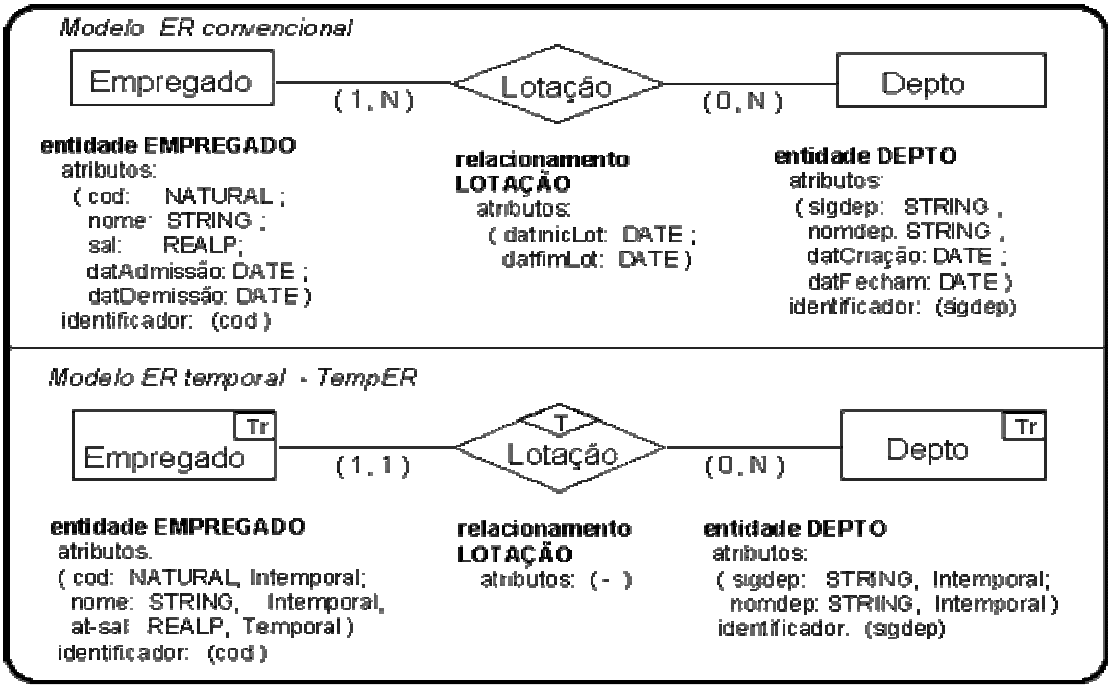
Para ilustrar os conceitos acima descritos, tem-se como exemplo um modelo de dados contendo dois conjuntos –entidade, Empregado e Depto, e mais um conjunto-relacionamento, denominado Lotação, associando estes dois conjuntos-entidade. Suponha-se que esta associação deva obedecer à seguinte restrição: um empregado obrigatoriamente deve estar lotado a um departamento em cada momento da sua existência como empregado, não podendo estar lotado em mais de um departamento ao mesmo tempo. Um outro requisito seria a necessidade de representar as possíveis lotações que um empregado pode apresentar ao longo do tempo, em função das suas transferências de um departamento para outro.

A figura 6 apresenta o caso acima descrito, modelado em ER convencional e em TempER. No ER convencional não é possível especificar a restrição que determina que um empregado não pode estar lotado em mais de um departamento em cada momento do tempo. A cardinalidade "(1,N)", que aparece na ligação Empregado e Lotação, especifica que um empregado deve estar associado a no mínimo um departamento, podendo estar associado a mais de um (adota-se a cardinalidade que determina o grau de participação mínima e máxima de uma entidade em um conjunto de relacionamentos).

Em virtude do modelo ER convencional não dispor de primitivas de modelagem para representar a associação dos objetos como a datAdmissão, datDemissão, datinicLot, DatCriação, datFecham encontrados na figura 6, isto implica em transferir para modelagem dinâmica a responsabilidade de tratar as restrições temporais, ou seja, no caso do exemplo em questão, são as transações dos sistema que devem se preocupar em impedir que um empregado possa estar lotado em dois ou mais departamentos ao mesmo tempo.

Este problema deixa de ocorrer quando se utiliza o modelo TempER. Como pode ser visto no diagrama situado na parte inferior da figura 6, a cardinalidade que aparece na ligação entre Empregado e Lotação agora é (1,1), a qual tem o seguinte significado: um empregado participa do conjunto-relacionamento Lotação no mínimo uma vez e no máximo uma vez em cada momento do tempo. Além disto, os atributos referindo-se a pontos do tempo que estavam no diagrama ER convencional deixam de ser necessários, pois são substituídos por rótulos temporais implícitos.

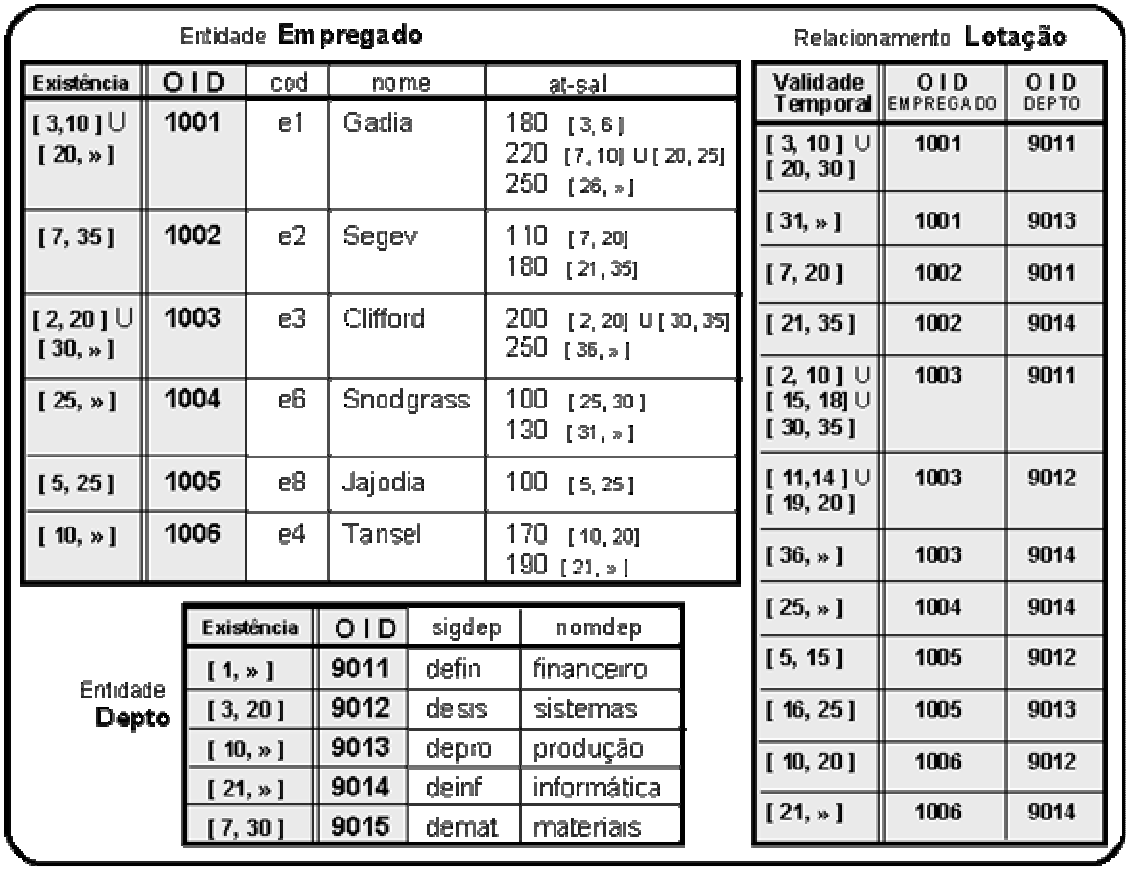
FIGURA 6 - COMPARAÇÃO ENTRE MODELAGEM ER CONVENCIONAL E TEMPER



FONTE: [ANT1999]

Uma outra vantagem em relação ao modelo ER convencional diz respeito ao conceito de reencarnação de objetos. Um empregado que foi demitido pode ser readmitido; um empregado que tenha sido alocado a um determinado departamento pode vir a ser realocado a este mesmo departamento em um outro período do tempo. O modelo TempER representa estas situações de forma natural, pois admite que a existência de um objeto possa ser segmentada em intervalos de tempo, como é o caso do empregado 1001 e do relacionamento < 1003,9011 > encontrados na figura 7.

FIGURA 7 - EXEMPLO DE POVOAMENTO DE ENTIDADES E RELACIONAMENTOS EM TEMPER



Entidade **Empregado**

Existência	O I D	cod	nome	at-sal
[3,10] ∪ [20, »]	1001	e1	Gadia	180 [3, 6] 220 [7, 10] ∪ [20, 25] 250 [26, »]
[7, 35]	1002	e2	Segev	110 [7, 20] 180 [21, 35]
[2, 20] ∪ [30, »]	1003	e3	Clifford	200 [2, 20] ∪ [30, 35] 250 [36, »]
[25, »]	1004	e6	Snodgrass	100 [25, 30] 130 [31, »]
[5, 25]	1005	e8	Jajodia	100 [5, 25]
[10, »]	1006	e4	Tansel	170 [10, 20] 190 [21, »]

Entidade **Depto**

Existência	O I D	sigdep	nomdep
[1, »]	9011	defin	financeiro
[3, 20]	9012	desis	sistemas
[10, »]	9013	depro	produção
[21, »]	9014	deinf	informática
[7, 30]	9015	demat	materiais

Relacionamento **Lotação**

Validade Temporal	O I D EMPREGADO	O I D DEPTO
[3, 10] ∪ [20, 30]	1001	9011
[31, »]	1001	9013
[7, 20]	1002	9011
[21, 35]	1002	9014
[2, 10] ∪ [15, 18] ∪ [30, 35]	1003	9011
[11,14] ∪ [19, 20]	1003	9012
[36, »]	1003	9014
[25, »]	1004	9014
[5, 15]	1005	9012
[16, 25]	1005	9013
[10, 20]	1006	9012
[21, »]	1006	9014

FONTE: [ANT1999]

3.2 O IDENTIFICADOR INTERNO DE ENTIDADES - OID

Segundo [ANT1999], é assumido que todas as instâncias dos conjuntos-entidade, e apenas dos conjuntos-entidade, possuem um identificador interno, gerado pelo sistema, que será denominado de OID (*object identifier*). Cada OID, por princípio, é único no âmbito do universo do discurso da aplicação, é invisível ao usuário e define a identidade de uma entidade. Quando duas entidades se associam, este fato pode ser representado pelo relacionamento dos OID respectivos. Isto faz com que um relacionamento seja identificável pela composição dos OID das entidades associadas.

A presença do OID não descarta a necessidade de que haja um atributo (ou composição de atributos) que desempenhe o papel de chave primária de uma entidade, de

forma que um usuário do sistema possa identificar e acessar esta entidade, já que o OID é invisível externamente. Embora possa parecer redundante coexistir OID e chaves primárias, o que se busca é aproximar o modelo TempER de um dos mais importantes princípios da orientação a objetos, o da identidade dos objetos. Além disso, existe a vantagem de ser possível alterar a chave primária de uma entidade sem que isto afete os relacionamentos em que ela participe.

A título de exemplo pode-se examinar a figura 7, onde tanto a tabela Empregado como Depto, possuem uma coluna referente aos OID, e a tabela referente ao conjunto-relacionamento Lotação possui duas colunas de OID referentes aos conjunto-entidades relacionados.

3.3 AS PERSPECTIVAS TEMPORAL E INTEMPORAL DAS ENTIDADES

As entidades são elementos básicos de um modelo de dados que emprega a abordagem Entidade-Relacionamento. No modelo TempER, em virtude da dimensão temporal, as entidades apresentam sempre duas perspectivas, uma perspectiva temporal e uma perspectiva intemporal, como se fossem as duas faces de uma mesma moeda.

3.3.1 PERSPECTIVA INTEMPORAL

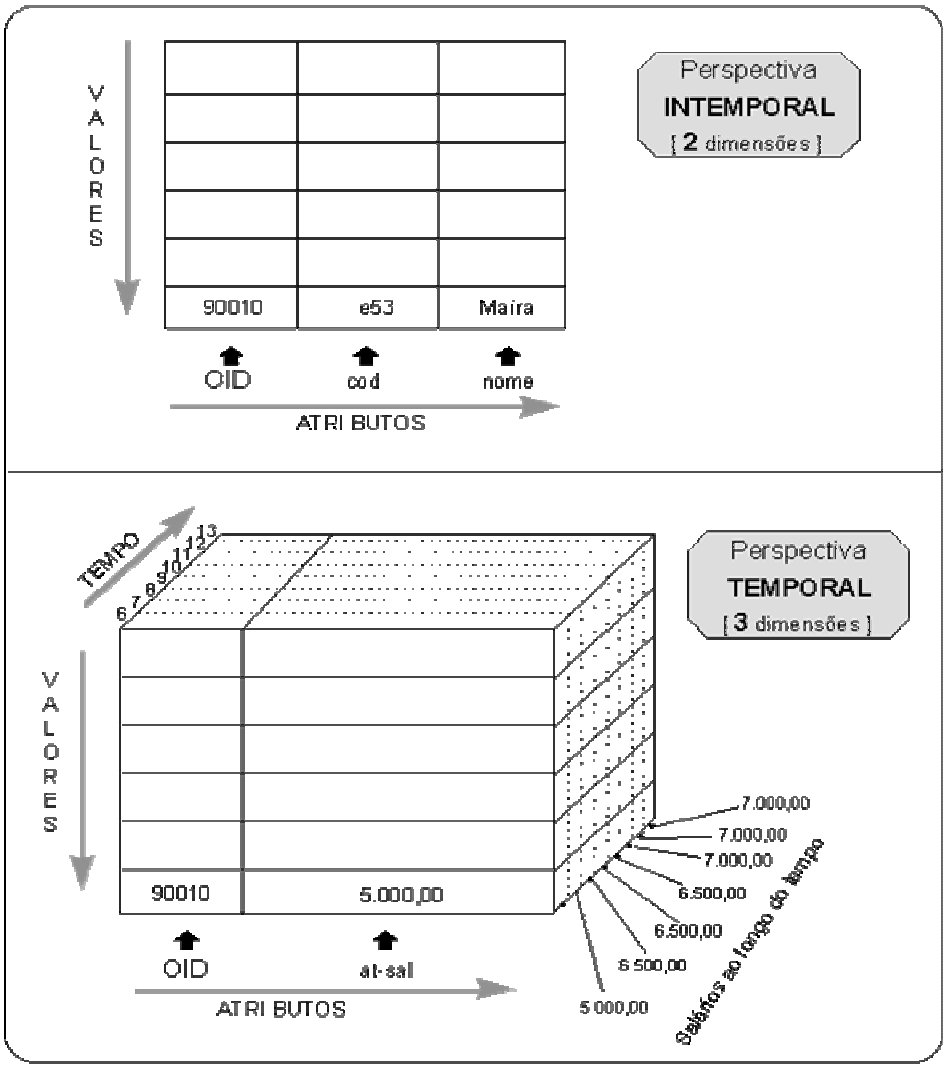
Não leva em consideração a dimensão temporal, isto é, o OID da entidade não é visualizado como associado a um conjunto de pontos do tempo. Nesta perspectiva as entidades apresentam apenas duas dimensões: a dos atributos e a dos valores. Como mostrado na figura 8.

3.3.2 PERSPECTIVA TEMPORAL

Leva em conta a dimensão temporal ao referenciar um dado. Nesta perspectiva, o OID de uma entidade é visualizado como um estado associado a um conjunto de pontos do tempo, conjunto este que define a validade temporal (ou existência) da entidade no contexto do banco de dados do sistema. As entidades, vistas pela perspectiva temporal, apresentam três dimensões: a dos atributos, a dos valores e a do tempo, como mostra a figura 8.

Na figura 8, o mesmo conjunto de entidades (Empregado) apresenta simultaneamente as duas perspectivas. Na perspectiva intemporal, as entidades do conjunto apresentam os atributos código e nome que por conseqüência, são do tipo intemporal. Por outro lado, na perspectiva temporal, as entidades apresentam o atributo at-sal (salário), que é um atributo do tipo temporal.

FIGURA 8 - AS DUAS PERSPECTIVAS EM RELAÇÃO AO TEMPO DO CONJUNTO-ENTIDADE EMPREGADO



FONTE: [ANT1999]

3.4 COMPONENTES DO MODELO DE DADOS TEMPER

A notação dos elementos gráficos do modelo TempER mantém os símbolos originais propostos por [CHE1976], adicionando-lhes apenas os sinais necessários para determinar qual é o tipo de relação destes elementos com a dimensão temporal. Por tanto, a representação de um conjunto-entidade continua sendo um retângulo e a representação de conjunto-relacionamento continua sendo um losango.

3.4.1 ENTIDADES

As entidades no modelo TempER são classificadas em transitórias e perenes, conforme a sua relação com a dimensão temporal.

3.4.1.1 ENTIDADES TRANSITÓRIAS

Entidades transitórias são aquelas cuja validade temporal é um subconjunto de pontos do tempo do eixo temporal. Normalmente lança-se mão deste tipo de entidade quando se quer modelar entidades que valem por um certo período de tempo.

O símbolo para denotar entidades transitórias é um retângulo contendo a partícula “Tr” no canto superior direito.

FIGURA 9 - ENTIDADE TRANSITÓRIA

FONTE: [ANT1999]

3.4.1.2 ENTIDADES PERENES

São aquelas cuja validade temporal é exatamente igual a todo o eixo temporal. Toda a vez que uma entidade perene é incluída no banco de dados do sistema, assume-se que seu rotulo temporal é igual a “[T,t’]”, isto é a sua validade temporal inicia no primeiro ponto do

eixo temporal e se estende até o ultimo. Normalmente as entidades que o modelador não necessita ou não deseja associar ao tempo são consideradas como perenes.

O símbolo para denotar entidades perenes é um retângulo contendo a partícula “Pe” no canto superior direito.

FIGURA 10 - ENTIDADE PERENE

FONTE: [ANT1999]

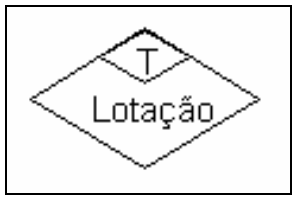
3.4.2 RELACIONAMENTOS

Relacionamentos são objetos resultantes da associação de duas ou mais entidades. No modelo TempER as entidades podem estar associadas entre si, ou na perspectiva temporal, ou na perspectiva intemporal.

3.4.2.1 RELACIONAMENTOS TEMPORAIS

Segundo [ANT1999], são os relacionamentos que associam duas entidades no âmbito da dimensão temporal, isto é, além dos OID das entidades, estes relacionamentos também se associam a pontos do tempo do eixo temporal. Este tipo de relacionamento serve para modelar as associações das quais se necessita conhecer a validade temporal. O símbolo utilizado para representar o relacionamento temporal é um losango contendo no canto superior um pequeno losango com a letra T como demonstrado na figura 11.

FIGURA 11 - RELACIONAMENTO TEMPORAL

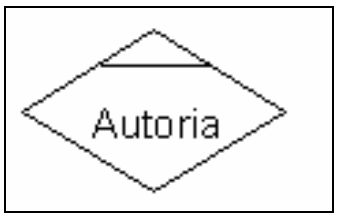


FONTE: [ANT1999]

3.4.2.2 RELACIONAMENTOS INTEMPORAIS

Segundo [ANT1999], são relacionamentos que não levam em consideração a dimensão temporal, ou seja neste tipo de relacionamento apenas participam os OID das entidades, sem qualquer menção a pontos no tempo. Os relacionamentos intemporais se materializam ao nível das perspectivas intemporais das entidades. O símbolo utilizado para representar o relacionamento intemporal é um losango contendo um pequeno risco horizontal no canto superior como demonstrado na figura 12.

Figura 12 - Relacionamento Intemporal



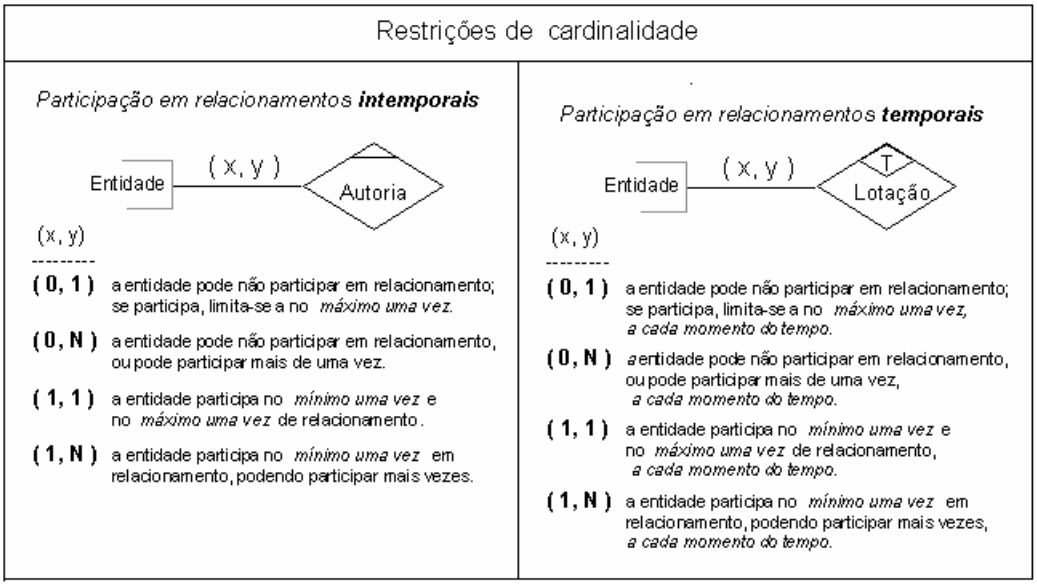
FONTE: [ANT1999]

3.4.2.3 RESTRIÇÕES DE CARDINALIDADE DOS RELACIONAMENTOS

A restrição de cardinalidade junto às linhas que conectam conjuntos-entidades (retângulos) a conjuntos relacionamentos (losangos) refere-se à participação das entidades nos conjuntos-relacionamento. Esta forma de anotar cardinalidades é a mesma adotada na metodologia Merise, segundo [ANT1997]. Segundo este princípio de notação de cardinalidade, uma entidade participa de relacionamentos como demonstrado na figura 13.

O significado das restrições de cardinalidade depende da classificação temporal do conjunto-relacionamento. Caso o conjunto-relacionamento seja temporal, a leitura da cardinalidade, necessariamente dá-se em relação a cada ponto do tempo. Por outro lado, caso o conjunto-relacionamento seja intemporal, deixa de existir esta referência a pontos do tempo e a leitura das restrições de cardinalidade fica semelhante à do ER convencional.

FIGURA 13 - RESTRIÇÕES DE CARDINALIDADE



FONTE: [ANT1999]

3.4.3 ATRIBUTOS

Segundo [ANT1997], tanto as entidades como os relacionamentos apresentam propriedades que os caracterizam, as quais são denominadas de atributos.

Os atributos dos conjuntos-entidade e conjuntos-relacionamento são especificados através do dicionário de dados, que complementa um diagrama TempER. A descrição de um atributo apresenta os seguintes elementos: o nome do atributo, o domínio dos valores primitivos que podem ser associados ao atributo ou a classificação do atributo em relação ao tempo (indicando se o atributo é temporal ou intemporal). Como demonstrado na figura 6.

Conforme [ANT1997], com o objetivo de simplificar os estudos, o modelo TempER não apresenta atributos opcionais e tampouco atributos multivalorados.

Um conjunto-entidade normalmente possui um identificador de uso externo, uma chave primária. Em geral trata-se de um único atributo, no entanto pode resultar da composição de dois ou mais atributos. O valor de um identificador é único dentro do contexto de um conjunto-entidade, isto é, não existem duas instâncias com o mesmo valor de identificador. O identificador deve sempre ser formado por atributo (s) intemporal (is).[ANT1997]

As entidades, sejam elas transitórias ou perenes, por apresentarem duas perspectivas em relação ao tempo, podem combinar atributos temporais com atributos intemporais. Os relacionamentos não. Se forem temporais podem possuir apenas atributos temporais, se forem intemporais podem apresentar apenas atributos intemporais.

3.4.3.1 ATRIBUTOS TEMPORAIS

Segundo [ANT1999], os atributos temporais são aqueles cuja valoração (conteúdo) deve ser referenciada a pontos do tempo.

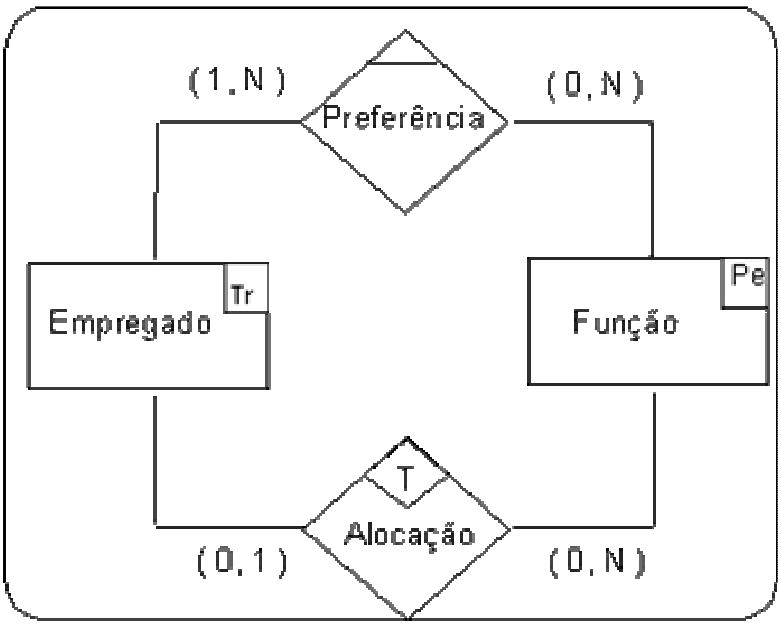
O objetivo dos atributos temporais é permitir que se registre a evolução dos valores de uma propriedade ao longo do tempo.

Em virtude de no modelo TempER não existir atributo opcional, um atributo temporal deve, obrigatoriamente, apresentar um valor para cada momento da existência das entidades ou relacionamentos temporais a que pertença. Quando se tratar de entidade perene, isto implica em que o atributo temporal apresenta valor em todos os pontos do eixo temporal.

3.4.4 MESCLAGEM DE OBJETOS

O modelo TempER também permite que se mescle objetos de diferentes classificações temporais em um mesmo diagrama, como mostra a figura 14, onde um conjunto-entidade do tipo perene (Função) relacione-se com um conjunto-entidade do tipo transitório (Empregado) de duas formas: uma temporariamente (Alocação) e outra intemporarialmente (Preferência).

FIGURA 14 - MESCLANDO OBJETOS DE DIFERENTES CLASSIFICAÇÕES TEMPORAIS



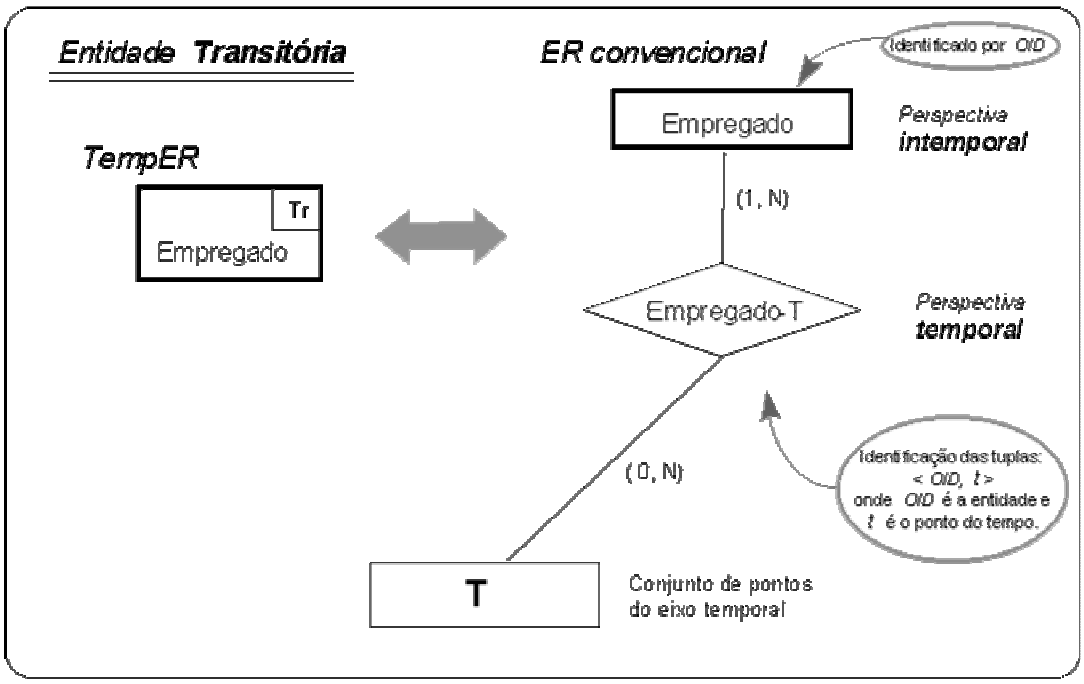
FONTE: [ANT1999]

3.5 MAPEAMENTO DO MODELO TEMPER PARA O BANCO DE DADOS RELACIONAL

3.5.1 ENTIDADES TRANSITÓRIAS

Segundo [ANT1999], o mapeamento das entidades transitórias para o modelo relacional pode ser exemplificado da seguinte maneira. Para mostrar como a entidade transitória Empregado relaciona-se com a dimensão temporal, ela apresenta um conjunto-entidade que contém todos os pontos do eixo temporal: o conjunto T. Cada empregado está associado a no mínimo, um destes pontos do tempo, podendo estar associado a mais de um. O conjunto-relacionamento Empregado-T contém uma instância para cada ponto do tempo da validade temporal de cada empregado que esteja presente no conjunto-entidade Empregado. Este exemplo pode ser visualizado na figura 15.

FIGURA 15 - MAPEAMENTO DA ENTIDADE TRANSITÓRIA PARA O ER CONVENCIONAL



FONTE: [ANT1999]

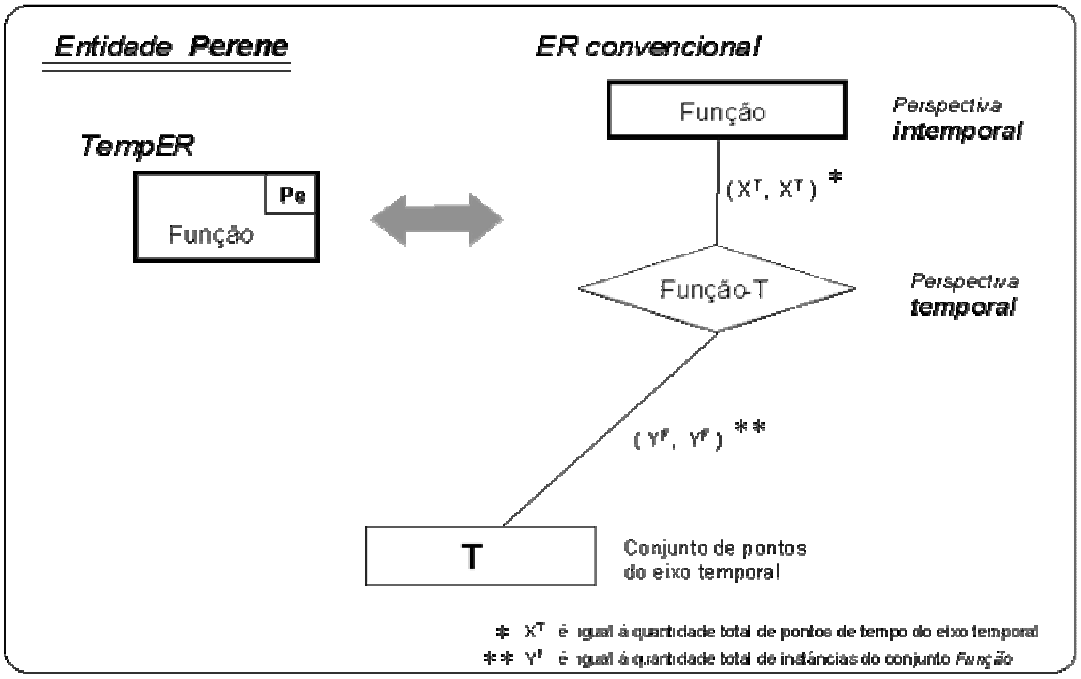
3.5.2 ENTIDADES PERENES

Conforme [ANT1999], o fato de ser perene não significa que uma entidade não possa ser eliminada do banco de dados. Entretanto, enquanto uma entidade perene estiver presente no banco de dados do sistema, a sua validade temporal é constante, igual ao conjunto de todos os pontos do eixo temporal.

Por ser constante, a validade temporal das entidades perenes não precisa ser registrada no banco de dados. O modelador deve visualizar o conjunto de pontos de tempo que define a existência das entidades perenes como sendo implicitamente especificado, ou seja, sempre igual ao conjunto total de pontos do eixo temporal.

Outra forma de representar uma entidade perene é através de um mapeamento para o diagrama ER convencional. É o que mostra a figura 16, onde para cada instância do conjunto entidade Função, o conjunto-relacionamento Função-T apresenta tantas tuplas <OID,t> quanto forem os pontos do tempo do eixo temporal.

FIGURA 16 - MAPEAMENTO DA ENTIDADE PERENE PARA O ER CONVENCIONAL



FONTE: [ANT1999]

3.5.3 RELACIONAMENTOS

Segundo [ANT1999], o mapeamento dos relacionamentos temporais pode ser visualizado no seguinte exemplo: o conjunto-relacionamento Alocação, que é do tipo temporal, é representado no diagrama ER convencional pelo conjunto-relacionamento que associa Empregado-T e Função-T, que são dois conjuntos-entidade derivados a partir de relacionamentos de empregados e funções com pontos do eixo temporal.

Uma instância de Empregado-T se associa apenas com instâncias de Função_T de mesmo ponto do tempo, e vice-versa, isto é, sendo < OIDE, OIDF, t > uma tupla do conjunto Alocação, esta é resultante da associação de uma instância < OIDE, tE > de Empregado-T com uma instância <OIDF, tF > da Função-T, onde tE = tF = t.

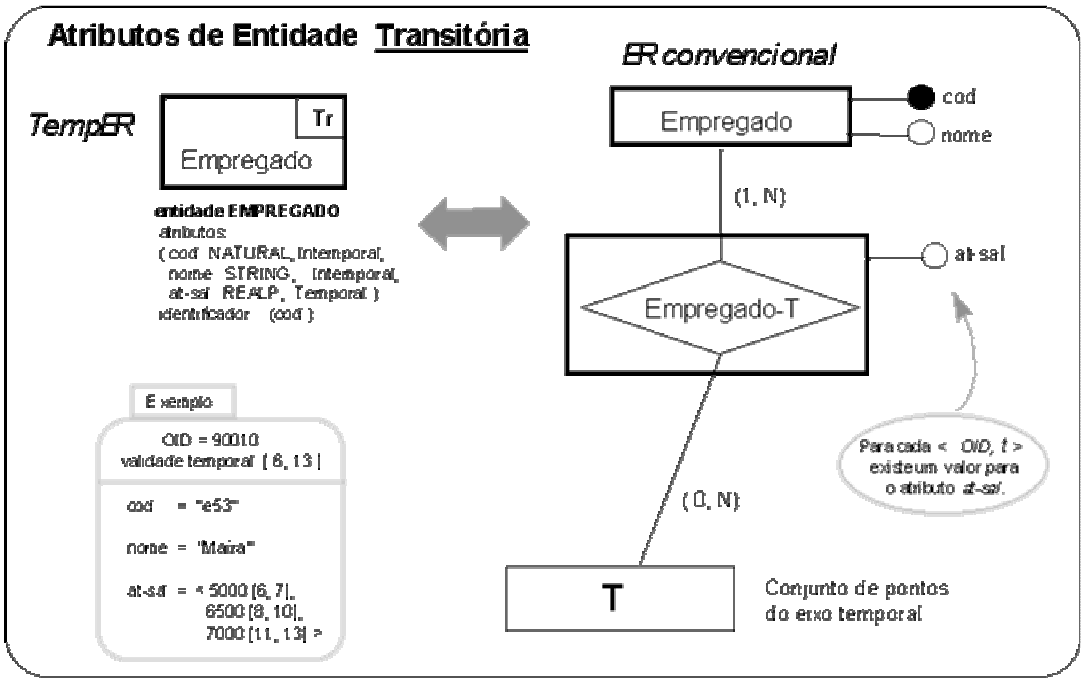
Os relacionamentos temporais só são válidos nos momentos específicos pelos seus rótulos temporais. Por exemplo, a lotação do empregado de OID 1003 no departamento de OID 9012 (ver tabela referente ao conjunto-relacionamento Lotação da figura 7) só é válido nos momentos 11, 12, 13, 14 19 e 20 do eixo temporal.

A validade no tempo de um relacionamento temporal sempre está contida dentro da interseção das existências das entidades associadas. Ou seja, o conjunto de pontos do tempo que define a validade de um relacionamento do tipo temporal é um subconjunto de pontos que definem as existências das entidades associadas.

3.5.4 ATRIBUTOS

A associação de um atributo temporal a uma entidade transitória é ilustrada através do mapeamento do modelo TempER para diagrama ER convencional. Trata-se do atributo at_Sal, que para cada empregado contém toda a história salarial. No diagrama ER convencional este atributo aparece conectado ao conjunto-entidade especial que modela a perspectiva temporal dos empregados (Empregado-T), ou seja, para cada par de < OID do empregado, ponto do tempo > existe um e apenas um valor de salário. Este mapeamento pode ser visto na figura 17.

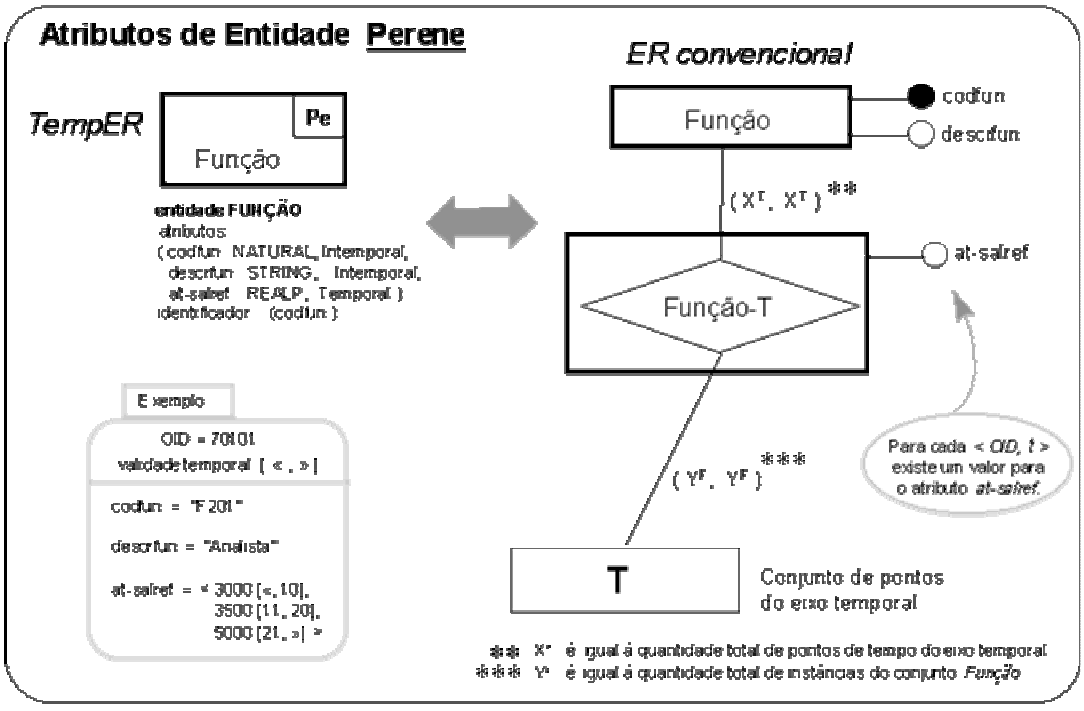
FIGURA 17 - MAPEAMENTO DE ATRIBUTOS TEMPORAIS DE ENTIDADE TRANSITÓRIA PARA ER CONVENCIONAL



FONTE: [ANT1999]

A associação de um atributo temporal a uma entidade perene poder ser ilustrada via o mapeamento para um diagrama ER convencional. Como demonstrado na figura 18, onde o atributo AT_SalRef, para cada função, contém todos os valores de referência que houve ao longo do tempo. Por pertencer a uma entidade perene, este atributo deve apresentar valor em todos os momentos do eixo temporal.

FIGURA 18 - MAPEAMENTO DE ATRIBUTOS TEMPORAIS DE UMA ENTIDADE PERENE PARA ER CONVENCIONAL



FONTE: [ANT1999]

4 ESPECIFICAÇÃO DO PROTÓTIPO

Este capítulo trata da especificação do protótipo, dos seus objetivos, das ferramentas utilizadas e da especificação.

O protótipo proposto é uma ferramenta de modelagem de dados, para o auxilio no projeto de criação de banco de dados temporal. Este protótipo tem por finalidade gerar um *script* de criação de uma base de dados com os aspectos temporais inclusos como também o tratamento da validade destes dados.

4.1 OBJETIVO GERAL

O objetivo geral do protótipo é ser uma ferramenta que auxilie no desenvolvimento de um projeto de banco de dados que incorpore os aspectos temporais, bem como os mecanismos de controle de inserção, alteração e exclusão dos dados, respeitando as integridades referentes à validade temporal.

4.1.1 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do protótipo são:

a) receber de entrada um arquivo contendo um *script* de criação de uma base de dados gerado a partir do programa ERWIN para o SGBD ORACLE ;

b) Incorporar os aspectos temporais;

c) gerar um *script* de saída para criação de uma base de dados com os aspectos temporais incluídos, bem como todos os controles para a integridade dos dados a serem inseridos nesta base.

4.2 FERRAMENTAS UTILIZADAS

Para o desenvolvimento do trabalho optou-se pelo ambiente de desenvolvimento DELPHI, por sua disponibilidade na Universidade. Para maiores informações sobre DELPHI pode ser encontrado em [CAN1997].

Analisaram-se dois Sistemas gerenciadores de bancos de dados, o SQL Server e o ORACLE. Optou-se pelo uso do ORACLE por estar disponível o seu uso nas dependências da Universidade. Sobre o banco de dados ORACLE pode ser encontrado em [SAR1999].

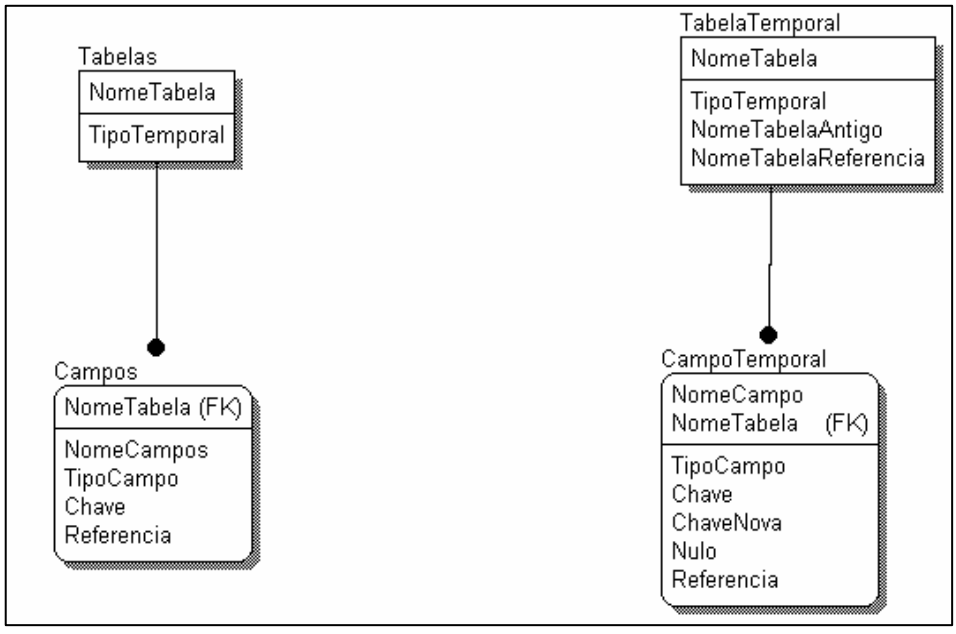
Foi utilizado o ErWin para a geração do *script* de entrada da ferramenta, pela sua facilidade de uso e pela disponibilidade na universidade.

Foram analisados diversos tipos de modelagem de banco de dados temporal, e optou-se pelo modelo TempER por ele ser um modelo que com poucas adaptações pode ser desenhado no modelo ER convencional.

4.3 MODELO ENTIDADE RELACIONAMENTO

O Modelo Entidade Relacionamento (MER) enfatiza os principais objetos ou entidades do protótipo. O protótipo utilizará uma estrutura na forma de tabelas de memórias, sumuladas pelo componente Delphi TmemoryTable da biblioteca de componentes RX. Na figura 19 é demonstrada a estrutura destas tabelas. As tabelas com o nome de Tabelas e campos são usadas para armazenar os dados de entrada, já as tabelas com o nome de TabelaTemporal e CampoTemporal são usadas para gerar a saída do protótipo.

FIGURA 19 – MODELO MER DAS TABELAS DE MEMÓRIA



4.4 DADOS DE ENTRADA

Para poder dar entrada ao modelo de dados do tipo TempER, optou-se por usar um modelo ER convencional com algumas adaptações, como descrito abaixo:

a) Para modelar uma Entidade Perene, é utilizado o prefixo “PE_” mais o nome da entidade. Exemplo: PE_Funcao;

b) Para uma entidade transitória, é utilizado o prefixo “TR_” mais o nome da entidade. Exemplo: TR_Empregado;

c) Para um relacionamento temporal, é utilizado o prefixo “T_” mais o nome do relacionamento. Exemplo: T_Alocação;

d) Para um relacionamento intemporal, não é utilizado nenhum prefixo;

e) Para um atribulo temporal, é utilizado o prefixo “AT_” mais o nome do atributo. Exemplo: AT_Salario;

f) Para atributos intemporais, não é utilizado nenhum prefixo.

Todas estas nomenclaturas já fazem parte do modelo TempER como foi visto no capítulo 3.4, onde trata dos componentes do modelo TempER.

Com estas pequenas adaptações consegue-se modelar um modelo TempER em um gerador de modelos ER convencional, como é o caso do ERWIN.

O protótipo agirá da seguinte forma:

- Quando encontrar uma entidade perene deve adicionar à entidade o campo OID como identificador. Não é necessário adicionar os atributos de tempo, pois a sua validade é para todo o período em que o dado existir;

- Quando encontrar uma entidade transitória deve adicionar à entidade o campo OID e criar uma nova entidade que contenha o OID , a data de início da validade e a data do fim da validade;

- Quando encontrar um relacionamento temporal deve criar uma tabela que contenha os OIDs dos relacionamentos, a data de início de validade e a data do fim da validade;

- Quando encontrar um relacionamento intemporal deve criar uma tabela contendo os OIDs dos relacionamentos sem menção de tempo;

- Quando encontrar um atributo temporal deve criar uma nova tabela contendo o OID, data de início da validade e a data de fim da validade, este atributo não é representado na tabela original;

- Quando encontrar atributos intemporais, estes são criados igual ao modelo ER convencional.

O protótipo também deverá criar mecanismos que garantam a validade dos dados a serem inseridos no banco de dados temporal, através do uso de triggers (gatilhos).

4.5 ESPECIFICAÇÃO DO PROTÓTIPO

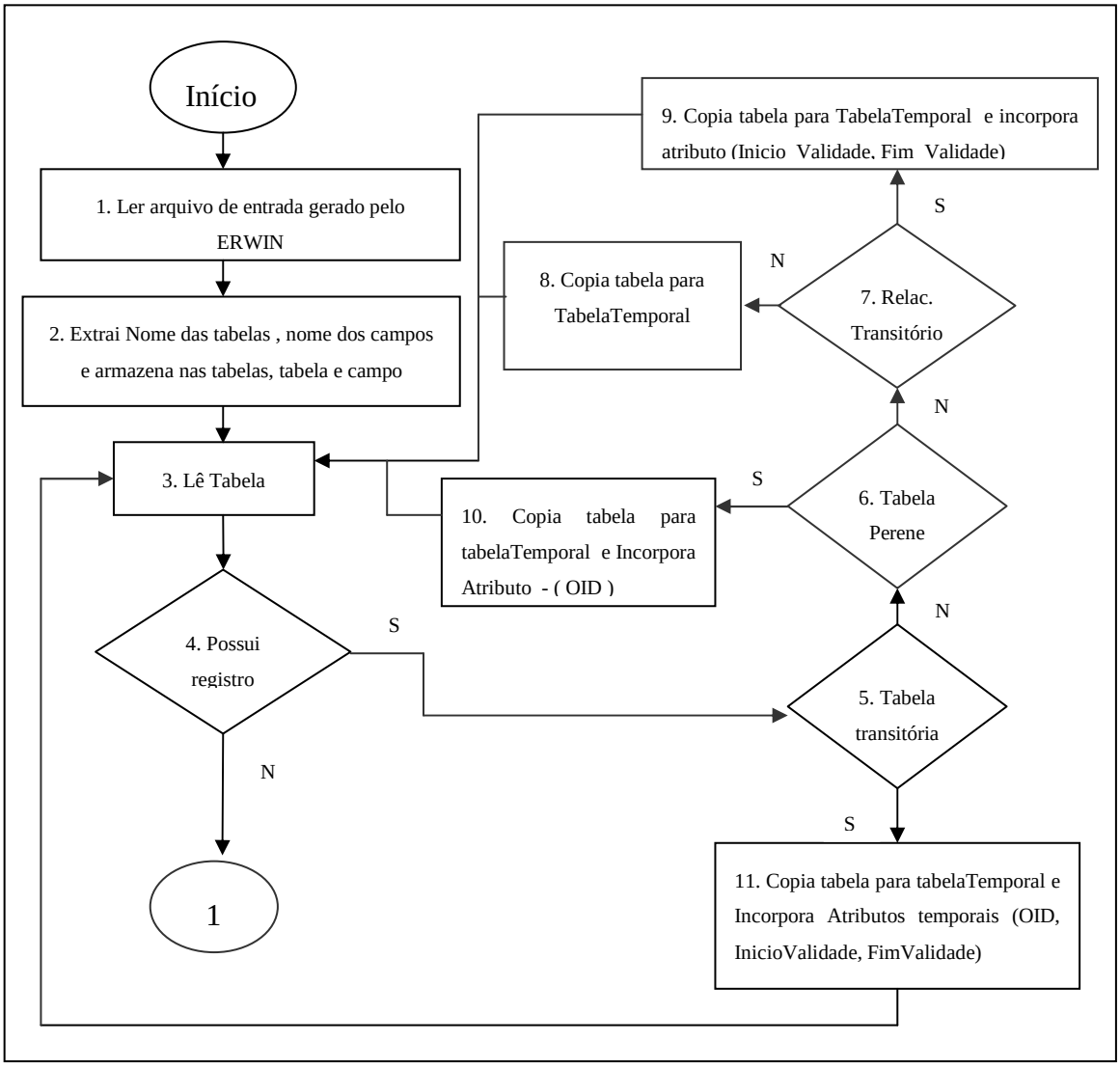
Na figura 20 é apresentado uma especificação do protótipo, que dá uma visão macro do seu funcionamento.

Cada passo citado nas figuras 20 e 21 será descrito abaixo.

- Passo 1 – Ler arquivo de entrada gerado pelo ERWIN, o protótipo copia o arquivo texto para a memória para ser processado e vai para o passo 2;

- Passo 2 – Extrai nome das tabelas e nome dos campos e armazena nas tabelas, “tabela” e “campo”. Neste passo o protótipo analisa cada *script* de criação de tabelas e extrai deles o nome da tabela e o nome dos campos. O nome da tabela é armazenado na tabela de memória “tabela” e os campos ele armazena na tabela de memória campo, juntamente com a informação do tipo de dado, se o campo é nulo, se o campo é chave e se o campo é referência de alguma tabela e vai para o passo 3;

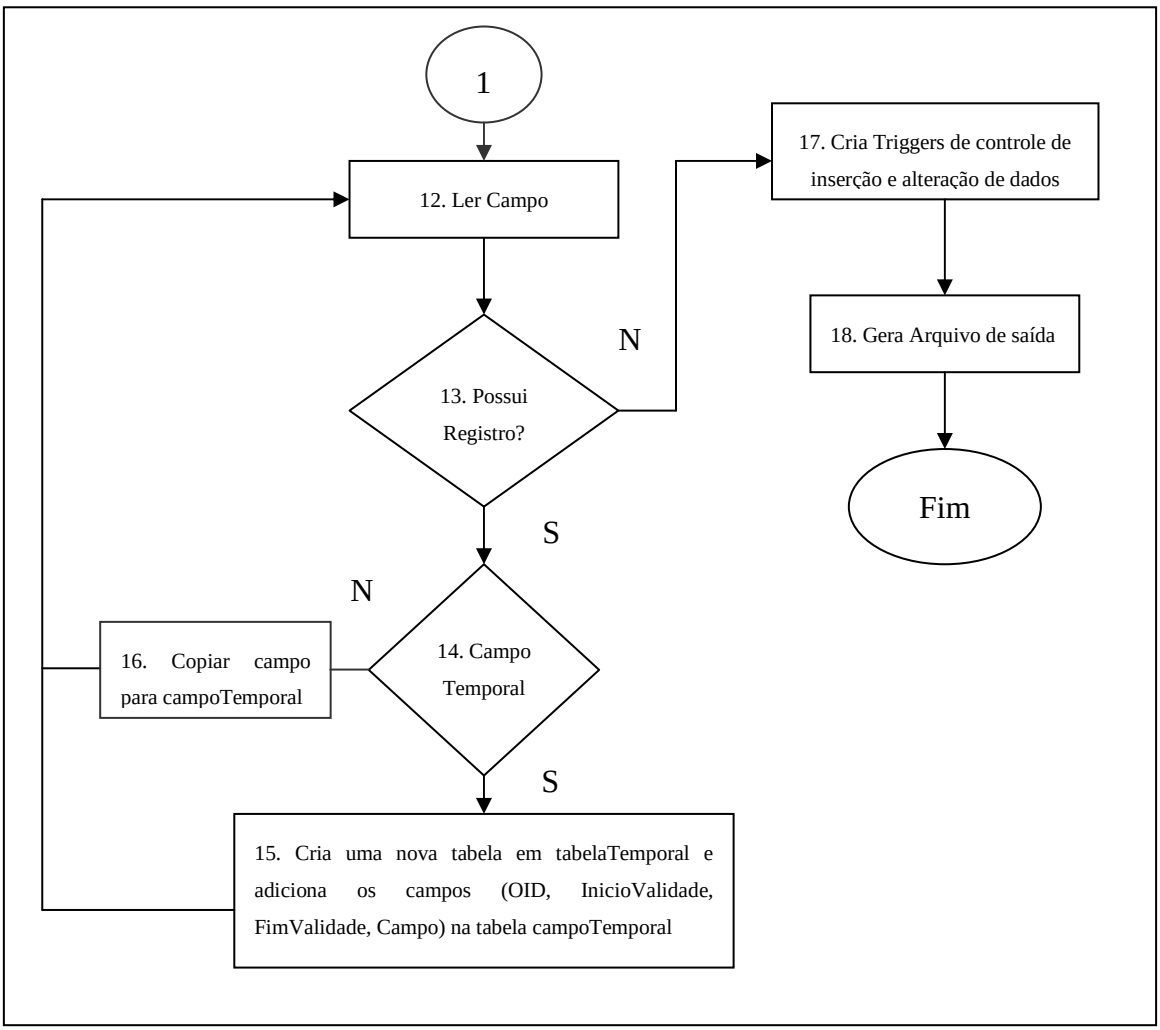
FIGURA 20 - ESPECIFICAÇÃO DO PROTÓTIPO



- Passo 3 – Lê tabela. Neste passo o protótipo faz um laço para ver todas as tabelas que estão armazenadas na tabela de memória “tabela” e vai para o passo 4;

- Passo 4 – Possui registro “tabela”. Neste passo o protótipo verifica se existe registro na tabela para ser processado. Se existir vai para o passo 5, senão vai para o passo 12;

FIGURA 21 – CONTINUAÇÃO DA ESPECIFICAÇÃO DO PROTÓTIPO



- Passo 5 - Tabela transitória. Neste passo o protótipo verifica se a tabela possui o prefixo ´TR_' de tabela transitória. Se for transitória ele vai para o passo 11, senão vai para o passo 6;

- Passo 6 – Tabela Perene. Neste passo o protótipo verifica se a tabela possui o prefixo 'PE_' de tabela perene. Se for perene ele vai para o passo 10, senão vai para o passo 7;

- Passo 7 – Relacionamento transitório. Neste passo o protótipo verifica se a tabela é um relacionamento transitório através do prefixo 'T_'. Se for

relacionamento transitório vai para o passo 9 senão é relacionamento intemporal e vai para o passo 8;

- Passo 8 – Copia tabela para TabelaTemporal. Neste passo o protótipo copia o nome da tabela para a “tabelaTemporal”;

- Passo 9 – Copia tabela para TabelaTemporal e incorpora atributo (Inicio Validade, Fim Validade). Neste passo o protótipo copia o nome da tabela para a “tabelaTemporal” retirando o prefixo ‘T_’. copia o nome da tabela para “NomeTabelaAntigo” e na tabela “CampoTemporal” adiciona os campos “Inicio_Validade” e “Fim_Validade”;

- Passo 10 – Copia tabela para tabelaTemporal e incorpora atributo – (OID). Neste passo o protótipo copia o nome da tabela para a “tabelaTemporal” retirando o prefixo ‘PE_’, copia nome da tabela para “NomeTabelaAntigo” e na tabela “CampoTemporal” adiciona o campo OID para esta tabela. Note que na entidade perene não é necessario criar os campos de validade temporal, pois eles podem ser simulados conforme descrito no mapeamento de entidade perene. Após executar este passo ele retorna para o passo 3;

- Passo 11 – Copia tabela para TabelaTemporal e incorpora atributos temporais (OID, Inicio_Validade, Fim_Validade). Neste passo o protótipo copia o nome da tabela para a “TabelaTemporal” retirando o prefixo ‘TR_’, copia nome da tabela para o campo “NomeTabelaAntigo” e na tabela “CampoTemporal” adiciona os campos “OID”, “Inicio_Validade” e “Fim_Validade” para esta tabela. Após executar este passo ele retorna para o passo 3;

- Passo 12 – Lê campo. Neste passo o protótipo faz um laço para ver todos os campos que estão armazenadas na tabela de memória campo e vai para o passo 9;

- Passo 13 - Possui registro. Neste passo o protótipo verifica se existe registro na tabela campo para ser processado. Se existir vai para o passo 14, senão vai para o passo 17;

- Passo 14 – Campo Temporal. Neste passo o protótipo verifica se o campo é temporal através do prefixo ‘AT_’. Se for atributo temporal, vai para o passo 15, senão vai para o passo 16;

- Passo 15 – Cria uma nova tabela em tabela temporal e adiciona os campos (OID, Inicio Validade, Fim Validade e o campo temporal. Nesta fase o protótipo cria uma nova tabela com o nome da junção do nome da tabela, mais o nome do campo, e adiciona os campos OID, Inicio_Validade, Fim_Validade e o campo que era um atributo temporal sem o prefixo ‘AT_’. Após retorna para o passo 12;

- Passo 16 – Cria campo para CampoTemporal. Neste passo o protótipo copia o nome do campo para a tabela CampoTemporal. Após retorna para o passo 12;

- Passo 17 – Cria triggers de controle de inserção e alteração de dados. Neste passo o protótipo cria todas as triggers necessárias para o controle de inclusão de dados e vai para fim;

- Fim.

Os passos 1 e 2 se referem a rotina ExtraiTabelas do código fonte que mostra de que maneira as tabelas e campos forma extraídos do *script* de entrada. Já os passos de 3 até 16 se

referem a rotina GeraTemporal, onde ele analiza as tabelas e campos e gera as tabelas e campos necessários para o controle temporal. No passo 17 a ferramenta gera as triggers necessarias para o controle da entrada de dados. Todos estes trechso de código estão no anexo 2.

4.5.1 RESTRIÇÕES DO MODELO

O protótipo tem como limitação a modelagem apenas do tempo de validade e a restrição de permitir um atributo temporal por tabela.

Na alteração ele não verifica se existe duplicidade na própria tabela..

Não é feita verificação em relação aos nomes repetidos de campos e tabelas.

Possibilidade de um campo temporal por tabela.

5 FUNCIONAMENTO DO PROTÓTIPO

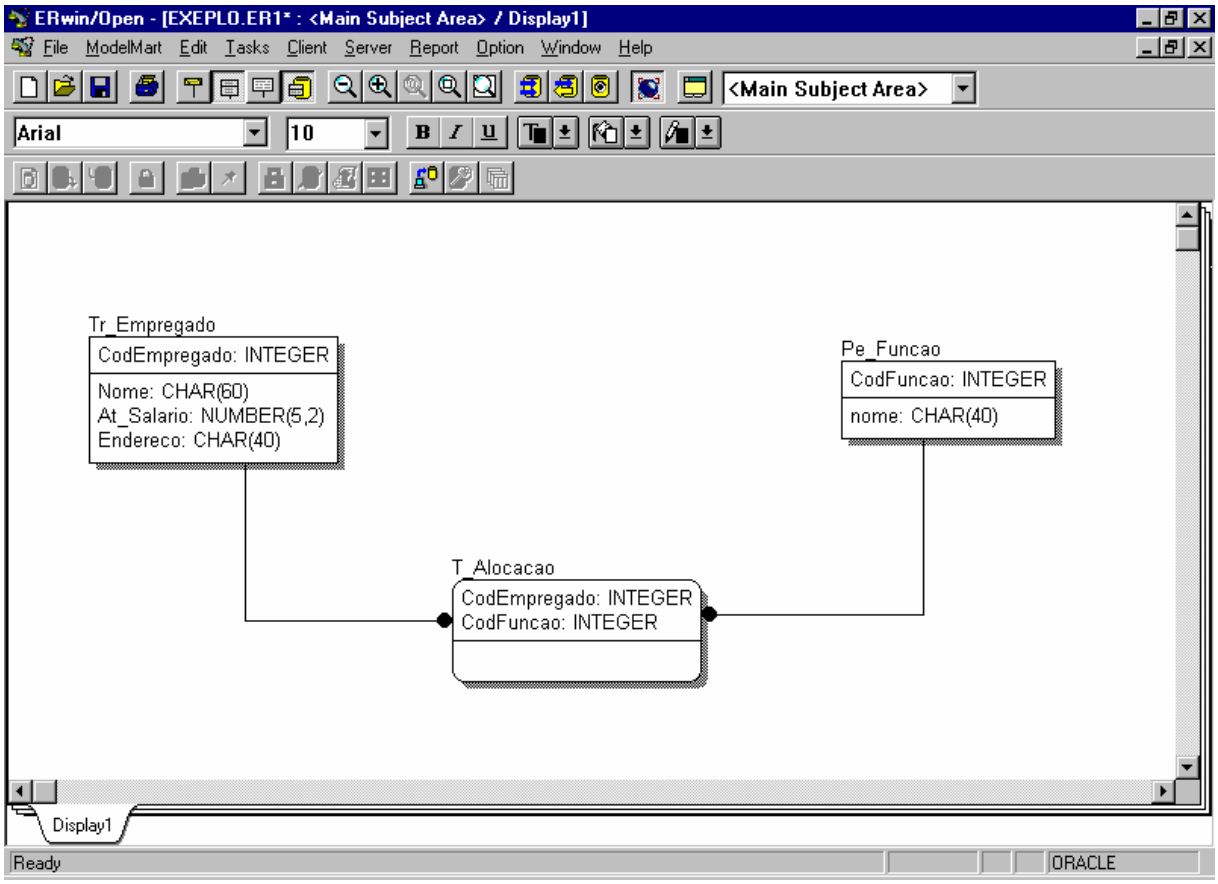
Serão descritas todas as fases do protótipo para que o seu entendimento seja de melhor forma.

Para melhor apresentar o protótipo, criou-se um estudo de caso, que será aplicado a solução do mesmo através do protótipo.

Um determinado Empregado que trabalha na empresa X tem uma data de admissão e uma de demissão, ou ser readmitido novamente. Este funcionário possui um determinado período de validade dentro da empresa. Enquanto o período for válido este funcionário pode ser alocado para as funções que existem na empresa, sempre uma para cada período válido. As funções por sua vez tem validade por todo o período, não sendo necessário o seu controle.

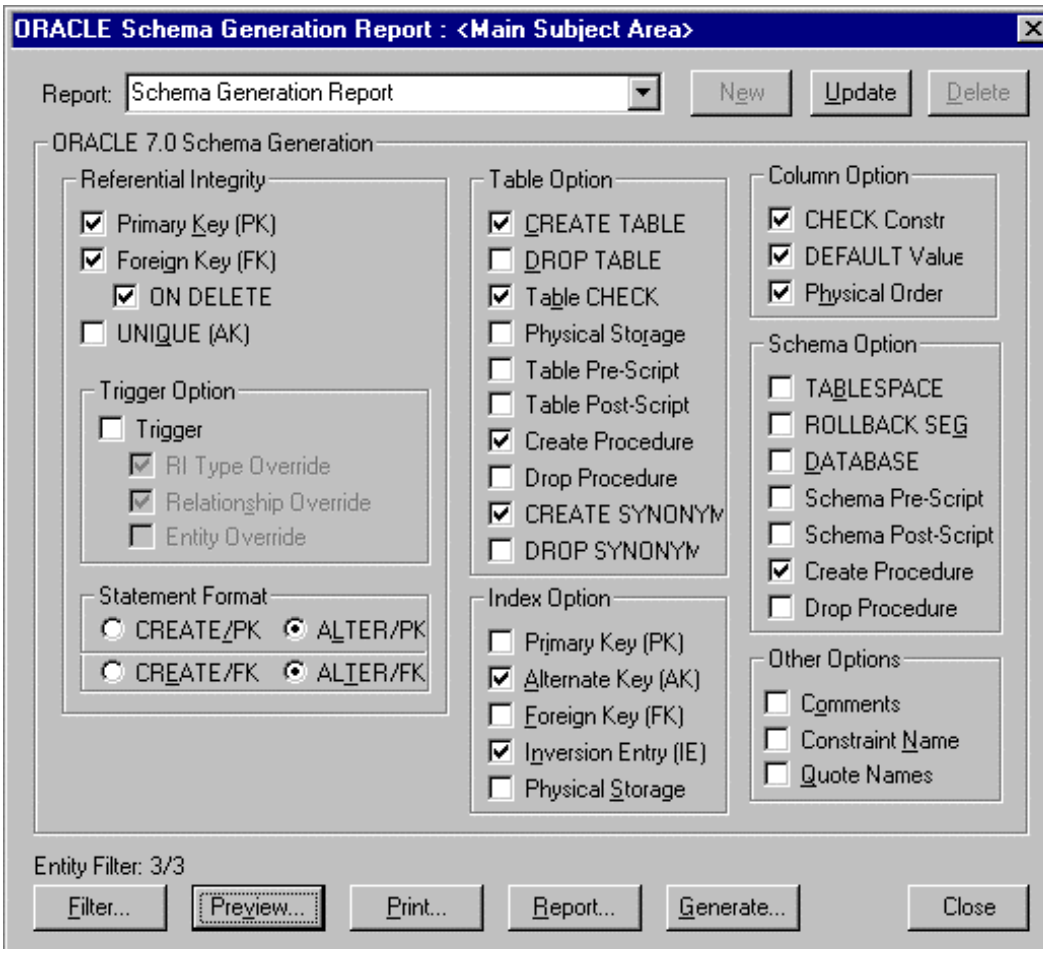
Primeiro cria-se um modelo de dados no programa ERWIN como demonstrado na figura 22.

FIGURA 22 – MODELO EXEMPLO MODELADO NO ERWIN



Após o modelo estar pronto no ERWIN é preciso gerar um script de criação de tabelas para usar como entrada no protótipo, para a geração do modelo temporal com os atributos inclusos. Este *script* é gerado no ERWIN no menu "*Tasks – Forward enginner / schema generator*". Como demonstrado na figura 23.

FIGURA 23 – TELA DO ERWIN PARA A GERAÇÃO DO SCRIPT



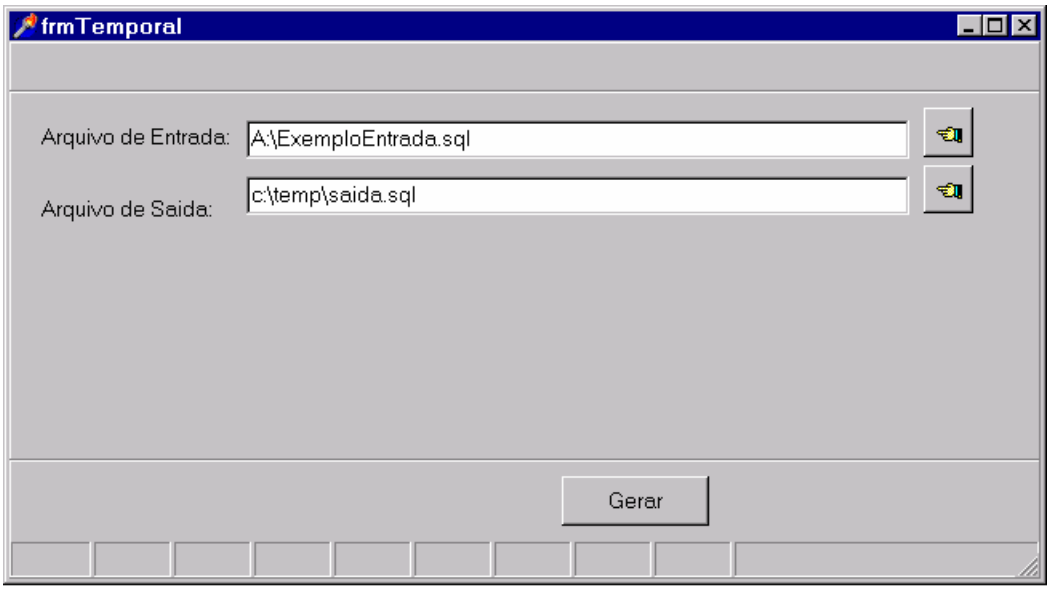
O ERWIN gera um *script* como demonstrado no quadro 1.

QUADRO 1 – SCRIPT GERADO PELA FERRAMENTA ERWIN

```
CREATE TABLE Pe_Funcao ( CodFuncao INTEGER NOT NULL, nome CHAR(40) NULL );
ALTER TABLE Pe_Funcao ADD  ( PRIMARY KEY (CodFuncao) ) ;
CREATE TABLE T_Alocacao ( CodEmpregado INTEGER NOT NULL,
     CodFuncao INTEGER NOT NULL );
ALTER TABLE T_Alocacao ADD (PRIMARY KEY (CodEmpregado,CodFuncao) );
CREATE TABLE Tr_Empregado ( CodEmpregado INTEGER NOT NULL,
     Nome CHAR(60) NULL, At_Salario NUMBER(5,2) NULL,  Endereco CHAR(40) NULL );
ALTER TABLE Tr_Empregado ADD  ( PRIMARY KEY (CodEmpregado) ) ;
ALTER TABLE T_Alocacao ADD  ( FOREIGN KEY (CodFuncao) REFERENCES Pe_Funcao ) ;
ALTER TABLE T_Alocacao  ADD  ( FOREIGN KEY (CodEmpregado)
                    REFERENCES Tr_Empregado ) ;
```

Próximo passo é abrir o protótipo e entrar com o nome do arquivo que contem o *script* gerado pelo ERWIN, e dar o nome de saída do arquivo que conterá o *script* para a criação do banco de dados com os aspectos temporais incorporados, como demonstrado na figura 24.

FIGURA 24 – TELA DO PROTÓTIPO



Como saída o protótipo gera um arquivo contendo os *script* de criação de tabelas bem como os demais mecanismos para controle da entrada e saída de dados nesta base de dados. O *script* de saída gerado pelo protótipo está no anexo 1.

QUADRO 2 – EXEMPLO DA INSERÇÃO DE DADOS

```
SQL> INSERT INTO EMPREGADO VALUES (1,1,'TESTE 1','ENDERECO TESTE');
1 linha criada.
SQL> INSERT INTO EMPREGADO_T VALUES (1,'01-JAN-2000','31-DEZ-2000');
1 linha criada.
SQL> INSERT INTO EMPREGADO_SALARIO VALUES (1,'01-JAN-2000','01-JAN-2001',500);
INSERT INTO EMPREGADO_SALARIO VALUES (1,'01-JAN-2000','01-JAN-2001',500)
*
ERRO na linha 1:
ORA-20004: TABELA Empregado_T NAO POSSUI ESTA CHAVE
ORA-06512: em "INSCRICAOVESTIBA.EMPREGADO_SALARIO_01", line 13
ORA-04088: erro durante a execução do gatilho 'INSCRICAOVESTIBA.EMPREGADO_SALARIO_01'
SQL> INSERT INTO EMPREGADO_SALARIO VALUES (1,'01-JAN-2000','31-DEZ-2000',500);
1 linha criada.
SQL> INSERT INTO FUNCAO VALUES (1,1,'AUXILIAR');
1 linha criada.
SQL> INSERT INTO ALOCACAO VALUES (1,1,'01-JAN-2000','01-JAN-2001',1,1);
INSERT INTO ALOCACAO VALUES (1,1,'01-JAN-2000','01-JAN-2001',1,1)
*
ERRO na linha 1:
ORA-20004: TABELA Empregado_T NAO POSSUI ESTA CHAVE
ORA-06512: em "INSCRICAOVESTIBA.ALOCACAO_01", line 13
ORA-04088: erro durante a execução do gatilho 'INSCRICAOVESTIBA.ALOCACAO_01'
SQL>INSERT INTO ALOCACAO VALUES (1,1,'01-JAN-2000','01-DEZ-2000',1,1);
1 linha criada.
```

Depois este *script* deve ser processado pelo Oracle, através de um programa de manipulação de dados como é o caso do SQL Plus.

No quadro 2 pode ser visto como o banco atua na manipulação dos dados. Na primeira linha é inserido um empregado de código 1.

Na segunda linha é inserido na tabela Empregado_T o empregado 1 com a data de início de validade como 01/01/2000 e a data de fim de validade 31/12/2000.

Na terceira linha é inserido na tabela Empregado_Salario o empregado 1 com a data de início de validade como 01/01/2000 e fim de validade como 01/01/2001 e nesta linha ocorreu um erro porque a data de fim de validade só tem validade até 31/12/2000, que está armazenado na tabela Empregado_T.

Na quarta linha é executada o mesmo comando, só que com a data de fim de validade como 31/12/2000, e ele aceitou normalmente, pois está dentro do período de validade.

Na quinta linha é inserido a função 1.

Na sexta linha é inserido na tabela Alocação o empregado 1, na função 1 com a data de início de validade como 01/01/2000 e fim de validade como 01/01/2001 e nesta linha ocorreu um erro porque a data de fim de validade só tem validade ate 31/12/2000, que esta armazenado na tabela Empregado_T.

Na sétima linha é executado o mesmo comando só que com a data de fim de validade para 01/12/2000 e ele aceitou, pois pertence ao período válido.

As chaves primárias que foram geradas no *script* de entrada foram transformadas em chaves secundarias no *script* de saída como pode ser visto no anexo 1.

6 CONCLUSÃO

Neste trabalho procurou-se levantar informações sobre modelos de banco de dados temporal e gerar uma ferramenta que desse suporte para a implementação destes aspectos de uma forma mais amigável para o usuário Utilizou-se o modelo TempER para o desenvolvimento e implementação da ferramenta.

A ferramenta constituída permitiu a modelagem do tempo de validade, o controle de inclusão, exclusão e alteração de dados respeitando a validade temporal através do uso de *triggers* (gatilhos).

A ferramenta permite o mapeamento de um modelo temporal para o SGBD Oracle, através da incorporação dos aspectos temporais.

As principais limitações da ferramenta incluem o suporte da modelagem de banco de dados temporal de tempo de validade, suporte do modelo TempER, e ainda pode-se apenas incluir um atributo temporal por tabela.

Uma dificuldade enfrentada no decorrer do trabalho foi encontrar um modelo temporal que fosse voltado para várias abordagens de análise como ER, orientação a objetos e etc.

Como sugestão para trabalhos futuros a ferramenta poderia implementar o tempo de transação e os bancos bitemporais, ou criar um ambiente para a modelagem do modelo TempER. Outra sugestão seria criar uma ferramenta para os modelos temporais orientados à objetos.

Outra sugestão seria a criação de uma linguagem de manipulação de dados para o modelo Temp.

ANEXOS

Anexo 1 – Script de saída do protótipo

```
CREATE TABLE Funcao (
      OID INTEGER NOT NULL
,     CodFuncao INTEGER NOT NULL
,     nome CHAR(40)
 );
CREATE TABLE Alocacao (
      OID_Empregado INTEGER NOT NULL
,     OID_Funcao INTEGER NOT NULL
,     INICIO_VALIDADE DATE
,     FIM_VALIDADE DATE
,     CodEmpregado INTEGER NOT NULL
,     CodFuncao INTEGER NOT NULL
 );
CREATE TABLE Empregado (
      OID INTEGER NOT NULL
,     CodEmpregado INTEGER NOT NULL
,     Nome CHAR(60)
```

```
,      Endereco CHAR(40)
 );
CREATE TABLE Empregado_T (
      OID INTEGER NOT NULL
,      INICIO_VALIDADE DATE
,      FIM_VALIDADE DATE
 );
CREATE TABLE Empregado_Salario (
      OID INTEGER NOT NULL
,      INICIO_VALIDADE DATE
,      FIM_VALIDADE DATE
,      Salario NUMBER(5,2)
 );
ALTER TABLE Funcao ADD ( PRIMARY KEY ( OID
 ) ) ;
ALTER TABLE Empregado ADD ( PRIMARY KEY ( OID
 ) ) ;
CREATE  INDEX Funcao_01 ON Funcao ("CodFuncao") ;
CREATE  INDEX Empregado_02 ON Empregado ("CodEmpregado") ;
CREATE OR REPLACE TRIGGER EMPREGADO_03 BEFORE DELETE
ON EMPREGADO
```

```
FOR EACH ROW
DECLARE NumLinha INTEGER;
BEGIN
  SELECT COUNT(*) INTO NumLinha
    FROM EMPREGADO_T
   WHERE :OLD.OID = OID;
  IF (NumLinha > 0) THEN
    RAISE_APPLICATION_ERROR('-20001','TABELA EMPREGADO_T POSSUI
OID');
  END IF;
END;
/


CREATE OR REPLACE TRIGGER Funcao_04 BEFORE DELETE
ON Funcao
FOR EACH ROW
DECLARE NumLinha INTEGER;
BEGIN
  SELECT COUNT(*) INTO NumLinha
    FROM Alocacao
   WHERE :OLD.OID = OID_Funcao;
```

```
  IF (NumLinha > 0) THEN
    RAISE_APPLICATION_ERROR('-20001','TABELA Alocacao POSSUI OID');
  END IF;
END;
/

CREATE OR REPLACE TRIGGER EMPREGADO_T_05 BEFORE DELETE
ON EMPREGADO_T
FOR EACH ROW
DECLARE NumLinha INTEGER;
BEGIN
  SELECT COUNT(*) INTO NumLinha
    FROM Empregado_Salario
   WHERE :OLD.OID = OID
     AND NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY'))
BETWEEN
       NVL(:OLD.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
       NVL(:OLD.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))
    AND NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY')) BETWEEN
       NVL(:OLD.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
       NVL(:OLD.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'));
```

```
        IF (NumLinha > 0) THEN
         RAISE_APPLICATION_ERROR('-20002','TABELA Empregado_Salario POSSUI
OID');
        END IF;
        SELECT COUNT(*) INTO NumLinha
         FROM Alocacao
         WHERE :OLD.OID = OID_Empregado
          AND NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY'))
BETWEEN
            NVL(:OLD.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
            NVL(:OLD.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))
          AND NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY')) BETWEEN
            NVL(:OLD.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
            NVL(:OLD.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'));
        IF (NumLinha > 0) THEN
         RAISE_APPLICATION_ERROR('-20003','TABELA Alocacao POSSUI OID');
        END IF;
       END;
       /

       CREATE OR REPLACE TRIGGER ALOCACAO_06 BEFORE INSERT OR
UPDATE
```

```
ON Alocacao
FOR EACH ROW
DECLARE NumLinha INTEGER;
BEGIN
  SELECT COUNT(*) INTO NumLinha
   FROM Empregado_T
  WHERE :NEW.OID_Empregado = OID
   AND  NVL(:NEW.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY'))
BETWEEN
     NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
     NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))
   AND  NVL(:NEW.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))
BETWEEN
     NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
     NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'));
 IF (NumLinha = 0) THEN
  RAISE_APPLICATION_ERROR('-20004','TABELA Empregado_T NAO POSSUI
ESTA CHAVE');
 END IF;
 SELECT COUNT(*) INTO NumLinha
  FROM funcao
 WHERE :NEW.OID_Funcao = OID;
```

```
 IF (NumLinha = 0) THEN

  RAISE_APPLICATION_ERROR('-20005','TABELA Funcao NAO POSSUI ESTA
CHAVE');

 END IF;

END;

/

CREATE OR REPLACE TRIGGER Empregado_Salario_07 BEFORE INSERT OR
UPDATE

ON Empregado_Salario

FOR EACH ROW

DECLARE NumLinha INTEGER;

BEGIN

 SELECT COUNT(*) INTO NumLinha

  FROM Empregado_T

 WHERE :NEW.OID = OID

  AND  NVL(:NEW.Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY'))
BETWEEN

    NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND

    NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))

  AND  NVL(:NEW.Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'))
BETWEEN

    NVL(Inicio_Validade,TO_DATE('01/01/1900','DD/MM/YYYY')) AND
```

```
        NVL(Fim_Validade,TO_DATE('31/12/2999','DD/MM/YYYY'));
  IF (NumLinha = 0) THEN
   RAISE_APPLICATION_ERROR('-20004','TABELA Empregado_T NAO POSSUI
ESTA CHAVE');
  END IF;
END;
/
CREATE OR REPLACE TRIGGER Empregado_T_08 BEFORE INSERT OR
UPDATE
ON Empregado_T
FOR EACH ROW
DECLARE NumLinha INTEGER;
BEGIN
  SELECT COUNT(*) INTO NumLinha
    FROM Empregado
   WHERE :NEW.OID = OID;
  IF (NumLinha = 0) THEN
   RAISE_APPLICATION_ERROR('-20004','TABELA Empregado NAO POSSUI
ESTA CHAVE');
  END IF;
END;
/
```

Anexo 2 – Partes do código DELPHI

```
procedure TfrmTemporal.btGerarClick(Sender: TObject);
begin
  ExtraiTabelas;   // Extrai as tabelas do arquivo de entrada
  GeraTemporal;    // Gera numa tabela os campos no formato Temporal
  Saida := TStringList.Create;
  CriaTabelas;     // Gera o script de criação das tabelas e campos
  CriaTriggers;    // Cria as triggers
  Saida.SaveToFile(edSaida.Text);
  Saida.Destroy;
end;

procedure TfrmTemporal.ExtraiTabelas;
  var i : integer;
      Entrada : Tstringlist;
      Linha, aux : String;
      Tabela,campo : string;

function CarregaPalavra : string;
begin
  while (length(trim(linha)) = 0) and (i < entrada.Count) do
```

```
  begin
    linha := trim(entrada.Strings[i]);
    inc(i);
  end;
  result := copy2SymbDel(linha,' ');
end;

function CarregaPalavraAte(caracteres : Array of char) : string;
  var Int,j : integer;
      achou : Boolean;
      campo : char;
begin
  while (length(trim(linha)) = 0)  and (i < entrada.Count) do
  begin
    linha := trim(entrada.Strings[i]);
    inc(i);
  end;
  int := 1;
  achou := false;
  while (length(linha) >= int) and (not achou) do
  begin
```

```
  for j := 0 to SizeOf(caracteres) - 1 do
   begin
    if linha[int] = caracteres[j] then
    begin
      achou := true;
      campo := caracteres[j];
    end;
   end;
   inc(int);
  end;
  result := copy2SymbDel(linha,campo);
  linha := copy(linha,1,length(linha));
end;


begin
  if Length(trim(edEntrada.Text)) = 0 then
    raise exception.Create('Arquivo de entrada não existe');
  if Length(trim(edSaida.Text)) = 0 then
    raise exception.Create('Arquivo de saida não existe');
  tTabelas.Open;
  tcampos.Open;
```

```
entrada := TstringList.Create;
entrada.LoadFromFile(edEntrada.Text);
i := 0;
try
 while i < entrada.Count do
 begin
  aux := CarregaPalavraAte([' ']);
  if aux = 'CREATE' then
  begin
    aux := CarregaPalavraAte([' ']);
    if aux = 'TABLE' then
    begin
      aux := carregaPalavra;
      // cria uma tabela na memoria
      ttabelas.insert;
      ttabelasnomeTabela.AsString := aux;
      ttabelas.Post;
      aux := carregaPalavra;
      if copy(aux,1,1) = '(' then
      begin
        aux := CarregaPalavra;
```

```
while aux <> ')' do
begin
 // cria um campo na tabela
 tCampos.Insert;
 tCamposNomeTabela.AsString := ttabelasNomeTabela.AsString;
 tCamposNomeCampo.AsString := aux;
 aux := CarregaPalavra;
 // copia para o campo o tipo de dados
 tCamposTipoCampo.AsString := aux;
 aux := CarregaPalavraAte([' ',',']);
 if not (aux[1] in [',',')']) then
 begin
  if UPPERCASE(aux) = 'NOT' then
  begin
   // tipo de dado nao nulo
   tCamposNulo.AsString := 'N';
   aux := CarregaPalavraAte([' ',',',')']);
  end;
  if UPPERCase(aux) = 'NULL' then
  begin
   // para desencargo de consiencia
```

```
          aux := CarregaPalavraAte([' ',';']);
        end;
        if aux = ',' then
          aux := CarregaPalavra;
      end;
    end;
    aux := CarregaPalavra;
  end;
 end;
end
else if aux = 'ALTER' then
begin
 aux := CarregaPalavra;
 if aux = 'TABLE' then
 begin
  aux := carregaPalavra;
  tabela := aux;
  aux := carregaPalavra;
  if aux = 'ADD' then
  begin
   aux := CarregaPalavra;
```

```
if copy(aux,1,1) = '(' then
begin
  aux := CarregaPalavra;
  if aux = 'PRIMARY' then
  begin
    aux := carregaPalavra;
    if aux <> 'KEY' then
      showmessage('Erro');
    aux := copy(linha,1,1);
    linha := copy(linha,2,length(linha));
    if copy(aux,1,1) = '(' then
    begin
      aux := carregaPalavraate([' ',',',')']);
      while aux <> ')' do
      begin
        MarcaChavePrimaria(Tabela,aux);
        aux := carregaPalavra;
        // localiza o campo na tabela e marca como chave
        if aux = ',' then
          aux := CarregaPalavraate([' ',',',')']);
      end;
```

```
    aux := CarregaPalavra;
    if aux = ')' then
      aux := carregaPalavra;
  end
  else
    showmessage('erro');
end
else if aux = 'FOREIGN' then
begin
  aux := carregaPalavraAte(['(']);
  aux := copy(linha,1,1);
  Linha := copy(linha,2,length(linha));
  if copy(aux,1,1) = '(' then
  begin
    aux := carregaPalavraate([' ',',',')']);
    while aux <> ')' do
    begin
      // campo de referencia
      campo := aux;
      aux := carregaPalavra;
      //if aux = ',' then
```

```
            //  aux := CarregaPalavra;
          end;
          aux := CarregaPalavra;
          if aux = 'REFERENCES' then
          begin
            aux := carregaPalavra;
            // tabela de referencia
            MarcaChaveExtrangeira(Tabela,Campo,aux);
          end;
          aux := CarregaPalavra;
          if aux = ')' then
            aux := carregaPalavra;
        end;
      end;
    end;
  end;
 end;
end;
//saida.SaveToFile(edSaida.Text);
finally
```

```
    entrada.Destroy;
   end;
  end;

  procedure TfrmTemporal.GeraTemporal;
   var aux : String;
  begin
   tTabelaTemporal.Open;
   tCampoTemporal.Open;
   tTabelas.First;
   tCampos.First;
   while not tTabelas.EOF do
   begin
    if (uppercase(copy(tTabelasNomeTabela.AsString,1,3)) = 'TR_') then
    begin
     tTabelaTemporal.Insert;
     tTabelaTemporalNomeTabela.asString :=
trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString)));
     tTabelaTemporalTipoTemporal.AsString := 'TR';
     tTabelaTemporalNomeTabelaAntigo.AsString := tTabelasNomeTabela.AsString;
     tTabelaTemporal.Post;
```

```
tTabelaTemporal.Insert;

tTabelaTemporalNomeTabela.asString := trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString))) + '_T';

tTabelaTemporalTipoTemporal.AsString := 'TT';

//tTabelaTemporalNomeTabelaAntigo.AsString := tTabelasNomeTabela.AsString;

tTabelaTemporal.Post;

tCampoTemporal.Insert;

tCampoTemporalNomeTabela.AsString := trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString)));

tCampoTemporalNomeCampo.AsString := 'OID';

tCampoTemporalTipoCampo.AsString := 'INTEGER';

tCampoTemporalNulo.AsString := 'N';

tCampoTemporalChaveNova.AsString := 'S';

tCampoTemporal.Post;

tCampoTemporal.Insert;

tCampoTemporalNomeTabela.AsString := trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString))) + '_T';

tCampoTemporalNomeCampo.AsString := 'OID';

tCampoTemporalTipoCampo.AsString := 'INTEGER';

tCampoTemporalNulo.AsString := 'N';

tCampoTemporalReferencia.AsString := trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString)));
```

```
            tCampoTemporalChave.AsString := 'S';
            tCampoTemporal.Post;
            tCampoTemporal.Insert;
            tCampoTemporalNomeTabela.AsString :=
trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString))) + '_T';
            tCampoTemporalNomeCampo.AsString := 'INICIO_VALIDADE';
            tCampoTemporalTipoCampo.AsString := 'DATE';
            tCampoTemporalChave.AsString := 'S';
            tCampoTemporal.Post;
            tCampoTemporal.Insert;
            tCampoTemporalNomeTabela.AsString :=
trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString))) + '_T';
            tCampoTemporalNomeCampo.AsString := 'FIM_VALIDADE';
            tCampoTemporalTipoCampo.AsString := 'DATE';
            tCampoTemporalChave.AsString := 'S';
            tCampoTemporal.Post;
          end
          else if uppercase(copy(tTabelasNomeTabela.AsString,1,3)) = 'PE_' then
          begin
            tTabelaTemporal.Insert;
            tTabelaTemporalNomeTabela.asString :=
trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString)));
```

```
            tTabelaTemporalTipoTemporal.AsString := 'PE';
            tTabelaTemporalNomeTabelaAntigo.AsString := tTabelasNomeTabela.AsString;
            tTabelaTemporal.Post;
            tCampoTemporal.Insert;
            tCampoTemporalNomeTabela.AsString :=
trim(copy(ttabelasNomeTabela.AsString,4,length(tTabelasNomeTabela.asString)));
            tCampoTemporalNomeCampo.AsString := 'OID';
            tCampoTemporalTipoCampo.AsString := 'INTEGER';
            tCampoTemporalNulo.AsString := 'N';
            tCampoTemporalChaveNova.AsString := 'S';
            tCampoTemporal.Post;
          end
          else if uppercase(copy(tTabelasNomeTabela.AsString,1,2)) = 'T_' then
          begin
            tTabelaTemporal.Insert;
            tTabelaTemporalNomeTabela.asString :=
trim(copy(ttabelasNomeTabela.AsString,3,length(tTabelasNomeTabela.asString)));
            tTabelaTemporalTipoTemporal.AsString := 'T';
            tTabelaTemporalNomeTabelaAntigo.AsString := tTabelasNomeTabela.AsString;
            tTabelaTemporal.Post;
```

```
InsereRelacionamentoOID(tTabelaTemporalNomeTabelaAntigo.AsString,tTabelaTemporalNomeTabela.asString,'T');
                tCampoTemporal.Insert;
                tCampoTemporalNomeTabela.AsString := tTabelaTemporalNomeTabela.asString;
                tCampoTemporalNomeCampo.AsString := 'INICIO_VALIDADE';
                tCampoTemporalTipoCampo.AsString := 'DATE';
                tCampoTemporalChave.AsString := 'S';
                tCampoTemporal.Post;
                tCampoTemporal.Insert;
                tCampoTemporalNomeTabela.AsString := tTabelaTemporalNomeTabela.asString;
                tCampoTemporalNomeCampo.AsString := 'FIM_VALIDADE';
                tCampoTemporalTipoCampo.AsString := 'DATE';
                tCampoTemporalChave.AsString := 'S';
                tCampoTemporal.Post;
              end
              else
              begin
                tTabelaTemporal.Insert;
                tTabelaTemporalNomeTabela.asString := ttabelasNomeTabela.AsString;
                tTabelaTemporalTipoTemporal.AsString := ' ';
```

```
            tTabelaTemporalNomeTabelaAntigo.AsString := tTabelasNomeTabela.AsString;
            tTabelaTemporal.Post;

InsereRelacionamentoOID(tTabelaTemporalNomeTabelaAntigo.AsString,tTabelaTemporalNomeTabela.asString,' ');
          end;
          tTabelas.Next;
        end;
        tCampos.First;
        while not tcampos.EOF do
        begin
          aux := LocalizaTabelaNova(tCamposNomeTabela.AsString);
          if UpperCase(copy(tCamposNomeCampo.AsString,1,3)) = 'AT_' then
          begin
            tTabelaTemporal.Insert;
            if Uppercase(copy(tCamposNomeTabela.AsString,1,3)) = 'PE_' then
            begin
              tTabelaTemporalNomeTabela.asString := aux + '_' + trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));
              tTabelaTemporalNomeTabelaReferencia.AsString := aux;
              tTabelaTemporalTipoTemporal.AsString := 'TT';
            end
```

```
        else
        begin
          tTabelaTemporalNomeTabela.asString := aux + '_' +
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));
          tTabelaTemporalTipoTemporal.AsString := 'TT';
          tTabelaTemporalNomeTabelaReferencia.AsString := aux + '_T';
        end;
        tTabelaTemporal.Post;
        tCampoTemporal.Insert;
        tCampoTemporalNomeTabela.AsString := aux + '_' +
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));
        tCampoTemporalNomeCampo.AsString := 'OID';
        tCampoTemporalTipoCampo.AsString := 'INTEGER';
        if Uppercase(copy(tCamposNomeTabela.AsString,1,3)) = 'PE_' then
          tCampoTemporalReferencia.AsString := aux
        else
          tCampoTemporalReferencia.AsString := aux + '_T';
        tCampoTemporalNulo.AsString := 'N';
        tCampoTemporalChave.AsString := 'S';
        tCampoTemporal.Post;
        tCampoTemporal.Insert;
```

```
tCampoTemporalNomeTabela.AsString := aux + '_' +
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));

tCampoTemporalNomeCampo.AsString := 'INICIO_VALIDADE';

tCampoTemporalTipoCampo.AsString := 'DATE';

//tCampoTemporalNulo.AsString := 'N';

tCampoTemporalChave.AsString := 'S';

tCampoTemporal.Post;

tCampoTemporal.Insert;

tCampoTemporalNomeTabela.AsString := aux + '_' +
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));

tCampoTemporalNomeCampo.AsString := 'FIM_VALIDADE';

tCampoTemporalTipoCampo.AsString := 'DATE';

//tCampoTemporalNulo.AsString := 'N';

tCampoTemporalChave.AsString := 'S';

tCampoTemporal.Post;

tCampoTemporal.Insert;

tCampoTemporalNomeTabela.AsString := aux + '_' +
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));

tCampoTemporalNomeCampo.AsString :=
trim(copy(tCamposNomeCampo.AsString,4,length(tCamposNomeCampo.asString)));

tCampoTemporalTipoCampo.AsString := tCamposTipoCampo.AsString;

tCampoTemporalNulo.AsString := tCamposnulo.AsString;
```

```
      tCampoTemporalChave.AsString := 'N';
      tCampoTemporal.Post;
    end
    else
    begin
      tCampoTemporal.Insert;
      tCampoTemporalNomeTabela.AsString := aux;
      tCampoTemporalNomeCampo.AsString := tCamposNomeCampo.AsString;
      tCampoTemporalTipoCampo.AsString := tCamposTipoCampo.AsString;
      tCampoTemporalChave.AsString := tCamposChave.AsString;
      tCampoTemporalNulo.AsString := tCamposnulo.AsString;
      tCampoTemporal.Post;
    end;
    tCampos.Next;
  end;
end;
procedure TfrmTemporal.CriaTriggers;
  var i : integer;
  Flag : Boolean;
  Tipo, campos, aux : String;
begin
```

```
i := 1;


// cria as triggers de controle de Alteração de dados
tTabelaTemporal.First;
while not tTabelaTemporal.Eof do
begin
  Saida.Add(' ');
  Saida.Add(' ');
  Saida.Add(' CREATE OR REPLACE TRIGGER ' + TTabelaTemporalNomeTabela.AsString + '_' + IntToStr(i));
  Saida.Add(' BEFORE UPDATE ON ' + TTabelaTemporalNomeTabela.AsString);
  Saida.Add(' FOR EACH ROW ');
  Saida.Add(' DECLARE NumLinha INTEGER; ');
  Saida.Add(' BEGIN ');
  campos := RetornaCamposTemporal(tTabelaTemporalNomeTabela.AsString);
  while length(campos) > 0 do
  begin
    aux := copy2SymbDel(campos,',');
    Saida.Add('   IF (:OLD.' + Aux + ' <> :NEW.' + Aux + ') THEN' );
    Saida.Add('     RAISE_APPLICATION_ERROR(-''' + IntToStr(20000 + i) + '''');
    Saida.Add('      ,"CAMPO ' + Aux + ' NÃO PODE SER ALTERADO");');
```

```
    Saida.Add('   END IF; ');
    if copy(Campos,1,1) = ',' then
      campos := copy(campos,2,length(Campos));
  end;
  Tipo := tTabelaTemporalTipoTemporal.AsString;

  Saida.Add(' END; ');
  Saida.Add('/ ');
  Inc(I);
  tTabelaTemporal.Next;
end;
// cria as triggers de controle de duplicidade
tTabelaTemporal.First;
while not tTabelaTemporal.Eof do
begin
  Saida.Add(' ');
  Saida.Add(' ');
  Saida.Add(' CREATE OR REPLACE TRIGGER ' +
TTabelaTemporalNomeTabela.AsString + '_' + IntToStr(i));
  Saida.Add(' BEFORE INSERT ON ' + TTabelaTemporalNomeTabela.AsString);
  Saida.Add(' FOR EACH ROW ');
```

```
Saida.Add(' DECLARE NumLinha INTEGER; ');
Saida.Add(' BEGIN ');
Saida.Add('   SELECT COUNT(*) INTO NumLinha ');
Saida.Add('     FROM ' + TTabelaTemporalNomeTabela.AsString);
campos := RetornaCamposTemporal(tTabelaTemporalNomeTabela.AsString);
flag := true;
while length(campos) > 0 do
begin
  aux := copy2SymbDel(campos,',');
  if flag then
  begin
    Saida.Add('    WHERE :NEW.' + Aux + ' = ' + Aux);
    Flag := False;
  end
  else
    Saida.Add('      AND :NEW.' + Aux + ' = ' + Aux);
  if copy(Campos,1,1) = ',' then
    campos := copy(campos,2,length(Campos));
end;
Tipo := tTabelaTemporalTipoTemporal.AsString;
if (Tipo = 'TT') OR (Tipo = 'T') then
```

```
begin
  Saida.Add('                                                    AND
(NVL(:NEW.INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))
BETWEEN ');
  Saida.Add('
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) AND ');
  Saida.Add('
NVL(FIM_VALIDADE,TO_DATE("31/12/2999","DD/MM/YYYY"))');
  Saida.Add('                                                    OR
NVL(:NEW.FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) BETWEEN ');
  Saida.Add('
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) AND ');
  Saida.Add('
NVL(FIM_VALIDADE,TO_DATE("31/12/2999","DD/MM/YYYY")))');
end;
Saida.Add('   ; ');
Saida.Add('   IF (NumLinha > 0) THEN ');
Saida.Add('     RAISE_APPLICATION_ERROR(-'" + IntToStr(20000 + i) + "");
if (Tipo = 'TT') OR (Tipo = 'T') then
begin
  Saida.Add('      ,"TABELA ' + TTabelaTemporalNomeTabela.AsString + ' POSSUI
OID E DATA DE INICIO E FIM DE VALIDADE PARA ESTE PERIODO");');
end
else
```

```
          begin
            Saida.Add('      ,"TABELA ' + TTabelaTemporalNomeTabela.AsString + ' POSSUI
OID");');
          end;
          Saida.Add('   END IF; ');
          Saida.Add(' END; ');
          Saida.Add('/ ');
          Inc(I);
          tTabelaTemporal.Next;
        end;

        // cria triggers de controle de Alteracao TABELA PAI
        tCampoTemporal.First;
        while not tCampoTemporal.Eof do
        begin
          if Length(trim(tCampoTemporalReferencia.AsString)) > 0 then
          begin
            Tipo := RetornaTipoTemporal(TCampoTemporalReferencia.AsString);
            if Tipo = 'TT' then
            begin
              Saida.Add(' ');
```

```
Saida.Add(' ');

Saida.Add('  CREATE  OR  REPLACE  TRIGGER  '  +
TCampoTemporalReferencia.AsString + '_' + IntToStr(i));

Saida.Add(' BEFORE UPDATE ON ' + tCampoTemporalReferencia.AsString);

Saida.Add(' FOR EACH ROW ');

Saida.Add(' DECLARE NumLinha INTEGER; ');

Saida.Add('       INICIOVELHO DATE; ');

Saida.Add('       FIMVELHO DATE; ');

Saida.Add('       INICIONOVO DATE; ');

Saida.Add('       FIMNOVO DATE; ');

Saida.Add(' BEGIN ');

Saida.Add('                                                    SELECT
NVL(:OLD.INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))');

Saida.Add('   INTO INICIOVELHO FROM SYS.DUAL ;');

Saida.Add('                                                    SELECT
NVL(:NEW.INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))');

Saida.Add('   INTO INICIONOVO FROM SYS.DUAL ;');

Saida.Add('   IF (INICIOVELHO <> INICIONOVO) THEN ');

Saida.Add('    IF (INICIOVELHO < INICIONOVO) THEN ');

Saida.Add('      SELECT COUNT(*) INTO NumLinha ');

Saida.Add('      FROM ' + TCampoTemporalNomeTabela.AsString);
```

```
Saida.Add('                    WHERE :OLD.OID = ' +
TCampoTemporalNomeCampo.AsString);

Saida.Add('                    AND ((INICIOVELHO <=
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                    AND INICIONOVO >
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))) ');

Saida.Add('                    OR (INICIOVELHO <=
NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                    AND INICIONOVO >
NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")))) ');

Saida.Add('    ; ');

Saida.Add('    IF (NumLinha > 0) THEN ');

Saida.Add('        RAISE_APPLICATION_ERROR(-'" + IntToStr(20000 + i) +
"");

Saida.Add('        ,"TABELA ' + TCampoTemporalNomeTabela.AsString + '
POSSUI OID E DATA DE INICIO E FIM DE VALIDADE PARA O PERIODO A SER
ALTERADO");');

Saida.Add('    END IF; ');

Saida.Add('  ELSE  ');

Saida.Add('    SELECT COUNT(*) INTO NumLinha ');

Saida.Add('    FROM ' + TCampoTemporalNomeTabela.AsString);

Saida.Add('                    WHERE :OLD.OID = ' +
TCampoTemporalNomeCampo.AsString);
```

```
Saida.Add('                                AND ((INICIOVELHO > NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                                AND INICIONOVO <= NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))) ');

Saida.Add('                                OR (INICIOVELHO > NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                                AND INICIONOVO <= NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")))) ');

Saida.Add('      ; ');

Saida.Add('      IF (NumLinha > 0) THEN ');

Saida.Add('          RAISE_APPLICATION_ERROR(-'" + IntToStr(20000 + i) + "");

Saida.Add('             ,"TABELA ' + TCampoTemporalNomeTabela.AsString + ' POSSUI OID E DATA DE INICIO E FIM DE VALIDADE PARA O PERIODO A SER ALTERADO");');

Saida.Add('      END IF; ');

Saida.Add('    END IF; ');

Saida.Add('   END IF;   ');

Saida.Add('                                SELECT NVL(:OLD.FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))');

Saida.Add('   INTO FIMVELHO FROM SYS.DUAL ;');

Saida.Add('                                SELECT NVL(:NEW.FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))');

Saida.Add('   INTO FIMNOVO FROM SYS.DUAL ;');
```

```
Saida.Add('   IF (FIMVELHO <> FIMNOVO) THEN ');

Saida.Add('    IF (FIMVELHO > FIMNOVO) THEN ');

Saida.Add('     SELECT COUNT(*) INTO NumLinha ');

Saida.Add('     FROM ' + TCampoTemporalNomeTabela.AsString);

Saida.Add('                    WHERE :OLD.OID = ' + TCampoTemporalNomeCampo.AsString);

Saida.Add('                    AND ((FIMVELHO >= NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                    AND FIMNOVO < NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))) ');

Saida.Add('                    OR (FIMVELHO >= NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                    AND FIMNOVO < NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")))) ');

Saida.Add('     ; ');

Saida.Add('     IF (NumLinha > 0) THEN ');

Saida.Add('         RAISE_APPLICATION_ERROR(-'" + IntToStr(20000 + i) + "");

Saida.Add('          ,"TABELA ' + TCampoTemporalNomeTabela.AsString + ' POSSUI OID E DATA DE INICIO E FIM DE VALIDADE PARA O PERIODO A SER ALTERADO");');

Saida.Add('     END IF; ');

Saida.Add('    ELSE  ');

Saida.Add('     SELECT COUNT(*) INTO NumLinha ');
```

```
Saida.Add('      FROM ' + TCampoTemporalNomeTabela.AsString);

Saida.Add('                                WHERE :OLD.OID = ' +
TCampoTemporalNomeCampo.AsString);

Saida.Add('                                AND ((FIMVELHO <
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                                AND FIMNOVO >=
NVL(INICIO_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY"))) ');

Saida.Add('                                OR (FIMVELHO <
NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")) ');

Saida.Add('                                AND FIMNOVO >=
NVL(FIM_VALIDADE,TO_DATE("01/01/1900","DD/MM/YYYY")))) ');

Saida.Add('      ; ');

Saida.Add('      IF (NumLinha > 0) THEN ');

Saida.Add('          RAISE_APPLICATION_ERROR(-'" + IntToStr(20000 + i) +
"");

Saida.Add('          ,"TABELA ' + TCampoTemporalNomeTabela.AsString + '
POSSUI OID E DATA DE INICIO E FIM DE VALIDADE PARA O PERIODO A SER
ALTERADO");');

Saida.Add('      END IF; ');

Saida.Add('    END IF; ');

Saida.Add('   END IF;   ');

Saida.Add(' END; ');

Saida.Add('/ ');

Inc(I);
```

```
    end;
  end;
  tCampoTemporal.Next;
end;
```

REFERÊNCIAS BIBLIOGRÁFICAS

[ANT1997] ANTUNES, Dante Carlos. **Modelagem temporal de sistemas: uma abordagem fundamentada em redes de Petri**. Porto Alegre: CPGCC da UFRGS, 1997. Dissertação de Mestrado.

[ANT1999] ANTUNES, Dante Carlos. **TempER - Uma Proposta de modelagem de dados temporal** 1999. Endereço Eletrónico: http://www.celepar.gov.br/batebyte/bb75/dados.htm. Data da consulta: 09/08/2000.

[CAN1997] CANTÚ, Marco. **Dominando o Delphi 3.** Trad. de José Carlos Barbosa dos Santos. São Paulo : Makron Books, 1997.

[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.

[CHE1976] CHEN, Peter S. **The entity-relationship model - toward a unified view of data. ACM Transactions on Dabatase Systems**, New York, v.1, n.1, p.9-28, Mar.1976.

[EDE1998] EDELWEIS, Nina. Bancos de Dados Temporais: Teoria e Prática. In: XVII Jornada de Atualização em Informática - **Anais do XVIII Congresso Nacional da Sociedade Brasileira de Computação "Rumo à Sociedade do Conhecimento"**, Volume II. Recife: Sociedade Brasileira de Computação, 1998. p.225-282.

[ELM1992] ELMASRI, Ramez; KOURAMA-JIAN, Vram. A temporal query language based on conceptual entities and roles. In: INTERNACIONAL

CONFERENCE ON THE ENTITY RELATIONSHIP APPROACH, 11., (1992 : Karlsruhe), **Anais**... Germany. 1992.

[ELM1994] ELMASRI, Ramez. **Fundamentals of Database Systems** : NAVATHE, Shamkant B.. Redwood City. CA : The Benjamin/cummings Publishing Company Inc, 1994.

[HUB1999] HÜBLER, Patrícia Nogueira. **Implementação de um gerenciador de banco de Dados Temporal para o Modelo TF-ORM**. 1999. Endereço Eletrónico:http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana99/hubler/hubler.html. Data da consulta: 16/08/2000.

[LOU1991] LOUCOPOULOS, P.; THEODOULIDIS, C.; WANGLER, B**.** The entity relationship time model and conceptual rule language. In: INTERNATIONAL CONFERENCE ON THE ENTITY RELATIONSHIP APPROACH, 1991. **Anais**... San Mateo, California. Proceedings... [S.l.: s.n.], 1991.

[SAR1999] SARAIVA, Armando dos Santos. **Programando em Oracle.** Rio de Janeiro : Infobook, 1999.

[TAU1991] TAUZOVICH, Branka. **Towards temporal extensions to the entity-relationship model**. **In: INTERNATIONAL CONFERENCE ON THE ENTITY RELATIONSHIP APPROACH, 10**., 1991, San Mateo, California. Proceedings... [S.l.: s.n.], 1991.

[TON2000] TONIAL, Fernando. **Implementação de um banco de dados temporal utilizando o modelo orientado a objeto TF-ORM**. Itajai, 2000. Graduação (Ciências da Computação) , Universidade do Vale do Itajai.