UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Implementação de Bancos de Dados Temporais com Versionamento de Esquemas: um estudo comparativo

por

CONSTÂNCIA DA SILVA SANTOS

Dissertação submetida à avaliação,
como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação.

Profª. Drª. Nina Edelweiss
Orientadora

Porto Alegre, abril de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Constância da Silva
Implementação de Banco de Dados Temporais com Versionamento de Esquemas: um estudo comparativo / por Constância da Silva Santos. Porto Alegre: PPGC da UFRGS, 2003.
68 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2003. Orientadora: Edelweiss, Nina.

1. Banco: Dados Temporais, 2. Evolução de Esquemas, 3. Versionamento de Esquemas. I. Edelweiss, Nina. II. Título.

Agradecimentos

À professora Nina Edelweiss por ter aceitado ser minha orientadora, por estar sempre disponível para esclarecer minhas dúvidas e pela orientação firme e segura. Meus sinceros agradecimentos.

À Renata Galante pela valiosa opinião durante a elaboração deste trabalho.

À minha família pelo carinho, incentivo e apoio, principalmente no período em que estive ausente.

A todos os colegas do MINTERCC, em especial a Ana Carla, Raquel e Miriam.

Aos amigos Antonio Tobias e Pedro Leon pela importante ajuda na implementação dos programas.

À Universidade Federal do Pará e a Universidade Federal do Rio Grande do Sul, pela realização do Mestrado Interinstitucional.

Sumário

Lista de Abreviaturas

BD	Banco de Dados
BDI	Banco de Dados da Intenção
BDE	Banco de Dados da Extensão
BDT	Banco de Dados Temporal
DBA	Database Administrator (administrador do banco de dados)
ODBC	Open database Conectivity
SGBD	Sistema Gerenciador de Bancos de Dados
SQL	Structured Query Language (linguagem de consulta estruturada)
TIK	Time Invariant Key
TRM	Temporal Relacional Model
TSQL	Temporal Structured Query Language

Lista de Figuras

Lista de Tabelas

Resumo

Nas aplicações do mundo real, os dados mudam com o passar do tempo. Devido à característica dinâmica das aplicações, o esquema conceitual também pode mudar para se adaptar às mudanças que freqüentemente ocorrem na realidade. Para representar esta evolução, uma nova versão do esquema é definida e os dados armazenados são adaptados à nova versão. Entretanto, existem aplicações que precisam acessar também o esquema sob os diversos aspectos de suas mudanças, requerendo o uso de versionamento de esquemas. Durante a evolução do esquema, o versionamento preserva todas as versões deste esquema e seus dados associados, possibilitando a recuperação dos dados através da versão com a qual foram definidos.

Ultimamente muitas pesquisas têm sido realizadas envolvendo as áreas de versionamento de esquemas e bancos de dados temporais. Estes bancos de dados provêm suporte ao versionamento de esquemas, pois permitem armazenar e recuperar todos os estados dos dados, registrando sua evolução ao longo do tempo. Apesar de muitos esforços, ainda não existem SGBDs temporais comercialmente disponíveis. A utilização de um modelo de dados temporal para a especificação de uma aplicação não implica, necessariamente, na utilização de um SGBD específico para o modelo. Bancos de dados convencionais podem ser utilizados desde que exista um mapeamento adequado entre o modelo temporal e o SGBD utilizado.

Este trabalho apresenta uma abordagem para a implementação de um banco de dados temporal permitindo o versionamento de esquemas, usando um banco de dados relacional, tendo como base o modelo temporal TRM (*Temporal Relational Model)*. Como forma de ilustrar apresenta-se um exemplo de implementação utilizando o SGBD DB2. O principal objetivo é avaliar diferentes técnicas de implementar e gerenciar o versionamento de esquemas em bancos de dados temporais. Para atingir esse objetivo, um protótipo foi desenvolvido para automatizar os mapeamentos do TRM para o DB2 e gerenciar o versionamento de esquemas e dados. Duas experiências de implementação foram realizadas utilizando formas diferentes de armazenar os dados - um repositório e vários repositórios - com o objetivo de comparar os resultados obtidos, considerando um determinado volume de dados e alterações. Um estudo de caso também é apresentado para validar as implementações realizadas.

Palavras-chave: Bancos de Dados Temporais, Evolução de esquemas, Versionamento de Esquemas.

TITLE: “IMPLEMENTATION OF TEMPORAL DATABASES WITH SCHEMA VERSIONING: A COMPARATIVE STUDY “

Abstract

In real world applications, data change over time. Due to the dynamic feature of the applications, the conceptual schema may also change to fit the alterations that reality happen. In order to represent this evolution, a new version of the schema is derived and data stored adapt to the new version. However, there are applications that need also to access the schema under several aspects of its changes, requiring the use of schemas versioning. As the schema evolves, versioning preserves all versions of this schema and its associated data, making possible data retrieval through the version with which they were defined.

Lately, much research has been carried out involving techniques of versioning areas of temporal schemas and database. These databases provide support to schemas versioning, once they make possible to store and retrieve all states of data, keeping their evolution over time. Despite all efforts, there are no temporal SGBDs commercially available. The use of a temporal data model for the specification of an application does not necessarily imply the use of a specific SGBD for the model. Conventional database can be used since there is an adequate mapping between the temporal model and SGBD used.

This work presents an approach for the implementation of a temporal database enabling the schemas versioning, using a relational database based on the TRM (*Temporal Relational Model)*. As a means of illustration, we show an example of an implementation that uses SGBD DB2.

The main objective is to evaluate different techniques of implementing and managing versioning of schemas in temporal database. In order to reach this objective, a prototype was developed to automate TRM mappings for DB2 and managing data and schemas versioning. Two experiences of implementation were accomplished using different ways of storing data – a repository and several repositories – with the objective of comparing results obtained, considering a certain volume of data and changes. A case study is also presented in order to validate implementations accomplished.

Keywords: Temporal database, schema evolution, schema versioning.

1 Introdução

Sistemas de bancos de dados são utilizados para armazenar informações do mundo real. Os bancos de dados convencionais armazenam apenas o estado corrente dos dados; sempre que uma informação nova é introduzida, a antiga é substituída. A necessidade de representar a evolução temporal dos dados e a constatação de que muitas vezes é preciso guardar o histórico desta evolução, levou à criação de bancos de dados temporais. Bancos de dados temporais armazenam todos os valores definidos aos dados, com rótulos temporais identificando suas validades. Desta forma, todos os estados do banco de dados, passados, presentes e futuros, podem ser recuperados. Dependendo das dimensões temporais suportadas, os seguintes tipos de bancos de dados temporais são identificados: banco de dados de tempo de transação, tempo de validade, bitemporal e multitemporal.

Devido à natureza dinâmica das aplicações, freqüentemente a evolução da estrutura do banco de dados (esquema conceitual) também é requerida, representada através do esquema conceitual. A modificação do esquema pode ser necessária para adaptar a aplicação aos novos requisitos, para aumentar a performance dos sistemas ou para corrigir eventuais erros de análise e projeto [EDE 94]. Para representar esta evolução, uma nova versão do esquema é definida e os dados já armazenados são adaptados à nova versão.

A área de evolução de esquemas tem sido amplamente estudada na comunidade científica. Segundo [GOR 97], a evolução de esquemas utilizando tempo é o processo que permite mudanças no esquema sem a perda de informações.

Porém, apenas a evolução do esquema nem sempre é suficiente; pode-se ter necessidade de acessar o esquema sob diversos aspectos de suas mudanças. Neste caso, o uso de versionamento de esquemas torna-se mais adequado.

Um sistema de banco de dados suporta o versionamento de esquemas quando permite a consulta de todos os dados, tanto retrospectivamente quanto prospectivamente, através das várias versões definidas pelo usuário [JEN 94].

Quando o objetivo é manter todo o histórico de uma aplicação, o sistema de banco de dados deve armazenar, além da evolução dos dados, também as versões antigas dos esquemas. Isto permite que os dados passados sejam recuperados de acordo com a versão do esquema correspondente.

Existem atualmente na literatura vários estudos sobre evolução de esquemas em bancos de dados temporais e propostas de ambientes que suportem evolução e versionamento de esquemas e dados [CAS 95, CAS 97, EDE 98, GAL 02, GAL 02a, MOR 99, ROD 99, WEI 99, WEI 2000]. Estas pesquisas tratam de diversos aspectos envolvidos no gerenciamento temporal de esquemas: a forma como os esquemas podem ser armazenados, como pode ser realizada a temporalidade de dados e de esquemas, formas de gerenciamento, etc. Apesar da importância da evolução de esquemas em

aplicações práticas, existem poucas experiências de sistemas implementados que gerenciem versões de esquemas.

Em [MOR 99] foi realizado um amplo estudo sobre os tipos de versionamento de esquemas e dados, suas formas de armazenamento e gerenciamento. Este estudo identificou 38 alternativas para a formação de um ambiente temporal que suporte o versionamento de esquemas e definiu uma estratégia para armazenamento das versões do esquema.

Com base em uma dessas alternativas, [ANG 2000] propôs uma estratégia de implementação, provando que é possível implementar um banco de dados temporal, permitindo a evolução e o versionamento de esquemas e dados, utilizando um banco de dados convencional.

Considerando as possibilidades de implementação, torna-se necessário um estudo que permita avaliar diferentes formas de implementar e gerenciar o versionamento de esquemas em bancos de dados temporais.

1.1 Objetivos

O objetivo deste trabalho é implementar, sobre um banco de dados convencional, um modelo de dados temporal que permita o versionamento de esquemas, utilizando duas alternativas diferentes para o armazenamento dos dados. A partir das implementações realizadas, comparar a eficiência de cada uma, com um determinado volume de dados e alterações.

O modelo de dados temporal escolhido para as implementações foi o TRM (*Temporal Relational Model*) [EDE 94, NAV 93], um modelo relacional que utiliza o tempo de validade para representar as informações temporais.

A utilização de um modelo de dados temporal para especificar uma aplicação não requer o uso de um SGBD específico para este modelo. Bancos de dados convencionais podem ser utilizados, através do mapeamento das informações temporais para atributos explícitos. Algumas implementações de modelos de dados temporais já foram realizadas sobre bancos de dados convencionais, mostrando sua viabilidade [ANG 2000, EDE 2000, HÜB 2000, SIM 98].

O BD convencional escolhido foi o DB2, um banco de dados muito utilizado comercialmente.

1.2 Organização do Texto

O trabalho está organizado da seguinte forma: o capítulo 2 apresenta os principais conceitos utilizados na especificação de aspectos temporais, bem como a classificação e as características de bancos de dados temporais; o capítulo 3 apresenta as características do TRM, o modelo temporal utilizado na implementação; o capítulo 4 apresenta as alternativas que permitem implementar bancos de dados temporais com suporte ao versionamento de esquemas; o capítulo 5 descreve o mapeamento do TRM

para o DB2, detalha a implementação do protótipo e apresenta um estudo de caso para validar a implementação; o capítulo 6 apresenta os resultados obtidos; o capítulo 7 apresenta a conclusão, identifica restrições e aponta trabalhos futuros.

2 Versionamento de Esquemas e Dados

Neste capítulo são apresentados os principais conceitos utilizados na especificação de aspectos temporais, bem como a classificação de bancos de dados temporais quanto ao seu suporte ao tempo. Também são apresentadas as definições dos termos: modificação de esquemas, evolução e versionamento de esquemas.

Nas aplicações do mundo real, freqüentemente os dados mudam com o passar do tempo. A constatação da evolução temporal dos dados e da necessidade de manter o registro desta evolução levou à criação de bancos de dados temporais. Bancos de dados temporais (BDT) armazenam todos os valores definidos para os dados, associando rótulos temporais que indicam quando essas definições foram efetuadas. Dessa forma, todos os estados dos dados (presentes, passados e futuros) podem ser recuperados [ETZ 98, TAN93, SNO 2000].

Dois diferentes rótulos temporais (*timestamps)* podem ser identificados em uma aplicação [JEN 98]:

- **tempo de transação -** tempo em que um valor é armazenado no banco de dados;
- **tempo de validade -** tempo em que um valor é verdadeiro no mundo real.

2.1 Classificação de Bancos de Dados Temporais

De acordo com o tipo de rótulo temporal utilizado, as seguintes categorias de bancos de dados são identificadas em [JEN 98]:

- **Bancos de Dados Instantâneos -** correspondem aos bancos de dados convencionais, que não possuem qualquer mecanismo para tratar com informações temporais de forma implícita. Sempre que um valor é modificado, o estado anterior é perdido e somente o último valor armazenado fica disponível para consultas. A manutenção das informações temporais só pode ser feita de forma explícita, através da inclusão de atributos definidos sobre o domínio tempo, e sua manipulação é feita através dos programas de aplicação;
- **Bancos de Dados de Tempo de Transação** – utilizam o tempo de transação dos dados como rótulo temporal. Neste tipo de banco de dados, a alteração do valor de uma propriedade não destrói o valor definido anteriormente. Todos os valores definidos ficam armazenados permanentemente, sendo válidos a partir do momento de sua definição. O tempo de transação é fornecido pelo SGBD;
- **Bancos de Dados de Tempo de Validade** – utilizam o tempo de validade dos dados como rótulo temporal. Neste tipo de BD, o tempo em que a informação foi definida não é armazenado, sendo armazenado somente o tempo em que a mesma é válida. Permitem a alteração de informações válidas em momentos presentes, passados ou futuros. O tempo de validade é fornecido pelo usuário;

- **Bancos de Dados Bitemporais** - utilizam o tempo de validade e o tempo de transação dos dados como rótulos temporais, permitindo armazenar as informações de forma mais completa. Toda a história do banco de dados fica armazenada. Pode-se ter acesso a todos os estados dos dados, tanto à história das transações realizadas quanto à história da validade dos dados. O estado atual do banco de dados é representado pelos dados atualmente válidos;
- **Bancos de Dados Multitemporais –** utilizam simultaneamente características de bancos de dados de tempo de transação, de tempo de validade e bitemporais. Nestes bancos de dados podem coexistir relações de diferentes formatos temporais. Permitem consultas ao passado, presente e futuro das informações.

Durante o ciclo de vida de um banco de dados, as modificações não se restringem somente aos dados. Devido à natureza dinâmica das aplicações, muitas vezes é necessário alterar também a estrutura do banco de dados, representada pelo esquema conceitual. Isto é devido a fatores tais como: adaptar a aplicação aos novos requisitos, aumentar a performance dos sistemas ou corrigir eventuais erros de análise e projeto [EDE 94].

Uma modificação no esquema conceitual pode afetar o sistema de diversas formas. Segundo Goralwalla, citado por [EDE 98], dois problemas fundamentais devem ser considerados:

- **semântica das alterações efetuadas no esquema -** efeitos das alterações no esquema na representação da aplicação, podendo ser perdidas informações importantes. O enfoque utilizado para solucionar este problema é definir um conjunto de atributos invariantes, que devem ser preservados nas modificações efetuadas. A cada alteração, o esquema deve ser validado através da verificação das restrições de integridade estrutural do modelo de dados;
- **propagação das alterações -** consiste nos efeitos da alteração na consistência da base de dados já existente. Este problema é resolvido através da adaptação da base de dados existente ao novo esquema, sem perder a sua consistência. Esta adaptação não deve destruir o esquema e nem os respectivos dados, mas sim encerrar a sua validade temporal. A evolução do esquema também pode afetar os programas da aplicação.

2.2 Modificação de Esquemas, Evolução e Versionamento

Há três níveis de suporte às mudanças de esquemas em bancos de dados: modificação, evolução e versionamento. No glossário de conceitos sobre bancos de dados temporais [JEN 98] são apresentadas as seguintes definições:

- **modificação de esquemas -** ocorre quando um SGBD permite alterações nas definições do esquema de uma base de dados povoada;
- **evolução de esquemas** - ocorre quando um SGBD permite modificações no esquema sem perder os dados da base de dados, garantindo a consistência e a integridade dos dados armazenados;

- **versionamento de esquemas** - ocorre quando um SGBD permite a consulta de todos os dados, tanto retrospectivamente quanto prospectivamente, através das várias versões do esquema definidas pelo usuário.

Ao contrário da evolução de esquemas, o versionamento requer suporte à história das alterações, retendo as definições passadas do esquema. O versionamento de esquemas pode ser classificado em dois tipos diferentes:

- **parcial -** as alterações são permitidas somente na versão atual do esquema, de acordo com as definições do usuário; é o tipo mais comumente utilizado;
- **total -** as alterações podem ser feitas sobre qualquer versão do esquema.

Em [MOR 99] são descritas algumas considerações sobre estas definições:

- versionamento de esquemas implica em evolução de esquemas, que implica em modificação de esquemas;
- as modificações não irão, necessariamente, resultar em uma nova versão. Uma nova versão deve ser ativada quando o usuário assim o desejar;
- as versões tendem a ser identificadas por um método definido pelo usuário enquanto as evoluções são identificadas pelo momento em que foram realizadas. Portanto, deve ser dada ao usuário a possibilidade de escolher entre a aplicação da evolução (adaptação da base de dados) ou do versionamento (criação de uma nova versão) para cada modificação sobre o esquema.

2.3 Considerações Finais

Neste capítulo foram apresentados alguns conceitos referentes à representação temporal, que surgiram da necessidade de representar a evolução temporal dos dados.

De acordo com o rótulo temporal utilizado, diferentes tipos de bancos de dados temporais foram identificados, sendo apresentadas suas características e formas de recuperar suas informações em relação ao tempo. A partir desses conceitos, observa-se que, associando o tempo de validade aos dados, é possível armazenar toda a história da evolução do BD. Por outro lado, a utilização do tempo de transação permite reconstruir o estado do BD em qualquer data passada, sem a necessidade de usar procedimentos de recuperação (*recovery* e *backup*), uma vez que todos os estados passados ficam disponíveis.

A distinção entre os termos *modificação*, *evolução* e *versionamento* foi também apresentada. Evolução de esquemas e versionamento de esquemas são duas técnicas utilizadas para modificar a estrutura do banco de dados, mantendo a consistência entre o esquema e a base de dados. Enquanto a evolução mantém apenas a versão atual do esquema e sua respectiva base de dados, o versionamento preserva todas as versões do esquema e seus dados associados durante a evolução.

O próximo capítulo apresenta um modelo que utiliza o tempo de validade para representar as informações temporais.

3 Modelo Temporal TRM

A modelagem de aspectos temporais é um importante tópico da modelagem de sistemas de informação. Através destes aspectos são representadas as características dinâmicas das aplicações e a interação temporal entre diferentes processos. A possibilidade de armazenar, manipular e recuperar dados temporais deve ser considerada na escolha de um método de modelagem [EDE 98].

Informações temporais tais como valores temporais, restrições temporais e características de evolução temporal, estão presentes em grande número de aplicações do mundo real. Deste modo, ao construir a especificação (modelo conceitual) de um sistema de informação, não só a estrutura dos dados manipulados deve ser definida, mas também sua dinâmica - seu comportamento com a passagem do tempo. Na coleta de requisitos dos sistemas devem ser identificados os requisitos temporais da aplicação em questão. O método utilizado na especificação do sistema de informação correspondente à aplicação deve permitir que todos os aspectos temporais sejam representados.

Nos modelos temporais, a variação do tempo é representada através de rótulos temporais associados aos dados, podendo ser:

- **ponto no tempo** – um único valor, representando o início da validade da informação;
- **intervalo temporal** – dois valores, identificando um intervalo, representando o tempo inicial e final da validade da informação;
- **elemento temporal** – conjuntos disjuntos de intervalos temporais.

Esses rótulos temporais podem ser adicionados às tuplas ou aos atributos.

Diversos modelos de dados temporais têm sido propostos na literatura, sendo a maioria constituída de extensões temporais aos modelos já existentes. Entre estes, podem ser encontrados modelos relacionais (HRDM [CLI 93], TRM [NAV 93]), entidade-relacionamento (TempER [ANT 97], ERT [LOU 91]) e orientados a objetos (TF-ORM [EDE 94], TOODM [LIN 93]). Alguns modelos associam aos dados apenas informações referentes ao tempo de transação, outros associam o tempo de validade e outros ainda associam tanto o tempo de transação quanto o tempo de validade.

Uma visão geral dos principais modelos de dados temporais existentes na literatura pode ser encontrada em [EDE 94, EDE 98, HÜB 2000, TAN 93].

Nos modelos temporais relacionais, a representação do tempo geralmente é feita através de atributos adicionais, correspondendo ao tempo de transação e/ou tempo de validade.

Os modelos de dados temporais Entidade-Relacionamento utilizam entidades, relacionamentos e atributos do modelo padrão (E-R), associando rótulos temporais a estas formas de representação.

Os modelos temporais orientados a objetos utilizam as primitivas básicas da orientação a objeto para manipular e armazenar informações. Trabalham com classes, atributos e métodos.

Entre as características existentes nos modelos de dados, a possibilidade de recuperar informações é fundamental. O tempo pode ser envolvido em consultas temporais de três formas diferentes [EDE 94]:

- para recuperar valores de propriedades cujo domínio é temporal;
- para se referir a um determinado instante ou intervalo temporal;
- para recuperar valores com base em restrições temporais.

Dentre as alternativas de modelos temporais, este trabalho baseia-se no modelo TRM (*Temporal Relational Model*) [NAV 93], que incorpora a semântica temporal do mundo real a um modelo de dados relacional. Juntamente com sua linguagem de consulta TSQL, o modelo permite a manipulação tanto de informações temporais como de informações não temporais, de forma coerente e consistente.

Combinando os recursos disponíveis no modelo TRM com informações adicionais relacionadas à versão do esquema, é possível não somente modelar uma aplicação considerando os aspectos temporais, como também implementar o versionamento de esquemas nesta aplicação.

Este capítulo apresenta as características do modelo TRM. Inicialmente são descritas a representação temporal e a linguagem de consulta do modelo. Complementando o capítulo, são identificadas as possíveis alterações que podem ocorrer na evolução de esquemas em modelos relacionais.

3.1 Características do Modelo TRM

Um banco de dados temporal é definido como a união de dois conjuntos de relações: as estáticas e as temporais. Neste modelo, toda relação temporal possui dois atributos de tempo obrigatórios: tempo de início (Ts) e tempo de fim (Te). Estes atributos correspondem aos limites inferior e superior de um intervalo de tempo. As relações temporais representam o tempo de validade. Por exemplo, o salário de um funcionário é válido durante o período compreendido entre Ts e Te.

Uma chave invariante no tempo (TIK) é utilizada como chave primária de uma versão estática de uma relação temporal. Um *timestamp* não pode ter diversos valores num determinado instante de tempo. Sendo assim, cada relação temporal possui duas chaves candidatas (TIK,Ts) e (TIK,Te).

Cada tupla deve conter um valor preciso para Ts. Porém, o valor de Te pode não ser conhecido no instante de criação da tupla. Neste caso, o valor é definido com o valor *default NOW* se Ts<=*NOW* e definido como *NULL* se Ts>*NOW*. No entanto, se um evento inicia no passado e seu término é desconhecido, assume-se que o valor de Te é válido somente até o presente (*NOW*), pois o uso de um valor infinito poderia ter a

significação errônea de que este valor é válido em qualquer tempo no futuro. Se o início de um evento ocorre no futuro e seu término é desconhecido, o mesmo assume o valor *NULL*. Um evento instantâneo pode ser modelado com Ts=Te. Dessa forma, a utilização de intervalos fechados permite modelar os conceitos de intervalo e de ponto no tempo.

Neste modelo, o tempo de validade constitui-se em uma parte integral das relações temporais. O tempo de transação e o tempo definido pelo usuário podem ser acrescentados através da definição de propriedades temporais, caso sejam necessários para alguma aplicação em particular.

3.2 Linguagem de Consulta do TRM

A linguagem de consultas do modelo TRM é a linguagem TSQL (**T**emporal **SQL**). Segundo [NAV 93], TSQL é um superconjunto da SQL que permite fazer consultas em BD temporais. Esta linguagem acrescenta novas construções sintáticas e semânticas ao SQL com a finalidade de garantir uma maior capacidade a este no que se refere à recuperação de informações temporais.

A TSQL possui as seguintes construções adicionais:

- expressões utilizando a cláusula *when*;
- recuperação de rótulos temporais;
- recuperação de informações ordenadas temporalmente;
- especificação do domínio de tempo utilizando a cláusula *time-slice*;
- funções de agregação e cláusula *group by* modificadas;
- especificação de comprimento de um intervalo temporal usando a cláusula *moving window*.

A seguir, a linguagem TSQL será apresentada através de exemplos, utilizando um banco de dados com o seguinte esquema conceitual:

EMPREGADO(MAT, NOME, ENDER, DT_NASC)
SALARIO(MAT, SALR, TS, TE)
GERENTE(MAT, MGR, TS, TE)
VIAGEM(MAT, CIDADE, PAIS, CUSTO, TS, TE)

Cláusula *WHEN*

A cláusula *WHEN* é similar à cláusula *WHERE* da SQL especificando condições temporais. Os seguintes operadores de comparação temporais são definidos: *BEFORE*, *AFTER*, *FOLLOWS*, *PRECEDS*, *OVERLAP*, *DURING*, *EQUIVALENT* e *ADJACENT*. Na cláusula *WHEN*, um ponto no tempo é considerado um intervalo degenerado; assim, todas as operações são definidas sobre intervalos.

Exemplo: Encontre o salário do empregado 125 quando Smith era seu gerente.

```
SELECT salr
FROM   SALARIO S, GERENTE G
WHERE  S.mat=G.mat AND G.mat=125
       AND G.mgr='Smith'
WHEN   S.INTERVAL FOLLOWS G.INTERVAL
```

Recuperação de Rótulos Temporais

Para a recuperação de *timestamps* a cláusula *SELECT* deve conter os operadores *TIME-START* (Ts) ou *TIME-END* (Te). Também é definido o operador *inter*, que produz a interseção de dois intervalos. Este operador pode ser utilizado na cláusula *SELECT*.

Exemplo: Encontre o gerente do funcionário 23 que sucedeu imediatamente o gerente Jones e encontre, também, o tempo de ocorrência deste evento.

```
SELECT B.mgr, B.TIME-START
FROM   GERENTE A, GERENTE B
WHERE  A.mat=B.mat AND A.mat=23
       AND A.mgr='Jones'
WHEN   B.INTERVAL FOLLOWS A.INTERVAL
```

Ordenação Temporal

As tuplas de uma relação temporal possuem uma chave temporal invariante, TIK, e um atributo temporal associado de forma unívoca (Ts ou Te). Assim, é possível classificar temporalmente todas as tuplas por um de seus rótulos temporais. Considerando que, para uma determinada TIK os períodos não se sobrepõem, existe uma ordenação total. Esta ordenação pode ter intervalos de descontinuidade quando, por exemplo, uma determinada propriedade não é válida durante um período. A referência a estes períodos disjuntos é realizada pelo uso das funções ordinais *FIRST*, *SECOND*, *THIRD*, *Nth* e *LAST*.

A ordenação temporal pode ser classificada de duas formas:

- na cláusula *where* ou *when*;
- na cláusula *select*.

Exemplo: Encontre o início do período e o salário para todos os empregados quando seus salários excederam 50K pela primeira vez.

```
SELECT FIRST (salr, TIME-START)
FROM   SALARIO
WHERE  salr>50K
```

Exemplo: Liste o salário do empregado 211 e a data em que ele foi recontratado pela segunda vez.

```
SELECT FIRST (salr,TIME-START)AFTER SECOND BREAK
FROM   SALARIO
WHERE  mat=211
```

Cláusula *TIME-SLICE*

Esta cláusula especifica um período ou ponto no tempo, significando que serão consideradas apenas as tuplas que estiverem contidas neste período. Consultas deste tipo devem incluir a palavra reservada *TIME-SLICE* e um indicador de período.

Exemplo: Liste todas as mudanças de salário durante os anos de 1972 a 1978 para todos os empregados cujo gerente foi Bradfort.

```
SELECT S.mat, salr, S.TIME-START
FROM   SALARIO S, GERENTE G
WHERE  S.mat=G.mat AND mgr='Bradfort'
WHEN   G.INTERVAL OVERLAP S.INTERVAL
TIME-SLICE year [1971,1978]
```

Agregação e *GROUP BY*

Em bancos de dados temporais somente são armazenados os instantes de início e de fim de validade de cada tupla. A duração do tempo decorrida entre os dois instantes é computada pela função *DURATION*. Sobre esta função podem ser utilizados os operadores de agregação (max, min, avg, sum, count). A cláusula *GROUP BY* é uma extensão da cláusula *GROUP BY* tradicional de SQL, que foi ampliada para permitir o emprego de rótulos temporais. Nos exemplos a seguir é considerada a composição dos atributos temporais como sendo *year, month, day*.

Exemplo: Encontre o período de tempo em que o empregado 45 trabalhou com o gerente Jones.

```
SELECT mat, SUM(DURATION)
FROM   GERENTE
WHERE  mgr='Jones' AND mat=45
```

Exemplo: Para cada empregado e para cada ano liste a cidade mais visitada naquele ano e o número de visitas.

```
SELECT mat, TIME-START.year, cidade, MAX(COUNT(*))
FROM   VIAGEM
GROUP BY mat, TIME-START.year, cidade
```

Janela Móvel

Neste modelo foi introduzido o conceito de janela móvel, que consiste na definição de um intervalo de tempo móvel especificado pela sua duração. Esta janela pode ser vista como a aplicação de uma função de agregação sobre as tuplas contidas no intervalo considerado. Esta funcionalidade pode ser utilizada, por exemplo, para a análise de falhas de equipamentos em um dado período, análise de séries temporais com a retirada de sazonalidade ou para o acompanhamento de recursos utilizados em projetos.

Exemplo: Encontre o período de dois anos em que o salário de um empregado teve a maior alta.

```
SELECT mat, MAX(LAST salr - FIRST salr), WINDOW
FROM   SALARIO
MOVING WINDOW 2 years
```

3.3 Evolução de Esquemas no Modelo Relacional

As possibilidades de evolução de um esquema conceitual e as conseqüentes alterações que devem ser efetuadas na extensão do banco de dados dependem do modelo de dados utilizado. No modelo relacional, o esquema conceitual contém as definições das relações (tabelas), que por sua vez contêm as definições dos atributos pertencentes a um domínio simples. Durante a evolução de um esquema relacional, várias modificações podem ser realizadas. A seguir estão identificadas as principais alterações.

Alterações sobre atributos:

- expansão no domínio;
- restrição no domínio;
- mudança no domínio.

Alterações sobre relações:

- acrescentar uma relação;
- desativar uma relação.

Alterações sobre atributo/relação:

- acrescentar um atributo a uma relação;
- renomear um atributo;
- desativar um atributo não chave;
- promover um atributo;
- rebaixar um atributo;
- dividir uma relação;
- particionar uma relação;
- juntar duas relações.

3.4 Considerações Finais

Inicialmente este capítulo apresentou as principais características dos modelos de dados temporais. Posteriormente foram apresentados detalhes próprios do modelo TRM, a forma de representação temporal no modelo e sua linguagem de consulta. Complementando o capítulo, foram identificadas as principais alterações que podem ocorrer num esquema relacional.

Neste trabalho, o TRM será utilizado como base para a implementação do banco de dados temporal. No capítulo seguinte são apresentadas as alternativas que devem ser levadas em consideração na implementação de BD temporais com versionamento de esquemas.

4 Alternativas para a Implementação de Bancos de Dados Temporais com Versionamento de Esquemas

Para manter todo o histórico de uma aplicação, o sistema de banco de dados deve armazenar, além da evolução dos dados, todas as versões antigas dos esquemas, permitindo que os dados antigos sejam recuperados de acordo com a versão do esquema correspondente. No entanto, o gerenciamento de várias versões do esquema, assegurando a consistência dos dados correspondentes, é uma tarefa complexa e requer uma definição precisa da forma como os esquemas e os dados evoluem.

Vários estudos têm sido desenvolvidos nos últimos anos, apresentando propostas que adicionam o versionamento de esquemas aos modelos de dados temporais [ANG 2000, CAS 95, EDE 95, MOR 99, ROD 99, WEI 99, WEI 2000]. Apesar disso, existem poucas experiências de implementação utilizando versões e bancos de dados temporais.

Como mencionado anteriormente, em [MOR 99] foi realizado um estudo sobre os tipos de versionamento de esquemas e dados, suas formas de armazenamento e gerenciamento. Esse estudo identificou 38 opções para a formação de um ambiente temporal que suporte o versionamento de esquemas e definiu uma estratégia para armazenar os dados da intenção.

Este capítulo apresenta as alternativas que permitem a implementação de bancos de dados temporais com o uso de versões do esquema.

As principais definições se referem à forma de armazenamento das versões do esquema e dos dados, e ao sincronismo entre esquemas e dados. Essas definições determinam a forma de gerenciamento do versionamento de esquemas em BDT.

Paralelamente à análise das alternativas, são escolhidas as opções utilizadas para realizar as implementações e apresentados os motivos que justificaram tais escolhas.

4.1 Armazenamento dos Dados (Extensão)

A primeira escolha a ser feita se refere ao tipo de BD temporal utilizado para armazenar os dados da extensão. Dependendo da necessidade da representação temporal da aplicação, diferentes tipos de BD temporais podem ser utilizados: BD instantâneo, de tempo de transação, de tempo de validade, bitemporal e multitemporal.

O modelo TRM, no qual este trabalho se baseia, utiliza o tempo de validade para representar as informações temporais. Em vista disso, escolheu-se um banco de dados de tempo de validade para armazenar os dados da extensão. Bancos de dados de tempo de validade representam melhor a semântica da realidade, pois associam a cada

informação o tempo em que a mesma é válida no mundo real. Com isso, alterações de informações válidas podem ser feitas para momentos passados, presentes, e futuros.

Outra escolha que deve ser feita é relativa à forma de armazenamento dos dados da extensão. As seguintes soluções são descritas em [CAS 95, EDE 98, MOR 99, WEI 99]:

- **um único repositório** (*single-pool*) - esta solução consiste de um repositório único no qual todos os dados da extensão são armazenados de acordo com um esquema global denominado de *esquema completado,* que inclui todos os atributos introduzidos pelas sucessivas mudanças de esquema. Os atributos indefinidos em alguma versão do esquema são substituídos por valores nulos. Na inclusão de um atributo ou expansão do domínio de um atributo, os dados correspondentes devem ser convertidos para o novo formato. Os atributos eliminados são mantidos no banco de dados, pois em bancos de dados temporais os dados nunca podem ser excluídos. Para situações que alterem o domínio de um atributo, tornando-o incompatível com o anterior, dois atributos devem ser mantidos, um para cada domínio;

- **múltiplos repositórios** (*multi-pool*) - nesta solução, a cada versão do esquema corresponde um repositório. Quando uma nova versão é criada, todos os dados são adaptados ao esquema criado e copiados para o novo repositório. Os dados inseridos depois da modificação devem ser armazenados no novo repositório.

A utilização de um único repositório minimiza o espaço necessário para o armazenamento dos dados da extensão, requerendo apenas que todo o repositório seja convertido para a forma aumentada sempre que novos atributos são adicionados. Na utilização de múltiplos repositórios, cada versão possui seu respectivo repositório; sendo assim, os mesmos dados podem ser armazenados várias vezes sob diferentes versões, gerando uma grande quantidade de dados redundantes.

Visando reduzir a quantidade de dados armazenados, poderia ser utilizada uma abordagem alternativa, combinando as duas soluções anteriores. Se houverem poucas alterações no esquema, originando versões com pequenas diferenças entre elas, os dados poderiam ser armazenados no mesmo repositório; a critério do DBA, em alguns casos um novo repositório poderia ser criado. Esta solução teria a vantagem de usar poucos repositórios, mas aumentaria a complexidade na resolução de consultas. Uma estratégia similar foi proposta em [WEI 2000] para banco de dados relacionais, denominada *partial multiple table version* (PMTV). Nesta solução, se a mudança de esquema for a inclusão de um atributo, é criada uma relação temporal contendo somente o novo atributo, mais o atributo chave da relação que está sendo modificada. Desta forma, não há inclusão de valores nulos nem há duplicação de dados.

Uma vez que o objetivo deste trabalho é realizar duas implementações, foram escolhidas duas alternativas para armazenar os dados da extensão: uma utilizando múltiplos repositórios e outra utilizando um único repositório. Estas escolhas determinam a forma de adaptação dos dados na criação de uma nova versão de esquema: usando múltiplos repositórios, cada alteração no esquema requer a criação de um novo repositório de dados; quando é utilizado um único repositório, sempre que o

esquema conceitual é modificado, os dados são adaptados ao novo formato, completando ou aumentando o repositório.

4.2 Armazenamento do Esquema Conceitual (Intenção)

Para implementar um banco de dados temporal que permita o versionamento de esquemas, é necessária a utilização de um BD também temporal para armazenar as diferentes versões do esquema, cada uma com seu rótulo temporal correspondente (tempo de transação e/ou tempo de validade), identificando o período no qual a versão estava ativa.

Uma alternativa para armazenar as várias versões do esquema foi proposta em [EDE 95, EDE 98]. Nesta proposta, as sucessivas versões são armazenadas em um meta-banco de dados temporal denominado Sistema de Banco de Dados Temporal Generalizado (SBDTG). Este sistema é composto de dois bancos de dados temporais, o primeiro para armazenar as versões do esquema (BD da intenção) e o segundo para armazenar os dados temporais (BD da extensão), cujo objetivo é manter informações temporais sobre as modificações realizadas nos esquemas e dados, mantendo, assim, todo o histórico da evolução.

Todos os estados dos dados são armazenados no BD da extensão, ficando disponíveis para as consultas do usuário. As diferentes versões do esquema também ficam disponíveis, armazenadas no BD da intenção, permitindo que as consultas sejam respondidas de acordo com o esquema válido no momento da definição dos dados. Esses dois bancos de dados evoluem de forma quase independente, relacionando-se apenas pela adaptação necessária dos dados da extensão cada vez que uma nova versão é criada.

A figura 4.1 apresenta uma idéia geral do comportamento de um SBDTG. Quando o banco de dados é criado, apresenta um esquema inicial e um correspondente estado inicial da extensão. Durante o período em que este esquema é válido, são feitas sucessivas atualizações nos dados, gerando novos estados do mesmo banco de dados. Uma modificação no esquema gera um novo esquema; conseqüentemente, um novo estado do banco de dados referente a este esquema deve ser construído.

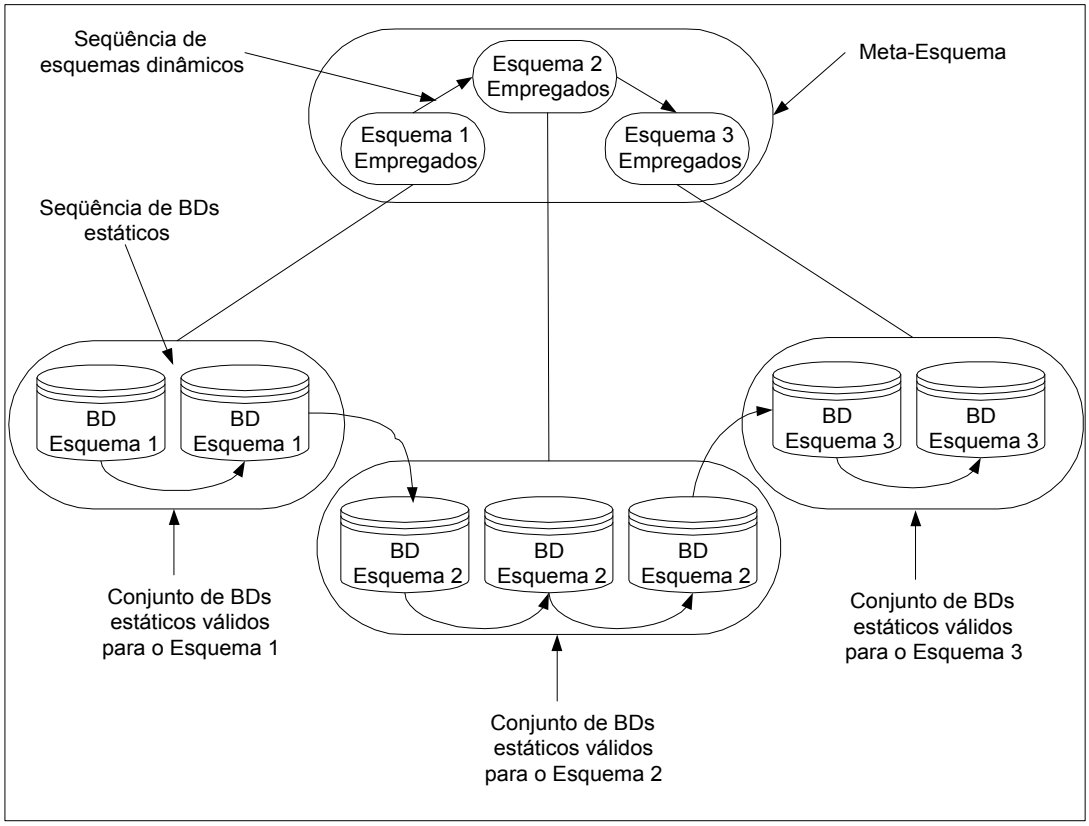

FIGURA 4.1 - Sistema de Banco de Dados Temporal Generalizado [ANG 2000]

Conforme a figura 4.1, todas as versões do esquema são armazenadas em um meta-banco de dados temporal. A cada modificação do esquema são acrescentadas informações temporais referentes à alteração. A forma utilizada para representar estas informações define o tipo de meta-banco de dados temporal que armazena a evolução.

De forma similar ao armazenamento dos dados da extensão, diferentes tipos de BD temporais podem ser utilizados para armazenar as versões do esquema. De acordo com o tipo de BDT selecionado, o versionamento de esquemas pode ser classificado em:

- **versionamento de esquema de tempo de transação** – neste tipo de versionamento, todas as sucessivas versões do esquema definidas ao longo do tempo ficam disponíveis para consultas, cada uma associada ao tempo de transação correspondente. Uma nova versão do esquema se torna válida no momento de sua definição, permanecendo válida até que uma nova versão seja definida. As modificações somente podem ser realizadas na versão corrente do esquema, não sendo permitidas mudanças retro ou proativas;
- **versionamento de esquema de tempo de validade** – cada versão do esquema é associada ao tempo de validade correspondente. Uma nova versão do esquema se

torna válida a cada vez em que um novo tempo de validade de uma alteração é alcançado. Este tipo de versionamento permite mudanças retro e proativas;

- **versionamento de esquema bitemporal** – neste tipo, cada versão do esquema é associada tanto ao tempo de transação quanto ao tempo de validade. O tempo de transação informa quando foi efetuada a alteração, enquanto que o tempo de validade define a partir de que instante essa alteração se tornará válida. Permite mudanças retro e proativas.

4.2.1 Armazenamento de Múltiplos Esquemas

Em [ANG 2000, MOR 99] foi proposta a criação de um meta-banco de dados para armazenar os múltiplos esquemas, definido através de quatro meta-relações contendo informações sobre as versões do esquema, suas tabelas, atributos e relacionamentos, servindo de base para o processo de versionamento. As seguintes meta-relações compõem o meta-banco proposto:

- **meta-relação das versões do esquema** – contém informações sobre as versões do esquemas, tais como identificador da versão, tempo de transação e/ou tempo de validade;
- **meta-relação das tabelas** – contém informações sobre as tabelas, tais como identificador da tabela, nome da tabela, localização física, formato temporal, tipo de temporalização, identificação da versão à qual a tabela pertence, tempo de transação e/ou tempo de validade;
- **meta-relação dos atributos** – contém informações sobre os atributos de cada tabela do esquema, tais como identificador do atributo, seu nome, tipo de dado, domínio, indicação do tipo de índice, a temporalização do atributo, identificadores da tabela e da versão à qual o atributo pertence, tempo de transação e/ou tempo de validade;
- **meta-relação dos relacionamentos -** contém informações sobre os relacionamentos existentes entre as tabelas de cada esquema, tais como identificador do relacionamento, tabelas e atributos que fazem parte do relacionamento, cardinalidade, identificador da versão à qual a tabela pertence, tempo de transação e/ou tempo de validade.

A estrutura do meta-esquema depende da solução escolhida para o armazenamento dos dados da extensão. Se a solução adotada for o armazenamento em múltiplos repositórios, a meta-relação das versões do esquema terá um atributo adicional para armazenar a identificação do repositório onde se encontram os dados correspondentes a uma determinada versão.

A partir das opções de versionamento apresentadas, escolheu-se neste trabalho o versionamento de tempo de transação, no qual as alterações somente são permitidas sobre a versão corrente do esquema (versionamento parcial), preservando, dessa forma, a evolução histórica da aplicação. Mudanças retro ou proativas sobre o esquema só são permitidas quando o versionamento de esquema de tempo de validade ou bitemporal é usado. Muitas pesquisas já publicadas analisam somente o versionamento de tempo de transação devido aos problemas que surgem quando se utiliza o versionamento de

tempo de validade, no qual uma única alteração pode afetar várias versões do esquema se o tempo de validade de uma nova versão se sobrepõe ao tempo de validade das versões já existentes. Além disso, não faz muito sentido modificar um esquema para tempos passados; isto representaria uma modificação retroativa no BD da extensão em relação ao esquema modificado, o que é pouco provável de acontecer na realidade e nem é significativo.

Quanto à forma utilizada para armazenar o BD da intenção, escolheu-se o SBDTG, pois possibilita a recuperação de informações de acordo com o esquema válido no momento considerado, possibilitando desta maneira, uma representação mais fiel da realidade.

As informações relativas às versões do esquema serão armazenadas usando as meta-relações descritas anteriormente.

4.3 Sincronismo entre Esquemas e Dados

Durante o processo de evolução de esquemas, é importante que a consistência entre o esquema conceitual e a base de dados seja mantida. Para que isto seja possível, é necessário utilizar mecanismos que gerenciem a interação entre o esquema e os respectivos dados armazenados. Existem duas estratégias que permitem este gerenciamento [CAS 95]:

- **gerenciamento síncrono -** neste tipo de gerenciamento, a pertinência temporal de uma versão do esquema deve incluir a pertinência temporal dos dados correspondentes, tornando-as dependentes. Neste caso, os dados são armazenados, recuperados e atualizados sempre de acordo com a versão do esquema que tenha a mesma pertinência temporal;
- **gerenciamento assíncrono -** a pertinência temporal de uma versão do esquema e a pertinência temporal dos dados correspondentes são completamente independentes. Os dados podem ser recuperados e atualizados através de qualquer versão do esquema, cuja pertinência é independente da pertinência dos dados, com a mesma dimensão temporal comum.

O gerenciamento entre os dados intencionais e extensionais é sempre síncrono em dimensões ortogonais, sempre síncrono com o tempo de transação e pode ser síncrono ou assíncrono com o tempo de validade.

Dentre essas opções, escolheu-se o gerenciamento síncrono, pois permite que os dados sejam recuperados e atualizados sempre de acordo com a versão do esquema correspondente.

4.4 Estratégia de Migração dos Dados

As modificações no esquema conceitual interferem na capacidade da base de dados em se adaptar ao novo esquema. Três estratégias são consideradas para gerenciar efetivamente a migração dos dados para o novo esquema [BOU 95]:

- **migração imediata -** todos os dados são convertidos imediatamente após a alteração do esquema;
- **migração adiada** - a conversão dos dados é atrasada até o término de todas as transações ativas;
- **migração oportuna** - permite o uso de múltiplas versões do esquema da base de dados, sendo ativada através de uma interface com o usuário.

Essas alternativas, quando aplicadas sobre um banco de dados temporal implicarão em perda ou alteração das informações passadas; os dados passados ficam disponíveis, mas precisam ser adaptados ao novo esquema. Como um dos principais conceitos de bancos de dados temporais é o de não perder informações passadas, a evolução do esquema deverá permitir a recuperação de informações passadas através do esquema vigente naquele momento. Sempre que ocorrer uma evolução do esquema conceitual, todas as versões deste esquema (a atual e as passadas) devem permanecer disponíveis na base de dados [EDE 98].

Existem duas formas diferentes de adaptar a base de dados, em decorrência de uma evolução de esquemas:

- ***off-line* -** todos os processos em andamento são interrompidos enquanto a base de dados é reorganizada. Portanto, a evolução deve ser executada em um período de baixa requisição de dados, pois os usuários não poderão acessar a base de dados;
- ***on-line*** - o SGBD é mantido em operação enquanto se realiza a evolução do esquema. É recomendada para sistemas críticos ou que envolvam grandes volumes de dados. Os processos de reorganização e de utilização são concorrentes, devendo ser controlados, permitindo que as requisições do usuário sejam atendidas enquanto uma parte do banco de dados é atualizada.

Neste trabalho optou-se em realizar a adaptação dos dados no momento em que a nova versão se tornar ativa, caracterizando a migração imediata dos dados. Para a conversão dos dados foi escolhida a forma *on-line*, na qual o SGBD permanece em operação, causando a mínima interferência possível nas aplicações dos usuários. Neste caso, os dados da versão corrente ficam disponíveis somente para leitura, sendo bloqueadas as operações de atualização. O acesso completo à nova versão só será permitido após o término do processo de conversão.

4.5 Gerenciamento do Versionamento de Esquemas e Dados

O gerenciamento das versões do esquema durante a fase operacional do sistema é um processo complicado. Nesta fase cada mudança no esquema deve considerar os dados previamente armazenados, levando ao problema de administrar diferentes versões do esquema e o relacionamento entre estas versões e seus dados correspondentes. Um sistema gerenciador de banco de dados temporal deve tratar o versionamento de esquemas e de dados de forma independente, executando operações que permitam gerenciar as versões de esquema, as transações e as consultas.

4.5.1 Gerenciamento das Versões do Esquema

Sempre que uma modificação é realizada no esquema (por exemplo, a definição de um novo atributo), uma nova versão do esquema é criada. Tais mudanças requerem o armazenamento dos esquemas antigos para garantir o acesso aos dados correspondentes.

No sistema de banco de dados temporal generalizado, bancos de dados temporais são utilizados para armazenar tanto as versões do esquema quanto os dados da extensão. Operações tradicionais (*insert*, *update*, *delete*, *select*) são usadas para gerenciar os dados extensionais. Porém, as operações executadas nos dados intencionais devem ser adaptadas ao gerenciamento das versões do esquema.

Quando uma nova versão de esquema é criada, várias modificações podem ser realizadas ao mesmo tempo definindo a nova versão. As modificações no esquema conceitual dependem do tipo de versionamento de esquema adotado. Se o versionamento de esquemas for parcial, todas as modificações são executadas sobre o esquema atual. No entanto, se for utilizado o versionamento total, não há a limitação de se modificar somente o último esquema; portanto, qualquer versão pode ser selecionada para gerar a nova versão.

Dependendo do tipo de banco de dados temporal utilizado para armazenar os dados intencionais, os rótulos temporais da nova versão podem ser: o tempo de transação, correspondendo ao tempo em que foi efetuada a alteração e/ou o tempo de validade informado pelo usuário, indicando o instante a partir do qual a alteração de esquema será válida. Esta validade deve ser restrita ao momento atual ou ao futuro, pois não faz sentido alterar um esquema para tempos passados. A definição de uma versão com validade no passado deveria ter sido usada para definir os dados naquele momento passado.

Quando uma nova versão é efetivada, é necessário adaptar as estruturas e os valores armazenados no BD da extensão, a fim de satisfazer a nova versão. Esta adaptação depende do tipo de armazenamento utilizado:

- se os dados extensionais são armazenados usando um único repositório, o esquema completado do BD da extensão deve ser atualizado para a nova versão de esquema, adicionando as novas informações definidas. Esta solução requer a utilização de valores nulos. Por exemplo, se um novo atributo é incluído, as tuplas de todas as versões do esquema adquirem o valor nulo para este atributo. Porém, nas tuplas correntes da versão, os valores nulos devem ser substituídos por valores significativos, enquanto que os valores nulos inseridos nas tuplas antigas não podem ser mudados, o que representa um desperdício de espaço de armazenamento;
- se o BD da extensão está armazenado em múltiplos repositórios, um novo repositório deve ser criado. Os dados devem ser copiados do repositório anterior para o novo repositório, formatados de acordo com a nova versão de esquema.

Se a extensão utilizar um BD de tempo de validade ou bitemporal, pode acontecer que uma determinada informação, inserida no BD sob uma determinada

versão do esquema, somente se torne válida sob outra versão. As adaptações feitas na extensão, cada vez que uma nova versão for definida, devem considerar esta possibilidade [MOR 99].

Um conceito importante na utilização de bancos de dados temporais é o de *vacuuming*, que eventualmente permite a eliminação física de dados temporais não relevantes, para os quais o custo de armazenamento é significativo. Este conceito pode ser estendido para o armazenamento de esquemas, permitindo eliminar versões, principalmente aquelas às quais não correspondem dados armazenados na extensão.

4.5.2 Gerenciamento das Transações

A definição dos dados deve ser feita de acordo com o tipo de banco de dados temporal usado para os dados da extensão, gerenciando automaticamente os rótulos temporais. Valores novos são sempre definidos considerando a versão corrente do esquema. Entretanto, o gerenciamento das transações depende também da configuração escolhida, principalmente da forma de armazenamento dos dados extensionais e da interação existente entre as versões do esquema e os dados.

Se a extensão utiliza um único repositório para armazenar os dados, a interação não influencia no gerenciamento da transação, ou seja, não há diferença entre o gerenciamento síncrono e assíncrono, pois os dados são definidos sob o esquema completado.

Se o BD da extensão é armazenado em múltiplos repositórios e o gerenciamento síncrono é usado, todo o gerenciamento de dados é feito somente no repositório correspondente ao esquema corrente.

Se o BD da extensão é armazenado em múltiplos repositórios com gerenciamento assíncrono, assim como qualquer versão de esquema pode ser usada na recuperação dos dados, qualquer definição de dados deve ser feita em todos os repositórios. Neste caso, o gerenciamento de dados opera simultaneamente em todos os repositórios de dados.

4.5.3 Gerenciamento de Consultas

A existência de várias versões do esquema deve ser levada em consideração nas operações de recuperação de informações. A resolução de consultas depende do tipo de sincronismo existente entre os esquemas e os dados.

No gerenciamento síncrono, a avaliação de uma consulta deve ser feita usando a versão do esquema válida na definição dos dados. Provavelmente várias versões do esquema serão usadas para solucionar a consulta, aumentando a complexidade do gerenciamento. Quando os dados são armazenados num único repositório, a escolha das respostas deve ser feita de acordo com a versão do esquema válida no momento da recuperação, mesmo que o esquema completado esteja sendo usado; a informação que não estiver presente na versão do esquema não deve ser informada. Quando são utilizados vários repositórios, cada repositório corresponde a uma versão do esquema;

portanto, a validade da versão do esquema define de qual repositório a resposta deve ser recuperada.

No gerenciamento assíncrono, qualquer versão do esquema pode ser usada para solucionar a consulta. Todos os valores dos dados, passados, presentes e futuros, ficam disponíveis e são analisados de acordo com a versão do esquema escolhida.

4.6 Considerações Finais

Este capítulo identificou as possíveis escolhas a serem feitas durante a implementação de bancos de dados temporais com versionamento de esquemas. As alternativas apresentadas incluem as várias possibilidades de gerenciamento da intenção e da extensão, tais como tipo de versionamento, forma de armazenamento dos esquemas, formas de armazenamento dos dados, interação entre esquemas e dados, estratégia de migração dos dados, gerenciamento do versionamento de esquemas e dados, incluindo o gerenciamento de transações e das consultas.

Observa-se que a definição das configurações depende das necessidades e restrições da aplicação. A escolha do tipo de versionamento empregado na intenção e na extensão depende das necessidades de armazenamento da história dos dados e dos esquemas. A escolha da solução usada para o armazenamento dos dados da extensão depende do espaço em disco disponível. A escolha do tipo de interação entre o versionamento dos dados e do esquema depende do nível de independência requerido.

Com base nas alternativas apresentadas, definiu-se uma proposta para implementar o gerenciamento temporal das versões de esquemas utilizando os principais conceitos que envolvem as áreas de versionamento de esquemas e bancos de dados temporais. As seguintes escolhas foram feitas:

- versionamento do esquema - tempo de transação;
- armazenamento do esquema - um repositório com múltiplos esquemas (SBDTG);
- tipo de versionamento de esquema - parcial;
- banco de dados da extensão - tempo de validade;
- armazenamento dos dados - uma alternativa usando múltiplos repositórios e outra usando um único repositório;
- interação entre esquemas e dados - síncrono;
- estratégia de migração dos dados - imediata;
- migração dos dados - *on-line*.

O capítulo seguinte descreve como são realizados os mapeamentos do modelo temporal relacional para um BD relacional e sua implementação em um BD comercial.

5 Implementação de um Banco de Dados Temporal com Versionamento de Esquemas

O objetivo deste capítulo é descrever como o modelo temporal relacional TRM foi mapeado para um banco de dados relacional e sua implementação em um banco de dados comercial. É importante ressaltar que a utilização de BD convencionais se deve à inexistência de um SGBD totalmente temporal. Neste caso, a implementação é realizada através do mapeamento adequado das informações temporais para atributos explícitos.

Na implementação de um BD temporal com suporte ao versionamento de esquemas, é necessário implementar no BD convencional, além do modelo conceitual, os recursos que permitam gerenciar as variações do esquema e sua sincronização com os dados.

O banco de dados comercial escolhido para a implementação do protótipo foi o DB2 *Universal Database*, desenvolvido pela IBM. Diversos fatores motivaram a escolha deste SGBD, entre os quais o fato de ser o mais próximo ao padrão SQL-92 no suporte aos tipos de dados temporais [SNO 2000]. Para isso, o DB2 possui os tipos *Date*, *Time* e *Timestamp*, juntamente com funções próprias de manipulação.

Na versão 5, o DB2 apresenta um conjunto de recursos que possibilitam implementar uma aplicação provendo um alto grau de segurança. Entre as principais funcionalidades, podemos destacar:

- integridade referencial (*referential-constraint*) – garante que os relacionamentos entre registros de tabelas relacionadas sejam válidos;
- *constraint* - permite definir restrições para os dados que são acrescentados às tabelas. As restrições são verificadas quando as linhas da tabela são inseridas ou atualizadas;
- tipos de dados definidos pelo usuário (*User-Defined Data Types*) – permite que o usuário defina os dados em função do negócio e do comportamento do banco de dados, em vez de na aplicação;
- funções definidas pelo usuário (*User-Defined Functions*)– permite criar funções específicas para o processamento dos dados;
- gatilho (*trigger*) – permite executar automaticamente uma função quando um valor é adicionado, excluído ou atualizado na base de dados;
- procedimento armazenado (*stored procedure*);
- tipo estruturado – possui suporte aos tipos estruturados.

Além disso, o DB2 permite definir um objeto chamado *schema* ao qual pode ser associado um conjunto de objetos do banco de dados (*alias,* tabelas, visões, índices, etc), fornecendo uma classificação lógica desses objetos. Desta forma é possível, por

exemplo, alterar o esquema de uma tabela sem perder o esquema anterior, associando cada versão da tabela a um *schema*. O DB2 também permite que o usuário escolha a forma como os repositórios de dados (*table spaces*) serão gerenciados: com baixo desempenho (SMS), onde o espaço é gerenciado pelo sistema operacional, ou com alto desempenho (DMS), gerenciado pelo SGBD. Com o DMS pode-se criar espaços separados para grandes dados e índices, permitindo gerar cópia de segurança, restaurar e definir desempenho para cada espaço separadamente.

5.1 Representação Temporal

Informações temporais foram associadas às versões, tabelas, atributos e relacionamentos. O tempo é considerado linear e a variação temporal é discreta. A temporalidade é representada através de intervalos temporais. Cada intervalo é representado por dois atributos de início e fim dos respectivos tempos (tempo de transação nos esquemas e tempo de validade nos dados).

A granularidade temporal adotada no modelo é o minuto.

O modelo conceitual da base de dados modelada com o TRM é armazenado em um meta-banco de dados definido através das meta-relações descritas no item 4.2.1, às quais foram acrescentadas novas informações, permitindo uma representação mais completa.

Visto que o modelo de versionamento adotado é de tempo de transação, os atributos correspondentes ao tempo de validade não são utilizados nas meta-relações. A estrutura das meta-relações está descrita a seguir.

Meta-relação das versões do esquema (Tabela **MRVersao**)

A meta-relação de versões (tabela 5.1) armazena informações sobre as versões do esquema. Para cada versão é definido um atributo identificador e um atributo estado, a partir do qual pode-se determinar quais operações poderão ser realizadas no gerenciamento das versões. Possui ainda atributos para identificar o local de armazenamento dos dados e os tempos de transação inicial e final. Os valores para o atributo estado de cada versão foram identificados como: ativa, inativa, em trabalho e bloqueada.

TABELA 5.1 – Tabela das versões do esquema

Atributo	**Tipo de dado**	**Descrição**
versão_id	num(4)	identificador da versão
estado	num(1)	indica o estado da versão domínio:ativa, inativa,em trabalho,bloqueada
local_da_tabela	char(18)	localização física da tabela
ttrans_ini	timestamp	tempo de transação inicial
ttrans_fim	timestamp	tempo de transação final

Meta-relação das tabelas (Tabela **MRTabela**)

A meta-relação de tabelas (tabela 5.2) armazena informações sobre todas as tabelas do esquema. Possui atributos para identificar cada tabela, seu nome, o formato temporal, e a temporalização dos dados. Também é definido um atributo para indicar à qual versão a tabela pertence e atributos que indicam os tempos de transação inicial e final.

TABELA 5.2 – Tabela das tabelas do esquema

Atributo	**Tipo de dado**	**Descrição**
tabela_id	num(3)	identificador da tabela
nome_da_tabela	char(18)	nome da tabela
formato_temporal	num(1)	indicação do formato temporal da tabela domínio: instantânea, tempo de transação, tempo de validade, bitemporal
tipo_temporal	num(1)	indicação da temporalização da tabela domínio: transitória,perene
versão_id	num(4)	identificador da versão
ttrans_ini	timestamp	tempo de transação inicial
ttrans_fim	timestamp	tempo de transação final

Meta-relação dos atributos (Tabela **MRAtrib**)

Esta meta-relação, descrita na tabela 5.3, armazena informações sobre os atributos (colunas) das tabelas: identificação do atributo, nome, tipo de dado, tamanho, tipo temporal para indicar se o atributo é temporal ou não, chave para indicar se o atributo faz parte da chave primária ou não. Também são definidos atributos para identificar a tabela e a versão à qual o atributo pertence e os tempos de transação inicial e final.

TABELA 5.3 – Tabela dos atributos do esquema (continua)

Atributo	**Tipo de dado**	**Descrição**
atributo_id	num(3)	identificador do atributo
nome_do_atributo	char(18)	nome do atributo
Tipo	num(2)	tipo de dado
Tamanho	num(8)	tamanho do atributo
Decimais	num(8)	número de casas decimais
valor_padrão	char(18)	indicação do valor padrão
Requerido	char(1)	indicação de atributo requerido domínio: sim, não
Índice	char(1)	indicação do tipo de índice domínio: sim, não, *unique*
Chave	char(1)	indicação de atributo chave domínio: sim, não
tipo_temporal	num(1)	indicação da temporalização do atributo: domínio: temporal, intemporal
tabela_id	num(3)	identificador da tabela

TABELA 5.4 – Tabela dos atributos do esquema (continuação)

versão_id	num(4)	identificador da versão
ttrans_ini	timestamp	tempo de transação inicial
ttrans_fim	timestamp	tempo de transação final

Meta-relação dos relacionamentos (Tabela **MRRelac**)

Esta meta-relação (tabela 5.4) armazena informações sobre os relacionamentos existentes entre as tabelas: a identificação e o nome do relacionamento, as tabelas e atributos que fazem parte do relacionamento, a versão à qual pertence e os tempos de transação inicial e final.

TABELA 5.4 – Tabela dos relacionamentos do esquema

Atributo	**Tipo de dado**	**Descrição**
RelacionamentoId	num(3)	identificador do relacionamento
nome_do_relacionamento	char(18)	nome do relacionamento
tabela1_id	num(3)	identificador da primeira tabela
atributo1_id	num(3)	identificador do atributo da 1ª tabela
tabela2_id	num(3)	identificador da segunda tabela
atributo1_id	num(3)	identificador do atributo da 2ª tabela
versão_id	num(4)	identificador da versão
ttrans_ini	timestamp	tempo de transação inicial
ttrans_fim	timestamp	tempo de transação final

Como mencionado anteriormente, o objetivo deste trabalho é realizar duas implementações, variando apenas a forma de armazenar os dados da extensão. Para permitir maior flexibilidade, foi definida uma tabela para armazenar os parâmetros da configuração que se deseja implementar, tais como o tipo de versionamento do esquema, o formato temporal e a forma de armazenamento dos dados da extensão. A definição dos parâmetros é feita no momento de criação da primeira versão do esquema. Assim, na criação de novas versões, não será preciso escolher os formatos temporais para os esquemas e dados, pois já estarão definidos na tabela de controle. A estrutura desta tabela está descrita a seguir.

Tabela de Controle (Tabela **Controle**)

A tabela de Controle é uma tabela instantânea (tabela 5.5) que armazena informações sobre a configuração escolhida para implementar. Possui atributos para indicar o tipo de versionamento do esquema, o formato temporal bem como a forma de armazenamento dos dados da extensão.

TABELA 5.5 – Tabela de Controle da Implementação (continua)

Atributo	Tipo de dado	Descrição
fmt_temporal_esq	Num(1)	identificador do tipo de versionamento do esquema domínio: tempo de transação, tempo de validade, bitemporal
fmt_temporal-ext	Num(1)	identificador do formato temporal dos dados domínio: tempo de transação, tempo de validade, bitemporal
armazenamento	Num(1)	identificador da forma de armazenamento dos dados. domínio: um repositório, múltiplos repositórios

5.2 Mapeamento dos Dados da Extensão

Em um modelo relacional, um banco de dados é considerado como sendo composto de um conjunto de relações (também chamadas de tabelas), cada uma contendo um conjunto de tuplas. Esse conjunto de tabelas e suas tuplas são uma instância do banco de dados. Nesta implementação o tempo é associado às tuplas, representando seu tempo de vida, e aos atributos, representando a variação de seus valores.

No modelo TRM os atributos de cada tabela podem ser definidos como **intemporais** ou **temporais**. Atributos intemporais não mudam de valor com o passar do tempo; para este tipo de atributo somente um valor pode ser definido, o qual permanece constante durante a existência de uma instância. Os atributos temporais podem apresentar muitos valores diferentes ao longo do tempo; porém, todas as alterações de valor devem ficar armazenadas permanentemente na base de dados, formando seu histórico.

A classificação dos atributos como temporais ou intemporais deve ser feita pelo usuário durante a especificação da aplicação.

Com base nessa classificação foi determinada a forma de mapeamento dos atributos.

Para os atributos classificados como intemporais é criada uma tabela denominada de tabela base, onde esses atributos são armazenados. A fim de facilitar a pesquisa aos valores atuais dos dados, que é bem mais comum do que a pesquisa histórica, os valores atuais dos atributos temporais também ficam incluídos nesta tabela. Para cada tupla da tabela base, são adicionados dois atributos representando seu tempo de vida (*lifespan*): *tvalid_ini*, indicando o instante em que uma tupla inicia sua existência, e *tvalid_fim*, que é atualizado somente quando uma tupla é excluída logicamente. Necessariamente deve ser definida uma chave primária para a tabela base.

Para cada atributo temporal é criada uma tabela denominada de tabela temporal, contendo a chave primária da tabela base, o atributo temporal em questão e os correspondentes rótulos temporais. Dessa forma, para todos os dados pertencentes às tabelas temporais são adicionados dois novos atributos: *tvalid_ini* e *tvalid_fim,*

respectivamente, tempo de validade inicial e tempo de validade final. O tempo de validade inicial representa o instante de início da validade da informação. Uma informação é considerada válida quando seu tempo de validade inicial é atingido e continuará neste estado até o início da validade de outro valor ou até atingir seu tempo de validade final.

O esquema relacional resultante do modelo é o seguinte:

```
Tabela_base ( Chave1,...,Chaven,
              AtrIntemp1,...,AtrIntempn,
              AtrTemp1,...,AtrTempn,
              tvalid_ini,tvalid_fim,
primary key ( Chave1,...Chaven));


Tabela_Temp1 ( Chave1,...,Chaven, AtrTemp1,
               tvalid_ini,tvalid_fim,
primary key  (Chave1,...Chaven, tvalid_ini));
…
…
Tabela_Tempn ( Chave1,...,Chaven, AtrTempn,
               tvalid_ini,tvalid_fim,
primary key  (Chave1,...Chaven, tvalid_ini));
```

onde:

$Chave_1,...,Chave_n$:	conjunto de atributos que fazem parte da chave primária da tabela base; devem necessariamente ser atributos invariantes no tempo;
$AtrIntemp_1,...,AtrIntemp_n$:	conjunto de atributos intemporais;
$AtrTemp_1,...,AtrTemp_n$:	conjunto de atributos temporais;
tvalid_ini:	atributo que representa o tempo de validade inicial;
tvalid_fim:	atributo que representa o tempo de validade final.

A chave primária na tabela temporal é composta pelos atributos $Chave_1,...,Chave_n$ mais o atributo *tvalid_ini*.

O relacionamento entre a tabela base e a tabela temporal é de 1:n.

5.2.1 Operações de Manutenção dos Dados da Extensão

Os bancos de dados convencionais gerenciam informações através das operações de inserção, atualização, exclusão e seleção. Nestes bancos de dados, as operações de atualização e exclusão podem ser realizadas sobre qualquer tupla, desde que o usuário tenha permissão para executá-las. Restrições de integridade mantêm a integridade do banco de dados.

Bancos de dados temporais utilizam as mesmas operações de gerenciamento, mas sua implementação não é exatamente a mesma. As operações de atualização devem garantir que nenhum dado que se torne antigo seja perdido; portanto, nenhuma informação pode ser substituída por outra, devendo ambas ficar armazenadas no BD.

A exclusão pode ser lógica e física. A exclusão lógica é realizada finalizando o tempo de vida da informação. A exclusão física pode eventualmente ser executada, removendo fisicamente a informação que se tornou irrelevante na aplicação. É uma operação raramente executada, somente realizada quando a redução do volume de dados armazenados é necessária e não é suportada pelo modelo de dados temporal. A exclusão física é realizada sob inteira responsabilidade do DBA, não sendo tratada pelos modelos temporais.

Hübler [HÜB 99] definiu um conjunto de regras que permitem gerenciar operações realizadas com os dados, levando em consideração cada um dos tipos de BD temporal e as possibilidades de variação temporal (pontos no tempo, intervalo temporal, elemento temporal).

Desse estudo foram consideradas as regras definidas para BD de tempo de validade com representação temporal na forma de intervalos, cujas operações estão descritas a seguir.

Inserção

A operação de inserção é a primeira definição dos valores para uma tupla.

O processo de inserção de tuplas nas tabelas transitórias é realizado em duas etapas:

1. inserção da nova tupla (todos os atributos) na tabela base;
2. inserção da nova tupla nas tabelas temporais - cada atributo temporal é inserido na sua respectiva tabela temporal.

De acordo com as definições do modelo TRM, o usuário deve fornecer o tempo de validade inicial e, opcionalmente, o tempo de validade final da tupla que está sendo inserida. O tempo de validade pode ser no passado, no presente ou no futuro.

Na seção 3.1 são definidos os procedimentos que devem ser adotados quando a validade final é desconhecida. Porém, para que esta operação se torne mais transparente ao usuário, caso não seja fornecida a validade inicial, a mesma assumirá o valor *now*. Se o usuário não fornecer o tempo de validade final, este será igual a *null* (valendo até que outra informação seja definida).

Na inclusão da nova tupla na tabela base, somente o tempo de validade inicial é inserido, ficando indefinido o tempo de validade final, pois seu tempo de exclusão é desconhecido neste momento.

Atualização

As operações de atualização resultam em novas inserções na base de dados, uma vez que as informações não podem ser substituídas por se tratar de um BD cujo

objetivo é manter o histórico das informações. Portanto, todas as operações de atualização são seguidas de operações para manutenção do valor antigo.

A cada novo valor de um atributo temporal, o tempo de validade da tupla atual deve ser encerrado e uma nova tupla é inserida na sua respectiva tabela, juntamente com o intervalo de tempo de validade correspondente. A tupla atual é aquela cujo tempo de validade está dentro do período referenciado ou seu tempo de validade final é *null*.

Na alteração do valor de um atributo intemporal na tabela base, o valor anterior é sobreposto, porém o tempo de validade inicial da tupla continua inalterado. É importante lembrar que os rótulos temporais da tabela base representam o tempo de "vida" da tupla toda.

Exclusão

A operação de exclusão é executada como uma exclusão lógica, encerrando-se o tempo de validade da informação. A exclusão física não é permitida.

Dessa forma, no processo de exclusão de uma tupla, o tempo de validade final é atualizado com o valor *now*, tanto na tabela base quanto nas tabelas temporais em que a tupla estiver válida. Conseqüentemente, a quantidade de dados é sempre crescente.

5.2.2 Recuperação de Informações em Bancos de Dados Temporais

A recuperação de informações é fundamental em bancos de dados. Cada modelo de dados deve apresentar uma linguagem de recuperação de informações associada às características do modelo.

A utilização de bancos de dados temporais permite consultar os dados armazenados e as informações temporais (*timestamps*) associadas aos mesmos. As consultas temporais que podem ser formuladas dependem do tipo de BD e da história considerada. Por exemplo, em bancos de dados de tempo de validade podem ser recuperadas informações válidas em momentos presentes, passados e futuros, de acordo com a atual percepção da história dos dados.

Segundo [EDE 98], uma consulta possui dois componentes ortogonais: um componente de seleção e outro de saída (projeção).

O **componente de seleção** geralmente é representado por uma condição lógica, que pode ser ou não temporal, envolvendo valores de dados e valores temporais associados aos dados (tempo de transação e/ou tempo de validade). Dependendo do componente de seleção, as consultas classificam-se em:

- **consulta de seleção sobre dados -** ocorre quando a condições são estabelecidas somente sobre os valores dos dados;
- **consulta de seleção temporal -** ocorre quando as condições de seleção são estabelecidas somente sobre as informações temporais associadas aos dados (tempo de transação e/ou de validade);
- **consulta de seleção mista -** ocorre quando a condição de seleção é aplicada sobre os dados e também sobre as informações temporais associadas a eles.

Com o **componente de saída** das consultas podem ser solicitados valores de dados bem como valores relativos às informações temporais associadas aos dados. De acordo com o componente de saída, as consultas podem ser:

- **consultas de saída de dados -** as informações selecionadas como resultado são exclusivamente os valores dos dados.
- **consultas de saída temporal -** são as consultas que requerem informações temporais associadas aos dados. Conforme a granularidade temporal do modelo podemos recuperar pontos no tempo, intervalos temporais e durações temporais.
- **consultas de saída mista -** recuperam simultaneamente valores de dados e valores temporais associados aos dados.

Nota-se que não é possível combinar elementos de seleção puramente temporal com saídas também puramente temporais; é necessário ter algum dado envolvido pelo menos em um dos componentes da consulta.

A linguagem de consulta do modelo TRM – TSQL – descrita na seção 3.2, baseia-se na linguagem SQL, apresentando extensões para suportar os aspectos temporais. A estrutura geral do comando SQL – *SELECT*, *FROM*, *WHERE* – é mantida: cláusula de especificação (determina os elementos que farão parte do resultado), cláusula de identificação (identifica os elementos sobre os quais será efetuada a busca) e cláusula de busca (apresenta as condições a serem satisfeitas pela consulta). Para suportar os aspectos temporais, a TSQL possui mais uma cláusula especial – WHEN – e um conjunto de operadores temporais que podem ser utilizados para recuperar informações temporais.

A fim de validar o protótipo implementado foi disponibilizado um módulo para realizar consultas simples, considerando as diversas formas de seleção e saída: somente dados, somente temporal ou mista – dados e tempo. A partir da interface do módulo de consultas, o usuário escolhe o tipo de seleção e saída desejada. As consultas são traduzidas diretamente para o comando SQL.

5.3 Mapeamento dos Componentes do Modelo para o DB2

Devido ao grande número de tabelas geradas, principalmente quando são utilizados vários repositórios para armazenar os dados, tornou-se necessário estabelecer uma forma de organizá-las, a fim de facilitar o gerenciamento das bases de dados.

Para isso, decidiu-se utilizar o objeto *schema* do DB2. O *schema* é definido como uma coleção de objetos identificados por um nome, com o objetivo de organizar de forma lógica os objetos do banco de dados. Quando se cria um objeto (por exemplo: *alias*, tabela, visão, índice, função, etc.), pode-se atribuí-lo a um esquema, especificando-se para isso o nome do mesmo. Os objetos criados dessa maneira passam a ser referenciados pelos seus nomes, precedidos do nome do esquema. Este nome é usado como a primeira de duas partes que identificam o objeto criado. Por exemplo, para referenciar a tabela `T` atribuída ao esquema `S`, usa-se `S.T`. Com essa sistemática, é possível criar objetos com nomes idênticos pertencendo a esquemas diferentes.

Essa estratégia foi usada para gerenciar os objetos do banco de dados temporal versionado (BDTV).

Todas as tabelas do BD intencional (BDI) são atribuídas a um esquema denominado SCHBDI. A mesma estratégia é usada para as tabelas do BD extensional (BDE), mas isso depende da forma de armazenamento dos dados:

- no armazenamento em um só repositório, todas as tabelas são atribuídas ao esquema denominado SBDE001, pois cada tabela possui somente uma versão ao longo de sua existência;
- no armazenamento em vários repositórios, para cada versão nova é criado um novo esquema ao qual são atribuídas todas as tabelas da nova versão. Com isso, as tabelas que- migram da versão anterior para a nova podem ficar com o mesmo nome que possuíam anteriormente, porém associadas a esquemas diferentes. Neste tipo de armazenamento, o nome do esquema é formado concatenando a constante 'SBDE' com o número da versão. Por exemplo, SBDE0001 é o identificador dos objetos da versão 0001.

Quanto ao mapeamento dos nomes de tabelas e dos repositórios de dados para o formato SQL , utilizou-se o padrão descrito a seguir.

Tabelas do meta-esquema

Cada tabela do meta-esquema é mapeada para uma tabela de mesmo nome no BD relacional, cujos atributos contém informações sobre a composição das tabelas, atributos e relacionamentos de cada versão.

Tabela base

Cada tabela base é mapeada para uma tabela com o mesmo nome da tabela original correspondente no modelo conceitual (meta-relação de tabelas).

Tabela temporal

Cada tabela temporal é mapeada para uma tabela cujo nome é formado concatenando a constante 'TAB' com o identificador da tabela original mais a constante '_ATR' e o identificador do atributo temporal. Por exemplo, TAB001_ATR001 indica a tabela temporal correspondente à tabela de código 001 para o atributo 001.

Repositório de dados para tabelas (*table space*)

O nome de cada *table space* é formado concatenando a constante 'TBS_BDE_V' com o identificador da versão. Porém, caso a opção de armazenamento seja em vários repositórios, cada vez que uma versão é incluída, um novo *table space* é criado. Usando um só repositório, o *table space* é criado apenas na definição da primeira versão, sendo usado para armazenar os dados de todas as versões.

Exemplo:

Consideremos a tabela FUNCIONARIO (Tabela_Id = 001), definida para a versão 00.01 com os seguintes atributos:

Nome do Atributo	**Atributo_Id**	**Tipo temporal**
Matricula	001	Intemporal
Nome	002	Temporal

Nascimento	003	Intemporal

A tabela FUNCIONARIO possui dois atributos intemporais e um atributo temporal, originando uma tabela base e uma tabela temporal. Considerando o armazenamento de dados usando vários repositórios, obtém-se a seguinte nomenclatura:

Tabela original:	FUNCIONARIO
Versão:	00.01
Nome da tabela base:	FUNCIONARIO
Nome da tabela temporal	TAB001_ATR002
Repositório para armazenar as tabelas da versão 0001:	TBS_BDE_0001
Identificador dos objetos da versão 0001:	SBDE_0001

Essa padronização foi adotada para adequar a formação de nomes ao padrão SQL do DB2, que admite, no máximo, 18 caracteres para identificar tabelas, esquemas, visões, índices, etc.

5.4 Gerenciamento da Evolução de Esquemas

Cada operação de modificação realizada no esquema requer a criação de uma nova versão. Tais modificações resultam no armazenamento de informações temporais no BD da intenção, a fim de registrar a evolução ocorrida.

A criação de novas versões é uma operação restrita ao administrador do banco de dados (DBA) e deve ser executada de forma tão transparente quanto possível aos usuários da aplicação.

Cada versão de esquema possui um *estado* associado indicando o estágio em que a mesma se encontra, podendo ser:

- **em trabalho** – estado que uma versão recebe ao ser criada, indicando que a mesma está em processo de definição, podendo sofrer várias alterações até ser efetivada (ativada). Neste estágio, como ainda não existem dados na extensão, a versão é considerada temporária podendo ser removida fisicamente, mas não pode ser usada para criar uma nova versão;
- **ativa** - este estágio é alcançado quando o usuário decide efetivar uma versão que está *em trabalho*; automaticamente seu estado muda para *ativa,* passando a ser a versão atual. Uma versão *ativa* não pode mais ser alterada nem removida, podendo apenas ser consultada, utilizada para criar uma nova versão ou ser arquivada. Somente uma versão deve estar *ativa* na base de dados;
- **inativa** - este estágio indica que uma versão atingiu seu estado definitivo, não podendo mais ser alterada ou removida, nem usada para definir uma nova; uma versão *inativa* somente pode ser consultada. Esta situação ocorre quando uma versão *em trabalho* é ativada pelo usuário; após a adaptação e conversão dos dados, a

versão que serviu de base para a evolução é arquivada, mudando seu estado para *inativa*;

- **bloqueada** – este estágio indica que uma versão está temporariamente bloqueada para alterações, podendo somente ser consultada. Quando uma nova versão é ativada, os dados da extensão precisam ser convertidos para correspoder à nova versão. Durante este processo, a versão *ativa* muda seu estado para *bloqueada*, permanecendo assim até o término da conversão.

5.4.1 Gerenciamento da Intenção

O gerenciamento da evolução de esquemas é definido através de três operações básicas: *Definir Versão*, *Ativar Versão* e *Remover Versão.* Estas operações estão descritas a seguir.

Definir Versão

É a operação que cria uma nova versão do esquema. Considerando que foi escolhido para implementar o versionamento parcial, todas as modificações são realizadas sobre a versão atual (última versão criada).

Toda versão nova sempre é definida com o estado *em trabalho*. Neste estado, a versão não possui tempos associados nem dados definidos na extensão.

O usuário (DBA) pode efetuar várias modificações ao mesmo tempo, definindo a nova versão; a versão atual, porém, continuará sendo válida até que o DBA decida ativar a nova versão.

As operações de modificação na versão atual do esquema resultam na atualização do banco de dados da intenção (BDI) para incluir, nas meta-relações, a nova versão, suas tabelas, atributos, e relacionamentos.

A definição de uma versão nova pressupõe a existência de uma versão anterior. Caso a mesma não tenha sido criada, um procedimento especial é executado para criar a primeira versão. Neste caso, é realizada uma definição completa da versão, incluindo:

- definição dos parâmetros de controle da configuração;
- atualização do banco de dados da intenção (BDI) para incluir o esquema conceitual;
- criação do banco de dados da extensão (BDE);
- inclusão dos dados no BDE.

Ativar versão

Esta operação ativa a nova versão que acabou de ser definida sobre o esquema atual. Uma versão é sempre ativada a partir da versão *em trabalho*.

Quando uma versão é ativada, os dados extensionais, inclusive os dados definidos com validade no futuro, são adaptados à nova versão. Durante este processo, a versão atual fica no estado *bloqueada*, impedindo que os dados sejam modificados pelos

usuários da aplicação, permanecendo neste estado até que termine a adaptação. Ao final deste processo, é encerrado o tempo de vida da versão que serviu de base para as modificações, isto é, o tempo de transação final é atualizado para *now* e seu estado muda para *inativa.* A nova versão passa do estado *em trabalho* para *ativa* e seu tempo de início de vida (tempo de transação inicial) é incluído.

Remover versão

Esta operação exclui uma versão do esquema, removendo fisicamente do BDI, as tabelas, atributos e relacionamentos correspondentes. Somente a versão que está *em trabalho* pode ser removida, pois a mesma ainda não possui dados definidos na extensão.

De acordo com o conceito de banco de dados temporais, nenhuma informação pode ser excluída fisicamente, pois todo o histórico dos dados deve ser preservado; sendo assim, somente a exclusão lógica é permitida, finalizando a validade da informação. Portanto, a exclusão de um esquema também não deveria ser permitida. Porém, como a versão *em trabalho* não foi ainda utilizada para definir dados, a mesma pode ser removida fisicamente.

5.4.2 Gerenciamento da Extensão

A criação de uma nova versão de esquema requer que os dados da extensão sejam atualizados para se adaptarem à nova versão. A forma de adaptação depende da solução usada para o armazenamento dos dados, sendo realizada conforme a descrição a seguir:

- solução com repositório único - o esquema completado do BD extensional é atualizado para a nova versão do esquema, adicionando as novas informações definidas. Se a mudança consiste na exclusão de um atributo ou na restrição de um domínio, a mesma não altera o esquema completado nem modifica fisicamente o repositório; somente os dados intencionais são afetados pela mudança. Os atributos excluídos não são removidos fisicamente. Por outro lado, se uma mudança adiciona um atributo ou aumenta o domínio, todas as tuplas do repositório são convertidas para o formato aumentado. O valor especial *null* é definido como os valores dos atributos incluídos. Se a mudança se refere à alteração de um domínio, o atributo é estendido para o maior tamanho definido até então. Se a mudança de domínio produz um novo domínio incompatível com o anterior, dois atributos são mantidos correspondendo a domínios diferentes e pertencendo a diferentes versões do esquema;
- solução com múltiplos repositórios - um novo repositório de dados é criado de acordo com a estrutura do novo esquema, contendo apenas os atributos definidos na nova versão. Os dados do repositório anterior são convertidos ao formato do novo esquema. As tuplas atualmente válidas e as tuplas com validade no futuro são copiadas para o novo repositório.

5.5 Descrição da Implementação

Utilizando os conceitos e técnicas apresentados anteriormente, foi desenvolvido um protótipo com o objetivo de mapear o modelo TRM para o DB2, permitindo o versionamento de esquemas. Através de uma interface gráfica, a ferramenta auxilia o usuário a especificar os componentes (meta-dados) do modelo, gerenciando a criação de novas versões do esquema conceitual.

Inicialmente o protótipo foi definido levando em consideração a proposta apresentada no capítulo 4: versionamento de esquemas de tempo de transação, dados de tempo de validade, armazenamento dos dados da extensão tanto em um repositório quanto em vários repositórios e gerenciamento síncrono.

Posteriormente verificou-se que, com a criação de regras para conversão/adaptação dos dados, essas definições poderiam ser ampliadas para suportar outras alternativas de implementação, de forma que o usuário pudesse escolher a configuração a ser implementada de acordo com as necessidades de sua aplicação. Com isso, a ferramenta se tornaria um gerenciador completo, capaz de implementar o versionamento de esquemas em BD temporais para um conjunto amplo de configurações. Porém, em virtude da complexidade que um trabalho tão abrangente apresenta, não foi possível implementar todas as alternativas. Além do mais, o objetivo era comparar uma implementação usando somente um repositório, com outra usando vários repositórios. Em vista disso, a complementação da ferramenta foi deixada para um trabalho futuro. A versão atual possui suporte às seguintes opções de configuração:

- versionamento de esquemas - somente tempo de transação;
- temporalidade dos dados - tempo de transação, tempo de validade e bitemporal;
- tipo de armazenamento - um repositório e múltiplos repositórios;
- gerenciamento - somente síncrono.

Com relação às operações de alteração no esquema, selecionamos as seguintes para implementar: inclusão e exclusão de tabela, inclusão e exclusão de atributo, expansão de tamanho do atributo e mudança de tipo (domínio) para tipo compatível. Essas operações foram escolhidas por serem as que mais afetam o espaço de armazenamento e a estrutura das tabelas. A mudança para um tipo incompatível impossibilita a adaptação dos dados da versão antiga para a nova, sendo necessário inseri-los novamente com o formato atual. Como o objetivo é manter as informações passadas, alterações que causam a perda de dados não são permitidas neste trabalho.

A figura 5.1 apresenta a arquitetura da implementação. O gerenciamento da evolução de esquemas é definido através de uma camada intermediária que controla as aplicações dos usuários e o banco de dados.

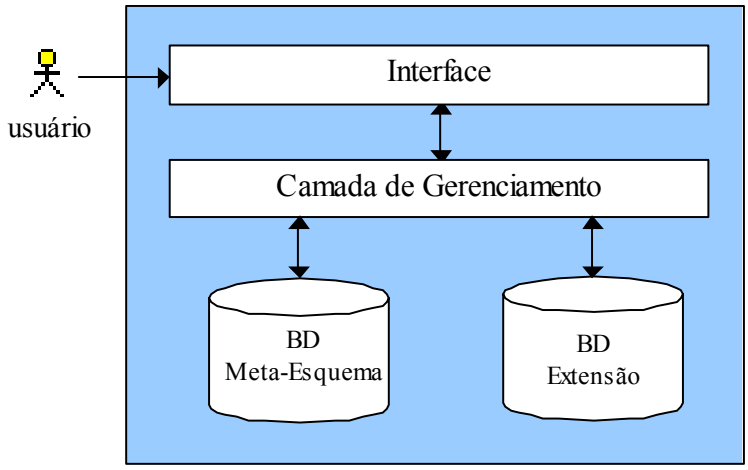

FIGURA 5.1 – Visão Geral da Implementação

Através da interface da ferramenta o usuário interage com o sistema definindo o primeiro esquema da sua aplicação e criando novas versões. Estas definições são armazenadas no meta-esquema. A camada intermediária realiza o mapeamento dos meta-dados para tabelas equivalentes no DB2, provendo os recursos que permitem controlar as versões e seus dados correspondentes.

As figuras 5.2 e 5.3 apresentam o mecanismo de gerenciamento em cada alternativa implementada.

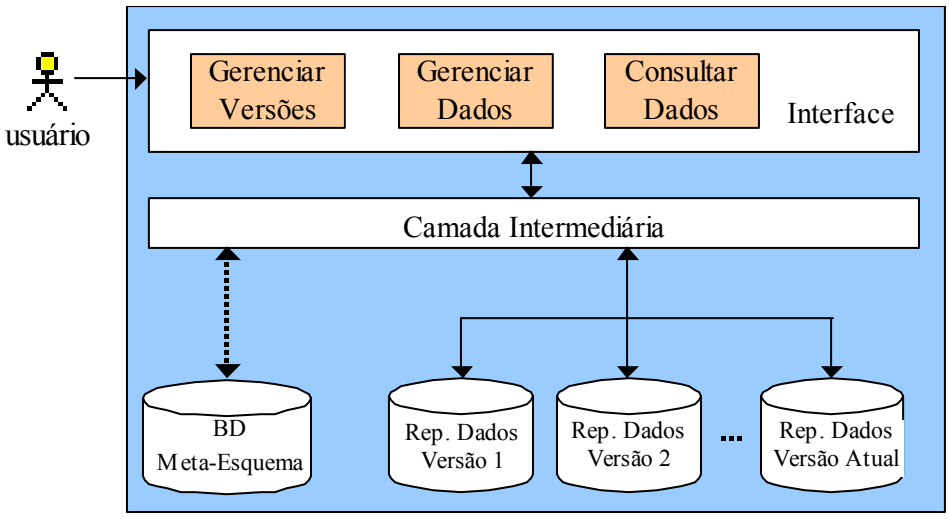

FIGURA 5.2 – Alternativa de Implementação Usando Vários Repositórios

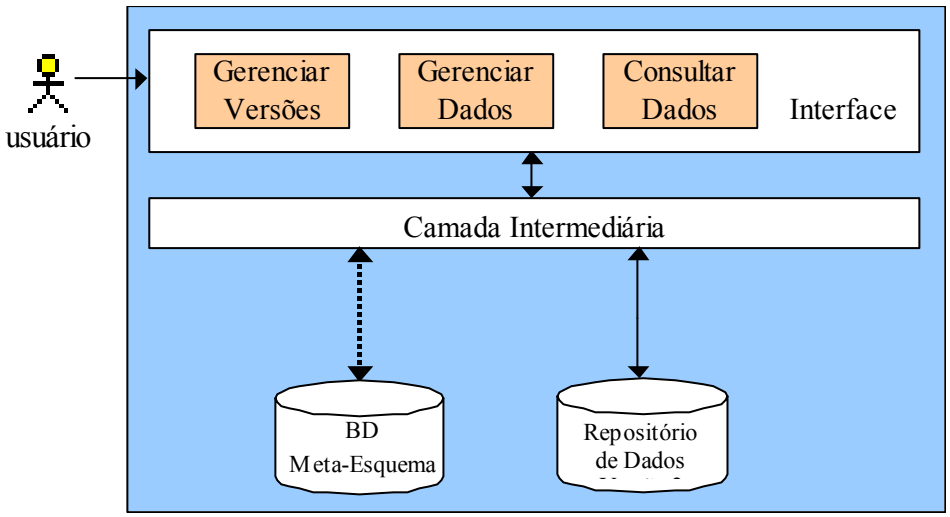

FIGURA 5.3 – Alternativa de Implementação Usando Apenas um Repositório

A ferramenta foi construída em Delphi utilizando o protocolo ODBC para conexão com o DB2.

A fim de permitir o funcionamento da ferramenta, a estrutura do meta-esquema foi criada previamente através de *scripts* executados no DB2. As tabelas do BD da extensão são criadas dinamicamente com a execução das operações de versionamento.

A interface da ferramenta é composta de quatro páginas: Versões, Tabelas, Relacionamentos e Consultas. Através delas o usuário interage com o sistema definindo novas versões do esquema ou realizando consultas.

As operações que controlam o versionamento de esquemas são disponibilizadas na página de **Versões** (figura 5.4).

No painel superior da janela de versões o usuário especifica a configuração do ambiente temporal a ser implementado, incluindo a temporalidade do esquema, a temporalidade dos dados da extensão e o tipo de armazenamento. Um nome simbólico deve ser definido para o esquema da aplicação. Para criar as versões do esquema é necessário que estas informações estejam armazenadas na base de dados. Portanto, ao definir a primeira versão, o usuário deve selecionar a configuração que deseja implementar. Esta configuração será válida para todas as versões que forem criadas. Se o usuário não escolher as opções, serão considerados os valores *default*: tempo de transação para as versões de esquema, tempo de validade para os dados da extensão e armazenamento dos dados em múltiplos repositórios. Uma vez definida a configuração, o processo de criação das versões pode ser iniciado pelo usuário. É importante destacar que os parâmetros da configuração não podem mais ser alterados depois que o esquema da aplicação estiver definido. Ou seja, depois que o esquema inicial é criado, não é possível definir uma versão nova e ao mesmo tempo mudar a temporalidade dos dados ou mudar o tipo de armazenamento, por exemplo. Só é permitido modificar o esquema

conceitual. Quando o usuário decide criar novas versões, o programa faz a adaptação dos dados levando em consideração o que foi definido nesta etapa inicial.

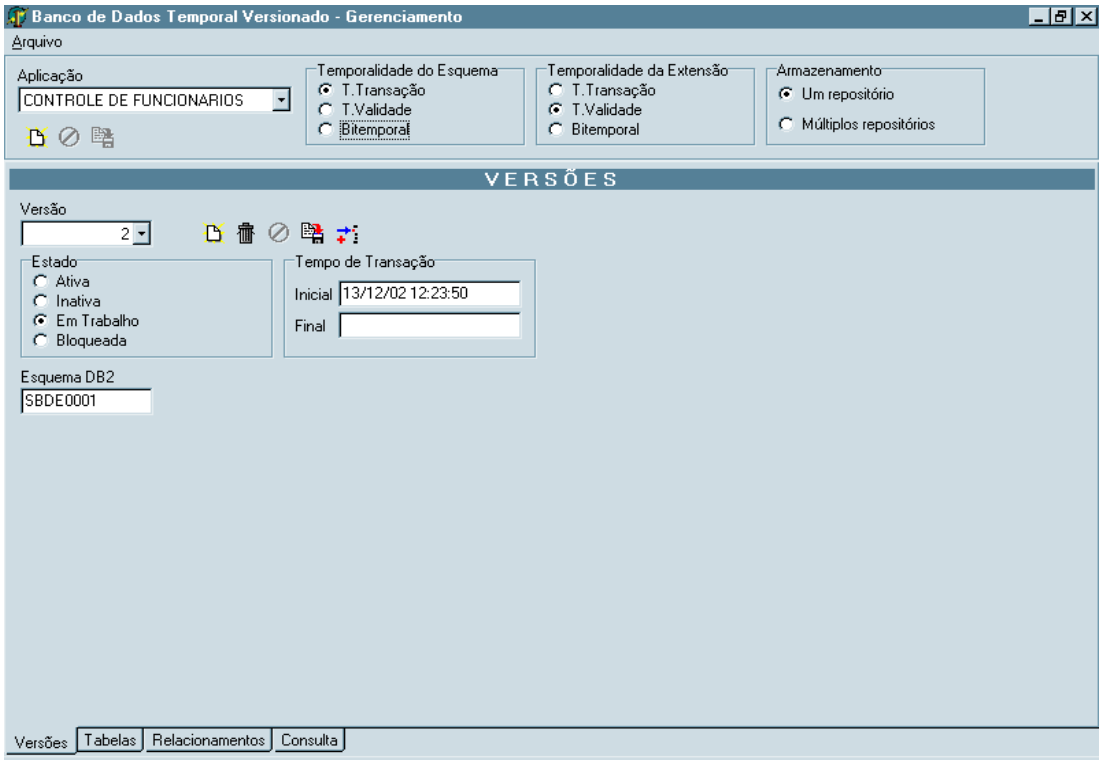

FIGURA 5.4 – Interface para Manutenção das Versões

No painel inferior da figura 5.4 o usuário pode visualizar informações sobre determinada versão ou selecionar as operações de gerenciamento das versões do esquema. O versionamento é realizado a partir da execução das operações **Definir Versão** e **Ativar Versão**. A primeira é responsável por definir os dados do BD intencional enquanto que a segunda é responsável por mapear os meta-dados para o DB2 bem como por adaptar os dados da extensão. A nova versão sempre é definida tendo por base a versão atual. Se necessário, o usuário pode remover a versão que estiver sendo definida através da operação **Excluir Versão.** Os detalhes de cada operação estão descritos a seguir.

Inclusão de Versão

Para gerar uma nova versão o usuário seleciona a operação *Definir Versão*, cuja função é cadastrar o esquema conceitual no meta-esquema. Ao executar esta operação, uma nova instância do esquema é inserida na meta-relação de versões. O número da versão é atribuído automaticamente pelo programa, sendo representado através de números inteiros gerados seqüencialmente. O atributo *estado* recebe o valor *em trabalho*. Com o objetivo de simplificar o trabalho do usuário durante a definição de versões, os meta-dados da versão atual (tabelas, atributos e relacionamentos) são copiados para a nova versão, exceto na criação do primeiro esquema da aplicação. Com

isso, somente as novas definições precisam ser inseridas no BD intencional. Os valores que foram copiados devem ser alterados para ficar de acordo com a nova versão.

Após cadastrar a nova versão, o usuário pode então executar as operações de manutenção dos meta-dados através das páginas **Tabelas** e **Relacionamentos**. Na interface mostrada na figura 5.5 o usuário especifica os detalhes das tabelas e de seus atributos (colunas). Para cada tabela são apresentados os campos: nome da tabela, formato temporal, tipo temporal. Para cada atributo são apresentadas as propriedades descritas na meta-relação de atributos, tais como nome do atributo, tipo de dado, tamanho, decimais, tipo temporal. Os rótulos temporais dos meta-dados são definidos pelo sistema e não podem ser modificados pelo usuário.

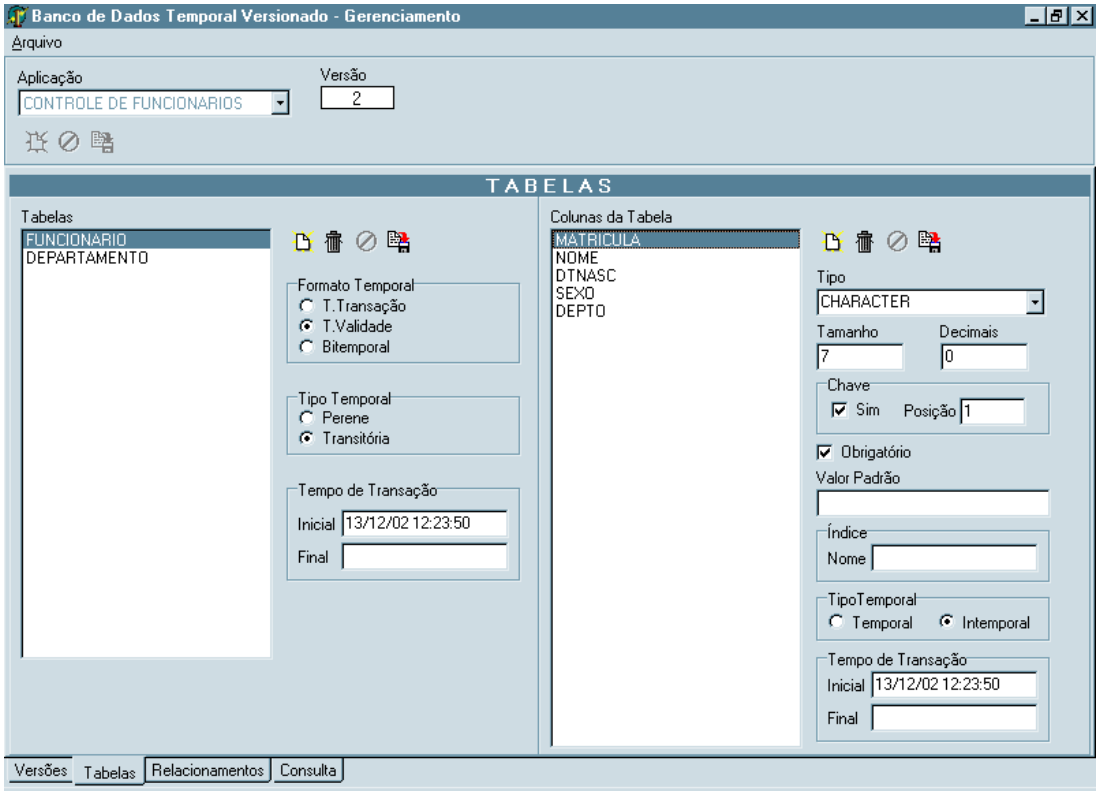

FIGURA 5.5 – Interface para Especificação de Tabelas e Atributos

Os relacionamentos são especificados na interface mostrada na figura 5.6. São apresentados os campos: nome do relacionamento, nomes das tabelas e atributos que participam do relacionamento.

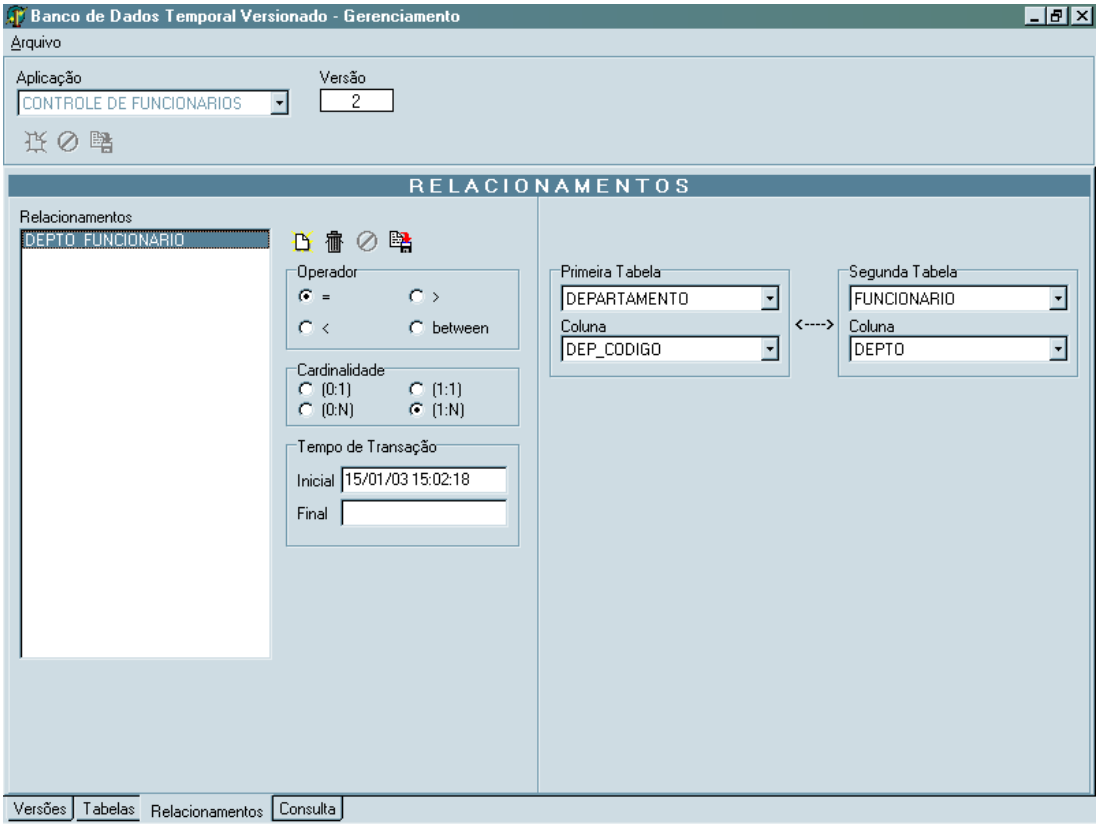

FIGURA 5.6 – Interface para Especificação de Relacionamentos

Ativação de Versão

Após definir os meta-dados o usuário pode ativar a nova versão do esquema escolhendo a operação *Ativar Versão*. Inicialmente o programa verifica se a versão selecionada está *em trabalho*; caso o estado seja outro, a ativação não será permitida. Dando prosseguimento à ativação, o programa recupera os meta-dados, verifica se as restrições de integridade estão mantidas para então iniciar o mapeamento do esquema conceitual para o DB2, gerando ou atualizando o banco de dados da extensão (BDE). Nesta fase, o gerenciamento do BDE depende da opção escolhida para o armazenamento.

Na implementação que utiliza apenas um repositório, todo o repositório é convertido para o formato aumentado para corresponder às definições do novo esquema. Para cada tabela nova são criadas a tabela base e as tabelas temporais correspondentes. As tabelas que foram modificadas são atualizadas para se adaptarem à nova versão. Dependendo do tipo de alteração, pode ser necessário redefinir o esquema das tabelas do BDE. Por exemplo, a exclusão de um atributo não causa alteração no esquema da tabela, pois apenas a exclusão lógica é permitida; porém a mudança de tipo (domínio), a alteração no tamanho ou a inclusão de um atributo requer a redefinição do esquema da tabela com a adaptação dos dados ao novo formato. Se o atributo é temporal, a tabela temporal correspondente também é atualizada. As alterações são realizadas diretamente

sobre a tabela existente, a qual contém todos os atributos já definidos. Desta forma, o repositório de dados sempre é formatado de acordo com a última versão.

Na alternativa de implementação que utiliza vários repositórios, o gerenciamento consiste de duas etapas: a criação do novo repositório de dados e o arquivamento do repositório anterior. Na primeira etapa, cada tabela definida na nova versão é mapeada para o SGBD, sendo geradas novas tabelas base e temporais. Após a criação do novo repositório, as tuplas do repositório anterior que estão atualmente válidas e as com validade no futuro são adaptadas ao novo esquema e copiadas para o novo repositório evitando, assim, a replicação de dados antigos. Na segunda etapa, as tuplas do repositório anterior são arquivadas implicitamente através do arquivamento da versão do esquema correspondente: o tempo de transação final dos meta-dados é atualizado para o tempo *now* da mudança de esquema.

Concluída a adaptação dos dados, o meta-esquema é atualizado: o tempo de transação final da versão anterior é atualizado para *now* e seu estado muda para *inativa* enquanto a versão nova passa do estado *em trabalho* para *ativa* e seu tempo de transação inicial é incluído.

Exclusão de Versão

A remoção de versões é realizada através da operação *Excluir Versão*. O processo de exclusão é bem simples: o usuário seleciona a versão para excluir; se a mesma estiver *em trabalho*, os meta-dados são removidos; caso contrário, a exclusão é cancelada. Nenhuma alteração é realizada no BD da extensão pois os dados ainda não foram definidos para esta versão.

5.6 Estudo de Caso

Esta seção apresenta o estudo de caso definido com o objetivo de avaliar a implementação realizada neste trabalho. Trata-se de uma aplicação bem simples, mas que apresenta as características temporais necessárias para permitir a exemplificação do versionamento de esquemas. A figura 5.7 apresenta o diagrama E-R da aplicação.

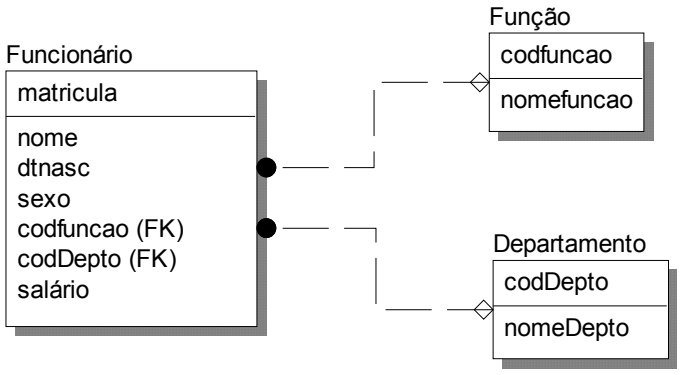

FIGURA 5.7 – Diagrama E-R do Estudo de Caso

5.6.1 Descrição da Aplicação

O estudo de caso consiste na modelagem de uma aplicação para o Controle de Funcionários de uma empresa composta de departamentos e de um conjunto de funcionários. O salário e o departamento do funcionário podem variar com a passagem do tempo enquanto seu sexo e sua data de nascimento não mudam. O nome do funcionário também pode mudar em conseqüência de casamento ou divórcio. A função também pode variar com o passar do tempo, pois o funcionário pode ser promovido para exercer uma nova função na empresa.

5.6.2 Mapeamento para o DB2

O esquema conceitual foi definido com as tabelas FUNCIONÁRIO (Id 001), DEPARTAMENTO (Id 002) e FUNÇÃO (Id 003). Porém, para demonstrar o processo de mapeamento, será usada a tabela Funcionário que possui vários atributos temporais.

As tabelas resultantes do mapeamento são apresentadas na figura 5.8. As colunas em destaque na tabela base correspondem aos atributos temporais.

Tabela Conceitual

FUNCIONARIO (matricula(pk), nome, dtnasc, sexo, depto, função, salario)

MAPEAMENTO

Tabela Base

Funcionario								
matricula	**nome**	**dtnasc**	**sexo**	**depto**	**função**	**salario**	**tvalid_ini**	**tvalid_fim**
0323576	Ana Jacob	12/03/1960	F	02101	035	1700,00	15/10/2002	30/12/2002
0267345	Paulo Pires	15/06/1965	M	02003	027	1100,00	15/10/2002	30/12/2002

Tabelas Temporais

Tab001_Atr002			
matricula	**nome**	**tvalid_ini**	**tvalid_fim**
0323576	Ana Jacob	15/10/2002	30/12/2002
0267345	Paulo Pires	15/10/2002	30/12/2002

Tab001_Atr005			
matricula	**depto**	**tvalid_ini**	**tvalid_fim**
0323576	02101	15/10/2002	30/12/2002
0267345	02003	15/10/2002	30/12/2002

Tab001_Atr006			
matricula	**funcao**	**tvalid_ini**	**tvalid_fim**
0323576	035	15/10/2002	30/12/2002
0267345	027	15/10/2002	30/12/2002

Tab001_Atr007			
matricula	**salario**	**tvalid_ini**	**tvalid_fim**
0323576	1700,00	15/10/2002	30/12/2002
0267345	1100,00	15/10/2002	30/12/2002

FIGURA 5.8 – Mapeamento da Tabela FUNCIONÁRIO para o DB2

Observando a figura 5.8 podemos notar que, no processo de mapeamento da tabela Funcionário foi criado um conjunto de tabelas: uma tabela base com o mesmo nome da tabela conceitual e quatro tabelas temporais, cada uma correspondendo a um atributo temporal. A tabela base foi definida com todos os atributos da tabela conceitual (temporais e não temporais), acrescida dos rótulos temporais *tvalid_ini* e *tvalid_fim,* cuja função é indicar o tempo de vida de cada tupla. Os atributos *nome*, *departamento, funcao* e *salario,* por terem sido definidos como temporais, foram mapeados cada um para uma tabela contendo a chave da tabela base (*matricula*), o atributo temporal correspondente e os rótulos temporais. A chave primária de cada tabela temporal é definida pelo par de atributos (*matricula,tvalid_ini*). Os relacionamentos entre a tabela base e as tabelas temporais são representados por *chaves estrangeiras* definidas por *restrições de integridade referencial* especificadas entre as tabelas correspondentes. Neste exemplo, a chave estrangeira é o atributo *matricula*.

5.6.3 Evolução do Esquema

Embora o esquema da aplicação tenha sido definido com três tabelas, a primeira versão foi criada apenas com as tabelas FUNCIONÁRIO e DEPARTAMENTO, cujos atributos são apresentados na tabela 5.6. Denominaremos a primeira versão de SV1.

TABELA 5.6 – Meta-esquema da Primeira Versão

Tabelas	**Atributos**	
FUNCIONARIO	matricula(pk)	
	nome	Temporal
	dtnasc	
	sexo	
	depto	Temporal
	salario	Temporal
DEPARTAMENTO	dep_codigo(pk)	
	dep_nome	

Na segunda versão (denominada SV2), o esquema conceitual foi modificado para incluir os atributos *função* e *endereco* e remover o atributo *sexo* da tabela FUNCIONÁRIO. Além disso, foi adicionada a tabela FUNCAO. A tabela 5.7 mostra o novo esquema criado.

TABELA 5.7 – Meta-esquema da Segunda Versão (continua)

Tabelas	**Atributos**	
FUNCIONARIO	matricula(pk)	
	nome	Temporal
	dtnasc	
	depto	Temporal
	salario	Temporal
	função	Temporal
	endereco	Temporal
DEPARTAMENTO	dep_codigo(pk)	
	dep_nome	

TABELA 5.7 – Meta-esquema da Segunda Versão (continuação)

FUNCAO	fun_codigo(pk)
	fun_nome

Finalmente, na terceira versão (SV3), a tabela FUNCIONÁRIO foi alterada mais uma vez para incluir os atributos *estado_civil* e *escolaridade* e excluir o atributo *endereco*. Foi também incluída a tabela ESCOLARIDADE, resultando no esquema apresentado na tabela 5.8.

TABELA 5.8 – Meta-esquema da Terceira Versão

Tabelas	**Atributos**	
FUNCIONARIO	matricula(pk)	
	nome	Temporal
	dtnasc	
	depto	Temporal
	salario	Temporal
	função	Temporal
	estado_civil	Temporal
	escolaridade	Temporal
DEPARTAMENTO	dep_codigo(pk)	
	dep_nome	
FUNCAO	fun_codigo(pk)	
	fun_nome	
ESCOLARIDADE	esc_codigo(pk)	
	esc_nome	

O resultado do processo de versionamento será demonstrado a partir das modificações realizadas na tabela FUNCIONARIO.

Na alternativa de armazenamento usando vários repositórios, para cada versão do esquema é criado um novo repositório de dados contendo somente os atributos definidos na versão correspondente. Neste caso, serão criadas três versões da tabela FUNCIONARIO, cada uma armazenada em um repositório específico formatado de acordo com os esquemas definidos em SV1, SV2 e SV3.

Na solução usando somente um repositório, os dados de todas as versões compartilham o mesmo repositório, formatado de acordo com um esquema global que inclui todos os atributos introduzidos pelas sucessivas mudanças de esquema. Sendo assim, após a modificação definida em SV3, a tabela FUNCIONARIO terá a estrutura física apresentada na tabela 5.9. Nela constam todos os atributos definidos na evolução do esquema.

TABELA 5.9 – Esquema Completo da Tabela Funcionario

Tabela	**Atributos**	
FUNCIONARIO	matricula(pk)	
	nome	Temporal
	dtnasc	
	sexo	
	depto	Temporal
	salario	Temporal
	função	Temporal
	endereço	Temporal
	estado_civil	Temporal
	escolaridade	Temporal

5.7 Considerações Finais

Inicialmente este capítulo apresentou o mapeamento do modelo TRM para o banco de dados DB2, sendo especificados os requisitos necessários para a criação de um ambiente temporal usando versões do esquema, incluindo desde a definição da estrutura de armazenamento dos meta-dados, o tipo de versionamento desejado até a forma de armazenamento dos dados da aplicação.

Em seguida foi apresentada uma ferramenta que auxilia na definição das versões do esquema, usando o TRM. Na interface da ferramenta o usuário especifica o esquema conceitual, através dos meta-dados, e executa as operações de gerenciamento das versões. A versão atual da ferramenta permite suporte às seguintes configurações:

- versionamento de esquemas - somente tempo de transação;
- temporalidade dos dados - tempo de transação, tempo de validade e bitemporal;
- tipo de armazenamento - um repositório e múltiplos repositórios;
- gerenciamento - sempre síncrono.

Pretende-se estender as funcionalidades da ferramenta para contemplar as outras alternativas de versionamento do esquema: tempo de validade e bitemporal. Além disso, incluir um controle que permita escolher o tipo de gerenciamento dos dados: síncrono ou assíncrono. Com essas complementações, a ferramenta poderá se tornar um gerenciador completo.

Finalizando o capítulo, foi apresentada a modelagem de uma aplicação através do TRM com o objetivo de demonstrar o uso da ferramenta.

No próximo capítulo serão mostrados os resultados obtidos com a experiência realizada.

6 Análise dos Resultados

Este capítulo apresenta o ambiente de testes e compara os resultados obtidos com o protótipo desenvolvido, a partir das duas experiências de implementação realizadas.

Nas duas implementações procurou-se manter as mesmas opções de temporalidade para os esquemas e os dados, respectivamente tempo de transação e tempo de validade, mudando-se somente a forma de armazenamento dos dados. Portanto, as comparações serão realizadas entre o uso de somente um repositório e o uso de vários repositórios, considerando versionamento de esquemas de tempo de transação e gerenciamento síncrono.

6.1 Ambiente Computacional

Todos os testes foram realizados em um computador AMD-Athlon 1.4 Ghz com 128 MB de memória, operando com o sistema operacional Windows 98, acessando o banco de dados DB2 local (desktop). O acesso da aplicação ao DB2 é realizado através de driver ODBC.

6.2 Critérios de Comparação

A comparação é baseada num conjunto de critérios que incluem: o espaço utilizado para armazenar os dados da extensão, o processamento de atualizações em conseqüência da mudança de esquema e o tempo de resposta das consultas.

Espaço de armazenamento

Um sistema de banco de dados temporal é um sistema que mantém o registro dos estados passados dos dados. Em tais sistemas, cada alteração nos dados cria uma nova versão dos mesmos. O sistema nunca remove os dados, apenas marca o seu tempo de remoção. Como conseqüência deste fato, a quantidade de dados é sempre crescente. Por outro lado, o suporte ao versionamento de esquemas requer que, além da evolução dos dados, também sejam armazenados as versões antigas dos esquemas e seus dados associados, exigindo mais espaço de armazenamento.

O espaço consumido no armazenamento das versões e dos dados depende da alternativa de armazenamento escolhida. A seguir apresentamos uma análise das alternativas usadas.

- múltiplos repositórios – nesta alternativa, cada vez que uma versão de esquema é criada, um novo repositório também é criado para armazenar os dados. À medida que aumenta o número de versões, aumenta o número de repositórios, gerando uma grande quantidade de dados redundantes. Isto ocorre porque, ao criar uma nova versão, os dados correntes da versão antiga precisam ser copiados para o novo repositório; portanto, esses dados podem ser armazenados várias vezes, em

diferentes repositórios. O tamanho do novo repositório depende do número de tuplas correntes, do número de atributos adicionados e do número de atributos excluídos. Os atributos excluídos não farão parte do novo repositório;

- repositório único – nesta solução os dados de todas as versões são armazenados no formato aumentado, sempre no mesmo repositório. Ao contrário da solução anterior, esta solução evita a duplicação dos dados. Porém, há um consumo de espaço adicional para armazenar os valores nulos. Quanto mais atributos forem adicionados, maior será o tamanho do repositório.

Atualização dos dados da extensão devido à mudança de esquema

Realizadas as modificações no esquema conceitual, há necessidade de atualizar os dados extensionais para que os mesmos reflitam as novas definições. A solução escolhida para o armazenamento dos dados influencia a atualização. Os detalhes da atualização em cada alternativa estão descritos a seguir.

- múltiplos repositórios – nesta solução, a atualização dos dados é relativamente simples, pois a cada versão corresponde um repositório. Assim sendo, a atualização consiste em criar as estruturas do novo repositório de acordo com as novas definições, selecionar do repositório antigo apenas os dados atuais ou com validade no futuro, adapta-los ao novo formato e copia-los para o novo repositório. Neste tipo de armazenamento, cada repositório contém apenas os atributos definidos na versão correspondente;
- repositório único – como existe apenas um repositório para armazenar todas as versões do esquema, cada vez que uma nova versão é criada todo o repositório é afetado pela mudança. O processo de atualização consiste em recuperar todos os dados numa estrutura temporária, redefinir a estrutura do repositório de acordo com o novo esquema e converter todos os dados ao novo formato. O tamanho do repositório nunca diminui, somente aumenta, pois nenhum atributo pode ser removido do repositório. Durante a realização deste processo o banco de dados fica indisponível.

Recuperação de Dados

A eficiência de um sistema de banco de dados está bastante relacionada com a facilidade de recuperação das informações armazenadas.

Comandos de recuperação de informações que envolvem o fator tempo são um tipo especial de consulta. Nestes comandos, o aspecto tempo pode ser envolvido de três formas diferentes:

- para recuperar valores cujo domínio é temporal;
- para se referir a um determinado instante ou intervalo temporal;
- para recuperar valores com base em restrições temporais.

As consultas que envolvem o fator tempo no seu contexto não dependem somente da especificação da informação, dependem também do tipo de banco de dados

implementado, do modelo de dados utilizado, da história considerada e da forma de armazenamento dos dados. As duas opções de armazenamento são analisadas a seguir:

- múltiplos repositórios – esta alternativa simplifica a resolução de consultas quando a versão do esquema é especificada explicitamente no comando de consulta; neste caso, a identificação da versão do esquema define de qual repositório a resposta será recuperada. Dessa maneira, a resposta é selecionada diretamente do repositório correspondente à versão informada, portanto o número de tuplas pesquisadas é limitado ao tamanho do repositório. Os dados são recuperados no formato com que foram inseridos. No entanto, se o período da consulta envolve várias versões, teremos a consulta particionada (multi-esquema), com a resposta sendo selecionada de vários repositórios; neste caso, somente são recuperados de cada repositório os dados inseridos sob a versão à qual corresponde o repositório;
- repositório único – todo o repositório é pesquisado para a recuperação das respostas. Como os dados são armazenados no formato aumentado, uma fase de filtragem é necessária com o objetivo de adaptar os dados recuperados às diferentes versões do esquema envolvidas na consulta.

6.3 Resultados Experimentais

Os testes de avaliação foram realizados usando somente a tabela *Funcionário* gerada com duas versões do esquema. A primeira versão foi criada com 8 atributos assim distribuídos: 3 atributos temporais, 3 atributos não temporais e dois atributos correspondendo aos rótulos temporais. Na segunda versão um atributo não temporal foi removido e foram adicionados dois atributos temporais. A descrição dos atributos de cada versão encontra-se nas tabelas 5.6 e 5.7.

Para avaliar o tempo e o espaço consumidos no processo de versionamento, várias experiências foram conduzidas, cada uma considerando um número diferente de tuplas. Nestes testes foram incluídas 10, 20, 30 e 40 mil tuplas na primeira versão. Em cada situação, após a inclusão dos dados, o esquema foi modificado gerando a nova versão. A mesma experiência foi realizada nos dois tipos de armazenamento. Os resultados obtidos estão apresentados nas tabelas 6.1 e 6.2

TABELA 6.1 – Resultados do versionamento com vários repositórios

		Número de tuplas			
		10000	20000	30000	40000
Primeira versão	Tempo de geração do primeiro esquema (em segundos)	2	2	2	2
	Tempo para inclusão das tuplas na base (em segundos)	335	677	1013	1335
	Tamanho do repositório (em Mb)	6,16	10,74	14,81	18,74
Segunda versão	Tempo de atualização dos esquemas e dados (em segundos)	15	650	1249	2096
	Tamanho do repositório (em Mb)	7,34	11,92	16,12	20,05
Espaço total utilizado nas 2 versões (em Mb)		13,50	22,66	30,93	38,79

TABELA 6.2 – Resultados do versionamento com apenas um repositório

		Número de tuplas			
		10000	20000	30000	40000
Primeira versão	Tempo de geração do primeiro esquema (em segundos)	2	2	2	2
	Tempo para inclusão das tuplas na base (em segundos)	335	676	1013	1335
	Tamanho do repositório (em Mb)	6,16	10,74	14,81	18,74
Segunda versão	Tempo de atualização dos esquemas e dados (em segundos)	6	8	11	15
	Tamanho do repositório (em Mb)	7,34	11,92	16,12	20,05
Espaço total utilizado nas 2 versões (em Mb)		7,34	11,92	16,12	20,05

Em todas as situações apresentadas foram utilizados os mesmos conjuntos de dados. Os tempos de geração e atualização dos esquemas referem-se ao tempo total desde que o usuário seleciona a operação *Ativar Versão*, incluindo a leitura dos meta-dados, o processo de mapeamento, até a conversão ou adaptação dos dados. Observa-se que o tempo de geração do primeiro esquema é o mesmo em todos os casos, uma vez que a base de dados ainda não existia; logo, esse tempo se refere somente ao tempo de criação dos esquemas da tabela base e das tabelas temporais associadas à tabela Funcionário. O tamanho do repositório após a inclusão dos dados da primeira versão também é igual nas duas versões.

Sob as mesmas condições dos dados, a principal diferença está no tempo de atualização da base de dados. Na opção de armazenamento em vários repositórios, cada vez que uma versão é criada um novo repositório de dados é também criado para armazenar a nova versão. No exemplo em questão, na primeira versão são criados os esquemas para quatro tabelas, enquanto na segunda versão são criados os esquemas para 5 tabelas. Cada conjunto de tabelas é armazenado em um repositório separado, portanto um novo local de armazenamento físico é criado. Quando aumenta o número de tuplas a serem copiadas, o tempo de atualização aumenta consideravelmente. No caso do armazenamento em repositório único, o tempo de atualização é bem menor, resumindo-se ao tempo necessário para adaptar os esquemas das tabelas às novas definições. Nesta alternativa não são criados outros repositórios, todas as modificações são aplicadas à mesma cópia. Todos os dados são mantidos no mesmo repositório.

O espaço também aumenta em função do número de tuplas e do número de atributos. No nosso exemplo, a diferença de espaço não é perceptível, pois o atributo excluído (*sexo*) contém apenas um caracter e o atributo adicionado (*endereço*) possui 50 caracteres. Essa diferença tende a aumentar à medida que mais atributos forem adicionados ou removidos.

6.4 Considerações Finais

Este capítulo apresentou o ambiente de testes e os resultados obtidos nas experiências. Foram analisados o espaço e o tempo consumidos na geração e atualização das estruturas de dados em conseqüência das modificações no esquema.

Os testes foram realizados considerando apenas uma configuração de ambiente temporal variando o tipo de armazenamento. As demais opções de configuração, bem como o processamento de consultas, podem ser tratados em um trabalho futuro.

7 Conclusão

Este trabalho apresentou o mapeamento do modelo temporal TRM para o DB2, no qual foram especificados os requisitos necessários para a criação de um ambiente temporal usando versões de esquema.

Para implementar essa proposta, foi definida uma ferramenta cuja finalidade principal é auxiliar o usuário nas operações de versionamento do esquema. Na interface da ferramenta o usuário especifica o esquema conceitual, através dos meta-dados, e executa as operações de gerenciamento das versões.

Duas experiências de implementação foram realizadas utilizando duas formas diferentes de armazenar os dados, com o objetivo de avaliar os resultados obtidos em cada implementação, considerando um determinado volume de dados e alterações.

Como base para este trabalho foram realizados os seguintes estudos, descritos nos capítulos 2, 3 e 4, respectivamente:

a) estudo sobre bancos de dados temporais, incluindo sua classificação e técnicas de versionamento de esquemas e dados;

b) estudo do modelo relacional temporal TRM, suas características, e operações de modificação no modelo relacional;

c) estudo das alternativas para a implementação de bancos de dados temporais com suporte ao versionamento de esquemas.

Dentre as contribuições, podemos destacar:

a) definição do mapeamento do TRM para o DB2;

b) desenvolvimento de uma aplicação capaz de gerenciar as variações do esquema conceitual de forma consistente com os repositórios dos dados;

c) comparação entre o uso de um repositório e de vários repositórios.

A partir desse estudo foi possível realizar as comparações entre as duas formas de armazenar os dados da extensão: um repositório e vários repositórios. O tipo de solução usado para o armazenamento depende dos requisitos da aplicação, levando-se em consideração, principalmente, o espaço de armazenamento, o tempo para a atualização dos dados e de resposta às consultas.

De acordo com os resultados obtidos, verificou-se que a utilização de somente um repositório para o armazenamento dos dados apresentou vantagem sobre o uso de vários repositórios, tanto no tempo de adaptação da base de dados quanto no espaço consumido.

Outra característica a ser considerada é a disponibilidade da base de dados. Com o uso de somente um repositório, o BD fica indisponível durante a adaptação dos dados, enquanto que usando vários repositório isso não acontece, o BD fica disponível para consultas.

Entre as restrições apresentadas, podemos destacar as seguintes, que podem ser sugestões para trabalhos futuros:

a) as modificações permitidas na evolução de esquema restringem-se às seguintes operações: inclusão e exclusão de tabela, inclusão e exclusão de atributo, expansão de tamanho do atributo e mudança de tipo (domínio) somente para tipo compatível. Seria interessante disponibilizar ao usuário todas as operações relacionadas na seção 3.3;

b) a versão atual da ferramenta permite suporte às seguintes configurações:

- versionamento de esquemas - somente tempo de transação;
- temporalidade dos dados - tempo de transação, tempo de validade e bitemporal;
- tipo de armazenamento dos dados - um repositório e múltiplos repositórios;
- gerenciamento - sempre síncrono.

Pretende-se estender as funcionalidades da ferramenta para contemplar as outras alternativas de versionamento do esquema: tempo de validade e bitemporal. Além disso, incluir um controle que permita escolher o tipo de gerenciamento dos dados: síncrono ou assíncrono.

c) na avaliação dos resultados, não está sendo considerado o tempo de processamento das consultas. O processamento de consultas é fundamental, portanto justifica a necessidade de realizar um estudo mais abrangente. Para que isso seja possível, deve-se complementar o módulo de consultas da ferramenta.

Referências

[ANG 2001] ANGONESE, S.F.; EDELWEISS, N. Gerenciador Temporal de Versões de Esquemas. In: CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA, 2001, Mérida, Venezuela. **Proceedings...** [S.1.:s.n.], 2001.

[ANT 97] ANTUNES, D.C.; HEUSER, C.A.; EDELWEISS, N. TempER: uma abordagem para modelagem temporal de banco de dados. **Revista de Informática Teórica e Aplicada**, Porto Alegre, v.4, n.1, p.49-85, 1987.

[BOU 97] BOUDJLIDA, N.; PERRIN O. Schema Updates and Database Restructuring. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 12., 1997, Fortaleza. **Anais...** Fortaleza: UFC, 1997. p.210-225.

[CAS 95] CASTRO, Cristina De; GRANDI, Fabio; SCALAS, Maria R. On Schema Versioning in Temporal Databases. In: CLIFFORD, J.; TUZHILIN, A. (Ed.). **Recents Advances in Temporal Databases**. Great Britain: Springer, 1995. p.272-291.

[CAS 97] CASTRO, Cristina De; GRANDI, F.; SCALAS, Maria R. Schema Versioning for Multitemporal Relational Databases. **Information Systems**, Oxford, v.22, n.5, p.249-290, 1997.

[CLI 93] CLIFFORD, J.; CROCKER, A. The historical relational data model (HRDM) Reviseted. In: TANSEL, A.U. (Ed.). **Temporal Databases**: Theory, Design and Implementation. Redwood City: Benjamim/Cummings, 1993.

[EDE 94] EDELWEISS, N.; OLIVEIRA, J.P.M. de. **Modelagem de Aspectos Temporais de Sistemas de Informação**. Recife: UFPE-DI, 1994. Trabalho apresentado na 9. Escola de Computação, 1994, Recife.

[EDE 95] EDELWEISS, N.; OLIVEIRA, J.P.M.; CASTILHO, José M.V. Evolução de Esquemas em Bancos de Dados Temporais. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 22., 1995, Canela. **Anais**...Porto Alegre: SBC, 1995. p. 375-386.

[EDE 98] EDELWEISS, N. Bancos de Dados Temporais: teoria e prática In: JORNADA DA ATUALIZAÇÃO EM INFORMÁTICA, JAI, 17., Belo Horizonte. **Anais...** Belo Horizonte: SBC,1998. p. 225-282.

[EDE 2000] EDELWEISS, N.; HÜBLER, P.N.; MORO, M.M.; DEMARTINI, G. A Temporal Database Management System Implemented on Top of a Conventional Database. In: INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY, 20., 2000, Santiago, Chile. **Proceedings…** Los Alamitos: IEEE Computer Society,

2000. p.58-67.

[ELM 2000] ELMASRI, R.; NAVATHE, S.B. **Fundamentals of Database Systems**. 3rd ed. Redwood City: Addison-Wesley Longman, 2000. 955 p.

[ETZ 98] ETZION, O.; JAJODIA, S.; SRIPADA, E. (Ed.). **Temporal databases:** Research and Pratice. Berlin: Springer-Verlag, 1998. (Lecture Notes in Computer Science; 1300)

[GAL 2001] GALANTE, R.M. **Evolução de Esquemas e o Emprego de Versões no Contexto de Bancos de Dados Orientados a Objetos**. 2001. Exame de Qualificação (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[GAL 2002] GALANTE, R.M.; EDELWEISS, N.; SANTOS, C. Change Management for a Temporal Versioned Object-Oriented Database. In: INTERNATIONAL WORKSHOP ON EVOLUTION AND CHANGE IN DATA MANAGEMENT, 2., 2002, Tampere, Finland. **Proceedings…** [S.l.:s.n.], 2002. p.1-12.

[GAL 2002a] GALANTE, R.M.; ROMA, A.B.S.; JANTSCH, A.; EDELWEISS, N.; SANTOS, C.S. Dynamic Schema Evolution Management using Version in Temporal Object-Oriented Databases. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 13., 2002, Aix-en-Provence, France. **Proceedings…** [S.l.]:Springer-Verlag, 2002. p.524-533.

[GOR 97] GORALWALLA, I.A.; SZAFRON, D.; ÖZSU, M.T.; PETERS, R.J. Managing Schema Evolution using a Temporal Object Model. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 16., 1997. **Proceedings...** [S.l.:s.n.], 1997.

[HÜB 2000] HÜBLER, P.N. **Definição de um Gerenciador para o Modelo de Dados Temporal T-FORM**. 2000. 109p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[JEN 94] JENSEN, C.S. et al. A Consensus Glossary of Temporal Databases Concepts. **SIGMOD Record,** New York, v.23, n.1, p.53-63, Mar.1994.

[JEN 98] JENSEN, C.S. et al. The Consensus Glossary of Temporal Database Concepts – February 1998 Version. In: ETZION, O.; JAJODIA S.; SRIPADA, S. (Ed.). **Temporal Databases Research and Practice.** Heidelberg: Springer-Verlag, 1998. p. 367-405.

[LIN 93] LINCOLN, C.M. de O. **Incorporação da Dimensão Temporal em Bancos de Dados Orientados a Objetos**. 1993. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Campinas, Campinas-SP.

[MOR 99] MOREIRA, V.P. **Consultas a Bancos de Dados Temporais que Suportam Versionamento de Esquemas**. 1999. 136p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MOR 99a] MOREIRA, V.; EDELWEISS, N. Queries to Temporal Databases Supporting Schema Versioning. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 14., 1999, Florianópolis. **Anais...** Florianópolis: UFSC, 1999. p. 299-313.

[MOR 99b] MOREIRA, V.; EDELWEISS, N. Schema Versioning and The Generalised Temporal Database System. In: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA, CLEI, 25., 1999, Assuncion. **Memórias**. Assuncion, Paraguay: Universidad Autônoma de Assuncion, 1999. p. 111-122.

[NAV 93] NAVATHE, S.; AHMED, R. Temporal Extensions to Relational Model and SQL. In: TANSEL, A.U. et al. (Ed.). **Temporal Databases:** theory, design and Implementation. Redwood City: Benjamim/Cummings, 1993. p. 92-109.

[ROD 94] RODDICK, J.F. **A Model for Temporal Inductive Inference and Schema Evolution in Relational Database Systems**. [S.l.]: Department of Computer Science and Computer Engineering, La Trobe University, 1994.

[ROD 96] RODDICK, J.F. A Model for Schema Versioning in Temporal Database Systems. In: AUSTRALIAN COMPUTER SCIENCE CONFERENCE, 19., 1996, Melbourne. **Procedings…** [S.l.:s.n.], 1996.

[ROD 99] RODDICK, J.F. **A Survey of Schema Versioning Issues for Database Systems**. [S.l.]: Advanced Computing Research Center, School of Computer and Information Science, University of South Australia, 1999.

[ROM 2001] ROMA, A.B.S. et al. Gerenciamento Temporal de Versões para Evolução de Esquemas em BDOO. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 2001, Rio de Janeiro, RJ, Brasil. **Anais...** Rio de Janeiro: UFRJ, 2001.

[SNO 95] SNODGRASS, R.T. et al. **The TSQL2 Temporal Query Language**. Noewell-MA: Kluwer Academic Publishers, 1995.

[SNO 2000] SNODGRASS, R.T. **Developing Time-Oriented Database Applications in SQL.** San Francisco: Morgan Kaufmann, 2000.

[SIM 98] SIMONETO, Eugênio de Oliveira. **Uma Proposta para a Incorporação de Aspectos Temporais, no Projeto Lógico do Banco de Dados, em SGBDs Relacionais**. 1998. 72p. Dissertação (Mestrado em Informática) – Instituto de Informática, Pontifícia Universidade

Católica do Rio Grande do Sul, Porto Alegre.

[TAN 93] TANSEL, A.U. et al. **Temporal Databases – Theory, Design and Implementation**. Redwood City: Benjamim/Cummings, 1993. 633p.

[WEI 99] WEI, Han-Chieh; ELMASRI, Ramez. Study and Comparison of Schema Versioning and Database Conversion Techniques for Bi-temporal Databases. In: INTERNATIONAL WORKSHOP ON TEMPORAL REPRESENTATION AND REASONING, 6., 1999. **Proceedings…** [S.l.]: IEEE, 1999. p.88-98.

[WEI 2000] WEI, Han-Chieh; ELMASRI, Ramez. PMTV: A Schema Versioning Approach for Bi-temporal Databases. In: INTERNATIONAL WORKSHOP ON TEMPORAL REPRESENTATION AND REASONING, 7., 2000. **Proceedings…** [S.l.]: IEEE, 2000.