MINERAÇÃO DE DADOS ENDÓGENOS

Alberto Sulaiman Sade Júnior

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Jano Moreira de Souza, Ph.D.

Julia Celia M. Strauch

Profª Julia Celia Mercedes Strauch, D.Sc.

Profª Marta Lima de Queirós Mattoso , D.Sc.

Prof. Astério Kiyoshi Tanaka, Ph.D.

Prof. Emmanuel Pisesses Lopes Passos, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2002

SADE JÚNIOR, ALBERTO SULAIMAN

Mineração de dados endógenos [Rio de Janeiro] 2002

X, 98 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Mineração de dados 2. armazém de dados 3. engenharia reversa

I. COPPE/UFRJ II. Título (série)

Aos meus Amores,

Simone e Sammir

Agradecimentos

Ao Professor Jano Moreira de Souza, por sua orientação e incentivo que possibilitaram o desenvolvimento deste trabalho.

À Professora Julia Celia Mercedes Strauch, por suas valiosas sugestões e incentivo, que muito acrescentaram ao desenvolvimento deste trabalho.

Aos demais membros de minha banca de tese, Prof. Astério Kiyoshi Tanaka, Prof. Emmanuel Pisesses Lopes Passos, Prof. Geraldo Bonorino Xexéo e Profa. Marta Lima de Queirós Mattoso.

Aos companheiros e ex-companheiros da linha de Banco de Dados do Programa de Engenharia de Sistemas e Computação.

À minha esposa, Simone Maria Fragoso Antonio, pelo apoio, paciência e compreensão, sem os quais não teria sido possível realizar o doutorado.

Ao meu filho Sammir, aos meus pais, meus sogros, enfim: minha família.

À amiga Mônica Fraga Soriano, pelo apoio e incentivo, nos momentos de angústia e solidão.

Ao amigo José Roberto Brasileiro de Siqueira pelo apoio na obtenção dos dados necessários para a conclusão da tese, bem como no entendimento do problema do leilão de títulos públicos do Banco Central do Brasil.

E aos demais professores, colegas e funcionários da COPPE/Sistemas, e a todos aqueles que de forma direta ou indireta contribuíram para a elaboração deste trabalho.

Aos professores, colegas e funcionários do IBICT/MCT, e a todos aqueles que de forma direta ou indireta contribuíram para a elaboração deste trabalho.

À CAPES e ao professor Rakesh Agrawal, bem como aos colegas do Almaden Research Center por ocasião do PDEE no. BEX0533/98-0.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de doutor em Ciências (DSc.).

MINERAÇÃO DE DADOS ENDÓGENOS

Alberto Sulaiman Sade Júnior

Março / 2002

Orientador: Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

A engenharia reversa de sistemas herdados é um problema complexo que vem merecendo atenção da academia e da indústria. Muitos métodos foram propostos para solucioná-la, mas não existe um método que resolva todas as situações apropriadamente. Uma abordagem para complementar as soluções de engenharia reversa de sistemas herdados é a análise dos *logs* de bancos de dados combinada com os catálogos dos sistemas herdados.

Logs de bancos de dados são usados principalmente para recuperação de transações, controle de concorrência, auditoria e análise de desempenho. Um *log* de comando de banco de dados é uma categoria especial de *log* com informação sobre comandos, acessos, arquivos, usuários e horários que podem ser cruzados facilmente. Como todas as ocorrências das transações de banco de dados são armazenadas no *log* de comando de banco de dados, é possível adquirir o conhecimento oculto neste *log*.

Partindo-se deste *log*, formatado como uma tabela de fatos chamada de CUBO CRUD, é possível gerar fluxos de trabalho e regras de negócios usando-se técnicas de OLAP e de Mineração de Dados. O conhecimento obtido pode auxiliar analistas a identificar conhecimento não documentado, resolvendo alguns problemas ainda não solucionados por outras técnicas de engenharia reversa de sistemas herdados, que são tradicionalmente usadas na migração de sistemas herdados para novos sistemas.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (DSc.).

ENDOGENOUS DATA MINING

Alberto Sulaiman Sade Júnior

March / 2002

Advisor Jano Moreira de Souza

Department: Systems and Computer Engineering

Legacy systems reverse engineering is a complex problem that has earned attention from the academy and the industry. Several methods have been proposed to perform it, but there is no method that solves all situations adequately. One approach to complement other legacy systems reverse engineering solutions is to analyze database logs combined with database catalogs from legacy systems.

Database logs are used primarily for transaction recovery, concurrence control analysis, auditing, and performance analysis. A database command log is a special category of log with information about commands, accesses, files, users, and timestamps that can be crossed easily. Since all occurrences of database transactions are recorded in the database command log, it is possible to acquire knowledge hidden therein.

From that log, formatted as a fact table named CRUD CUBE, it is possible to generate workflows and business rules using OLAP and Data Mining techniques. The knowledge obtained can help system analysts to identify undocumented knowledge, solving some problems not yet addressed by other legacy systems reverse engineering techniques, traditionally used in the database migration from legacy systems to new systems.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo I - Introdução

I.1. Motivação

O desenvolvimento das Tecnologias da Informação, principalmente no que se refere à geração e armazenamento de dados, possibilita às corporações registrar qualquer transação realizada, em grandes bases de dados, sempre crescentes. Instituições governamentais possuem bases de dados demográficas, previdenciárias e de contribuintes, entre outras. Laboratórios de pesquisa armazenam mais e mais dados sobre clima, solo, uso da terra e rastreamento de imagens por satélite. Sabe-se que existe informação escondida nestes dados: o seu registro assumiu um valor estratégico.

O enorme crescimento destas bases, empresariais, governamentais ou científicas, vem ultrapassando a habilidade técnica e a capacidade humana na sua interpretação. Entretanto, a automação no processo de armazenamento dos dados não foi acompanhada no que se refere à sua análise.

Isto se deve ao fato de que interpretação e análise são atividades essencialmente humanas: envolvem raciocínio e heurísticas complexas, nem sempre evidentes ou facilmente programáveis em um sistema de computador.

A capacidade de interpretação de grandes quantidades de dados pode ser aumentada através da automação parcial destes processos. Uma nova geração de ferramentas e técnicas para análise em bancos de dados, tais como mineração e armazém de dados, está auxiliando nesta tarefa.

Novas formas de representação de dados merecem atenção especial, porém é na grande quantidade de dados ditos convencionais, armazenados em sistemas herdados, que repousa a maior parte de conhecimento que pode ser recuperada pelas organizações que os possuem. Como, em geral, não há documentação sobre os processos de negócio que geraram os sistemas herdados, um método que possibilite a realização desta recuperação, mesmo que parcialmente, é desejável, por razões econômicas.

I.2. Justificativa

Existem muitas possibilidades de representação de taxonomias para classificar aplicações computacionais. Uma possibilidade é a necessidade de que determinada aplicação tenha de ser modelada mais intensamente neste ou naquele eixo de abstração (ZACHMAN, 1987). Outra possibilidade é a classificação destas aplicações segundo sua natureza, endógena ou exógena, em relação a uma categoria de sistemas.

Aplicações exógenas em relação a Sistemas Gerenciadores de Bancos de Dados (SGBDs) são programas genéricos que pertencem ao universo dos usuários e especialistas. Controle de estoques e transações bancárias são relacionadas a usuários. Sistemas de aconselhamento médico e para concessão de empréstimo bancário são relacionados com especialistas.

Aplicações endógenas em relação a SGBDs são programas de administração destes sistemas. Exemplos típicos são o controle de restrições de integridade referencial, sistemas de reorganização periódica de arquivos, e sistemas de gerência de cópias de segurança (*backup*).

Neste trabalho, adotam-se as definições de: (INMON, 1992) "Um armazém de dados é um conjunto de dados baseado em assuntos, integrado, não volátil, e variável em relação ao tempo, de apoio às decisões gerenciais"; (GRAY & REUTER, 1993): "Um *log* de Banco de Dados é o banco de dados temporal. As sessões *online*, tabelas contextos, filas e outros objetos duráveis são apenas suas versões correntes. O *log* possui todo o seu histórico"; e ainda: "O gerenciador de *log* fornece acesso para leitura e gravação à tabela de *log* para todos os outros gerenciadores de recursos do gerenciador de transações".

Deste modo, o *log* a que este trabalho se refere é aquele de natureza endógena, gerado pelo SGBD, e não aqueles gerados por programas de aplicação, mesmo que estes estejam armazenados como procedimentos em SGBDs. Esta diferenciação é fundamental para os objetivos deste trabalho, uma vez que a maioria das análises acontece sobre o rastreamento de transações provenientes de SGBDs programadas nas aplicações que utilizam facilidades dos SGBDs.

Partindo-se de *log*s endógenos, ainda é necessário que se faça à diferenciação quanto ao tipo de *log*. O principal uso de *logs* é a recuperação do estado original e consistente da base de dados, *desfazendo* a transação ou *refazendo-a* em caso de acidente, enquanto esta ainda está ativa, tal como descrito em manuais de produtos

(SOFTWARE AG, 1997, IBM, 1998, ORACLE, 1998) ou na literatura (ELMASRI & NAVATHE, 2000, GRAY & REUTER, 1993, GARCIA-MOLINA *et al.*, 2001).

Além desta funcionalidade, um *log* pode ser utilizado para análise de desempenho e contabilização do uso de recursos, gerando material de auxílio para a identificação de "gargalos" potenciais.

*Log*s de bancos de dados constituem um tipo de saída de dados que todo SGBD é passível de gerar e podem ser modelados e implementados como um tipo de armazém de dados endógenos. Como em qualquer armazém de dados, é possível realizar-se contra este as operações típicas dos *On-Line Analytical Processing* (OLAP), bem como de mineração de dados.

O uso destas operações, contra os armazéns de dados desta natureza, auxilia no entendimento do processo do negócio do sistema que o gerou, pela obtenção parcial de *workflow* e regras de negócios que teriam sido modelados por ocasião da concepção do sistema original.

Evidentemente, poder-se-ia advogar contra esta abordagem no sentido de que nada de novo estaria sendo descoberto. Porém, a grande quantidade de sistemas herdados ainda existentes no mercado aponta para soluções que auxiliem na obtenção das suas características essenciais originais, de modo que a sua reengenharia possa vir a acontecer de modo menos penoso e mais barato.

O método descrito neste trabalho possui diversas fontes inspiradoras na literatura (AGRAWAL *et al.*,1993, AGRAWAL & SRIKANT,1995, AGRAWAL *et al.*,1997, AGRAWAL *et al.*,1998, LEYMANN & ROLLER, 2000) entre outros. Por exemplo em (AGRAWAL *et al.*,1998) são gerados grafos, que representam processos de negócios, partindo-se de *log*s provenientes de software de *workflow*. Porém, diferentemente de (AGRAWAL *et al.*,1998), onde a execução de um processo, com suas atividades, é completamente representada em um registro de *log* do software de *workflow*, não existe nenhuma garantia de que o registro de *log* de banco de dados representará completamente uma atividade.

Exemplificando-se estas diferenças pode-se citar, entre outras: em um *software de workflow*, o tipo de grafo representado por seu editor gráfico, simboliza processos de negócios e suas atividades. Em SGBDs, a ênfase não está no processo do negócio, porém nos modelos de dados, bem como no processamento de transações ditas *On-Line Transaction Processing* OLTP contra as bases de dados contidas em seu Banco de

Dados.

Usando-se o *Framework* de Zachman (ZACHMAN, 1987, SOWA & ZACHMAN, 1992, INMON *et al.*,1997), é possível entender estas diferenças relacionadas com os eixos e níveis de abstração. A Figura I.1 exibe o *Framework* de Zachman (ZACHMAN, 1992) no seu formato final original. O *software* de *workflow* representa melhor as abstrações do eixo das funções. Já o *software* de banco de dados representa melhor as abstrações do eixo dos dados.

Uma vez que os *softwares* de *workflow* representam conhecimento na forma de grafos que simbolizam processos, tal como em diagramas de atividades da *Unified Modeling Language* (UML), enquanto os *software* de banco de dados representam conhecimento na forma de grafos que simbolizam objetos de dados, tal como em diagramas de classes da UML, o tipo e a granularidade da informação gerada nos arquivos de *log* de cada um destes *software* é também diferente.

Em geral, *log*s de transações provenientes de SGBDs, grupados no tempo, se referem a uma única atividade. Algumas vezes transações agrupadas resolvem parte de uma atividade previamente definida. Em outros casos, mais de uma atividade pode estar embutida em uma transação de banco de dados.

Neste trabalho usa-se o termo "transação abstrata" (SULAIMAN & SOUZA, 1998d), em contrapartida ao termo "transação concreta", para representar a atividade inicialmente idealizada pelo projetista, implementada em um banco de dados, que em geral é complexa, porém cuja implementação é realizada por um conjunto de *flat transaction* original aparentemente sem relações entre si.

Recordando (AGRAWAL *et al.*,1998), "um processo de negócios é um conjunto de atividades separadas" e "uma atividade é definida como uma ação que é uma unidade semântica em algum nível, ou uma função que modifica o estado de um processo". Porém, sob o ponto de vista de (ELMASRI & NAVATHE, 2000), "uma transação é a execução de um programa que faz acesso ou modifica o conteúdo de um banco de dados" e uma transação atômica é "uma unidade de trabalho ou uma unidade lógica do processamento de um banco de dados".

Ao minerar o *log* de transações de banco de dados, é possível descobrir conhecimento em forma de regras e de redes semânticas, relativas aos eixos de abstração de controle e funcional, respectivamente. As regras assim mineradas podem ser combinadas com regras obtidas dos metadados do banco de dados alvo, para realizar

a engenharia reversa.

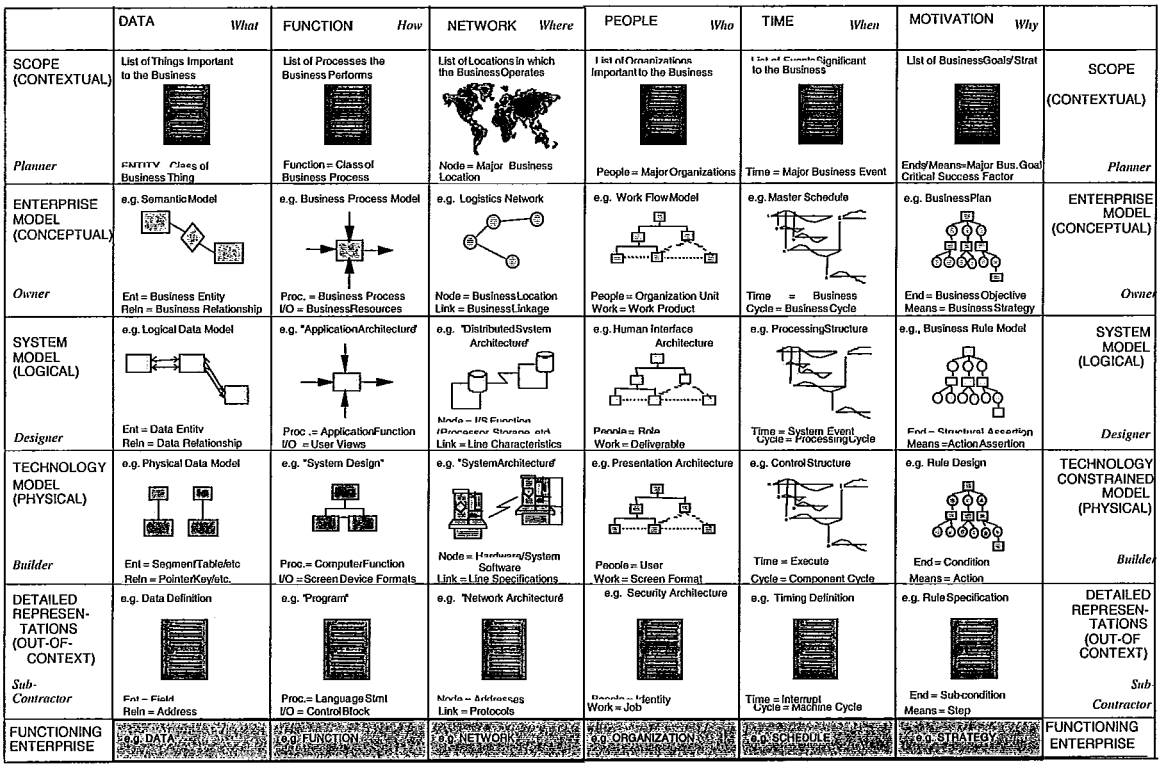


Figura I.1: Níveis e eixos de abstração (SOWA & ZACHMAN, 1992)

I.3. Objetivo

O objetivo deste trabalho é demonstrar que o uso das tecnologias de armazém de dados e de mineração de dados, aplicadas às bases de dados endógenas provenientes especificamente de catálogos e de *log*s de transações de bancos de dados, pode auxiliar na obtenção parcial de fluxos de trabalho e regras de negócio, constituindo-se em um método para a engenharia reversa de sistemas herdados. Analisando-se o seguinte conjunto de premissas:

1. *Log*s de Banco de Dados são orientados por assunto. O assunto é o armazenamento das transações de banco de dados;
2. *Log*s de Banco de Dados são integrados. Uma vez que o *log* registra o histórico da transação a informação obtida é semanticamente relacionada com suas aplicações de origem;
3. *Log*s de Banco de Dados evoluem com o tempo. Os *log*s registram as modificações dos objetos a cada alteração de estado;
4. *Log*s de Banco de Dados são não-voláteis. Os *log*s têm de ser preservados para recuperação de falhas, auditoria de sistemas e para controle de

concorrência.

Formula-se a seguinte hipótese:

O ponto de vista deste trabalho está no fato de que o software gerenciador de log é uma aplicação endógena ao SGBD e o log de banco de dados é um armazém de dados endógeno ao SGBD onde é possível realizar-se mineração de dados para o apoio à decisão do Planejador de Sistemas de Informação.

Partindo-se desta hipótese usa-se *log*s endógenos provenientes de execução de aplicações, como uma espécie de tabela de fatos, chamada CUBO CRUD, na extração de informação e conhecimento pela aplicação de técnicas de armazém de dados e mineração de dados.

I.4. Organização do trabalho

Este trabalho está organizado da seguinte forma:

O segundo capítulo, intitulado "armazém de dados e mineração de dados" aborda estas duas tecnologias de obtenção de informação e conhecimento. É apresentada uma visão geral, bem como a inserção de cada uma delas na prospecção de conhecimento em bancos de dados. São enfatizados os métodos relevantes no contexto do trabalho.

O terceiro capítulo, "regras de negócios e fluxos de trabalho", apresenta uma revisão de literatura de regras de negócios e fluxos de trabalho, elegendo um modelo para cada uma destas formas de representação do conhecimento do negócio da empresa, configurando-se os formatos de informação e conhecimento a serem obtidos neste trabalho. Este capítulo faz uma revisão da literatura dos conceitos de *business rules* ou Regras de Negócios, e de *workflow* ou Fluxos de Trabalho, sob a perspectiva da discussão clássica: "conhecimento declarativo" versus "conhecimento procedimental". Historicamente sempre houve, e ainda há, controvérsias sobre o uso predominante deste ou daquele paradigma, no contexto de desenvolvimento de sistemas, na Tecnologia da Informação.

O quarto capítulo, "mineração de dados endógenos", caracteriza os *log*s de transações de bancos de dados, bem como os catálogos dos SGBDs como bases de dados endógenas, passíveis de serem utilizadas na obtenção de conhecimento do negócio da empresa, sob forma de regras de negócios e fluxos de trabalho. Trata-se do material sobre o qual o conhecimento é extraído. São detalhados a intuição, o

formalismo, bem como o método proposto. É apresentada a descrição da construção do experimento realizado sobre o problema do leilão de títulos públicos do Banco Central do Brasil.

O quinto capítulo apresenta a conclusão, bem como propostas de trabalhos futuros.

Capítulo II - Armazém de Dados e Mineração de Dados

II.1. Introdução

O sucesso de qualquer organização depende da sua capacidade na realização de análise e planejamento, bem como da operacionalização de reações rápidas e imediatas às mudanças no ambiente dos negócios. O suporte a estas atividades é reconhecidamente a gerência de mais e melhores informações para o constante aperfeiçoamento dos processos do negócio. Os Sistemas de Informação constituem a resposta a esta demanda.

Porém não existe uma definição universalmente aceita para o termo "Sistema de Informação". Uma definição possível é aquela que diz que é "um esforço organizado para prover informações que permitam à empresa decidir e operar" (DIAS, 1985).

Todo e qualquer conjunto de atividades necessárias ao controle e uso da informação constitui parte de um Sistema de Informação. Trata-se de um Sistema Sociotécnico cujos componentes são "indivíduos, tarefas e equipamentos" (DIAS, 1985).

Dentro deste enfoque, e levando-se em consideração a pirâmide organizacional, um Sistema de Informação possui três grandes componentes, a saber: o componente Humano, o Organizacional e o Computacional.

Já uma definição de Sistemas de Apoio à Decisão (SADs) pode ser considerada entre dois extremos conceituais. Uma conceituação ampla é aquela que diz: "SADs são aqueles que contribuem de alguma forma para a tomada de decisão" (SPRAGUE & CARLSON, 1982). Uma conceituação restrita é aquela que diz: "SADs são sistemas baseados em computador, interativos, que auxiliam gerentes a utilizar dados e modelos para resolver problemas não-estruturados" (SPRAGUE & CARLSON, 1982).

Existe uma classificação que representa "uma dicotomia intuitiva" (SPRAGUE & WATSON, 1991, PASSOS, 1989) que rotula as atividades de processamento de informações em dois tipos:

- As soluções do TIPO I se referem à natureza predominantemente algorítmica de toda uma categoria de problemas e possuem estrutura definida. Em geral os

Sistemas de Informação são sistemas que contemplam estas atividades. Na maior parte das vezes é implementada como aplicações do dia a dia do negócio, os sistemas de produção; e

- As soluções do TIPO II se referem à natureza predominantemente heurística de outra categoria de problemas, que não possuem uma estrutura definida e envolvem: acompanhamento, monitoramento, aconselhamento, análise, elaboração e resolução de problemas. Na maior parte das vezes é implementada como aplicações sobre o negócio. São sistemas que auxiliam em tomada de decisão tais como SADs.

As soluções do TIPO I estão mais próximas do conceito de "sistemas de respostas planejadas" (MACMENAMIM & PALMER, 1984), e podem ser instanciadas na citação de (OZSU & VALDURIEZ, 1999): "Ao longo dos anos, as empresas vem construindo bancos de dados operacionais para suportar suas operações do dia-a-dia com aplicações OLTP".

As soluções do TIPO II são mais próximas do conceito de "sistemas ad hoc" (MACMENAMIM & PALMER, 1984), e podem ser instanciadas na citação de (OZSU & VALDURIEZ, 1999): "aplicações de apoio à decisão foram conceituadas como OLAP, para melhor refletir seus requisitos diferenciados. Aplicações OLAP, tais como análise de tendências e predição, necessitam de analisar dados históricos, resumidos, provenientes dos bancos de dados operacionais".

Este capítulo apresenta uma visão geral de duas tecnologias que possibilitam soluções do TIPO II: armazém de dados (*data warehouse*, DW) e mineração de dados (*data mining*, DM). Ambas estão descritas no contexto da prospecção de conhecimento em banco de dados (*knowledge discovery in databases,* KDD). Cada uma destas tecnologias é enfocada brevemente com sua descrição e definição adotada. São feitas analogias e comparações entre os dois paradigmas. Estas são as tecnologias usadas neste trabalho para a obtenção semi-automática de informação e conhecimento.

II.2. Armazém de Dados

O termo *datawarehousing* se refere ao processo de criação de um armazém de dados pela depuração e o armazenamento destes dados por assunto, tornando-os disponíveis para recuperação *online*. Este armazém contém dados históricos integrados por assunto, ou domínio de aplicação, para utilização em ambientes de análise de dados

e SADs.

A comunidade de Banco de Dados desenvolveu uma série de ferramentas para a análise de armazéns de dados chamada de OLAP. As ferramentas OLAP enfocam análises multidimensionais de dados de modo superior aos mecanismos de *summary* ou resumo e *break-down* ou lotes encaixantes ou de "quebra" entre dimensões, oferecidos pelas ferramentas tradicionais.

Thomsen (THOMSEN, 1997) estabelece as relações entre o armazém de dados e o OLAP. Nesse trabalho fica caracterizada a complementaridade entre armazéns de dados e OLAP: o foco do DW está na obtenção dos dados provenientes de diversas fontes e a ênfase OLAP está na modelagem dos requisitos analíticos do usuário final.

Segundo (SINGH, 1998) para algumas organizações um armazém de dados é uma arquitetura. Para outras é um repositório de dados semanticamente consistente, separado e sem interferir nos sistemas de bancos de dados operacionais, que proporciona acessos diferenciados aos dados, possibilitando a geração de relatórios específicos.

A construção de um armazém de dados também pode ser vista como um processo contínuo que extrai dados de várias fontes heterogêneas, incluindo dados históricos e dados herdados, porém suportando consultas aleatórias e mesmo analíticas, para suporte à decisão.

Recorrendo-se a todos estes pontos de vista, um armazém de dados deve possuir pelo menos dois componentes principais (SINGH, 1998):

1. Um banco de dados de eventos históricos, o DW propriamente dito, ou uma versão departamental, denominada *datamart*; e
2. Ferramentas de análise estratégica, do tipo OLAP.

II.2.1. Definição adotada para armazém de dados

Observando-se a literatura é possível encontrar um grande número de definições para armazém de dados. A definição clássica de Inmon (INMON, 1992) é: "Um armazém de dados é um conjunto de dados baseado em assuntos, integrado, não volátil, e variável em relação ao tempo, de apoio às decisões gerenciais".

Existem outras definições que são encontradas na literatura tais como em (GILL & RAO, 1996, INMON e HACKATHORN, 1997, INMON, 1996, KIMBALL, 1996, SINGH, 1998).

II.2.2. Elementos de modelagem lógica para armazém de dados

Este texto se refere ao termo "modelagem de dados" tal como descrito em (ELMASRI & NAVATHE, 2000), pg. 532, "*The Database Design Process*". Nesta abordagem, após a elicitação de requisitos, existem basicamente três produtos, originários das fases de modelagem de dados correspondentes: o projeto conceitual, o projeto lógico e o projeto físico de dados.

Consultando-se a literatura chega-se à conclusão que a modelagem conceitual dos dados, que gera o projeto conceitual, não se aplica uniformemente aos armazéns de dados, tal como se aplica aos sistemas de produção. Isto se deve à natureza diferenciada dos sistemas do TIPO II, descritos na introdução deste capítulo.

A modelagem conceitual de dados possibilita uma independência no que se refere à escolha do modelo de dados a ser utilizado na etapa subseqüente do processo de desenvolvimento. Diversos autores sugerem que o Modelo Entidade-Relacionamento (MER) é inadequado para a modelagem conceitual de armazéns de dados, uma vez que o modelo relacional deve ter alguns dos seus fundamentos relaxados, particularmente a normalização, para que se realize modelagem para os DW (INMON *et al.*, 1997, INMON & HACKATHORN, 1997, SILVERSTON *et al.*, 1997). Porém (DATE, 2000a) alerta para o fato de que autores da área de armazéns de dados confundem projeto lógico e projeto físico dos dados, sugerindo haver falta de amadurecimento na área de modelagem de DW e citando (THOMSEN, 1997) como uma tentativa séria de preencher esta lacuna.

Na falta de uma representação de modelo conceitual para armazéns de dados, sem que haja confusão com a representação dos sistemas de produção, encontram-se na literatura diversas representações diagramáticas que reaproveitam modelos existentes, porém se referindo à representação dos depósitos de dados do armazém de dados como sendo tabelas, o que lhes confere uma natureza de representação relacional.

Se Junta a isto o fato de que uma imensa maioria dos bancos de dados atuais estarem armazenados em sistemas herdados ou então em SGBDs relacionais, e ainda ao fato de que em geral há um esforço de migração de bases de dados de sistemas herdados para os SGBDs relacionais. Deste modo é muito comum, na prática, poluir o projeto conceitual com terminologia relacional.

Quando se realiza a modelagem lógica de dados, que tem como produto o projeto lógico de dados, é necessário que se escolha então o modelo de dados alvo,

como por exemplo: hierárquico, redes, relacional ou orientado para objetos. Uma vez que esta seção se refere a uma "modelagem lógica", este texto adota como modelo de dados alvo o modelo relacional, bem como suas restrições, salvo quando o contrário estiver explícito. O modelo relacional é fundamentado nos conceitos de "dependência funcional" e de "normalização", muito apropriados na modelagem de sistemas de produção. Encontra-se na literatura uma série de justificativas para estes critérios de modelagem (ELMASRI & NAVATHE, 2000, DATE, 2000a).

A seguir é exposto o método proposto em (SILVERSTON *et al.*, 1997), para a criação de um modelo DW. O ponto de partida deste método é um modelo de dados corporativo baseado em um modelo conceitual, MER, fortemente influenciado pelo modelo lógico de dados relacional. Este método "desconstrói" os modelos conceitual e lógico corporativos, gerando o "Modelo de Dados Data Warehouse". As etapas da transformação propostas por (SILVERSTON *et al.*, 1997) são as seguintes:

1. **Remover dados puramente operacionais.** É uma etapa em que são avaliados os atributos a serem desprezados na atividade de apoio à decisão. Trata-se de uma avaliação arbitrária, em que devem ser levados em consideração os riscos da exclusão deste ou daquele atributo, segundo a chance de utilização no ambiente DW. Como todo e qualquer atributo pode vir a ser útil, é necessário avaliar aspectos físicos, como a gerência de grandes volumes de dados;
2. **Adição do elemento tempo na estrutura chave do DW, caso ainda não esteja presente.** É de fundamental importância para uma base de dados histórica, na qual o atributo temporal faça parte do identificador do registro da operação. Em alguns casos é necessário que se especifique, não só esta data como também o intervalo de validade do registro;
3. **Adição dos dados derivados apropriados.** Trata-se de uma alternativa ao cálculo de determinado índice a cada acesso ao registro físico. Fere a terceira forma normal (3NF) do modelo relacional cujo enunciado diz que "a relação não deve possuir um atributo não chave determinado por outro atributo não chave (ou por um conjunto de atributos não chave). Ou seja: não deve acontecer a dependência transitiva de um atributo não chave sobre a chave primária" (ELMASRI e NAVATHE, 2000);
4. **Transformação dos dados de relacionamento em itens de medidas de dados.** Esta prática significa que campos de outro registro são armazenados no registro

alvo, com o qual este se relaciona. Novamente a vantagem desta abordagem repousa no fato de que estes dados espelham uma realidade histórica e que não devem sofrer alterações com o tempo. Contudo é uma decisão de implementação e que costuma padecer de anomalias em transações de inserção, alteração e retirada;

5. **Acomodação de diferentes níveis de granularidade encontrados no DW.** Esta é uma decisão de projeto delicada: ao mesmo tempo em que é desejável se ter uma granularidade tão fina quanto possível, de modo que se possam ser realizadas consultas em todos os níveis de granularidade maiores, se faz necessário que se avalie o custo do armazenamento de dados históricos com granularidade fina, bem como se o sistema de produção possui a possibilidade de armazenamento na granularidade alvo. Por exemplo: uma vez que se resolve armazenar dados históricos com granularidade mensal, é impossível que se realize um desmembramento semanal posteriormente;
6. **Fusão de dados provenientes de tabelas distintas em uma única tabela.** Trata-se da implementação de uma visão, eventualmente "materializada" e que portanto pode padecer dos mesmos males descritos no item (4), podendo vir a ferir a segunda forma normal (2NF), cujo enunciado diz que "Em relações onde a chave primária contém mais de um atributo, qualquer atributo não pertencente à chave não pode ser funcionalmente dependente de parte da chave" (ELMASRI & NAVATHE, 2000);
7. **Criação de *arrays* de dados.** Fere frontalmente a primeira forma normal (1NF) que diz que "a relação não deve conter atributos não atômicos ou relações aninhadas" (ELMASRI & NAVATHE, 2000). Na prática cada célula desta nova tabela não possui atributos multivalorados, porém acontecem itens de *array* em campos diferentes do mesmo registro; e
8. **Separação dos atributos de dados de acordo com suas características de estabilidade.** Este quesito de modelagem para armazém de dados sugere a separação destes atributos em tabelas distintas, segundo o critério da afinidade na volatilidade dos atributos não-chave.

A seguir encontra-se a Tabela II.1 resumindo as três primeiras formas normais, em contraste com o método citado acima:

Tabela II.1: Resumo das formas normais baseadas em chaves primárias e a normalização correspondente (ELMASRI & NAVATHE, 2000)

Forma normal	Teste	Solução (normalização)
Primeira (1NF)	A relação não deve conter atributos não atômicos ou relações aninhadas.	Criar nova relação para cada atributo não atômico ou relação não aninhada.
Segunda (2NF)	Em relações onde a chave primária contém mais de um atributo, qualquer atributo não pertencente à chave não pode ser funcionalmente dependente de parte da chave.	Decomposição e criação de nova relação para cada chave parcial com seu(s) atributo(s) dependente(s) correspondente(s) Lembrar-se de manter a relação com a chave primária original juntamente com todos os atributos que sejam totalmente dependentes dela.
Terceira (3NF)	A relação não deve possuir um atributo não chave determinada por outro atributo não chave (ou por um conjunto de atributos não chave). Ou seja: não deve acontecer a dependência transitiva de um atributo não chave sobre a chave primária.	Decompor e criar relação que inclua os atributos não chave que determinem funcionalmente outros atributos não chave.

Armazéns de dados têm como característica principal a integração de dados oriundos de diversas fontes e formatos para consultas posteriores. Deste modo, tanto as cargas destes dados quanto às operações realizadas contra eles possuem periodicidade, granularidade e complexidade distintas dos sistemas de banco de dados ditos operacionais.

Quanto à periodicidade é necessário que se defina a freqüência em que os registros das transações dos sistemas em produção devam ser carregados. Quanto à granularidade, a decisão repousa na unidade que se deseje, ou que se possa registrar. Tecnicamente, é desejável que o grão seja tão pequeno quanto possível de modo que se possa agregar ou desagregar informação de acordo com a necessidade imediata da consulta. Esta flexibilidade possui um custo em espaço em disco, tempo de carga e gerência de arquivo morto que devem ser avaliados pelo projetista. Quanto à complexidade, é preciso prever quais dimensões e fatos podem vir a ser alvo de consultas, sabendo-se de antemão que "é impossível antecipar todas as consultas ou análises durante a fase de projeto do *datawarehouse*" (ELMASRI & NAVATHE, 2000).

Sob o ponto de vista histórico pode-se dizer que "armazéns de dados são remanescentes dos antigos sistemas geradores de relatórios baseados em *mainframes*"

(OZSU & VALDURIEZ, 1999). Naturalmente os DW incorporam muito mais funcionalidade do que um gerador automático de relatórios, porém a essência permanece, particularmente no que se refere às estruturas de dados envolvidas.

Enfocando-se os geradores automáticos de relatórios, está representado na Figura II.1 um esquema estrela para um cubo. A notação é o Diagrama de Estrutura de Dados ou de Bachman (BACHMAN, 1969).

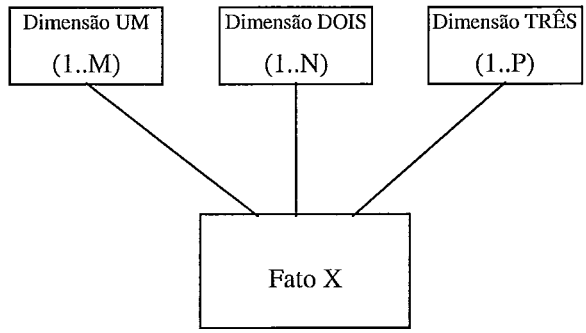


Figura II.1: Esquema estrela para modelagem de *Data Warehouse*

A Figura II.1 representa um cubo onde a dimensão "UM" possui cardinalidade M, a dimensão "DOIS" possui cardinalidade N e a dimensão "TRÊS" possui cardinalidade P. Isto significa que a base de fatos X possui M x N x P possibilidades. Como algumas destas possibilidades podem não existir, a base de fatos X pode se tornar uma matriz esparsa. Neste caso sua representação se dá pela omissão dos "não-fatos".

Por outro lado, em uma modelagem usual, espera-se que seja possível realizar operações de resumo ou *roll-up* e de detalhamento ou *drill-down* de cada dimensão, alterando-se as cardinalidades das dimensões usadas pela base de fatos dinamicamente.

Em um modelo real, estas operações e seu impacto devem ser previstos. Neste exemplo simples, estas operações estão sendo omitidas propositadamente.

Uma instância para este relatório hipotético seria, por exemplo, o relatório de cálculo dos totais de saldos em conta corrente para um banco X, para cada agência, cada cidade, cada ano e o total geral do país.

Uma solução algorítmica não otimizada, "força bruta", que hipoteticamente atua sobre um arquivo seqüencial "base de fatos X", correspondente a um "algoritmo de quebra", que gera totais parciais para cada dimensão, bem como o total geral, previamente ordenado pelas suas dimensões, está na Figura II.3. Neste caso supõe-se que:

- uma base de fatos X onde suas dimensões "UM", "DOIS" e "TRÊS", bem como suas respectivas cardinalidades M, N e P, estão pré-determinadas, ou seja: não variam; e
- estas dimensões estão previamente ordenadas.

As Figuras II.2 e II.3 são representações de instâncias procedimentais de uma consulta específica contra a base de fatos X. O que se espera de um armazém de dados com funcionalidade OLAP é a construção dinâmica de consultas. Desta maneira é possível mudar-se a granularidade de uma dimensão do mesmo esquema estrela (o mesmo modelo estrutural de dados), dinamicamente. Um primeiro passo para que se possa alcançar tal objetivo é expressar as mesmas consultas em um modelo lógico intermediário que possibilite a tradução da consulta em uma linguagem alvo, usando-se por exemplo, SQL. Na próxima seção, são abordados elementos que viabilizam esta modelagem lógica. A Figura II.2. representa o processamento procedimental de geração de relatório com o cálculo de totais parciais para cada dimensão definida, impresso em cada rodapé correspondente, bem como o cálculo de total geral, no rodapé geral, usando-se o método de (JACKSON, 1975).

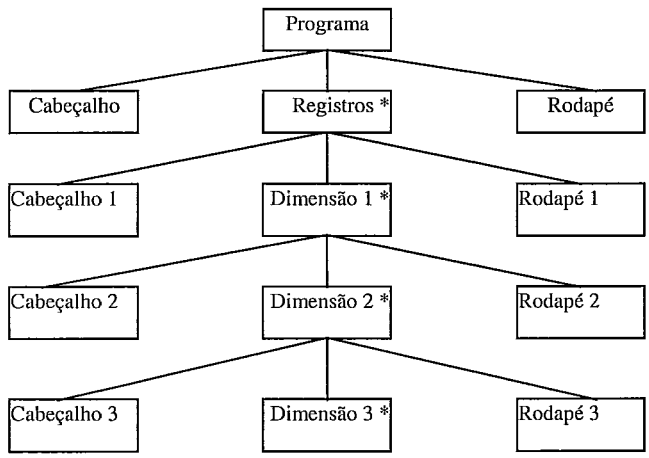


Figura II.2: Diagrama de Jackson correspondente ao "esquema estrela"

```
Inicio
tipo R = registro
 Um_Tipo_Qualquer: D1,D2,D3;
 Um_Tipo_Qualquer: A, B, C; /* outros atributos */
fim-registro;
R: Reg;
/* outras declarações... */
Abrir arquivo FatoX
Ler Reg em FatoX
Aux1 := Reg.D1; Aux2 := Reg.D2; Aux3 := Reg.D3;
Trata o cabeçalho geral
Enquanto não é fim de arquivo
        Trata cabeçalho da dimensão1
        Enquanto Aux1 = Reg.D1
                & não é fim de arquivo
                Trata cabeçalho da dimensão2
                Enquanto Aux2 = Reg.D2
                        & Aux1 = Reg.D1
                    & não é fim de arquivo
                        Trata cabeçalho da dimensão3
                        Enquanto Aux3 = Reg.D3
                    & Aux2 = Reg.D2
                            & Aux1 = Reg.D1
                                não é fim de arquivo
                                PROCESSA "CUBO"
                                Ler Reg em FatoX
                        Aux1 := Reg.D1;
                                Aux2 := Reg.D2;
                                Aux3 := Reg.D3;
                        Fim do Enquanto
                        Trata rodapé da dimensão 3
                Fim do Enquanto
                Trata rodapé da dimensão 2
        Fim do Enquanto
        Trata rodapé da dimensão 1
Fim do Enquanto
Trata rodapé geral
Fechar arquivo FatoX
Fim.
```

Figura II.3: Algoritmo típico que corresponde ao diagrama de Jackson

II.2.3. Elementos de modelagem lógica para processamento analítico

O modelo multidimensional de dados, onde os dados são representados por matrizes multidimensionais de valores numéricos, tais como vendas e receita (OZSU & VALDURIEZ, 1999), usado na representação estrutural de DWs, pode ser representado em esquemas "estrela", "*snowflake*" ou "*constellation*" (HAN & KAMBER, 2001).

O esquema estrela consiste de uma tabela de fatos, com uma única tabela para cada dimensão. O esquema *snowflake* é uma variação do esquema estrela no qual tabelas de dimensão são organizadas em hierarquias (KIMBALL, 1996, ELMASRI & NAVATHE, 2000). Já o esquema *constellation* consiste em diversas tabelas de fatos que compartilham dimensões (HAN & KAMBER, 2001).

Um esquema estrela típico que represente um cubo, possui três tabelas de dimensão e uma tabela fato. Quando existem mais de três tabelas de dimensão pode ser usado o termo "hipercubo", porém em geral usa-se apenas "cubo".

A Figura II.4. é um exemplo na qual as dimensões são agência, cidade e ano, sendo a tabela de fatos saldo_em_reais, para um banco. Trata-se de um reticulado isomorfo ao esquema estrela da Figura II.1 e representa os "cubóides" passíveis de serem gerados. Como neste exemplo existem três dimensões, o número possível de cubóides é igual a 2**3 = 8.

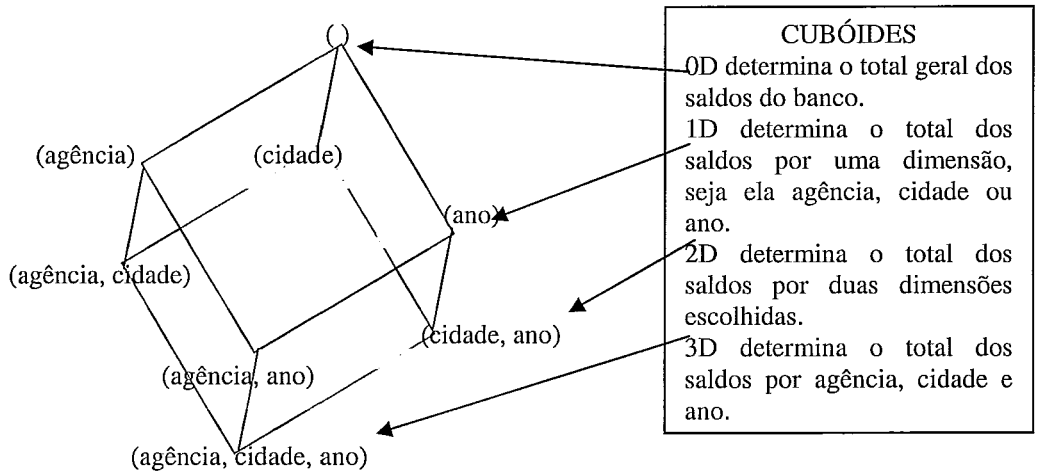


Figura II.4: Reticulado de cubóides

Tradicionalmente este tipo de modelagem é realizado com dimensões que representam tempo e espaço. Neste exemplo a dimensão tempo é representada por *ano*, porém é possível existir uma hierarquia nesta dimensão, do tipo *dia* < *semana* < *mês* < *trimestre* < *semestre* < *ano*. De modo análogo à dimensão espaço representada por *cidade* neste exemplo, poderia ser criada em uma das seguintes instâncias *bairro* < *cidade* < *estado* < *região* < *país.*

Evidentemente o que se espera de um sistema com facilidades OLAP, diferentemente da técnica proposta nas Figuras II.2 e II.3, é que se possa variar a granularidade da consulta dinamicamente de acordo com a instância escolhida na hierarquia determinada para uma dimensão. A escolha do grão mais fino, no caso da Figura II.4, *dia* para tempo ou *bairro* para espaço significa que existem dados armazenados, pelo menos com esta granularidade, o que oferece maior flexibilidade de consulta por um lado, e maior ocupação de espaço em disco, por outro.

Além da escolha do modelo lógico de dados específico para o DW, que representa o lado servidor ou *backend* das transações de consulta, existe a necessidade da modelagem do lado cliente ou *frontend* que consiste das operações OLAP. A seguir estão descritas as principais operações OLAP citadas em (HAN & KAMBER, 2001), do modelo de dados multidimensional, bem como uma formalização proposta em (AGRAWAL *et al.*, 1997) :

1. *Roll-up:* a o operação r*oll-up* (também conhecida como *drill-up*) executa agregação em um cubo de dados de duas formas: subindo uma hierarquia de conceito em uma dimensão; ou reduzindo uma dimensão. Quando a operação *roll-up* é executada por redução de dimensão, uma ou mais dimensões são suprimidas do cubo de origem. Ilustrando:
   - Caso 1: usando-se o reticulado de cubóides da Figura II.4 e escolhendo-se a dimensão relativa a "espaço", temos a seguinte hierarquia de granularidade, que é também uma relação de ordem: *bairro* < *cidade* < *estado* < *região* < *país.* Uma operação do tipo *Roll-up* agrega valores de qualquer granularidade considerada "menor" para uma granularidade "maior" Neste caso supondo-se ter os totais de saldos das agências bancárias por cidade, obter-se-ia os totais por região, por exemplo;
   - Caso 2: poder-se-ia omitir a dimensão "espaço", obtendo-se totais por período de tempo, digamos os totais de saldo da década de 90, omitindo-se as localidades específicas e gerando-se apenas totais gerais, no caso por país;
2. *Drill-down:* A operação *drill-down* é a inversa da operação *roll-up*. *Drill-down* pode ocorrer pelo detalhamento de uma hierarquia de conceito de uma dimensão ou através da introdução de dimensões adicionais. Como o *drill-down* acrescenta mais detalhe aos dados, também pode ser executada pelo acréscimo de novas dimensões a um cubo. Para ilustrar o caso 1: tendo-se totais de saldo de agências bancárias por região, obter-se-ia os totais por cidades. Além desta possibilidade seria possível adicionar uma nova dimensão ao modelo, digamos, "tipo de cliente";
3. *Slice and dice:* A operação *slice and dice* executa uma seleção em uma ou mais dimensões de um cubo de origem, resultando em um sub cubo; e
4. *Pivot (rotate):* a operação de *pivot ou rotate* é uma operação de auxílio à visualização em que é possível a observação de um cubo em novas perspectivas, privilegiando eixos alternativos ao eixo ora visualizado.

Em (AGRAWAL *et al.*, 1997) há um formalismo para a modelagem de bancos de dados multidimensionais, onde são propostos "um modelo de dados baseado em hipercubo e algumas operações algébricas que fornecem o fundamento semântico para bases de dados multidimensionais e estendem sua funcionalidade".

Essa proposta se auto define como um modelo lógico de dados e "não obriga o uso de nenhum mecanismo de armazenamento" em particular, preenchendo a lacuna da modelagem do lado cliente ou *frontend* das operações OLAP, de forma a oferecer operadores projetados de maneira a serem traduzidos para SQL, no caso de uma implementação ROLAP (*Relational* OLAP), bem como também podem ser implementados sobre uma máquina de banco de dados multidimensional ou MOLAP. A seguir são descritos os fundamentos, a notação e os operadores definidos em (AGRAWAL *et al.*, 1997), onde pode ser encontrado o formalismo de cada operador. A concepção deste modelo de dados foi realizada respeitando os seguintes fundamentos:

- **Tratamento simétrico para dimensões e fatos.** Apesar do esquema escolhido, *estrela*, *snowflake ou constellation*, definir dimensões e fatos, este modelo permite visualização mais flexível, possibilitando a escolha de eixos e rotação para a visualização dos dados livremente;
- **Flexibilidade na agregação de múltiplas hierarquias e agregados aleatoriamente.** O modelo possibilita que se escolha dinamicamente a granularidade do dado que se deseja visualizar para um dado eixo de dados, tal como no exemplo: *dia* < *semana* < *mês* < *trimestre* < *semestre* < *ano*;
- **Cada operador do modelo é definido no cubo e produz como saída um novo cubo.** Portanto os operadores obedecem à propriedade do fechamento e podem ser reordenados livremente. Segundo os autores, esta natureza algébrica do modelo possibilita a otimização das consultas multidimensionais;
- **Os operadores obedecem ao princípio da minimalidade.** Nenhum operador pode ser expresso em termos de outros, bem como a ausência de qualquer dos operadores sacrifica a funcionalidade do modelo; e
- **Este modelo separa a gerência da interface gráfica do usuário**, usada por analistas de negócios, **da gerência do armazenamento físico dos dados**, administrado pela corporação.

Neste modelo os dados são organizados em um ou mais hipercubos. Um cubo possui os seguintes componentes (AGRAWAL *et al.*, 1997):

- K dimensões e para cada dimensão um nome D_i , um domínio dom_i dos quais são definidos valores;
- Os elementos são definidos como um mapeamento E(C) de dom_1 X dom_2 X ... X dom_k para uma n-tupla , 0 ou 1. Portanto $E(C)(d_1, ..., d_k)$ se refere ao elemento na posição $d_1, ..., d_k$ do cubo C. A notação d_is se refere a valores e não posições; e
- Parte dos metadados é uma n-tupla de nomes onde cada um dos nomes descreve um dos membros de um elemento da n-tupla do cubo. Caso não possua elementos nas n-tuplas, então esta é a descrição de uma tupla vazia.

Os elementos de um cubo podem ser 0, 1 ou uma n-tupla $< X_1, ..., X_n >$. Se o elemento correspondente a $E(C)(d_1, ..., d_k)$ é 0, então esta combinação de valores de dimensões não existe na base de dados. A ocorrência de 1 significa existência de uma combinação particular. Finalmente uma n-tupla indica que existe informação adicional para esta combinação de valores. Se qualquer dos elementos de um cubo é 1 então nenhum dos elementos pode ser uma n-tupla e vice-versa. Caso todos os elementos do cubo sejam 0, então o cubo está vazio. Caso o domínio de uma dimensão não possua valores então o cubo também é considerado vazio.

São definidos operadores usando-se um cubo C com k dimensões. Usa-se $D_1, ...,$ D_k para referir-se às dimensões e $dom_i(C)$ para referir-se ao domínio da dimensão. A seguir são descritos os operadores do modelo:

- *Push:* este operador é usado para converter dimensões em elementos, que possam ser manipulados usando-se a função $f_{elemem.}$, suprimindo a dimensão. Veja o caso 2 da operação *roll-up*. Este operador é necessário para que se possa tratar fatos e dimensões uniformemente;
- *Pull:* este operador é o inverso do anterior: converte elementos em dimensões. Cria uma nova dimensão para um membro específico de cada elemento. É útil na conversão de um elemento em dimensão de modo que este elemento possa ser usado em operações de *merge* e *join*. Também é importante no tratamento simétrico entre dimensões e fatos;
- *Merge:* este operador é um operador de agregação. Esta operação é utilizada de modo que e produza uma dimensão de domínio menor, partindo-se de dimensões com múltiplos elementos. Veja o caso 1 da operação *roll-up*. Um único elemento do cubo resultante pode ser gerado da operação de "merge" de cada dimensão, onde múltiplos elementos são mapeados para ele;

- *Destroy Dimension:* este operador remove a dimensão D que possua em seu domínio um único valor. A presença de um único valor implica que, para as k-1 dimensões restantes, exista um único cubo de dimensão k-1. Portanto caso a dimensão D seja eliminada, então o espaço dimensional k-1 resultante é ocupado por este único cubo;
- *Restriction:* o operador *restrict* é aplicado sobre uma dimensão de um cubo removendo os valores do cubo que não satisfaçam a restrição. Trata-se do operador que realiza a operação *slicing and dicing* de um cubo, na terminologia de bancos de dados multidimensional;
- *Join:* este operador relaciona informação entre dois cubos. O resultado da junção dos cubos C m-dimensional e C1 n-dimensional em k dimensões, chamadas dimensões de junção, é o cubo C_{rsp}, com m + n – k dimensões. Cada dimensão de junção D_i de C combina com exatamente uma dimensão D_{xi} de C1. A dimensão resultante correspondente terá valores que são a união dos valores D_i e D_{xi}.

II.3. Mineração de Dados

Enquanto o armazém de dados é a tecnologia predominante no auxílio à obtenção dos dados necessários ao processo KDD, a mineração de dados é a tecnologia predominante no auxílio à interpretação destes dados.

Esta interpretação é uma atividade essencialmente humana. A mineração de dados auxilia na aquisição semi-automática de conhecimento, gerando este conhecimento em alguma das suas formas de representação.

Nas palavras de (CARVALHO, 2001) "Sistemas OLAP emitem respostas para algumas perguntas do tipo *que dados se encaixam neste padrão?* enquanto os sistemas de *datamining* respondem à pergunta *que padrões existem nestes dados?*".

Dados, informação e conhecimento são conceitos difíceis de definir. Em geral define-se um destes conceitos em função de outro, como por exemplo em (PRESSMAN, 1992): "dado é a informação bruta" e "sem associatividade"; "informação é o dado trabalhado" ou "com associatividade dentro de um contexto"; e ainda "conhecimento é a informação associada em múltiplos contextos".

Uma interpretação possível é aquela que diz que existe uma hierarquia de complexidade onde os dados constituem a parte mais simples desta hierarquia e o

conhecimento constitui a parte mais complexa.

Outra forma de abordar estes termos se refere à maneira como vem sendo empregados. O termo "processamento de dados" vem sendo utilizado desde os primórdios da Ciência da Computação. Refere-se a tipos bem conhecidos de processos, algoritmos, estruturas de dados e arquivos. Já o termo "tecnologia da informação" vem sendo utilizado mais recentemente. De uma maneira geral existe uma associação bastante comum entre a Tecnologia da Informação e o uso de SGBDs. O termo "representação do conhecimento" vem sendo usado pela Ciência da Computação com um enfoque bem definido (PASSOS, 1989). Quatro são os tipos de recursos mais utilizados para representar conhecimento: redes semânticas, quadros e roteiros, lógica matemática e regras de produção.

As redes semânticas constituem um tipo de diagrama criado a partir de um conjunto limitado de arcos e nós. Quaisquer diagramas que auxiliam na especificação de sistemas podem ser incluídos nesta categoria. Por exemplo, nos diagramas de fluxo de dados os nós são: processos (bolhas), entidades externas (retângulos) e depósitos de dados (linhas paralelas). Os arcos são os fluxos de dados. Quadros e roteiros representam o conhecimento em termos de objetos com o sua estrutura (quadro) e comportamento (roteiro). Acredita-se que esta forma de representação é muito semelhante à orientação para objetos.

A lógica matemática é uma estrutura abstrata cuja utilização permite um processo rigoroso de raciocínio. Uma lógica é definida através de uma linguagem simbólica, cujos símbolos possuam uso e significado bem definido. Na lógica clássica proposicional, estes símbolos representam sentenças e operadores que agem sobre estas sentenças. As operações básicas são a negação ("não"), a conjunção ("e") e a disjunção ("ou") lógicas.

As regras de produção são a maneira mais utilizada de representar conhecimento neste enfoque. O conhecimento é representado através do par "condição-ação". As regras possuem duas partes: uma antecedente ("se") e outra conseqüente ("então"), onde ambas podem conter os operadores da lógica.

Basicamente, se atribuímos algum significado especial a um dado, este se transforma em uma informação (ou um fato). Se os especialistas no domínio do problema elaboram uma norma (ou regra), a interpretação do confronto entre este fato (informação) e esta regra (norma) podem constituir um conhecimento. Exemplificando:

em um cadastro de pessoal, a data de aniversário de um funcionário é um dado. A qualificação de um funcionário como aniversariante do mês é uma informação (fato). A norma que concede um abono salarial aos aniversariantes é uma regra. A identificação do aniversariante como um daqueles que tem direito ao abono é um "conhecimento".

A Mineração de Dados não se dá de maneira isolada em um sistema de informação. São necessárias diversas etapas de preparação dos dados antes desta tarefa, bem como a consolidação e o armazenamento do conhecimento obtido após a sua conclusão.

O processo que contém todas estas etapas é o processo KDD. Este termo foi criado para se referir ao processo de encontrar conhecimento em dados, e para enfatizar as aplicações de métodos de mineração de dados em particular .

Pode-se dizer que a mineração de dados é uma etapa do processo KDD, usada para auxiliar os especialistas em uma determinada área a atualizar suas bases de conhecimento na busca de alguma vantagem competitiva em seu negócio.

Segundo (CARVALHO, 2001), "cinco técnicas gerais abraçam didaticamente todas as outras formas de apresentação e permitem uma visão mais global e apropriada para uma introdução ao assunto: classificação, estimativa, previsão, análise de afinidade e análise de agrupamentos". Já segundo (ELMASRI & NAVATHE, 2000) o objetivo específico da mineração de dados é o uso de algoritmos e estruturas de dados para alcançar uma das seguintes categorias de solução de problemas: predição, identificação, classificação e otimização. Existem ainda outras referências que exploram as formas de apresentação da teoria. A seguir encontra-se uma breve descrição das categorias citadas em (ELMASRI & NAVATHE, 2000), sem prejuízo da citação de outros autores. Esta escolha é intencional pela analogia que é feita com o do *market basket analysis* em (AGRAWAL *et al.*, 1993).

A **predição** se refere à possibilidade de previsão do comportamento de determinados atributos ao longo do tempo, tal como o impacto que a descontinuidade da venda de um produto pode causar na venda de outro. Determinados padrões podem ser usados no reconhecimento de um item, evento ou atividade. Este reconhecimento pode se dar na **identificação** de fraudes, no uso de cartões de crédito. Grandes redes de varejo precisam estabelecer categorias de consumidores. Neste caso é usada a **classificação**. A **otimização** é um problema típico da pesquisa operacional que pode ser

abordado na mineração de dados, na busca da administração de recursos limitados, tal como o espaço nas prateleiras das lojas do varejo.

II.3.1. Definição adotada para mineração de dados

Data Mining ou Mineração de Dados é a tarefa do estabelecimento de novos padrões de "conhecimento", geralmente imprevistos, partindo-se de uma massa de dados previamente coletada e preparada para este fim.

Os sistemas de mineração de dados existentes geram conhecimento pelo menos nas seguintes formas:

- **Regras associativas**. Consiste em encontrar itens em uma transação que determinem a presença de outros itens na mesma transação . Exemplo: sempre que se compra pão e leite, compra-se manteiga. Regras associativas possuem formato semelhante ao de regras de produção. A regra anterior pode ser representada como segue: pão, leite => manteiga;
- **Hierarquias de classificação**. Consiste na criação de um modelo baseado em dados conhecidos. É possível utilizar-se deste modelo para analisar o porque de uma dada classificação, ou mesmo classificar novos dados partindo-se de uma classificação existente. O objetivo é trabalhar a partir de um conjunto de eventos ou transações para criar uma hierarquia de classes. Exemplo: uma população pode ser dividida em 5 categorias de limite de crédito, baseado no histórico de transações de crédito anteriores. Qualquer classificação pode ser representada visualmente no formato de árvore;
- **Padrões Seqüenciais**. Consiste em encontrar padrões ou comportamento previsível em um período de tempo. Isto significa que um comportamento particular em um dado momento pode ter como conseqüência outro comportamento ou seqüência de comportamentos dentro de um período de tempo. Trata-se de uma situação parecida com a das regras associativas, porém com um componente temporal. Exemplo: sempre que uma mulher jovem compra sapatos de couro, comprará cinto(s) e bolsa(s) nos próximos 30 dias;
- **Padrões em Séries Temporais**. Consiste em obter todas as ocorrências de subseqüências similares em uma base de dados. Exemplo: bolsas e sapatos fechados de couro possuem padrão de vendas similar no inverno, o que não

ocorre no verão, quando o padrão de similaridade ocorre entre bolsas e sandálias de couro; e

- **Categorização e Segmentação.** Consiste em agrupar registros que contenham características similares. Exemplo: os consumidores podem ser categorizados por estilo e ter estas categorias segmentadas para a compra de um novo produto entre "pouco", "medianamente" ou "muito" compradores.

As duas técnicas de mineração de dados enfocadas neste trabalho são a "mineração de regras associativas" e a "mineração de padrões seqüenciais" a serem detalhadas a seguir.

II.3.2. Mineração de Regras Associativas

Segundo (CARVALHO., 2001) "determinar que fatos ocorrem simultaneamente com probabilidade razoável (co-ocorrência) ou que itens de uma massa de dados estão presentes juntos com uma certa chance (correlação) são tarefas típicas da análise de afinidade". A obtenção de **regras associativas** se dá em uma situação em que se deseja obter a afinidade entre itens que possam ocorrer conjuntamente em uma única transação. A "afinidade" entre itens é representada por regras, como segue:
"**Se** compro pão e leite **então** compro manteiga".
Ou seja:

[pão] AND [leite] ==> [manteiga]

O antecedente da regra "[pão] AND [leite]", ou premissa, é também chamado de corpo (*body*) da regra. O conseqüente da regra "[manteiga]", ou conclusão, é também chamado de cabeça (*head*) da regra.

Este é um exemplo clássico, trata-se do *market basket analysis*. Nesta aplicação deseja-se saber, por exemplo, qual o impacto que determinado item possui na compra de outro. Para isto é preciso que se definam alguns parâmetros. São eles o suporte, a confiabilidade e o tamanho máximo de regra.

Com o advento das embalagens com códigos de barra e das leitoras ópticas, é possível o registro de cada compra (cesta de compras ou *basket*). Neste caso, uma transação é uma compra e "item" se refere a um item comprado. A repetição de itens em uma transação conta como um único item, com a especificação da quantidade comprada.

O suporte é obtido pela divisão do número de transações que suportam a regra, ou seja, o número de transações em que os itens analisados satisfazem as regras, pelo número total de transações. A confiabilidade é determinada pela divisão do número de transações que suportam a regra, pelo número de transações que suportam apenas o corpo da regra.

Enquanto o suporte se refere à significância estatística da regra, a confiabilidade é uma medida relativa à sua "força". O tamanho máximo da regra é o número máximo de itens que podem ocorrer na regra. No exemplo anterior o tamanho da regra é 3: esta regra possui 2 itens em seu corpo e um único item em sua cabeça.

O objetivo do processo de mineração neste caso é a obtenção de regras associativas em que se determine previamente um suporte e confiabilidade mínimos, bem como um tamanho máximo de regra. Além destas restrições, é possível se considerar apenas alguns itens dentre um universo de itens.

Seja $I = I_1, I_2, ..., I_m$ um conjunto de atributos binários, chamados itens. Seja T uma base de dados de transações. Cada transação t é representada como um vetor binário, com t[k] = 1 caso o item I_k tenha sido comprado na transação t e t[k] = 0, caso contrário.

Há uma tupla para cada transação na base de dados. Seja X um conjunto de alguns itens em I. Diz-se que uma transação t satisfaz X se, para todos os itens I_k em X, t[k] = 1.

A seguir estão descritos formalmente os elementos do modelo:

- **Regra Associativa:** é a implicação na forma X => I_j, onde X é um conjunto de alguns itens em *I* e I_j é um único item em *I* que não está presente em X. A regra X => I_j é satisfeita no conjunto de transações T com fator de confiança $0 \leq c \leq 1$, se e somente se, pelo menos *c*% das transações T que satisfaçam X, também satisfaçam I_j. A notação X => I_j | *c* especifica que o fator de confiança desta regra é *c*. Dado um conjunto de transações T o interesse usual é a geração de todas as regras que satisfaçam dois tipos adicionais de restrições: restrições sintáticas e restrições de suporte.
- **Restrições Sintáticas:** estas restrições se referem aos itens que podem aparecer em uma regra. É possível que o interesse recaia sobre um item I_x, específico, que se espera acontecer no conseqüente da regra. Outra possibilidade é a identificação de regras em que o item I_y apareça no antecedente da regra. O

modelo prevê a combinação destes dois tipos de restrição: a identificação de itens predeterminados tanto no antecedente, quanto no conseqüente da regra.

- **Restrições de Suporte:** estas restrições se referem ao número de transações T que suportam a regra. O suporte de uma regra é definido pela fração de transações em T que satisfaçam a união dos itens contidos no antecedente e no conseqüente da regra.

Maiores detalhes sobre o formalismo, bem como os algoritmos correspondentes a este método são encontrados em (AGRAWAL *et al.*, 1993).

II.3.3. Mineração de Padrões Seqüenciais

Padrões seqüenciais representam seqüências de conjuntos de itens que ocorrem em transações diferentes. A concepção deste conceito (AGRAWAL & SRIKANT, 1995) foi direcionada para solucionar um problema de natureza exógena, que é a descoberta de padrões de itens de uma cesta de compras (*market basket analysis*) : seqüências de conjuntos de itens que ocorrem nas transações de diferentes consumidores, com determinada seqüência e ordem especificadas. Formalizando:

- Dada uma base de dados *D* de transações de consumidores, onde cada transação possui os seguintes campos: identificação do consumidor, hora da transação e itens consumidos. As restrições do modelo são as seguintes: nenhum consumidor possui mais do que uma transação com a mesma hora de transação e não são consideradas as quantidades de itens comprados, apenas a ocorrência da compra do item;
- Neste contexto um conjunto de itens é não vazio e uma seqüência é um lista ordenada de conjuntos de itens. A cada item corresponde um inteiro, para efeito de facilitar o processamento;
- Um conjunto de itens é representado na forma $(i_1, i_2, ..., i_m)$, onde i_j é um item. Uma seqüência é representada na forma $<s_1, s_2, ..., s_n>$, onde s_j é um conjunto de itens;
- Uma seqüência $<a_1, a_2, ..., a_n>$ de itens está contida em outra seqüência $<b_1, b_2, ..., b_m>$ de itens se existirem inteiros $i_1<i_2<...<i_n$ tais que $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, ..., $a_n \subseteq b_{in}$, e;

- Em um conjunto de seqüências, uma seqüência é maximal se não está contida em nenhuma seqüência.

Maiores detalhes sobre o formalismo, bem como os algoritmos correspondentes a este método são encontrados em (AGRAWAL & SRIKANT, 1995). Na próxima seção é enfocado um método que se utiliza da mineração de padrões seqüenciais na obtenção de modelos de processos.

II.3.3.1. Mineração de Modelos de Processos

A proposta em (AGRAWAL *et al.*,1998) apresenta uma abordagem para um sistema que constrói modelos de processos, justificando este objetivo pela crescente utilização do paradigma de *workflow* nos empreendimentos contemporâneos, na modelagem dos processos de negócios.

A representação de conhecimento escolhida recaiu sobre a teoria dos grafos, uma vez que na maior parte dos casos, a representação de *workflow* acontece por meio de diagramas. Desta maneira foram realizados experimentos sobre *logs* provenientes de instâncias de *workflow* armazenados, na ocasião, em um ambiente IBM *Flowmark.*

O objetivo do experimento foi o aperfeiçoamento de um *workflow* já existente, partindo-se da análise dos logs gerados pelo rastreamento do uso do *workflow* alvo, armazenado neste ambiente. A justificativa desta pesquisa se baseou no fato de que "um processo de negócio especifica a maneira na qual os recursos de um empreendimento são usados" bem como "o desempenho de um empreendimento depende da qualidade e precisão de um processo de negócio".

Segundo (AGRAWAL *et al.*,1998) sistemas de *workflow* assumem que um processo pode ser dividido em unidades menores, atômicas, chamadas atividades. No funcionamento de um processo alguém pode ativar um conjunto (ou subconjunto) de atividades. Além disso pode haver dependências entre atividades diferentes.

Nesta teoria a principal abordagem usada em sistemas de *workflow* é a modelagem do processo como um *Direct Acyclic Graph.* Seus nós representam as atividades atômicas e seus arcos representam as dependências entre eles. Em outras palavras, se a atividade A tem de ser executada antes da atividade B, deve existir um arco orientado de A para B no grafo.

Na prática algumas execuções do processo podem incluir uma dada atividade enquanto outras, não. Portanto, cada arco A=>B é lido como uma função booleana que

determina quando há um fluxo de controle de A para B.

Os sistemas de *workflow* atuais assumem que o modelo de processo está disponível e a tarefa principal do sistema é garantir que todas as atividades sejam executadas na ordem correta, bem como o processo terminar com sucesso.

O usuário é estimulado a fornecer o modelo de processo. A construção do modelo de processo do negócio desejado a partir de um modelo de execução não estruturado é tarefa difícil, custosa e, na maior parte das vezes, é necessária a contribuição de um especialista.

Segundo (AGRAWAL *et al.*,1998) a contribuição deste trabalho foi apresentar uma nova abordagem para a solução do problema em que, dado um *log* de execuções não estruturadas de um processo, é gerado um modelo gráfico do processo. O grafo resultante representa o processo de negócio que satisfaz os seguintes objetivos:

- **É completo:** o grafo deve preservar todas as dependências entre atividades que estejam presentes no *log*. Deve permitir todas as execuções do processo presentes no *log*;
- **É não redundante:** o grafo não deve introduzir dependências espúrias entre atividades; e
- **É mínimo:** para o melhor entendimento o grafo deve possuir um número mínimo de nós.

Um modelo simplificado da teoria proposta em (AGRAWAL *et al.*,1998) é encontrado em (LEYMANN & ROLLER, 2000) e sua representação encontra-se nas Tabelas II.2 e II.3. Trata-se de um modelo de como obter um modelo de processos partindo-se de um log de transações oriundo de um Sistema Gerenciador de *Workflow* (SGW). O exemplo assume um processo de negócio que contém quatro atividades discretas.

Supõe-se que o processo entre em produção e execute por quatro vezes. Os processos estão representados na Tabela II.2 com os nomes P1, P2, P3 e P4. Para cada um dos processos o SGW grava no log a informação relativa ao momento de início e fim de cada atividade. Estes momentos estão representados na tabela iniciando-se no tempo 1 e assim sucessivamente. Atividades que ocorram no mesmo intervalo configuram paralelismo.

No exemplo da Tabela II.2, para o processo P1, a atividade B é processada no intervalo de tempo (3,7) e a atividade C é processada no intervalo de tempo (4,5). Isto

configura que as atividades B e C ocorrem em paralelo. O momento de "início" de um conjunto de atividades corresponde ao menor valor de início de todas as atividades deste conjunto. De modo análogo, o momento de "fim" de um conjunto de atividades é o maior valor de fim de todas atividades.

Tabela II.2: Informação de atividades derivada do "*audit trail*" (LEYMANN & ROLLER, 2000), pg. 46

Modelo de Processos	Processo	Atividade	Intervalo
P	P1	A	1,2
P	P1	B	3,7
P	P1	C	4,5
P	P1	D	8,9
P	P2	A	1,3
P	P2	B	5,6
P	P2	C	4,6
P	P2	D	7,8
P	P3	A	1,3
P	P3	B	5,6
P	P3	C	4,6
P	P3	D	7,8
P	P4	A	2,3
P	P4	B	4,6
P	P4	C	5,6
P	P4	D	7,9

A Tabela II.3 representa as relações de precedência no tempo. Deste modo O processo P é representado por cada uma de suas instâncias P1, P2, P3 e P4. Para cada uma destas instâncias, ou execução do processo P, são representadas as relações de precedência no tempo, onde as atividades do lado esquerdo da "seta", ou antecedentes, precedem estritamente as atividades do lado direito da seta, ou conseqüentes. As atividades representadas entre chaves possuem algum nível de paralelismo em sua execução, não havendo relação de precedência estrita.

Estes formatos de armazenamento favorecem a aplicação de tecnologias de *datawarehousing* e *datamining* tanto no armazenamento quanto na busca e interpretação dos padrões de precedência de atividades de uma instância de um mesmo processo, bem como relações interprocessos. O formalismo proposto pode ser encontrado em (AGRAWAL *et al.*,1998).

Tabela II.3: Processamento da ordem de atividades (LEYMANN & ROLLER, 2000), pg. 46

Nome do processo	Instância	Relações de precedência
P	P1	A ->{B,C} {B,C}->D
P	P2	A ->{B,C} {B,C}->D
P	P3	A->B B->C C->D
P	P4	A ->{B,C} {B,C}->D

II.4. Prospecção de Conhecimento em Banco de Dados

O que caracteriza o processo KDD é o aumento na automação da identificação e do reconhecimento de padrões. O desenvolvimento deste processo está sendo possível através de uma nova geração de ferramentas e técnicas para análise de dados em bancos de dados. A Mineração de Dados é um passo no processo de Prospecção que consiste da utilização de algoritmos que produzem uma enumeração particular de padrões. Assim, as fases do processo de Prospecção de Conhecimento em Banco de Dados são identificadas na literatura (FAYYAD *et al.*, 1996), como segue:

1. Desenvolver a compreensão do domínio da aplicação, o conhecimento anterior relevante, e os objetivos do usuário final;
2. Criar um conjunto de dados alvo onde a prospecção deverá ser efetuada;
3. Realizar a limpeza e o pré-processamento dos dados;
4. Realizar a redução e projeção de dados reduzindo o número efetivo de variáveis consideradas ou encontrar representações não variáveis para os dados;
5. Escolher as tarefas de mineração de dados: decidindo se o objetivo do processo KDD é classificação, regressão, clusterização ou outro;
6. Escolher os algoritmos de mineração de dados, selecionando métodos para serem usados na busca de padrões nos dados;
7. Minerar dados;
8. Interpretar padrões obtidos; e
9. Consolidar conhecimento obtido.

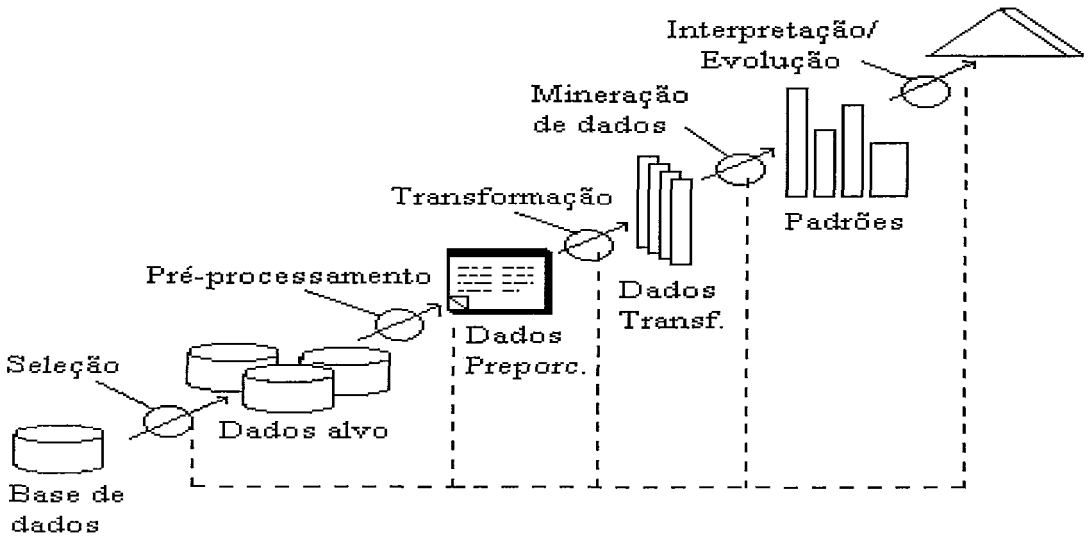


Figura II.5: O processo KDD (FAYYAD *et al.*, 1996)

Certamente este tipo de processo já vem sendo realizado há anos. O que diferencia o processo KDD atual das alternativas do passado, em geral de processamento de dados, são as novas tecnologias envolvidas, que, como já citado, possibilitam um melhor nível de automação do processo como um todo.

Levando-se em consideração as definições disponíveis chega-se à conclusão de que um armazém de dados é um Sistema de Informação que possibilita a implementação dos passos 1 a 4 do processo KDD (1. Desenvolver a compreensão do domínio da aplicação; 2. Criar um conjunto de dados alvo; 3. Realizar a limpeza dos dados; e 4. Realizar a redução e projeção de dados).

Já a mineração de dados, visto como parte do processo KDD, possibilita a implementação dos passos 5 a 7 (5. Escolher as tarefas de mineração de dados; 6. Escolher os algoritmos de mineração de dados; e 7. Minerar dados). Os passos 8 e 9 (8. Interpretar padrões obtidos; e 9. Consolidar conhecimento obtido) pertencem ao domínio dos especialistas da aplicação, podendo ser automatizados, por exemplo, através de Sistemas Especialistas que utilizem Regras de Produção. Em (HAN & KAMBER, 2001) há um comentário pertinente sobre o termo KDD, em que se admite que a mineração de dados é uma etapa do processo KDD como um todo, porém tornou-se um termo mais popular sendo compreendido como o "processo de obtenção de conhecimento interessante a partir de grandes quantidades de dados armazenados, seja em bases de dados, armazéns de dados ou outros repositórios de informação". Para efeito deste trabalho e, evitando a possível polêmica de nomenclatura, adota-se o processo KDD tal como descrito nesta seção.

Capítulo III - Regras de Negócios e Fluxos de Trabalho

III.1. Introdução

A Gestão de Conhecimento é uma disciplina de concepção recente, apesar de estar presente na civilização humana desde seus primórdios, principalmente no que se refere aos repositórios de conhecimento por excelência, as bibliotecas e as pessoas envolvidas na sua administração.

A evolução no suporte da produção, organização, distribuição e acesso à informação, proporcionado pelas novas tecnologias, estimula a pesquisa e o desenvolvimento nesta área.

Neste contexto cabe citar a dicotomia "conhecimento tácito versus explícito" (DAVENPORT & PRUSAK, 1998): é evidente que não é possível o registro de todo tipo de conhecimento humano explicitamente.

Neste trabalho está sendo enfocado o conhecimento explícito, passível de ser registrado em meio magnético, estruturado em uma das formas de representação do conhecimento descritas no capítulo II.

São apresentados modelos, conceitos e exemplos, caracterizando-se regras de negócios e fluxos de trabalho como formas de representação do conhecimento. Para cada uma destas é escolhido um modelo adequado para os objetivos deste trabalho: a configuração de formatos de informação e conhecimento a serem obtidos.

Como o método proposto neste trabalho adere ao processo KDD, conforme citado na seção II.4, a etapa de "consolidação do conhecimento obtido", que constitui uma etapa posterior à mineração de dados, está prevista como um trabalho futuro baseado nas experiências descritas nas seções III.2.2. e III.3.2.

III.2. Regras de Negócios

Qualquer programa encerra, em seu código e em suas estruturas de dados, algum conhecimento sobre o domínio do problema que soluciona. Este conhecimento, geralmente implícito no código, se localiza basicamente nos procedimentos de decisão, que determinam a seqüência de execução dos diversos algoritmos referenciados, e se

manifestam nas diferentes saídas obtidas para cada entrada específica. Por exemplo, um programa de processamento da folha de pagamentos tem embutido em seu código conhecimento relativo à legislação necessária, para desconto de impostos e direitos trabalhistas, bem como a política da empresa (SULAIMAN & XEXEO, 1997). Nas palavras de (DAVENPORT & PRUSAK, 1998) "Empresas de software vendem produtos que, em essência, são idéias - propriedade intelectual - incorporadas em linhas de código".

A maior parte do conhecimento sobre o domínio de um problema não pode ser encontrada no código dos programas. Normalmente, está com os especialistas humanos, que para transmiti-lo podem utilizar algum recurso de Representação do Conhecimento, que pode ser: informal, como português corrente, semiformal, como diagramas da UML, ou, raramente, formal, como lógica matemática.

Filosoficamente, optar pela utilização de regras de produção para representar o conhecimento de um especialista significa acreditar que, ao menos em parte, as pessoas raciocinam em função de um conjunto de regras dedutivas (SULAIMAN & XEXEO, 1997).

Regras dedutivas proporcionam a habilidade de derivar informação a partir da informação preexistente, ou mesmo testar a consistência da informação armazenada em um banco de dados. Como as regras dedutivas não possibilitam mudanças no conteúdo das bases de dados, também são chamadas de regras passivas (CERI & FRATERNALI, 1997).

Em contraste com as regras dedutivas, as regras ativas possibilitam alterações no conteúdo das bases de dados. Nos produtos relacionais são chamadas de *triggers* (gatilhos) e proporcionam comportamento reativo. Trata-se de uma forma de computação motivada pela ocorrência de algum evento, tipicamente uma operação de banco de dados, executando uma reação a este estímulo (CERI & FRATERNALI, 1997), como por exemplo ilustrado no item c da Figura III.1.

Restrições de integridade é um exemplo de regras dedutivas que testam a consistência de uma base de dados. Já as rotinas que reparam a violação destas restrições, são regras ativas. Sejam dedutivas ou ativas, regras que tratam restrição de integridade são endógenas aos SGBDs: resolvem questões próprias da administração do banco de dados.

As regras que representam o conhecimento dos especialistas do negócio (tal

como nos sistemas especialistas), no contexto de banco de dados, são chamadas regras "de negócio".

Regras de negócios especificam restrições de alguma particularidade do negócio. Podem ser explicitamente armazenadas ou não. Em geral, quando se utiliza a Tecnologia da Informação, estas regras são representadas implicitamente no código fonte dos programas, através dos seus algoritmos e estruturas de dados.

Porém quando as regras de negócio são armazenadas apenas implicitamente nos programas, o esforço aumenta demasiadamente na modelagem, criação e manutenção destas regras. O armazenamento explícito destas regras, de forma declarativa, possibilita a administração deste conhecimento de forma mais organizada, por que contempla o conceito de "independência de conhecimento" (CERI & FRATERNALI, 1997).

Quando um projeto de sistemas de banco de dados possui a maior parte da semântica do domínio da aplicação armazenada de forma declarativa, compartilhada por todos os programas, sem estar codificada neles, tal como as restrições e regras, diz-se que existe a independência de conhecimento neste projeto. Este conceito é relacionado aos conceitos de independência de dados física e lógica de banco de dados (ELMASRI e NAVATHE, 2000).

Regras constituem um construto teórico bastante bem disseminado na ciência da computação. Sejam classificadas conforme o algoritmo de inferência de "encadeamento para trás" ou de "encadeamento para frente" ou quanto a possibilidade de alteração da base de dados, regra "dedutiva" ou "ativa", ou mesmo quanto a linguagens apropriadas para a sua programação, por exemplo Prolog e Datalog.

O enfoque específico em regras de negócios associada com o modelo conceitual de dados tem sua descrição exaustiva em (ROSS, 1997). Este método propõe sete categorias de regras "atômicas", assim definidas: verificadores de instância, verificadoras de tipo, verificadoras de posição, verificadores funcionais, avaliadores comparativos, avaliadores matemáticos, controladores de projeção. Partindo-se desta classificação, é possível construir-se regras mais complexas. Nesta classificação (ROSS, 1997) compara sua abordagem a uma tabela periódica.

Os verificadores de instância verificam a existência de um objeto em uma classe e, em geral, se referem a restrições de integridade referencial, estas regras regulam instâncias ou ocorrências em classes;

Os verificadores de tipo também podem restringir valores em objetos de classes e restrições de generalização. estas regras controlam a criação de múltiplas instâncias em várias classes de objetos;

Verificadores de posição se referem a restrições de ordenação de valores de objetos em classes, critérios de ordenação e impacto em classes associadas. Estas regras se referem à posição de um valor, de um atributo ou de um identificador de tributo de uma classe em seqüência de valor ou cronológica;

Verificadores funcionais testam valores tal como funções de instâncias de uma classe. Devem poder ser escritos na forma y = f(x) e se prestam para a representação de funções matemáticas pré-definidas ou do negócio;

Avaliadores comparativos restringem valores de atributos de objetos, comparando-os (comparação lógica). Os avaliadores matemáticos se referem a cálculos, tais como totais e médias de coleções de atributos de objetos e suas variações.

Em geral regras de negócio funcionam como restrições que se manifestam quando algum valor indevido é alcançado, não permitindo a atualização de determinado atributo, tal como nas restrições de integridade. No caso dos controladores de projeção é possível usá-los na criação ou exclusão de instâncias de objetos, bem como em mecanismos de inferência.

Com uma abordagem mais sucinta e atualizada (ROSS, 1998) define as regras de negócios como o "sistema nervoso" que controla os *scripts* ou *use-cases* e os Fluxos de Trabalho de um sistema corporativo, reafirmando a sua natureza declarativa, bem como o conceito de independência do conhecimento.

Em (GUIDE, 2000) encontra-se uma introdução mais didática ao tema, bem como outra taxonomia, um meta-modelo, expresso em IDEF1X. Segundo esta abordagem uma regra de negócios pode ser classificada em três grandes grupos: assertivas estruturais, assertivas de ação e regras de derivação.

As assertivas estruturais constituem o conceito definido que expressa aspectos estruturais do modelo, tais como classes e associações. As assertivas de ação são restrições ou condições que limitam ou controlam a ação no modelo, tais como restrições de integridade, condições e autorizações. As regras de derivação possibilitam a criação de novas instâncias de objetos, ou mesmo invocam a execução de uma ou mais ações, tal como os controladores de projeção.

Em (DATE, 2000b) são adotadas duas taxonomias para regras de negócios: a

primeira é uma modificação da proposta de (MARTIN & ODELL, 1997) e classifica as regras em duas grandes categorias: regras de restrição e regras de derivação. As regras de restrição podem ser do tipo restrição de estado, restrição de transição e restrição de estímulo/resposta. As regras de derivação podem ser do tipo "de computação" e do "tipo de inferência".

A segunda taxonomia para regras de negócios apresentada por (DATE, 2000b) é voltada para o modelo relacional de dados e é chamada pelo autor de "taxonomia de restrições de integridade". São quatro as restrições explicitamente enumeradas: restrição de domínio, de coluna, de tabela, e de banco de dados. Cada uma destas especifica valores válidos de restrição para o objeto de banco e dados correspondente.

Quatro referências são dignas de menção. A primeira, (CERI & FRATERNALI, 1997), descreve um método de projeto e implementação de regras dedutivas e ativas em diversos sistemas gerenciadores de banco de dados relacionais e orientados a objetos, com exemplos. A segunda, (WARMER & KEPPLE, 1999), descreve uma linguagem de restrição de objetos, utilizada como padrão na especificação de restrições da UML. A terceira, (BORGIDA, 1995), é uma revisão bibliográfica (do tipo *survey*) para lógica e gerência de dados. A quarta, (TANAKA, 1992), é um trabalho pioneiro na área, que associa entidades e relacionamentos a eventos e regras, modelo *Entity Relationship Events and Rules* (ER^2) na modelagem conceitual para Regras de Negócios.

Além destas quatro primeiras, podem ser citados os seguintes trabalhos correlatos com participação de componentes do projeto SpeCS: (SULAIMAN *et al.*, 2000, SOUZA *et al.*, 2001, SILVA *et al.*, 2001).

As duas primeiras Figuras deste capítulo (III.1 e III.2), contém cada uma, alguns exemplos de regras. As regras da primeira Figura são baseadas na primeira taxonomia de (DATE, 2000b) e no estudo de caso "*company*" descrito em (ELMASRI & NAVATHE, 2000) pg. 65.

a. Restrição de estado
"O número de horas trabalhadas por empregado em um dado projeto não pode ultrapassar 60 horas semanais": TRABALHO.HORAS <= 60
b. Restrição de transição
"Salários não podem diminuir": :NEW EMPREGADO.SALARIO >= :OLD EMPREGADO.SALARIO
c. Regra de estímulo/resposta
"Ao excluir um empregado, seus dependentes são automaticamente excluídos":

```
CREATE TABLE DEPENDENTE (
CPF_Empr          CHAR(12) NOT NULL,
Nome              VARCHAR(30) NULL,
Sexo              CHAR(1) NULL,
Data_nascimento   DATE NULL,
Relacionamento    CHAR(12) NULL,
PRIMARY KEY (CPF_Empr, Nome),
FOREIGN KEY (CPF_Empr)
REFERENCES EMPREGADO
ON DELETE CASCADE );.
```

d. Regra de computação
"Calcula total de dependentes de determinado empregado":

```
SELECT COUNT(*)
FROM EMPREGADO, DEPENDENTE
WHERE EMPREGADO.CPF = DEPENDENTE.CPF_EMPR AND
EMPREGADO.CPF = 123456789;
```

e. Regra de inferência
"Gerentes cujas horas trabalhadas em um dado projeto exceda 40 horas tem 10% de aumento no seu salário".:
Se empregado é gerente E horas trabalhadas >= 40
Então salário := salário *1,1;

Figura III.1: Primeiro conjunto de exemplos de regras (SULAIMAN *et al.*, 2000)

A segunda Figura contém exemplos de regras definidas em (ROSS, 1997) e representadas em *Object Constraint Language* OCL (WARMER & KEPPLE, 1999).

a. "O cliente deve possuir um endereço"
Customer
Self.address notEmpty
b. "Enquanto o alarme estiver soando, o mecanismo de braços mecânicos que são monitorados automaticamente, devem ser desligados, caso ainda não tenham sido"
MechanicalArm
Self.alarm.sounding implies Self.power = off
c. "Um auditor não pode auditar nenhum gerente residente na mesma cidade"
auditor
self.manager.city <> self.city
d. "Um gerente não pode ser auditado por auditor residente na mesma cidade"
manager
self.auditor.city <> self.city
e. "Um grupo não pode incluir nenhum membro sindicalizado caso inclua algum membro não sindicalizado e vice-versa"
Group

```
( (self.non-union -> includes(x) implies self.union -> not(includes(x))) and
   (self.union -> includes(x) implies self.non-union -> not(includes(x))) )
```

Figura III.2: Segundo conjunto de exemplos de regras (SULAIMAN *et al.*, 2000)

A Figura III.3 é um diagrama de classes, exibindo o meta-modelo de (CERI & FRATERNALI, 1997), e publicado em (SULAIMAN *et al.*, 2000).

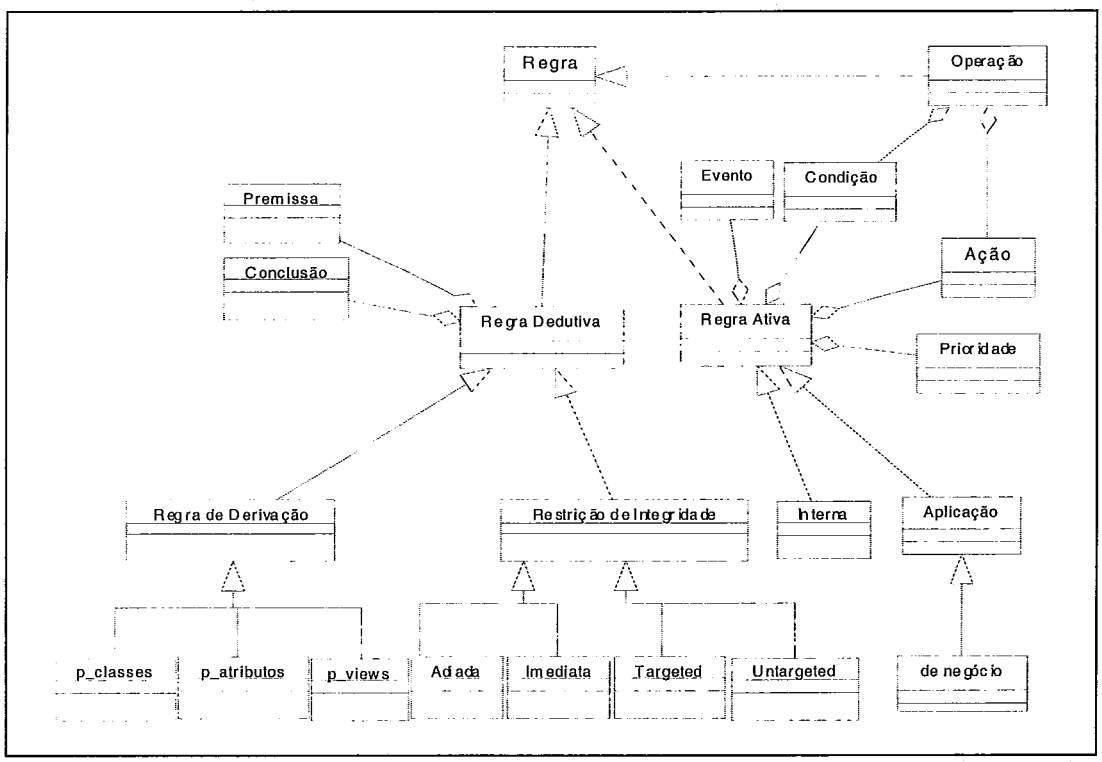


Figura III.3: Diagrama de classes exibindo o meta-modelo de (CERI & FRATERNALI, 1997, SULAIMAN *et al.*, 2000)

III.2.1. A abordagem declarativa

Regras de negócios e fluxos de trabalho são conceitos intimamente relacionados. Ambos podem estar presentes tanto em tempo de elicitação de requisitos de um sistema, através da aquisição de conhecimento na elaboração das políticas da empresa, bem como no planejamento da maneira que estas políticas serão postas em prática.

Independentemente da escolha ou preferência, vale lembrar com (ROSS, 1998) que "façamos uma revisão dos pontos sobre o relacionamento entre regras e eventos, e então examinemos as implicações. Lembre-se que *workflow*, procedimentos e/ou *scripts* que os usuários seguem ao realizarem seu trabalho, são geralmente baseados em eventos". Estas relações estão bem examinadas no modelo ER^2.

Do mesmo modo é possível vislumbrar estas duas faces da mesma moeda em níveis de abstração mais baixos quando, por exemplo, se defende o uso de linguagens declarativas em detrimento de linguagens procedimentais, na implementação de sistemas. Sempre é possível imaginar que em um procedimento podem ocorrer diversas

regras, bem como uma regra pode conter diversos procedimentos, para que seja efetivamente satisfeita.

Entre os diversos métodos e modelos de representação de regras de negócios estudados: (BORGIDA, 1995, CERI & FRATERNALI, 1997, DATE, 2000b, GUIDE, 2000, ROSS, 1997, ROSS, 1998, TANAKA, 1992, WARMER & KEPPLE, 1999), aquele que mais adere a uma representação estritamente declarativa é (DATE, 2000b), com seu título sugestivo: *What not How, the Business Rules Approach to Application Development.*

É evidente que a escolha deste método, em detrimento de outros, revela uma escolha pelo modelo relacional, no que se refere à representação de regras no nível lógico e físico de banco de dados.

Porém, além disso, é uma opção pela forma declarativa de representá-las. Nos diversos métodos e taxonomias estudados a fronteira da representação das regras de negócios costuma ultrapassar os limites da especificação das regras, passando-se à especificação dos procedimentos que as contém. Neste trabalho a opção é a representação do conhecimento procedimental com os fluxos de trabalho e regras de negócios na representação do conhecimento declarativo.

Para efeito deste trabalho o método de (DATE, 2000b) é o método escolhido, por atender plenamente os objetivos do trabalho, salvo quando houver explicitamente alguma ressalva.

O princípio de (DATE, 2000b) em favor da representação das regras de negócios de modo declarativo repete o argumento segundo o qual "linguagens de programação vem evoluindo através de diversas gerações". De modo resumido (DATE, 2000b) cita "linguagens de 1ª geração são as linguagens de máquina; de 2ª geração, linguagens *assembler*; de 3ª geração, as chamadas de alto nível (COBOL, Fortran, entre outras); de 4ª geração são as diversas linguagens proprietárias, tais como FOCUS. Alguns consideram SQL uma linguagem de 4ª geração".

Além desta hierarquia, quanto à "geração" a que pertence cada tipo de linguagem, (DATE, 2000b) também cita a hierarquia existente quanto à automação na guarda e acesso aos dados: "arquivos seqüenciais => arquivos indexados (ISAM) => bancos de dados hierárquicos e em redes => tabelas SQL". Finalmente (DATE, 2000b) cita a tendência na automação das interfaces: "RPG, SQL, QBE, visual *programming* (QBF, ABF)".

Deste modo a argumentação de (DATE, 2000b) defende um ponto de vista evolucionário quanto:

1. às linguagens de programação, ou "aspectos específicos do negócio";
2. aos repositórios de dados e sua gerência, ou "aspectos de bancos de dados" - SGBDs; e
3. às linguagens de interface, ou "aspectos de apresentação".

Chegando a defender explicitamente "declarativo é melhor do que procedimental". Seguindo esta argumentação chega-se ao objetivo de "eliminar o código", criando-se sistemas a partir de suas especificações usando-se uma "abordagem baseada em regras".

Em determinado ponto de sua teoria (DATE, 2000b) expõe o esquema representado na Figura III.4. Esta taxonomia trata regras de bancos de dados e de aplicação, omitindo-se regras de apresentação.

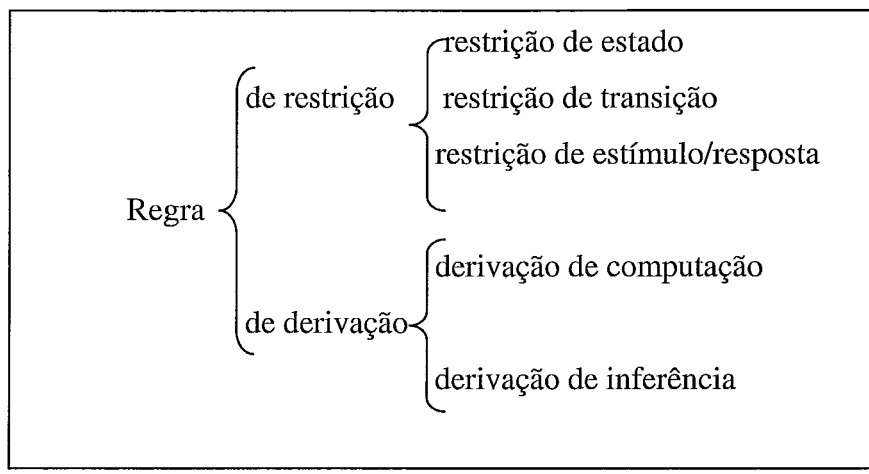


Figura III.4: Taxonomia apresentada por (DATE, 2000b)

Usando-se esta taxonomia as regras de negócios podem ser do tipo de restrição e de derivação. Exemplos de cada um dos subtipos destes tipos são encontrados na Figura III.1, com as respectivas implementações em SQL.

Ainda segundo (DATE, 2000b), das três partes de uma aplicação: aspectos de apresentação, aspectos de bancos de dados e aspectos específicos da aplicação, a parte que carece de um melhor nível de automação é a terceira, deste modo elegendo a regras de negócios como solução para este problema.

Recordando a segunda taxonomia para regras de negócios apresentada por (DATE, 2000b) na seção anterior, chamada "taxonomia de restrições de integridade". São quatro as restrições explicitamente enumeradas:

1. restrição de domínio – especifica valores válidos para um dado domínio (tipo);
2. de coluna – especifica valores válidos para uma dada coluna (atributo);
3. de tabela – especifica valores válidos para uma dada tabela (relação); e
4. de banco de dados- especifica valores válidos para um banco de dados.

Usando-se a argumentação ainda informal de (DATE, 2000b), toda relação possui duas partes componentes:

1. Cabeçalho – que constitui o conjunto dos nomes das colunas, com seus respectivos domínios; e
2. Corpo – que é o conjunto de linhas que estão em conformidade com o cabeçalho correspondente.

O início da intuição da teoria de (DATE, 2000b) se dá quando este formula: "dada uma relação R, seu cabeçalho denota um certo predicado", e ainda "domínios são conjuntos de coisas as quais podemos falar a respeito; relações são afirmações verdadeiras sobre estas coisas" e portanto "domínios estão para relações assim como nomes estão para sentenças". Partindo-se destas premissas (DATE, 2000b) formula sua *golden rule*: "nenhuma operação de alteração pode ser permitida, em uma relação, caso a leve a assumir um estado que viole seu próprio predicado" e, portanto, "o projeto de banco de dados é a definição de restrições".

Finalmente (DATE, 2000b) conclui sua teoria com a seguinte afirmação: "um banco de dados não é apenas um repositório de dados; ao invés disto é uma coleção de proposições verdadeiras. Estas proposições são instâncias verdadeiras de predicados que correspondem às relações". Abordagens formais das idéias que fundamentam (DATE, 2000b) podem ser encontradas em (ABITEBOUL *et al.*, 1995, ULLMAN, 1988, DATE, 2000a, ELMASRI & NAVATHE, 2000).

III.2.2. Experiência anterior

Em termos arquiteturais (DATE, 2000b) defende ainda uma máquina de regras logicamente independente do SGBD alvo, porém armazenando-se as regras homogeneamente no SGBD alvo. Idéias assemelhadas são defendidas em (SULAIMAN, 1992, PASSOS *et al.*, 1995, SULAIMAN *et al.*, 1995) e mais recentemente em (SOUZA *et al.*, 2001), conforme a Figura III.5, que é o modelo lógico da ferramenta de *Bill of Experiments* (BOE), um sistema dedutivo de apoio a Gestão do Conhecimento Científico.

O modelo conceitual desta experiência é suficientemente expressivo: em (SULAIMAN *et al.*, 1995) a implementação é realizada em um SGBD orientado a objetos, o O2, enquanto que em (SULAIMAN, 1999) a implementação é realizada em um SGBD relacional, o DB2, tal como em (SOUZA al., 2001) cuja implementação é feita em ACCESS.

Conceitualmente o BOE é baseado no *Bill Of Materials* (BOM). Na literatura de Banco de Dados um dos exemplos mais comuns para expressar "auto-relacionamento", ou relacionamento unário, tanto na modelagem conceitual quanto na modelagem lógica de dados é o BOM. Este exemplo ocorre em (CHEN, 1976, ELMASRI e NAVATHE, 2000, DATE, 2000a), entre outros. Possui um desdobramento natural, chamado de "agregado recursivo" em (RUMBAUGH *et al.*, 1991).

Trata-se do relacionamento entre peças cujos componentes, em geral, podem ser outras peças, e cada nova peça pode possuir outros componentes, estabelecendo-se o relacionamento recorrente.

A instância de BOM encontrada em (SULAIMAN, 1999) consiste em um aplicativo de planejamento de compra de peças (partes) para construção ou montagem de componentes eletrônicos. Cada BOM é formado de componentes e cada componente é composto de itens. Um componente só pode ser considerado disponível quando todos os seus itens estiverem disponíveis também. Cada item simples é considerado disponível somente quando é escolhido a partir das opções existentes. Se um item é composto, pode ser expresso como um novo BOM.

Em (SULAIMAN, 1999) considera-se o segmento do mercado de eletrônicos e a existência de um agente de busca que encontra e armazena em um sistema de banco de dados a oferta de peças dos diversos fornecedores, examinando-se preços e prazos de disponibilidade. As transações entre o agente de busca e os fornecedores de peças constituem um sistema WEB fundamentado no conceito *business to business* (B2B), tendo sido concebido o código XML correspondente às bases de regras do modelo. A interação de um cliente potencial com um sistema WEB é fundamentada no conceito *business to customer* (B2C) e é planejada de forma que o comprador possa avaliar uma compra, baseando-se em melhores preços, prazos mais curtos, marcas, ou um balanceamento destas variáveis de forma que se atinja uma melhor relação custo-benefício.

Assume-se ainda que um cliente tem o conhecimento de quais peças necessita

comprar para a correta montagem de um componente. Um componente é considerado como pronto para entrega a partir do momento em que todos os seus itens são instanciados e, conseqüentemente, cada peça é completamente identificada com o respectivo preço, prazo estimado de entrega, fornecedor de origem e disponibilidade.

As restrições do modelo são as seguintes: 1. o prazo de entrega de um componente sempre é, no mínimo, igual ao maior prazo de entrega entre todos os itens envolvidos; 2. o menor preço de um componente consiste no somatório dos preços dos itens de menores preços, nos quais os agregados formem o componente. Explorando-se a natureza dedutiva do sistema, o usuário pode solicitar explicações sobre preços, prazos e montagem de algum componente. Em (SULAIMAN, 1999) encontram-se propostas de solução para o componente de melhor custo-benefício, com os usuários inserindo limites de tolerância para as variáveis de preço e prazo, e atribuindo pesos para estas grandezas. Entende-se peso como uma relação de importância de uma grandeza em relação à outra. Por exemplo, caso o prazo de entrega de um componente seja de maior relevância que seu custo, a variável "prazo" terá maior peso, alterando a escolha da melhor solução.

Para uma melhor compreensão desta proposta, no caso do BOE, estabelece-se uma analogia entre a preparação do experimento científico e a compra de itens para montagem de componentes.

O processo de montagem de uma peça composta pode ser associado à aplicação de uma simulação ou experimento utilizando um ou mais conjuntos de dados científicos, gerando com isto uma nova informação relevante para a comunidade pesquisadora. Da mesma maneira que uma peça é composta por diversos itens, um novo conhecimento é composto por dados, experiências utilizando tais dados e a execução de simulações. Assim como uma nova peça gerada pode servir como um item a ser utilizado na composição de outras novas peças, uma nova massa de dados gerada no BOE pode servir de base para a aplicação de um outro modelo matemático ou experimento que gerará uma terceira. A descrição de cada entidade e seus respectivos relacionamentos é mostrada na Figura III.5 e descrito a seguir.

- INSTITUIÇÃO – Entidade referente aos "fornecedores" que alimentarão a Base de Conhecimento. Instituições que atuam na mesma área de pesquisa podem ter dados semelhantes, mas com custo diferentes.
- PESQUISADOR – Profissional especialista para o qual o BOE foi desenvolvido.

Seus experimentos são o principal recurso deste sistema.

- TIPO DE DADO – Descreve o tipo de dado contido na base. Podendo ser espacial, temporal, escalar, vetorial, textual e multimídia.
- CLÁUSULA – É a Base de Conhecimento do sistema, base de dados onde estão contidos todos os experimentos, modelos, dados singulares e regras do sistema. Cada nova descoberta é inserida na base e pode ser utilizada em futuros experimentos. A Base de Conhecimento é composta por: Fatos, Premissas e Conclusões e pode ser categorizada em Dados, Experimentos e Modelos.

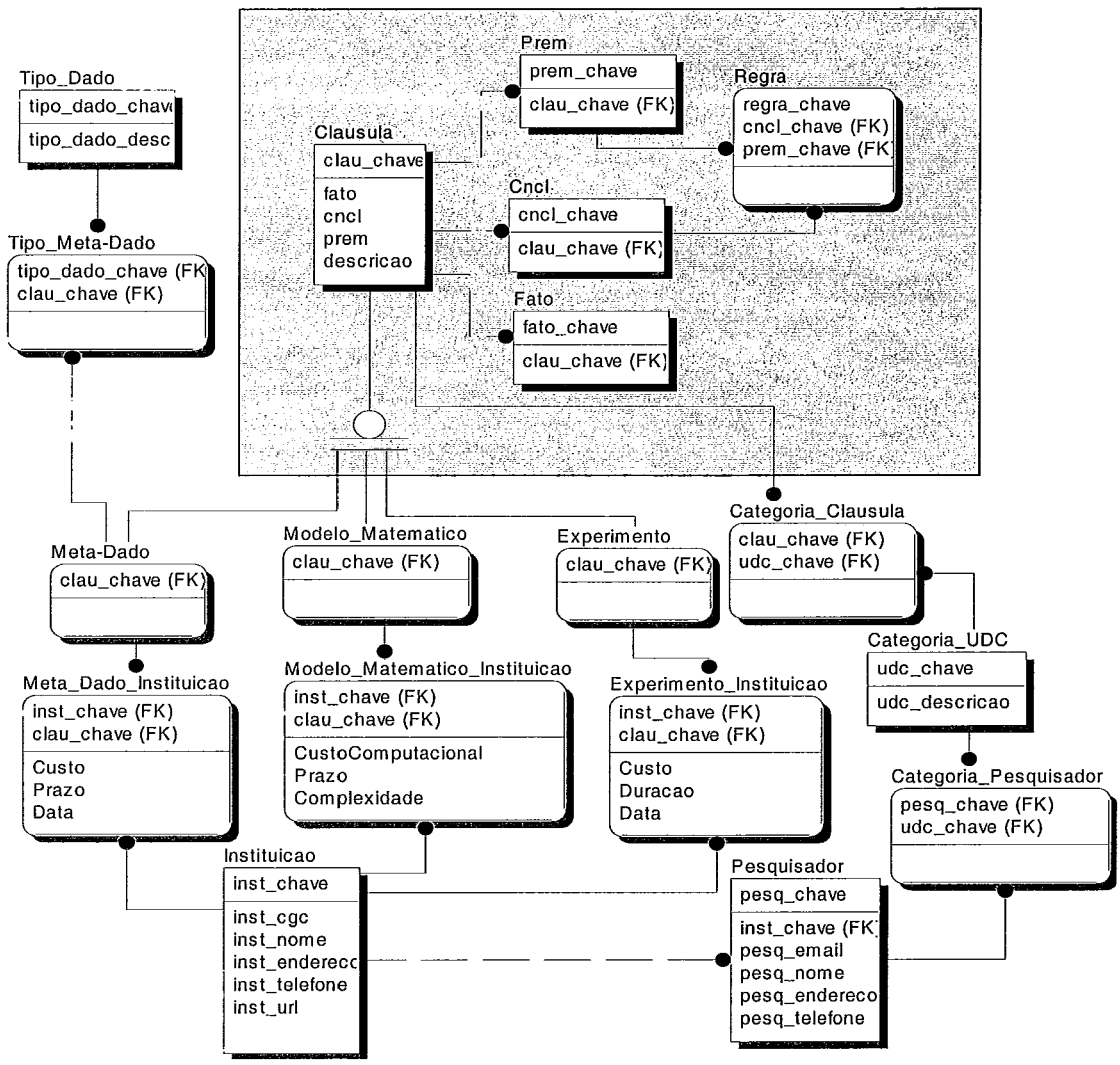


Figura III.5: Esquema lógico do BOE (SOUZA *et al.*, 2001)

- REGRA – Consiste na ligação entre premissas e conclusões. Existe para dar suporte ao ambiente dedutivo.
- PREMISSA – O que se quer provar para alcançar uma conclusão.
- FATO – Tudo que é provado como verdadeiro.

- CONCLUSÃO – É um experimento bem sucedido, caso suas premissas tenham sido provadas, que tem no uso de regras e dados contidos na Base de Conhecimento o seu fundamento. Cada conclusão provada é inserida também na Base de Conhecimento para que sirva como fato em novos experimentos.
- CATEGORIA UDC – Cada dado presente na Base do Conhecimento é classificado segundo a classificação UDC. A Classificação Decimal Universal (CDU ou UDC) é um esquema de classificação por especialidade, detalhado, compreensivo e hierárquico. É usado para fichários, identificações, índices, documentos, livros de consultas, idéias ou termos, classificados por categorias em qualquer grau, geral ou específica, abrangendo qualquer campo de conhecimento ou assunto (IBICT, 1997).

Dentro do contexto do reuso de experimentos científicos e, fazendo-se analogia com fundamentos de modelagem de dados, particularmente o auto-relacionamento peça-componente (CHEN, 1976), bem como seu desdobramento natural em agregados recursivos (RUMBAUGH *et al.*, 1991), utiliza-se o conceito clássico de BOM revisto em (SULAIMAN, 1999), integrado a uma máquina de inferência baseada em regras de produção (PASSOS *et al.*, 1995, SULAIMAN *et al.*, 1995).

O protótipo "*Bill Of Experiments*", ou BOE (SOUZA *et al.*, 2001), é o primeiro resultado de uma pesquisa em que o foco principal é a Gestão do Conhecimento de Experimentos Científicos. O principal objetivo desta ferramenta é prover dados, modelos, programas, definições de *workflow* e metadados sobre experimentos científicos, e na inexistência destes, orientar o pesquisador a como obtê-los, tornando-se uma ferramenta para melhorar o acesso, compartilhamento e conseqüente inovação do conhecimento científico.

III.2.3. Compatibilidade com a Mineração de Regras

Conforme descrito e formalizado na seção II.3.2. sobre a mineração de regras associativas, em que se obtém regas no formato: "**Se** faz calor **então** chove", ou seja, [faz calor] ==> [chove].

O antecedente da regra "[faz calor]", ou premissa, é também chamado de corpo (*body*) da regra. O conseqüente da regra "[chove]", ou conclusão, é também chamado de cabeça (*head*) da regra.

As regras descritas em (SULAIMAN, 1992, PASSOS *et al.*, 1995, SULAIMAN, 1999, SOUZA *et al.*, 2001) são compatíveis com este formato, prestando-se aos

mecanismos de inferência para frente e para trás. Deste modo, este modelo, bem como suas implementações, podem ser utilizadas nos passos 8 e 9 do processo KDD (8. Interpretar os padrões obtidos; e 9. Consolidar o conhecimento obtido) que pertencem ao domínio dos especialistas da aplicação, podendo ser automatizados.

A seguir é descrita a Tabela III.1. explicativa dos estados de uma cláusula para a máquina de inferência de (PASSOS *et al.*, 1995, SULAIMAN, *et al.*, 1995, SULAIMAN, 1999, SOUZA *et al.*, 2001).

Tabela III.1: Estados das cláusulas e suas interpretações para o esquema da Figura III.5 (PASSOS *et al.*, 1995)

ESTADO	INTERPRETAÇÃO
000	Hipótese em seu estado inicial. Quando a base de conhecimento recebe uma mensagem este estado deve ser alterado, ou então a inferência não pode acontecer, não haveria conhecimento.
001	Premissa (antecedente da regra de produção) exclusiva. Neste caso se a cláusula não mudar de estado. Esta premissa nunca será utilizada na inferência.
010	Conclusão (conseqüente da regra de produção) exclusiva. Neste caso se a cláusula não mudar de estado, esta conclusão nunca será utilizada na inferência.
100	Fato exclusivo. Trata-se de uma verdade. Caso seja uma hipótese também, a hipótese está provada.
011	Premissa e conclusão. Necessita mudar de estado, ou a cláusula não pode ser provada.
101	Fato e premissa. A cláusula está provada. A base de conhecimento deve ser reorganizada.
110	Fato e conclusão. A cláusula está provada. A base de conhecimento deve ser reorganizada.
111	Fato e premissa e conclusão. A cláusula está provada. A base de conhecimento deve ser reorganizada.

III.3. Fluxos de Trabalho

Uma menção digna de nota quanto à relação entre regras de negócios e fluxos de trabalho pode ser encontrada em (DATE, 2000b). Apesar da defesa enfática do uso declarativo das regras de negócios, (DATE, 2000b) chega a descrever a intuição de uma ferramenta de gerência de fluxos de trabalho primitiva, como segue: "a regra de negócio será usada como parte de cada seqüência do fluxo de trabalho para a aplicação como um

todo, ou seja, formulários de *transições de estados* podem ser definidos, através de alguma interface interativa do tipo *caixas e setas*. O diagrama de caixas e setas resultante representa efetivamente a aplicação como um todo em um nível alto de abstração; para ser específico, este diagrama mostra a seqüência na qual as ações que a constituem devem ser executadas" (DATE, 2000b).

De modo semelhante (ROSS, 1998) sugere uma analogia, também informal, entre o desenvolvimento de sistemas pela abordagem das regras de negócios e o organismo humano. Esta analogia está baseada em três componentes: estrutura, força e controle.

A estrutura se refere aos ossos e esqueleto, seus análogos são as entidades e seus relacionamentos; a força se refere aos músculos que proporcionam o movimento ou o comportamento percebido do corpo, seus análogos são os processos de negócios e suas atividades; o controle é realizado pelo sistema nervoso, que segundo o autor, conecta os músculos, segundo uma rede de conexões coordenadas pelo cérebro. Seus análogos são as regras de negócios, que restringem os processos, de modo a controlar o sistema como um todo.

A resposta a cada estímulo se dá pelo disparo de impulsos nervosos. "Sem os disparos, não há movimento e, portanto, não há comportamento", e ainda "regras são a chave para a reengenharia de fluxos de trabalho" (ROSS, 1998).

Segundo (ROSS, 1998), na abordagem de regras de negócios, um *script* é um procedimento que consiste de uma série de solicitações a atores humanos e não-humanos que interagem de modo colaborativo na execução de um processo e adverte: "lembre-se que fluxos de trabalho, procedimentos e/ou *scripts* que os usuários seguem ao trabalharem são geralmente organizados em eventos". A relação entre regras e eventos acontece, segundo (ROSS, 1998), da seguinte maneira: toda regra produz dois ou mais eventos de alteração ou *update* quando violado; e todo evento de alteração pode disparar mais de uma regra. A Figura III.6. representa intuitivamente as relações entre regras de negócios e fluxos de trabalho.

Com o advento dos métodos de desenvolvimento de sistemas de informação em geral, e dos métodos orientados a objetos em particular, e enfocando o método UML como expressão de Engenharia de Software para o caso deste trabalho, foi citada a OCL (WARMER & KEPPLE, 1999) na representação de regras de negócios, na seção III.2, Figura III.2. Apesar da OCL não ter sido a representação de escolha para este trabalho,

há uma experiência com OCL de alguns dos participantes do grupo SpeCS relatada em (SOUZA *et al.*, 2001, SILVA *et al.*, 2001).

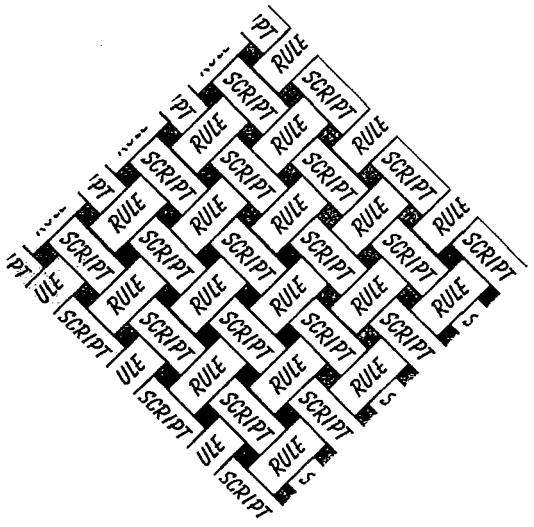


Figura III.6: regras e *scripts* (ROSS, 1998)

O tratamento dado aos fluxos de trabalho pela UML, particularmente em (BOOCH *et al.*, 1999) é a representação dos diagramas de atividades: "diagramas de atividades constituem um dos cinco diagramas da UML para a modelagem de aspectos dinâmicos de sistemas. Um diagrama de atividades é essencialmente um fluxograma, mostrando o fluxo de controle de atividade para atividade" e "atividade é uma execução não atômica em uma máquina de estados".

Em (BOOCH *et al.*, 1999), há uma outra citação que enfatiza os processos de negócios, como um contraponto da abordagem meramente diagramática, da citação anterior: "especialmente em sistemas de missão crítica, de software de empresas, são encontrados sistemas executando no contexto de processo de negócios de alto nível. Estes processos de negócios são tipos de *workflow* por que representam os fluxos de trabalho e objetos através do negócio". No mesmo (BOOCH *et al.*, 1999), pg. 269, há uma proposta de método para a modelagem de *workflow*:

- Estabeleça o foco do fluxo de trabalho. Para sistemas não triviais é impossível mostrar todos os fluxos interessantes em um único diagrama;
- Selecione os objetos do negócio que possuem responsabilidades de alto nível para partes do fluxo de trabalho como um todo. Estes podem ser coisas reais originárias do vocabulário do sistema, ou podem ser mais abstratas. Em qualquer

caso, crie uma raia (*swimlane*) para cada objeto do negócio que seja importante;

- Identifique as pré-condições dos estados iniciais e as pós-condições do estado final do fluxo de trabalho. Isto é importante na modelagem dos limites do *workflow*;
- Comece a especificar as atividades e ações no estado inicial do fluxo de trabalho, documentando no diagrama de atividades, passo a passo, a cada ação ou atividade;
- Para ações complexas, ou conjuntos de ações que aparecem múltiplas vezes, subdivida-as em sub estados de atividades e providencie um diagrama de atividades separado que expanda cada caso;
- Documente as transições que conectam estas atividades e estados de ação. Comece com os fluxos seqüenciais primeiramente no fluxo de trabalho e depois considere um *branching*. Somente depois considere as operações de *fork* ou *join*; e
- Caso haja objetos importantes que estejam envolvidos no fluxo de trabalho, documente-os no diagrama de atividades. Exiba os seus valores de mudança de estados quando necessário para a comunicação do objetivo do fluxo do objeto.

Em (FOWLER & SCOTT, 2000) há algumas citações à modelagem de fluxos de trabalho usando-se os diagramas de atividades da UML e adverte: "ao contrário das outras técnicas da UML, diagramas de atividades não tem origens claras nos trabalhos anteriores dos três amigos (BOOCH *et al.*, 1999). Ao contrário o diagrama de atividades, combina idéias de várias técnicas: os diagramas de eventos de Jim Odell, técnicas de modelagem de estados SDL, modelagem de *workflow* e redes de Petri".

Em uma abordagem mais voltada para a elicitação e requisitos da UML, (SCHNEIDER & WINTERS, 2001) cita: "diagramas de *use-case* fornecem uma imagem estática da funcionalidade do sistema. Diagramas de atividades são usados para mostrar *workflow*". E aconselha: "quando os *use-cases* precisarem executar em determinada ordem, quando houver condições que precisam ser satisfeitas entre a execução de alguns *use-cases*, ou quando é necessário exibir processos manuais misturados com o *software*, então utilize os diagramas de atividades no nível do processo do negócio".

Em uma abordagem mais voltada para a modelagem de negócios com UML, (MARSHALL, 2000) cita: "*Workflow* é uma parte integrante do processo de negócio, e

não precisa de um projeto separado", e ainda "processos de negócios são definidos com uma variedade de níveis de flexibilidade e precisão. Desde processos não estruturados *ad hoc* até processos de produção precisos podem ser representados de diversos modos usando-se UML, porém os diagramas de atividades são, em geral, suficientes para este propósito". Em (MARSHALL, 2000) há ainda uma citação ao (WFMC, 1994), que define: "*workflow* é a facilidade computadorizada ou componente automatizado de um processo".

A definição proposta pela *Workflow Management Coalition* (WFMC) é a seguinte: "é a automação de um processo de negócios, no todo ou em parte, durante o qual documentos, informação ou tarefas são passadas de um participante para o outro, objetivando a ação, de acordo com **regras** procedimentais".

Os tipos de fluxos de trabalho proposto por (ALLEN, 2001), *chair* do comitê de relações externas da WFMC, são os seguintes: fluxos de trabalho de produção, administrativos, colaborativos e ad-hoc. (ALLEN, 2001) previne que existem exceções e áreas "cinzentas" nesta classificação.

Para efeito neste trabalho não é essencial que as definições de cada uma destas classificações seja esmiuçada, porém cabe a seguinte citação: "fluxos de trabalho de produção podem gerenciar processos altamente complexos e podem ser fortemente integrados com sistemas existentes (herdados). De fato a tendência é embutir componentes de fluxos de trabalho em aplicações maiores onde seu papel é atuar como uma **máquina de regras**" (ALLEN, 2001). Nas duas citações anteriores (ALLEN, 2001) são grifados os termos "regra" e "máquina de regras" intencionalmente, caracterizando mais uma vez a interdependência destes dois conceitos.

Existe o cuidado por parte da WFMC de caracterizar SGW e diferenciá-los dos SGBDs. Há ainda a seguinte advertência em (ALLEN, 2001): "é importante que os usuários estejam habilitados a diferenciar os setores baseados em regras de uma aplicação que são normalmente ativados por gatilhos de bancos de dados e máquinas baseadas em fluxos de trabalho. O primeiro caso é normalmente escrito por autor de aplicação e apenas opera dentro de sua aplicação, suportando funções relevantes. O segundo é normalmente um componente reutilizável, ou seja, a mesma máquina vai trabalhar em muitas aplicações. Normalmente estas máquinas fornecem mais interfaces funcionais as quais são baseadas em padrões".

A terminologia proposta por (ALLEN, 2001), e que está representada na Figura

III.7., define todos os sistemas de fluxos de trabalho como sendo orientados por processos, que podem conter subprocessos. Cada processo, ou sub processo, contém algumas atividades. Uma atividade é um passo lógico único no processo. Como em alguns casos não é prático automatizar todas as atividades de um processo, fluxos de trabalho executam as atividades automáticas, enquanto as definições de processos descrevem todos os processos, sejam automáticos ou manuais.

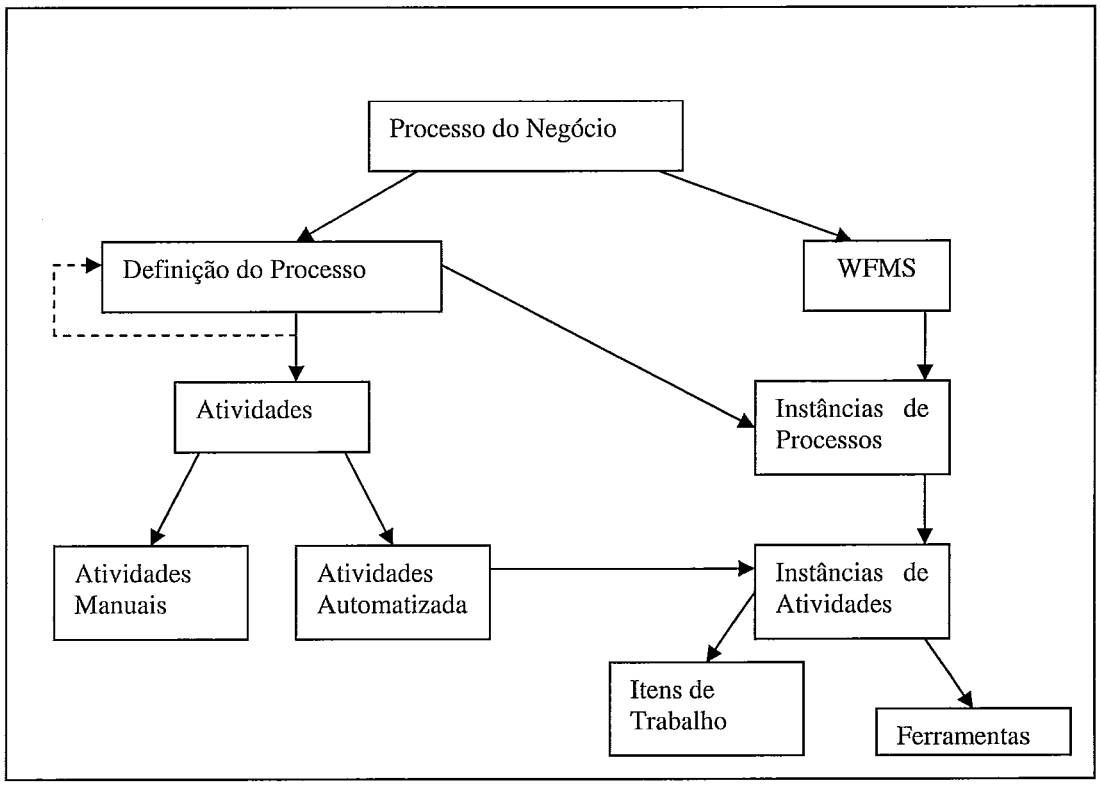


Figura III.7: Modelo de processo de negócio (ALLEN, 2001)

III.3.1. Fluxos de trabalho e transações de bancos de dados

Segundo (GRAY & REUTER, 1993) o termo transação possui diversos significados, que dependem do ponto de vista. Entre os pontos de vista citados, são encontrados: a perspectiva do usuário, a perspectiva do programador e a perspectiva do administrador. Essas perspectivas podem ser exemplificadas da seguinte maneira: "o pedido ou mensagem de entrada que inicia uma operação", ou transação de pedido/resposta; "todos os efeitos da execução de um programa", ou transação propriamente; "os programas que executaram a operação, ou programa da transação". A perspectiva escolhida em (GRAY & REUTER, 1993) é a perspectiva do programador e a definição adotada é a seguinte: "uma transação é uma coleção de operações nos estados físicos e abstratos de uma aplicação".

Sob o ponto de vista de (DATE, 2000b), uma transação é uma unidade lógica de trabalho, em geral envolvendo diversas operações de banco de dados, em particular, várias operações de atualização.

Segundo (OZSU & VALDURIEZ, 1999) "transações vem sendo classificadas de diversos modos. Um deles é quanto à duração da transação. Segundo este critério a transação pode ser classificada como *on-line* ou *batch*. Nomes mais adequados para estas duas categorias seriam transações de vida curta e longa, respectivamente". Ainda segundo (OZSU & VALDURIEZ, 1999) "Outra classificação proposta diz respeito à organização das ações de leitura e gravação".

O primeiro tipo de transação, segundo esta taxonomia é o tipo "geral". Sua característica é a mistura desordenada de operações leitura e gravação. O segundo tipo é chamado transação em "dois passos", cuja característica é a ocorrência de todas as operações de leitura antes de qualquer operação de gravação. De modo similar se a transação é restrita de modo que um item de dado tenha que ser lido antes de ser gravado, a categoria correspondente é chamada "restrita". Se uma transação contiver as características da transação em dois passos, bem como da transação restrita simultaneamente, diz-se que esta é uma transação de "dois passos restrita" Há ainda o modelo de "ação" cuja restrição é a seguinte: é um subtipo da categoria "restrita" em que se garante que cada par <leitura, gravação> em um dado objeto de dados é atomicamente executado.

Há também a classificação das transações segundo sua estrutura. Segundo esta classificação há uma hierarquia crescente de complexidade: *flat transactions*, *closed nested transactions*, *open nested transactions* e *workflow models*.

Usualmente as transações de bancos de dados relacionais são designadas como *flat transaction* e respeitam em certo nível as propriedades ACID, *Atomicity, Consistency, Isolation* e *Durability*. Sua principal característica é a de possuírem um único ponto de início e outro de terminação. As *nested transactions*, ou aninhadas, permitem que uma transação inclua outras transações que contenham cada uma seu próprio ponto de início e fim. As transações embutidas são usualmente chamadas de subtransações. Quando se respeita a restrição de uma subtransação terminar antes da sua transação chamadora, diz-se que se a transação chamadora é aninhada fechada. Em caso contrário, trata-se de uma transação aninhada aberta.

Outro modelo de transação de banco de dados é o modelo de *workflow*, ou fluxo

de trabalho, em que se procura obter todas as características necessárias na implementação de um processo de negócios. Se, por um lado, o modelo de *flat transactions* é apropriado na implementação de atividades curtas e simples, por outro lado não é suficientemente expressivo para implementar processo de negócios complexos. O modelo de *workflow*, ou de fluxo de trabalho combina todos os anteriores e em qualquer ordem, sejam aninhados abertos ou fechados, o que torna a tarefa de controle deste tipo de transação extremamente complexo.

A classificação identificada por (OZSU & VALDURIEZ, 1999) para o modelo de transações de fluxo de trabalho identifica três tipos: fluxos de trabalho orientados a humanos, fluxos de trabalho orientados a sistemas e fluxos de trabalho transacionais. O fluxo de trabalho transacional contém elementos humanos e de sistemas.

Como não há uma unificação do conceito de fluxo de trabalho, o "ponto comum" encontrado por (OZSU & VALDURIEZ, 1999), é a definição de conceito de atividade que consiste de um conjunto de tarefas, onde as relações de precedência são bem definidas entre elas.

Porém , nos sistemas atuais em produção, o tipo de transação predominante é a *flat transaction*, como pode ser visto em (DATE, 2000a, ELMASRI & NAVATHE, 2000, ULLMAN & WIDOM, 1997, CERI & FRATERNALI, 1997), por exemplo. Como ficará mais evidente no capítulo V, a mineração sobre logs de transações é a maneira proposta neste trabalho para a reengenharia de transações abstratas, que são aquelas que compõem um fluxo de trabalho idealizado por seu projetista original.

III.3.2. Experiência anterior

Nesta seção é enfocada experiência anterior (SILVA *et al.*, 2001), particularmente a representação de diagramas de atividades para os elementos básicos de fluxos de trabalho e sua representação intuitiva de modo declarativo. A Figura III.8 exibe um diagrama de atividades que representa seqüência, decisão, repetição e paralelismo, descritas brevemente a seguir:

- Em um diagrama de atividades, que representa uma seqüência, é garantida a precedência da atividade A_1 sobre a atividade A_2 , ou seja ($A_1 < A_2$) . Deste modo, ao fim da execução deste processo as atividades A_1, A_2 terão sido realizadas nesta ordem;
- Em um diagrama de atividades, que representa uma escolha conforme a Figura

III.8, ao fim desta operação, sabe-se que terão sido realizadas as atividades A_1 e A_3 ou então A_1 e A_4 e também que há uma relação de precedência entre A_1 e A_3 e também entre A_1 e A_4, ou seja $((A_1<A_3)||(A_1<A_4))$.

- Em um diagrama de atividades, que representa uma estrutura de repetição conforme a Figura III.8, ao fim desta operação terão sido executadas as operações A_1 e A_6, sendo que A_5 pode ter sido realizada um número de vezes que varia de 0 a n. Sabe-se ainda que A_1 precede qualquer ocorrência de A_5 e que qualquer ocorrência de A_5 precede A_6, ou seja $(A_1 < (A_5<...<A_5)<A_6)$.
- Em um diagrama de atividades, que representa uma estrutura de paralelismo conforme a Figura III.8, onde a atividade A_1 é seguida de uma operação de *fork*, ocorrendo então as atividades A_3 e A_4 em qualquer ordem, somente quando A_3 e A_4 forem satisfeitas é realizada a operação de *join* e, só então A_7 executa, ou seja $(A_1<((A_3\leq A_4)||(A_4\leq A_3))<A_7)$.

Em (SILVA *et al.*, 2001) estes componentes procedimentais foram adicionados a uma máquina de regras OCL (WARMER & KEPPLE, 1999), para geração de código SQL e PL/SQL.

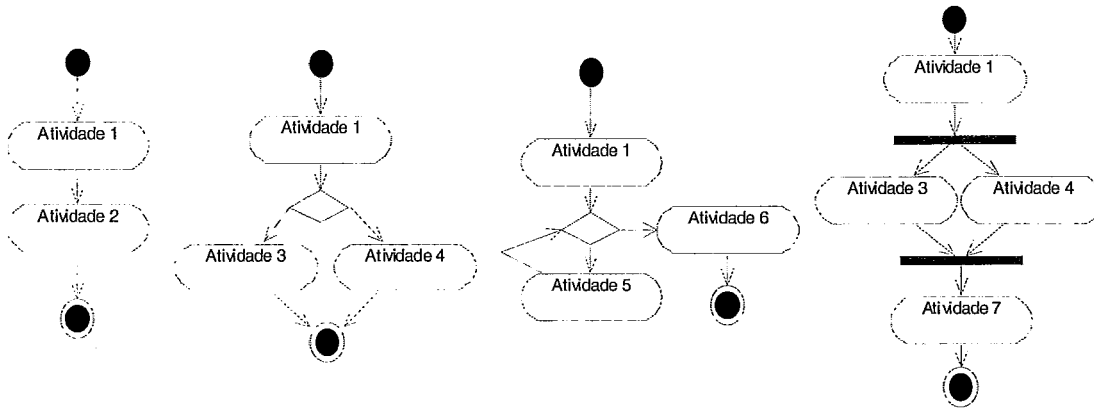


Figura III.8: Estruturas de seqüência, decisão, repetição e paralelismo

III.3.3. Compatibilidade com a Mineração de Padrões Seqüenciais

Neste trabalho padrões seqüenciais representam seqüências de programas que são referenciados por suas respectivas transações. A concepção deste conceito (SULAIMAN & SOUZA, 1998d) foi direcionada para solucionar um problema de natureza endógena, que é a descoberta de padrões de seqüências de chamadas a programas: seqüências de programas que ocorrem em transações diferentes, com determinada seqüência e ordem especificada. Formalizando:

- Dada uma base de dados *D* de *logs* de transações de banco de dados, onde cada transação possui os seguintes campos: identificação da seqüência, hora da transação e programas chamados. As restrições do modelo são as seguintes: nenhuma seqüência possui mais do que um programa com o mesmo horário de transação;
- Neste contexto, um conjunto de programas é não vazio e uma seqüência é um lista ordenada de conjuntos de programas. A cada item corresponde um inteiro, para efeito de facilitar o processamento;
- Uma seqüência $< a_1, a_2, ..., a_n>$ de programas está contida em outra seqüência $<b_1, b_2, ..., b_m>$ de programas se existirem inteiros $i_1<i_2<...<i_n$ tais que $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, ..., $a_n \subseteq b_{in}$; $_e$Em um conjunto de seqüências, uma seqüência é maximal se não está contida em nenhuma seqüência.

Maiores detalhes sobre o formalismo, bem como os algoritmos correspondentes a este método são encontrados em (AGRAWAL & SRIKANT, 1995).

Capítulo IV - Mineração de Dados Endógenos

IV.1. Introdução

Neste capítulo, é caracterizado o material sobre o qual é realizado o processo KDD: os catálogos e os *logs* dos SGBDs, chamados dados endógenos. Além disso são descritos o método proposto e o experimento.

Diferentemente de outras abordagens nas quais é realizada a análise dos dados de uma aplicação, neste trabalho o enfoque são os dados gerados pelo SGBD, na sua tarefa de registro das transações efetivamente acontecidas. Estas bases de dados são endógenas aos SGBDs. A abordagem deste estudo possui características cognitivistas, uma vez que estão sendo analisadas duas instâncias de entradas e saídas do SGBD, conforme ilustrado na Figura IV.1:

1. A entrada a ser analisada é o esquema do banco de dados, que é armazenado no SGBD em seu catálogo. Os metadados de cada banco de dados são armazenados de modo que itens de dados sejam agrupados em estruturas e haja o mapeamento das relações entre estruturas. Este esquema, em conjunto com as restrições armazenadas, configura a ontologia parcial de cada banco de dados; e
2. A saída a ser analisada são os *logs* de transações de banco de dados. Estes registram as ocorrências de criação, recuperação, alteração ou retirada de cada item de dados, relacionado-as com sua transação de origem. A esta saída correspondem as transações efetivamente realizadas contra o SGBD: cada solicitação de criação, recuperação, alteração ou retirada de um ou mais itens de dados é um evento de entrada no SGBD.

A seguir são descritos brevemente o catálogo e os logs dos SGBDs, caracterizando-os como bases de dados endógenos no contexto deste trabalho.

IV.2. Catálogos de Sistemas Gerenciadores de Bancos de Dados

O catálogo é o lugar em que os vários esquemas de banco de dados sejam eles externos, conceituais ou internos, bem como os mapeamentos correspondentes, são mantidos. O catálogo contém informações detalhadas, metadados, referentes aos

diversos objetos que são de interesse do próprio sistema.

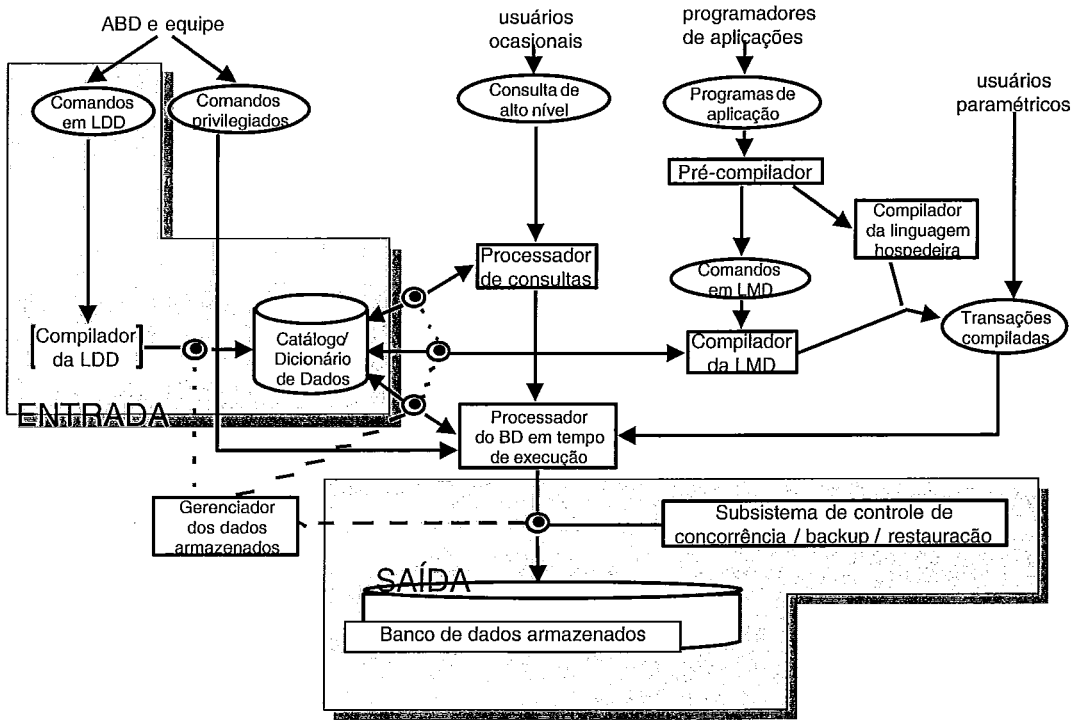


Figura IV.1: Entrada e saída utilizadas, adaptadas de (ELMASRI & NAVATHE, 2000)

Os SGBDs comerciais adaptam diferentes convenções e terminologia com respeito aos seus sistemas de catálogo. Porém, em geral, os sistemas de catálogo possuem informação similar, tais como, as descrições conceituais, lógicas e físicas de metadados.

Os SGBDs implementam seus catálogos em formato semelhante ao formato em que armazenam os próprios dados. Em particular SGBDs relacionais armazenam os metadados em tabelas, passíveis de serem consultadas, do mesmo modo que os seus dados.

Um catálogo é estreitamente relacionado com o *software* de SGBDs. Ele é acessado pelos diversos módulos do SGBD, tais como os compiladores das linguagens de definição de dados (DDL) e de manipulação de dados (DML), bem como o otimizador de consultas, o processador de transação, o gerador de relatórios e o gerente de restrição.

As informações armazenadas em um catálogo de SGBD relacional incluem a descrição dos nomes de relações, atributos, domínios dos atributos, chaves primárias, chaves secundárias, chaves estrangeiras, e outros tipos de restrição, bem como

descrições de nível externo de visão e descrição de nível interno de armazenamento de estruturas e índices. Além disso, também armazenam a informação de segurança e autorização, que especifica a autorização de acesso dos usuários às relações do banco de dados e às visões, e os respectivos donos e criadores de cada relação. As Figuras IV.2. e IV.3. representam catálogos relacionais e em redes, respectivamente.

No caso dos SGBDs relacionais, é uma prática comum o armazenamento do catálogo na forma de relações e o uso do próprio software do SGBD na gerência do catálogo. Isto permite às rotinas, bem como aos usuários, fazer acesso à informação armazenada no catálogo, desde que autorizado para tanto, usando a própria linguagem de consulta do SGBD.

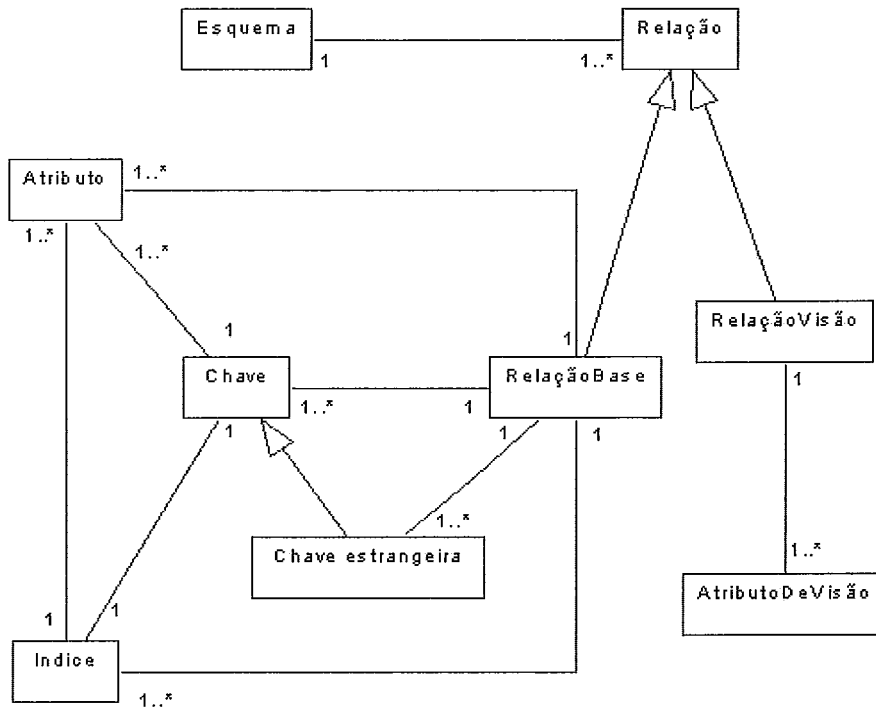


Figura IV.2: Diagrama de classes para catálogo relacional (ELMASRI e NAVATHE, 2000)

A informação básica a ser armazenada em um catálogo de um SGBD em redes é uma descrição de tipos de registros e tipos de conjuntos. A informação de como cada tipo conjunto é implementado, e outras escolhas de armazenamento físico devem ser também armazenadas no catálogo, assim como a informação de segurança, a informação de autorização e informação relativa aos subesquemas externos.

IV.3. Logs de Sistemas Gerenciadores de Bancos de Dados

Uma transação atômica típica corresponde a uma unidade de trabalho que pode ser completada totalmente ou então desfeita completamente. Para efeito de recuperação de falhas, o SGBD registra quando a transação se inicia, termina, é confirmada ou é abortada. As operações típicas são as seguintes:

- *Begin_transaction:* marca o início da execução da transação;
- *Read ou write:* especifica operações de leitura ou gravação sobre os itens de banco de dados que são executados como parte da transação;
- *End_transaction:* especifica o fim das operações de leitura e gravação e marca o fim da execução da transação;
- *Commit_transaction:* sinaliza término da transação com sucesso e que a transação não pode ser desfeita;
- *Rollback* (ou *abort*)*:* sinaliza que a transação terminou de modo anormal de modo que todas as operações realizadas no escopo desta transação devem ser desfeitas.

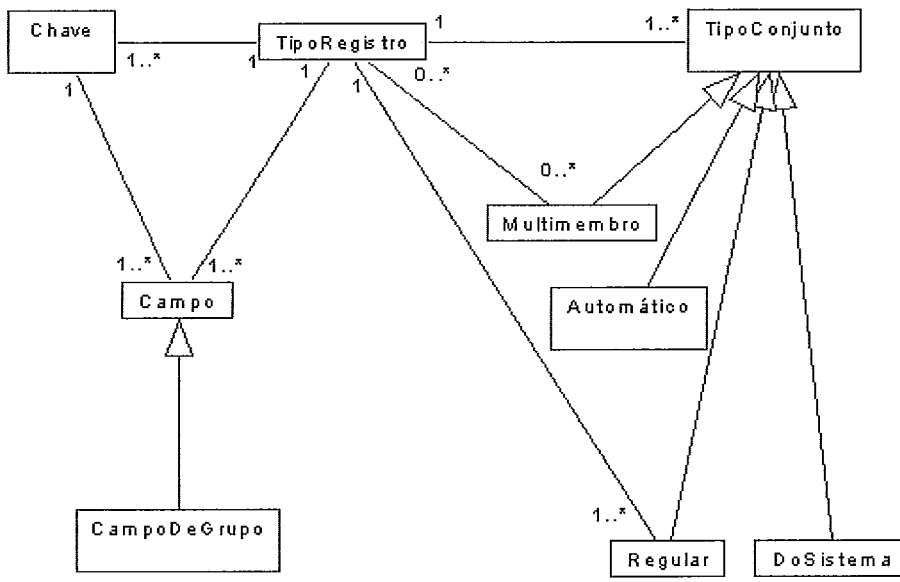


Figura IV.3: Diagrama de classes para catálogo em redes (ELMASRI e NAVATHE, 1994)

Quando uma transação é iniciada (*begin_transaction*) entra imediatamente no seu estado ativo. Após seu início, a transação pode realizar operações de leitura ou gravação (*read, write*). Após o término destas operações (*end_transaction*), ela entra em

estado de confirmação parcial. A transação é confirmada (*commit*) após o SGBD conferir os estados da base de dados, em caso de sucesso. Porém, a transação pode entrar em um estado anormal e ser abortada (*abort*). Neste caso, todas as operações realizadas pela transação até aquele momento devem ser desfeitas. A Figura IV.4. exibe o diagrama de estados de uma transação típica.

Uma propriedade importante no processamento de transações é a serializabilidade (ELMASRI & NAVATHE, 2000, DATE, 2000a). A serializabilidade é o critério de correção geralmente aceito para a execução de um dado conjunto de transações. Mais precisamente, uma dada execução de um conjunto de transações é considerada correta se for serializável, isto é, produz o mesmo resultado de alguma execução serial das mesmas transações, executadas uma de cada vez. A justificativa desta assertiva é a seguinte:

1. Transações individuais são consideradas corretas, ou seja, elas supostamente transformam um estado correto do banco de dados em outro estado correto;
2. Por esta razão, a execução das transações uma de cada vez em qualquer ordem serial também é correta. Diz-se "qualquer" ordem serial porque transações individuais são consideradas independentes umas das outras; e
3. Portanto, uma execução intercalada é correta se é equivalente a alguma execução serial, isto é, se ela é serializável.

De modo a permitir a recuperação dos estados originais dos objetos de dados da transação em caso de falhas, que podem afetar negativamente as transações, o SGBD mantém *logs* ou *system journal*, que registram todas as operações que afetam os valores dos itens de bancos de dados. Antes que uma transação alcance seu ponto de confirmação qualquer registro do *log* deve ter sido gravado em disco. Este processo chama-se *force-writing*.

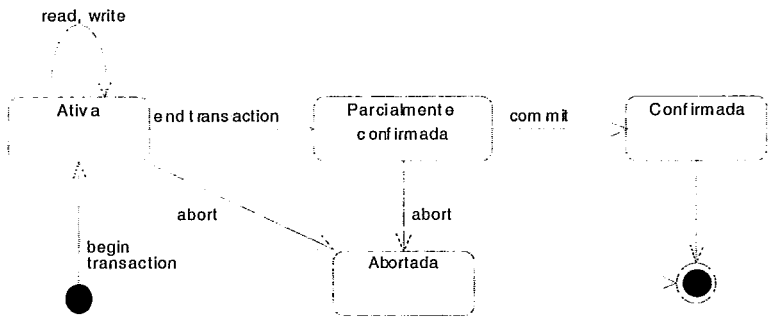


Figura IV.4: Diagrama de estados da execução de uma transação típica (ELMASRI & NAVATHE, 2000)

Os registros típicos de um *log*, ou *log records*, se referem a uma única identificação de transação, ou *transaction-id*, que é gerada automaticamente pelo SGBD e é usada para identificar univocamente a transação:

- [*start_transaction*, T]: indica que a transação T iniciou sua execução;
- [*write_item*, T, *old_value*, *new_value*]: indica que a transação T alterou o valor do item X da base de dados, do valor *old_value* para o valor *new_value*;
- [*read_item*, T, X]: indica que a transação T leu o valor do item X da base de dados;
- [*commit*, T]: indica que a transação T terminou com sucesso e os valores eventualmente alterados na base de dados estão armazenados permanentemente;
- [*abort*, T]: indica que a transação T foi abortada.

Os protocolos de recuperação que evitam *rollback* em cascata, que incluem todos os protocolos e, na prática, não requerem que as operações de leitura sejam registradas no *log*. Porém, caso o *log* seja usado para outros propósitos, tal como auditoria, os registros das operações de leitura podem ser incluídos (ELMASRI e NAVATHE, 2000).

Cada registro de log costuma possuir um número de seqüência de *log* (LSN), que é incrementado monotonicamente e indica o endereço do registro de *log* no disco. Cada LSN corresponde a uma ação específica de alteração de uma transação. Os campos de dados típicos em um registro de *log* são os seguintes:

1. O LSN anterior para aquela transação;
2. A identificação da transação;
3. O tipo do registro de transação;
4. A identificação da página que inclui o item de dados;
5. O tamanho do item;
6. O *offset* do início da página;
7. A imagem anterior do item; e
8. A imagem posterior do item.

IV.4. Método Proposto para Mineração de Dados Endógenos

Partindo-se da revisão de literatura a respeito de *data warehouse*, *logs* de bancos de dados podem ser modelados e implementados como um tipo de armazém de

dados (SULAIMAN & SOUZA, 1998d). Neste armazém de dados, é possível realizar-se operações OLAP (AGRAWAL *et al.*,1997), bem como mineração de dados, tal como descrito em (AGRAWAL *et al.*, 1993, AGRAWAL & SRIKANT, 1995).

Neste trabalho é introduzida a tabela de fatos CUBO CRUD que pode ser alvo de operações OLAP. Por exemplo, caso seja aplicada uma operação *drill-up* ou *roll-up* contra um CUBO CRUD, gera-se automaticamente uma matriz CRUD. Além disso, um CUBO CRUD também pode ser minerado utilizando-se pelo menos duas técnicas:

1. Mineração de Regras Associativas; e
2. Mineração de Padrões Seqüenciais.

No primeiro caso, o resultado permite a obtenção de alguns tipos de regras de negócios. No segundo caso, o resultado permite a obtenção de partes de fluxos de trabalho. O processo de obtenção é semi-automático.

Nas seções IV.4.1., IV.4.2. e IV.4.3, são apresentados respectivamente exemplos de operações OLAP, regras associativas e fluxos de trabalho extraídos do CUBO CRUD.

Conforme citado na seção II.4, este método adere ao processo KDD. Deste modo são desenvolvidas as nove etapas previstas desde a "compreensão do domínio da aplicação" até a "consolidação do conhecimento obtido", estando esta última etapa com previsão de implementação baseada na máquina de regras descrita na seção III.2.2. e no experimento descrito na seção III.3.2.

As quatro primeiras etapas do processo KDD, no caso da mineração endógena de dados, são muito dependentes da aplicação-alvo: como os *logs* endógenos são gerados pelos próprios SGBDs, a tarefa de extração destes dados precisa de ser feita "sob medida", para que sejam obtidos dados em um formato canônico. Este formato canônico é o CUBO CRUD.

IV.4.1. O CUBO CRUD e as operações OLAP

No desenvolvimento de sistemas de informação, é comum utilizar-se do recurso da criação de uma referência cruzada entre as funções do negócio e as entidades do sistema. Esta matriz possui o nome de "matriz CRUD" (IBM, 1978, MARTIN, 1982, MONTGOMERY, 1994) onde o acrônimo CRUD se refere a "*Create*", "*Retrieve*", "*Update*" e "*Delete*".

Usualmente, esta matriz representa em suas linhas os programas ou transações e nas suas colunas os arquivos ou tabelas. No encontro de uma linha e uma coluna é registrado uma ou mais operações do tipo CRUD.

Em geral, a criação de uma matriz CRUD ocorre em etapas preliminares no processo de criação de um sistema de informação desenvolvido de cima para baixo (*top down*) e auxilia tanto na estimativa da complexidade das transações como também no projeto da distribuição física de dados e processos.

Neste trabalho, de engenharia reversa, um dos produtos é uma matriz CRUD, ilustrada na Figura IV.5., e também o que será chamado a partir de agora de CUBO CRUD, proveniente do *log* endógeno. Trata-se de uma abordagem de baixo para cima (*bottom-up*). Após a extração do *log* endógeno gerado no formato alvo, cria-se a tabela de fatos CUBO CRUD, representativa do histórico de transações acontecidas, com suas respectivas operações CRUD. A tabela de fatos CUBO CRUD deve conter ao menos o registro de cada operação CRUD, para cada transação, arquivo e horário.

Primeiramente é exposto o formalismo de matriz CRUD tradicional: Seja $T = t_1, t_2, ..., t_m$ o conjunto das transações de uma aplicação; Seja $A = a_1, a_2, ..., a_n$ o conjunto de arquivos/tabelas da mesma aplicação; e Seja $R \subseteq A \times T = t_1a_1, t_1a_2, ..., t_ma_n$ a relação que representa a matriz CRUD, onde cada t_ia_j é representada pelo seguinte valor (ou qualquer combinação de valores, sem repetição) :

C = *Create* /* Transação t_i cria um registro em A_j */

R = *Retrieve* /* Transação t_i recupera um registro em A_j */

U = *Update* /* Transação t_i atualiza um registro em A_j */

D = *Delete* /* Transação t_i exclui um registro em A_j */

	A1	**A2**	**A3**
T1	RU		C
T2	R	U	R
T3	C	CUD	U

Figura IV.5: Exemplo de matriz CRUD

Uma vez que o *log* captura a ordem das operações de cada transação, esta matriz pode ser estendida em uma dimensão, a dimensão tempo, tornando-se um "cubo", o CUBO CRUD, ilustrado na Figura IV.6. Formalizando:

Seja $T = t_1, t_2, ..., t_m$ o conjunto das transações de uma aplicação;

Seja $A = a_1, a_2, ..., a_n$ o conjunto de arquivos/tabelas da mesma aplicação; e

Seja $H = h_1, h_2, ..., h_p$ o conjunto de períodos de tempo observados, onde $h_n < h_{n+1}$; e

Seja a (nova) relação $R \subseteq A \times T \times H = t_1a_1h_1, t_1a_2h_1, ..., t_ma_nh_p$ a relação que representa a tabela de fatos CUBO CRUD, onde cada $t_ia_jh_k$ é uma operação. Especificamente:

C = *Create* /* Transação t_i cria um registro em A_j no tempo H_k*/

R = *Retrieve* /* Transação t_i recupera um registro em A_j no tempo H_k*/

U = *Update* /* Transação t_i atualiza um registro em A_j no tempo H_k*/

D = *Delete* /* Transação t_i exclui um registro em A_j no tempo H_k */

H_1	A1	A2	A3	H2	A1	A2	A3	H3	A1	A2	A3
T1	R			T1	U			T1			C
T2	R			T2		U		T2			R
T3	C			T3		C		T3		U	

H_4	A1	A2	A3	H5	A1	A2	A3	H6	A1	A2	A3
T1				T1				T1			
T2				T2				T2			
T3		D		T3			U	T3			

Figura IV.6: Decomposição da Figura IV.5 em um CUBO CRUD.

Levando-se em consideração a Figura IV.6 é possível imaginar-se as seguintes consultas do tipo OLAP, ilustrado na Figura IV.7.:

- Caso 1: fixando-se a transação e o arquivo, variando-se a dimensão tempo, é possível determinar a ordem de execução dos comandos (CRUD) para um arquivo determinado que é referenciado pela transação "Y". Exemplificando: seja a transação e o arquivo alvo, T_3 e A_2 da figura IV.6, respectivamente, sabe-se que as operações realizadas são C, U e D, nesta ordem;
- Caso 2: fixando-se apenas a transação variando-se as dimensões arquivo e tempo, é possível determinar a ordem de execução dos comandos (CRUD) para cada arquivo que é referenciado pela transação "X". Exemplificando: seja a transação alvo, T_3 da figura IV.6, sabe-se que as operações realizadas são C(A1), C(A2), U(A2), D(A2), U(A3), *commit*, nesta ordem;
- Caso 3: fixando-se uma FAIXA (período) de tempo é possível determinar que transações ocorrem naquele período, de modo a que no futuro seja possível

agrupar transações que ocorrem proximamente. Exemplificando, para os tempos variando de H1 a H5:

T_1: R(A1), U(A1), C(A3), *commit*;

T_2: R(A1),U(A2),R(A3), *commit*;

T_3: C(A1),C(A2),U(A2),D(A2),U(A3), *commit*; nesta ordem.

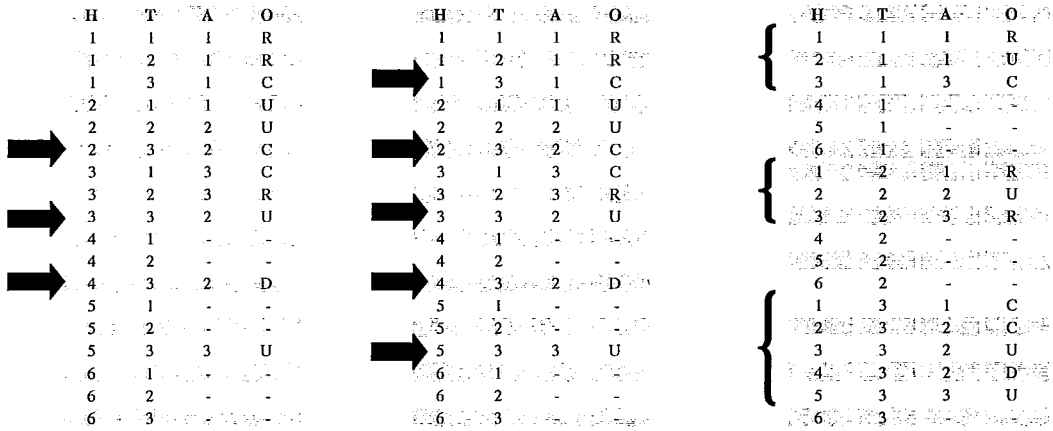


Figura IV.7: Casos 1, 2 e 3

E, principalmente, tendo sido obtido um CUBO CRUD a partir do *log* endógeno, é possível suprimir a dimensão tempo, deixando-se apenas as dimensões transação e arquivo/tabela, bem como as suas operações, sem repetição, obtendo-se a matriz CRUD, por engenharia reversa, conforme o proposto, conforme exibido na Figura IV.8.

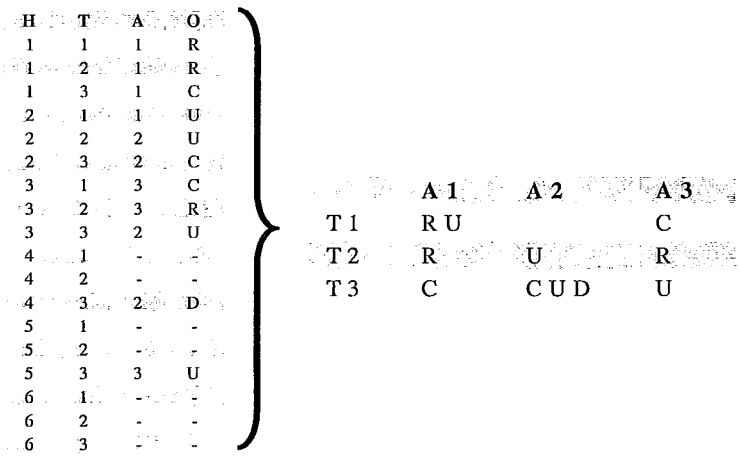


Figura IV.8: Geração de matriz CRUD

IV.4.2. O CUBO CRUD e as Regras de Negócios

Ao minerar o *log* de transações de banco de dados é possível descobrir regras relativas aos eixos de abstração funcional e de controle. Estas regras podem ser combinadas com regras obtidas dos metadados, para realizar-se a engenharia reversa

das regras de negócios que possuam formato assemelhado às regras associativas.

Porém, estas regras associativas associam operações CRUD no tempo, de modo que a ordem de ocorrência das operações dentro de uma transação de banco de dados, neste caso, é importante, conforme a descrição dos casos 1, 2 e 3 da seção IV.4.1.

Em (AGRAWAL *et al.*, 1993) a ordem dos itens de compra em uma transação de compra, na mineração de regras associativas, é irrelevante.

A seguir, são descritos exemplos de propostas de funcionalidades de mineração de dados desejáveis no processamento de consultas de um SGBD (AGRAWAL *et al.*, 1993), em aplicações exógenas, para o caso do comércio varejista (*market basket analysis*), com o seu correspondente endógeno (SULAIMAN & SOUZA, 1998d) para o caso da mineração de *logs*:

Caso 1:

- Encontre todas as regras que possuam "Diet Coke" como conseqüente. Estas consultas podem auxiliar em um planejamento para aumentar as vendas de Diet Coke (AGRAWAL *et al.*, 1993);
- Encontre todas as regras que possuam "exclua registro da tabela A" como conseqüente. Estas consultas podem auxiliar identificação de regras de negócios do tipo de "inferência" ou de "estímulo/resposta" cujo conseqüente é pré-determinado, ou todos os tipos de estímulo que levam à esta resposta (SULAIMAN & SOUZA, 1998d).

Caso 2:

- Encontre todas as regras que possuam "rosquinhas" como antecedente. Estas consultas podem auxiliar na determinação do impacto que a descontinuidade da venda de rosquinhas pode causar em outros produtos (AGRAWAL *et al.*, 1993);
- Encontre todas as regras que possuam "inserir registro na tabela B" como antecedente. Estas consultas podem auxiliar na determinação do impacto desta operação em outras transações, identificando-se as regras de negócios cujos conseqüentes correspondem aos antecedentes pré-determinados (SULAIMAN & SOUZA, 1998d).

Caso 3:

- Encontre todas as regras que possuam "salsicha" no antecedente AND "mostarda" no conseqüente. Estas consultas podem auxiliar na prospecção

de determinados padrões que determinem que outros produtos podem ser vendidos quando se vende salsicha e mostarda (AGRAWAL *et al.*, 1993);

- Encontre todas as regras que possuam "inserir registro na tabela C" no antecedente AND "excluir registro da tabela D" no conseqüente. Estas consultas podem auxiliar na prospecção de determinados padrões de comportamento semelhantes em transações distintas (SULAIMAN & SOUZA, 1998d), ou regras de inferência mais específicas.

Caso 4:

- Encontre todas as regras que relacionem itens localizados nas prateleiras *A* e *B* da loja. Estas regras podem auxiliar no planejamento da arrumação das prateleiras, determinando se a venda dos itens da prateleira *A* se relacionam com as vendas dos itens da prateleira *B* (AGRAWAL *et al.*, 1993);
- Encontre todas as regras que relacionam operações entre as tabelas *X* e *Y* no esquema *Z*. Estas consultas podem auxiliar no Planejamento de Sistemas de Informação, pela determinação das relações existentes entre a transação *Q* na tabela *X* e a transação *W* na tabela *Y* (SULAIMAN & SOUZA, 1998d). Trata-se de uma abordagem intertransações, beirando o tratamento de fluxos de trabalho.

Lendo-se (AGRAWAL *et al.*, 1993, SULAIMAN & SOUZA, 1998d) fica evidente a analogia que foi feita entre as prateleiras de uma loja e o esquema conceitual e lógico de banco de dados, entre as transações de vendas e as transações de bancos de dados e entre itens vendidos e operações de inserção e exclusão.

Uma tabela de fatos resultante da extração proveniente dos *logs* de transação possibilita ao menos dois tipos de consultas por análise de afinidade e análise de agrupamentos, respectivamente:

1. Relacionamentos intratransações do CUBO CRUD, que se referem a uma única operação no consequente da regra e que vão revelar regras mais simples; e
2. Relacionamentos intertransações do CUBO CRUD, que podem referenciar uma ou mais transações de banco de dados e que vão revelar regras mais complexas, possivelmente com componentes temporais.

Nos relacionamentos intratransações do CUBO CRUD, o foco do estudo é a identificação de regras cujo antecedente leva uma determinada operação CRUD no consequente.

Nos relacionamentos intertransações do CUBO CRUD, que pertencem à mesma transação de banco de dados, o foco do estudo se encontra na identificação de regras pela determinação de padrões de comportamento diferentes pelas relações de precedência e ordem entre estas operações.

Nos relacionamentos intertransações do CUBO CRUD, que pertencem à diferentes transações de banco de dados, o foco do estudo é a possível identificação da interferência da ocorrência da transação *A* na ocorrência da transação *B*, novamente por ordem ou precedência.

Levando-se em consideração a dimensão tempo, conforme já descrito, as regras associativas a que este trabalho se refere são implicações do tipo $X \Rightarrow Y$ onde X é um conjunto de operações da transação T_i, bem como Y.

Há aqui uma diferença sutil: pode haver em X uma operação U(x)' e, no conseqüente, Y também haver uma operação U(x)''. Porém, como há uma relação de precedência no tempo, então se U(x)' < U(x)'' $\Rightarrow$ U(x)' $\neq$ U(x)''. Deste modo são diferentes as operações "R(A1)", nos tempos H1 e H3:

T_1: R(A1),U(A2),R(A1), *commit*;

Além disso são distintas as transações abaixo:

T_1: R(A1),U(A2),R(A3), *commit*;

T_2: R(A3), R(A1),U(A2), *commit*;

Estas relações de precedência são parecidas com aquelas descritas na tabela II.3. Com estas observações, o restante do formalismo é idêntico a (AGRAWAL *et al.*, 1993): seja $I=\{i_1, i_2,..., i_n\}$ o conjunto de operações sobre dados da base de dados alvo. Uma regra de associação R definida sobre I é uma implicação da forma $X \Rightarrow Y$ onde:

1. $X \subset I$; $Y \subset I$;
2. $X \neq \emptyset$; $Y \neq \emptyset$; e
3. $X \cap Y = \emptyset$, observando-se a diferença entre operações idênticas em tempos distintos.

O fator de suporte de uma regra $X \Rightarrow Y$ é definido pela porcentagem de transações que incluem todos os itens do conjunto $X \cup Y$, e indica a relevância da regra: suporte(R) = T_{XUY} / T. O fator de confiança de uma regra $X \Rightarrow Y$ é definido pela

porcentagem de transações que incluem os itens X e Y em relação a todas que incluem os itens de X, e se refere ao grau de satisfatibilidade da regra: confiança(R) = T_{XUY} / T_X

Nem todas as regras de negócios são passíveis de serem representadas através de regras associativas, usando-se este método. Recordando a Figura III.4, usando-se a taxonomia de (DATE, 2000a), regras de negócios podem ser de restrição ou de derivação. Dentre os tipos de regras apresentadas na Figura III.4 e exemplificadas na Figura III.1 (SULAIMAN *et al.*, 2000), somente as regras de "restrição de estímulo/resposta" e de "inferência", podem ser diretamente mineradas usando-se este método no contexto apresentado.

Isto não significa que outros tipos de regras não possam ser mineradas com este método: quaisquer tipos de regras em que se registrem operações CRUD no *log* de banco de dados são passíveis de serem mineradas.

Outros tipos de regras, onde o fluxo de controle não é registrado por operações CRUD, e portanto no momento da sua execução as regras de negócio são executadas exclusivamente por operações em memória, são melhor mineradas com análise de códigos fonte, utilizando-se técnicas de mineração de dados combinadas às técnicas de compiladores e engenharia de software, que em geral, também não se prestam a minerar as categorias de regras com o enfoque ora em questão.

Além disso, restrições de estímulo/resposta são "uma combinação de um evento (*triggering event*) ou o estímulo, e a ação (*triggered action*), ou a resposta" (DATE, 2000a), que constituem toda uma importante categoria de regras: os *triggers*. Em geral *triggers* acabam sendo implementados de modo procedimental, apesar do exemplo a seguir ser totalmente declarativo. Recordando-se a Figura III.1, item c: "Ao excluir um empregado, seus dependentes são automaticamente excluídos".

O evento a ser registrado é a retirada de um empregado, ou antecedente da regra. Quando este antecedente é satisfeito, ativa a retirada de seus dependentes, ou seja: a ação, o conseqüente da regra. O padrão a ser encontrado é o seguinte:

...,D(empregado),...,D(dependente),...,D(dependente),..., *commit*

Ou seja: "D(empregado) $\Rightarrow$ D(empregado.dependente)". Podendo acontecer isoladamente em uma transação:

- D(empregado), ou seja empregado sem dependente;
- D(dependente), ou seja dependente emancipado; e
- Não ocorrência de D(empregado) e D(dependente).

Existe um grande número de regras deste tipo, não mapeadas diretamente por nenhum outro método, escondidas no código fonte de programas procedimentais, com conseqüências diretas na recuperação de regras que são significativas na reengenharia do sistema herdado, conforme será visto a seguir.

Supondo-se relações binárias em um esquema conceitual de banco de dados e adotando-se as restrições da Figura IV.9, exemplifica-se restrições utilizando-se apenas o "relacionamento um para muitos", onde a entidade do lado esquerdo em cada um dos casos, 1, 2, 2a e 3 representa "departamento" e entidade do lado direito representa "empregado". Uma abordagem exaustiva pode ser encontrada em (SULAIMAN e TANAKA, 1997b).

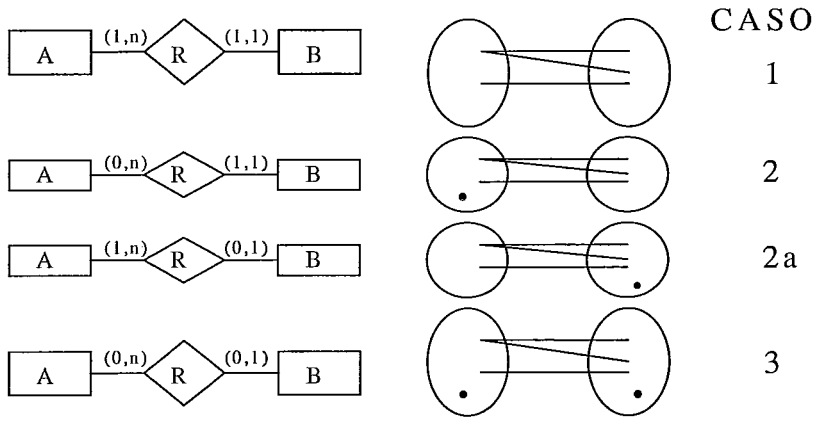
Sempre que há obrigatoriedade de participação de um elemento de um conjunto na relação com outro conjunto, tal como nos dois lados do caso 1, no lado direito do caso 2 ou no lado esquerdo do caso 2a da Figura IV.9, é possível encontrar-se seqüências de operações CRUD em determinada ordem, na mesma transação. Mesmo quando estas operações residem em transações distintas, a ordem das operações CRUD é mantida. Este *template* é semelhante ao exemplo anterior entre empregados e dependentes: "não existem dependentes sem empregado".

É interessante notar que a regras c, d, e da Figura III.2, expressas em OCL (WARMER & KEPPLE, 1999) poderiam vir a ser obtidas pela mineração de seus *logs*, em um contexto bem mais sofisticado do que o apresentado neste trabalho.

IV.4.3. O CUBO CRUD e os Fluxos de Trabalho

Em geral, *logs* de transações grupados no tempo se referem a uma atividade. Algumas vezes transações agrupadas resolvem parte de uma atividade maior definida previamente. Neste trabalho usa-se o termo "transação abstrata" em oposição a "transação concreta", da seguinte maneira:

- Transação concreta – transação obtida por engenharia reversa pelo simples registro das atividades acontecidas em um período de tempo; e
- Transação abstrata – registro da transação obtida por engenharia reversa pela obtenção das seqüências maximais (AGRAWAL & SRIKANT, 1995) de atividades e que representam "a interpretação computacional de uma atividade do negócio" (SULAIMAN & SOUZA, 1998d), ou a porção computacional da atividade, implementada em um banco de dados.



Caso 2: A (0,n) - B (1,1).

Restrições informais: A classe A é o conjunto dos departamentos, a classe B é o conjunto dos empregados e não existe empregado em mais de um departamento. Pode-se ter departamentos com mais de um empregado. **Não existem empregados sem departamento.** Pode-se ter departamentos sem empregados. Este é o caso mais comum de relacionamento binário. Restrições formais:

$\forall a \in A, \forall b \in B, \forall (a,b) \in R$

1. $(\exists a_1 \in A, \forall b \in B \mid (a_1,b) \notin R)$ /* existe um a que não participa */
2. $(\exists b_2, \exists b_3 \in B, \exists a_2 \in A \mid (a_2,b_2) \in R, (a_2,b_3) \in R)$ /* existe um a que participa em mais de um par */
3. **$(\forall b \in B, \exists a_1 \in A \mid (a_1,b) \in R)$ /* todo b participa */**
4. $((\forall b \in B, \exists a_1 \in A \mid (a_1,b) \in R) \Rightarrow \neg\exists a_2 \in A, a_2 <> a_1 \mid (a_2,b) \in R)$ /* todo b participa no máximo uma vez */

Figura IV.9: Relações binárias e suas restrições: exemplo

Recordando (AGRAWAL *et al.*,1998), "um processo de negócios é um conjunto de atividades separadas" e "uma atividade é definida como uma ação que é uma unidade semântica em algum nível, ou uma função que modifica o estado de um processo". Porém, sob o ponto de vista de (ELMASRI e NAVATHE, 2000) "uma transação é a execução de um programa que faz acesso ou modifica o conteúdo de um banco de dados" e uma transação atômica é "uma unidade de trabalho ou uma unidade lógica do processamento de um banco de dados".

É possível encontrar formalismos para transações em bancos de dados em (OZSU e VALDURIEZ, 1999, GRAY e REUTER, 1993, LYNCH *et al.*, 1994). Todos estes são igualmente expressivos e também citam tipos de transações quanto às seqüências de operações de leitura e gravação sobre objetos de dados. O formalismo utilizado neste trabalho é baseado em (OZSU e VALDURIEZ, 1999), com pequenas alterações. A alteração marcante é quanto ao nível de abstração: ao invés das operações

serem apenas de *read* ou *write*, as operações usadas neste modelo são as operações CRUD.

Deste modo, qualquer operação $O_{ij}(x)$ denota a operação O_j de uma transação T_i que opera sobre a entidade x da base de dados, sendo que $O_{ij} \in$ {*Create, Retrieve, Update, Delete*} e OS_i é o conjunto de todas as operações em T_i , ou seja, $OS_i = \bigcup O_{ij}$, e N_i é a condição de terminação para T_i, onde $N_i \in$ {*abort, commit*}.

Além disso, T_i é uma Quasi Order, sobre suas operações e a condição de terminação. Uma Quasi Order $Q=\{\Sigma_i,<_i\}$ define uma ordem entre os elementos de Σ, sendo que Σ consiste das operações e a condição de terminação de uma transação e $<$ indica a ordem de execução destas operações, ou seja, sua relação de precedência.

As restrições do modelo são as seguintes: a primeira define o domínio como sendo o conjunto de seqüências de operações CRUD , que compõem a transação mais a condição de terminação, que pode ser *commit* ou *abort*; a segunda especifica a relação de ordem entre as operações conflitantes de uma transação, sendo que operações conflitantes são aquelas em que há gravação de dados sobre o mesmo objeto de dados, no mesmo momento; e a terceira determina que a condição de terminação sempre segue todas as outras operações da transação. Deste modo uma transação de banco de dados T_i é uma Quasi Order $T_i=\{\Sigma_i, <_i\}$, onde:

1. $\Sigma_i = OS_i \cup \{N_i\}$;
2. Para quaisquer duas operações O_{ij}, $O_{ik} \in OS_i$,

   Se $O_{ij} =$ {C(x)} or {R(x)} or {U(x)} or {D(x)} e

   $O_{ik} =$ {C(x)} or {U(x)} or {D(x)}

   Para qualquer item de dados x,

   Então $O_{ij} < O_{ik}$ ou $O_{ik} < O_{ij}$;
3. $\forall O_{ij} \in OS_i$, $O_{ij} <_i N_i$

Onde $O_{ij} \in$ {*Create, Retrieve, Update, Delete*}; e

$OS_i = \bigcup_j O_{ij}$ e $N_i \in$ {*Abort, Commit*}

Um CUBO CRUD também identifica de modo simples todos os programas, ou transações que foram ativadas em uma faixa de tempo pré-determinada. A visualização é algo semelhante à Tabela II.2. Desde que existam listas comparáveis de execuções de programas, é possível estimar padrões de seqüências que sejam candidatos a fluxos de trabalho.

A teoria de Mineração de Padrões Seqüenciais é descrita em (AGRAWAL & SRIKANT, 1995) e, neste trabalho é feita uma instanciação para o caso de transações de banco de dados. Evidentemente, o que se quer obter de modo semi-automático são fluxos de trabalho tão fidedignos quanto possível ao modelo real, tendo-se como entrada as seqüências de programas ou transações efetivamente ocorridas em um período de tempo. As restrições do modelo são as seguintes:

1. Uma seqüência de dados, ou simplesmente seqüência, é uma lista ordenada de programas;
2. Uma seqüência é um padrão seqüencial;
3. A ordem dos programas em cada seqüência se refere à ordem de execução de cada programa;
4. Cada programa é mapeado para um identificador inteiro;
5. Um leilão suporta uma seqüência *s* se *s* está contida na seqüência de programas deste leilão;
6. O suporte de uma seqüência é a fração do total de leilões que suportam esta seqüência.

   Supondo-se as seguintes ocorrências de leilões:

   A <1,5,2,3,4>; B <1,3,4,3,5>; C <1,2,3,4>; D <1,3,5>; E <4,5>

A seqüência "A" é um leilão, no qual a seqüência de programas executados é <1,5,2,3,4>. Na seqüência "B" a seqüência de programas é <1,3,4,3,5> e assim por diante. Na Tabela IV.1 há um resumo das seqüências de programas A, B, C, D e E juntamente com a ocorrência do que se convenciona chamar neste trabalho de "transação concreta", que consiste no registro do rastreamento, ou *log*, puro e simples da ocorrência de cada programa ou transação, para todos os comprimentos possíveis para as seqüências existentes. Trata-se de uma abordagem assemelhada à criação de um dicionário inspirado no algoritmo LZ77 tal como reportado em (NELSON, 1991).

Tabela IV.1: Transações concretas

Seqüências de programas	Janela de tamanho 1	Janela de tamanho 2	Janela de tamanho 3	Janela de tamanho 4	Janela de tamanho 5
A<1,5,2,3,4> B<1,3,4,3,5> C<1,2,3,4> D<1,3,5> E<4,5>	<1> 4 <2> 2 <3> 5 * <4> 4 <5> 4	<1,5> 1 <5,2> 1 <2,3> 2 <3,4> 3 <1,3> 2 <4,3> 1 <3,5> 2 <1,2> 1 <4,5> 1	<1,5,2> 1 <5,2,3> 1 <2,3,4> 2 <1,3,4> 1 <3,4,3> 1 <4,3,5> 1 <1,2,3> 1 <1,3,5> 1	<1,5,2,3> 1 <5,2,3,4> 1 <1,3,4,3> 1 <3,4,3,5> 1 <1,2,3,4> 1	<1,5,2,3,4> 1 <1,3,4,3,5> 1

1. Chama-se "transação concreta" qualquer seqüência contígua de execução de programas; 2. Ao lado de cada seqüência de programas de tamanhos 1,2,3,4 e 5, está registrado sua freqüência; 3. No caso grafado com (*) foram contadas as repetições de ocorrências em uma mesma seqüência.

Em oposição ao conceito de "transação concreta" é utilizado o conceito de "transação abstrata" que se refere à transação obtida por engenharia reversa pela obtenção das seqüências maximais (AGRAWAL e SRIKANT, 1995) de atividades e que representam "a interpretação computacional de uma atividade do negócio" (SULAIMAN e SOUZA, 1998d), ou a porção computacional da atividade, implementada em um banco de dados. Formalizando:

Uma seqüência $< a_1, a_2, ..., a_n>$ de programas está contida em outra seqüência $<b_1, b_2,..., b_m>$ de programas se existirem inteiros $i_1<i_2<...<i_n$ tais que $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, ..., $a_n \subseteq b_{in}$.

Utilizando-se as mesmas seqüências de programas anteriores: A <1,5,2,3,4>; B <1,3,4,3,5>; C <1,2,3,4>; D <1,3,5>; E <4,5> e usando-se a propriedade de uma seqüência "estar contida" em outra, bem como com uma restrição adicional, o conceito de "suporte mínimo", ou seja, a geração de todas as seqüências contidas em pelo menos um número pré-determinado de seqüências, exemplifica-se estas duas restrições na Tabela IV.2. Definição: Em um conjunto de seqüências, uma seqüência é maximal se não está contida em nenhuma outra seqüência (AGRAWAL e SRIKANT, 1995). No caso da Tabela IV.2, as seqüências maximais com suporte mínimo 40% (2 ou mais ocorrências) são as seguintes: <1,2,3,4> 2, <1,3,5> 2 e <4,5> 2.

Tabela IV.2: A caminho de transações abstratas

Seqüências de programas	tamanho 1	tamanho 2	tamanho 3	tamanho 4	tamanho 5
A<1,5,2,3,4> B<1,3,4,3,5> C<1,2,3,4> D<1,3,5> E<4,5>	<1> 4 <2> 2 <3> 3* <4> 4 <5> 4	<1,2> 2 <1,3> 4 <1,4> 3 <1,5> 3 <2,3> 2 <2,4> 2 <3,4> 3 <3,5> 2 **<4,5> 2**	<1,2,3> 2 <1,2,4> 2 <1,3,4> 3 **<1,3,5> 2** <2,3,4> 2	**<1,2,3,4> 2**	----------

1. Suporte mínimo: 40% (ou seja, participação em pelo menos 2 leilões) 2. Ao lado de cada seqüência de programas de tamanhos 1,2,3,4 e 5, está registrado o seu suporte; 3. No caso grafado com (*) **não** foram contadas as repetições de ocorrências em uma mesma seqüência; 4. seqüências em negrito são maximais.

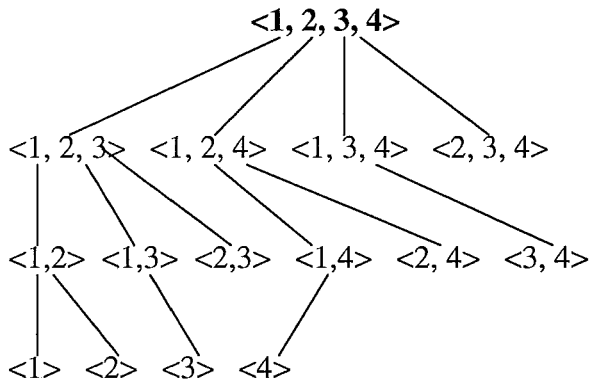


Figura IV.10: Exemplo de seqüência maximal

IV.5. Descrição do Experimento

Nesta seção é descrito o experimento realizado sobre as bases de dados endógenas. Para um melhor entendimento do problema, será enfocado primeiramente o negócio, sob o ponto de vista exógeno, de modo a que se possam inferir as fronteiras entre os dois pontos de vista. Trata-se de uma breve descrição dos mecanismos básicos das políticas monetárias, exercidas pelo Banco Central do Brasil (BACEN), através dos seguintes departamentos da Diretoria de Política Monetária (DIPOM):

- Departamento de Operações das Reservas Internacionais (DEPIN);
- Departamento de Operações Bancárias e de Sistemas de Pagamentos (DEBAN); e
- Departamento de Operações de Mercado Aberto (DEMAB), que será o

enfoque neste estudo.

IV.5.1. Descrição do problema exógeno

Segundo (COLLARES *et al.*, 2001) "O artigo 10 da Lei 4595 de 31 de dezembro de 1964, outorgou ao Banco Central do Brasil a atribuição de efetuar operações de compra e venda de títulos públicos federais, as operações de mercado aberto".

Em linhas gerais o mecanismo é operacionalizado da seguinte maneira (COLLARES *et al.*, 2001): "quando o BACEN vende títulos federais ao sistema bancário, provoca a redução das reservas bancárias; inversamente, no caso de compra destes títulos pelo BACEN, acontece o aumento destas reservas. As ditas intervenções, ou operações de compra e venda de títulos são de dois tipos: compromissadas e definitivas. Nas operações compromissadas, o BACEN toma (ou empresta) recursos a um prazo definido – usualmente um dia (*overnight*) – vendendo (ou comprando) títulos com o compromisso de recomprá-los (ou revendê-los) em data combinada, a um determinado preço.

"Nesse tipo de operação, dito leilão informal ou *go-around*, o BACEN atua no mercado através de instituições *dealers*, credenciados periodicamente pelo BACEN. Nas operações definitivas, o título incorpora-se à carteira da instituição compradora. A compra ou venda definitiva realizada pelo BACEN se dá também através dos leilões informais ou formais, dos quais podem participar todas as instituições financeiras".

"Nas operações definitivas, o título incorpora-se à carteira da instituição compradora. A compra ou venda definitiva realizada pelo BACEN se dá também através dos leilões informais ou dos leilões formais, dos quais podem participar todas as instituições financeiras. Os leilões informais realizam-se por via telefônica apenas com os *dealers*, enquanto os formais se processam mediante propostas enviadas por escrito. O BACEN opera leilões formais com títulos novos (mercado primário) e com os que fazem parte de sua carteira e, portanto já tem prazo decorrido".

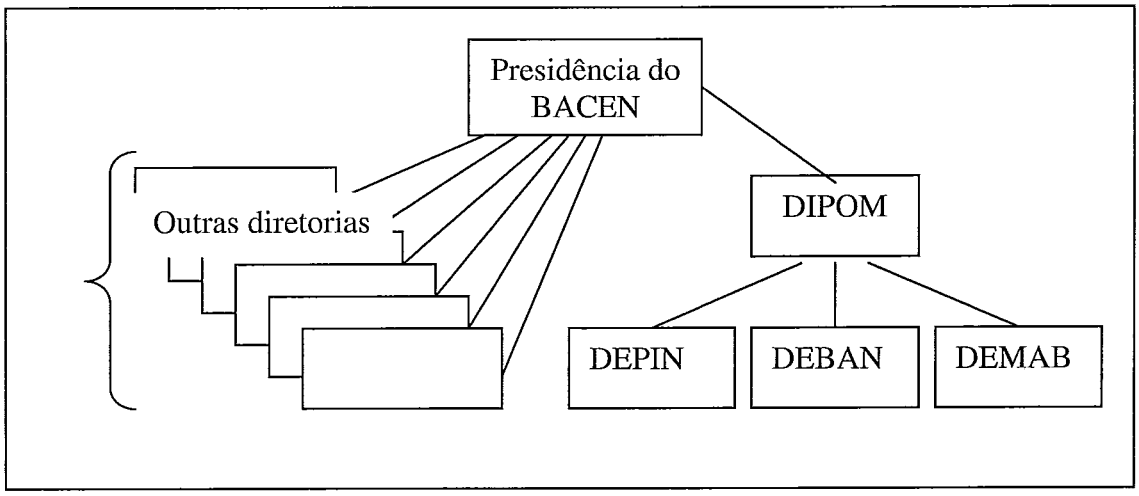


Figura IV.11: Esquema simplificado do organograma do BACEN

"O ajuste diário da liquidez é realizado através das operações compromissadas, com várias intervenções do BACEN. O processo pode ser descrito, sinteticamente, da seguinte forma: antes do mercado operar, o BACEN estima se há excesso de reservas no sistema bancário (neste caso o BACEN está *undersold*). Essa estimativa é obtida através de consultas a diversas fontes, entre as quais os *dealers*, referentes a operações que afetam as reservas bancárias".

"Com sede no Rio de Janeiro, a missão do DEMAB é manter organizado e ativo o mercado de títulos públicos. É a unidade do BACEN que gerencia e executa as operações de mercado aberto, com vistas a adequar os agregados monetários e a taxa de juros às diretrizes e disposições da política monetária. Sua atenção visa reacomodar, diretamente, a liquidez entre os agentes financeiros superavitários e deficitários. Com base em estimativas sobre o comportamento das operações cotidianas que afetam o nível das reservas bancárias, por meio da mesa de *open*, o DEMAB compra e vende moeda no curtíssimo prazo. O DEMAB é também responsável pelo Sistema Especial de Liquidação e Custódia (SELIC)".

"Atualmente, o processo transcorre em ambiente eletrônico, o que exige que a instituição financeira esteja registrada no **Sistema Oferta Pública Formal Eletrônica – OFPUB**, conforme estabelecido na Circular 2727, de 14 de novembro de 1996, que regulamenta o funcionamento do SELIC."

Maiores detalhes sobre o negócio da política de mercado aberto do BACEN podem ser obtidos em (COLLARES *et al.*, 2001). Uma perspectiva sobre bancos centrais em geral pode ser obtida em (BLINDER, 1999). Uma abordagem histórica do Sistema Financeiro Nacional (SFN) pode ser obtida em (ANDREZO & LIMA, 2001). A

referência usual para uma introdução sobre mercado financeiro no Brasil é (FORTUNA, 2000).

IV.5.2. Descrição do problema endógeno

A geração de *logs* para a recuperação de desastres em SGBDs é corriqueira, porém a geração de *logs* endógenos para auditoria, apesar de previsto nos manuais dos produtos, não costuma ser realizada. A geração de *logs* de auditoria costuma ser custosa em tempo de processamento e espaço em disco, e a cultura corrente na geração deste tipo de *log* é a inclusão de comandos de gravação para o rastreamento de determinadas transações nos próprios programas de aplicação.

Os *logs* endógenos são gerados em formato proprietário, com aspecto de *dump* de memória, dependente do SGBD, e portanto é necessário que se estabeleça um processo de conversão, que corresponda genericamente aos passos 2, 3 e 4 do processo KDD, com o objetivo de alimentar uma base de dados em um formato canônico chamado CUBO CRUD.

Nesta seção são descritos os requisitos do sistema OFPUB, em operação há mais de dez anos e implementado em DMSII com programas escritos em COBOL. Esta descrição foi obtida com o auxílio de um dos analistas deste sistema e trata-se de um esforço de reengenharia baseada no conhecimento humano deste analista.

O formato em que este conhecimento é exposto é compatível com a UML, e particularmente ao método proposto em (SCHNEIDER e WINTERS, 2001) para a elicitação de requisitos de sistemas com diagramas de *use case*.

A Figura IV.12 exibe uma versão de um diagrama de *use cases*, relativo ao leilão de ofertas de títulos públicos do Banco Central do Brasil, estão em destaque os *use-case* harchurados, que constituem os processos relevantes para este trabalho. As etapas de uma ocorrência normal do processo de negócio OFPUB são as seguintes:

1. cadastramento da oferta;
2. liberação da oferta para o mercado;
3. lançamento de propostas;
4. apuração preliminar;
5. apuração definitiva;
6. divulgação do resultado;
7. liquidação da oferta.

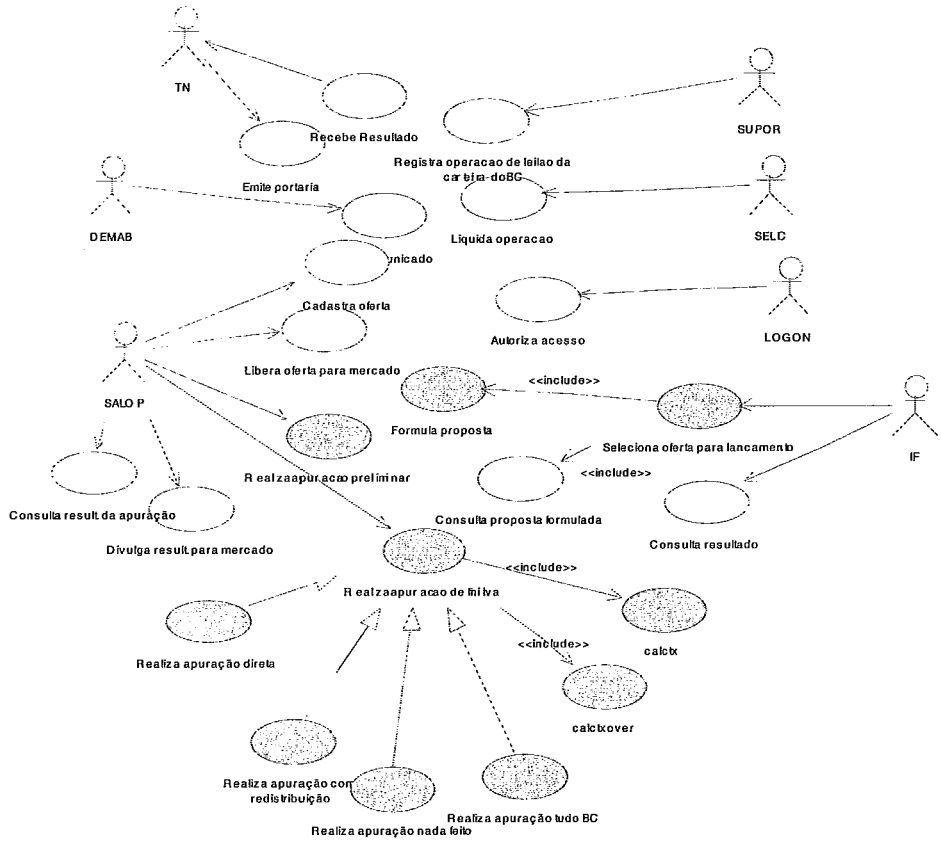


Figura IV.12: Diagrama de *use cases* com casos de uso harchurados

Conforme a experiência junto às instituições pesquisadas durante a confecção deste trabalho, também não é comum às empresas disponibilizarem seus *log*s, por motivos de segurança. Afinal, apesar de se tratar de uma análise sobre *logs* endógenos, nada impediria a análise do comportamento dos usuários de um dado sistema, por exemplo.

Porém, esta análise está descartada nos objetivos deste trabalho, havendo um compromisso com a instituição fornecedora das bases de dados endógenas neste sentido, como pré-requisito para o uso das bases geradas.

Não há qualquer referência neste trabalho que permita a identificação de um indivíduo ou empresa, bem como seu comportamento na interação com o sistema ora em análise.

O sistema OFPUB realiza a automação do leilão de ofertas públicas do BACEN. Cada leilão registra as propostas das instituições financeiras, apurando os melhores preços, de acordo com critérios pré-definidos pelo emissor, o Tesouro Nacional, ou em comunicados, o BACEN, enviando o resultado para as instituições. O sistema OFPUB

é um subsistema do SELIC, e funciona vinculado ao sistema de segurança de acesso e permissionamento do BACEN, sistema LOGON.

Existe um período crítico para ao qual o sistema não pode falhar e nem ter queda de desempenho. Considera-se o período crítico do leilão o intervalo de tempo de compreendido entre os dez minutos anteriores ao horário final para o lançamento das propostas e os dez minutos subsequentes, quando se realiza a apuração preliminar. As restrições de alto nível, regras de negócios ou pré requisitos, para a ocorrência normal de leilão são as seguintes:

- A instituições financeiras devem estar cadastradas no sistema SELIC;
- As instituições financeiras precisam pertencer a determinadas categorias (naturezas), interagindo com o sistema através de contas específicas que devem pertencer a determinados tipos de conta e de custódia;
- O leilão só pode executar em uma rede proprietária com características de segurança previamente definida, estado registrada na Rede de Teleprocessamento do Mercado, RTM;
- O responsável pela Instituição Financeira deve entrar em contato com o BACEN para obter uma senha de administrador no sistema LOGON;
- A instituição financeira deve cadastrar supervisores e operadores conforme a classificação do sistema LOGON para poder operar; e
- O papel a ser ofertado tem que estar cadastrado no sistema SELIC.

IV.5.3. Resultados obtidos no experimento

No caso específico dos *logs* provenientes do OFPUB, no ambiente DMSII/UNISYS, são gerados dois arquivos endógenos, de formato assemelhado, porém de funções complementares:

1. O primeiro arquivo, de tamanho menor, mapeia todos os programas referenciados em um dado leilão; e
2. O segundo arquivo, de tamanho maior, contém o registro de todas as alterações, inclusões e retiradas de cada objeto de dados pertencentes a qualquer estrutura de dados referenciada. Cada registro lógico do arquivo maior contém também um identificador do programa que o gerou, cujo nome completo se encontra no arquivo menor.

A extração do CUBO CRUD se dá sobre os dois arquivos além de informações provenientes do catálogo.

Neste experimento os leilões analisados são aqueles ocorridos nos dias 11, 15, 16, 17 e 18 de outubro de 2001, bem como 21, 22, 26, 27, 28 e 29 de novembro de 2001, havendo ainda leilões dos dias 18/07, 03/08, 12,13,15, 18 e 19 de setembro e 03 de outubro de 2000.

Os arquivos menores obtidos em outubro e novembro de 2001 possuem 454 kb e 441 kb, respectivamente. Já os arquivos maiores correspondentes possuem 75412 kb e 60315 kb, respectivamente. Para efeito de exemplificação, a Figura IV.13. exibe o menor leilão reportado, dentre os pesquisados, ocorrido em 17 de outubro de 2001.

A parte superior esquerda da Figura IV.13. mostra a seqüência de execução dos programas em 17 de outubro de 2001, obtida após a limpeza do arquivo menor correspondente. A parte superior à direita contém a identificação numérica de cada programa correspondente. A parte inferior da Figura IV.13. contém a seqüência de programas ocorridos em 17 de outubro de 2001.

```
10/17/2001 8:19:04 BOFPUB/DBCONTRO
10/17/2001 8:19:06 OFPUB/PO/002000
10/17/2001 8:24:24 OFPUB/PO/006000
10/17/2001 8:24:48 OFPUB/PO/PROPSTM
10/17/2001 8:24:59 OFPUB/PO/CALCTX
10/17/2001 8:25:05 OFPUB/PO/PRELIM
10/17/2001 8:25:22 OFPUB/PO/LIBERA
10/17/2001 8:29:10 OFPUB/PO/APUDIR
10/17/2001 8:29:22 OFPUB/PO/RELTXM
10/17/2001 8:29:23 OFPUB/PO/TMPPROP
10/17/2001 8:29:33 OFPUB/PO/LIBERA
10/17/2001 8:29:39 OFPUB/PO/015000
10/17/2001 11:43:30OFPUB/PO/004000
10/17/2001 12:11:24OFPUB/PO/004000
10/17/2001 12:57:05OFPUB/PO/004000
10/17/2001 17:24:30OFPUB/PO/004000
10/17/2001 23:14:09OFPUB/PO/DELETE
10/17/2001 23:14:55OFPUB/PO/FIM ON
```

```
01 DBOFPUB/CATCHUP/
02 DBOFPUB/DBCONTRO
03 OBJECT/INQUIRY/D
04 OFPUB/PO/002000
05 OFPUB/PO/004000
06 OFPUB/PO/005000
07 OFPUB/PO/006000
08 OFPUB/PO/009000
09 OFPUB/PO/013000
10 OFPUB/PO/015000
11 OFPUB/PO/APUDIR
12 OFPUB/PO/CALCTX
13 OFPUB/PO/DELETE
14 OFPUB/PO/FIM ON
15 OFPUB/PO/LIBERA
16 OFPUB/PO/PRELIM
17 OFPUB/PO/PROPSTM
18 OFPUB/PO/RELTXM
19 OFPUB/PO/TMPPROP
```

```
Programas ocorridos em 17/10/2001:
2; 4; 7; 17; 12; 16; 15; 11; 18; 19; 15; 10; 5; 5; 5; 13; 14;
```

Figura IV.13: Exemplo de leilão com codificação de programas.

A Figura IV.14. mostra um exemplo de registro de arquivo menor com seus trechos de cabeçalho e conteúdo. Estão destacados os principais campos a serem

extraídos deste tipo de registro: a identificação do programa ativado, a data e a hora em que ocorreu a ativação.

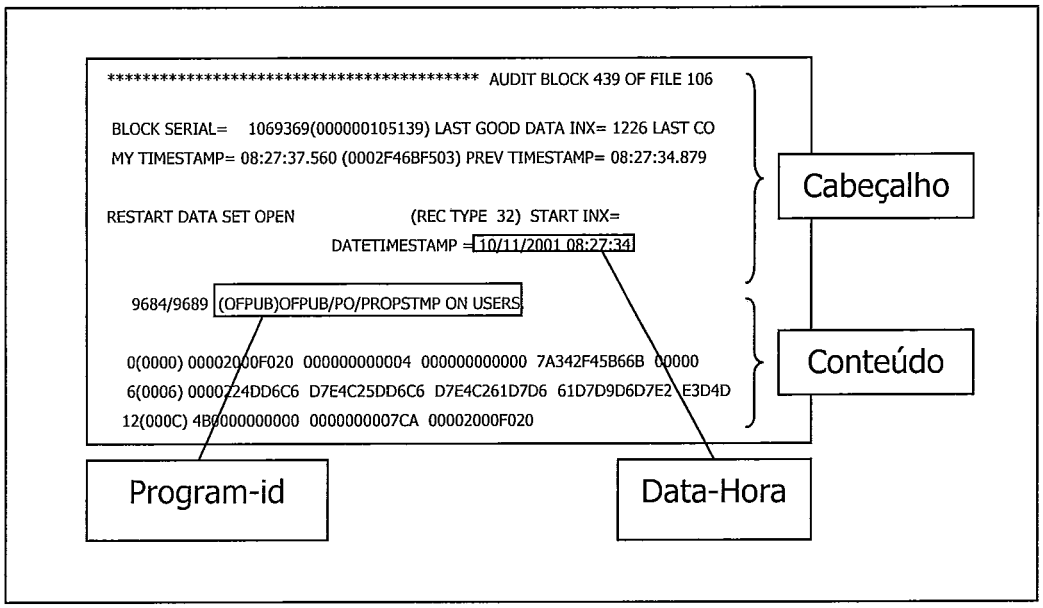


Figura IV.14: exemplo de trecho de log do "arquivo pequeno"

A Figura IV.15. combina exemplos de criação, modificação e retirada de estruturas de dados, que foram realizados pelo programa correspondente. Cada estrutura encontrada possui seu correspondente no catálogo do sistema alvo.

O CUBO CRUD, que constitui o formato canônico proposto, possui os seguintes campos:

COD: código seqüencial;

PGM: código do programa;

DATASTR: código da estrutura de dados;

CRUD: código da operação CRUD;

DATA: data da operação CRUD;

HORA: hora da operação CRUD;

HORARIO: horário da operação CRUD.

Neste CUBO CRUD poderiam ser suprimidos os campos COD e HORA, porém foram mantidos por questões de implementação. A Tabela IV.3. exibe um pequeno extrato de registros do CUBO CRUD.

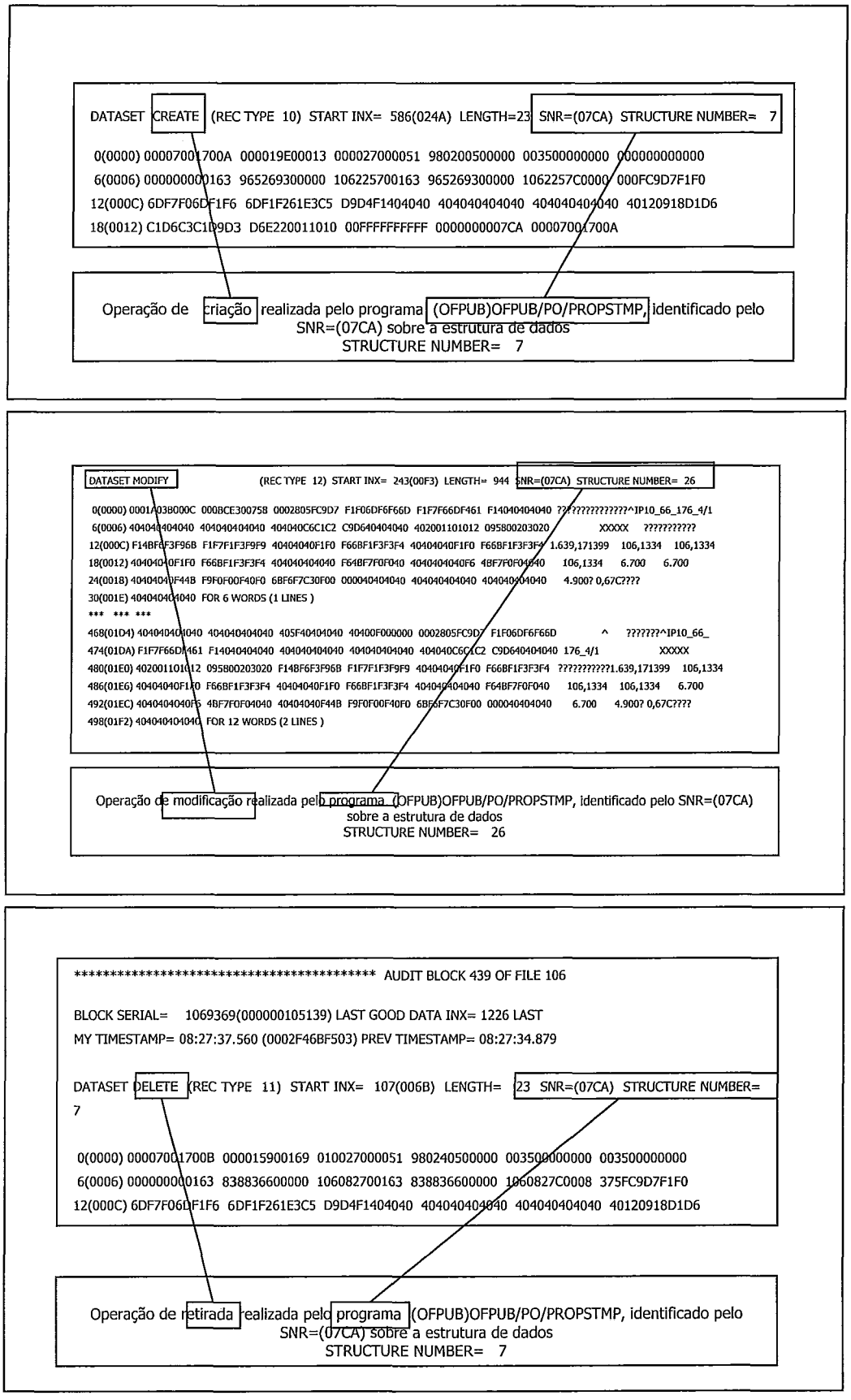


Figura IV.15: Exemplos de criação/modificação/retirada em *log* endógeno

Tabela IV.3: Alguns registros do CUBO CRUD

COD	PGM	DATASTR	CRUD	DATA	HORA	HORARIO
28	2	2	U	10/11/2001	08:14:16	08:36:49.9
29	2	2	U	10/11/2001	08:14:16	08:39:19.3
30	2	2	U	10/11/2001	08:14:16	08:59:20.1
375	17	7	D	10/11/2001	08:27:34	08:27:50.2
376	17	7	D	10/11/2001	08:27:34	08:27:50.5
377	17	7	D	10/11/2001	08:27:34	08:27:50.6
683	4	2	C	10/11/2001	09:49:44	10:03:31.4
684	8	2	C	10/11/2001	09:52:56	10:03:01.2
685	5	2	C	10/11/2001	09:58:57	10:13:31.1

A seguir são exemplificadas consultas OLAP, obtenção de regras de negócios e de trechos de fluxos de trabalho contra o CUBO CRUD, correspondentes às seções IV.4.1., IV.4.2. e IV.4.3 respectivamente.

IV.5.3.1. Exemplos de Consultas OLAP

Exemplo de consulta para o caso 1:

```
SELECT cod, pgm, datastr, data, hora, horario, crud
FROM cubocrud
WHERE pgm=4 And datastr=4 and data='10/17/2001';
```

Tabela IV.4: Resultado da consulta para o caso 1

olap1						
cod	pgm	datastr	data	hora	horario	crud
3942	4	4	10/17/2001	08:19:06		U
3943	4	4	10/17/2001	08:19:06		U
3944	4	4	10/17/2001	08:19:06		C
3945	4	4	10/17/2001	08:19:06	08:24:10.9	U
3946	4	4	10/17/2001	08:19:06	08:24:12.6	U
3947	4	4	10/17/2001	08:19:06	08:32:18.7	U

Exemplo de consulta para o caso 2:

```
SELECT cod, pgm, datastr, data, hora, horario, crud
FROM cubocrud
WHERE pgm=4 And data='10/15/2001';
```

Tabela IV.5: Resultado da consulta para o caso 2

olap2						
cod	pgm	datastr	data	hora	horario	crud
1751	4	4	10/15/2001	08:32:48		U
1752	4	4	10/15/2001	08:32:48	08:39:48.5	C
1753	4	4	10/15/2001	08:32:48	08:39:49.9	U
1754	4	4	10/15/2001	08:32:48	08:41:34.9	U
1755	4	2	10/15/2001	08:32:48	08:51:17.7	C
1759	4	2	10/15/2001	09:19:27	09:33:18.5	C
2495	4	4	10/15/2001	12:37:51		U
2496	4	4	10/15/2001	12:37:51		U
2497	4	4	10/15/2001	12:37:51	12:46:08.9	C
2498	4	4	10/15/2001	12:37:51	12:46:10.3	U
2499	4	4	10/15/2001	12:37:51	12:46:20.8	U
2500	4	2	10/15/2001	12:37:51	12:56:18.7	C

Exemplo de consulta para o caso 3:

```
SELECT cod, pgm, datastr, data, hora, horario, crud
FROM cubocrud
WHERE (hora>="11:00:01" And hora<="11:10:00")
ORDER BY cod, hora, horario, pgm, datastr, crud;
```

Tabela IV.6: Resultado da consulta para o caso 3

olap3						
cod	pgm	datastr	data	hora	horario	crud
689	4	4	10/11/2001	11:02:01		U
690	4	4	10/11/2001	11:02:01	11:02:35.8	U
691	4	2	10/11/2001	11:02:01	11:14:01.9	C
692	10	4	10/11/2001	11:02:56	11:03:38.5	U
693	10	2	10/11/2001	11:02:56	11:13:31.6	C
5638	4	4	10/18/2001	11:09:52		C
5639	4	4	10/18/2001	11:09:52		C
5640	4	4	10/18/2001	11:09:52	11:11:24.6	U
5641	4	4	10/18/2001	11:09:52	11:11:29.4	U
5642	4	4	10/18/2001	11:09:52	11:12:06.2	U
5643	4	4	10/18/2001	11:09:52	11:12:08.7	U
5644	4	2	10/18/2001	11:09:52	11:21:55.7	C

Exemplo de consulta para o caso 4, que corresponde à matriz CRUD, com contagem de freqüência:

```
SELECT DISTINCT pgm, datastr, crud, count(*)
```

FROM cubocrud

GROUP BY pgm, datastr, crud

ORDER BY pgm, datastr, crud;

Tabela IV.7: Resultado da consulta para o caso 4, com contagem de freqüência de cada tupla

olap4			
pgm	datastr	crud	Expr
2	2	D	74
2	2	U	253
3	4	U	1
4	2	C	18
4	4	C	9
4	4	U	29
5	2	C	20
6	2	C	1
7	1	U	18
7	2	C	8
8	2	C	8
9	2	C	8
9	26	C	809
9	26	U	667
10	2	C	12
10	4	U	12

olap4			
pgm	datastr	crud	Expr
11	4	U	9
11	7	U	720
12	7	U	1107
13	4	D	4
13	7	D	343
13	26	D	1198
14	1	U	5
15	1	U	18
16	4	U	9
17	4	U	9
17	7	C	720
17	7	D	496
17	26	U	292
19	26	U	365

	1	2	4	7	26
2		D U			
3			U		
4		C	C U		
5		C			
6		C			
7	U	C			
8		C			
9		C			C U
10		C	U		
11			U	U	
12				U	
13			D	D	D
14	U				
15	U				
16			U		
17			U	C D	U
19					U

Figura IV.16: Matriz CRUD

IV.5.3.2. Exemplos de Regras de Negócios obtidas

Para a geração de regras de negócios, o WIZRULE 3.05, um gerador automático de regras associativas, foi utilizado contra o CUBO CRUD, com os seguintes parâmetros:

1. Número total de registros: 7242;
2. Probabilidade mínima de regras "se-então": 0.95 (fator de confiança);
3. Número mínimo de casos: 40 (suporte mínimo de qualquer regra obtida: 0,55%).

Os horários foram agrupados em três períodos: o primeiro que vai desde o início da manhã ate o meio dia, quando são feitos os cadastramentos de ofertas. O segundo entre meio dia e uma hora, quando o leilão efetivamente acontece. O terceiro, após 13 horas, quando são realizadas as apurações.

Para esta consulta especifica foram geradas dezoito regras, das quais foram selecionadas as seguintes:

Regra 13: If PGM is 17.00 and CRUD is D

Then PERIODO is 1.00

Rule's probability: 1.000

The rule exists in 496 records.

Significance Level: Error probability < 0.001

Significado: esta regra reporta que quando o programa 17, que e o "PROPSTM", realiza operações de retirada, estas operações ocorrem no período de cadastramento de ofertas.

Regra 18: If PGM is 17.00 and DATASTR is 7.00 and PERIODO is 3.00

Then CRUD is C

Rule's probability: 1.000

The rule exists in 224 records.

Significance Level: Error probability is almost 0

Significado: quando o programa 17 opera sobre a estrutura de dados 7 no período de apuração, trata-se de uma operação de criação.

Regra 16: If DATASTR is 26.00 and CRUD is C

Then PERIODO is 2.00

Rule's probability: 0.994

The rule exists in 804 records.

Significance Level: Error probability is almost 0

Deviations (records' serial numbers):

5669, 5672, 5673, 5681, 5682

Significado: a estrutura de dados 26 e criada no período em que o leilão acontece.

Regra 17: If DATASTR is 26.00 and CRUD is D

Then PERIODO is 3.00

Rule's probability: 1.000

The rule exists in 1198 records.

Significance Level: Error probability is almost 0

Significado: a estrutura de dados 26 e destruída no período subseqüente, de apuração.

IV.5.3.3. Exemplos de trechos de Fluxos de Trabalho obtidos

Na obtenção de trechos de *workflow*, foi desenvolvido um programa baseado na teoria de (AGRAWAL & SRIKANT, 1995). Foi realizado um estudo exploratório sobre os leilões ocorridos nos dias 11, 15, 16, 17 e 18 de outubro, que tiveram as seguintes configurações, para cada leilão neste caso:

1. Tamanho: 22 - 2 4 7 17 12 16 15 11 18 19 15 5 8 5 4 4 4 8 5 4 3 5
2. Tamanho: 22 - 2 4 8 8 5 4 8 5 5 9 5 4 5 7 17 12 16 15 11 18 19 15
3. Tamanho: 22 - 2 9 4 3 5 3 9 1 8 7 17 12 16 15 11 18 19 15 1 6 10 9
4. Tamanho: 18 - 2 4 7 17 12 16 15 11 18 19 15 10 5 5 5 5 13 14
5. Tamanho: 22 - 2 4 5 7 17 12 16 15 17 12 16 15 11 18 19 15 11 18 19 15 10 4

As maximais obtidas foram as seguintes:

1. 2 4 7 17 12 16 15 11 18 19 15 4 — Freqüência: 2 Tamanho: 12
2. 2 4 7 17 12 16 15 11 18 19 15 5 5 5 5 — Freqüência: 2 Tamanho: 15
3. 2 4 5 8 7 17 12 16 15 11 18 19 15 — Freqüência: 2 Tamanho: 13
4. 2 4 5 9 7 17 12 16 15 11 18 19 15 — Freqüência: 2 Tamanho: 13
5. 2 9 4 5 7 17 12 16 15 11 18 19 15 — Freqüência: 2 Tamanho: 13
6. 2 4 5 7 17 12 16 15 11 18 19 15 10 — Freqüência: 2 Tamanho: 13

Uma análise mais detalhada sobre estes resultados parciais pode ser feita sobre a

"força" de cada trecho de seqüência, da seguinte maneira: para as seqüências representativas dos cinco leilões acima foram obtidas as seis seqüências maximais subseqüentes. Estas seqüências constituem as transações abstratas que, combinadas, são candidatas a trechos de fluxo de trabalho. A seqüência "7 17 12 16 15 11 18 19 15" é a mais forte: ocorre em 100% dos casos de modo contíguo, tanto nas seqüências de transações concretas quanto nas de transações abstratas. Como os sinais mais fortes constituem trechos de seqüência candidatas à espinha dorsal do fluxo de trabalho, este o caso desta seqüência.

A seqüência "2 4 7" é também forte, porém trata-se de uma seqüência que não se manifesta de forma contígua em todas as seqüências de transações concretas e que sugere, nas transações abstratas, a ocorrência de estruturas de decisão, que podem ensejar ocorrências de "2 4 5 8 7", "2 4 5 9 7","2 9 4 5 7" e "2 4 5 7".

A seqüência "15 5 5 5 5" sugere a ocorrência de uma estrutura de repetição, porém mereceria uma análise pormenorizada, por se tratar de um sinal ainda muito fraco. A Figura IV.17. ilustra uma representação esquemática da força destas seqüências, anotada junto as setas, de modo a representar a freqüência em que ocorrem nas transações abstratas. As setas mais espessas representam seqüências mais fortes, ou mais freqüentes, enquanto as setas mais finas representam seqüências mais fracas, ou menos freqüentes.

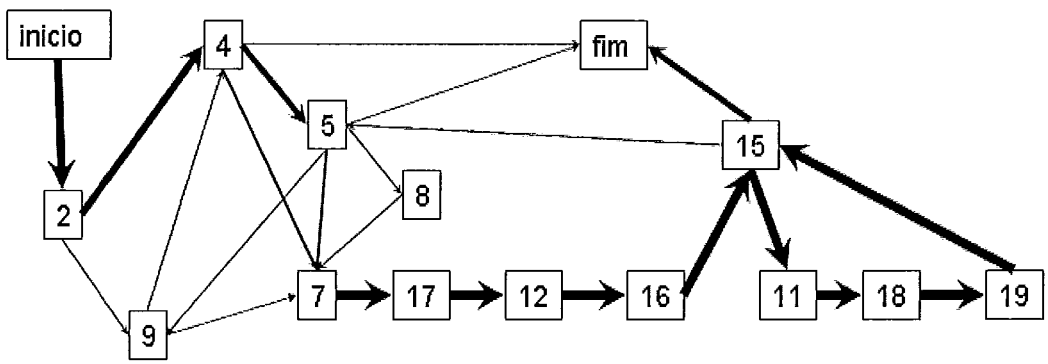


Figura IV.17: Trechos de seqüências de fluxos de trabalho

Capítulo V – Conclusões e Perspectivas Futuras

Neste capítulo encontram-se as conclusões do trabalho, que consistem de um resumo, análise das contribuições, bem como as perspectivas futuras.

V.1. Resumo

A engenharia reversa de sistemas herdados é um problema complexo que vem merecendo atenção da academia e da indústria. Muitos métodos foram propostos para solucioná-la, mas não existe um método que resolva todas as situações apropriadamente. Uma abordagem para complementar às soluções de engenharia reversa de sistemas herdados é a análise dos *logs* de bancos de dados combinado com os catálogos dos sistemas herdados.

Este trabalho propõe uma abordagem original para o tratamento do problema da engenharia reversa de sistemas herdados: um método que se utiliza de técnicas de armazéns de dados e de mineração de dados, na obtenção parcial de regras de negócios e fluxos de trabalho.

V.2. Análise das contribuições

Partindo-se da hipótese fundamental:

O ponto de vista deste trabalho está no fato de que o software gerenciador de log é uma aplicação endógena ao SGBD e o log de banco de dados é um armazém de dados endógeno ao SGBD onde é possível realizar-se mineração de dados para o apoio à decisão do Planejador de Sistemas de Informação.

Foi criado um método original para engenharia reversa de regras de negócios e fluxos de trabalho de sistemas herdados, baseado na abordagem não convencional da mineração de dados endógenos. Foram exploradas técnicas de armazém de dados e mineração de dados para a extração parcial de regras de negócios e fluxos de trabalho sobre estas bases. Para isto, foi criado o conceito de CUBO CRUD, bem como a possibilidade de realização de operações OLAP e de mineração de dados contra esta base de fatos. Porém, esta abordagem não resolve completamente o problema, antes porém configura um método barato, confiável e de caráter semi-automático.

V.3. Perspectivas futuras

Partindo-se da experiência deste trabalho é possível estendê-lo em pelo menos quatro vertentes:

1. A primeira e principal é a construção de uma metodologia de migração de sistemas herdados cuja composição contemple a mineração de dados endógenos, combinada com a mineração de regras de negócios em programas fontes dos mesmos sistemas;
2. A segunda contempla a última etapa do processo KDD, "consolidação do conhecimento obtido", que constitui uma etapa posterior à mineração de dados, e que está prevista como um trabalho futuro baseado nas experiências descritas nas seções III.2.2. e III.3.2.;
3. A terceira contempla a Integração de Sistemas Herdados proveniente da combinação do conhecimento advindo deste método em particular e da possível elaboração de uma metodologia citada no item anterior com métodos de integração existentes, tais como (SOUZA, 1986, SOUZA *et al.*, 1994) entre outros; e
4. A quarta contempla a pesquisa de logs em ambientes distribuídos e federados, o que pode tornar o modelo mais complexo, porém mais completo, trazendo à tona questões que envolverão não só a dimensão temporal, explorada neste trabalho, como também a dimensão espaço-temporal, referente à análise de *logs* de sistemas distribuídos e federados, que proporcionam situações de paralelismo e conflitos ainda não resolvidos na literatura.

Referências Bibliográficas

ABITEBOUL, S. , HULL, R. , VIANU, V., 1995, *Foundations of Databases*, 2 ed., Reading, USA, Addison-Wesley.

AGRAWAL, R. & SRIKANT, R., 1995, "Mining Sequential Patterns", In: *Proceedings of the International Conference on Data Engineering*, pp, 3-14, Taipei, Taiwan, Mar.

AGRAWAL, R. , GUNOPULOS, D. , LEYMANN, F., 1998, "Mining Process Models from Workflow Logs", In: *Proceedings of the International Conference on Extending Database Technology* , pp. 469-483, Valencia, Spain, Mar.

AGRAWAL, R. , IMIELINSKI, T., SWAMI A., 1993, "Mining Rules between Sets of Items in Large Databases", In: *Proceedings of the International Conference on Management of Data*, pp. 207-216, Washington, D.C., USA, May.

AGRAWAL, R. GUPTA, A SARAWAGI., S., 1997, "Modeling Multidimensional Databases", In: *Proceedings of the 13th International Conference on Data Engineering*, pp. 232-243, Birmingham, U.K., Apr.

ALLEN, R., 2001, "Workflow an Introduction". In: Layna Fischer (Ed), *The Workflow Handbook*, Florida, USA, Workflow Management Coalition, WFMC.

ANDREZO, A. F. & LIMA, I. S., 2001, *Mercado Financeiro aspectos históricos e conceituais*, São Paulo, FIPECAFI/USP e Pioneira.

BACHMAN, C. W., 1969, "Data Structure Diagrams", *Journal of ACM SIGBDP*. v. 1, n. 2 (Mar), pp. 4-10.

BLINDER, A.S., 1999, *Bancos centrais: teoria e prática*, São Paulo, Editora 34.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 1999, *The Unified Modeling Language User Guide*, Reading, USA, Addison-Wesley.

BORGIDA, A., 1995, "Description logics in data management", *IEEE Transactions on Knowledge and Data Engineering*, v. 7, n. 5 (Oct), pp. 671-682.

BRODIE, M. L. & STONEBRAKER, M., 1995, *Migrating Legacy Systems*, San Francisco, USA, Morgan Kaufmann Publishers.

CARVALHO, L. A. V., 2001, *Datamining*, São Paulo, Érica.

CERI, S. & FRATERNALI, P., 1997, *Designing Database Applications with Objects and Rules*, Reading, USA, Addison-Wesley.

CHEN, P., 1976, "The entity-relationship model - towards a unified view of data", *ACM Transactions on Database Systems*, v.1, n. 1. (Mar), pp. 9-36.

COLLARES, E. S. CABRAL, J. S. LIMA FILHO, L. G. SCHEIDER, V. M., 2001, *Bancos centrais e o Banco Central do Brasil*, Brasília, Banco Central do Brasil e Escola de Administração Fazendária.

DATE, C. J., 2000a, *What not How The Business Rule Approach to Application Development*, Reading, USA, Addison-Wesley.

DATE, C. J., 2000b, *Introdução a Sistemas de Banco de Dados*, 7 ed., Rio de Janeiro, Campus.

DAVENPORT, T. H. & PRUSAK, L., 1998, *Conhecimento Empresarial*, Rio de Janeiro, Campus.

DIAS, D. S., 1985, *O Sistema de Informação e a Empresa*, Rio de Janeiro, LTC.
ELMASRI, R. & NAVATHE, S. B., 1994, *Fundamentals of Database Systems*, 2 ed., Reading, USA, Addison-Wesley.
ELMASRI, R. & NAVATHE, S. B., 2000, *Fundamentals of Database Systems*, 3 ed., Reading, USA, Addison-Wesley.
FAYYAD, U. M. PIATESKY-SHAPIRO, G., PADRAIC, S., 1996, "From Data Mining to Knowledge Discovery An Overview", In: Fayyad, U. M. & Piatesky-Shapiro, G., *Advances in Knowledge Discovery and Data Mining*, Cambridge, USA, AAAI Press.
FORTUNA, E., 2000, *Mercado Financeiro: produtos e serviços*, São Paulo, Qualitymark.
FOWLER, M. & SCOTT, K., 2000, *UML Essencial*, 2 ed., São Paulo, Bookman.
GARCIA-MOLINA, H. ULLMAN, J. D., WIDOM, J., 2001, *Implementação de Sistemas de Bancos de Dados*, Rio de Janeiro, Campus.
GILL, H. S. & RAO, P. C., 1996, *The Official Client/Server Computing Guide to Data Warehousing*, Indianapolis, USA, Que Corporation.
GRAY, J. & REUTER A., 1993, *Transaction Processing: Concepts and Techniques*, San Francisco, USA, Morgan Kaufmann.
GUIDE, *Defining Business Rules ~ What Are They Realy*, 2000, extraído de http://www.businessrulesgroup.org;br01c0.htm em maio de 2000.
HAN, J. & KAMBER, M., 2001, *Data Mining Concepts and Techniques*, San Francisco, USA, Morgan Kaufmann.
IBICT–UDC CONSORTIUM, 1997, *Classificação decimal universal tabelas sistemáticas: edição-padrão internacional em língua portuguesa*, Brasília, publicação IBICT UDC P023.
IBM, 1978, *Business System Planning. Information Systems Planning Guide*, Armonk, New York, USA, Publication No. GE 20-0527-4, IBM.
IBM, 1998, *System Monitor Guide and Reference, cross platform books for DB2*, Armonk, New York, USA, Publication No S10J-8164, db2f0x50.
IMMON W. H., 1992, *Building the Data Warehouse*, New York, USA, John Wiley & Sons.
IMMON W. H., 1997, *Como construir o Data Warehouse*, 2 ed., Rio de Janeiro, Campus.
IMMON W., ZACHMAN J. & GEIGER J., 1997, *Data Stores Data Warehousing and the Zachman Framework*, New York, McGraw-Hill.
INMON, W. H. & HACKATHORN, R. D., 1997, *Como usar o Data Warehouse*, Rio de Janeiro, Infobook.
INMON, W. H., 1996, *Building the Data Warehouse*, 2 ed., New York, USA, John Wiley & Sons.
JACKSON, M. A., 1975, *Principles of Program Design*, New York, Academic Press.
KIMBALL, R, 1996, *The Data Warehouse Toolkit*, New York, USA, John Wiley & Sons.
LEGEY, L. MARCONDES, C. OLINTO G., SULAIMAN, A. SOUZA, S., 2000, "Grupo de Trabalho 5: Conteúdo e Organização da Informação", *Workshop Formação de Recursos Humanos*. In: *Tecnologia da Informação para o Estado do Rio de Janeiro*, Instituto de Matemática Pura e Aplicada, IMPA/FAPERJ/RNP, Rio de Janeiro, 4 a 6 de Setembro.
LEYMANN, F. & ROLLER, D., 2000, *Production Workflow Concepts and Techniques*, New Jersey, USA, Prentice-Hall.
LYNCH, N. MORRIT, M. WOLHL, W. FOKETE, A., 1994, *Atomic Transactions*, San

Francisco, USA, Morgan Kaufmann.
MACMENAMIM, S. & PALMER, J. F., 1984, *Análise Essencial de Sistemas*, São Paulo, Makron.
MARSHALL, C., 2000, *Enterprise Modeling with UML*, Reading, USA, Addison-Wesley.
MARTIN, J. & ODELL, J., 1997, *Object Oriented Methods: A foundation*, 2 ed., Englewood Clifs, New Jersey, USA, Prentice-Hall.
MARTIN, J., 1982, *Strategic Data Planning Methodologies*, New Jersey, USA, Prentice-Hall.
MENDES, S. B. T. , XEXÉO, G. B. , SULAIMAN A., 1997, "Uma Proposta de Modelo de Coordenação para o Sistema de Regras do Projeto HIP2 baseado na Lógica do Fluxo de Informação", *Encontro Nacional de Inteligência Artificial – ENIA97*, Brasília, Jul.
MONTGOMERY, S. L., 1994, *Object-Oriented Information Engineering*, New York, Academic Press.
NELSON, M., 1991, *The Data Compression Book,* New Jersey, USA, Prentice-Hall.
ORACLE, 1998, Oracle: *Database Administration volume 1 Student Guide*, ORACLE 30020GC10.
OZSU, M., VALDURIEZ, P., 1999, *Principles of Distributed Database Systems*, 2 ed., New Jersey, USA, Prentice-Hall.
PASSOS, E. P. L., 1981, *Algumas Idéias e Experimentos sobre Demonstração Automática de Teoremas*, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
PASSOS, E. P. L., 1989, *Inteligência Artificial e Sistemas Especialistas*, Rio de Janeiro, LTC,.
PASSOS. E. P. L., SULAIMAN A., GARCEZ, C. R. E TANAKA, A. K., 1995, "A Conceptual Model for a Knowledge Base Homogeneously Stored in a Database Environment". In: *Lecture Notes in Artificial Intelligence 991, 12th Brazilian Symposium on Artificial Intelligence, SBIA'1995*, pp. 295-301, Campinas, Oct.
PRESSMAN, R., 1992, *Software Engineering A Pratictioner's Approach*, 3 ed., New York, McGraw-Hill.
ROCHA, A. R C., 1987, *Análise e Projeto Estruturado de Sistemas*, Rio de Janeiro, Campus.
ROSS, R., 1997, *The Business Rule Book: Classifying, Defining, and Modeling Rules*, 2 ed., Houston, USA, Business Rule Solution.
ROSS, R., 1998, *Business Rule Concepts*, Houston, USA, Business Rule Solutions.
RUMBAUGH, J. , BLAHA, M. , PREMERLANI, W. , FREDERICK, E. , LORENSEN, W., 1991, *Object-Oriented Modeling and Design*, New Jersey, USA, Prentice-Hall.
SCHNEIDER, G. & WINTERS, J. P., 2001, *Applying Use Cases, a Practical Guide*, 2 ed., Reading, USA, Addison-Wesley.
SILVA, G. Z. ALMEIDA, V. T. SOUZA, J. M. SULAIMAN, A.PEREIRA NETO, F. G., 2001, "ATENAS Um Sistema Gerenciador de Regras de Negócio". In: *Simpósio Brasileiro de Engenharia de Software – SBES '2001*, pp. 338-343, Rio de Janeiro, Out.
SILVERSTON, L. , INMON, W. H. , GRAZIANO, K., 1997, *The Data Model Resource Book*, New York, USA, John Wiley & Sons.
SINGH H. S., 1998, *Data Warehousing*, New Jersey, USA, Prentice-Hall.

SOFTWARE AG, 1997, *ADABAS DBA Reference Manual*, version 6.2, São Paulo, CONSIST, Software AG, ADA621-030IBB.

SOUZA, J. , MATTOSO M. L.Q. XEXÉO, G. B., 1994, *Heterogeneidade, Interoperabilidade e Paralelismo em SGBDOOs com Aplicações em Sistemas de Informações Geográficas*, Projeto Integrado de Pesquisa CNPq.

SOUZA, J. M. SULAIMAN, A. OLIVEIRA, J. CARDOSO, L., 2001, "Gestão de Conhecimento para Experimentos Científicos: Bill of Experiments". In: *4º International Symposium on Knowledge Management/Document Management - ISKM/DM'2001*, pp. 261-275, Curitiba, aug..

SOUZA, J. M., 1986, *Software Tools for Conceptual Schema Integration*. Ph.D. Thesis, School of Information Systems , University of East Anglia, U.K.

SOWA, J. F. & ZACHMAN, J. A., 1992, "Extending and Formalizing the Framework for Information Systems Architecture", *IBM Systems Journal*, v. 31, n. 3, pp. 590-616, IBM Publication G321-5488.

SPRAGUE JR., R. H. & CARLSON, E. D., 1982, *Building Effective Decision Support Systems*, New Jersey, Prentice Hall.

SPRAGUE JR., R. H. & WATSON, H. J., 1991, *Sistema de Apoio à Decisão*, Rio de Janeiro, Campus.

SULAIMAN, A. & SOUZA, J., 1997, "Prospecção de Conhecimento em Banco de Dados", *Developers Magazine*, v. 1, n. 6 (Fev.), pp. 38-39.

SULAIMAN, A. & SOUZA, J. M., 1998a, "Técnicas de Modelagem de Dados em Projetos de Data Warehouse", *Developers Magazine*, v. 2, n. 18 (Fev.), pp. 12-14.

SULAIMAN, A. & SOUZA, J. M., 1998b, "Modelagem de Dados e DW: Tecnologia da Informação", *Developers Magazine*, v. 2, n. 19 (Mar.), pp. 42-44.

SULAIMAN, A. & SOUZA, J. M., 2000, "Panorama da Mineração de Dados no Brasil", *IV Encontro Nacional da Associação Nacional de Pesquisa e Pós-Graduação em Ciência da Informação*, Brasília, Nov.

SULAIMAN, A. & SOUZA , J. M., 1998c, "Representação do Conhecimento em Data Warehouse", *Developers Magazine*, v. 2 n. 20 (Abr.), pp. 42-43.

SULAIMAN, A. & SOUZA, J. M., 1998d, "A Decision Support System that Reverse Engineers Abstract Database Transactions - The Conceptual Model", In: *Data Mining*, v. 1, *International Conference on Data Mining*, WIT Press, pp. 401-411.

SULAIMAN, A. & SOUZA, J. M., 2001, "Mineração de Dados". In: Tarapanoff, K. (ed), *Inteligência Organizacional e Competitiva*, capítulo Brasília, Editora UNB.

SULAIMAN, A. & TANAKA, A. K., 1997a, "Banco de Dados Ativos e o Modelo ER**2", *Developers Magazine*, v. 2, n. 14 (Out.),pp. 14-16.

SULAIMAN, A. & TANAKA, A. K., 1997b, *Fundamentos e Projetos de Bancos de Dados, atelier de Orientação a Objetos*, Rio de Janeiro, Universidade Santa Úrsula.

SULAIMAN, A. & XEXÉO, G. B., 1997, "Um Servidor de Regras de Produção", *Developers Magazine*,v. 1, n. 9 (Mai.), pp. 32-33.

SULAIMAN, A. SILVA, G. Z. ALMEIDA, V. T. SOUZA, J. M., 2001, "Regras de Negócios e Workflow: revisão e experiência". In: *4º International Symposium on Knowledge Management/Document Management - ISKM/DM'2001*, pp. 31-42, Curitiba, Ago.

SULAIMAN, A. SOUZA, J. M. BLASCHEK, J. R. PEREIRA NETO, F., 1999,

Gerenciando Regras de Negócios: Estado Atual de um Projeto, *Developers Magazine*, v. 4, n. 39 (Nov.), pp. 44-45.

SULAIMAN, A. SOUZA, J. M. NETO, F. P. SILBERMAN, C. TELES, M. P. LIMA, D., 2000, "Tecnologia da Informação para a Gestão do Conhecimento: Regras de Negócios", In: *3º International Symposium on Knowledge Management/Document Managemet - ISKM/DM'2000*, pp. 105-124 Curitiba, Nov.

SULAIMAN, A., 1992, *Ambiente Inteligente de Apoio à Decisão para Aplicações Demográficas e Estatísticas*. Tese de M.Sc., IME, Rio de Janeiro, RJ, Brasil.

SULAIMAN, A., 1998, *Mineração de Dados Endógena para Auxílio à Integração de Bases de Dados*. Exame de Qualificação ao D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ. Brasil.

SULAIMAN, A., 1999, "Using Datalog Data Model and Data Mining to solve Bill of Materials personalization problems". In: *International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pp. 161-163, Santa Clara, California, USA, Apr.

SULAIMAN, A., MATTOSO, M .L Q. E SOUZA, J. M., 1995, "Um Shell de Sistema Especialista Implementado em O2, USE O2!". In: *II Workshop de Bancos de Dados não convencionais*, pp. 49-51, Niterói, Dez.

TANAKA, A., 1992, *On Conceptual Design of Active Databases*, PhD Thesis, Georgia Institute of Technology, Georgia, USA.

THOMSEN, E., 1997, *Olap Solutions*, New York, USA, John Wiley & Sons.

ULLMAN, J. D. & WIDOM, J., 1997, *A First Course in Database Systems* New Jersey, USA, Prentice-Hall.

ULLMAN, J. D., 1988, *Principles of Database and Knowledge-Base Systems*, vol. I e II, Rockville, USA, Computer Science Press.

WARMER, J. & KLEPPE, A., 1999, *The Object Constraint Language*, Reading, USA, Addison-Wesley.

WFMC, 1994, *Glossary: A Workflow Management Coalition Specification*, Florida, USA, Workflow Management Coalition, WFMC.

ZACHMAN J. A., 1987, "A Framework for Information Systems Architecture", *IBM Systems Journal*, v. 26, n. 3 pp., 276-292, IBM publication G3212-5298.

Apêndice A – Regras de Negócios obtidas

A.1. Conjunto de regras obtidas

Neste apêndice está ilustrado o conjunto de regras gerado automaticamente no experimento descrito na seção IV.5.3.2.

WIZRULE REPORT GENERAL DETAILS:

File Name: D:Periodos123.txt

Total No. of Records: 7243

All Rules

Minimum Probability of If-then Rules: 0.95

Minimum Accuracy Level of Formula Rules: 0.95

Minimum Number of Cases in a Rule: 40

WizRule Version 3.05

UNCONDITIONAL RULES

1) CRUD is D or U or C

Rule's probability: 1.000

The rule exists in 7242 records.

IF-THEN RULES:

2) If PGM is 11.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 729 records.

Significance Level: Error probability < 0.001

3) If PGM is 12.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 1107 records.

Significance Level: Error probability < 0.001

4) If PGM is 19.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 365 records.

Significance Level: Error probability < 0.01

5) If PGM is 2.00 ... 6.00 (mean = 2.44)

Then

PERIODO is 1.00

Rule's probability: 0.965

The rule exists in 391 records.

Significance Level: Error probability < 0.001

Deviations (records' serial numbers):

5667, 5687, 5689, 5692, 4632, 4634, 4635, 4636, 4637, 4638,

4639, 4640, 4641, 4645

6) If PGM is 9.00

Then

PERIODO is 2.00

Rule's probability: 0.993

The rule exists in 1474 records.

Significance Level: Error probability is almost 0

Deviations (records' serial numbers):

2208, 2209, 5669, 5670, 5672, 5673, 5674, 5681, 5682, 5683

7) If PGM is 13.00

Then

PERIODO is 3.00

Rule's probability: 1.000

The rule exists in 1545 records.

Significance Level: Error probability is almost 0

8) If DATASTR is 1.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 41 records.

Significance Level: Error probability < 0.1

9) If PGM is 17.00

and DATASTR is 26.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 292 records.

Significance Level: Error probability < 0.01

10) If DATASTR is 26.00

and PERIODO is 1.00

Then

CRUD is U

Rule's probability: 1.000

The rule exists in 450 records.

Significance Level: Error probability < 0.01

11) If PGM is 2.00

and CRUD is D

Then

PERIODO is 1.00

Rule's probability: 1.000

The rule exists in 74 records.

Significance Level: Error probability < 0.01

12) If PGM is 2.00 ... 4.00 (mean = 2.21)
and CRUD is U
Then
PERIODO is 1.00
Rule's probability: 0.986
The rule exists in 279 records.
Significance Level: Error probability < 0.001
Deviations (records' serial numbers):
4635, 4636, 4638, 4639

13) If PGM is 17.00
and CRUD is D
Then
PERIODO is 1.00
Rule's probability: 1.000
The rule exists in 496 records.
Significance Level: Error probability < 0.001

14) If DATASTR is 2.00
and CRUD is D
Then
PERIODO is 1.00
Rule's probability: 1.000
The rule exists in 74 records.
Significance Level: Error probability < 0.01

15) If DATASTR is 2.00
and CRUD is U
Then
PERIODO is 1.00
Rule's probability: 1.000
The rule exists in 253 records.
Significance Level: Error probability < 0.001

16) If DATASTR is 26.00
and CRUD is C
Then
PERIODO is 2.00
Rule's probability: 0.994
The rule exists in 804 records.
Significance Level: Error probability is almost 0
Deviations (records' serial numbers):
5669, 5672, 5673, 5681, 5682

17) If DATASTR is 26.00
and CRUD is D
Then
PERIODO is 3.00
Rule's probability: 1.000
The rule exists in 1198 records.
Significance Level: Error probability is almost 0

18) If PGM is 17.00
and DATASTR is 7.00
and PERIODO is 3.00
Then
CRUD is C
Rule's probability: 1.000
The rule exists in 224 records.
Significance Level: Error probability is almost 0