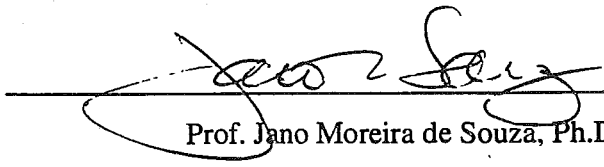
DWFIST: UMA ABORDAGEM BASEADA EM DATA WAREHOUSE PARA EXPLORAÇÃO E ANÁLISE DE CONJUNTOS DE ITENS FREQUENTES

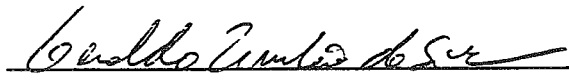
Rodrigo Salvador Monteiro

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.

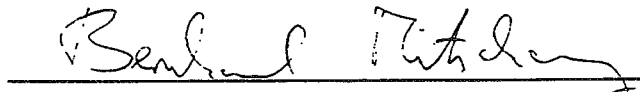
Aprovada por:



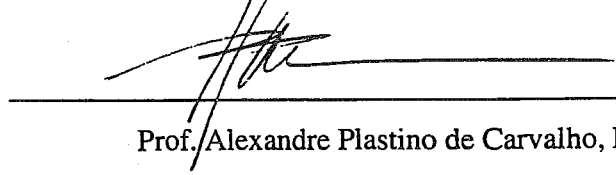
Prof. Jano Moreira de Souza, Ph.D.



Prof. Geraldo Zimbrão da Silva, D.Sc.

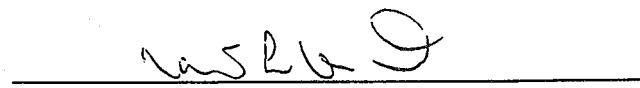


Prof. Bernhard Mitschang, Dr.-Ing.



Prof. Alexandre Plastino de Carvalho, D.Sc.

Profa. Maria Luiza Machado Campos, Ph.D.



Profa. Marta Lima de Queirós Mattoso, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2005

MONTEIRO, RODRIGO SALVADOR

DWFIST: Uma abordagem baseada em Data Warehouse para exploração e análise de conjuntos de itens frequentes [Rio de Janeiro] 2005.

VI, 136 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Data Warehouse
2. Mineração de Dados

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DWFIST: UMA ABORDAGEM BASEADA EM DATA WAREHOUSE PARA EXPLORAÇÃO E ANÁLISE DE CONJUNTOS DE ITENS FREQUENTES

Rodrigo Salvador Monteiro

Agosto/2005

Orientadores: Jano Moreira de Souza

Geraldo Zimbrão da Silva

Programa: Engenharia de Sistemas e Computação

Esta tese propõe a abordagem DWFIST, que se preocupa em dar suporte à análise e exploração de conjuntos de itens frequentes e padrões derivados, e. g. regras de associação, em bases de dados transacionais. A meta desta nova abordagem pode ser resumida com a seguinte dupla contribuição: oferecer (1) funcionalidades flexíveis para recuperação de padrões sem requerer os dados originais durante a fase de análise e (2) uma modelagem padrão para data warehouses de conjuntos de itens frequentes facilitando o desenvolvimento e reutilização de ferramentas para análise e exploração de padrões baseados em conjuntos de itens. Um data warehouse que armazena conjuntos de itens frequentes válidos em diferentes partições das transações originais desempenha um papel central na nossa abordagem. Depois de discutir as tarefas de pré-processamento efetuadas na staging area, nós apresentamos esquemas padrões nos níveis conceitual e lógico visando uma modelagem padrão. Propriedades desta modelagem padrão permitem uma combinação flexível de qualquer conjunto de partições. Os conjuntos de itens frequentes válidos em qualquer conjunto de partições podem ser recuperados junto com limites superiores e inferiores em suas contagens de frequência. Questões de completude e precisão relacionadas a um conjunto de itens frequentes recuperado do data warehouse também são discutidas.

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

DWFIST: THE DATA WAREHOUSE OF FREQUENT ITEMSETS TACTICS APPROACH

Rodrigo Salvador Monteiro

August/2005

Advisors: Jano Moreira de Souza

Geraldo Zimbrão da Silva

Department: Systems and Computing Engineering

This thesis proposes the DWFIST approach, which is concerned with supporting the analysis and exploration of frequent itemsets and derived patterns, e.g. association rules, in transactional datasets. The goal of this new approach can be summarized as the following twofold contribution: provide (1) flexible pattern-retrieval capabilities without requiring the original data during the analysis phase, and (2) a standard modeling for data warehouses of frequent itemsets allowing an easier development and reuse of tools for analysis and exploration of itemset-based patterns. A data warehouse storing frequent itemsets holding on different partitions of the original transactions plays a central role in our approach. After discussing pre-processing tasks performed in the staging area, we present standard conceptual and logical schemas aiming at a standard modeling. Properties of this standard modeling allow for a flexible combination of any set of partitions. The frequent itemsets holding on any set of partitions can be retrieved along with upper and lower bounds on their frequency counts. Completeness and precision issues related to the retrieved set of frequent itemsets are discussed as well.

Índice

1 Introdução

Avanços em mecanismos de coleta de dados, o uso de código de barras na maior parte dos produtos comerciais, e informação sobre muitas transações governamentais e de negócio têm nos inundado com dados, criando uma necessidade urgente por novas técnicas e ferramentas para suportar inteligentemente e automaticamente a transformação destes dados em conhecimento útil (Fayyad et al., 1998). A área de pesquisa de mineração de dados forneceu muitas técnicas diferentes para explorar os dados e revelar diferentes tipos de comportamento padrão ou não-padrão (Han and Kamber, 2001). Recentemente, algumas destas técnicas foram adaptadas para atender aos estritos requisitos de tempo impostos por streams de dados.

Minerar padrões frequentes em dados transacionais, em particular o domínio dos padrões de conjuntos de itens, é uma técnica que merece atenção especial, devido à sua grande aplicabilidade (Boulicaut, 2004). A mais reconhecida é dar suporte à mineração de regras de associação, que foi apresentada por Agrawal et al. (1993). Desde então, muitos trabalhos propondo algoritmos para mineração de regras de associação afirmam a importância e atenção dedicada a esta tarefa de mineração de dados. Manku e Motwani (2002) e Giannella et al. (2003) apresentam algoritmos para minerar regras de associação em streams de dados. Computação de iceberg cube (Beyer e Ramakrishnan, 1999), classificação associativa (Liu et al., 1998), clusterização baseada em padrões frequentes (Wang et al., 2002) e mineração de regras generalizadas (Mannila e Toivonen, 1996) são outros exemplos onde os conjuntos de itens frequentes são úteis.

Esta tese propõe a abordagem DWFIST, que visa dar suporte à análise e exploração de conjuntos de itens frequentes e padrões derivados, e.g. regras de associação, em bases de dados transacionais. O acrônimo DWFIST significa *Data Warehouse of Frequent Itemsets Tactics*. Em outras palavras, nossa abordagem organiza conjuntos de itens que ocorrem frequentemente juntos em transações aplicando uma modelagem dimensional e almejando o suporte à análise e exploração de padrões.

A seguir apresentamos a motivação do nosso trabalho e estabelecemos o escopo e objetivos da tese. As contribuições são apresentadas na sequência e a estrutura da tese é fornecida. Uma versão estendida deste texto em inglês está disponível em Monteiro (2005c).

1.1 Motivação

Algumas tarefas de mineração de dados podem produzir uma quantidade de dados tão grande que um novo problema de gestão de conhecimento surge (Klemettinen et al., 1994). A mineração de conjuntos de itens frequentes é conhecida por se enquadrar nesta categoria há muito tempo. A análise dos resultados da tarefa de mineração de conjuntos de itens frequentes está muito longe de ser trivial. O mesmo é verdade para muitos padrões construídos sobre conjuntos de itens frequentes, como as regras de associação. O analista pode facilmente se confrontar com um grande número de padrões durante este tipo de análise. Identificar os padrões que são realmente interessantes é uma tarefa difícil. Ferramentas analíticas e de exploração especializadas devem ser projetadas de forma a auxiliar o analista neste tipo de tarefa (Monteiro et al., 2003). A falta de um procedimento padrão para organizar, armazenar e acessar conjuntos de itens frequentes faz com que o esforço para desenvolver este tipo de ferramentas seja muito alto uma vez que impede a reutilização de soluções genéricas para ambientes distintos.

Aplicações recentes, como a análise de tráfego de redes, web click stream mining, medição de consumo de energia, análise de dados de redes de sensores, e monitoramento dinâmico da flutuação de bolsa de valores, são alguns exemplos onde um novo tipo de dado surge, o chamado stream de dados. Um stream de dados é contínuo e potencialmente infinito. É desafiador minerar padrões frequentes em streams de dados porque esta tarefa é essencialmente um conjunto de operações de junção onde o operador de junção é tipicamente um operador de bloqueio, i.e., a computação para qualquer conjunto de itens não pode completar antes de olhar o conjunto de dados como um todo (Giannella et al., 2003). Oferecer flexibilidade para minerar conjuntos de itens frequentes em algum subconjunto de um stream de dados é ainda mais desafiador, especialmente quando o subconjunto que será analisado não é conhecido a priori.

1.2 Definição do Problema

Esta tese provê uma modelagem padrão para conjuntos de itens frequentes válidos em bases de dados transacionais assim como meios padrões para acesso e recuperação. Estas características visam preencher a lacuna existente para a definição de ferramentas de análise e exploração de tais padrões que possam ser facilmente reutilizadas em contextos diferentes. Mostramos também que a abordagem proposta

pode ser usada com fontes de dados contínuos (stream data sources) tornando possível novos tipos de análises temporais inviáveis até então.

Nós definimos formalmente os problemas que estamos tratando, de forma a delinear o escopo da tese, como:

- **Modelagem padrão para data warehouses de conjuntos de itens frequentes:** Definir um padrão de modelagem de data warehouse aplicável a qualquer base de dados transacional e adequado para tarefas de análise que se baseiem em conjuntos de itens frequentes.

- **Mineração de conjuntos de itens frequentes baseada em calendário sobre streams de dados:** Seja *D* uma base de dados transacional que é fornecida como um stream de dados por uma fonte de dados qualquer. Portanto, somente um subconjunto das transações em *D* está disponível em qualquer ponto no tempo. Entretanto, queremos minerar conjuntos de itens frequentes em *D*, considerando um conjunto de restrições baseadas em calendário definida ad-hoc. ■

Alguns exemplos de restrições baseadas em calendário são: *weekday* in {Monday, Friday}; *day_period* = "Morning"; *holiday* = "yes"; etc.

1.3 Contribuições

As contribuições desta tese estão compiladas nesta seção, apresentando um resumo das nossas soluções para os problemas mencionados na seção anterior.

A área de pesquisa de data warehouse tem sido extremamente bem sucedida em oferecer formas eficientes e efetivas de armazenar e organizar grandes quantidades de dados. É bem sucedida também em oferecer um padrão de modelagem sobre o qual ferramentas analíticas reutilizáveis podem ser projetadas e implementadas. Esta tese apresenta uma modelagem padrão para organizar conjuntos de itens frequentes aplicável a qualquer base de dados transacional. O produto desta modelagem padrão é um data warehouse de conjuntos de itens frequentes, o qual é o componente principal da abordagem DWFIST. Este organiza o conjunto completo de transações em partições disjuntas e armazena informações sobre conjuntos de itens frequentes válidos em cada partição. As partições podem ser combinadas para obtenção de conjuntos de itens

frequentes, com contagens de frequência aproximada, válidos em qualquer conjunto de partições. Além disso, este trabalho oferece as seguintes contribuições:

- Funcionalidades de recuperação flexível de padrões sem necessitar dos dados originais durante a fase de análise;
- Uma modelagem padrão para data warehouses de conjuntos de itens frequentes permitindo um desenvolvimento mais fácil e reutilização de ferramentas para análise e exploração de padrões baseados em conjuntos de itens;
- Uma visão conceitual (modelo dimensional) para análise de padrões que é familiar a profissionais de negócios e adequada para a análise de grandes volumes de dados;
- Definição de um processo ETL (Extração, Transformação e Carga) para um data warehouse de conjuntos de itens frequentes;
- Um conjunto de propriedades do data warehouse de conjuntos de itens frequentes, incluindo a completude do conjunto de padrões recuperados e questões de precisão;
- Garantia de precisão para medidas de padrões derivados do data warehouse de conjuntos de itens frequentes;
- Apresentar como DWFIST pode ser aplicada para alavancar a mineração de padrõs baseados em calendário em um cenário de stream de dados;

Nossa abordagem não propõe um novo algoritmo para mineração de conjuntos de itens frequentes diretamente em streams de dados. Ao invés disso, nós utilizamos algoritmos existentes para executar uma mineração comum de conjuntos de itens frequentes em pequenos lotes de transações de um stream de dados, e oferecemos meios para combinar de forma flexível os conjuntos de itens frequentes de lotes diferentes. Ao invés de propor novas estruturas de dados, a abordagem DWFIST é uma abordagem centrada em banco de dados e, portanto, se beneficia de estruturas de índice, otimização de consulta, facilidades de gerência de armazenamento e assim por diante. Estas são características vitais uma vez que o data warehouse de conjuntos de itens frequentes compreende potencialmente um grande, porém gerenciável, volume de dados. Além

disso, nossa abordagem pode ser implementada usando bancos de dados comerciais convencionais.

Nenhuma abordagem anterior é capaz de derivar padrões baseados em calendário em streams de dados.

1.4 Estrutura da Tese

O restante desta tese está organizado da seguinte forma. O capítulo 2 apresenta algumas definições básicas necessárias para o entendimento deste trabalho. Uma visão geral da abordagem DWFIST é apresentada no capítulo 3. O componente principal, o data warehouse de conjuntos de itens frequentes (Data Warehouse of Frequent Itemsets), é detalhado no capítulo 4. O capítulo5 discute a recuperação de padrões do data warehouse. O capítulo 6 é dedicado à avaliação experimental. A conclusão e alguns tópicos para pesquisas futuras são apresentados no capítulo 7.

2 Definições Básicas

Este capítulo apresenta as definições essenciais ao entendimento do trabalho. Informações mais detalhadas sobre conjuntos de itens frequentes e data warehouse estão disponíveis em Monteiro (2005c). Os trabalhos relacionados a nossa abordagem também são apresentados e discutidos em Monteiro (2005c).

Minerar padrões frequentes em dados transacionais, em particular o domínio de conjuntos de itens, é uma técnica que merece atenção especial devido a sua grande aplicabilidade (Boulicaut, 2004). A terminologia relacionada a conjuntos de itens frequentes usada ao longo da tese é introduzida a seguir.

Definição 1 (bases de dados transacionais): Seja *itens* um conjunto finito de símbolos denotados por letras maiúsculas, e.g., *itens* = {A, B, C, …}. Uma transação t é um subconjunto de *itens*. Uma base de dados transacionais D é um conjunto $D = \{t_1, t_2, \ldots, t_n\}$ de transações. ■

Dados de cesta de mercado (transações são conjuntos de produtos que são comprados por clientes) é um exemplo clássico de base de dados transacionais. Dados textuais (transações são conjuntos de palavras chave que caracterizam documentos) corresponde a um outro exemplo (Boulicaut, 2004).

Definição 2 (conjuntos de itens): um *conjunto de itens* é um subconjunto de *itens*. ■

Definição 3 (frequência de conjunto de itens): Uma transação t suporta um *conjunto de itens* I se todo *item* em I pertence a t. A *frequência* de I em um conjunto de transações T, onde $T \subseteq D$, é o número de transações em T que suportam I e é denotada por $F_T(I)$. ■

Definição 4 (suporte de conjunto de itens): Dado $|T|$ o número de transações em T, onde $T \subseteq D$, o *suporte* de um *conjunto de itens I* em T é a *frequência* de I dividida por $|T|$ e é denotada por $S_T(I)$:

$$S_T(I) = \frac{F_T(I)}{|T|} \quad (1)$$

o suporte de um conjunto de itens corresponde a uma taxa denotando a porcentagem de transações em T suportando I. ■

Definição 5 (conjuntos de itens frequentes): Dado um suporte mínimo σ, um *conjunto de itens I* é considerado frequente em um conjunto de transações T se $S_T(I) \geq \sigma$. Tal *conjunto de itens I* é um conjunto de itens σ-frequente em T. ■

Definição 6 (conjuntos de itens frequentes baseados em calendário): Sejam X um conjunto de restrições baseadas em calendário, Φ o conjunto de transações que satisfazem X, onde $\Phi \subseteq D$, e σ um suporte mínimo. O conjunto de *conjuntos de itens frequentes baseados em calendário* é definido por todo *conjunto de itens I* com $S_\Phi(I) \geq \sigma$. ■

Alguns exemplos de restrições baseadas em calendário são: *weekday* in {Monday, Friday}; *day_period* = "Morning"; *holiday* = "yes"; *first working day of the month* = "yes"; *season* in {summer, spring}; *year* = "2003"; etc.

3 A Abordagem DWFIST

O acrônimo DWFIST significa "*Data Warehouse of Frequent ItemSets Tactics*" (Monteiro et al., 2005b). O dicionário e thesaurus Oxford define "*Tactics*" como: dispositivo habilidoso; esquema, estratégia. Neste sentido, a abordagem DWFIST visa fornecer suporte a análise e exploração de conjuntos de itens frequentes e padrões derivados, e.g. regras de associação, em bases de dados transacionais.

O campo de pesquisa de data warehouse tem sido extremamente bem sucedido em oferecer formas eficientes e efetivas de analisar uma grande quantidade de dados. Apesar das técnicas de data warehouse não poderem ser diretamente aplicadas para armazenar e analisar padrões de conjuntos de itens frequentes, elas podem ser adaptadas procurando manter seus princípios e melhores práticas aprendidas ao longo dos anos. O armazenamento de conjuntos de itens frequentes pode soar estranho a princípio já que se poderia dizer que seria melhor armazenar os dados originais e minerar os padrões frequentes à medida que fossem solicitados pelo analista. Neste ponto, nós gostaríamos de chamar a atenção do leitor para fazer algumas observações relevantes. Primeiro, não é sempre possível armazenar os dados originais. É inviável armazenar todo um stream de dados e ao mesmo tempo ainda é extremamente interessante analisar diferentes tipos de padrões válidos em streams, como por exemplo, padrões baseados em calendário. Segundo, através do pré-processamento dos conjuntos de itens frequentes, o esforço computacional necessário durante as tarefas de análise feitas pelo usuário é reduzido melhorando a interatividade. Finalmente, é também possível armazenar conjuntos de itens frequentes junto com os dados originais. A informação redundante pode ser justificada na medida que os dados sejam publicados de forma a facilitar a análise (Kimball e Ross, 2002).

Este capítulo apresenta os principais requisitos para a abordagem DWFIST na seção 3.1. Uma visão geral dos componentes da nossa abordagem é apresentada na seção 3.2. A seção 3.3 apresenta um exemplo que servirá de base para ilustrar nossas idéias ao longo da tese.

3.1 Requisitos

A informação armazenada no data warehouse de conjuntos de itens frequentes deve ser suficiente para responder consultas que requisitem padrões válidos em algum subconjunto das transações originais. Em outras palavras, deve ser possível especificar restrições em características das transações originais ao recuperar padrões. Para cada conjunto de itens recuperado do data warehouse, deve ser possível recuperar sua contagem de frequência no subconjunto de transações originais sendo consultado. Sempre que a frequência exata não for conhecida, deve ser possível oferecer uma resposta aproximada. Além disso, deve ser possível garantir que o resultado da consulta seja completo, i.e., garantir que não exista um padrão válido no subconjunto das transações originais consultado que não seja recuperado do data warehouse. A Figura 1 esboça a recuperação de padrões a partir do data warehouse de conjuntos de itens frequentes.

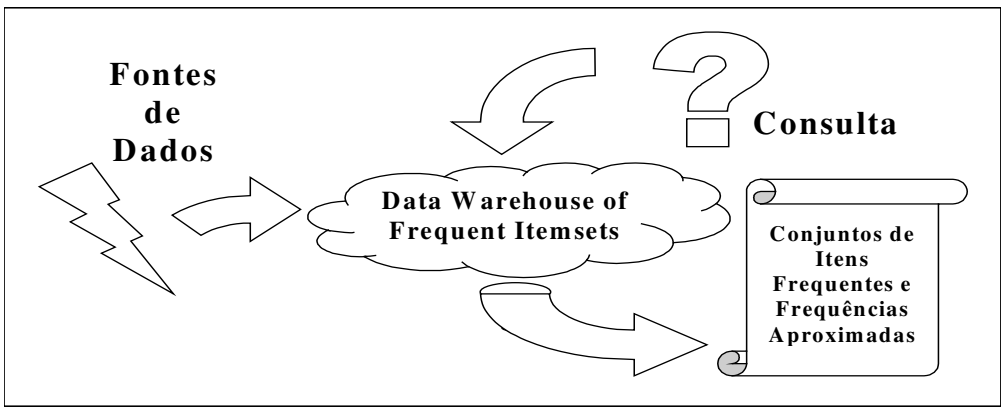


Figura 1 – Esboço da recuperação de padrões do data warehouse de conjuntos de itens frequentes

Também deve ser possível usar os conjuntos de itens frequentes e suas contagens de frequência aproximadas para derivar outros padrões (e.g. regras de associação) e medidas de interesse associadas a estes padrões (e.g. confiança de regras de associação).

O esquema lógico do data warehouse de conjuntos de itens frequentes deve ser definido para oferecer uma forma padrão de acesso e recuperação de conjuntos de itens frequentes, tornando possível desenvolver ferramentas que possam ser reutilizadas em diferentes cenários.

Um outro requisito importante se refere à viabilidade de atender a restrições de tempo de processamento impostas por um stream de dados. Queremos armazenar em um data warehouse os conjuntos de itens frequentes válidos em diferentes conjuntos disjuntos de transações (partições). Desde que consigamos processar os conjuntos de itens frequentes válidos em cada partição antes que a seguinte esteja pronta para ser processada, estamos atendendo aos requisitos impostos por um stream de dados. Por outro lado, temos o processo de carga do data warehouse. A periodicidade da carga do data warehouse deve ser completamente independente da granularidade temporal e deve ser definida para atender requisitos de análise e disponibilidade.

3.2 Componentes

Os principais componentes da abordagem DWFIST e seus relacionamentos são apresentados na Figura 2. O componente "*Pre-processing and Loading step*" compreende três tarefas principais: agrupar as transações em conjuntos disjuntos (partições) de acordo com um critério pré-definido (grão do data warehouse); minerar os conjuntos de itens frequentes válidos em uma partição usando um suporte mínimo de mineração pré-definido; e carregar os conjuntos de itens frequentes minerados no data warehouse.

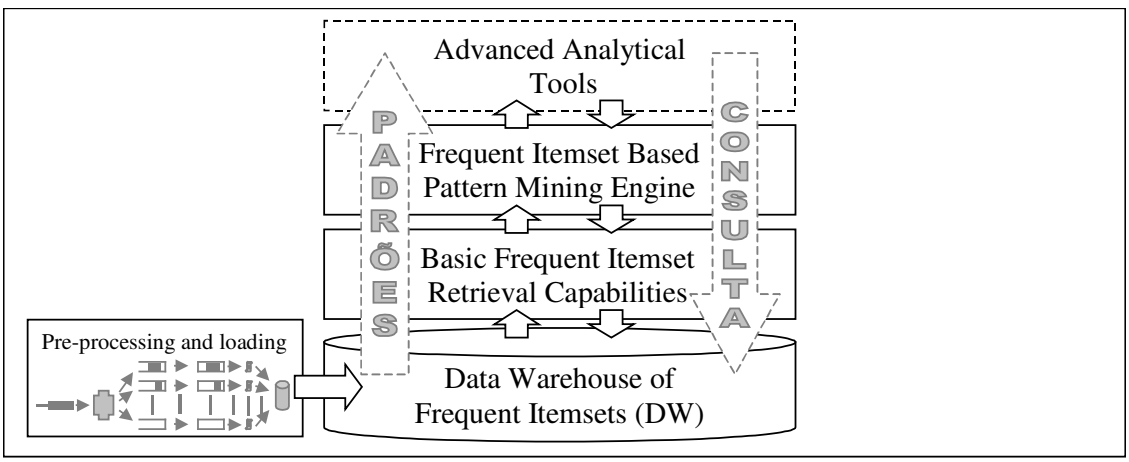


Figura 2 – Componentes da abordagem DWFIST

O componente "*Data Warehouse of Frequent Itemsets*" corresponde ao data warehouse de conjuntos de itens frequentes, referenciado simplesmente como DW, é o principal componente da abordagem. Sua tarefa é armazenar e organizar os conjuntos de

itens frequentes em partições. Uma modelagem padrão fornece uma visão lógica padronizada.

O papel do componente "*Basic Frequent Itemset Retrieval Capabilities*" é recuperar conjuntos de itens frequentes com frequências aproximadas dada uma consulta especificando restrições sobre as partições do data warehouse a serem consideradas. Um exemplo deste tipo de consulta é: "Recuperar os conjuntos de itens frequentes considerando somente o período da manhã".

O componente "*Frequent Itemset Based Pattern Mining Engine*" gera padrões que podem ser obtidos a partir de conjuntos de itens frequentes. Este componente recebe conjuntos de itens frequentes com freqüências aproximadas como entrada, os quais são fornecidos pelo componente "Basic Frequent Itemset Retrieval Capabilities". Regras de associação (Agrawal et al., 1993) e regras Booleanas (Mannila e Toivonen, 1996) são dois exemplos de padrões que podem ser derivados por este componente. Obviamente, os próprios conjuntos de itens frequentes são padrões que também podem ser "derivados". Este componente é também responsável por computar algumas medidas de interesse aproximadas relacionadas a padrões derivados, e.g., garantia de confiança aproximada para regras de associação.

O componente "*Advanced Analytical Tools*" compreende ferramentas de análise e exploração que podem ser construídas usando os outros componentes. É possível construir, por exemplo, uma ferramenta que ofereça uma visão de um cubo de padrões, similar ao que uma ferramenta OLAP oferece para dados dimensionais convencionais. Métodos de exploração automática existentes, como os apresentados em Sathe e Sarawagi (2001), podem ser adaptados para explorar os padrões, assim como novos métodos também podem ser desenvolvidos.

Os capítulos 4 e 5 detalham o núcleo da abordagem DWFIST, desde o componente "Pre-processing and Loading step" até o "Frequent Itemset Based Pattern Mining Engine". O componente "Advanced Analytical Tools" corresponde a qualquer ferramenta que possa ser construída usando as funcionalidades oferecidas pelos outros módulos. Especificar estas ferramentas não é um objetivo desta tese. Uma breve motivação e algumas idéias referente a esse componente são discutidas em Monteiro (2005c).

Para melhor ilustrar e explicar os conceitos a serem discutidos, apresentamos a seguir um exemplo que será usado ao longo do restante da tese.

3.3 Exemplo de Cenário

Nós usamos o clássico exemplo de dados de cesta de mercado para ilustrar os conceitos da nossa abordagem. Nos dados de cesta de mercado, as transações são conjuntos de produtos que são comprados pelos clientes. Como estamos interessados em discutir questões relacionadas a streams de dados, consideraremos que as transações dos clientes ocorrem a uma taxa média de 100 transações por segundo. Este exemplo foi simulado em nosso protótipo e detalhes desta simulação são oferecidos no capítulo 6.

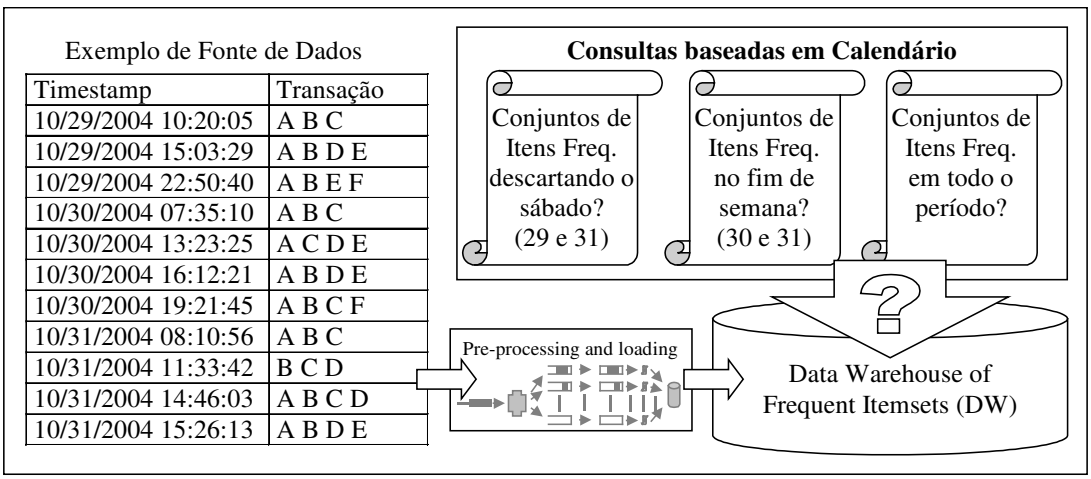


Figura 3 – Exemplo de fonte de dados e consultas baseadas em calendário

Dado o cenário acima, queremos recuperar os padrões baseados em calendário válidos neste stream de dados. Em nosso exemplo, a unidade mínima de tempo que poderá ser analisada corresponde ao período de uma hora. Isto corresponde ao grão do data warehouse que será discutido em detalhes no próximo capítulo. Devido a propriedades do DW, qualquer conjunto de períodos de uma hora pode ser livremente combinado para compor uma partição de calendário. Podemos estar interessados, por exemplo, em recuperar regras de associação válidas no período da manhã em dias de semana. Neste caso, os períodos de uma hora, especificamente [08:00AM, 09:00AM), [09:00AM, 10:00AM), [10:00AM, 11:00AM) e [11:00AM, 12:00PM), pertencentes aos dias de semana serão combinados.

Um exemplo de fonte de dados simplificada é mostrado na Figura 3. A mesma também apresenta outros exemplos de consultas baseadas em calendário e esboça a recuperação de conjuntos de itens frequentes baseados em calendário na nossa bordagem. O data warehouse de conjuntos de itens frequentes recebe e armazena informação de uma fonte de dados transacionais. As informações armazenadas podem ser usadas para fornecer respostas aproximadas para consultas baseadas em calendário. As consultas apresentadas na Figura 3 somente têm como objetivo ilustrar a recuperação de padrões baseados em calendário. Qualquer partição de calendário que possa ser construída combinando períodos de uma hora (e.g. tardes, dias de semana, feriados, primeiro trimestre de 2004, segundo semestre, etc.) representa uma consulta baseada em calendário que pode ser respondida pelo nosso exemplo.

4 Data Warehouse de Conjuntos de Itens Frequentes

O data warehouse de conjuntos de itens frequentes (Data Warehouse of Frequent Itemsets) (Monteiro et al., 2005a), apesar de apresentar particularidades, pode ser visto como um data warehouse comum. Por isso, muitas considerações apresentadas em Kimball and Ross (2002), desde o uso de chaves surrogadas até o "Business Dimensional Lifecycle", são aplicáveis. Nós manteremos nosso foco nas características particulares. Este capítulo começa explicando como os dados transacionais são pré-processado na staging area. Na sequência, esquemas lógicos e conceituais padronizados são apresentados para o DW. Um exemplo de data warehouse de conjuntos de itens frequentes para o nosso cenário de exemplo é fornecido. Por último, as propriedades básicas de um data warehouse de conjuntos de itens frequentes são introduzidas.

4.1 Staging Area

Uma visão geral da staging area é apresentada na Figura 4. Ela apresenta três tarefas principais: separar e acumular transações em diferentes partições; minerar os conjuntos de itens frequentes válidos em cada partição; e carga dos conjuntos de itens frequentes no data warehouse.

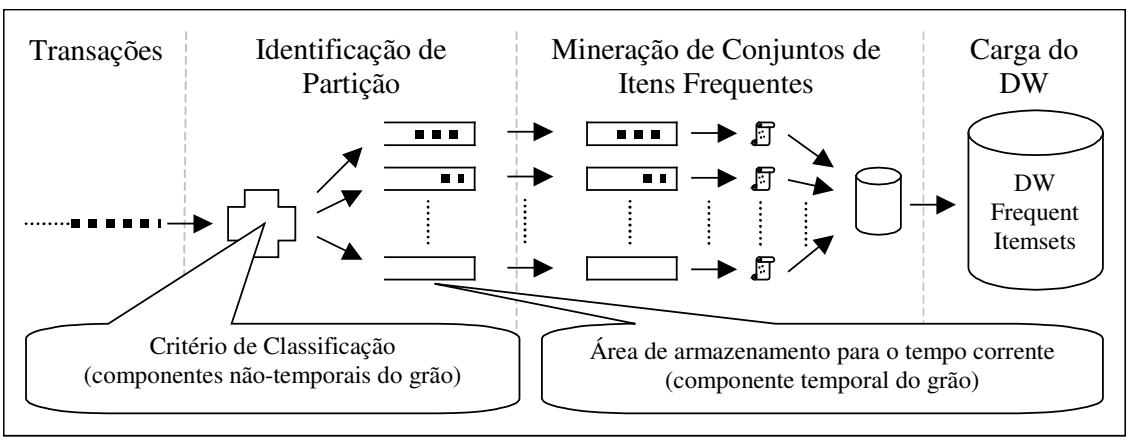


Figura 4 – Visão geral da Staging Area

Primeiramente, uma importante decisão de projeto precisa ser tomada: a definição do grão do DW. O grão do DW deve definir uma partição na fonte de dados transacionais, separando-a em conjuntos disjuntos. Uma regra bem conhecida para data warehouses convencionais diz para armazenar o dado no nível mais granular, em outras

palavras, diz para definir o grão no nível mais detalhado possível. Entretanto, esta regra deve ser quebrada algumas vezes em nossa abordagem. Isto se deve a um importante fator que deve ser levado em conta ao se definir o grão. Um grão mais detalhado oferece mais flexibilidade para análise. Entretanto, quanto mais detalhado for o grão do DW, mais contagens de frequência precisam ser armazenadas, aumentando os requisitos de espaço de armazenamento. Portanto, requisitos de análise devem ser considerados para definir o grão e testes de requisitos de armazenamento têm que ser efetuados para checar a viabilidade da escolha. Em nosso cenário de exemplo, definimos o grão como os conjuntos de itens frequentes por hora. Alguns exemplos de partições definidas por este grão são: "10/29/2005 [08:00AM, 09:00AM)", "10/29/2005 [09:00AM, 10:00AM)", "10/29/2005 [10:00AM, 11:00AM)", e assim por diante. A definição deste grão impede que se analise o comportamento padrão em períodos mais curtos do que uma hora. Por outro lado, todas as combinações possíveis de partições de uma hora podem ser analisadas, oferecendo flexibilidade suficiente para análises importantes baseadas em calendário. Para fontes de dados contínuas (streams de dados), o grão escolhido também deve ser verificado contra restrições de tempo de processamento impostas. Para maiores detalhes quanto à viabilidade de atender a restrições de tempo de processamento e requisitos de espaço de armazenamento em streams de dados veja Monteiro (2005c).

Assim como em qualquer data warehouse, a informação temporal exerce um papel importante no data warehouse de conjuntos de itens frequentes. O grão do DW deve possuir um componente temporal. Até mesmo em aplicações onde a característica temporal não é importante para análise um componente temporal deve fazer parte da definição do grão para, no mínimo, separar diferentes cargas do DW. Outros componentes, como componentes espaciais, podem ser usados adicionalmente para definir o grão. Como um exemplo, o grão pode ser definido como os conjuntos de itens frequentes por hora por loja.

Uma vez definido o grão do DW, as três tarefas apresentadas na Figura 4 podem ser implementadas. A primeira é separar e acumular transações em partições diferentes. Quando o grão possui componentes não-temporais, assim como loja, uma nova transação recebida tem que ser analisada para definir a partição a que pertence. Cada partição para o lote de tempo corrente deve ter uma área de armazenamento correspondente para acumular suas transações. Por exemplo, o grão "por hora por loja"

requer uma área de armazenamento para cada loja. Se o grão é definido somente por um componente temporal, como no nosso cenário de exemplo, então uma área de armazenamento será suficiente.

O componente temporal do grão é usado para identificar quando os conjuntos de transações pertencentes às partições do lote de tempo corrente foram completamente coletados. Os conjuntos de transações completos são passados ao próximo passo onde a mineração acontece.

A mineração de conjuntos de itens frequentes é efetuada em cada conjunto de transações completamente coletado compreendendo uma partição. Um suporte mínimo deve ser especificado para esta tarefa, o qual chamamos de *mining minimum support* ou suporte mínimo de mineração. O limite inicial definido pelo suporte mínimo de mineração não é um compromisso definitivo. O suporte mínimo de mineração pode ser modificado livremente ao longo do tempo assim como limites customizados podem ser especificados para partições diferentes. É necessário somente que cada partição esteja associada à exatamente um suporte mínimo de mineração. A mineração de conjuntos de itens frequentes resulta numa lista de conjuntos de itens frequentes com suas contagens de frequência correspondentes válidos nas transações de uma partição. O suporte mínimo de mineração é definido como 0.1% no nosso cenário de exemplo.

Os conjuntos de itens frequentes, suas contagens de frequência, o número de transações na partição e o suporte mínimo de mineração devem ser armazenados em alguma área intermediária antes de serem finalmente carregados no DW. Esta área de armazenamento intermediária oferece isolamento entre o processo de mineração e o processo de carga do DW. Por causa deste isolamento, é possível, por exemplo, definir um procedimento de carga diário do DW tendo um grão horário. Assim que uma partição for minerada e sua informação armazenada na área intermediária, as transações da partição podem ser descartadas.

Finalmente, a informação armazenada na área intermediária será carregada no data warehouse periodicamente. Muitas questões de um data warehouse convencional surgem neste passo e devem ser tratadas do mesmo modo que nos data warehouses comuns. Alguns exemplos destas questões são a atribuição de chaves surrogadas e a criação de novas instâncias de dimensão. O isolamento oferecido pela área de armazenamento intermediária torna mais fácil cuidar destas questões e conseguir atender a restrições de tempo de processamento impostas por um stream de dados.

4.2 Projeto Conceitual

Apresentamos nesta seção um esquema conceitual padronizado para data warehouses de conjuntos de itens frequentes. Figura 5 e Figura 6 mostram esse esquema usando a notação StarER (Tryfona et al., 1999). Nosso esquema conceitual baseia-se na distinção entre dimensões de item e dimensões de partição:

- Uma dimensão de item descreve itens semelhantes que podem fazer parte de um conjunto de itens frequentes. Caso existam grupos de itens com características distintas, dimensões de itens diferentes devem ser usadas de forma a melhor descrever e classificar os itens. No contexto de procedimentos médicos, os materiais hospitalares poderiam ser representados em uma dimensão de item e a equipe médica em outra.
- Uma dimensão de partição organiza o espaço de transações em conjuntos disjuntos, os quais são chamados de partições do DW. Cada dimensão de partição descreve um componente do grão do DW. Dimensões temporais e espaciais são alguns exemplos de dimensões de partição candidatas.

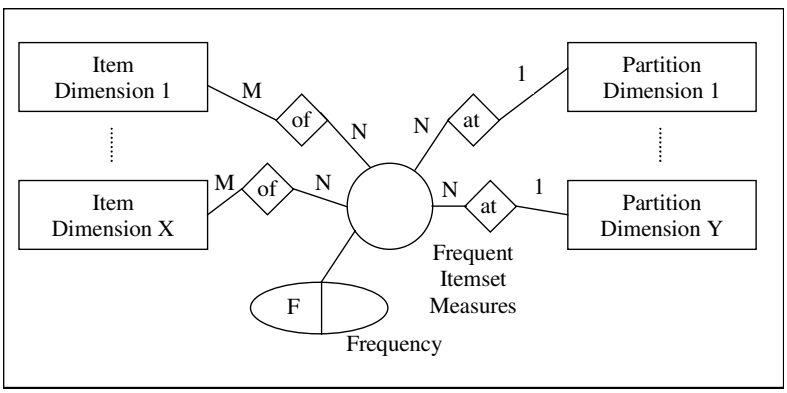


Figura 5 – Esquema conceitual padronizado para medidas de conjuntos de itens frequentes

Outra distinção que pode ser observada na Figura 5 é que a cardinalidade do relacionamento entre uma dimensão de item e os fatos é N para M, enquanto que a do relacionamento entre uma dimensão de partição e os fatos é 1 para N. O relacionamento N para M ocorre porque um conjunto de itens pode conter mais de um item da mesma dimensão de item.

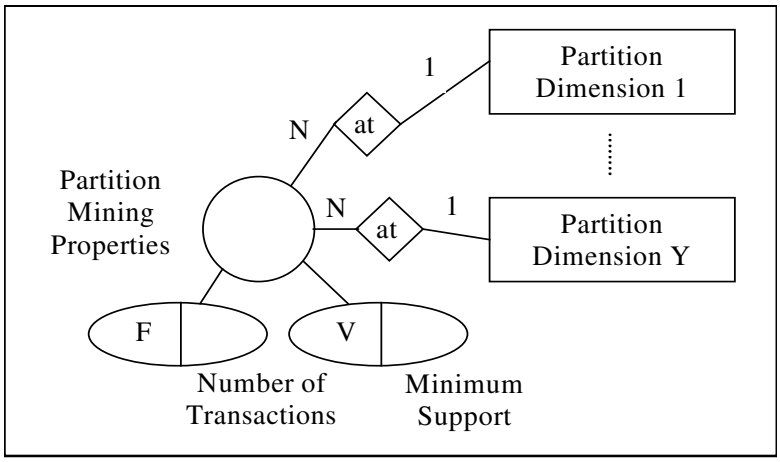


Figura 6 – Esquema conceitual padronizado para propriedades de mineração das partições

Ao acessar o DW, as dimensões de itens são usadas para restringir características dos conjuntos de itens que devem ser recuperados. No cenário de exemplo, poderia ser requisitada a recuperação de conjuntos de itens que contenham pelo menos um produto que pertença à categoria “bebidas”. As dimensões de partição são usadas para definir a porção das transações originais que devem ser consideradas. Recuperar os conjuntos de itens frequentes considerando somente as transações do período da manhã seria uma possibilidade no cenário de exemplo.

É interessante relacionar as informações armazenadas na área intermediária com o nosso esquema conceitual. Figura 5 representa a parte do DW que descreve os conjuntos de itens frequentes e respectivas contagens de frequência. Uma contagem de frequência é um atributo do fato e um conjunto de itens frequentes é representado pelos relacionamentos com as dimensões de itens. Conforme mostrado na Figura 6, para cada partições informações adicionais devem ser armazenadas, i.e. o número de transações e o suporte mínimo usada na mineração. Na figura estas informações são representadas como atributos do fato. Note que o esquema apresentado na Figura 6 somente possui dimensões de partição porque as propriedade de mineração são as mesmas para uma partição, em outras palavras, não estão relacionadas a um conjunto de itens frequentes específico.

As dimensões devem estar em conformidade com a arquitetura de barramento (Kimball and Ross, 2002). Esta conformidade visa permitir a realização de operações de

cross over para relacionar os atributos dos dois fatos apresentados na Figura 5 e na Figura 6 através das dimensões de partição correspondentes.

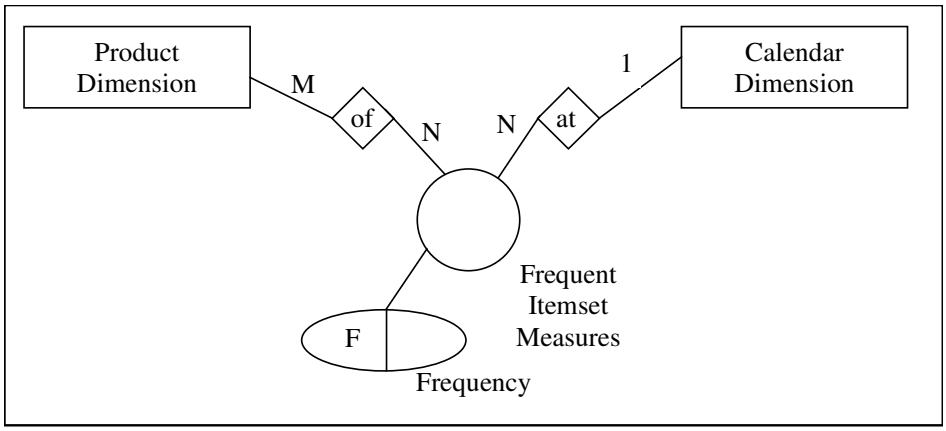


Figura 7 – Esquema conceitual para medidas de conjuntos de itens frequentes do cenário de exemplo

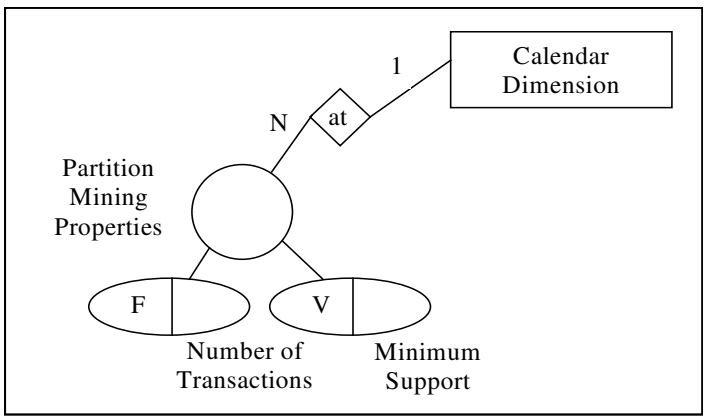


Figura 8 – Esquema conceitual para propriedades de mineração das partições do cenário de exemplo

Figura 7 e Figura 8 apresentam os esquemas conceituais para o cenário de exemplo. Uma dimensão calendário desempenha o papel de uma dimensão de partição organizando o espaço de transações originais em conjuntos disjuntos representando períodos de uma hora sem sobreposição. O esquema da Figura 8 representa o número de transações em cada período de uma hora e o suporte mínimo usada na mineração dos conjuntos de itens frequentes na etapa de pré-processamento. Uma dimensão produto descreve os produtos que podem aparecer como parte de um conjunto de itens frequentes, desempenhando portanto o papel de uma dimensão de item. O esquema da Figura 7 representa, para cada período de uma hora, as contagens de frequência de cada

conjunto de produtos minerado como frequente. Os atributos das dimensões produto e calendário não são apresentados na Figura 7 e na Figura 8 para não sobrecarregar a apresentação. Apenas para mencionar alguns, nome do produto, categoria e departamento de vendas são alguns exemplos de atributos da dimensão produto. Período do dia (manhã, tarde,...), dia da semana e feriado (sim ou não) são alguns atributos da dimensão calendário.

4.3 Projeto Lógico

Um esquema lógico padronizado é apresentado na Figura 9 e na Figura 10. No projeto lógico introduzimos uma dimensão de conjunto de itens para cada dimensão de item como pode ser visto na Figura 9. A dimensão de conjunto de itens funciona como uma tabela ponte representando os relacionamentos N para M entre as dimensões de item e o fato no esquema conceitual. Ferramentas analíticas normalmente escondem a existência de tabelas ponte do usuário. Uma ferramenta analítica desenvolvida sobre um data warehouse de conjuntos de itens frequentes deve possuir informações a respeito das dimensões de item e respectivas dimensões de conjunto de itens para prover esta transparência e efetuar automaticamente os mapeamentos necessários.

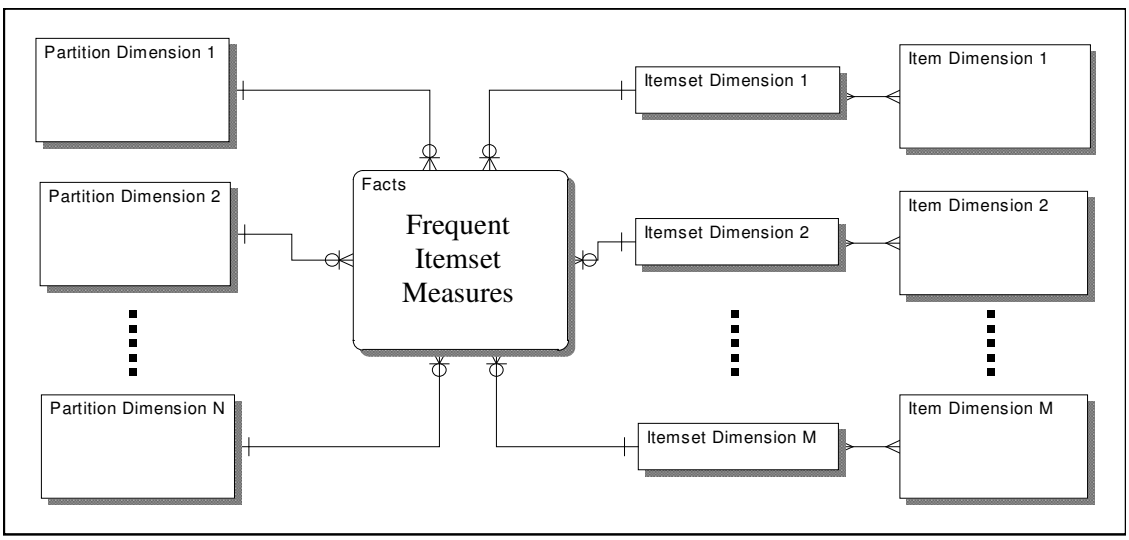


Figura 9 – Esquema lógico padronizado para medidas de conjuntos de itens frequentes

Dimensões multivaloradas (relacionamento N para M com a tabela de fatos) são comumente resolvidas com o auxílio de uma tabela ponte (Kimball and Ross, 2002). As

dimensões de conjunto de itens contém um identificador para cada conjunto de itens distinto, o qual será usado como chave estrangeira na tabela de fatos e também para agrupar os itens em conjuntos. O esquema lógico apresentado na Figura 10 é derivado diretamente do esquema conceitual da Figura 6.

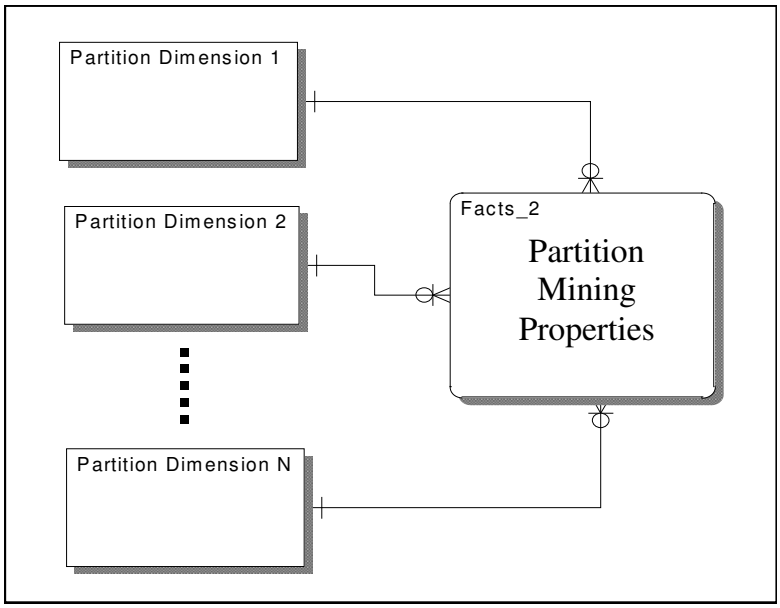


Figura 10 – Esquema lógico padronizado para propriedades de mineração das partições

O esquema lógico para o cenário de exemplo é apresentado na próxima seção junto com uma amostra de dados para cada tabela. Usando a modelagem padrão apresentada, ferramentas de análise e de exploração podem ser desenvolvidas de forma independente. Esta modelagem padrão desempenha para conjuntos de itens frequentes o mesmo papel que os esquemas estrela e snowflake desempenham para dados convencionais.

4.4 Um Exemplo de Data Warehouse de Conjuntos de Itens Frequentes

Um exemplo de data warehouse de conjuntos de itens frequentes é apresentado na Figura 11 e na Figura 12. Este exemplo possui uma dimensão de partição e uma dimensão de item.

Dimensões de partição organizam o espaço de transações originais em conjuntos disjuntos, os quais chamamos de partições do DW. No nosso exemplo usamos uma dimensão calendário com uma granularidade de uma hora. Isto significa que cada tupla

da dimensão calendário representa um período de uma hora, e.g., “02/15/2005 [08:00AM, 09:00AM)”, “02/15/2005 [09:00AM, 10:00AM)”, e assim por diante. É claro que informações de calendário adicionais têm que ser armazenadas, tais como período do dia (manhã, tarde,...), dia da semana, feriado (sim ou não) e qualquer outra unidade de calendário que represente uma agregação da granularidade básica. A granularidade básica também define o critério para coletar as transações na etapa de pré-processamento.

As dimensões de item provêm informação adicional sobre os itens que podem aparecer em uma transação. No nosso exemplo usamos uma única dimensão produto. Exemplos de informações adicionais são: descrição, departamento, categoria, etc.

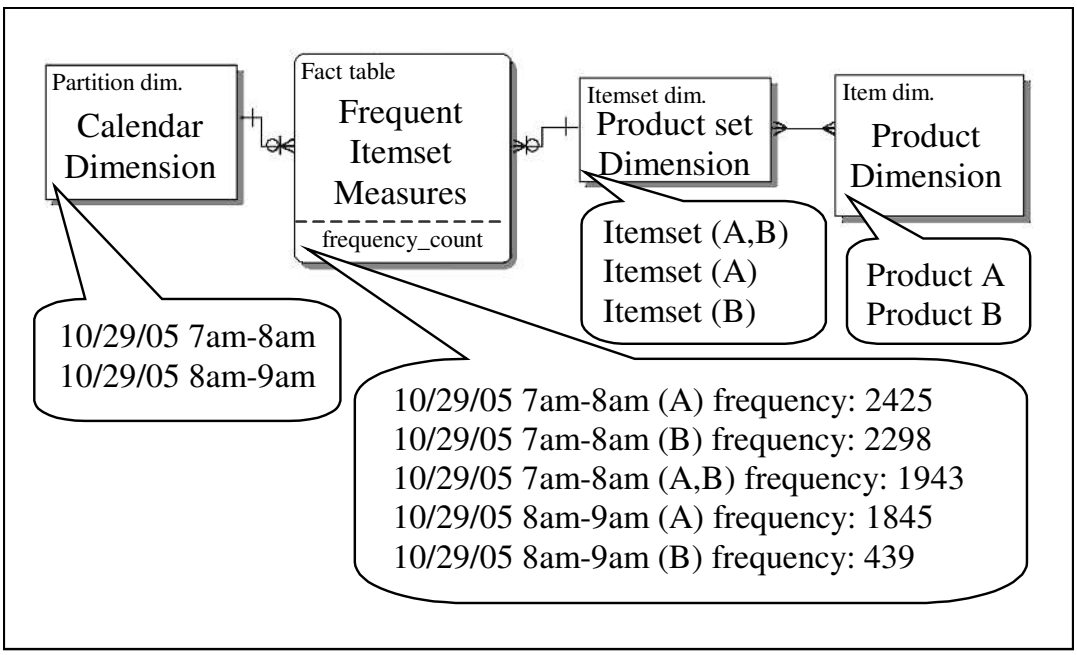


Figura 11 – Esquema lógico para medidas de conjuntos de itens frequentes com algumas amostras de dados

Um data warehouse de conjuntos de itens frequentes é composto de duas tabelas de fatos. A primeira tabela de fatos, apresentada na Figura 11, armazena as contagens de frequência para cada conjunto de produtos minerado como frequente em um grão de calendário específico (1 hora). Figura 11 também apresenta uma dimensão de conjunto de itens que funciona como uma tabela ponte entre a dimensão multivalorada e a tabela de fatos. A segunda, apresentada na Figura 12, armazena algumas propriedades de partição e, portanto, possui relacionamentos apenas com dimensões de partição. Esta

tabela de fatos armazena o número de transações e o suporte mínimo usado para mineração de conjuntos de itens frequentes por grão de calendário (1 hora).

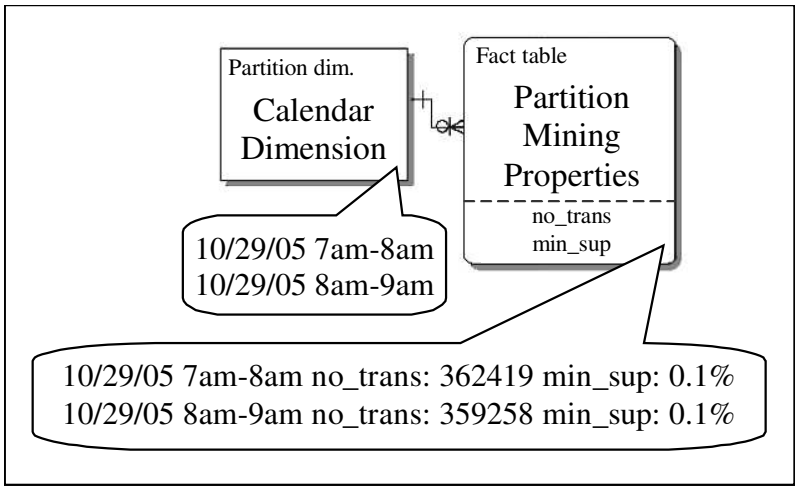


Figura 12 – Esquema lógico para propriedades de mineração das partições com algumas amostras de dados

Duas instâncias representando períodos de uma hora são mostradas na dimensão calendário. Na Figura 11 dois produtos são listados na dimensão produto e alguns conjuntos de itens relacionados são apresentados na dimensão de conjunto de itens. As contagens de frequência dos conjuntos de itens para os dois períodos de uma hora são apresentadas na tabela de fatos. Repare que não há contagem de frequência para o conjunto de itens (A, B) relacionada à partição "10/29/2005 [08:00AM, 09:00AM)". Isto significa que o conjunto de itens (A, B) não foi frequente nessa partição. Na Figura 12 o número de transações dos dois períodos de uma hora são apresentados junto com o suporte mínimo de mineração (0.1%) na tabela de fatos de propriedades de mineração de partição.

Este exemplo de data warehouse de conjuntos de itens frequentes registra os conjuntos de produtos frequentes por hora. Esta é a definição da nossa granularidade mínima de análise. Com tal granularidade mínima, não é possível analisar intervalos de tempo mais curtos do que uma hora. Entretanto, Todas as possíveis combinações de períodos de uma hora podem ser analisadas. Estas combinações baseadas em termos de calendário especificam um conjunto de partições do DW, o qual mais tarde nos referimos como ρ.

4.5 Propriedades Básicas

Um data warehouse de conjuntos de itens frequentes possui duas propriedades básicas que representam os blocos elementares para a recuperação de padrões. Estas duas propriedades são a propriedade de partições disjuntas e a propriedade de limite superior do erro, as quais são definidas a seguir.

Propriedade 1 (partições disjuntas): As partições do DW representam conjuntos de transações completamente disjuntos e, portanto, a contagem de frequência de um conjunto de itens específico pode ser somada ao longo de qualquer conjunto de partições. ■

Propriedade 2 (limite superior do erro): Dado $|\rho_i|$ o número de transações originais na partição ρ_i e σ_{mi} o suporte mínimo de mineração usado na partição ρ_i, o produto $|\rho_i|\sigma_{mi}$ provê um limite superior estrito no número de frequências perdidas na partição ρ_i. Mais além, um limite superior estrito UBe_ρ para um conjunto de partições do DW ρ é obtido simplesmente somando os limites superiores das partições individuais:

$$\text{UBe}_\rho = \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} \quad (2)$$

No caso especial quando σ_{mi} é constante ao longo de ρ temos:

$$\text{UBe}_\rho = \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} = \sigma_m \sum_{\rho_i \in \rho} |\rho_i| = \sigma_m |\rho| \quad (3)$$ ■

A propriedade 1 é válida para qualquer data warehouse de conjuntos de itens frequentes. A propriedade 2 é válida quando armazenamos todos os conjuntos de itens frequentes ou quando usamos uma representação condensada para conjuntos de itens sem perda. Ao usar uma representação condensada com perda, a definição do limite superior do erro deve ser ajustada para considerar o erro introduzido pela representação específica.

5 Recuperação de Padrões do Data Warehouse de Conjuntos de Itens Frequentes

Este capítulo trata da recuperação de padrões baseados nas informações armazenadas no data warehouse de counjuntos de itens frequentes. A seção 5.1 descreve como a informação armazenada no data warehouse de conjuntos de itens frequentes pode ser usada para recuperar conjuntos de itens frequentes válidos nas transações originais representadas por um conjunto arbitrário de partições do DW. Esta tarefa é desempenhada pelo componente "Basic Frequent Itemset Retrieval Capabilities" da abordagem DWFIST. A completude e precisão dos conjuntos de itens frequentes recuperados também são discutidas. A seção 5.2 se dedica a tarefa do componente "Frequent Itemset Based Pattern Mining Engine", o qual é responsável por derivar padrões que podem ser construídos usando conjuntos de itens frequentes. Essa tarefa é exemplificada pela obtenção de regras de associação. Outros exemplos (regras Booleanas e regras de classificação) são descritos em Monteiro (2005c).

5.1 Recuperação de Conjuntos de Itens Frequentes

Esta seção apresenta as atribuições do componente "Basic Frequent Itemset Retrieval Capabilities". A tarefa de recuperação de conjuntos de itens frequentes válidos nas transações originais representadas por um conjunto arbitrário de partições do DW é definida a seguir.

Tarefa de Recuperação de Conjuntos de Itens Frequentes: Dado um conjunto de partições do DW ρ e um suporte mínimo de consulta σ_q, recuperar o conjunto Δ de todos os conjuntos de itens I com $S_\rho(I) \geq \sigma_q$. O conjunto Δ também deve conter contagens aproximadas de frequência para cada I representadas por intervalos [LB_$F_\rho(I)$,UB_$F_\rho(I)$]. Assim sendo, se I é um conjunto de itens σ_q-frequente válido nas transações originais representadas por ρ com contagem de frequência $F_\rho(I)$ então:

$$I \in \Delta \text{ e } \text{LB\_}F_\rho(I) \leq F_\rho(I) \leq \text{UB\_}F_\rho(I) \quad (4)$$

A assinatura de função a seguir representa esta tarefa:

$$\Delta = FI\_retrieval(\rho, \sigma_q) \quad (5)\blacksquare$$

Em outras palavras, o conjunto Δ de conjuntos de itens σ_q-frequentes deve ser completo, não perdendo nenhum conjunto de itens σ_q-frequente válido nas transações originais representadas por ρ, e limites superiores e inferiores nas contagens de frequência também devem ser recuperados.

Esta tarefa não pode ser realizada para todos os valores de σ_q uma vez que o DW não possui todas as contagens de frequência de conjuntos de itens. Ao invés disso, o DW contém somente as frequências que foram mineradas como frequentes. A completude de Δ deve ser verificada e os limites inferiores e superiores de contagem de frequência computados. Questões referentes à precisão do conjunto Δ também devem ser consideradas. As propriedades do DW apresentadas na seção 4.5 fornecem a base para tratar dessas questões, as quais serão detalhadas no restante da seção 5.1.

5.1.1 Recuperando Conjuntos de Itens Frequentes e Contagens de Frequência

A propriedade a seguir especifica como os limites de contagem de frequência de conjuntos de itens recuperados a partir do DW podem ser calculados.

Propriedade 3 (limites de contagem de frequência): Limites inferiores e superiores para a contagem de frequência de um conjunto de itens *I* em um conjunto de partições do DW ρ podem ser calculadas usando as propriedades 1 e 2 da seguinte forma:

$$\mathrm{LB\_}F_{\rho}(I) = \sum_{\rho_i \in \rho_{known}} F_{\rho_i}(I) \qquad \mathrm{UB\_}F_{\rho}(I) = \mathrm{LB\_}F_{\rho}(I) + \mathrm{UBe}_{\rho - \rho_{known}} \quad (6)$$

onde ρ_{known} representa o subconjunto de partições de ρ onde o conjunto de itens *I* foi minerado como frequente e, portanto, sua contagem de frequência é conhecida. A expressão ρ - ρ_{known} representa o subconjunto complementar de ρ onde a contagem de frequência de *I* é desconhecida. ■

As propriedades 1 e 2 fornecem os blocos elementares para a construção da propriedade 3. A propriedade 3 é usada na prática para computar e recuperar do DW as frequências aproximadas de um conjunto de itens em um conjunto de transações originais. O conjunto de transações originais é representado por um conjunto arbitrário de partições do DW.

As amostras de dados apresentadas na Figura 11 e na Figura 12 podem ser usadas para exemplificar como os limites de contagem de frequência são computados. Considerando ρ igual às duas partições apresentadas na Figura 12 podemos detalhar como os limites de frequência seriam computados para os conjuntos de itens (A), (B) e (A,B) usando a propriedade 3. O limite inferior de frequência é calculado simplesmente somando as contagens de frequência conhecidas. Assim sendo, temos 4270 (2425+1845) para (A); 2737 (2298+439) para (B); e 1943 para (A,B). O limite superior de frequência é calculado usando o limite inferior de frequência e as informações armazenadas na tabela de fatos de propriedades de mineração de partição. Como as contagens de frequência dos conjuntos de itens (A) e (B) são conhecidas em ambas as partições a frequência exata é conhecida e, portanto, o limite superior de frequência é igual ao limite inferior de frequência. A frequência do conjunto de itens (A,B) é desconhecida na partição "10/29/2005 [08:00AM, 09:00AM)". O limite superior do erro para esta partição é calculado usando a equação (2) e é 359.258 (0.1% of 359258). Portanto, o limite superior de frequência para o conjunto de itens (A,B) é 2302.258 (1943+359.258). A parte decimal pode ser descartada ao final, uma vez que a contagem de frequência é uma medida inteira. Finalmente, os limites de contagem de frequência calculados são [1943,2302] para o conjunto de itens (A,B), [4270,4270] para o conjunto de itens (A) e [2737,2737] para o conjunto de itens (B).

Na sequência mostramos como derivar conjuntos de itens frequentes a partir de um DW que armazena todos os conjuntos de itens frequentes (i.e. nenhuma representação condensada é usada) através de consultas SQL. Considerando o data warehouse apresentado como exemplo, a propriedade 1 nos diz que podemos somar as contagens de frequência de um conjuntos de itens específico ao longo de qualquer partição de calendário representada na dimensão calendário. Além disso, como discutido anteriormente, a dimensão calendário fornece atributos representando diferentes características de calendário. No exemplo, um atributo *day_period* na dimensão calendário pode classificar os períodos de uma hora em manhã, tarde, noite e

madrugada. A Figura 13 apresenta um exemplo de consulta que pode ser submetida ao DW de exemplo para recuperar os conjuntos de itens e seus limites inferiores de frequência considerando o período da manhã. Retornando a definição da tarefa de recuperação de conjuntos de itens frequentes, isto corresponde a definir ρ como o conjunto de partições do DW que representam o período da manhã e computar a equação da esquerda em (6) para cada conjunto de itens.

```
Select
  FIM.itemset_id,
  sum(FIM.frequency_count) as LB_Frequency
From Calendar_Dimension CD,
     Frequent_Itemset_Measures FIM
Where CD.CD_id = FIM.CD_id and
      CD.day_period = 'Morning'
Group by FIM.itemset_id
```

Figura 13 – Consulta de limite inferior de frequência

Com base na propriedade 2 podemos calcular dois tipos de limite superior de erro: limite superior de erro global e limite superior de erro para um conjunto de itens específico. O limite de erro global calcula uma margem de erro única para todos os conjuntos de itens em um conjunto de partições do DW. O limite superior de erro para um conjunto de itens específico calcula uma margem de erro para cada conjunto de itens em um conjunto de partições do DW. O primeiro é mais fácil de calcular ao passo que o segundo é mais preciso.

Um limite de erro global para o nosso exemplo é obtido calculando a equação (2) para todas as partições relacionadas ao período da manhã (ρ). A Figura 14 apresenta a consulta correspondente.

```
Select
  sum(PMP.no_trans*PMP.min_sup)as Global_Error
From Calendar_Dimension CD,
     Partition_Mining_Properties PMP
Where CD.CD_id = PMP.CD_id and
      CD.day_period = 'Morning'
```

Figura 14 – Consulta de limite de erro global

Um limite superior de erro para um conjunto de itens específico no nosso exemplo também é obtido pela equação (2), porém, para cada conjunto de itens, somente são consideradas as partições onde a contagem de frequência é desconhecida (ρ - ρ_{known}). No paradigma relacional é mais eficiente computar o erro relacionado às

partições onde a contagem de frequência é conhecida e subtrair do erro global para obter o erro relacionado às partições onde as frequências são desconhecidas. A consulta apresentada na Figura 15 calcula para cada conjunto de itens o erro referente as partições onde a contagem de frequência é conhecida (ρ_{known}).

```
Select
  FIM.itemset_id,
  sum(PMP.no_trans*PMP.min_sup)as Known_Part_Error
From Calendar_Dimension CD,
     Frequent_Itemset_Measures FIM,
     Partition_Mining_Properties PMP
Where CD.CD_id = PMP.CD_id and
      CD.CD_id = FIM.CD_id and
      CD.day_period = 'Morning'
Group by FIM.itemset_id
```

Figura 15 – Consulta de limite superior de erro para um conjunto de itens específico

```
Select
  S.itemset_id,
  S.LB_Frequency,
  ( S.LB_Frequency +
    G.Global_Error -
    S.Known_Part_Error) as UB_Frequency
From ( Select
         FIM.itemset_id,
         sum(FIM.frequency_count) as LB_Frequency,
         sum(PMP.no_trans*PMP.min_sup) as Known_Part_Error,
       From Calendar_Dimension CD,
            Frequent_Itemset_Measures FIM,
            Partition_Mining_Properties PMP
       Where  CD.CD_id = PMP.CD_id and
              CD.CD_id = FIM.CD_id and
              CD.day_period = 'Morning'
       Group by FIM.itemset_id) S,
     ( Select
         sum(PMP.no_trans) as Total_no_trans,
         sum(PMP.no_trans*PMP.min_sup) as Global_Error
       From Calendar_Dimension CD,
            Partition_Mining_Properties PMP
       Where  CD.CD_id = PMP.CD_id and
              CD.day_period = 'Morning') G
Where( S.LB_Frequency +
       G.Global_Error -
       S.Known_Part_Error) >=
     ( G.Total_no_trans * :query_minimum_support)
```

Figura 16 – Consulta única para recuperação de conjuntos de itens frequentes

Um suporte mínimo de consulta (σ_q) deve ser especificado na recuperação. A resposta deve compreender os conjuntos de itens considerados frequentes de acordo com o suporte mínimo de consulta fornecido. As consultas apresentas na Figura 13, na Figura 14 e na Figura 15 podem ser executadas em sequência e ter os resultados combinados considerando um dado suporte mínimo de consulta porém, esta tarefa

também pode ser computada por uma única consulta. A Figura 16 mostra um exemplo. Essa consulta implementa a assinatura de função apresentada na equação (5). Uma ferramenta que disponibilize uma interface para especificação de condições nas dimensões e acesse of conjuntos de itens frequentes através do esquema lógico padronizado proposto pode construir este tipo de consulta automaticamente.

Como as contagens de frequência recuperadas são representadas por intervalos, somente podemos descartar um conjunto de itens quando a seu intervalo de frequência ficar completamente abaixo do limite especificado pelo suporte mínimo de consulta. Algumas questões importantes surgem imediatamente: (1) "Quais valores de suporte mínimo de consulta são suportados pelo DW já que este não contém informações completas sobre as transações originais?"; (2) "É possível garantir que estamos recuperando todos os conjuntos de itens frequentes válidos nas transações originais?"; e (3) "Quão precisa é a resposta obtida a partir do DW em relação ao conjunto real de conjuntos de itens frequentes válidos nas transações originais?". Estas questões são abordadas a seguir.

5.1.2 Completude

Nesta seção discutimos sob que condições a completude do conjunto Δ de conjuntos de itens frequentes pode ser garantida.

Propriedade 4 (Completude 1): Se $F_\rho(I)$ é maior ou igual a UBe_ρ então I está representado em pelo menos uma partição $\rho_i \in \rho$.

Prova: Considerando que nenhuma partição $\rho_i \in \rho$ possui qualquer informação sobre I significa que $F_{\rho i}(I) < |\rho_i|\sigma_{mi} \ \forall \ \rho_i \in \rho$. Usando as propriedade 1 e 2, e comparando com (2):

$$\sum_{\rho_i \in \rho} F_{\rho_i}(I) < \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} = \text{UBe}_\rho \quad (7) \ \blacksquare$$

Propriedade 5 (Completude 2): Seja Δ uma lista de conjuntos de itens σ_q-frequentes recuperada a partir de um conjunto de partições do DW ρ. A completude de Δ pode ser garantida para qualquer σ_q tal que:

$$\sigma_q \geq \frac{\mathrm{UBe}_\rho}{|\rho|} \quad (8)$$

Prova: Para ser considerado σ_q-frequente em um conjunto de partições do DW ρ um conjunto de itens deve ter uma frequência mínima de $\sigma_q|\rho|$. Da propriedade 4 temos que todo conjunto de itens I com $F_\rho(I) \geq \mathrm{UBe}_\rho$ estará presente em pelo menos uma partição $\rho_i \in \rho$. Se a contagem de frequência mínima $\sigma_q|\rho|$ for maior ou igual ao limite superior de erro UBe_ρ então a completude de Δ pode ser garantida:

$$|\rho|\sigma_q \geq \mathrm{UBe}_\rho \quad (9)$$

A equação (8) é diretamente obtida de (9). Além disso, no caso especial onde σ_{mi} é constante em ρ, da equação (3) temos:

$$|\rho|\sigma_q \geq |\rho|\sigma_m \qquad\qquad \sigma_q \geq \sigma_m \quad (10)\ \blacksquare$$

As propriedades 4 e 5 nos dizem na prática que a completude do conjunto de padrões recuperado é garantida desde que seja usado um suporte mínimo de consulta maior ou igual ao suporte mínimo de mineração. Em outras palavras, dada esta condição em σ_q não estamos perdendo nenhum conjunto de itens σ_q-frequente válido em um conjunto de transações originais. Mais uma vez, o conjunto de transações originais é representado por um conjunto arbitrário de partições do DW.

5.1.3 Precisão

Tratamos a precisão do resultado segundo dois aspectos: falsos positivos e amplitude do intervalo de frequência. Para uma discussão mais simples e sem perda de generalidade, consideramos nesta seção que um suporte mínimo σ_m constante foi usado para mineração na etapa de pré-processamento de todas as partições ρ_i pertencendo ao

conjunto de partições do DW ρ. Se este não foi o caso, podemos simplesmente assumir σ_m como o maior σ_{mi} usado em todos ρ_i.

O primeiro aspecto se refere a falsos positivos, i.e., conjuntos de itens que são reportados como frequentes embora a sua frequência real esteja abaixo do limite especificado pelo suporte mínimo de consulta. Tais falsos positivos podem aparecer porque a frequência recuperada do DW é uma contagem de frequência aproximada. Entretando, uma propriedade interessante de tais conjuntos de itens é apresentada a seguir:

Propriedade 6 (conjuntos de itens frequentes locais): Desde que a completude do resultado esteja garantida, um conjunto de itens falsamente reportado como frequente, devido a contagem de frequência aproximada, para um suporte mínimo de consulta σ_q em um conjunto de partições do DW ρ é necessariamente um conjunto de itens σ_q–frequente no subconjunto de partições de ρ onde a sua frequência é conhecida.

Prova: Seja I um conjunto de itens falsamente reportado como frequente para um suporte mínimo de consulta σ_q em um conjunto de partições do DW ρ. Seja também ρ_{known} o subconjunto de partições de ρ onde a frequência de I é conhecida, $F_{\rho known}(I)$ a frequência de I em ρ_{known}, $|\rho_{known}|$ o número de transações em ρ_{known} e $\mathrm{UBe}_{(\rho - \rho known)}$ o limite superior do erro para o conjunto de itens I em ρ. A seguinte equação expressa a propriedade que queremos garantir:

$$\frac{F_{\rho known}(I)}{|\rho_{known}|} \geq \sigma_q \quad (11)$$

Para que seja reportado como frequente, $F_{\rho known}(I) + \mathrm{UBe}_{(\rho - \rho kown)}$ deve ser pelo menos igual a $|\rho|\sigma_q$, portanto:

$$F_{\rho known}(I) \geq |\rho|\sigma_q - \mathrm{UBe}_{(\rho - \rho known)}$$

O limite superior do erro para o conjunto de itens específico $\mathrm{UBe}_{(\rho - \rho kown)}$ pode ser expresso como $(|\rho| - |\rho_{known}|)\sigma_m$:

$$F_{\rho known}(I) \geq |\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m$$

Substituindo $F_{\rho known}(I)$ em (11):

$$\frac{|\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m}{|\rho_{known}|} \geq \sigma_q$$
$$|\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m \geq |\rho_{known}|\sigma_q$$
$$-(|\rho| - |\rho_{known}|)\sigma_m \geq |\rho_{known}|\sigma_q - |\rho|\sigma_q$$

Multiplicando por menos um (-1):

$$(|\rho| - |\rho_{known}|)\sigma_m \leq |\rho|\sigma_q - |\rho_{known}|\sigma_q$$
$$(|\rho| - |\rho_{known}|)\sigma_m \leq (|\rho| - |\rho_{known}|)\sigma_q$$
$$\sigma_m \leq \sigma_q$$

A mesma condição que garante a completude do resultado é uma condição suficiente para a propriedade 6. ■

A propriedade acima diz que os conjuntos de itens reportados em excesso são na verdade conjuntos de itens que apresentam um comportamento frequente em um subconjunto do conjunto de partições do DW sendo analisado e, portanto, merecem provavelmente uma análise mais detalhada. Uma simples medida de cobertura relacionada ao número de partições onde a frequência de um conjunto de itens é conhecida pode auxiliar o analista em separar os conjuntos de itens que apresentam uma alta taxa de cobertura para o conjunto de partições do DW sendo analisada dos conjutos de itens frequentes locais (baixa taxa de cobertura). Uma simples contagem (Count) na consulta apresentada na Figura 15 pode facilmente computar esta medida.

O segundo aspecto se refere à amplitude do intervalo de frequência. Para discutir este aspecto introduzimos uma medida de erro relativo:

Definição 7 (erro de frequência relativo): Dada uma contagem de frequência exata $F(I)$ e um aproximação $F(I)$' com um erro ε igual a $|F(I) - F(I)'|$, o erro de frequência relativo (FRE) é definido como:

$$\text{FRE} = \frac{\varepsilon}{F(I)'} \quad (12)$$

O erro de frequência relativo quantifica o tamanho do erro em relação à contagem de frequência aproximada.

Propriedade 7 (erro de frequência relativo de pior caso): Dada uma tarefa de recuperação de conjuntos de itens frequentes e usando como aproximação de frequência o ponto médio do intervalo de frequência, o pior caso para o erro de frequência relativo é dado por:

$$\text{Worst\_case\_FRE} = \frac{\sigma_m}{2\left(\sigma_q - 0.5\sigma_m\right)} \quad (13)$$

Prova: O limite superior do erro UBe_ρ corresponde ao tamanho máximo do intervalo de frequência. Usando o ponto médio do intervalo de frequência como a frequência aproximada, o ε máximo é metade de UBe_ρ. Além disso, $F(I)$' é pelo menos ($|\rho|\sigma_q - 0.5\text{UBe}_\rho$) para ser considerado frequente. Sendo assim, o erro de frequência relativo de pior caso é:

$$\frac{\mathrm{UBe}_{\rho}}{2\left(|\rho|\sigma_q - 0.5\mathrm{UBe}_{\rho}\right)}$$

Substituindo UBe_{ρ} usando a equação (3) :

$$\frac{|\rho|\sigma_m}{2\left(|\rho|\sigma_q - 0.5|\rho|\sigma_m\right)} =$$
$$\frac{|\rho|\sigma_m}{2|\rho|\left(\sigma_q - 0.5\sigma_m\right)} =$$
$$\frac{\sigma_m}{2\left(\sigma_q - 0.5\sigma_m\right)}$$

O que nos dá o pior caso na equação (13). ■

A propriedade acima provê uma garantia de precisão para a contagem de frequência de conjuntos de itens frequentes recuperados do DW. Para $\sigma_q = \sigma_m$ a equação (13) resulta em 100% e diminui à medida que σ_q aumenta se afastando de σ_m. Como um exemplo, considere um conjunto de partições onde um suporte mínimo de mineração de 0.1% foi usado. Se recuperarmos os conjuntos de itens frequentes usando um suporte mínimo de consulta de 0.3%, a equação (13) nos diz que o erro de frequência relativo de pior caso é 20%. A completude do resultado também é garantida já que o suporte mínimo de consulta (0.3%) é maior que o suporte mínimo de mineração (0.1%).

A equação (13) pode ser usada para auxiliar na escolha do suporte mínimo de mineração. Caso esperemos recuperar conjuntos de itens frequentes usando um suporte mínimo de consulta de *S*% e queiramos garantir um erro de frequência relativo de pior caso de E%, a equação (13) pode ser aplicada para computar um suporte mínimo de mineração adequado.

5.2 Obtendo Padrões Derivados de Conjuntos de Itens Frequentes

Uma vez recuperada uma lista de conjuntos de itens frequentes e respectivas contagens de frequência aproximadas, válidos em um conjunto de partições do DW ρ usando um suporte mínimo de consulta σ_q, podemos construir outros padrões sobre a mesma. A incerteza sobre as contagens de frequência exatas deve ser levada em conta. Esta é a tarefa do componente "Frequent Itemset Based Pattern Mining Engine". Nesta seção exemplificamos esta tarefa discutindo como regras de associação podem ser obtidas de conjuntos de itens frequentes recuperados do DW. Além disso, analisando o pior caso podemos derivar limites teóricos fornecendo uma garantia de precisão para medidas de interesse. A confiança de regras de associação é apresentada como exemplo.

5.2.1 Regras de Associação

Extrair regras de associação baseadas em uma lista de conjuntos de itens frequentes é um problema bem conhecido e solucionado. Um algoritmo eficiente é apresentado em Agrawal et al. (1996). Desde que a completude do conjunto de itens frequentes seja garantida pelas propriedades 4 e 5, a completude do conjunto de regras de associação também está garantida. A questão relevante no nosso contexto é como calcular medidas relacionadas a regras de associação usando as contagens de frequência aproximadas de conjuntos de itens. Apresentamos a seguir como a confiança pode ser calculada fornecendo limites específicos para regra individuais e também uma garantia de precisão.

5.2.1.1 Confiança

A confiança de uma regra $\alpha \rightarrow \gamma$ representa a probabilidade de encontrar o consequente (γ) em uma transação dado que o antecedente (α) está presente. Dado um conjunto de partições do DW ρ, a confiança é calculada como:

$$\mathrm{conf}_{\rho}(\alpha \rightarrow \gamma) = \frac{F_{\rho}(\alpha \cup \gamma)}{F_{\rho}(\alpha)} \quad (14)$$

Limites inferiores e superiores para a confiança de uma regra individual podem ser calculadas como:

$$\text{LBconf}_{\rho}(\alpha \rightarrow \gamma) = \frac{\text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\text{UB\_}F_{\rho}(\alpha)} \qquad \text{UBconf}_{\rho}(\alpha \rightarrow \gamma) = \frac{\text{UB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\text{LB\_}F_{\rho}(\alpha)} \quad (15)$$

O raciocínio é muito simples. Selecionamos o limite inferior ou superior de frequência no numerador ou denominador de forma a minimizar ou maximizar o resultado. Usando as amostras de dados da Figura 11 e da Figura 12 podemos computar os limites de confiança para a regra A→B. Lembrando as contagens de frequência do conjunto de itens (A) [4270,4270] e do conjunto de itens (A,B) [1943,2302], e aplicando a equação (15) temos LBconf$_{\rho}$(A→B) como 45% e UBconf$_{\rho}$(A→B) como 54%. Portanto, uma confiança aproximada de 49.5% ± 4.5 pontos percentuais pode ser recuperada.

À medida que aumentamos o suporte mínimo de consulta (σ_q) para a tarefa de recuperação de conjuntos de itens frequentes, a taxa UBe$_{\rho}$/LB_$F_{\rho}(\alpha)$ entre os erros de contagem de frequência e o limite inferior de frequência dos conjuntos de itens é reduzida. O raciocínio simples é que o erro é mantido constante e os limites inferiores dos conjuntos de itens são aumentados. Sendo assim, espera-se obter menores erros de confiança à medida que aumentamos σ_q.

A seguinte propriedade provê uma garantia de precisão para a medida de confiança de qualquer regra de associação recuperada de um conjunto de partições do DW ρ usando um suporte mínimo de consulta σ_q. Para uma apresentação simples e sem perda de generalidade, consideramos que um suporte mínimo σ_m constante foi usado para mineração na etapa de pré-processamento de todas as partições ρ_i pertencendo a ρ. Se esse não for o caso, basta assumir σ_m como o maior σ_{mi} usado em todos ρ_i.

Propriedade 8: Dados um conjunto de partições do DW ρ, um suporte mínimo de mineração σ_m usado em ρ, um suporte mínimo de consulta σ_q e tomando o ponto médio do intervalo [LBconf,UBconf] como o valor aproximado da confiança, temos um erro de pior caso para a confiança expresso por:

$$\text{Worst\_Case\_conf\_error}_{\rho} = \frac{\sigma_m}{\sigma_q - \sigma_m} \quad (16)$$

Prova: Tomando o ponto médio do intervalo [LBconf,UBconf] temos o erro máximo como metade da amplitude do intervalo para uma regra de associação individual. Maximizando o erro temos:

$$\text{conf\_error}_{\rho}(\alpha \to \gamma) = \frac{\text{UBconf}_{\rho}(\alpha \to \gamma) - \text{LBconf}_{\rho}(\alpha \to \gamma)}{2} = \left(\frac{\text{UB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\text{LB\_}F_{\rho}(\alpha)} - \frac{\text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\text{UB\_}F_{\rho}(\alpha)} \right) \frac{1}{2} \leq$$

$$\left(\frac{\text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma) + \text{UBe}_{\rho}}{\text{LB\_}F_{\rho}(\alpha)} - \frac{\text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\text{LB\_}F_{\rho}(\alpha) + \text{UBe}_{\rho}} \right) \frac{1}{2} = \frac{\text{UBe}_{\rho}\left(\text{LB\_}F_{\rho}(\alpha) + \text{UBe}_{\rho}\right) + \text{UBe}_{\rho}\text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{2\text{LB\_}F_{\rho}(\alpha)\left(\text{LB\_}F_{\rho}(\alpha) + \text{UBe}_{\rho}\right)} =$$

$$\frac{1}{2\text{LB\_}F_{\rho}(\alpha)} \left(\text{UBe}_{\rho} + \frac{\text{UBe}_{\rho}\, \text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma)}{\left(\text{LB\_}F_{\rho}(\alpha) + \text{UBe}_{\rho}\right)} \right) < \frac{\text{UBe}_{\rho} + \text{UBe}_{\rho}}{2\text{LB\_}F_{\rho}(\alpha)} = \frac{\text{UBe}_{\rho}}{\text{LB\_}F_{\rho}(\alpha)} <$$

$$\frac{\text{UBe}_{\rho}}{\sigma_q|\rho| - \text{UBe}_{\rho}} = \frac{\sigma_m|\rho|}{\sigma_q|\rho| - \sigma_m|\rho|} = \frac{\sigma_m}{\sigma_q - \sigma_m}$$

■

O gráfico da Figura 17 foi desenhado usando a equação (16), Os valores do eixo horizontal referem-se a taxa σ_q/σ_m entre o suporte mínimo de consulta e o suporte mínimo de mineração. No nosso cenário de exemplo usamos um suporte mínimo de mineração de 0.1%. Assim, a Figura 17 refere-se a valores variando de 0.2% to 2% para o suporte mínimo de consulta quando consideramos o nosso exemplo. Se recuperarmos regras de associação do DW usando um suporte mínimo de consulta de 1.1% no nosso exemplo, uma precisão de 10 pontos percentuais é garantida para a medida de confiança.

O gráfico da Figura 17 pode ser usado para ajustar a configuração do DW. Podemos olha-lo de um ponto de vista diferente e usá-lo para auxiliar na escolha do suporte mínimo de mineração. Por exemplo, caso esperemos minerar regras de associação usando um suporte mínimo de consulta de *S*% e queiramos garantir um erro máximo da confiança de 11 pontos percentuais, a Figura 17 nos diz que deveríamos usar um suporte mínimo de mineração de um décimo de *S*.

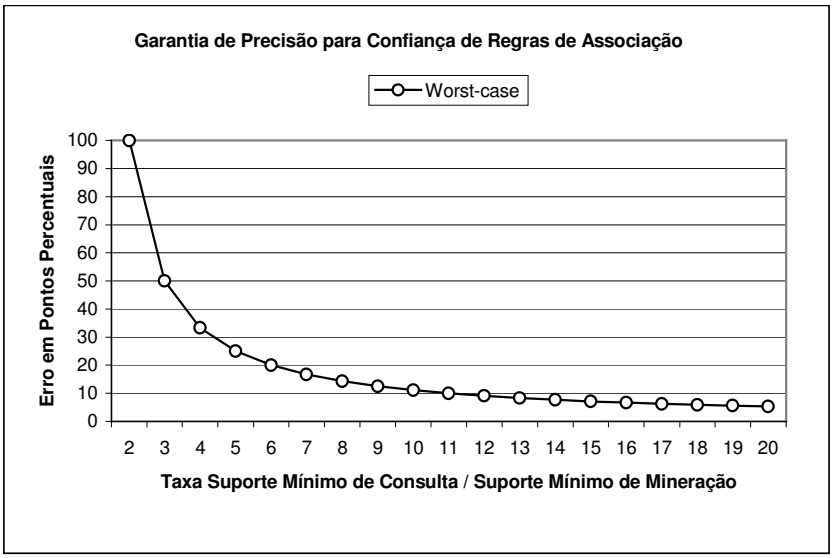


Figura 17 – Garantia de precisão para confiança de regras de associação

6 Avaliação Experimental

Executamos uma série de experimentos para avaliar as questões discutidas ao longo desta tese. Nos concentramos nas etapas de pré-processamento e carga, e nos componentes "Data Warehouse of Frequent Itemsets" e "Basic Frequent Itemset Retrieval Capabilities". A avaliação não considera os outros componentes da abordagem DWFIST porque desde que seja possível pré-processar, armazenar e recuperar conjuntos de itens frequentes, também será possível desempenhar as tarefas dos outros componentes.

O cenário de exemplo da seção 3.3 foi avaliado em um protótipo. Todos os experimentos foram executados num PC Pentium IV 1.2 GHz com 768 MB de RAM. Usamos o Oracle 10g para implementar um data warehouse idêntico ao exemplo apresentado na seção 4.4. Os únicos aspectos relevantes do projeto físico que merecem ser mencionados são o uso de índices de mapa de bits para as junções entre dimensões e tabelas de fatos; e o uso de um extent de 1MB para a tabela de fatos de medidas de conjuntos de itens frequentes.

A seguir descrevemos como os dados sintéticos usados em nossos testes foram criados e na sequência apresentamos as medidas obtidas em nossos experimentos. Uma descrição da implementação do protótipo pode ser obtida em Monteiro (2005c).

6.1 Descrição dos Dados de Teste

Os dados de teste foram criados usando o gerador de dados sintéticos de cesta de mercado da IBM que é gerenciado pelo grupo de mineração de dados Quest. Dois streams de dados foram gerados ambos representando um período de uma semana. O primeiro stream (Stream1) possui 60 milhões de transações, 1000 itens distintos e uma média de 7 itens por transação. As transações do primeiro stream foram uniformemente distribuídas em 168 partições representando períodos de uma hora (uma semana). Este stream de dados não apresenta um comportamento de padrões de calendário forte. O segundo stream de dados (Stream2) foi criado em duas etapas. Primeiro, uma base de dados foi criada com 50 milhões de transações, 1000 itens distintos e uma média de 7 itens por transação. Segundo, uma outra base de dados com 10 milhões de transações, 1500 itens distintos e uma média de 7 itens por transação foi criado. A base de 50

milhões de transações foi uniformemente distribuída em 140 partições de uma hora e a base de dados de 10 milhões de transações foi distribuída em 28 partições de uma hora. Finalmente, as 28 partições de uma hora da base de 10 milhões de transações foram associadas ao período da manhã (08:00AM até 12:00PM) dos dias úteis, e também ao período de 08:00AM a 17:00 excluindo o almoço (12:00PM até 13:00) do sábado. As 140 partições de uma hora da base de dados de 50 milhões de transações foram associadas aos períodos restantes completando uma semana. Este segundo stream de dados apresenta um forte comportamento baseado em calendário nos períodos da manhã e no sábado devido aos itens adicionais existentes na base de dados de 10 milhões de transações.

6.2 Experimentos

O foco de nossos experimentos está relacionado aos tempos de pré-processamento e carga, requisitos de espaço de armazenamento do DW, e precisão dos conjuntos de itens frequentes recuperados do DW.

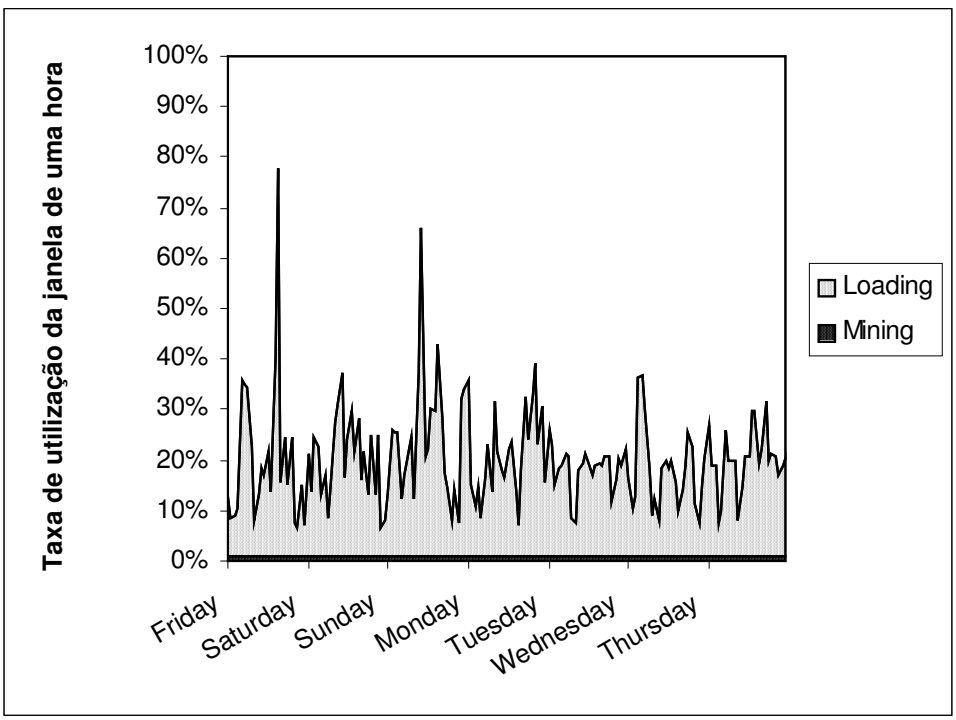


Figura 18 – Tempo de pré-processamento e carga para o Stream1

Os tempos de pré-processamento e carga são apresentados na Figura 18 e na Figura 19. Nesses dois gráficos o eixo Y representa o percentual da janela de uma hora que foi necessária para o processamento enquanto que o eixo X representa as partições

de uma hora. A etapa de pré-processamento refere-se à mineração de conjuntos de itens baseada em uma implementação otimizada do algoritmo FP-Growth (Han et al., 2000) cuja descrição pode ser obtida em Grahne and Zhu (2003). Usamos um suporte mínimo de mineração de 0.1% para todas as partições de uma hora. Como esta tarefa foi executada em menos de um minuto para todas as partições, ela aparece apenas como uma fina camada na base dos gráficos. Figura 18 e Figura 19 mostram que para a maioria das partições as etapas de pré-processamento e carga foram completadas em menos de 30% da janela de tempo disponível. Mesmo as raras partições com picos de processamento, o pré-processamento e carga foram completados dentro da janela de uma hora.

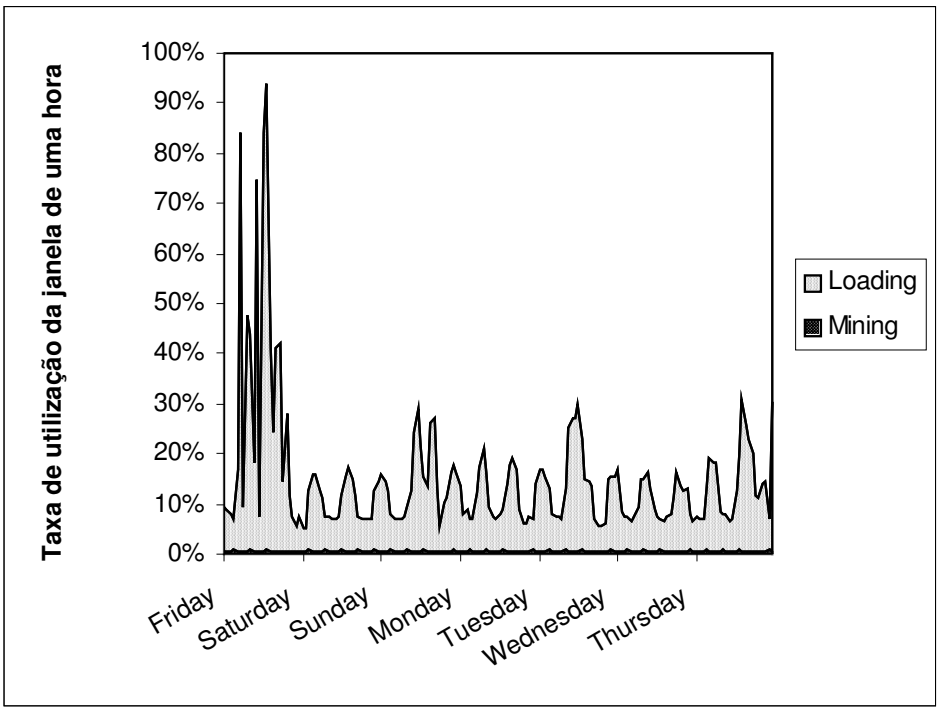


Figura 19 – Tempo de pré-processamento e carga para o Stream2

Figura 20 e Figura 21 apresentam os requisitos de espaço de armazenamento comparando os tamanhos dos streams com o DW correspondente. É evidente que os requisitos de armazenamento para os streams rapidamente atingem níveis não gerenciáveis. Para o nosso exemplo, somente uma semana já representa 2GB. Do outro lado, o tamanho do DW aumenta a uma taxa menor e é ordens de grandeza menor do que o tamanho do stream de dados fonte. Em ambas bases de testes, o tamanho do DW não alcançou 20MB incluindo as estruturas de indexação.

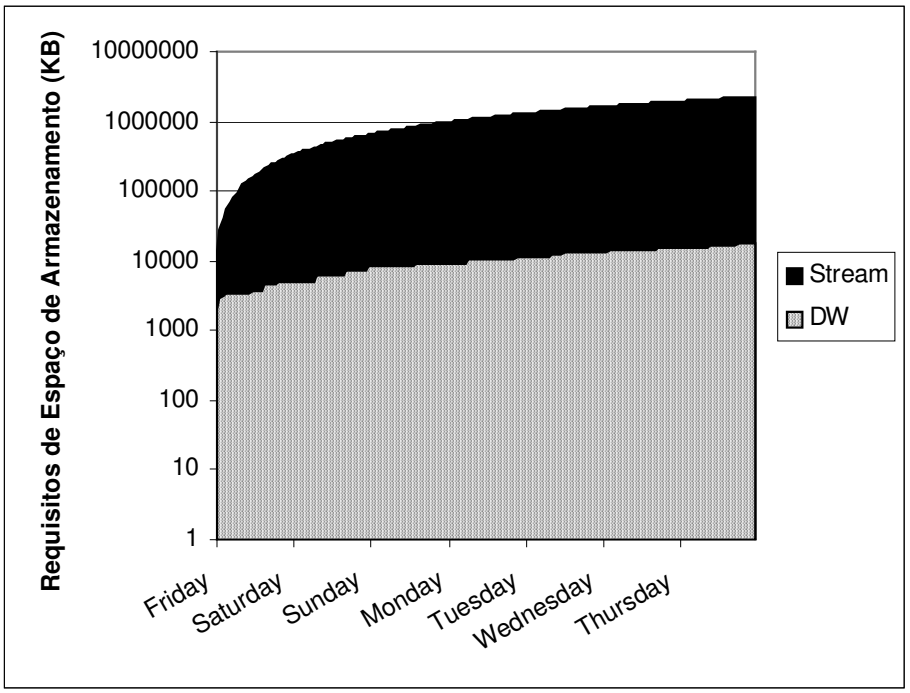


Figura 20 – Requisitos de espaço de armazenamento para o Stream1

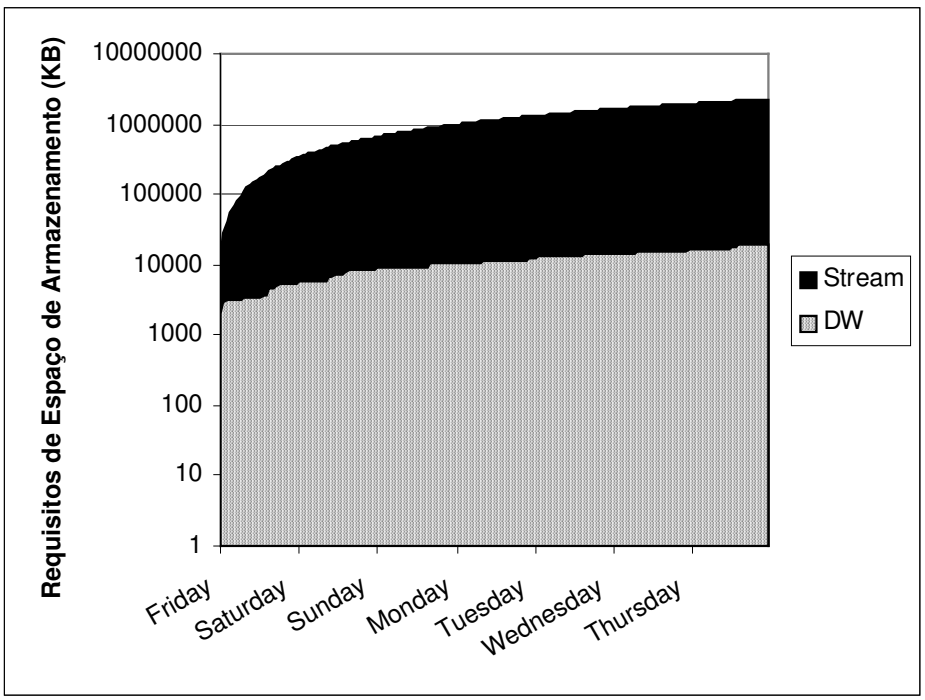


Figura 21– Requisitos de espaço de armazenamento para o Stream2

A última parte de nossos experimentos apresenta medidas relacionadas à precisão dos conjuntos de itens frequentes recuperados do DW. Recuperamos conjuntos de itens frequentes em três calendários diferentes: sábado (00:00 to 24:00); segunda, quarta e sexta de 08:00 às 17:00; e período da manhã de dias úteis (08:00 to 12:00). Estes calendários se relacionam com o forte comportamento de calendário introduzido

no Stream2 de formas diferentes. “sábado” compreende 24 partições das quais 8 são relacionadas a base de dados de 10 milhões de transações. “segunda, quarta e sexta de 08:00 às 17:00” apresenta 12 de 27. “período da manhã de dias úteis” é relacionada somente a partições da base de dados de 10 milhões de transações. Estas partições de calendário foram definidas de forma a medir no Stream2 o efeito de um forte comportamento de calendário na precisão do resultado. Executamos as mesmas consultas no Stream1, o qual não apresenta um comportamento de calendário.

O conjunto de itens frequentes foi recuperado em menos de 5 segundos em todos os três casos. A tarefa de mineração correspondente acessando diretamente os dados originais foi executada em torno de 50 segundos, o que atesta a redução do esforço computacional.

A precisão dos conjuntos de itens frequentes recuperados é medida segundo dois aspectos: número de falso positivos e erro de frequência relativo. Para obter estas medidas, executamos uma mineração convencional de conjuntos de itens frequentes para cada consulta diretamente nas transações correspondentes dos streams. A resposta recuperada do DW foi comparada com a resposta exata.

Como os resultados das três consultas executadas no Stream1 apresentaram o mesmo comportamento, apresentamos somente os resultados para uma delas na Figura 22. A diferença entre as duas barras do gráfico superior corresponde ao número de falso positivos. Repare que com um suporte mínimo de consulta ligeiramente maior que o suporte mínimo de mineração (0.1%) já é possível recuperar o conjunto exato de conjuntos de itens frequentes. O gráfico inferior apresenta medidas do erro de frequência relativo. “Worst-case” corresponde ao pior caso e é o valor obtido aplicando a equação (13). “Maximum DW” e “Average DW” são computadas calculando o erro como a diferença entre a frequência aproximada e o limite superior de frequência, correspondendo portanto à amplitude do intervalo de frequência. “Maximum Real” e “Average Real” usam o erro como a diferença entre a frequência aproximada e a frequência real exata recuperada diretamente do stream. O erro cai rapidamente a zero para valores do suporte mínimo de consulta ligeiramente maiores que o suporte mínimo de mineração.

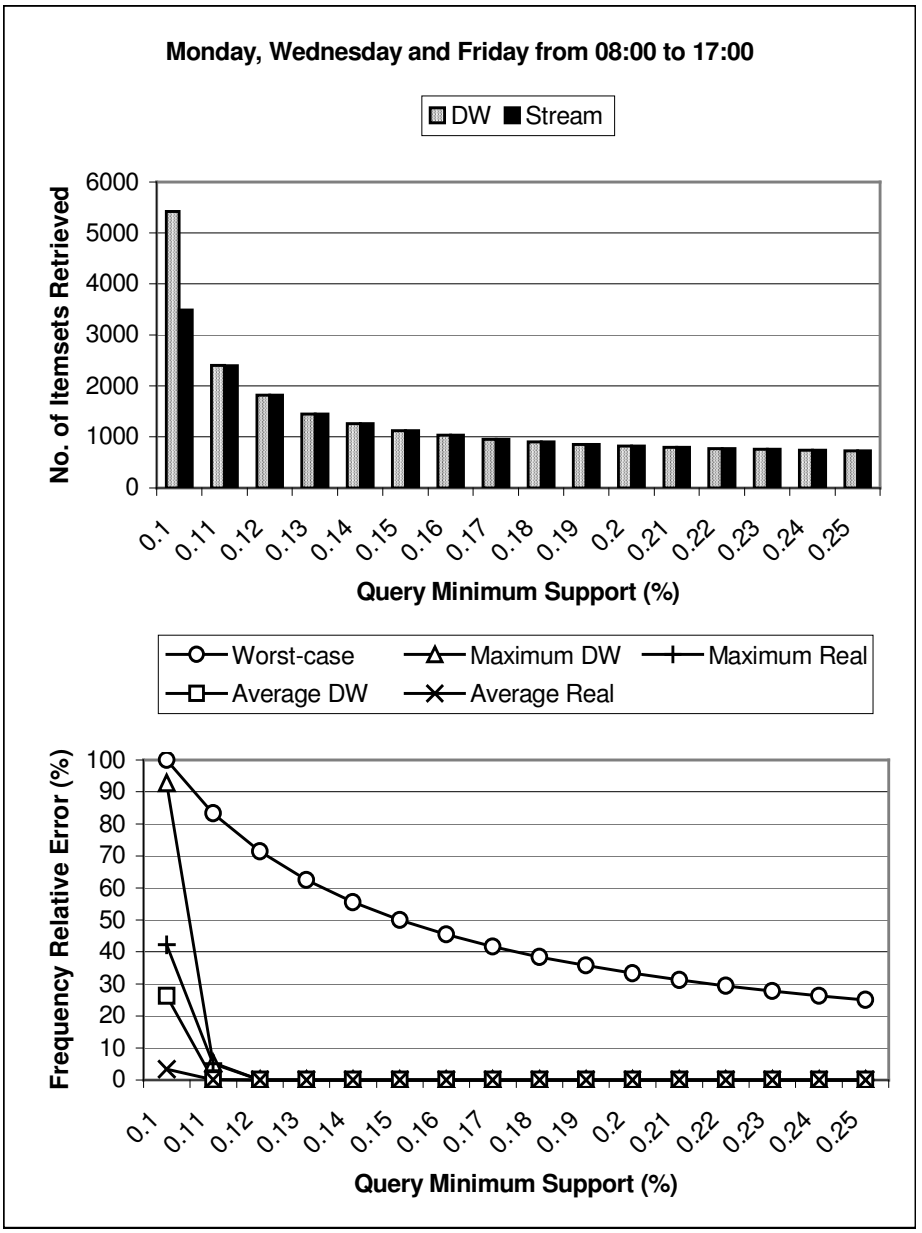


Figura 22 – Medidas de precisão para o Stream1

A Figura 23 apresenta os mesmos gráficos para as três consultas executadas no Stream2. Mais uma vez, para valores do suporte mínimo de consulta ligeiramente maiores que o suporte mínimo de mineração o resultado já apresenta uma boa precisão. Por exemplo, com um suporte mínimo de consulta de 0.12% o erro de frequência relativo medido pelo "Average DW" já é reduzido a taxas inferiores a 10%. É especialmente importante observar os baixos valores para a medida "Average DW". Como esta medida não requer conhecimento sobre a frequência real exata ela representa a garantia de precisão média que será fornecida para um usuário consultando o DW. Além disso, a consulta para o período da manhã de dias úteis apresentou o mesmo comportamento que as consultas executadas no Stream1.

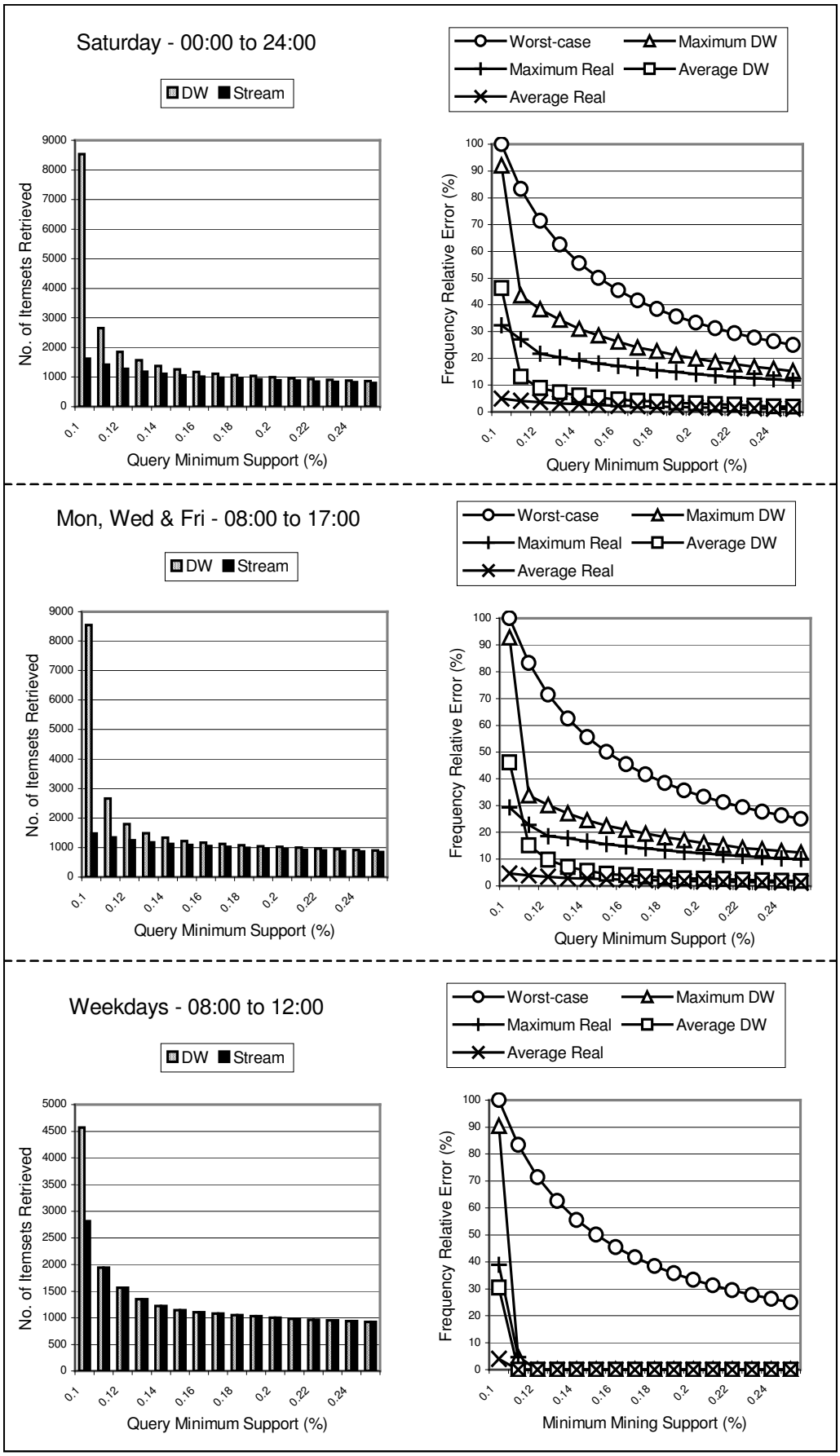


Figura 23 – Medidas de precisão para o Stream2

7 Conclusões e Trabalhos Futuros

Nesta tese nós propusemos uma abordagem que visa dar suporte à análise e exploração de conjuntos de itens frequentes e padrões derivados, e.g. regras de associação, em bases de dados transacionais. Este capítulo conclui esta tese apresentando um sumário do trabalho e algumas direções para pesquisas futuras.

7.1 Sumário

Esta tese propôs a abordagem "Data Warehouse of Frequent Itemsets Tactics" (DWFIST) focando em dois tópicos: (1) prover uma modelagem padrão para data warehouses de conjuntos de itens frequentes e (2) minerar conjuntos de itens frequentes baseados em calendário em stream de dados. Um data warehouse desempenha um papel central na nossa abordagem. Este armazena conjuntos de itens frequentes válidos em diferentes partições das transações originais retendo informação para análises futuras. Discutimos questões que devem ser desempenhadas na staging area. Um cenário de exemplo para mineração de conjuntos de itens frequentes baseados em calendário foi apresentado. Este exemplo foi implementado em um protótipo e medidas foram apresentadas para ilustrar como a abordagem proposta funciona na prática. A modelagem padrão proposta para data warehouses de conjuntos de itens frequentes fornece uma visão lógica padronizada sobre a qual ferramentas de análise e de exploração podem ser desenvolvidas independentemente. Um conjunto de propriedades do data warehouse de conjuntos de itens frequentes permite a recuperação flexível de conjuntos de itens frequentes válidos em um conjunto qualquer de partições do DW. Usando o cenário de exemplo, exemplificamos como a flexibilidade oferecida para recuperação de padrões pode ser usada para minerar padrões baseados em calendário. Finalmente, a recuperação de padrões baseados em conjuntos de itens frequentes do DW foi exemplificada através de regras de associação.

A completude dos conjuntos de itens frequentes recuperados do DW pode ser garantida e um limite de pior caso teórico para o erro relativo da frequência de conjuntos de itens frequentes recuperados do DW foi apresentado.

A precisão de diferentes conjuntos de itens recuperados do DW foi medida. O número de falso positivos cai rapidamente à medida que o suporte mínimo de consulta

se afasta do suporte mínimo de mineração. Além disso, a amplitude do intervalo de frequência apresenta um comportamento médio muito melhor do que o pior caso teórico.

7.2 Trabalhos Futuros

Como direções futuras gostaríamos de destacar o desenvolvimento de ferramentas de análise e exploração para investigar grandes conjuntos de padrões. Tais ferramentas podem se basear no esquema lógico padronizado proposto pela nossa abordagem para acessar e recuperar padrões. Avaliar a aplicabilidade e uso de diferentes representações condensadas para representar conjuntos de itens frequentes numa partição do DW é também um tópico interessante. O uso de representações condensadas para conjuntos de itens frequentes, como closed frequent itemsets, pode reduzir os requisitos de espaço de armazenamento e também trazer benefícios para o processo de carga do DW. Entretanto, um processamento adicional será introduzido na recuperação de padrões do DW. Esta relação deve ser melhor pesquisada e analisada. Por último, prover outros padrões baseados em conjuntos de itens frequentes é um tópico adicional de pesquisa futura.

8 Referências

Agrawal, R., Imielinski, T., and Swami, A. (1993). “Mining Association Rules between Sets of Items in Large Databases”. Proc. ACM SIGMOD Conf., pp. 207-216, Washington.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996). “Fast Discovery of Association Rules”. In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramaswamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp. 307-328, AAAI/MIT Press.

Beyer, K., and Ramakrishnan, R. (1999). “Bottom-up computation of sparse and iceberg cubes”. In Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99), pp. 359-370.

Boulicaut, J. (2004). “Inductive databases and multiple uses of frequent itemsets: the cInQ approach”. In Database support for Data Mining Applications, R. Meo et al. Eds., LNCS 2682. pp. 3-26.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1998). “Advances in Knowledge Discovery and Data Mining”. AAAI Press.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P.S. (2003). “Mining Frequent Patterns in Data Streams at Multiple Time Granularities”. H. Kargupta et al. Eds. Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT Press.

Grahne, G., and Zhu, J. (2003). “Efficiently Using Prefix-trees in Mining Frequent Itemsets”. Proceeding of the First IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, November.

Han, J., and Kamber, M. (2001). “Data mining: concepts and techniques”. Academic Press.

Han, J., Pei, J., and Yin, Y. (2000). “Mining frequent patterns without candidate generation”. In Proceeding of the 2000 SIGMOD Conference, pp. 1-12, Dallas, Texas, May.

Kimball, R., and Ross, M. (2002). “The Data Warehouse Toolkit: The Complete Guide to Dimensional Modelling”, Wiley Publishers, second edition, ISBN 0471200247.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. I. (1994). “Finding Interesting Rules from Large Sets of Discovered Association Rules”. Proceeding of the Third International Conference on Information and Knowledge Management, pp. 401-407.

Liu, B., Hsu, W., and Ma, Y. (1998). “Integrating classification and association rule mining.” In Proceedings KDD’98, pp. 80–86, AAAI Press, New York, USA.

Manku, G., and Motwani, R. (2002). “Approximate frequency counts over data streams”. In Proc. Int. Conf. Very Large Data Bases (VLDB'02), 346.357.

Mannila, H., and Toivonen, H. (1996). “Multiple Uses of Frequent Sets and Condensed Representations”. In Proceedings KDD'96, pp. 189-194, AAAI Press, Portland.

Monteiro, R. S., (2005c). “DWFIST: The Data Warehouse of Frequent Itemsets Tactics Approach”. Programa de Engenharia e Sistemas e Computação, COPPE/UFRJ, Relatório Técnico.

Monteiro, R. S., Zimbrao, G., Schwarz, H., Mitschang, B., and Souza, J. M. (2005a). “Building the Data Warehouse of Frequent Itemsets in the DWFIST Approach”. Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems, pp. 294-303, LNCS 3488, Saratoga Springs, NY, USA.

Monteiro, R. S., Zimbrao, G., Schwarz, H., Mitschang, B., and Souza, J. M. (2005b). “DWFIST: The Data Warehouse of Frequent Itemsets Tactics Approach”. Book chapter to appear in *Processing and Managing Complex Data for Decision Support*, Idea Group Inc.

Monteiro, R. S., Zimbrão, G., and Souza, J. M. (2003). “An Analytical Approach for Handling Association Rule Mining Results”. Proceedings of the 2003

Australasian Data Mining Workshop, (AusDM'03), Canberra, Australia, December.

Tryfona, N., Busborg, F., and Christiansen, J. G. B. (1999). "starER: A Conceptual Model for Data Warehouse Design". Proc. Int. Workshop on Data Warehousing and OLAP, pp. 3-8.

Sathe, G., and Sarawagi, S. (2001). "Intelligent rollups in multidimensional OLAP data". Proceedings of the 2001 International Conference on Very Large Data Bases.

Wang, H.; Yang, J.; Wang, W.; and Yu, P. S. (2002). "Clustering by pattern similarity in large data sets". In Proc. ACM-SIGMOD Int. Conf. on Management of Data, pp. 418-427.

Anexo A – Relatório Técnico Versão em Inglês

DWFIST: The Data Warehouse of Frequent Itemsets Tactics Approach

Rodrigo Salvador Monteiro

ABSTRACT

This thesis proposes the DWFIST approach, which is concerned with supporting the analysis and exploration of frequent itemsets and derived patterns, e.g. association rules, in transactional datasets. The goal of this new approach can be summarized as the following twofold contribution: provide (1) flexible pattern-retrieval capabilities without requiring the original data during the analysis phase, and (2) a standard modeling for data warehouses of frequent itemsets allowing an easier development and reuse of tools for analysis and exploration of itemset-based patterns.

A data warehouse storing frequent itemsets holding on different partitions of the original transactions plays a central role in our approach. After discussing pre-processing tasks performed in the staging area, we present standard conceptual and logical schemas aiming at a standard modeling. Properties of this standard modeling allow for a flexible combination of any set of partitions. The frequent itemsets holding on any set of partitions can be retrieved along with upper and lower bounds on their frequency counts. Completeness and precision issues related to the retrieved set of frequent itemsets are discussed. Besides, through a worst-case analysis, a precision guarantee is provided for some interestingness measures of patterns that can be derived from frequent itemsets (e.g. confidence of association rules).

One particularly important application of our approach is that it can be used to leverage calendar-based pattern mining in data streams. This is an especially challenging task as the calendar partitions of interest are not known a priori and at each point in time only a subset of the detailed data is available. Potential calendar partitions are for example: every Monday, every first working day of each month, every holiday. We discuss how the required staging area tasks can cope with tight time constraints imposed by a data stream scenario. Furthermore, a series of experiments support the effectiveness of the proposed approach.

Index

1 Introduction

Advances in data gathering mechanisms, the use of bar codes in most commercial products, and information on many business and governmental transactions have been flooding us with data, creating an urgent need for new techniques and tools to intelligently and automatically support the transformation of this data into useful knowledge (Fayyad et al., 1998). The research field of Data Mining has provided many different techniques to explore the data and reveal different kinds of pattern or non-pattern behavior (Han and Kamber, 2001). Recently, some of these techniques have been adapted to cope with tight time requirements imposed by data streams.

Mining frequent patterns in transactional data, in particular the pattern domain of itemsets, is a technique that deserves special attention due to its large applicability (Boulicaut, 2004). The most recognized one is to support association rule mining, which was introduced by Agrawal et al. (1993). Since then, many works proposing algorithms for association rule mining affirm the importance and attention that has been dedicated to this data mining task. Manku and Motwani (2002) and Giannella et al. (2003) presented algorithms to mine association rules in data streams. Iceberg cube computation (Beyer and Ramakrishnan, 1999), associative classification (Liu et al., 1998), frequent pattern-based clustering (Wang et al., 2002) and generalized rule mining (Mannila and Toivonen, 1996) are other examples where the frequent itemsets are useful.

This thesis proposes the DWFIST approach, which is concerned with supporting the analysis and exploration of frequent itemsets and derived patterns, e.g. association rules, in transactional datasets. The acronym DWFIST stands for Data Warehouse of Frequent Itemsets Tactics. In other words, our approach organizes sets of items occurring frequently together in transactions applying a dimensional modeling and aiming at supporting the analysis and exploration of patterns.

Following we present the motivation of our work and establish the scope and goals of this thesis. The contributions are presented in the sequence and the thesis outline is provided.

1.1 Motivation

Some data mining tasks can produce such great amounts of data that there has arisen a new knowledge management problem (Klemettinen et al., 1994). Frequent itemset mining is long known for fitting in this category. The analysis of the results of a frequent itemset mining task is far away from being trivial. The same is true for many patterns built upon frequent itemsets, such as association rules. The analyst can be easily confronted with a huge number of patterns during such an analysis. Identifying the patterns that are really interesting is a difficult task. Specialized analytical and explorative tools must be devised in order to aid the analyst in such task (Monteiro et al., 2003). The lack of a standard way for organizing, storing and accessing frequent itemsets makes the effort of developing such tools very high as it avoids the reuse of general solutions for different environments.

Recent applications, such as network traffic analysis, web click stream mining, power consumption measurement, sensor network data analysis, and dynamic tracing of stock fluctuation are some examples where a new kind of data arises, the so called data stream. A data stream is continuous and potentially infinite. It is challenging to mine frequent patterns in data streams because this task is essentially a set of join operations whereas join is a typical blocking operator, i.e., computation for any itemset cannot complete before seeing the past and the future data sets (Giannella et al., 2003). Providing flexibility to mine frequent itemsets in some subset of the data stream is even more challenging, especially when the subset is not known a priori.

1.2 Problem Statement

This thesis focus on providing a standard modeling for frequent itemsets holding on a transactional dataset as well as standard access and retrieval. These features aim at filling the existing gap for devising analytical and explorative tools for such patterns that can be easily reused in different contexts. Furthermore, we show that our approach can be used with stream data sources leveraging kinds of temporal analysis for such data that were not possible so far.

We formally state the problems we are addressing, in order to define the scope of this thesis, as follows:

- **Standard modeling for data warehouses of frequent itemsets:** Define a standard data warehouse modeling applicable to any transactional dataset and suitable for analysis tasks that rely on frequent itemsets.

- **Mining calendar-based frequent itemsets on data streams:** Let *D* be a transactional dataset that is provided as a data stream by some data source. Hence, only a subset of the transactions in *D* is available at any point in time. Nevertheless, we want to derive frequent itemsets from *D* based on a set of ad-hoc calendar-based constraints. ■

Some examples of calendar-based constraints are: weekday in {Monday, Friday}; day_period = “Morning”; holiday = “yes”; etc.

1.3 Contributions

The contributions of this thesis are compiled at this section, presenting a summary of our solutions to the problems mentioned in the previous section.

The research field of data warehousing has been extremely successful in providing efficient and effective ways to store and organize huge amounts of data. It succeeded also in providing a standard modeling upon which reusable analytical tools could be designed and implemented. This thesis presents a standard modeling for organizing frequent itemsets applicable to any transactional dataset. The product of this standard modeling is a Data Warehouse of Frequent Itemsets, which is the main component of the DWFIST approach. It organizes the whole set of transactions by disjoint partitions and stores information about the frequent itemsets holding on each partition. Different partitions can be combined to obtain the frequent itemsets with approximate support guarantees holding on any set of partitions. Moreover, this work provides the following contributions:

- Flexible pattern-retrieval capabilities without requiring the original data during the analysis phase;

- A standard modeling for data warehouses of frequent itemsets allowing an easier development and reuse of tools for analysis and exploration of itemset-based patterns;

- A conceptual view (dimensional model) for pattern analysis that is familiar to business professionals and suitable for the analysis of huge volumes of data;
- Definition of an ETL (Extraction, Transformation, Load) process for a data warehouse of frequent itemsets;
- A set of properties of the data warehouse of frequent itemsets, including the completeness of the set of retrieved patterns and precision issues;
- Precision guarantee for measures of different patterns derived from the data warehouse of frequent itemsets;
- Present how DWFIST can be applied to leverage calendar-based pattern mining capabilities in the data stream scenario;

Our approach does not propose a new algorithm for mining frequent itemsets directly on streams. Instead, we rely on existing algorithms to perform regular frequent itemset mining on small batches of the stream data, and provide means to flexibly combine the frequent itemsets of different batches. Instead of devising new data structures, the DWFIST approach is a database centered approach thus it benefits from database index structures, query optimization, storage management facilities and so on. These are vital features because the data warehouse of frequent itemsets potentially comprises a large, but manageable, volume of data. Also, our approach can be implemented using conventional commercial databases.

To the best of our knowledge, no previous approach is capable of deriving calendar-based patterns in data streams.

1.4 Thesis outline

The remainder of this thesis is organized as follows. Chapter 2 presents some background information for our work, namely frequent itemsets and data warehouse. Other approaches related to our work are presented as well. The DWFIST approach is introduced in chapter 3. An overview is provided along with requirements, components, and a running example that is used throughout the thesis. The main component of the DWFIST approach, the Data Warehouse of Frequent Itemsets, is depicted in chapter 4. Staging area issues, including data stream scenario considerations, are discussed in this

chapter together with conceptual and logical designs. Basic properties of the Data Warehouse of Frequent Itemsets are also presented. Chapter 5 discusses the retrieval of patterns from the data warehouse. It starts with the retrieval of frequent itemsets and proceeds to other patterns that can be derived from frequent itemsets. Chapter 6 is dedicated to the experimental evaluation. The prototype implemented is described and experiments related to calendar-based pattern mining in data streams are presented and discussed. The conclusion and some future research topics are presented in chapter 7.

2 Background

This chapter contains some background information providing the basic knowledge for understanding our work. It is composed of three main sections. Section 2.1 depicts the required background on frequent itemsets. The main concepts of the data warehouse field are presented in section 2.2. Finally, section 2.3 addresses existing works related to our approach discussing why they are not capable of solving the problems we are dealing with.

2.1 Frequent Itemsets

Mining frequent patterns in transactional data, in particular the pattern domain of itemsets, is a technique that deserves special attention due to its broad applicability (Boulicaut, 2004). In the following sub-section we present definitions for important terms and introduce some notation that will be used throughout this thesis. In the sequence, the frequent itemset mining task is addressed. Sub-section 2.1.3 briefly illustrates the idea of condensed representation for frequent itemsets. Finishing this section, sub-section 2.1.4 deals with some examples of patterns that can be derived from frequent itemsets.

2.1.1 Basic Definitions

Let us begin from scratch in order to precisely define which kind of data and patterns we are dealing with. The terminology related to frequent itemsets used throughout this thesis is introduced as well.

Definition 1 (transactional dataset): Let *items* be a finite set of symbols denoted by capital letters, e.g., *items* = {A, B, C, …}. A transaction t is a subset of *items*. A transactional dataset D is a non-empty multiset $D = \{t_1, t_2, \ldots, t_n\}$ of transactions. ■

Market basket data (transactions are sets of products that are bought by customers) is a classic example of a transactional dataset. Textual data (transactions are

sets of keywords that characterize documents) and gene expression data (transactions are sets of genes that are over-expressed in given biological conditions) are other examples (Boulicaut, 2004).

Definition 2 (itemsets): An *itemset* is a subset of *items*. ■

Definition 3 (frequency of itemsets): A transaction t supports an *itemset* I if every *item* in I belongs to t. The *frequency* of I over a set of transactions T, where $T \subseteq D$, is the number of transactions in T supporting I and is denoted by $F_T(I)$. ■

Definition 4 (support of itemsets): Given $|T|$ the number of transactions in T, where $T \subseteq D$, the *support* of an *itemset* I over T is the *frequency* of I divided by $|T|$ and is denoted by $S_T(I)$:

$$S_T(I) = \frac{F_T(I)}{|T|} \quad (1)$$

the support of an itemset is a rate denoting the percentage of transactions in T supporting I. ■

Definition 5 (frequent itemsets): Given a minimum support σ, an *itemset* I is considered frequent over a set of transactions T if $S_T(I) \geq \sigma$. Such an *itemset* I is a σ-frequent itemset over T. ■

Definition 6 (calendar-based frequent itemsets): Let X be a set of calendar-based constraints, Φ the set of transactions satisfying X, where $\Phi \subseteq D$, and σ a minimum support. The set of *calendar-based frequent itemsets* is defined by every *itemset* I with $S_\Phi(I) \geq \sigma$. ■

Some examples of calendar-based constraints are: weekday in {Monday, Friday}; day_period = “Morning”; holiday = “yes”; first working day of the month = “yes”; season in {summer, spring}; year = “2003”; etc.

2.1.2 Frequent Itemset Mining Algorithms

Extracting frequent itemsets holding on some transactional dataset is a fundamental data mining task. This task was first discussed in Agrawal et al. (1993) as the first step for mining association rules. Since then, various approaches for frequent itemset mining were developed. The attention dedicated to this mining task is partially justified by its computational complexity, which grows exponentially with the number of items involved. Table 1 presents a list of the most relevant frequent itemset mining algorithms proposed in the literature. The list is presented in chronological order.

Table 1 – Frequent itemset mining algorithms

Algorithm	Original work
Apriori	Agrawal and Srikant (1994)
Offline Candidate Determination (OCD)	Mannila et al. (1994)
Set Oriented Mining (SETM)	Houtsma and Swami (1995)
Direct Hashing and Pruning (DHP)	Park et al. (1995)
MONET System	Holsheimer et al. (1995)
Partition	Savasere et al. (1995)
Sampling	Toivonen (1996)
Apriori TID, AprioriHybrid	Agrawal et al. (1996)
Eclat, MaxEclat, Clique, MaxClique	Zaki et al. (1997)
Dynamic Itemset Counting (DIC)	Brin et al. (1997)
Max-Miner	Bayardo (1998)
Calendric Frequent Itemsets	Ramaswamy et al. (1998)
Carma	Hidber (1999)
A-Close	Pasquier et al. (1999)
FP-Growth	Han et al. (2000)
Free Itemsets	Boulicaut et al. (2000)
CLOSET	Pei et al. (2000)
LPMiner	Seno and Karypis (2001)
MAFIA	Burdick et al. (2001)
Tree Projection	Agarwal et al. (2001)
Temporal Apriori	Li et al. (2001)
GenMAX	Gouda and Zaki (2001)
Direct Count and Intersect (DCI)	Orlando et al. (2002)
Lossy Counting	Manku and Motwani (2002)
CHARM	Zaki and Hsiao (2002)
FPMax	Grahne and Zhu (2003b)
FP-Stream	Giannella et al. (2003)

It is not our purpose to discuss the details and differences between the mentioned frequent itemset algorithms. The list above is intended to provide a starting point for readers interested in studying this mining task. Hipp et al. (2000), Ganti et al. (1999) and the survey in Ayan (1999), although outdated, present a good overview of the existing scenario by 1999 and 2000. By reading these surveys and following the most recent works, such as Manku and Motwani (2002) and Giannella et al. (2003) that presented algorithms to mine frequent itemsets in data streams, it is possible to quickly get the big picture of the state of the art.

2.1.3 Condensed Representation for Frequent Itemsets

The goal of condensed representations for frequent itemsets (Mannila and Toivonen, 1996) is to represent a set of frequent itemsets in a more compact way. Some of them do not loose any information while others accept some loss of information to achieve more compact sets. Closed Itemsets (Pasquier et al., 1999) and Free Itemsets (Boulicaut et al., 2000) are two examples of condensed representations without loss. Mining frequent closed itemsets reveals exactly the same information as mining all frequent itemsets, because it is possible to retrieve precisely the frequency of all frequent itemsets from the frequent closed itemsets (Pasquier et al., 1999). The same is valid for mining frequent free itemsets. Before defining Closed Itemsets and Free Itemsets it is required to understand the closure of itemsets concept:

Definition 7 (closure of itemsets): The *closure* of an *itemset I* over a set of transactions *T*, where $T \subseteq D$, (denoted by $closure_T(I)$) is the maximal (for set inclusion) superset of *I* which has the same support as *I*. In other terms, the closure of *I* is the set of items that are common to all transactions in *T* that support *I*. ■

Definition 8 (closed itemsets): A *closed itemset* is an itemset that is equal to its *closure* in *T*, formally $closure_T(I) = I$. In other terms, closed itemsets are maximal sets of items that are supported by a multiset of transactions. ■

Free itemsets are sets of items that are not "strongly" correlated (Boulicaut et al., 2000). They have been designed as a useful intermediate representation for computing closed itemsets since the closed itemsets are the closures of the free itemsets.

Definition 9 (free itemsets): An *itemset I* is *free* if no logical rule holds between its items, i.e., it does not exist two distinct itemsets X, Y such that $I = X \cup Y$, $Y \neq \emptyset$ and $X \rightarrow Y$ is always true (logical rule). ■

Free itemsets can also be defined as the itemsets that are not included in any closure of their proper sub-set, in other words, all the proper subsets of a free itemset I have a different frequency than I.

Aiming at obtaining even more compact sets some condensed representations allow for some loss of information. The δ-free itemsets presented in (Boulicaut et al., 2000) is one example of such a representation. We present in the following its definition.

Definition 10 (δ-closure): Let δ be an integer and I an itemset. The δ-closure of I over a set of transactions T, where $T \subseteq D$, δ-closure$_T(I)$ is the maximal (for set inclusion) superset Y of I such that for every item $A \in Y - I$, $F_T(I \cup \{A\})$ is at least $F_T(I) - \delta$. In other terms, $F_T(\delta\text{-closure}_T(I))$ has almost the same value than $F_T(I)$ when δ is small w.r.t. the number of transactions. ■

Definition 11 (δ-free itemsets): Let δ be an integer and I an itemset, an itemset I is δ-free if no association rule with at most δ exceptions holds between its subsets. δ-free itemsets satisfy the constraint $(\forall I' \subset I) \rightarrow I \not\subset \delta\text{-closure}_T(I')$. ■

The use of condensed representations can decrease considerably the storage requirements for frequent itemsets. Moreover, mining a condensed representation instead of all frequent itemsets can also reduce the time required for the mining task.

2.1.4 Patterns based on Frequent Itemsets

It is possible to derive other kinds of patterns using the frequent itemsets holding on some transactional dataset. The most widespread use of frequent itemsets is to support association rule mining (Agrawal et al., 1993). Boolean rules and classification rules can also be obtained through frequent itemsets. The term associative classification (Liu et al., 1998) is used in the literature to refer to the mining task of deriving classification rules using frequent itemsets. Generalized rule mining (Mannila and Toivonen, 1996) refers to the task of mining Boolean rules. These three patterns will be used in chapter 5 to exemplify the derivation of patterns based on frequent itemsets retrieved from the data warehouse. We introduce in the following the definitions of association rules, classification rules and Boolean rules.

Definition 12 (association rules): An association rule is denoted as $\alpha \rightarrow \gamma$ where α and γ are itemsets and $\alpha \cap \gamma = \varnothing$. α is the body of the rule, also called antecedent, and γ is the head of the rule, also called consequent. Semantically, an association rule expresses a tendency of having γ whenever α is present. ■

Consider a classification task with $C_{1,\ldots,}C_k$ as the k classes values:

Definition 13 (classification rules): A classification rule is a rule that concludes on one class value (i.e., C_i). It is denoted by $\alpha \rightarrow C_i$, where $1 \leq i \leq k$. ■

Definition 14 (Boolean formula): Given *items* a finite set of symbols of 0/1-valued attributes, a *Boolean formula* over *items* is a formula ϖ built from the atomic formulae $A = 0$ (false) and $A = 1$ (true), where $A \in$ *items* using the connectives $\wedge$ (and), $\vee$ (or), $\neg$ (not), and parenthesis. ■

Definition 15 (Boolean rules): A *Boolean rule* over R is an expression of the form $\varpi \rightarrow \psi$, where ϖ and ψ are Boolean formulae. ■

Definition 16 (frequency of rules): A transaction *t* supports a *rule* $\alpha \rightarrow \gamma$ if α and γ are valid in *t*. The *frequency* of $\alpha \rightarrow \gamma$ over a set of transactions *T*, where $T \subseteq D$, is the number of transactions in *T* supporting $\alpha \rightarrow \gamma$ and is denoted by $F_T(\alpha \rightarrow \gamma)$. ■

Definition 17 (support of rules): Given $|T|$ the number of transactions in *T*, where $T \subseteq D$, the *support* of a *rule* $\alpha \rightarrow \gamma$ over *T* is the *frequency* of $\alpha \rightarrow \gamma$ divided by $|T|$ and is denoted by $S_T(\alpha \rightarrow \gamma)$:

$$S_T(\alpha \rightarrow \gamma) = \frac{F_T(\alpha \rightarrow \gamma)}{|T|} \quad (2)$$

the support of a rule is a rate denoting the percentage of transactions in *T* supporting $\alpha \rightarrow \gamma$. ■

Definition 18 (confidence of rules): The confidence of a *rule* $\alpha \rightarrow \gamma$ represents the probability of having the consequent (γ) in a transaction given that the antecedent (α) is present. Given a set of transactions *T*, where $T \subseteq D$, it can be computed as follows:

$$\text{conf}_T(\alpha \rightarrow \gamma) = \frac{F_T(\alpha \cup \gamma)}{F_T(\alpha)} \quad (3) \blacksquare$$

Other examples where frequent itemsets are useful comprise iceberg cube computation (Beyer and Ramakrishnan, 1999) and frequent pattern-based clustering (Wang et al., 2002).

2.2 Data Warehouse

According to Inmon (2002) a data warehouse is a subject-oriented, integrated, time variant, non-volatile collection of data in support of management's decision-making process. More specifically:

- Subject-oriented: the data is organized by subject or entity (e.g. customer) and not by application;
- Integrated: the data is held in a consistent form, combining information from different sources;
- Time-variant: the data is not only current (as it is on an operational database) but is historic and time-stamped;
- Non-volatile: the data, once stored in the warehouse, does not change and is not updated as it would be if held on an operational database.

Data warehouses contain a wide variety of data that present a coherent picture of business conditions at a single point in time. Development of a data warehouse includes development of systems to extract data from operating systems plus installation of a warehouse database system that provides managers flexible access to the data. The data entering the data warehouse comes from the operational environment in almost every case. The data warehouse is always a physically separate repository of data transformed from the application data found in the operational environment. The major goal is to provide valuable information for analysis in an easy accessible and understandable way.

At this high level of abstraction the two gurus of the date warehouse field, Bill Inmon and Ralph Kimball, agree. Although, they follow different paradigms that can be summarized as:

- **Bill Inmon's paradigm:** Data warehouse is one part of the overall business intelligence system. An enterprise has one data warehouse, and data marts source their information from the data warehouse. In the data warehouse, information is stored in 3rd normal form.
- **Ralph Kimball's paradigm:** Data warehouse is the conglomerate of all data marts within the enterprise. Information is always stored in the dimensional model.

There is no right or wrong between these two ideas, as they represent different data warehousing philosophies. In reality, the data warehouses in most enterprises are closer to Ralph Kimball's idea. This is because most data warehouses started out as a departmental effort, and hence they originated as a data mart. Only when more data marts are built later do they evolve into a data warehouse. In this thesis we are also

closer to Ralph Kimball's view. The simplicity of the dimensional model and its easy understandable representation were the key reasons of our choice.

In the following we present some concepts of the data warehouse field serving as background information for our work. The information summarized here was extracted from Kimball and Ross (2002).

2.2.1 Terminology

The term *fact* is used to represent some measurement of the business. For example, when registering the quantity of each product sold for each day in each store we are measuring a fact related to the amount of sales. The total of sales in dollar for each product, day and store combination is another example. A measurement is taken on the intersection point of all *dimensions* (e.g. day, product and store). Each *dimension* describes some feature of the related facts. The list of dimensions defines the *granule* and tells us the scope of the measurement. The dimensions constitute the entry points for the facts implementing the data warehouse user interface.

2.2.2 Dimensional Modeling

Dimensional modeling is known as the technique of designing data warehouse structures to consist of a series of logical data-marts each consisting of a fact table and a number of conformed dimension tables in a star schema. Figure 1 presents an example of a star schema produced by the dimensional modeling.

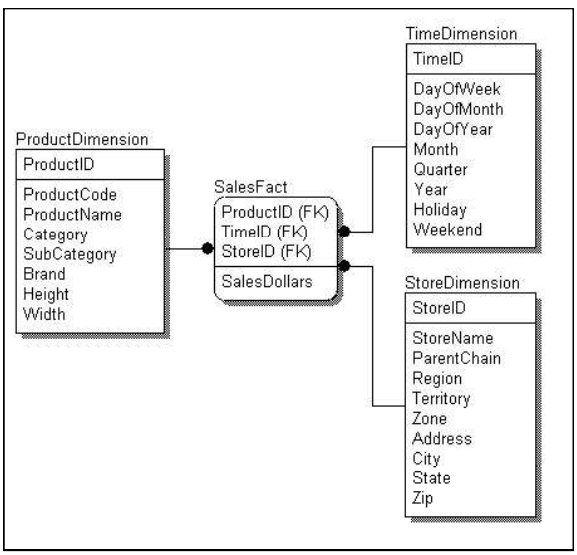


Figure 1 – Dimensional Model (Star Schema)

Kimball and Ross (2002) define a process with 4 steps for creating a dimensional model. The 4 steps are namely: business process selection; granule definition; choice of dimensions; and facts identification.

- Business process selection: select the business process that will be modeled. A business process is some natural business activity, which is normally supported by some information system collecting data;
- Granule definition: define the level of detail of the selected business process. The granule definition means to specify precisely what a tuple in the fact table represents;
- Choice of dimensions: choose the dimensions that are applicable to each tuple in the fact table. The main question is "How business people describe the data resulting from this business process?";
- Facts identification: identify the facts that will be stored in the fact table. The facts are identified answering the question "What are we measuring?".

2.2.3 Components

Figure 2 presents the interaction between the 5 major components of a data warehouse environment: sources, staging area, presentation area, business intelligence tools and metadata components.

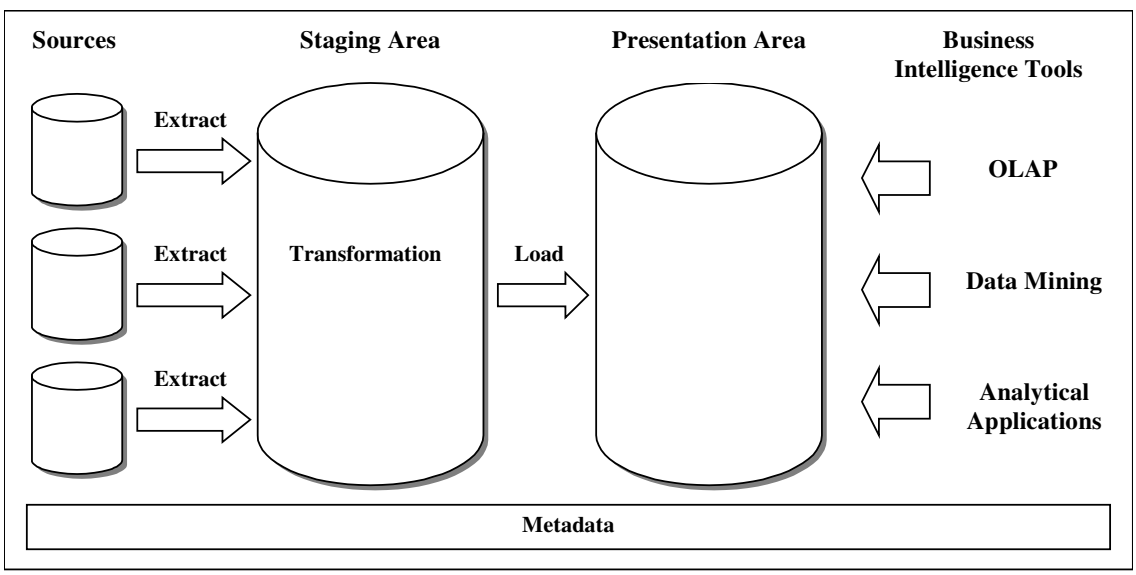


Figure 2 – Data Warehouse Environment Components

A brief description of each one of the 5 major components is provided in the following.

- Sources: the sources of a data warehouse are normally the systems in the operational environment. These systems capture and register the enterprise transactions. The source systems must be considered as an external component of the data warehouse because normally the data warehouse team has only some or even no control over the operational environment.
- Staging Area: the data staging area of the data warehouse is a storage area and a set of processes known as ETL processes (Extraction-Transformation-Load). Everything between the operational source systems and the presentation area belongs to the data staging area.
- Presentation Area: the data presentation area is the place where the data is organized, stored and accessible to the data warehouse users.
- Business Intelligence Tools: this component comprises all tools that aid the user in analysing the data accessible in the presentation area. Online Analytical Processing Tools (OLAP), data mining algorithms and specialised analytical applications are some examples.
- Metadata Component: the metadata component is responsible for maintaining and managing every metadata related to the data warehouse environment. Technical information about the ETL processes and descriptions on the data published in the presentation area are two examples.

2.3 Related Work

Facing the problems presented in section 1.2 in chapter 1 the challenge is to provide a standard modeling for frequent itemsets and flexible pattern retrieval capabilities without requiring the source data to be available during the analysis phase. Moreover, it must be feasible to cope with tight time constraints imposed by the data stream scenario. An overview of related works is presented in this section and major problems related to the issues we are addressing are highlighted.

The OLAP Mining approach (Han, 1997) integrates on-line analytical processing (OLAP) with data mining. It provides an interesting solution for analyzing the results of data mining tasks. OLAP Mining assumes that we are able to express a data mining task execution result as a data cube in order for such integration to take place. This is straightforward for some data mining tasks, such as classification and clustering, by considering the classes or clusters as instances of a new dimension. When it comes to frequent itemset mining, we cannot easily represent the results as a data cube (Han, 1997). This approach also needs to access the original data during the analysis phase when a user requests the frequent itemsets holding on a set of transactions.

Regarding pattern storage and pattern management, Kimball and Ross (2002) presented an approach for market basket analysis storing the frequency counts in a fact table. Frequency counts for every pair of items that appear together in any transaction are stored. The number of possible combinations grows exponentially with the number of items. The information about all combinations is likely to require more storage than the original transactions, which is infeasible for data streams. Furthermore, it only handles itemsets with two items thus avoiding, for example, to derive association rules involving three or more items. Inductive Databases introduced by Imielinski and Mannila (1996) are databases that contain inductive generalizations about the data, which are called the database theory. In such a database the user can simply query the database theory using a conventional query language. However, the database theory relates to the data stored in the database and, therefore, we need the original data during the analysis phase. Also, it does not provide a standardized logical view for frequent itemsets representation. The PANDA project (PANDA, 2004) studies current state-of-the-art in pattern management and explores novel theoretical and practical aspects of a Pattern Base Management System. It deals with patterns in a broad sense, considering no predefined pattern types. The main focus lies in devising a general and extensible model for patterns. The definition of a standardized logical view for frequent itemsets representation and specific calendar-based pattern mining issues are not considered in this project.

The focus in data stream mining has been on stream data classification and stream clustering. Only recently, mining frequent counts in streams gained attention. An algorithm to find frequent items using a variant of classic majority algorithm was developed simultaneously by Demaine et al. (2002) and Karp et al. (2003). A

framework to compute frequent items and itemsets was provided in Manku and Motwani (2002). The presented algorithm, called Lossy Counting, mines frequent patterns in data streams by assuming that patterns are measured from the start of the stream up to the current moment. Lossy Counting considers always the whole stream and does not provide flexibility to mine the frequent itemsets holding on a subset of the stream. Closer to our focus, the work presented in Giannella et al. (2003) proposes a new model for mining frequent patterns from data streams, the FP-Stream. This model is capable of answering user queries considering multiple time granularities. A fine granularity is important for recent changes whereas a coarse granularity is adequate for long-term changes. FP-Stream supports this kind of analysis by a tilted-time window, which keeps storage requirements very low but prevents calendar-based pattern analysis. An example of a tilted-time window (in minutes) is 15, 15, 30, 60, 120, 240, 480, etc. It is possible to answer queries about the last 15 minutes or the last 4 hours (15+15+30+60+120 minutes), but it is not possible to answer queries about *last Friday* or *every morning*, for example. The lack of a uniform partitioning prevents calendar-based pattern analysis.

In order to discover temporal association rules, the work in Özden et al. (1998) states very clearly the problem of omitting the time dimension. It is assumed that the transactions in the database are timestamped and a time interval is specified by the user to divide the data into disjoint segments, such as months, weeks, days, etc. Cyclic association rules are defined as association rules with a minimum confidence and support at specific regular time intervals. A disadvantage of the cyclic approach is that it does not deal with multiple granularities of time intervals. Also, an example of a calendar pattern that cannot be represented by cycles is the simple concept of the first working day of every month. Ramaswamy et al. (1998) introduces the notion of calendar algebra, which basically defines a set of time intervals. Each time interval is a set of time units (e.g. days). A rule is called calendric if it has the minimum support and confidence over every time unit in the calendar. A disadvantage of this approach is that the user must have prior knowledge about the temporal patterns in the transactional database to define the calendars. The work in Li et al. (2001) tries to overcome this problem by mining the calendars. Calendar schemas were proposed as a semantically rich representation of time intervals and used to mine temporal frequent itemsets by Li et al. (2001) and Ramaswamy et al. (1998). Such schemas are used to specify the search

space for different possible calendars. An example of a calendar schema is (year, month, day, day_period), which defines a set of calendar patterns, such as *every morning of January of 1999* (1999, January, *, morning) or *every 16th day of January of every year* (*, January, 16, *). The rules mined by Li et al. (2001) are presented together with their mined calendars. All these approaches perform a calendar-based frequent itemset mining step. However, they require the transactions to be available during the calendar-based mining task. The calendars being analyzed are only defined when retrieving the patterns and thus the transactions belonging to the specified calendars must be accessible during the pattern retrieval. This feature disables these approaches from being applied to data streams because it is not possible to store the transactions for future analysis.

Partitioning methods for parallel frequent itemset mining (Zaki, 1999) aim at avoiding performance problems when the data structures involved become too large for main memory (Ahmed et al., 2004). During the parallel frequent itemset mining task, information about candidate frequent itemsets must be exchanged between the partitions in order to compute the exact frequency count. It is important to understand the distinction from this research area to the partitioning applied in our approach. The partitioning criteria applied in DWFIST are guided by analysis requirements instead of performance issues. Moreover, in the DWFIST approach there is no need for information exchange between different partitions during the mining task. The itemsets are considered frequent or not w.r.t. the transactions belonging to one partition.

None of the above mentioned works provides a standard way for organizing and retrieving frequent itemsets. Also, no previous approach is capable of performing calendar-based pattern mining on data streams.

3 The DWFIST Approach

The acronym DWFIST stands for Data Warehouse of Frequent ItemSets Tactics (Monteiro et al., 2005b). The Oxford Dictionary and Thesaurus defines Tactics as: skilful device; scheme, strategy. In this sense, the DWFIST approach aims at supporting the analysis and exploration of frequent itemsets and derived patterns, e.g. association rules, in transactional datasets.

The research field of Data Warehouse has been extremely successful in providing efficient and effective ways to analyze huge amounts of data. Although data warehouse techniques cannot be directly applied to store and analyze frequent itemset patterns, they can be adapted while keeping their principles and best practices learned over the years. Storing frequent itemsets may sound strange at first as one could say that it is better to store the original data and obtain the frequent patterns upon ad-hoc request. At this point, we would like to draw the readers' attention to some relevant remarks. First, it is not always possible to store the original data. It is infeasible to store the whole stream data and at the same time it is still extremely interesting to analyze different kinds of patterns holding on the stream, as for example, calendar-based patterns. Second, by pre-processing the frequent itemsets we are able to reduce the computational effort required during the analysis tasks performed by a user. Finally, it is also possible to store the frequent itemsets in addition to the original data. The redundant information can be justified as far as it publishes the data in such a way that can be more effectively analyzed (Kimball and Ross, 2002).

This chapter presents the major requirements for the DWFIST approach in section 3.1. An overview of the components of our approach is presented in section 3.2. Section 3.3 introduces one example that will be used throughout the thesis to illustrate our ideas.

3.1 Requirements

The information stored in the Data Warehouse of Frequent Itemsets must be sufficient for answering queries requesting pattern holding on some subset of the original transactions. In other words, it must be possible to specify constraints on features of the original transactions when retrieving patterns. For each itemset retrieved

from the data warehouse it must be possible to retrieve its frequency count over the subset of original transactions being queried. Whenever the exact frequency is not known, it must be possible to provide an approximate answer. Moreover, it must be possible to guarantee that the result of the query is complete, i.e., that there is no pattern holding on the queried subset of original transactions that is not retrieved when querying the data warehouse. Figure 3 sketches the retrieval of patterns from the data warehouse of frequent itemsets.

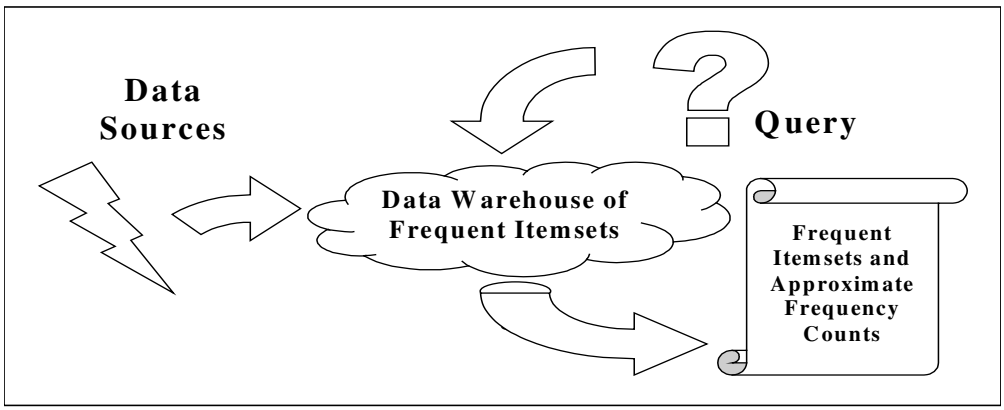


Figure 3 – Sketch of the pattern retrieval from the Data Warehouse of Frequent Itemsets

Furthermore it must be possible to use the frequent itemsets and their approximate frequency counts to derive other patterns (e.g. association rules) and interestingness measures associated to such patterns (e.g. confidence of association rules).

The logical schema of the data warehouse of frequent itemsets must be defined in order to provide a standard way of accessing and retrieving frequent itemsets, making it possible to develop tools that can be reused in different scenarios.

Another important requirement refers to the feasibility of coping with the tight time constraints imposed by data streams. We want to store in a data warehouse the frequent itemsets holding on different disjoint sets (partitions) of transactions. As far as we are able to compute the frequent itemsets holding on each partition before the next one is ready to be processed we are coping with data stream requirements. On the other hand we have the load process of the data warehouse. The periodicity of the DW load process must be completely independent from the time granularity and should be set to meet analysis requirements and availability.

3.2 Components

The major components of the DWFIST approach and their relationships are presented in Figure 4. The *Pre-processing and Loading step* is composed of three major tasks: gather the transactions into disjoint sets (partitions) according to a pre-defined criteria (data warehouse granule); mine the frequent itemsets holding on a partition using a pre-defined mining minimum support; and load the mined frequent itemsets into the data warehouse.

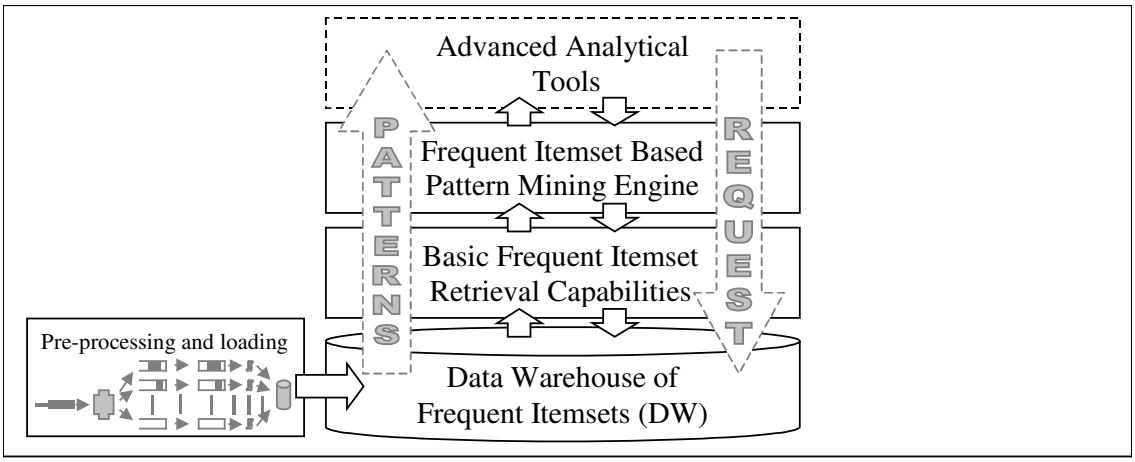


Figure 4 – Components of the DWFIST approach

The *Data Warehouse of Frequent Itemsets*, referenced simply as DW, is the main component of the approach. Its task is to store and organize the frequent itemsets into partitions. A standard modeling provides a standardized logical view.

The role of the *Basic Frequent Itemset Retrieval Capabilities* component is to retrieve a set of frequent itemsets with approximate support guarantees given some query specifying the portion of the data warehouse to be considered. An example of such a query is: "Retrieve the frequent itemsets considering only the morning period".

The *Frequent Itemset Based Pattern Mining Engine* generates patterns that can be obtained from frequent itemsets. It uses a set of frequent itemsets with approximate support guarantees as input. These itemsets are provided by the Basic Frequent Itemset Retrieval Capabilities component. Association Rules (Agrawal et al., 1993) and Boolean Rules (Mannila and Toivonen, 1996) are two examples of patterns that can be derived by this engine. Obviously, the frequent itemsets themselves are also patterns that can be "derived". This component is also responsible for computing some

approximate interestingness measure guarantees related to the derived patterns, e.g., approximate confidence guarantee for association rules.

The *Advanced Analytical Tools* component comprises analytical and explorative tools that can be built on top of the other components. It is possible to build, for example, a tool that provides a cube view of the patterns, similar to what an OLAP tool provides to conventional dimensional data. Existing automatic exploration methods, as the one presented in Sathe and Sarawagi (2001), can be adapted to explore the patterns and also new ones can be developed.

Chapters 4 and 5 depict the core of the DWFIST approach, from the Pre-processing and Loading step to the Frequent Itemset Based Pattern Mining Engine. The Advanced Analytical Tools component corresponds to any tool that can be built using the functionalities provided by the other modules. Specifying such tools is not the goal of this thesis. Despite of this, we briefly motivate and discuss some ideas in Appendix A.

In order to better illustrate and explain the concepts we introduce an example that will be used throughout the remainder of the thesis.

3.3 Running Example

We use the classical example of market basket data to illustrate the concepts of our approach. In market basket data the transactions are sets of products that are bought by customers. As we are interested in discussing issues related to data streams, we will consider that customer's transactions arrive at an average rate of 100 transactions per second. This example was simulated in our prototype and details about this simulation are provided in chapter 6.

Given the above scenario we want to retrieve calendar-based patterns holding on such a data stream. In our example the minimum slot of time that can be analyzed corresponds to a period of 1 hour. This corresponds to the Data Warehouse granule that will be further discussed in the next chapter. Due to DW properties, any set of 1-hour periods can be freely combined to compose a calendar partition. We may be interested, for example, in retrieving association rules valid in the morning period on weekdays. In this case, the 1-hour periods, namely [08:00AM, 09:00AM), [09:00AM, 10:00AM),

[10:00AM, 11:00AM) and [11:00AM, 12:00PM), belonging to weekdays will be combined.

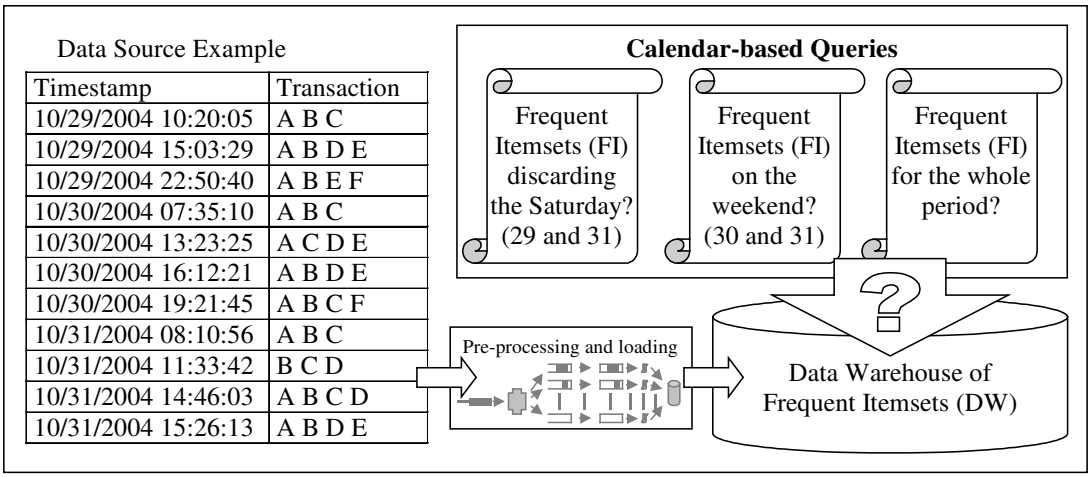


Figure 5 – Data source and calendar-based query examples

A simplified data source example is shown in Figure 5. It also presents other calendar-based queries and sketches the retrieval of calendar-based frequent itemsets in our approach. The Data Warehouse of Frequent Itemsets receives and stores information from a transactional data source. The stored information can be used to provide approximate answers to calendar-based queries. The queries presented in Figure 5 are meant only to illustrate the calendar-based pattern retrieval. Any calendar partition that can be built using 1-hour periods (e.g. afternoon, weekends, holidays, first quarter of 2004, second semester, etc.) represents a calendar-based query that can be answered by our running example.

4 Data Warehouse of Frequent Itemsets

The Data Warehouse of Frequent Itemsets (Monteiro et al., 2005a), although presenting particularities, can be seen as a regular Data Warehouse. Therefore, many considerations presented in Kimball and Ross (2002), ranging from the use of surrogate keys to the Business Dimensional Lifecycle, are applicable. We will keep our focus on the particular features. This chapter starts explaining how the transactional data is pre-processed in the staging area. Storage requirements and time constraint issues are discussed, specially for data streams. Afterwards, conceptual and standardized logical schemas for the DW are presented. A data warehouse of frequent itemsets example is provided in the sequence. At last, the basic properties of a data warehouse of frequent itemsets are presented.

4.1 Staging Area Issues

A general view of the staging area is presented in Figure 6. It presents three main tasks: separate and accumulate transactions in different partitions; mine the frequent itemsets holding on each individual partition; and load the frequent itemsets into the data warehouse.

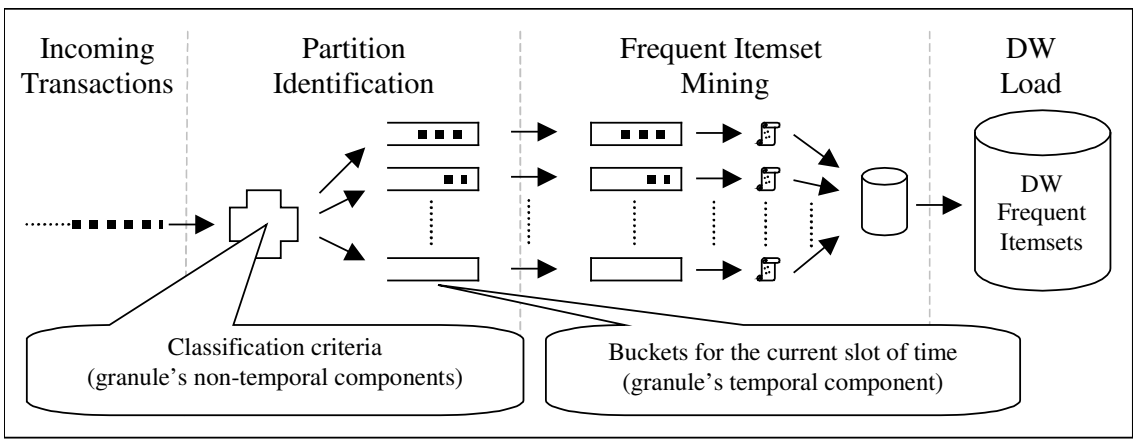


Figure 6 – Overview of the Data Staging Area

First of all, one important design decision must be taken, namely the granule definition. The granule of the DW must define a partition on the transactional data, separating it into disjoint sets. One well-known rule for conventional data warehouses says to store the data at the most granular level, in other words, it says to define the granule at the most detailed level. However, this rule must sometimes be broken in our

approach. This is due to an important tradeoff that has to be taken into account when defining the granule. A more detailed granule provides more flexibility for analysis. However, the more detailed the granule of the DW, the more frequency counts need to be stored increasing the storage requirements. Hence, analysis requirements must be considered to define the granule and storage requirement tests have to be carried out to check the feasibility of the choice. In our running example we define the granule as the frequent itemsets per hour. Examples of partitions defined by such granule are "10/29/2005 [08:00AM, 09:00AM)", "10/29/2005 [09:00AM, 10:00AM)", "10/29/2005 [10:00AM, 11:00AM)", and so on. Defining such a granule avoids analyzing the pattern behavior for periods shorter than one hour. On the other hand, all possible combinations of 1-hour partitions can be analyzed, providing sufficient flexibility for important calendar-based analysis. For stream data sources the chosen granule must also be verified against imposed time constraints. Later we will discuss these time constraint issues as well as the storage requirements in more details.

As in any data warehouse, the temporal information plays an important role in the Data Warehouse of Frequent Itemsets. The granule of the DW must have a temporal component. Even for applications where the temporal feature is not important for analysis, a temporal component must be part of the granule definition at least in order to separate different DW loads. Other components, like spatial components, can additionally be used to define the granule. As an example, it could be defined as the frequent itemsets per store per hour.

Once the granule of the DW is defined, the three tasks presented in Figure 6 can be implemented. The first one is to separate and accumulate transactions in different partitions. When the granule has non-temporal components, such as store, a new incoming transaction has to be analyzed in order to define the partition it belongs to. Each partition for the current slot of time must have one corresponding bucket to accumulate its transactions. For example, the granule "per store per hour" requires one bucket for each store. If the granule is defined by a temporal component only, as in our running example, then one bucket will be sufficient.

The temporal component of the granule is used to identify when the set of transactions pertaining to the partitions of the current slot of time are completely collected. The complete sets of transactions are passed to the next step where the mining is performed.

The frequent itemset mining is performed on each completely collected set of transactions comprising one partition. A minimum support threshold must be specified for this task, which we call *mining minimum support*. The initial threshold value is not a definitive commitment. The mining minimum support threshold can be changed freely over time as well as customized thresholds can be specified for different partitions. It is only required that each partition is associated with exactly one mining minimum support threshold. The frequent itemset mining results in a list of frequent itemsets with their corresponding frequency counts holding on the transactions of a partition. The mining minimum support is set to 0.1% in the running example.

The frequent itemsets, their frequency counts, the number of transactions in the partition and the applied mining minimum support threshold must be stored in some intermediate storage area before they are finally loaded into the DW. This intermediate storage area provides isolation between the mining process and the DW load process. By this isolation, it is possible, for example, to define a daily DW load procedure having an hourly granule. As soon as one partition is mined and its information stored in the intermediate storage area, the partition's transactions can be discarded.

Finally, the information stored in the intermediate storage area will be loaded into the data warehouse periodically. Many conventional data warehouse issues arise in this step and should be treated the same way as in regular data warehouses. Some examples of such issues are assigning surrogate keys and creating new dimension instances. The isolation provided by the intermediate storage area makes it easier to take care of such issues while still having to cope with tight time constraints.

4.1.1 Time Constraint Issues

As far as we are able to pre-process and load the data of one slot of time before the next one is available we are coping with data stream time constraint requirements. Let us discuss the related issues through the running example. The granule was defined as the frequent itemsets per hour. Hence, each slot of time covers data for one hour and thus a one-hour window is available for pre-processing and load. While the transactions of the current slot of time are being gathered, the previous one is being mined and loaded. Let us discuss the mining and loading steps in more detail.

Many frequent itemset mining algorithms were proposed since the statement of this mining task in Agrawal et al. (1993). Any efficient algorithm can be applied in our mining pre-processing step. As an example of an efficient algorithm, Giannella et al. (2003) uses the FP-Growth algorithm (Han et al., 2000) to mine frequent itemsets on batches of transactions from a data stream. The efficiency of this step depends mainly on the mining minimum support and the number of transactions. In our running example, a mining minimum support of 0.1% was used and the average number of transactions per partition was 360000. We used an optimized implementation of FP-Growth described in Grahne and Zhu (2003). Every partition was mined in less than one minute.

When a non-temporal component is used to define the DW granule, the different partitions of one slot of time can be processed in parallel. It is important to note that we are not talking about parallel mining. We just have to identify the frequent itemsets for each partition individually. For this purpose, a non-parallel frequent itemset algorithm could be executed on different partitions at the same time. In parallel data mining an additional task would be to combine the frequency counts of all partitions and compute global frequent itemsets from them. As we want to keep frequent itemsets for each partition in the DW, this is not necessary in our approach.

The load step mainly comprises the insertion of new frequent itemset frequency counts and the creation of new dimension tuples. An important property of the load step is that the time required is proportional to the number of frequent itemsets to be loaded instead of the number of transactions. Also, an increase in the number of transactions for a frequent itemset mining task keeping a fixed mining minimum support does not lead to a proportional increase in the number of frequent itemsets. In most cases, the number of frequent itemsets tends to stabilize or even decrease. The partitions in the running example contain 3650 frequent itemsets (using a mining minimum support of 0.1%) on average. The average time required for the load step was 12 minutes using a non-optimized procedure.

The definition of the granule plays a central role for coping with time constraints. First of all, it defines the time window available for processing. Second, an increase in the time window makes the task of the load step easier and the task of the mining step harder. The task of the mining step gets harder because a bigger window means a potentially higher number of transactions. Fortunately, current frequent itemset

mining algorithms are able to process efficiently a considerable large number of transactions. Therefore, an increase in the time window is likely to bring more benefits for the load step than losses for the mining step.

Some scenarios may present prohibitive arrival rates even for the most efficient existing frequent itemset mining algorithm. Such fast arrival rates result in large partitions containing a large number of transactions that increases the time required for the mining task. An alternative approach for extreme scenarios is to apply the framework presented in Giannella et al. (2003), called FP-Stream, to perform our mining task. This framework is capable of answering queries such as "retrieve the frequent itemsets holding on the transactions of the last 15 minutes" or "retrieve the frequent itemsets holding on the transactions of the last 4 hours" on a stream. The important feature of this framework is that the required processing time is independent from the period being queried. This means that using this framework it is possible to increase the time window without increasing the time required for the mining step. The disadvantage of applying this framework to perform our mining task is that it introduces an error in the frequency counts. Therefore, the known frequency counts that will be loaded into the DW are not exact counts anymore. This additional error must then be considered when computing the frequency upper bound during the retrieval from the DW.

The conclusion about adjusting the temporal component of the DW granule is that it provides an effective way of coping with time constraints. Although, we cannot forget that analysis requirements must be taken into account as well, and that a more detailed temporal component provides more flexible analysis capabilities. This tradeoff must be evaluated through experimental tests to define the optimal granule for each specific scenario.

The isolation provided by the intermediate storage area helps to cope with tight time constraints as well. Using a daily load procedure, for example, it is not strictly required that the load of each one of the 24 1-hour partitions must be performed in less than one hour. It is sufficient that the set of 24 partitions can be loaded in less than 24 hours. In this way, eventual peeks of processing in the load step, caused for example by an extension of the DW storage area, can be handled. In the running example, even the peeks of processing in load step could be performed in less than one hour.

4.1.2 Storage Requirement Issues

An important observation regarding storage requirements is that the size of DW increases in a lower rate compared to the size of the considered input data stream. The reasoning that supports this assertion is two-fold. First, the frequent itemsets require less storage space than the original stream transactions. Second, the reuse of information stored in the dimensions reduces the storage requirements. The DW in the running example occupies 10MB related to the first 1GB of the data stream. It reaches 17.5MB when considering the whole test dataset of 2GB. The mentioned DW sizes include indexes structures as well.

Non-stream data sources present a less critical scenario. Nevertheless, storage requirement tests must also be carried out. In such data sources it can be interesting to build a data warehouse of frequent itemsets even if you store the original transactions. First, by pre-processing the frequent itemsets we are able to reduce the computational effort required during the analysis tasks performed by a user. Second, the redundant information can be justified as far as it provides the data in such a way that it can be more effectively analyzed (Kimball and Ross, 2002).

Nevertheless, it is important to make a remark at this point. Dense correlated databases may produce a large number of frequent itemsets. An extensive amount of work has been done on condensed representations of frequent itemsets aiming to represent a set of frequent itemsets in a compact way. The DWFIST approach allows for the use of condensed representations to describe the frequent itemsets holding on each partition. The mining and loading steps will also benefit from the use of condensed representations. However, the overhead on the pattern retrieval step must be taken into account. This additional processing is required to discover the frequency of itemsets that are not explicitly expressed in the applied representation.

A similar discussion on adjusting the DW granule presented in the previous section applies to storage requirements issues. In an analogous way, the storage requirements are proportional to the number of frequent itemsets being stored.

4.2 Conceptual Design

We present a standardized conceptual schema for data warehouses of frequent itemsets. Figure 7 and Figure 8 show this schema using the StarER notation (Tryfona et

al., 1999). Our conceptual schema is based on a distinction between item dimensions and the partition dimensions:

- An item dimension describes similar items that can appear as part of a frequent itemset. If there are groups of items with different features, several item dimensions should be provided in order to better describe and classify the items. In the context of medical procedures, the hospital materials could be represented in one item dimension and the medical staff in another one.
- A partition dimension organizes the space of the original transactions into disjoint sets, which we call DW partitions. Each partition dimension describes one component of the DW granule. Temporal and spatial dimensions are some examples of candidate partition dimensions.

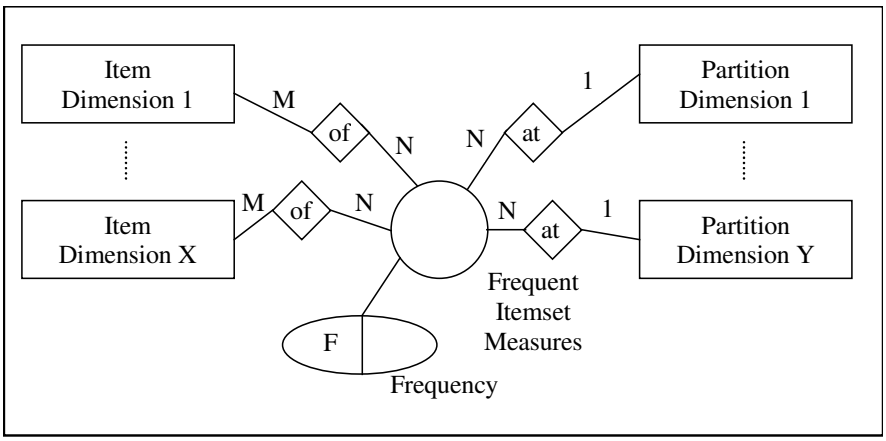


Figure 7 – Frequent itemset measures standardized conceptual schema

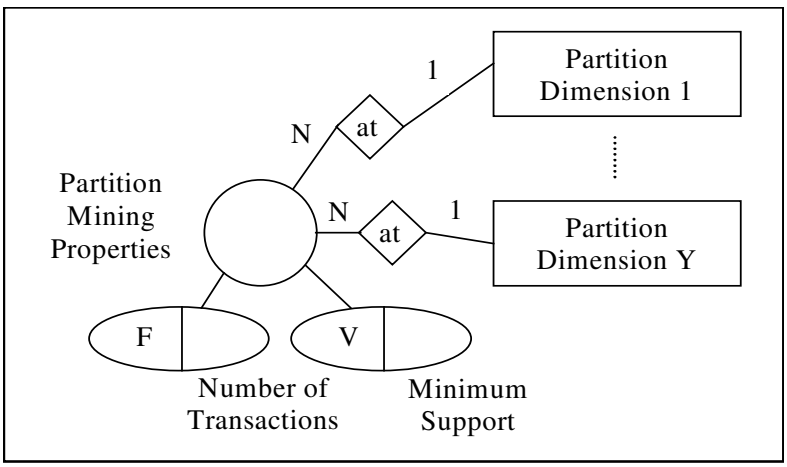


Figure 8 – Partition mining properties standardized conceptual schema

Another distinction that can be observed in Figure 7 is that the relationship between an item dimension and an itemset fact is an N to M relationship, while the relationship between a partition dimension and an itemset fact is a 1 to N relationship. The N to M relationship is applied because one itemset may contain different items of the same dimension.

When accessing the DW, the item dimensions are used to define features of the itemsets that must be retrieved. In the running example, it could be requested to retrieve only the itemsets containing at least one product belonging to the category “beverages”. The partition dimensions are used to define the portion of the original transactions that must be considered. Retrieving the frequent itemsets holding on the transactions related to the morning period would be one possibility in the running example.

It is interesting to relate the information stored in the intermediate storage area with our conceptual schema. Figure 7 represents the part of the DW that describes the frequent itemsets and their frequency counts. A frequency count is a fact attribute and a frequent itemset is represented by relationships with item dimensions. As shown in Figure 8, for each partition additional information has to be stored, i.e. the number of transactions and the applied mining minimum support threshold. In the figure they are represented as fact attributes. Note, that the schema presented in Figure 8 has only partition dimensions because the partition mining properties are the same for a complete partition, thus independent from a specific frequent itemset.

The dimensions should conform to the Data Warehouse Bus Architecture (Kimball and Ross, 2002). This allows for cross over operations that relate the attributes of the two different facts presented in Figure 7 and Figure 8 through the corresponding partition dimensions.

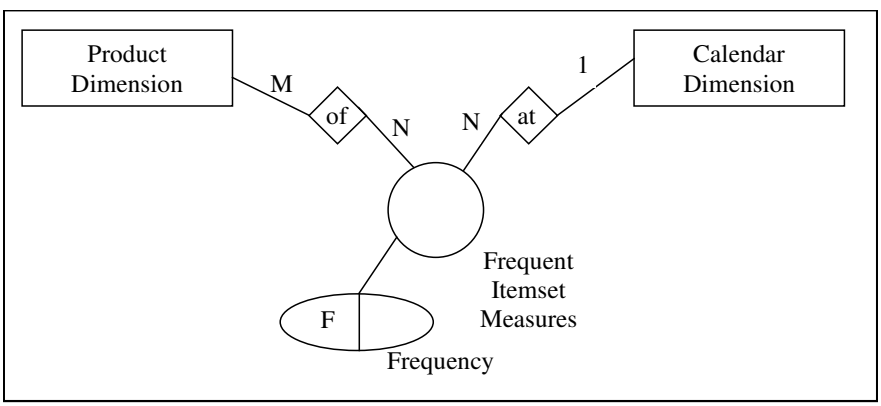


Figure 9 – Frequent itemset measures conceptual schema for the running example

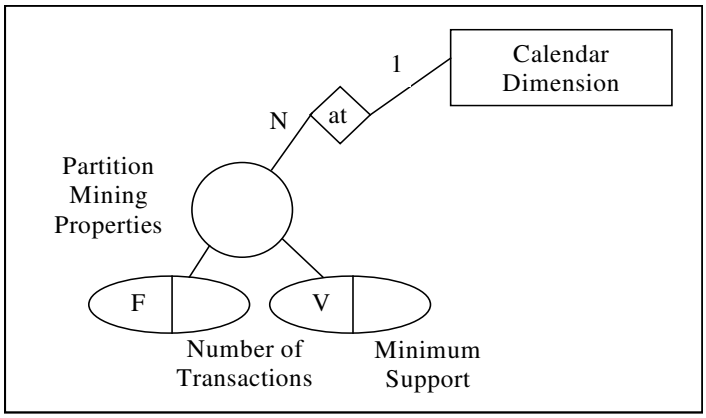


Figure 10 – Partition mining properties conceptual schema for the running example

Figure 9 and Figure 10 present the conceptual schema for our running example. A calendar dimension plays the role of a partition dimension organizing the space of the original transactions into disjoint sets representing non-overlapping 1-hour periods. The schema in Figure 10 represents the number of transactions on each 1-hour period and the minimum support used for mining the frequent itemsets in the pre-processing step. A product dimension describes the products that can appear as part of a frequent itemset, thus playing the role of an item dimension. The schema in Figure 9 represents, for each 1-hour period, the frequency counts of each set of products mined as frequent. The attributes of the product and calendar dimensions are not presented in Figure 9 and Figure 10 for the sake of a clear presentation. Just to mention a few, product name, category and sales department are some examples of attributes of the product dimension. Period of the day (morning, afternoon,…), weekday and holiday (yes or no) are some attributes of the calendar dimension.

4.3 Logical Design

A standardized logical schema is presented in Figure 11 and Figure 12. In the logical design we introduce one itemset dimension for each item dimension as can be seen in Figure 11. The itemset dimension works as a bridge table representing the n-to-m-relationships between the item dimensions and the facts in the conceptual schema. Analytical tools usually hide the existence of bridge tables. An analytical tool developed upon a data warehouse of frequent itemsets should know about the item dimensions and the corresponding itemset dimensions in order to provide such a transparency and perform the required mappings.

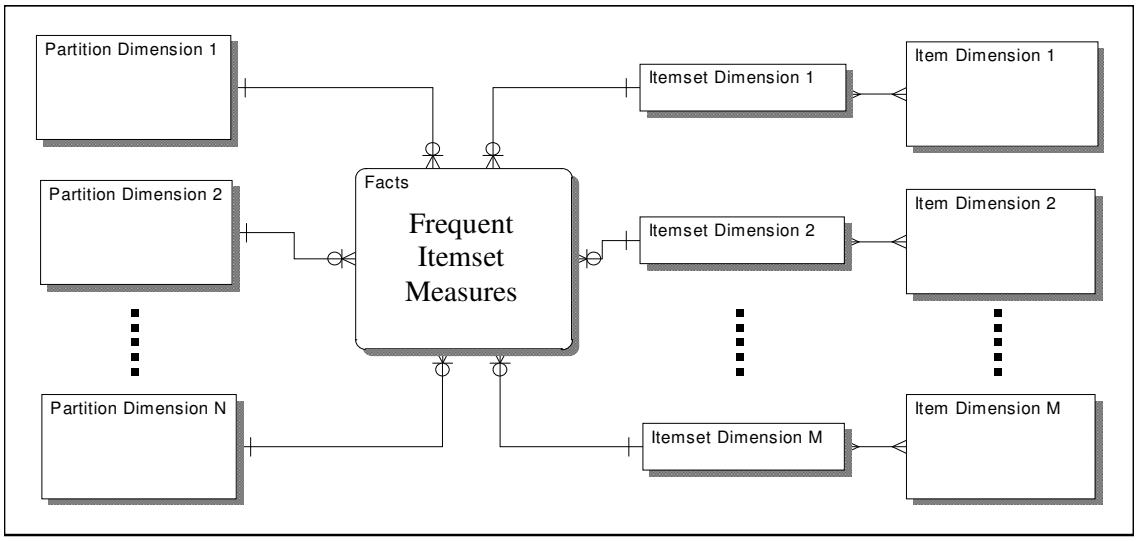


Figure 11 – Frequent itemset measures standardized logical schema

Multivalued dimensions (N:M relationship with the fact table) are commonly solved with the help of a bridge table (Kimball and Ross, 2002). The itemset dimensions contain one identifier for each distinct itemset, which will be used as a foreign key in the fact table and to group the items into itemsets as well. The logical schema presented in Figure 12 is derived straightforward from the conceptual schema presented in Figure 8.

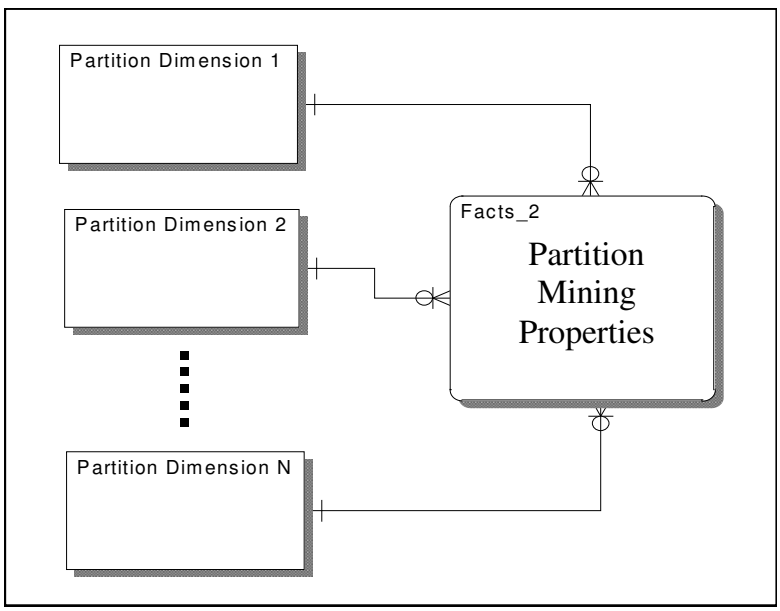


Figure 12 – Partition mining properties standardized logical schema

The logical schema for the running example is presented in the next section together with some sample data for each table. Relying on the presented standard

modeling, analytical and explorative tools can independently be developed. The standard modeling provided plays for frequent itemsets the same role as star and snowflake schemas play for regular data.

4.4 A Data Warehouse of Frequent Itemsets Example

One instance of a data warehouse of frequent itemsets is shown in Figure 13 and Figure 14. It has one partition dimension and one item dimension.

Partition dimensions organize the space of the original transactions into disjoint sets, which we call DW partitions. In our example we use a calendar dimension with a granularity of 1 hour. This means that each tuple in the calendar dimension represents a period of 1 hour, e.g., "02/15/2005 [08:00AM, 09:00AM)", "02/15/2005 [09:00AM, 10:00AM)", and so on. Of course, additional calendar information has to be stored, like the period of the day (morning, afternoon, …), weekday, holiday (yes, no) and any other calendar unit that represents an aggregation of the basic granularity. This basic granularity also sets the criteria for gathering the transactions in the pre-processing step.

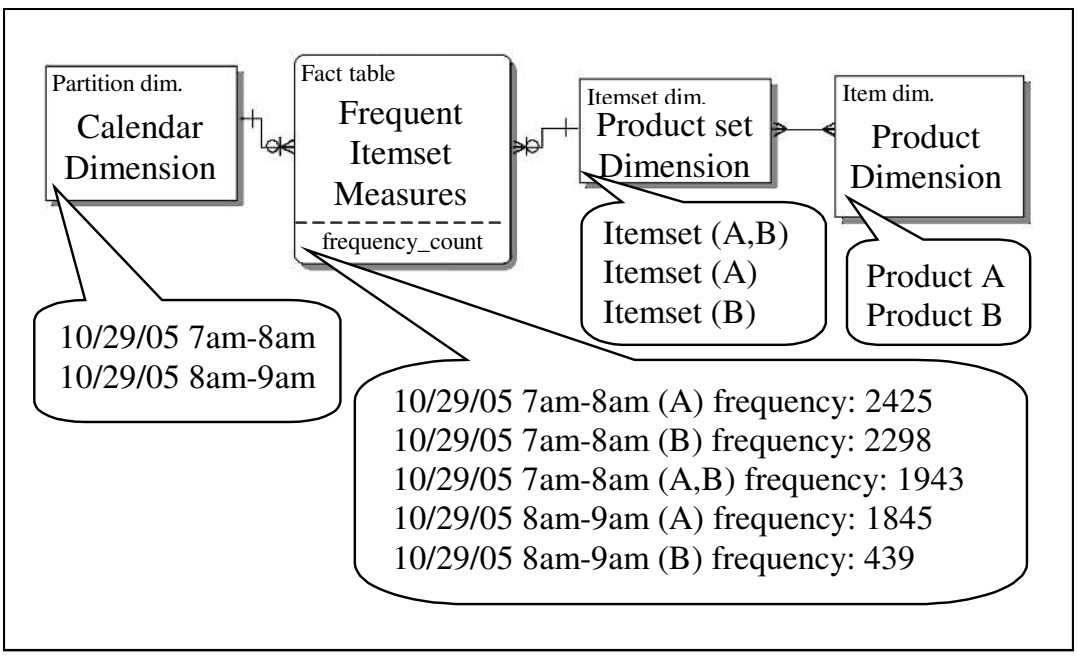


Figure 13 – Frequent itemset measures logical schema for the running example with some instance samples

The item dimensions provide additional information about the items that may appear in a transaction. In our sample instance we use a single product dimension. Examples of additional information are: description, department, category, etc.

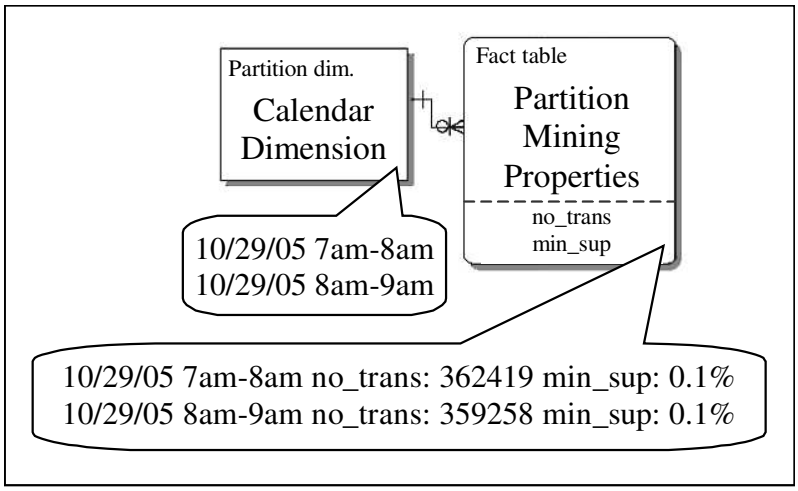


Figure 14 – Partition mining properties logical schema for the running example with some instance samples

A data warehouse of frequent itemsets is composed of two fact tables. The first fact table, presented in Figure 13, stores the frequency count for each set of products mined as frequent in a specific calendar granule (1 hour). Figure 13 also presents an itemset dimension that works as a bridge table between the multivalued dimension and the fact table. The second, presented in Figure 14, stores some partition properties and thus has relationships only with partition dimensions. It stores the number of transactions and the minimum support used for the frequent itemset mining task per calendar granule (1 hour).

Two instances representing 1-hour periods are shown in the calendar dimension. In Figure 13 two products are listed in the product dimension and some related itemsets are presented in the itemset dimension. The frequency counts of the itemsets for the two 1-hour periods are presented in the frequent itemset frequency measures fact table. Note that there is no frequency count for the itemset (A, B) related to the partition "10/29/2005 [08:00AM, 09:00AM)". This means that the itemset (A, B) was not frequent in this partition. In Figure 14 the number of transactions of the two 1-hour periods is presented along with the mining minimum support (0.1%) in the partition mining properties fact table.

This sample data warehouse of frequent itemsets tracks the frequent sets of products per hour. This is the definition of our minimum granularity of analysis. By such a minimum granularity, it is not possible to analyze time intervals shorter than one hour. On the other hand, all possible combinations of one-hour periods can be analyzed.

These combinations based on calendar-terms specify a set of DW partitions, which later we refer as ρ.

4.5 Basic Properties

A data warehouse of frequent itemsets presents two basic properties that constitute the building blocks for the pattern retrieval. These two properties are namely the disjoint partitions property and the error upper bound property, which are defined in the following.

Property 1 (Disjoint partitions): The DW partitions represent completely disjoint sets of transactions and thus the frequency counts of a specific itemset can be summed over any set of partitions. ■

Property 2 (**Error upper bound):** Given $|\rho_i|$ the number of original transactions in partition ρ_i and σ_{mi} the mining minimum support used on partition ρ_i, the product $|\rho_i|\sigma_{mi}$ provides a strict upper bound on the number of missed frequencies over partition ρ_i. Moreover, a strict error upper bound UBe_ρ for a set of DW partitions ρ is provided by simply summing the upper bounds of the individual partitions:

$$\mathrm{UBe}_\rho = \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} \quad (4)$$

In the special case where σ_{mi} is constant over ρ we have:

$$\mathrm{UBe}_\rho = \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} = \sigma_m \sum_{\rho_i \in \rho} |\rho_i| = \sigma_m |\rho| \quad (5) \blacksquare$$

Property 1 holds for any data warehouse of frequent itemsets. Property 2 holds when storing all frequent itemsets in the data warehouse or when using a condensed representation for frequent itemsets without loss of information. When using condensed representations with loss, the error upper bound definition must be adjusted to consider the error introduced by the specific representation.

5 Retrieving Patterns from the Data Warehouse of Frequent Itemsets

The present chapter deals with the pattern retrieval based on the information stored in the data warehouse of frequent itemsets. Section 5.1 describes how the information stored in the Data Warehouse of Frequent Itemsets can be used to retrieve frequent itemsets holding on original transactions represented by any arbitrary set of DW partitions. This task is performed by the Basic Frequent Itemset Retrieval Capabilities component of DWFIST. The completeness and precision of the retrieved set of frequent itemsets are discussed as well. Section 5.2 addresses the task of the Frequent Itemset Based Pattern Mining Engine component, which is responsible for deriving patterns that can be built using frequent itemsets. Three patterns are presented as examples: association rules, Boolean rules and classification rules.

5.1 Basic Frequent Itemset Retrieval Capabilities

Before presenting the formal aspects of the Basic Frequent Itemset Retrieval Capabilities component we provide one simple example to illustrate how this task is accomplished. For this purpose, we return to the data source and calendar-based query examples in Figure 5. Figure 15 presents the frequent itemsets that would be stored in the DW using a daily granule and a minimum support of 60% for mining. The answers for the calendar-based query examples are also presented.

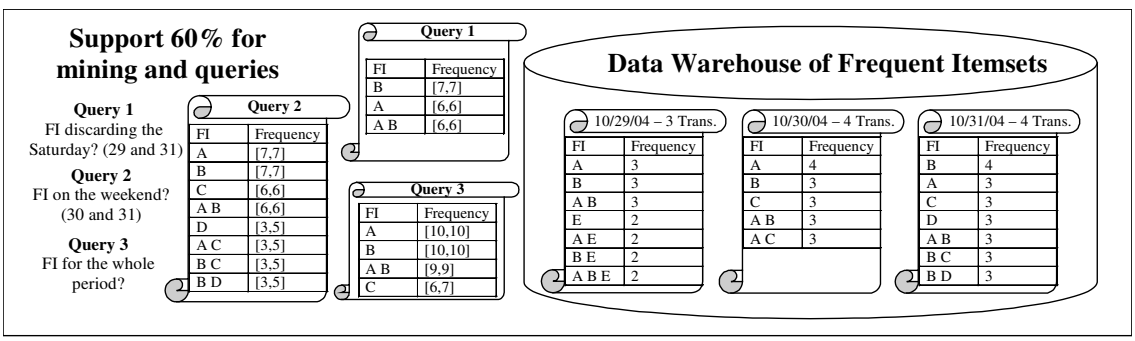


Figure 15 - Examples of Frequent Itemset Retrieval from the DW

For the sake of simplicity, we use the same minimum support (60%) for the mining task and for the queries. Note, that the result is the union of the frequent itemsets of the DW partitions that satisfy the constraints excluding the frequent itemsets with frequency count below the minimum support requested. For example, Query 2 is the

complete union once that no frequent itemset is guaranteed to be below the requested minimum support (60% of 8 = 4.8). Using the basic properties of the DW, the stored frequency counts are summed providing a lower bound. The upper bound is computed considering the partitions where the itemset is missing. For instance, considering the item "C" reported in Query 3 we have the lower bound 6 as the sum of frequency counts in partitions "10/30/2004" and "10/31/2004". The item "C" is missing in partition "10/29/2004" and this partition satisfy Query 3 constraints. Using the number of transactions and the mining minimum support stored for this partition we can easily calculate the maximum number of missed transactions. In this case, we have 3 transactions and a mining minimum support of 60% that gives us a maximum of 1 missed transaction (60% of 3 = 1.8). The upper bound 7 is computed by summing the maximum number of missed transactions and the lower bound (6 + 1).

Let us now discuss formally the task of the Basic Frequent Itemset Retrieval Capabilities component.

5.1.1 Frequent Itemset Retrieval Task

The task of retrieving frequent itemsets holding on original transactions represented by any arbitrary set of DW partitions is defined as follows:

Frequent Itemset Retrieval Task: Given a set of DW partitions ρ and a query minimum support σ_q, retrieve the set Δ of all itemsets I with $S_\rho(I) \geq \sigma_q$. The set Δ must contain also approximate frequency counts for each I represented by intervals [LB_$F_\rho(I)$,UB_$F_\rho(I)$]. Therefore, if I is a σ_q-frequent itemset holding on the original transactions represented by ρ with frequency count $F_\rho(I)$ then:

$$I \in \Delta \text{ and } \text{LB\_}F_\rho(I) \leq F_\rho(I) \leq \text{UB\_}F_\rho(I) \quad (6)$$

The following function signature represents this task:

$$\Delta = FI\_retrieval(\rho, \sigma_q) \quad (7)\blacksquare$$

In other words, the set Δ of retrieved σ_q-frequent itemsets must be complete, not missing any σ_q-frequent itemset holding on the original transactions represented by ρ, and lower and upper bounds on the frequency counts must be provided as well.

This task cannot be accomplished for all values of σ_q as the DW does not contain every itemset frequency count. Instead, it contains only the ones that were mined as frequent. The completeness of Δ must be checked and the frequency lower and upper bounds computed. Precision issues regarding the set Δ must be considered as well. The basic DW properties presented in section 4.5 provide the basis for addressing all these issues, which will be done in the remainder of section 5.1.

5.1.2 Retrieving Frequent Itemsets and Frequency Counts

The following property specifies how the frequency count bounds for itemsets retrieved from the DW can be calculated.

Property 3 (Frequency count bounds): Lower and upper bounds on the frequency count of an itemset I over a set of DW partition ρ can be calculated using properties 1 and 2 as follows:

$$\text{LB\_}F_{\rho}(I) = \sum_{\rho_i \in \rho_{known}} F_{\rho_i}(I) \qquad \text{UB\_}F_{\rho}(I) = \text{LB\_}F_{\rho}(I) + \text{UBe}_{\rho - \rho_{known}} \quad (8)$$

where ρ_{known} represents the subset of partitions from ρ where the itemset I was mined as frequent and thus its frequency count is known. The expression ρ - ρ_{known} represents the complementary subset from ρ where the frequency count of I is unknown. ■

Properties 1 and 2 provide the building blocks for property 3. Property 3 is used in practice to compute and retrieve from the DW the approximate frequency of an itemset over a set of original transactions. The set of original transactions is represented by any arbitrary set of DW partitions.

The samples presented in Figure 13 and Figure 14 can be used to exemplify how the frequency count bounds are computed. Considering ρ equal to the two sample partitions presented in Figure 14 we can depict how the frequency bounds would be computed for the itemsets (A), (B) and (A,B) using property 3. The frequency lower bound is simply computed by summing the known frequency counts. Therefore, we have 4270 (2425+1845) for (A); 2737 (2298+439) for (B); and 1943 for (A,B). The frequency upper bound is computed using the frequency lower bound and the information stored on the partition mining properties fact table. As the frequency counts

of itemsets (A) and (B) are known in both partitions the exact frequency is known and thus the frequency upper bound is equal to the frequency lower bound. The frequency of the itemset (A,B) is unknown in partition "10/29/2005 [08:00AM, 09:00AM)". The error upper bound for this partition is calculated using equation (4) and is 359.258 (0.1% of 359258). Therefore, the frequency upper bound for itemset (A,B) is 2302.258 (1943+359.258). The decimal part can be discarded at the end, as frequency counts are integer measures. Finally, the computed frequency count bounds are [1943,2302] for itemset (A,B), [4270,4270] for itemset (A) and [2737,2737] for itemset (B).

In the following we will show how to derive frequent itemsets from a DW that stores all frequent itemsets (i.e. no condensed representation used) by means of SQL queries. Considering our data warehouse of frequent itemsets example, Property 1 tells us that we can sum the frequencies of a specific itemset over any calendar partition represented in the Calendar Dimension. Also, as discussed before, the Calendar Dimension provides attributes representing different calendar features. In our example, an attribute *day_period* in the calendar dimension may classify the periods of 1 hour into morning, afternoon, evening and dawn. Figure 16 presents an example of a query that can be executed on our DW example to retrieve the itemsets and their lower bound frequency counts considering the morning period. Relating to the frequent itemset retrieval task, this corresponds to set ρ as the set of DW partitions representing the morning period and to compute the left equation in (8) for each itemset.

```
Select
  FIM.itemset_id,
  sum(FIM.frequency_count) as LB_Frequency
From Calendar_Dimension CD,
     Frequent_Itemset_Measures FIM
Where CD.CD_id = FIM.CD_id and
      CD.day_period = 'Morning'
Group by FIM.itemset_id
```

Figure 16 – Frequency lower bound query

Based on Property 2 we can compute two kinds of error upper bounds, namely a global error upper bound and an itemset-specific error upper bound. The global error upper bound computes a single error margin for all itemsets over a set of DW partitions. The itemset-specific error upper bound computes an error margin for each individual itemset over a set of DW partitions. The first is easier to compute while the second is more precise.

A global error upper bound for our example is obtained computing equation (4) for all partitions related to the morning period (ρ). Figure 17 presents the corresponding query.

```
Select
  sum(PMP.no_trans*PMP.min_sup)as Global_Error
From Calendar_Dimension CD,
     Partition_Mining_Properties PMP
Where CD.CD_id = PMP.CD_id and
      CD.day_period = 'Morning'
```

Figure 17 – Global error upper bound query

An itemset-specific error upper bound for our example is obtained also by equation (4), however, for each itemset, only the partitions where its frequency is unknown are considered (ρ - ρ_{known}). The reasoning is that as far as the itemset frequency is known for a partition there is no reason to consider the error related to that partition. In the relational paradigm, it is more efficient to compute the error related to the partitions where the itemset frequency is known and subtract it from the global error to obtain the error related to the partition with unknown frequencies. The query presented in Figure 18 computes for each itemset the error related to the partitions where the itemset frequency is known (ρ_{known}).

```
Select
  FIM.itemset_id,
  sum(PMP.no_trans*PMP.min_sup)as Known_Part_Error
From Calendar_Dimension CD,
     Frequent_Itemset_Measures FIM,
     Partition_Mining_Properties PMP
Where CD.CD_id = PMP.CD_id and
      CD.CD_id = FIM.CD_id and
      CD.day_period = 'Morning'
Group by FIM.itemset_id
```

Figure 18 – Itemset-specific error upper bound query

A query minimum support (σ_q) must be specified for retrieval. By doing so, the answer must comprise the itemsets considered as frequent according to the query minimum support provided. The queries presented in Figure 16, Figure 17 and Figure 18 can be executed one after the other and have their results combined considering a query minimum support but, this task can also be computed by one single query. Figure 19 presents such a query for our example. This query implements the function signature

presented in equation (7). A tool that provides an interface for specifying conditions on the dimensions and accesses the frequent itemsets through the presented standardized logical schema can automatically build such a query.

```
Select
  S.itemset_id,
  S.LB_Frequency,
  ( S.LB_Frequency +
    G.Global_Error -
    S.Known_Part_Error) as UB_Frequency
From ( Select
         FIM.itemset_id,
         sum(FIM.frequency_count) as LB_Frequency,
         sum(PMP.no_trans*PMP.min_sup) as Known_Part_Error,
       From Calendar_Dimension CD,
            Frequent_Itemset_Measures FIM,
            Partition_Mining_Properties PMP
       Where  CD.CD_id = PMP.CD_id and
              CD.CD_id = FIM.CD_id and
              CD.day_period = 'Morning'
       Group by FIM.itemset_id) S,
     ( Select
         sum(PMP.no_trans) as Total_no_trans,
         sum(PMP.no_trans*PMP.min_sup) as Global_Error
       From Calendar_Dimension CD,
            Partition_Mining_Properties PMP
       Where  CD.CD_id = PMP.CD_id and
              CD.day_period = 'Morning') G
Where( S.LB_Frequency +
       G.Global_Error -
       S.Known_Part_Error) >=
     ( G.Total_no_trans * :query_minimum_support)
```

Figure 19 – Single query for frequent itemset retrieval

As the retrieved itemset frequencies are represented by intervals, we can only discard an itemset when its frequency interval lies completely below the specified threshold. Some important questions arise immediately at this point: (1) "Which range of query minimum support can be supported by the DW as it does not contain complete information about the original transactions?"; (2) "Is it possible to guarantee that we are retrieving all the frequent itemsets holding on the original data?"; and (3) "How close is the answer provided by the DW to the real set of frequent itemsets holding on the original transactions?". These issues are addressed in the following.

5.1.3 Completeness

Let us now discuss under which conditions the completeness of the set of retrieved frequent itemsets Δ can be guaranteed.

Property 4 (Completeness 1): If $F_\rho(I)$ is equal or greater than UBe_ρ then I is represented in at least one partition $\rho_i \in \rho$.

Proof: Considering that no partition $\rho_i \in \rho$ has any information about I means that $F_{\rho i}(I) < |\rho_i|\sigma_{mi} \; \forall \; \rho_i \in \rho$. Using property 1 and 2, and comparing with (4):

$$\sum_{\rho_i \in \rho} F_{\rho_i}(I) < \sum_{\rho_i \in \rho} |\rho_i| \, \sigma_{mi} = \text{UBe}_\rho \quad (9) \; \blacksquare$$

Property 5 (Completeness 2): Let Δ be a list of σ_q-frequent itemsets retrieved from a set of DW partitions ρ. The completeness of Δ can be guaranteed for any σ_q such that:

$$\sigma_q \geq \frac{\text{UBe}_\rho}{|\rho|} \quad (10)$$

Proof: In order to be considered a σ_q-frequent itemset over a set of DW partitions ρ an itemset I must have a minimum frequency count of $\sigma_q|\rho|$. From property 4 we have that all itemset I with $F_\rho(I) \geq \text{UBe}_\rho$ will be present in at least one partition $\rho_i \in \rho$. If the minimum frequency count $\sigma_q|\rho|$ is equal or greater than the error upper bound UBe_ρ then the completeness of Δ can be guaranteed:

$$|\rho|\sigma_q \geq \text{UBe}_\rho \quad (11)$$

Equation (10) is directly obtained from (11). Also, in the special case where σ_{mi} is constant over ρ, from equation (5) we have:

$$|\rho|\sigma_q \geq |\rho|\sigma_m \qquad\qquad \sigma_q \geq \sigma_m \quad (12) \; \blacksquare$$

Properties 4 and 5 tell us in practice that the retrieved set of patterns is guaranteed to be complete as far as we request the frequent itemsets using a query minimum support equal or greater than the mining minimum support. In other words, given this condition on σ_q we are not missing any σ_q -frequent itemset holding on a set of original transactions. Once again, the set of original transactions is represented by any arbitrary set of DW partitions.

5.1.4 Precision

We address the precision of the result according to two aspects: false positives and frequency interval tightness. For the sake of simplicity and without loss of generality, in this subsection we consider that a constant minimum support σ_m was used for mining during the pre-processing step of all partitions ρ_i belonging to a set of DW partitions ρ. If this is not the case, we can simply assume σ_m as the maximum σ_{mi} used for all ρ_i.

The first aspect relates to false positives, i.e., itemsets that are reported to be frequent although their real frequencies lie below the specified threshold. Such false positives are likely to appear because the frequency retrieved from the DW is an approximate frequency count. Nevertheless, an interesting property of such itemsets is presented as follows:

Property 6 (local frequent itemset): As far as the completeness is guaranteed, a false frequent itemset reported as frequent, due to the approximate frequency count, for a query minimum support σ_q over a set of DW partitions ρ is guaranteed to be a σ_q–frequent itemset over the subset of partitions from ρ where its frequency is known.

Proof: Let I be a false frequent itemset reported as frequent for a query minimum support σ_q over a set of DW partitions ρ. Also, let ρ_{known} be the subset of partitions on ρ where the frequency of I is known, $F_{\rho known}(I)$ the frequency of I over ρ_{known}, $|\rho_{known}|$ the number of transactions over ρ_{known} and $\mathrm{UBe}_{(\rho - \rho known)}$ the itemset-specific error upper bound for I over ρ. The following equation expresses the property we want to guarantee:

$$\frac{F_{\rho known}(I)}{|\rho_{known}|} \geq \sigma_q \quad (13)$$

In order to be reported as frequent, $F_{\rho known}(I) + \mathrm{UBe}_{(\rho - \rho known)}$ must be at least equal to $|\rho|\sigma_q$, therefore:

$$F_{\rho known}(I) \geq |\rho|\sigma_q - \mathrm{UBe}_{(\rho - \rho known)}$$

The itemset-specific error upper bound $UBe_{(\rho - \rho known)}$ can be expressed as $(|\rho| - |\rho_{known}|)\sigma_m$:

$$F_{\rho known}(I) \geq |\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m$$

Substituting $F_{\rho known}(I)$ in (13):

$$\frac{|\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m}{|\rho_{known}|} \geq \sigma_q$$
$$|\rho|\sigma_q - (|\rho| - |\rho_{known}|)\sigma_m \geq |\rho_{known}|\sigma_q$$
$$-(|\rho| - |\rho_{known}|)\sigma_m \geq |\rho_{known}|\sigma_q - |\rho|\sigma_q$$

Multiplying by minus one (-1):

$$(|\rho| - |\rho_{known}|)\sigma_m \leq |\rho|\sigma_q - |\rho_{known}|\sigma_q$$
$$(|\rho| - |\rho_{known}|)\sigma_m \leq (|\rho| - |\rho_{known}|)\sigma_q$$
$$\sigma_m \leq \sigma_q$$

The same condition to guarantee the completeness of the result is a sufficient condition for Property 6.■

The above property tells that the additional reported itemsets are indeed itemsets that present a frequent behavior on a subset of the current set of DW partitions being analyzed and thus are probably worth of a more detailed analysis. A simple coverage measure related to the number of partitions where the itemset frequency is known can aid the analyst, aiming at separating the itemsets that present a high coverage rate for the current set of DW partitions being analyzed from the locally frequent ones (low coverage rate). A simple count statement on the query presented in Figure 18 can easily compute this measure.

The second aspect relates to frequency interval tightness. In order to discuss this aspect we introduce a relative error measure:

Definition 19 (frequency relative error): Given an exact frequency count $F(I)$ and an approximation $F(I)$' with an error ε equals to $|F(I) - F(I)'|$, the frequency relative error (FRE) is defined as:

$$\text{FRE} = \frac{\varepsilon}{F(I)'} \quad (14)$$

The frequency relative error quantifies the size of the error with respect to the approximate frequency count.

Property 7 (worst-case frequency relative error): Considering a frequent itemset retrieval task and using as frequency approximation the middle point of the frequency interval, the worst-case for the frequency relative error is given by:

$$\text{Worst\_case\_FRE} = \frac{\sigma_m}{2\left(\sigma_q - 0.5\sigma_m\right)} \quad (15)$$

Proof: The error upper bound UBe_ρ corresponds to the maximum length of the frequency interval. Using the middle point of the frequency interval as the frequency approximation, the maximum ε is half of UBe_ρ. Also, $F(I)$' is at least $(|\rho|\sigma_q - 0.5\text{UBe}_\rho)$ in order to be considered frequent. Therefore, the worst-case frequency relative error is:

$$\frac{\text{UBe}_\rho}{2\left(|\rho|\sigma_q - 0.5\text{UBe}_\rho\right)}$$

Substituting UBe_ρ using equation (5):

$$\frac{|\rho|\sigma_m}{2\left(|\rho|\sigma_q - 0.5|\rho|\sigma_m\right)} =$$

$$\frac{|\rho|\sigma_m}{2|\rho|\left(\sigma_q - 0.5\sigma_m\right)} =$$

$$\frac{\sigma_m}{2\left(\sigma_q - 0.5\sigma_m\right)}$$

This provides the worst-case in equation (15). ■

The above property provides a precision guarantee for the frequency count of itemsets retrieved from the DW. For $\sigma_q = \sigma_m$ equation (15) results in 100% and it decreases as the value of σ_q increases moving away from σ_m. As an example, consider a set of partitions in which a mining minimum support of 0.1% was used. If we request the frequent itemsets using a query minimum support of 0.3%, equation (15) tells us that the worst-case frequency relative error is 20%. The completeness of the result is guaranteed as well because the query minimum support (0.3%) is greater than the mining minimum support (0.1%).

Equation (15) can be used to aid at the choice of the mining minimum support. If we expect to retrieve frequent itemsets using a query minimum support of *S*% and want to guarantee a worst-case frequency relative error of E%, equation (15) can be applied to compute a suitable mining minimum support.

5.2 Frequent Itemset Based Pattern Mining Engine

Once we have retrieved a list of frequent itemsets and their corresponding approximate frequency counts, holding on a set of DW partitions ρ using a query minimum support σ_q, we can build other patterns upon it. In doing so, the uncertainty about the exact frequency counts must be taken into account. This is the task of the Frequent Itemset Based Pattern Mining Engine component. We exemplify this task by discussing how association rules, Boolean rules and classification rules can be obtained from frequent itemsets retrieved from the DW. Furthermore, analyzing the worst-case we can derive theoretical bounds providing a precision guarantee for some interestingness measures of such patterns.

5.2.1 Association Rules

Extracting association rules based on a list of frequent itemsets is a well-known and solved problem. An efficient algorithm is presented in Agrawal et al. (1996). As far as the completeness of the set of frequent itemsets is guaranteed by properties 4 and 5, the completeness of the set of association rules is also guaranteed. The relevant issue in our context is how to compute measures related to association rules using the approximate frequency counts of itemsets. Two measures are discussed in the

following, namely confidence and gain. For both, we present how specific bounds for individual rules can be computed. Afterwards, a precision guarantee is provided.

5.2.1.1 Confidence

The confidence of a rule $\alpha \rightarrow \gamma$ represents the probability of having the consequent (γ) in a transaction given that the antecedent (α) is present. Given a set of DW partitions ρ, it can be computed as follows:

$$\text{conf}_{\rho}(\alpha \rightarrow \gamma) = \frac{F_{\rho}(\alpha \cup \gamma)}{F_{\rho}(\alpha)} \quad (16)$$

Lower and upper bounds on the confidence for an individual rule can be computed as:

$$\text{LBconf}_{\rho}(\alpha \rightarrow \gamma) = \frac{\text{LB\_}F_{\rho}(\alpha \cup \gamma)}{\text{UB\_}F_{\rho}(\alpha)} \qquad \text{UBconf}_{\rho}(\alpha \rightarrow \gamma) = \frac{\text{UB\_}F_{\rho}(\alpha \cup \gamma)}{\text{LB\_}F_{\rho}(\alpha)} \quad (17)$$

The reasoning is quite simple. We select the frequency lower bound or upper bound in the numerator or denominator in order to minimize or maximize the result. Using the sample data in Figure 13 and Figure 14, we can compute the confidence bounds for the rule A→B. Remembering the retrieved frequency counts from itemset (A) [4270,4270] and itemset (A,B) [1943,2302], and applying equation (17) we have LBconf_{ρ}(A→B) as 45% and UBconf_{ρ}(A→B) as 54%. Therefore, an approximate confidence of 49.5% ± 4.5 percentual points can be retrieved.

As we raise the query minimum support (σ_q) for the frequent itemset retrieval task, the ratio $\text{UB}_{\rho}/\text{LB\_}F_{\rho}(\alpha)$ between the frequency count errors and the itemsets' lower bound frequency is reduced. The simple reasoning is that the error is kept constant and the itemsets' lower bounds are increased. Therefore, it can be expected to get lower confidence errors when raising σ_q.

The following property provides a precision guarantee for the confidence measure of every association rule retrieved from a set of DW partitions ρ using a query minimum support σ_q. For the sake of simplicity and without loss of generality, we consider that a constant minimum support σ_m was used for mining during the pre-

processing step of all partitions ρ_i belonging to ρ. If this is not the case, we can simply assume σ_m as the maximum σ_{mi} used for all ρ_i.

Property 8: Given a set of DW partitions ρ, σ_m a mining minimum support used over ρ, σ_q a query minimum support and taking the central value of the interval [LBconf,UBconf] as the confidence approximate value, we have a worst-case error on the confidence expressed by:

$$\text{Worst\_Case\_conf\_error}_\rho = \frac{\sigma_m}{\sigma_q - \sigma_m} \quad (18)$$

Proof: Taking the central value of the interval [LBconf,UBconf] we have a maximum error of half of the interval's length for one individual association rule. Maximizing the error we have:

$$\text{conf\_error}_\rho(\alpha \rightarrow \gamma) = \frac{\text{UBconf}_\rho(\alpha \rightarrow \gamma) - \text{LBconf}_\rho(\alpha \rightarrow \gamma)}{2} = \left(\frac{\text{UB\_}F_\rho(\alpha \,\text{Y}\, \gamma)}{\text{LB\_}F_\rho(\alpha)} - \frac{\text{LB\_}F_\rho(\alpha \,\text{Y}\, \gamma)}{\text{UB\_}F_\rho(\alpha)} \right) \frac{1}{2} \leq$$

$$\left(\frac{\text{LB\_}F_\rho(\alpha \,\text{Y}\, \gamma) + \text{UBe}_\rho}{\text{LB\_}F_\rho(\alpha)} - \frac{\text{LB\_}F_\rho(\alpha \,\text{Y}\, \gamma)}{\text{LB\_}F_\rho(\alpha) + \text{UBe}_\rho} \right) \frac{1}{2} = \frac{\text{UBe}_\rho \left(\text{LB\_}F_\rho(\alpha) + \text{UBe}_\rho\right) + \text{UBe}_\rho \text{LB\_}F_\rho(\alpha \,\text{Y}\, \gamma)}{2\text{LB\_}F_\rho(\alpha)\left(\text{LB\_}F_\rho(\alpha) + \text{UBe}_\rho\right)} =$$

$$\frac{1}{2\text{LB\_}F_\rho(\alpha)} \left(\text{UBe}_\rho + \frac{\text{UBe}_\rho \, \text{LB\_}F_\rho(\alpha \,\text{Y}\, \gamma)}{\left(\text{LB\_}F_\rho(\alpha) + \text{UBe}_\rho\right)} \right) < \frac{\text{UBe}_\rho + \text{UBe}_\rho}{2\text{LB\_}F_\rho(\alpha)} = \frac{\text{UBe}_\rho}{\text{LB\_}F_\rho(\alpha)} <$$

$$\frac{\text{UBe}_\rho}{\sigma_q|\rho| - \text{UBe}_\rho} = \frac{\sigma_m|\rho|}{\sigma_q|\rho| - \sigma_m|\rho|} = \frac{\sigma_m}{\sigma_q - \sigma_m}$$

■

The graph in Figure 20 was drawn using equation (18). The values on the horizontal axis refer to the σ_q/σ_m ratio between the query minimum support and the mining minimum support. In the running example we used a mining minimum support of 0.1%. Therefore, Figure 20 refers to values ranging from 0.2% to 2% for the query minimum support when considering our running example. If we retrieve the association rules from the DW using a query minimum support of 1.1% in our example, a precision of 10 percentual points is guaranteed for the confidence measure.

The graph in Figure 20 can be used to adjust the DW configuration. We can look at it from a different point of view and use it to aid at the choice of the mining minimum support. For example, if we expect to mine association rules using a query minimum support of *S*% and want to guarantee a maximum confidence error of 11 percentual

points, Figure 20 tell us that we should use a mining minimum support of one-tenth of *S*.

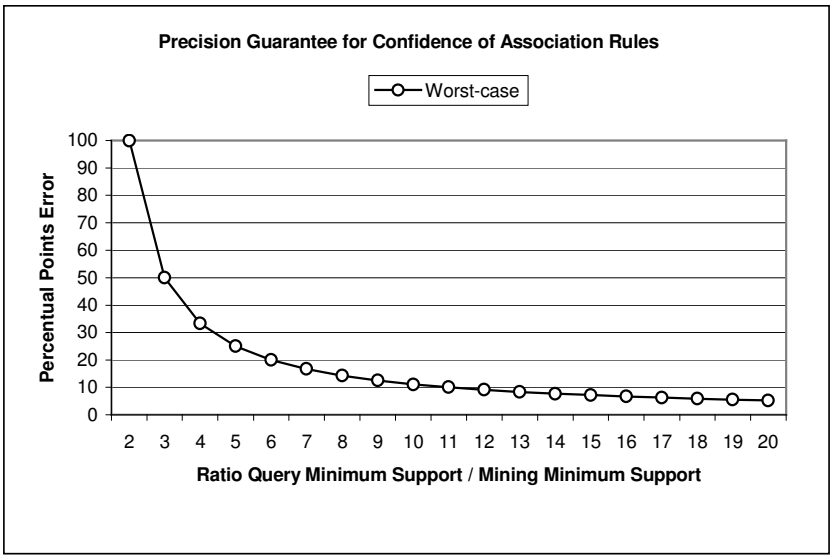


Figure 20 – Precision guarantee for confidence of association rules

5.2.1.2 Gain

The gain function of Fukuda et al. (1996) is given below. The parameter θ is a fractional constant between 0 and 1:

$$\text{gain}_{\rho}(\alpha \rightarrow \gamma) = F_{\rho}(\alpha \text{ Y } \gamma) - \theta\, F_{\rho}(\alpha) \quad (19)$$

Lower and upper bounds of the gain for an individual rule can be computed as:

$$\begin{aligned} \text{LBgain}_{\rho}(\alpha \rightarrow \gamma) &= \text{LB\_}F_{\rho}(\alpha \text{ Y } \gamma) - \theta\, \text{UB\_}F_{\rho}(\alpha) \\ \text{UBgain}_{\rho}(\alpha \rightarrow \gamma) &= \text{UB\_}F_{\rho}(\alpha \text{ Y } \gamma) - \theta\, \text{LB\_}F_{\rho}(\alpha) \end{aligned} \quad (20)\ \blacksquare$$

The following property provides a precision guarantee for the gain measure of association rules retrieved from a set of DW partitions ρ.

Property 9: Using the central value in the interval [LBgain,UBgain] as the gain approximate value, we have a worst-case error on the gain expressed by:

$$\text{Worst\_Case\_gain\_e}_{\rho}(\alpha \rightarrow \gamma) = \text{UBe}_{\rho} \quad (21)$$

Proof: Taking the central value of the interval [LBgain,UBgain] we have a maximum error of half of the interval's length:

$$\text{gain\_e}_\rho(\alpha \to \gamma) = \frac{\text{UBgain}_\rho(\alpha \to \gamma) - \text{LBgain}_\rho(\alpha \to \gamma)}{2} =$$

$$\frac{\text{UB\_}F_\rho(\alpha \cup \gamma) - \theta\,\text{LB\_}F_\rho(\alpha) - \left(\text{LB\_}F_\rho(\alpha \cup \gamma) - \theta\,\text{UB\_}F_\rho(\alpha)\right)}{2} =$$

$$\frac{\text{LB\_}F_\rho(\alpha \cup \gamma) + \text{UBe}_\rho - \theta\,\text{LB\_}F_\rho(\alpha) - \text{LB\_}F_\rho(\alpha \cup \gamma) + \theta\left(\text{LB\_}F_\rho(\alpha) + \text{UBe}_\rho\right)}{2} =$$

$$\frac{\text{UBe}_\rho + \theta\,\text{UBe}_\rho}{2} \le \text{UBe}_\rho \qquad \max(\theta) = 1$$

■

The worst-case error on the gain measure is exactly the same as the upper bound on the frequency count error and is completely independent of the query minimum support.

Similar precision guarantees can be provided for other interestingness measures of association rules.

5.2.2 Boolean Rules

In Mannila and Toivonen (1996) a simple rule formalism of general Boolean rules is presented. Besides, it shows that the frequencies of sets are sufficient to determine the confidence of Boolean rules. In this section we extend this work to a wider scenario where an approximation of the frequencies of sets is known instead of the exact counts. First, let us present the proposition stated in Mannila and Toivonen (1996):

Proposition: For all Boolean formulae ϖ over *items* there are *itemsets* $W_1,\ldots,W_m \subseteq$ *items* and coefficients $e_1,\ldots,e_m \in \{-1,+1\}$ such that for a set of transactions T:

$$F_T(\varpi) = \sum_{i=1}^{m} e_i F_T(W_i) \quad (22)$$

That is, knowing the special terms $F_T(W_i)$ is sufficient to determine the number of transactions satisfying any Boolean formula, and therewith also the confidence of any Boolean rule. ■

$$\mathrm{conf}_\rho(\mathrm{X} \rightarrow \mathrm{Y} \vee \mathrm{Z}) = \frac{F_\rho(\mathrm{X\,Y\,Y}) + F_\rho(\mathrm{X\,Y\,Z}) - F_\rho(\mathrm{X\,Y\,Y\,Y\,Z})}{F_\rho(\mathrm{X})}$$

$$\mathrm{conf}_\rho(\mathrm{X} \vee \mathrm{Y} \rightarrow \mathrm{Z}) = \frac{F_\rho(\mathrm{X\,Y\,Z}) + F_\rho(\mathrm{Y\,Y\,Z}) - F_\rho(\mathrm{X\,Y\,Y\,Y\,Z})}{F_\rho(\mathrm{X}) + F_\rho(\mathrm{Y}) - F_\rho(\mathrm{X\,Y\,Y})}$$

$$\mathrm{conf}_\rho(\mathrm{X} \wedge \neg\mathrm{Y} \rightarrow \mathrm{Z}) = \frac{F_\rho(\mathrm{X\,Y\,Z}) - F_\rho(\mathrm{X\,Y\,Y\,Y\,Z})}{F_\rho(\mathrm{X}) - F_\rho(\mathrm{X\,Y\,Y})}$$

$$\mathrm{conf}_\rho(\mathrm{X} \rightarrow \mathrm{Y} \wedge \neg\mathrm{Z}) = \frac{F_\rho(\mathrm{X\,Y\,Y}) - F_\rho(\mathrm{X\,Y\,Y\,Y\,Z})}{F_\rho(\mathrm{X})}$$

Figure 21 – Computing a Boolean rule confidence over a set of DW partitions ρ

The confidence for a Boolean rule is expressed in the same way as for an association rule, thus expressed by equation (16). Some examples are presented in Figure 21, where ρ is a set of DW partitions.

Mannila and Toivonen (1996) provides bounds on the frequency of a Boolean formula considering that only a subset of the frequency counts of the itemsets $W_1,\ldots,W_\mathrm{m}$ is known. In the following, we generalize this problem to a wider scenario where the frequency counts of the itemsets $W_1,\ldots,W_\mathrm{m}$ can be either unknown or imprecise.

Property 10: Let the frequency counts of the *itemsets* $W_1,\ldots,W_\mathrm{m}$ be expressed by upper and lower bounds as presented in equation (8). Bounds on the frequency count of a Boolean formula ϖ over a set of DW partitions ρ can be expressed as:

$$\begin{aligned} \mathrm{LB\_}F_\rho(\varpi) &= \sum_{i=1}^{\mathrm{m}} e_i \mathrm{LB\_}F_\rho(W_i) + \sum_{\forall W_i \mid e_i = -1} e_i \mathrm{UBe}_\rho(W_i) \\ \mathrm{UB\_}F_\rho(\varpi) &= \sum_{i=1}^{\mathrm{m}} e_i \mathrm{LB\_}F_\rho(W_i) + \sum_{\forall W_i \mid e_i = +1} e_i \mathrm{UBe}_\rho(W_i) \end{aligned} \tag{23}$$

■

In other words, to compute the lower bound of a Boolean formula ϖ we take the lower bound of all itemsets W_i where e_i is positive and the upper bounds of all itemsets W_i where e_i is negative. We do exactly the opposite to compute the upper bound. It is important to note that the unknown itemsets are just a special case where the lower bound is zero and the upper bound is the error upper bound provided by property 2. We can also express the Boolean formula ϖ frequency upper bound as a function of the lower bound:

$$\begin{aligned} &\mathrm{UBe}_\rho(\varpi) = \mathrm{UB\_}F_\rho(\varpi) - \mathrm{LB\_}F_\rho(\varpi) = \sum_{i=1}^{\mathrm{m}} \mathrm{UBe}_\rho(W_i) \\ &\mathrm{UB\_}F_\rho(\varpi) = \mathrm{LB\_}F_\rho(\varpi) + \mathrm{UBe}_\rho(\varpi) \end{aligned} \quad (24)$$

Considering the special case where the error upper bound for the frequency of an itemset is expressed by equation (5), the worst-case frequency error upper bound for a Boolean formula is:

$$\mathrm{UBe}_\rho(\varpi) = \sum_{i=1}^{\mathrm{m}} \mathrm{UBe}_\rho(W_i) = \sum_{i=1}^{\mathrm{m}} \sigma_m |\rho| = \mathrm{m}\sigma_m |\rho| \quad (25)$$

It can be seen easily in equation (25) that the errors may grow large quickly as the complexity of the Boolean formulae grows. As the Boolean formula complexity grows, more itemsets are required to compute the formula's frequency, which increases **m** raising the frequency error upper bound that can be guaranteed. Fortunately, simple Boolean rules, as the examples in Figure 21, can already reveal interesting insights in the data behavior. For such rules it is feasible to provide useful error bounds.

5.2.3 Classification Rules

Classification rule mining aims at discovering a small set of rules in a dataset to form an accurate classifier. Many approaches to perform this data mining task have been presented. Based on the solution of Crémilleux and Boulicaut (2002) we illustrate how classification rule mining can be performed in our context. Our choice is based on several advantages of this approach, such as providing the simplest rules that characterize classes w.r.t. their left-hand sides and properties avoiding rule conflicts. In the following we present some definitions from Crémilleux and Boulicaut (2002) before we show how it fits in our work.

Definition 9 (δ-strong rules): Given a set R of 0/1-valued attributes, a 0/1 relation r with attributes R, a support threshold σ, and an integer δ, a δ-strong rule on r is an association rule $X \rightarrow B$, where $S_r(X \cup \{B\}) \geq \sigma$, $F_r(X) - F_r(X \cup \{B\}) \leq \delta$, $X \subseteq R$, and $B \in R \setminus X$. ■

A δ-strong rule is violated by at most δ examples. In other words, its confidence is at least equal to $1 - (\delta / (\sigma |r|))$. Consider a classification task with $C_{1,\ldots,} C_k$ as the k class values:

Definition 10 (δ-strong rules characterizing classes): A δ-strong rule characterizing classes is a δ-strong rule with a minimal body that concludes on one class value (i.e., C_i). ■

In a typical scenario of classification rule mining, each example is associated to a unique class value. Thus, we have the following equality:

$$\sum_{i=1}^{k} F_r(C_i) = |r| \quad (26)$$

Crémilleux and Boulicaut (2002) shows that using a support threshold σ that provides a frequency threshold $\sigma|r|$ greater than δ is a sufficient condition to avoid different types of rule conflicts. A rule conflict occurs when for an unseen example two different rules predict two different classes. The reasoning is quite simple. Once we are only mining rules that allow a maximum δ number of exceptions, constraining the rule frequency to a $\sigma|r|$ greater than δ will make exceptions to be considered as infrequent, thus avoiding conflicting rules constructed upon exceptions.

Considering the DWFIST approach context, the information available in partition dimensions provides different sets of classes for the classification task. Every transaction is related to one and only one partition thus building a typical scenario in classification rule mining. Any choice of classes is valid as far as: (1) each class C_i is

related to a set of DW partitions; (2) one DW partition is not related to more than one class; and (3) the training set consisting of a set of DW partitions ρ is selected in such a way that all partitions of ρ are related to one class. These properties are summarized by:

$$\sum_{i=1}^{k} \left| \rho \cap C_i \right| = \left| \rho \right| \quad (27)$$

Once we have defined classes $C_{1,...,} C_k$ and a training set ρ, we can derive bounds on the number of exceptions τ for a rule $X \rightarrow C_i$, where X is a σ-frequent itemset, as:

$$\mathrm{LB}\,\tau_{\rho,C_i}(X) = \mathrm{LB\_}F_{\rho}(X) - \mathrm{LB\_}F_{\rho \cap C_i}(X) \qquad \mathrm{UB}\,\tau_{\rho,C_i}(X) = \mathrm{LB}\,\tau_{\rho,C_i}(X) + \mathrm{UBe}_{\rho - C_i}(X)$$

(28)

The next step is setting the δ parameter. As in our context we are dealing with incomplete information, some values of δ are not meaningful. For instance, there is no sense in trying to retrieve from a data warehouse that stores frequent itemsets with a minimum frequency of 100 transactions in the average, rules with at most 10 exceptions! Of course, there will not be enough detailed information to cope with that. Considering a set of DW partitions ρ we can obtain the lowest meaningful value for δ as:

$$\delta > \mathrm{Min}\left(\mathrm{UBe}_{\rho_i}\right), \forall \rho_i \in \rho \quad (29)$$

A superset of the δ-strong rules can be obtained checking the pairs (X, C_i) satisfying:

$$\mathrm{LB}\,\tau_{\rho,C_i}(X) \leq \delta \quad (30)$$

A general upper bound δ' on the number of exceptions of the rules in the superset of the δ-strong rules can be obtained from (28) and (30) as a function of ρ and $C_{1,...,} C_k$:

$$\delta' = \delta + \mathrm{Max}\left(\mathrm{UBe}_{\rho - C_i}\right) \quad (31)$$

Therefore, all rules belonging to the superset of δ-strong rules obtained by (30) are δ'-strong rules. This means that using a frequency threshold greater than δ' to

retrieve the frequent itemsets X is a sufficient condition to avoid the rule conflicts described in Crémilleux and Boulicaut (2002). At last, without any loss of generality, only the rules with a minimal body can be selected (Crémilleux and Boulicaut, 2002).

6 Experimental Evaluation

We performed a series of experiments to evaluate the issues discussed throughout this thesis. We concentrate on the pre-processing and loading step, Data Warehouse of Frequent Itemsets and Basic Frequent Itemset Retrieval Capabilities components. The evaluation do not consider the other DWFIST components because as far as we are able to pre-process, store and retrieve frequent itemsets in an efficient and effective way, we will be able to perform the task of the other components.

The running example of section 3.3 was evaluated in a prototype. All experiments were performed on a PC Pentium IV 1.2 GHz with 768 MB of RAM. We used Oracle 10g to implement a data warehouse identical to the example presented in section 4.4. The only worth mentioning aspects of the applied physical design are: the use of bitmap indexes for joins between dimensions and fact tables; and the use of a 1MB extent for the frequent itemset measures fact table.

In the following we first describe how the synthetic data used in our tests was created and then present the prototype implementation. Section 6.3 presents measurements obtained in our experiments and section 6.4 presents further discussion on the results.

6.1 Test Data Description

The stream data was created using the IBM synthetic market-basket data generator that is managed by the Quest data mining group. Two data streams were generated both representing a period of one week. The first data stream (Stream1) has 60 million transactions, 1000 distinct items and an average of 7 items per transaction. The transactions of the first data stream were uniformly distributed over 168 one-hour partitions (1 week). This data stream does not present strong calendar pattern behavior. The second data stream (Stream2) was created in two steps. First, a dataset was created with 50 million transactions, 1000 distinct items and an average of 7 items per transaction. Second, another dataset with 10 million transactions, 1500 distinct items and an average of 7 items per transaction was created. The 50M-transactions dataset was uniformly distributed over 140 one-hour partitions and the 10M-transactions dataset over 28 one-hour partitions. Finally, the 28 one-hour partitions of the 10M-transactions

dataset were associated to the morning period (08:00AM until 12:00PM) over the weekdays, and also to the period from 08:00AM to 17:00PM excluding lunch (12:00PM until 13:00) on the Saturday. The 140 one-hour partitions of the 50M-transactions were associated to the remaining slots completing a week. This second data stream presents strong calendar behavior over the morning period and on Saturday due to the additional items on the 10M-transactions dataset.

6.2 Prototype Description

This section provides an overview of our prototype implementation. Let us start discussing the synthetic data creation. The IBM synthetic market-basket data generator creates one single file with all synthetic transactions. Each transaction is a set of items without any additional information, such as the timestamp or the place where the transaction took place. The Batches Generator module presented in Figure 22 receives as input a file containing the transactions created by the IBM generator and distributes the transactions over the partitions specified by the user in the interface.

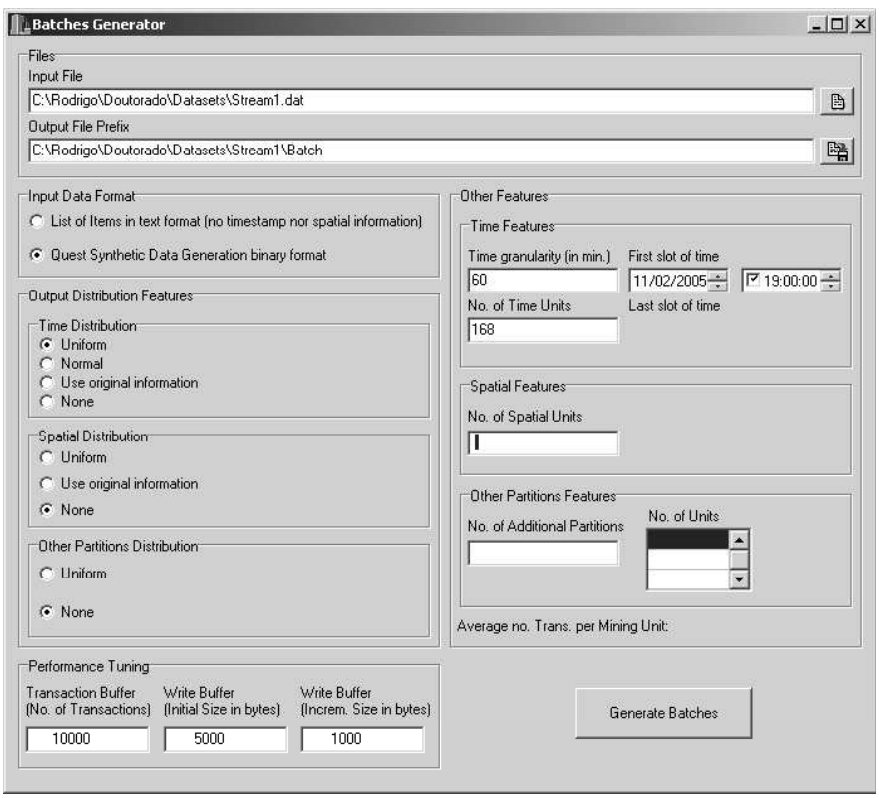


Figure 22 – Batches Generator Module

Therefore, the Batches Generator module adds information to each transaction by classifying them into different partitions. For example, in Figure 22 the transactions

will be distributed over 168 time units, where each time unit represents 1 hour (60 minutes), starting on 11/02/2005 at 19:00. The transactions can be distributed uniformly over the user defined partitions or use a normal distribution.

The next step is to mine the frequent itemsets holding on each partition. Figure 23 shows the interface of the Batches Miner module. This module receives as input the files (one per partition) created by the Batches Generator module and perform the frequent itemset mining task on each partition. Three frequent itemset mining algorithms are available: all itemsets, closed itemsets and maximal itemsets. The mining minimum support can be adjusted as well.

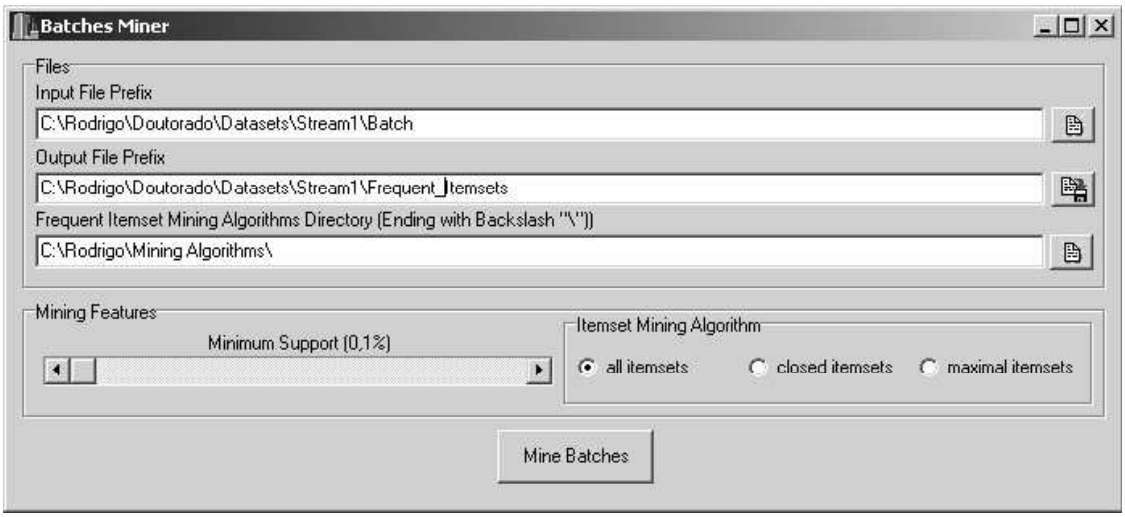


Figure 23 – Batches Miner Module

The Batches Miner module creates for each partition one file with the mined frequent itemsets and their frequency counts. One additional file with the extension .mdf that contains information about the mining task (basically the mining minimum support and the frequent itemset algorithm used) is also created. This file is the input for the DW Loader module, which will use a package implemented in PL/SQL to load the frequent itemsets into the data warehouse. The DW Loader module is shown in Figure 24. The DWFIST context refers to which data warehouse the data will be loaded. Each Data Warehouse of Frequent Itemsets must have one DWFIST context associated, which is provided together with some metadata information about the DW in the schema presented in Figure 25. The DW Loader and other modules use this metadata information about the DW schema to automatically build SQL statements. That is way the DW Loader only receives as input the DWFIST context. All other required

information is retrieved from the metadata schema. Everything that must be done is to create the DW schema and insert its description in the metadata schema.

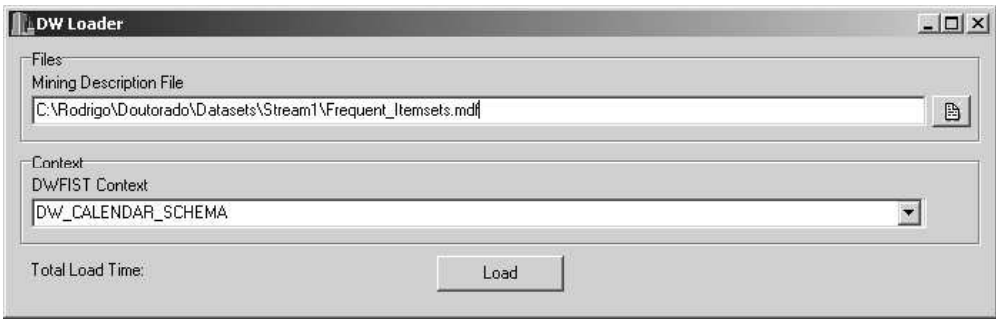


Figure 24 – DW Loader Module

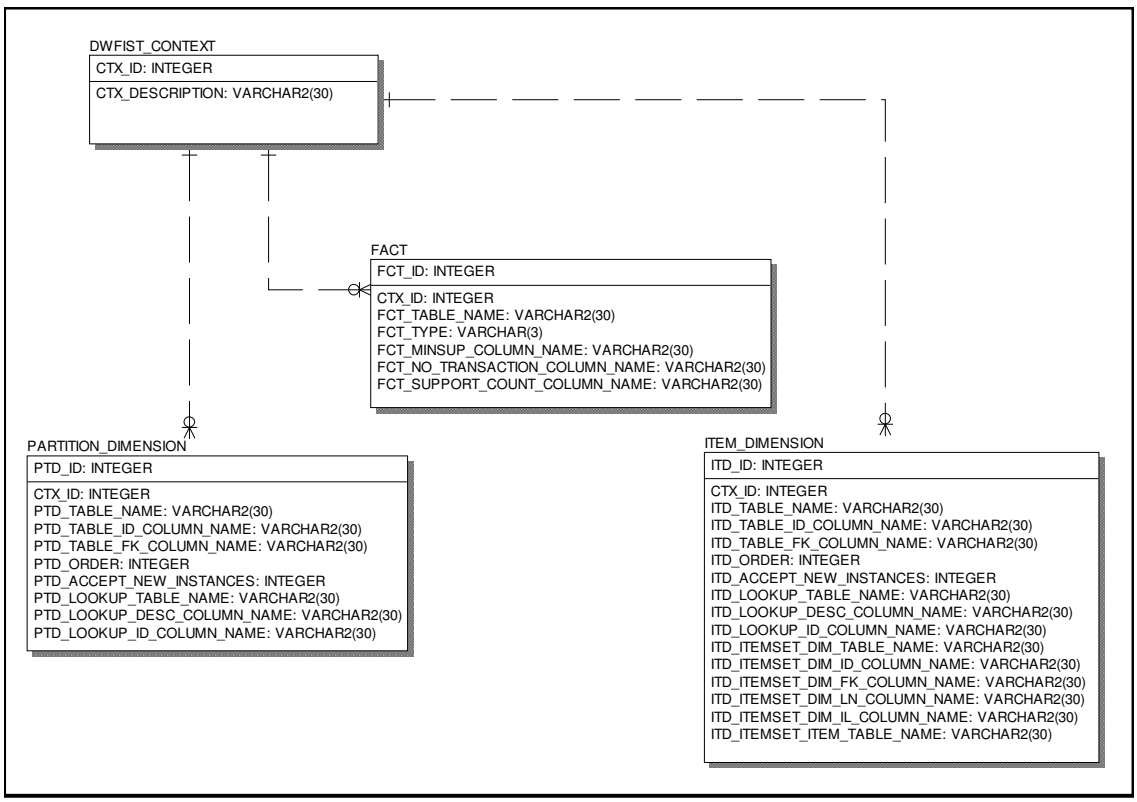


Figure 25 – Metadata schema

Once the frequent itemsets are loaded in the DW we can retrieve patterns from it. APIs for the Basic Frequent Itemset Retrieval Capabilities and Frequent Itemset Based Pattern Mining Engine components are provided. Both receive a piece of SQL statement defining constraints on the dimensions and retrieve a set of patterns from the DW. The SQL statement is completed by the Basic Frequent Itemset Retrieval Capabilities component using the information available in the metadata schema. This piece of SQL statement that specifies constraint on the dimensions can be automatically

built by an analytical tool accessing the metadata schema as the simple example presented in Appendix A. The interface presented in Figure 26 is used to provide the parameters to the APIs manually.

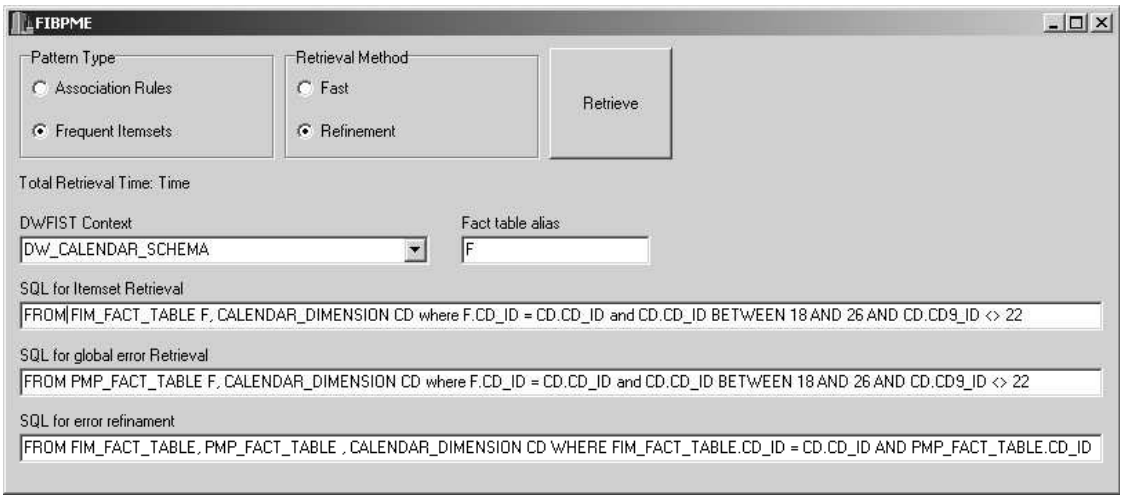


Figure 26 – Frequent Itemset Based Pattern Mining API interface

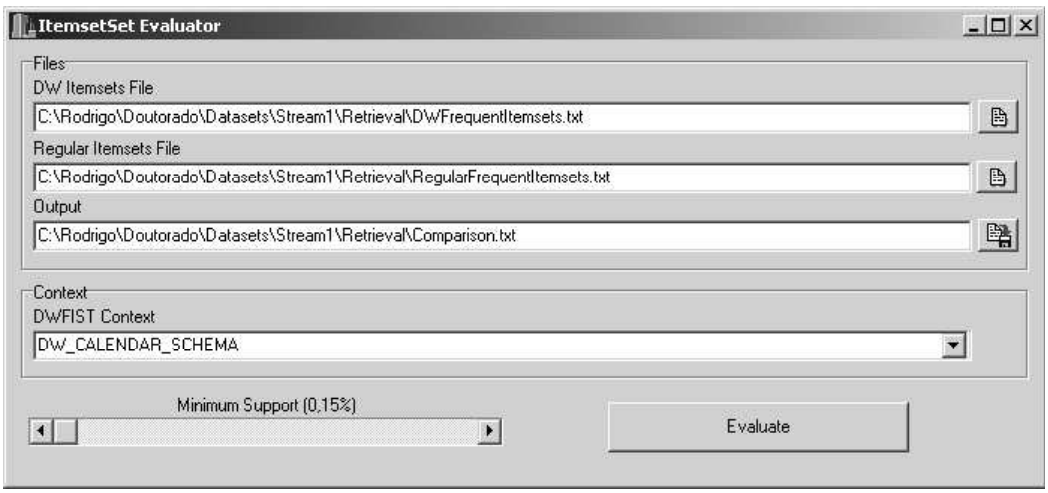


Figure 27 – ItemsetSet Evaluator Module

Finally, the set of patterns retrieved from the DW must be compared with the patterns holding on the original data. For this purpose we perform the same frequent itemset mining task (considering exactly the same set of transactions) directly on the original data. The ItemsetSet Evaluator module (Figure 27) performs the comparison receiving as input two files: one with the frequent itemsets and frequency count intervals retrieved from the DW, and another with the frequent itemsets and exact frequency counts retrieved from the original data. A query minimum support greater or equal than the mining minimum support is another input parameter.

6.3 Measurements

We focus on measurements related to pre-processing and load times, data warehouse storage requirements, and precision of the sets of frequent itemsets retrieved from the DW.

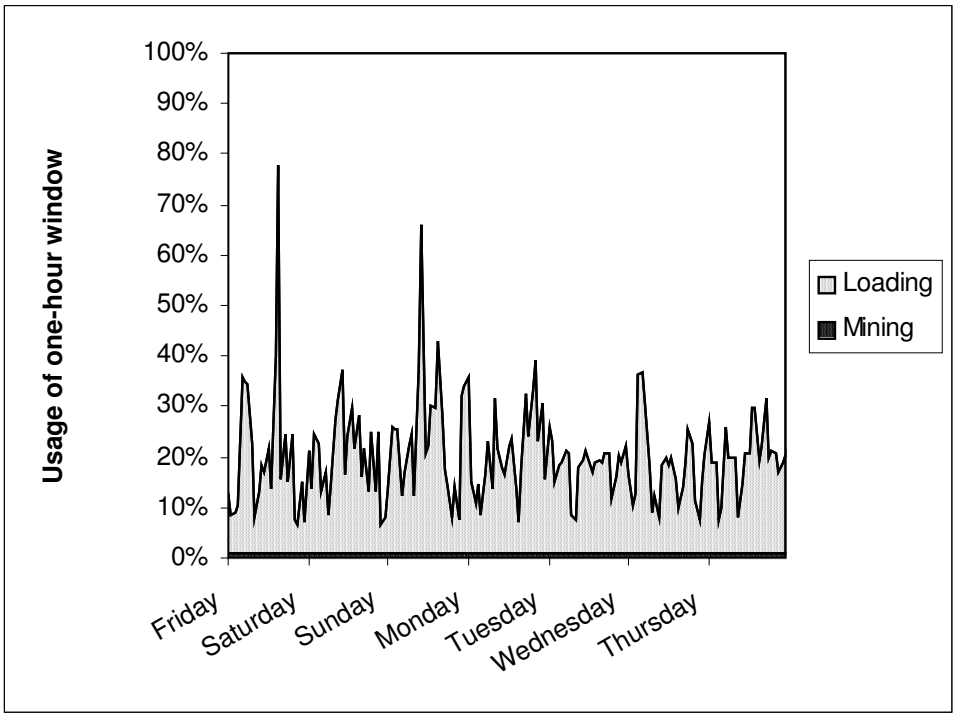


Figure 28 – Pre-processing and load time for Stream1

The pre-processing and load times are presented in Figure 28 and Figure 29. In these two graphs the Y-axis represents the percentage of the one-hour window that was required for processing whereas the X-axis represents the one-hour partitions. The pre-processing step covers the frequent itemset mining based on an optimized implementation of FP-Growth (Han et al., 2000) described in Grahne and Zhu (2003). We used a mining minimum support of 0.1% for all one-hour partitions. As this task was performed in less than one minute for all partitions, it appears only as a thin layer at the bottom of the graphs. Figure 28 and Figure 29 show that for most of the partitions pre-processing and loading is completed in less than 30% of the available time window. Even for the rare partitions with peek processing time, the pre-processing is completed within the one-hour window.

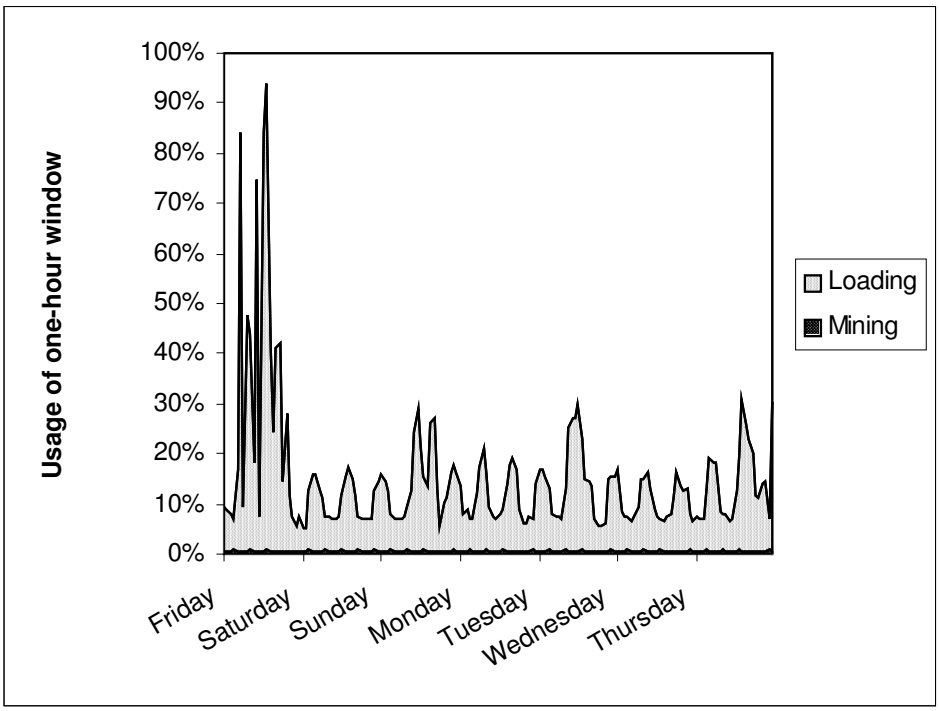


Figure 29 – Pre-processing and load time for Stream2

Figure 30 and Figure 31 present the storage requirements comparing the sizes of the streams with the corresponding DW. It is clear that the storage requirements for the streams quickly reach unmanageable levels. For our examples, only one week already represents 2 GB. On the other side, the size of the DW increases at a lower rate and is orders of magnitude lower than the size of the source data stream. For both examples, the DW size did not reach 20 MB including indexing structures.

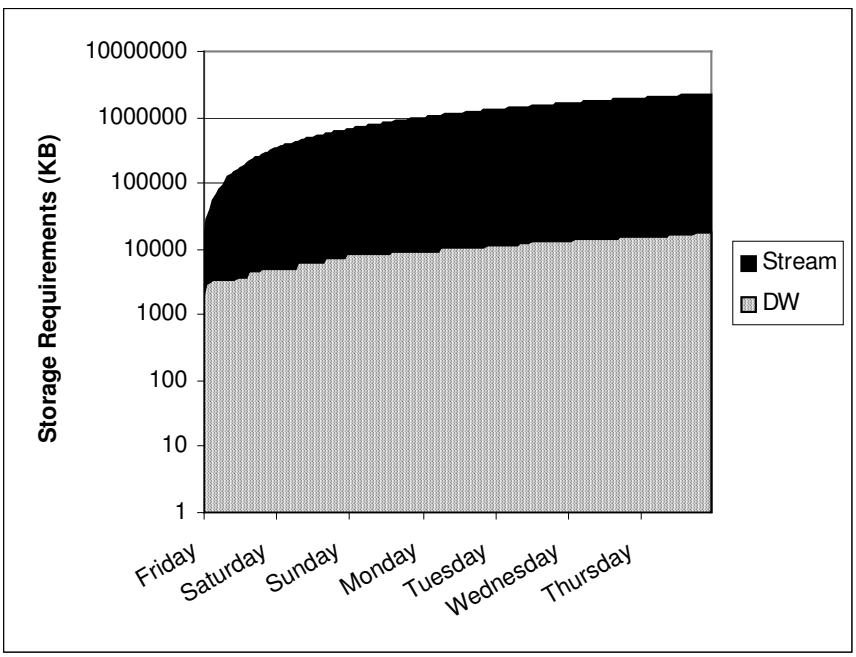


Figure 30 – Storage Requirements for Stream1

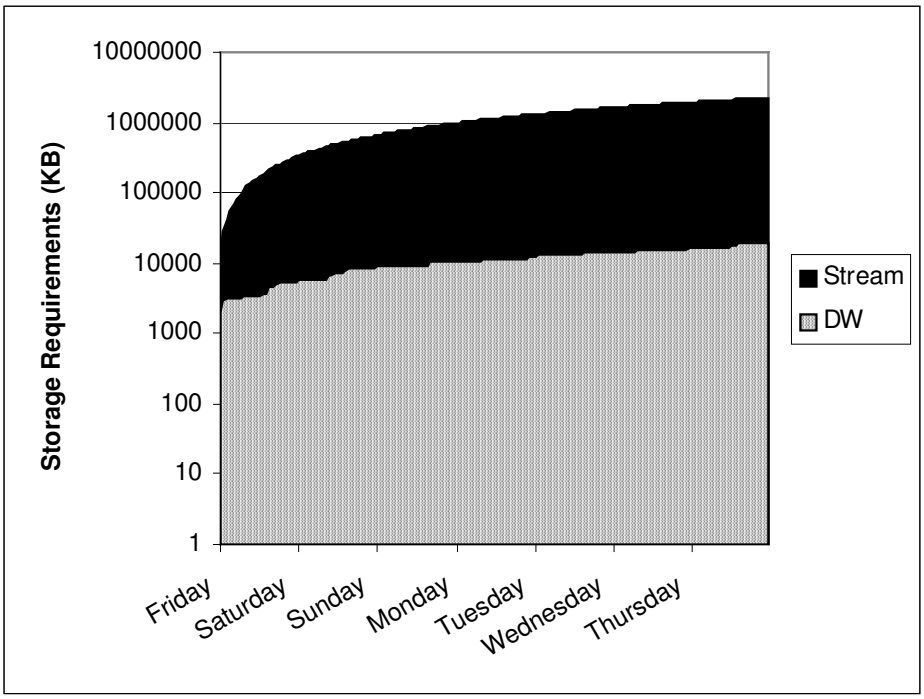


Figure 31– Storage Requirements for Stream2

The loading step is performed by a package implemented in Oracle PL/SQL. For each one-hour partition, this package first inserts a new tuple in the Calendar Dimension and in the Partition Mining Properties fact table. The latter one stores the minimum support used for mining (0.1%) and the number of transactions on the stream over the corresponding period. Afterwards, each itemset mined as frequent in the pre-processing step is loaded with its frequency count. For each itemset, the package checks if it already exists in the Itemset Dimension. For those that already exist, inserting a new tuple in the Frequent Itemset Measures fact table is sufficient. Otherwise, a new tuple in the Itemset Dimension is required before inserting the new fact. Also, the items belonging to the new itemset that are new for the Product Dimension must be created. Finally, the items are associated to the new itemset (through an NxM relationship table).

The detailed analysis of the loading step reveals that it is mainly influenced by three factors: the number of itemsets being loaded, the number of new itemsets being created, and the need for extending the physical storage area. The processing peeks of Figure 28 and Figure 29 take place during an extension of the physical storage area. Also, as the data warehouse dimensions are initially empty, many new itemsets must be created during the load of the first partitions. The high processing peeks are related to periods where the three factors presented a negative influence.

The last part of our experiments presents measurements related to the precision of the set of frequent itemsets retrieved from the DW. We retrieved frequent itemsets for three different calendars: whole Saturday (00:00 to 24:00); Monday, Wednesday and Friday from 08:00 to 17:00; and morning periods of weekdays (08:00 to 12:00). These calendars are related to the strong calendar behavior introduced in Stream2 in different ways. "Saturday" comprises 24 partitions from which 8 come from the 10M-transactions dataset. "Monday, Wednesday and Friday from 08:00 to 17:00" presents 12 out of 27. "Morning periods of weekdays" is related only to partitions from the 10M-transactions dataset. They were defined in order to measure in Stream2 the effect of a strong calendar behavior on the precision of the result. We also executed the same queries on Stream1, which does not present a calendar behavior.

The set of frequent itemsets was retrieved in less than 5 seconds in all three cases. The corresponding mining task took about 50 seconds when performed directly on the original data thus affirming the reduction of computational effort.

The precision of the retrieved set of frequent itemsets is measured according to two aspects: number of false positives and frequency relative error. In order to obtain these measurements, we performed a regular frequent itemset mining task for each query directly on the corresponding stream transactions. The answer retrieved from the DW was compared with the exact answer.

As the results for the three queries on Stream1 presented the same behavior, we present only the results for one of them in Figure 32. The difference between the two bars on the upper graph corresponds to the number of false positives. Note that with a query minimum support slightly greater than the mining minimum support (0.1%) it is already possible to retrieve the exact set of frequent itemsets. The graph on the bottom of Figure 32 presents the frequency relative error measurements. The Worst-case corresponds to the value obtained applying equation (15). The Maximum DW and Average DW are computed calculating the error as the difference between the approximate frequency and the frequency upper bound thus corresponding to the tightness of the frequency interval. The Maximum Real and Average Real use the error as the difference between the approximate frequency and the exact real frequency retrieved directly from the stream. The error drops quickly to zero for query minimum support values slightly greater than the mining minimum support.

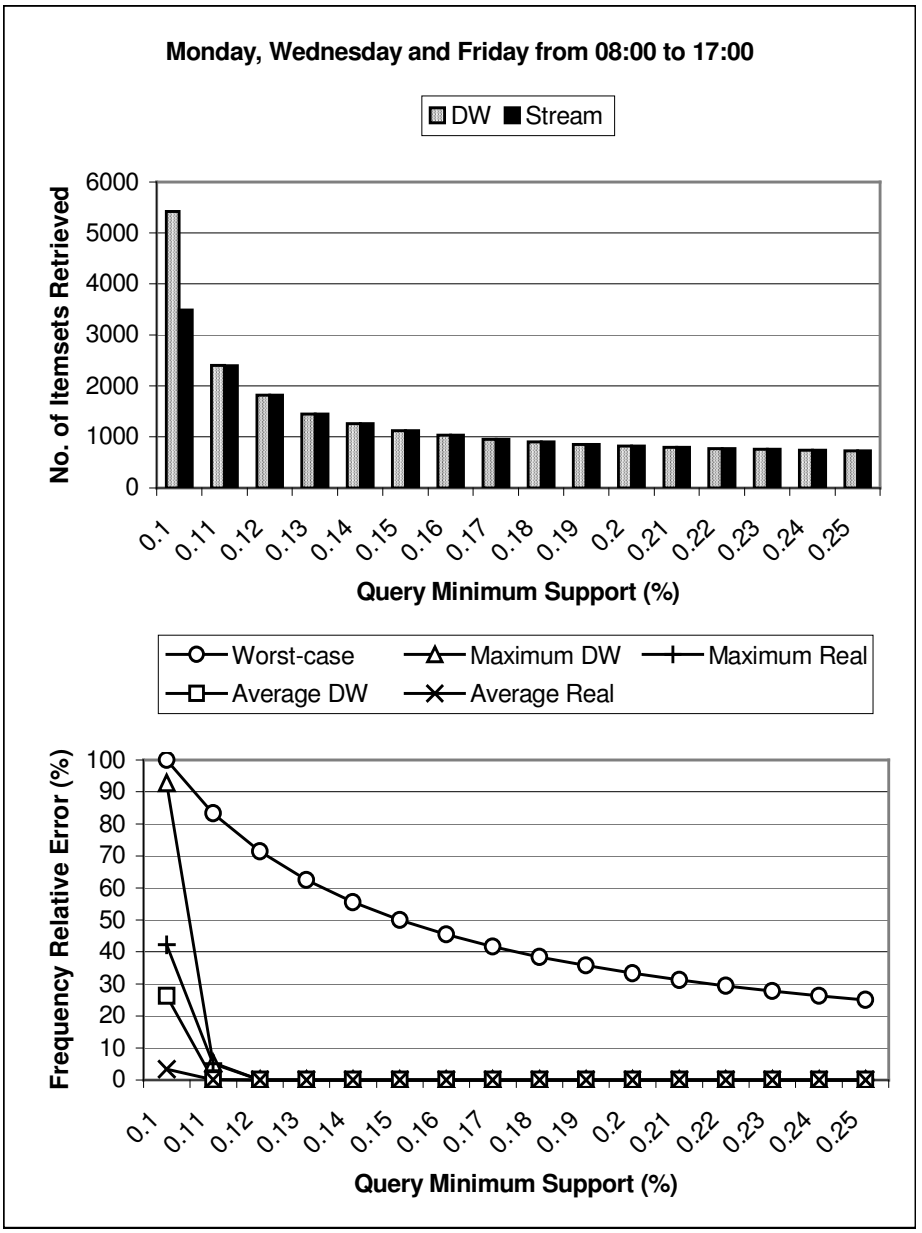


Figure 32 – Precision measurements on Stream1

Figure 33 presents the same graphs for the three queries executed on Stream2. Once again, for values of the query minimum support slightly greater than the mining minimum support the result already presents a good precision. For example, with a query minimum support of 0.12% the Average DW frequency relative error already drops below 10%. It is especially important to observe the low values of the Average DW. As it does not require knowledge about the exact real frequency it represents the average precision guarantee that will be provided for a user querying the DW. Also, the query for the morning period of weekdays presents the same behavior as the queries executed on Stream1.

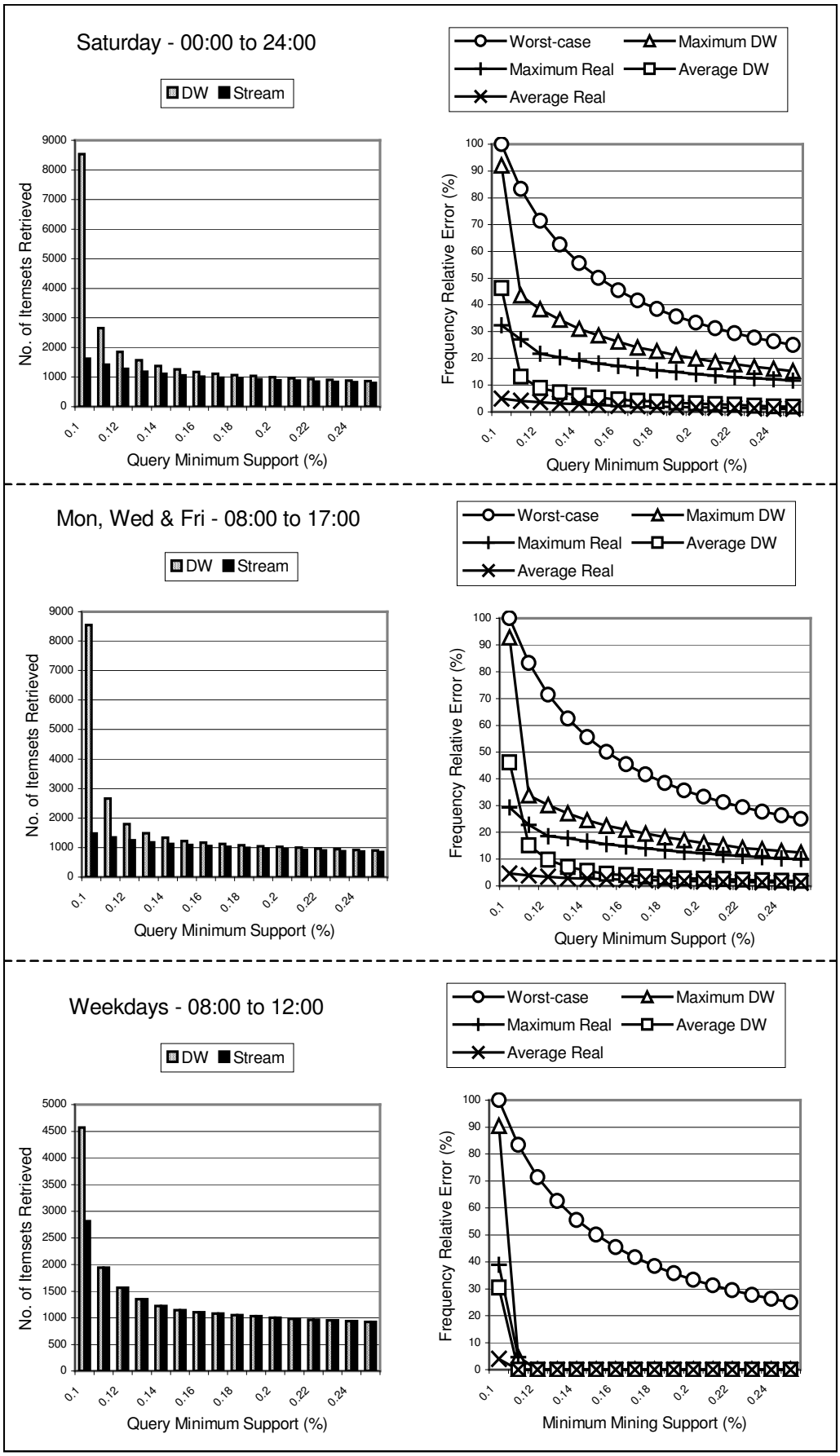


Figure 33 – Precision measurements on Stream2

6.4 Further Discussion on the Experimental Results

Some specific aspects related to the experimental results require some further discussion. Considering the three main factors that influence the time required for the loading step (namely the number of itemsets being loaded, the number of new itemsets being created, and the need for extending the physical storage area) it is not likely that all three factors will negatively influence the load processing of a small number of consecutive periods. For example, consecutive periods normally should not require an extension of the physical storage area. Moreover, if they happen to do, it is most likely that the size of the extent is not properly set. This suggests that it should be possible to compensate some processing peeks out of the window within a few consecutive periods. This would lead to a less conservative definition of the DW granule. Of course, an increase in the storage requirements, due to the need of a buffer to keep the itemsets with a delayed load, must also be taken into account.

Another aspect relates to the relationship between an error in the frequency count retrieved from the DW and a potential local behavior on some subset of the partitions being queried. The greater the error in the frequency count is, the bigger the chances of reporting it as a false positive. At the same time, the greater the error in the frequency count is, the stronger local behavior it suggests. Property 6 tell us that this frequent behavior on some subset of the partitions being queried is at least as strong as the patterns we are requesting. In fact, it gets stronger as the query minimum support moves far away from the mining minimum support. This allows us to come to an interesting conclusion: excluding the query minimum support values that are too close to the mining minimum support, it is interesting to investigate the local behavior of the reported frequent itemsets that present a large frequency count interval, i.e., an imprecise frequency count. An analytical tool can, for example, easily retrieve from the DW the partitions where the frequency count of a specific itemset is known. Looking at the partitions with the respective frequency counts can reveal a pattern in the local behavior.

7 Conclusions and Future Work

We proposed an approach concerned with supporting the analysis and exploration of frequent itemsets and derived patterns, e.g. association rules, in transactional datasets. This chapter concludes this thesis presenting a summary of the work and some future research directions.

7.1 Summary

This thesis proposed the Data Warehouse of Frequent Itemsets Tactics (DWFIST) approach focusing on two issues: (1) provide a standard modeling for data warehouses of frequent itemsets and (2) mine calendar-based frequent itemsets on data streams. A data warehouse plays a central role in our approach. It stores frequent itemsets holding on different partitions of the original transactions retaining information for future analysis. We discussed staging area tasks that must be performed. Time constraints and storage requirements issues were considered in a data stream scenario. A tradeoff between analysis capabilities and the time window available for pre-processing must be taken into account for defining the DW granule. A running example for mining calendar-based frequent itemsets was provided. This example was implemented in a prototype and measurements were presented to illustrate how the proposed approach works in practice. The standard modeling presented for frequent itemset warehousing provides a standardized logical view upon which analytical and explorative tools can independently be developed. A set of properties of the data warehouse of frequent itemsets allows for a flexible retrieval of frequent itemsets holding on any set of DW partitions. Using a running example, we exemplified how the flexibility provided for pattern retrieval can be used for calendar-based pattern mining. Finally, the retrieval of frequent itemset based patterns from the DW was exemplified by association rules, Boolean rules and classification rules.

Current frequent itemset mining algorithms can cope with data stream time constraints when processing size-limited batches of transactions. For the loading step, three major factors were pointed out as influencing the load time: the number of itemsets being loaded, the number of new itemsets being created, and the need for extending the physical storage area. The DW granule, the mining minimum support and

DBMS parameters (e.g. extent size) provide ways of tuning the performance of the load step.

The storage requirements are directly related to the number of frequent itemset frequencies being stored. The DW granule and the mining minimum support play an important role. The proper setting of these parameters allows to keep the storage requirements at manageable levels.

The completeness of the set of frequent itemsets retrieved from the DW can be guaranteed and a theoretical worst-case bound for the relative error of the itemset frequency retrieved from the DW is provided.

The precision of different sets of frequent itemsets retrieved from the DW was measured. The number of false positives decreases rapidly as the query minimum support moves away from the mining minimum support. Moreover, the tightness of the frequency interval presented an average behavior much better than the theoretical worst-case.

7.2 Future Work

As future directions we would like to highlight the development of analytical and explorative tools to exploit large sets of patterns. Such tools can rely on the standardized logical schema provided by our approach to access and retrieve patterns. Evaluating the applicability and usage of different condensed representations for representing the frequent itemsets on a DW partition is an interesting topic as well. The use of condensed representations for frequent itemsets, such as closed frequent itemsets, can reduce the storage requirements and also bring benefits to the DW load process. However, an overhead will be introduced for the retrieval of patterns from the DW. This tradeoff must be further explored and analyzed. At last, deploying other frequent itemset based patterns is an additional issue for future research.

8 References

Agarwal, R. C., Aggarwal, C. C., and Prasad, V. V. V. (2001). "A Tree Projection Algorithm for Generation of Frequent Item Sets". Journal of Parallel and Distributed Computing 61(3): 350-371.

Agrawal, R., Imielinski, T., and Swami, A. (1993). "Mining Association Rules between Sets of Items in Large Databases". Proc. ACM SIGMOD Conf., pp. 207-216, Washington.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. (1996). "Fast Discovery of Association Rules". In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramaswamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp. 307-328, AAAI/MIT Press.

Agrawal, R., and Srikant, R. (1994). "Fast Algorithms for Mining Association Rules in Large Databases". Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499, Santiago de Chile, Chile.

Ahmed, S., Coenen, F. and Leng, P. (2004). "A Tree Partitioning Method for Memory Management in Association Rule Mining". Proc. DAWAK 2004 conference, Zaragosa, September, LNCS 3181, Springer, pp. 331-340.

Ayan, N. F. (1999). "Updating Large Itemsets With Early Pruning". M.Sc. Thesis, Bilkent University, Ankara, Turkey.

Bayardo Jr, R. J. (1998). "Efficiently Mining Long Patterns from Databases". Proceedings of the 1998 SIGMOD Conference, pp. 85-93.

Bayardo, R. J. and Agrawal, R. (1999). "Mining the Most Interesting Rules". In Proc. of the Fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 145-154.

Beyer, K., and Ramakrishnan, R. (1999). "Bottom-up computation of sparse and iceberg cubes". In Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99), pp. 359-370.

Boulicaut, J. (2004). "Inductive databases and multiple uses of frequent itemsets: the cInQ approach". In Database support for Data Mining Applications, R. Meo et al. Eds., LNCS 2682. pp. 3-26.

Boulicaut, J., and Bykowski, A. (2000). “Frequent closures as a concise representation for binary data-mining”. In Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp. 62-73.

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). “Dynamic Itemset Counting and Implication Rules for Market Basket Data”. Proceeding of the 1997 SIGMOD Conference, pp. 255-264.

Burdick, D., Calimlim, M., and Gehrke, J. E. (2001). “MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases”. Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, April.

Crémilleux B., Boulicaut J-F. (2002). “Simplest Rules Characterizing Classes Generated by delta-Free Sets”. Proceedings of the twenty-second Annual International Conference Knowledge Based Systems and Applied Artificial Intelligence (ES 02), Springer, pp. 33-46, Cambridge, United-Kingdom, December.

Demaine, E. D., L´opez-Ortiz, A., and Munro, J. I. (2002). “Frequency estimation of internet packet streams with limited space”. In Proc. of the 10th Annual European Symposium on Algorithms.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1998). “Advances in Knowledge Discovery and Data Mining”. AAAI Press.

Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T. (1996). “Data Mining using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization”. In Proc. of the 1996 ACM-SIGMOD Int’l Conf. on the Management of Data, pp. 13-23.

Ganti, V., Gehrke, J., and Ramakrishnan, R. (1999). “Mining Very Large Databases”. IEEE Computer, 32(8), pp. 38-45,.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P.S. (2003). “Mining Frequent Patterns in Data Streams at Multiple Time Granularities”. H. Kargupta et al. Eds. Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT Press.

Gouda, K., and Zaki, M. J. (2001). “Efficiently Mining Maximal Frequent Itemsets”. Proceeding of the 2001 ICDM Conference, pp. 163-170.

Grahne, G., and Zhu, J. (2003). “Efficiently Using Prefix-trees in Mining Frequent

Itemsets". Proceeding of the First IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, November.

Grahne, G., and Zhu, J (2003b). "High Performance Mining of Maximal Frequent Itemsets". Proceedings of the 6th International Workshop on High Performance Data Mining.

Han, J. (1997). "OLAP Mining: An Integration of OLAP with Data Mining". Proceedings of the 1997 IFIP Conference on Data Semantics (DS-7), pp. 1-11, Leysin, Switzerland.

Han, J., and Kamber, M. (2001). "Data mining: concepts and techniques". Academic Press.

Han, J., Pei, J., and Yin, Y. (2000). "Mining frequent patterns without candidate generation". In Proceeding of the 2000 SIGMOD Conference, pp. 1-12, Dallas, Texas, May.

Hidber, C. (1999). "Online Association Rule Mining". Proceedings of the ACM SIGMOD International Conference on Management of Data.

Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000). "Algorithms for Association Rule Mining - A General Survey and Comparison". SIGKDD Explorations 2(1): pp. 58-64.

Holsheimer, M., Kersten, M. L., Mannila, H., and Toivonen, H. (1995). "A Perspective on Databases and Data Mining", Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pp. 150-155, Montreal, Canada.

Houtsma, M., and Swami, A. (1995). "Set-oriented mining of association rules". Proceedings of the 11th International Conference on Data Engineering, pp. 25-34.

Imielinski, T., and Mannila, H. (1996). "A database perspective on knowledge discovery". Communications of ACM, vol. 39, pp. 58-64.

Inmon, W. H. (2002). "Building the Data Warehouse**",** Wiley Computer Publishing, John Wiley & Sons, Inc. u.a. New York, Third Edtion.

Karp, R. M., Papadimitriou, C. H., and Shenker, S. (2003). "A simple algorithm for

finding frequent elements in streams and bags". ACM Trans. Database Systems.

Kimball, R., and Ross, M. (2002). "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modelling", Wiley Publishers, second edition, ISBN 0471200247.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. I. (1994). "Finding Interesting Rules from Large Sets of Discovered Association Rules". Proceeding of the Third International Conference on Information and Knowledge Management, pp. 401-407.

Li, Y., Ning, P., Wang, X. S., and Jajodia, S. (2001). "Discovering calendar-based temporal association rules". Proceedings of the 8th International Symposium on Temporal Representation and Reasoning, pp. 111-118.

Liu, B., Hsu, W., and Ma, Y. (1998). "Integrating classification and association rule mining." In Proceedings KDD'98, pp. 80–86, AAAI Press, New York, USA.

Manku, G., and Motwani, R. (2002). "Approximate frequency counts over data streams". In Proc. Int. Conf. Very Large Data Bases (VLDB'02), 346.357.

Mannila, H., and Toivonen, H. (1996). "Multiple Uses of Frequent Sets and Condensed Representations". In Proceedings KDD'96, pp. 189-194, AAAI Press, Portland.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1994). "Efficient Algorithms for Discovering Association Rules". KDD Workshop, pp. 181-192.

Monteiro, R. S., Zimbrao, G., Schwarz, H., Mitschang, B., and Souza, J. M. (2005a). "Building the Data Warehouse of Frequent Itemsets in the DWFIST Approach". Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems, pp. 294-303, LNCS 3488, Saratoga Springs, NY, USA.

Monteiro, R. S., Zimbrao, G., Schwarz, H., Mitschang, B., and Souza, J. M. (2005b). "DWFIST: The Data Warehouse of Frequent Itemsets Tactics Approach". Book chapter to appear in *Processing and Managing Complex Data for Decision Support*, Idea Group Inc.

Monteiro, R. S., Zimbrão, G., and Souza, J. M. (2003). "An Analytical Approach for

Handling Association Rule Mining Results". Proceedings of the 2003 Australasian Data Mining Workshop, (AusDM'03), Canberra, Australia, December.

Orlando, S., Palmerini, P., Perego, R., and Silvestri, F. (2002). "Adaptive and Resource-Aware Mining of Frequent Sets". Proceedings of the IEEE International Conference on Data Mining, pp. 338-345. Maebashi City, Japan, December.

Özden, B., Ramaswamy, S., and Silberschatz, A. (1998). "Cyclic association rules". In Proc. of the 14th Int'l Conf. on Data Engineering, pp. 412–421.

PANDA Project (2004). http://dke.cti.gr/panda/.

Park, J. S., Chen, M., and Yu, P. S. (1995). "An Effective Hash-Based Algorithm for Mining Association Rules". Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 175-186, San Jose, California.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). "Discovering Frequent Closed Itemsets for Association Rules". In Proc. ICDT Int. Conf. Database Theory, pp. 398-416.

Pei, J.; Han, J.; and Mao, R. (2000). "CLOSET: An efficient algorithm for mining frequent closed itemsets". In Proc. 2000 ACM-SIGMOD Int. Workshop Data Mining and Knowledge Discovery (DMKD'00), 11.20.

Ramaswamy, S., Mahajan, S., and Silberschatz, A. (1998). "On the discovery of interesting patterns in association rules". In Proc. of the 1998 International Conference on Very Large Data Bases, pp. 368–379.

Sathe, G., and Sarawagi, S. (2001). "Intelligent rollups in multidimensional OLAP data". Proceedings of the 2001 International Conference on Very Large Data Bases.

Savasere, A., Omiecinski, E., and Navathe, S. B. (1995). "An Efficient Algorithm for Mining Association Rules in Large Databases". Proceedings of the 21th International Conference on Very Large Data Bases, pp. 432-444, Zurich, Switzerland.

Seno, M., and Karypis, G. (2001). "LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint". In Proceeding of the IEEE International Conference on Data Mining, pp. 505-512, San Jose,

California.

Toivonen, H. (1996). “Sampling Large Databases for Association Rules”. Proceedings of the 22th International Conference on Very Large Data Bases, pp. 134-145, Bombay, India.

Tryfona, N., Busborg, F., and Christiansen, J. G. B. (1999). “starER: A Conceptual Model for Data Warehouse Design”. Proc. Int. Workshop on Data Warehousing and OLAP, pp. 3-8.

Wang, H.; Yang, J.; Wang, W.; and Yu, P. S. (2002). “Clustering by pattern similarity in large data sets”. In Proc. ACM-SIGMOD Int. Conf. on Management of Data, pp. 418-427.

Zaki, M. J. (1999). “Parallel and distributed association mining: A survey”. IEEE Concurrency, 7(4), pp. 14-25.

Zaki, M. J., and Hsiao, C. J. (2002). “CHARM: An efficient algorithm for closed itemset mining”. In Proc. 2002 SIAM Int. Conf. Data Mining, pp. 457-473.

Zaki, M. J., Parthasarathy, S., Ogihara, M., and Li, W. (1997). “New Algorithms for Fast Discovery of Association Rules”. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 283-286, California, USA.

Appendix A – Advanced Analytical Tools

The Advanced Analytical Tools component comprises analytical and explorative tools that can be built on top of the other components. We present some examples of such tools to motivate the use of the approach as a framework to develop new mining and analytical tools.

A.1 OLAP Tools for Pattern Analysis

New tools providing a user interface similar to current OLAP tools can be developed in order to exploit the dimensional view of the patterns provided by the data warehouse model. Such tools can provide a cube of patterns (pattern cube). One important difference between a cube of patterns and a traditional data cube is that the most granular information on a pattern cube may fit into different dimension values for the same dimension. For instance, if we exploit frequent itemsets using a cube view and one dimension of the cube represents products, a frequent itemset related to products A and B will fit into more than one specific product. Due to this basic difference, the similarity between OLAP tools for pattern analysis and regular OLAP tools does not go much far away from the interface level. The semantics of OLAP operations (drill-down, rollup, slicing, dicing, etc.) in this new scenario have to be reevaluated and redefined when required.

A pattern cube cell corresponds to the set of patterns related to the corresponding dimension instances. It should be possible to visualize different aggregate values representing the set, from simple counting to more sophisticated interestingness measures. The pattern set visualization must also be possible, varying from simple listings to more elaborate visualization techniques (Han, 1997) (Klemettinen et al., 1994). As the sets of different cells are not disjoint, some mechanism to separate explored patterns from unexplored patterns must be provided. One example is to present separate measures for the explored and unexplored sets. The possibility to save and restore the exploration context turns out to be important as well.

Using interesting measures (Bayardo and Agrawal, 1999) as dimensions in a pattern cube can also be useful to depict the patterns into ranges. Also, crossing twice the same dimension can provide interesting insights. For instance, using a product

dimension twice in a pattern cube containing association rules will provide an overview of the correlation between different products.

We present in Figure 34 and Figure 35 an example of a simple prototype that provides an interface for retrieving and visualizing a pattern cube. This prototype accesses the metadata schema in Figure 25 and automatically builds the pieces of SQL statement specifying constraints on the dimensions. The user can select any attribute of a partition dimension and assign it to an axis in the pattern cube. The type of pattern (association rules or frequent itemsets) and its parameters (e.g. query minimum support and minimum confidence) can be specified. Each cell in the cube corresponds to one query submitted to the DW. The number in each cell corresponds to the number of patterns retrieved. It took about one minute to retrieve the information in Figure 34. The corresponding operation was also performed directly on the original data and took about 35 minutes.

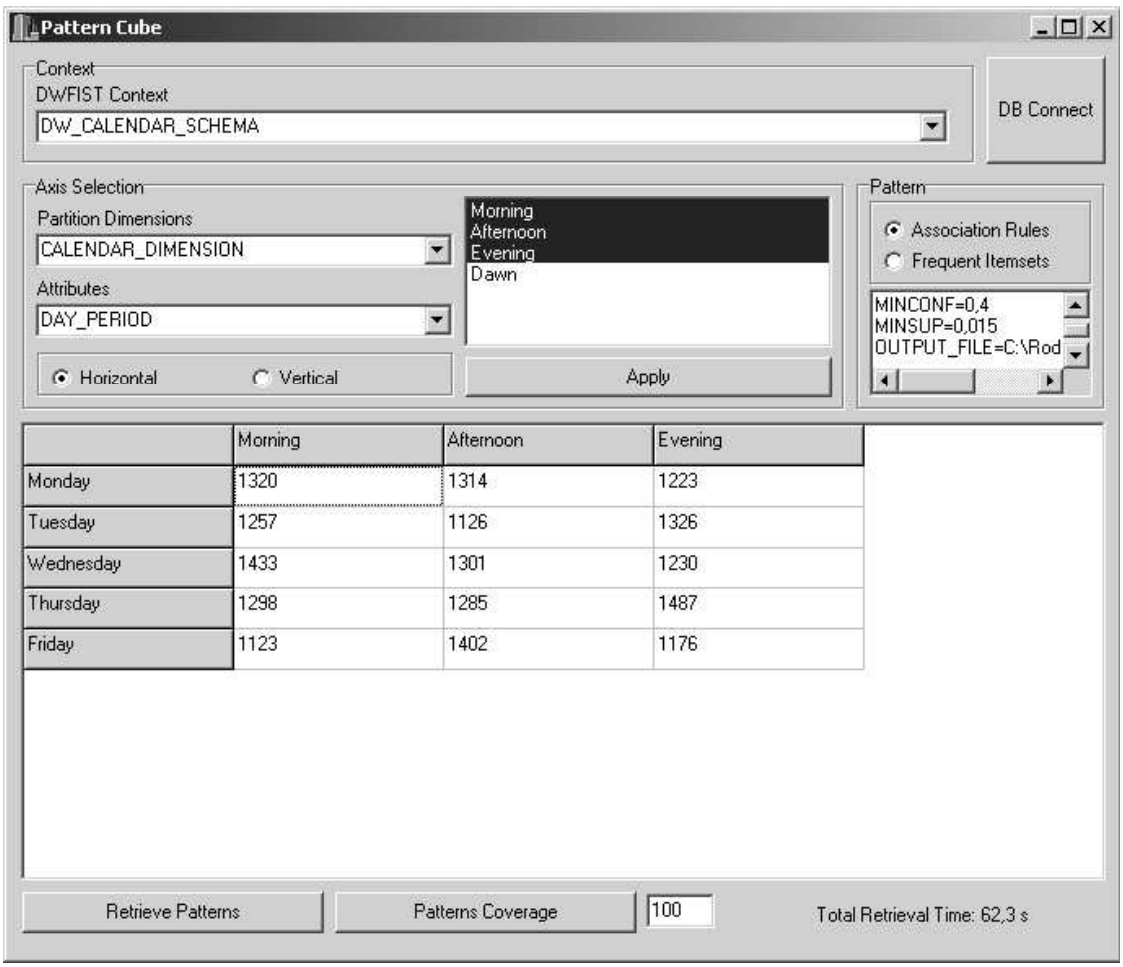
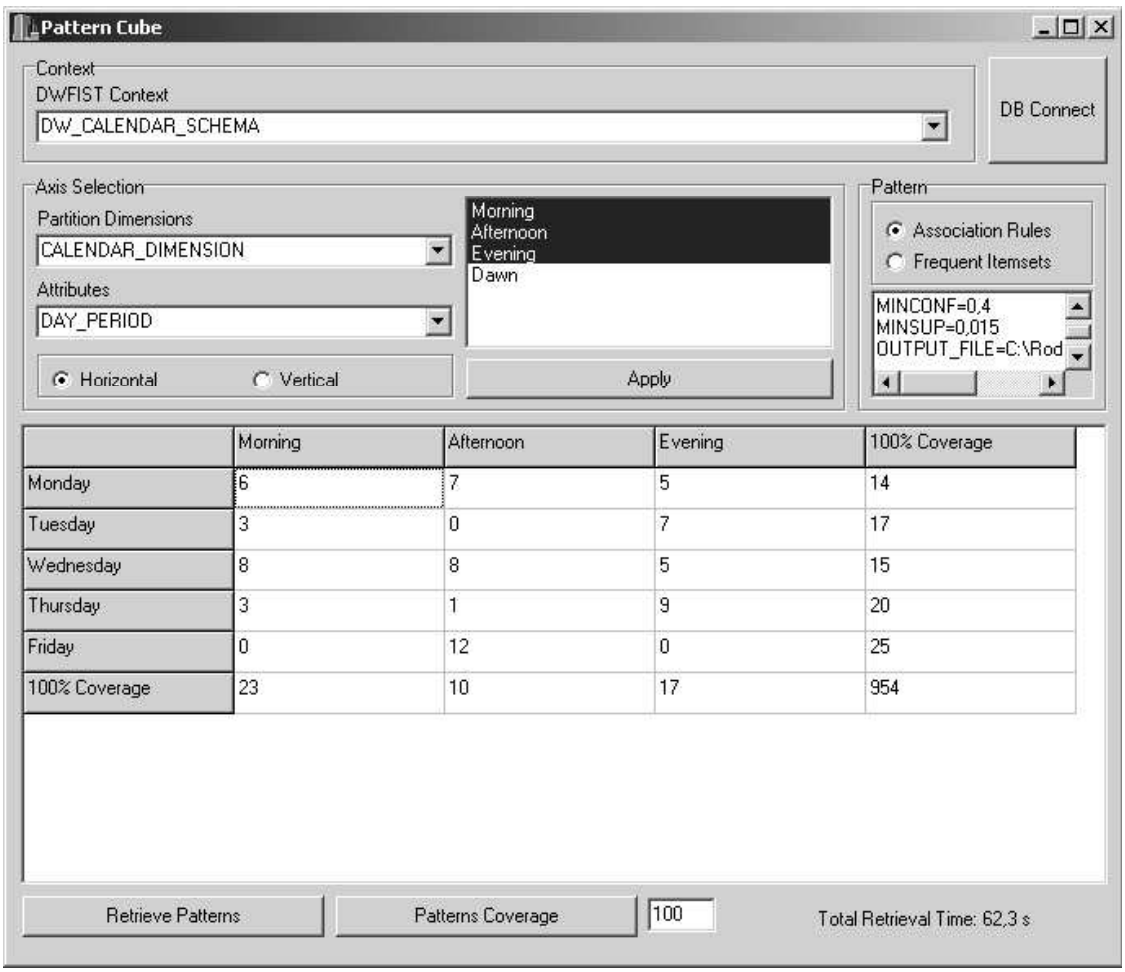


Figure 34 – Pattern Cube Prototype

After retrieving the pattern cube it is important to provide functionalities to aid the analyst in exploring the result. Figure 35 presents a simple example depicting the patterns using a coverage measure. In this example, a coverage of 100% was selected. The effect on the pattern cube is that all patterns that hold on 100% of the cells are moved to the cell in the intersection of the "100% Coverage" column and row, which presents the number 954. The cell in the intersection of the "100% Coverage" column with the "Monday" row, for example, presents the patterns that hold on 100% of the cells in the "Monday" row excluding those that hold on 100% of the cells. The patterns that remain on the original cells correspond to patterns that do not present a coverage of 100% in a column nor in a row. This procedure is useful to separate patterns expressing local behavior from those expressing global behavior.



	Morning	Afternoon	Evening	100% Coverage
Monday	6	7	5	14
Tuesday	3	0	7	17
Wednesday	8	8	5	15
Thursday	3	1	9	20
Friday	0	12	0	25
100% Coverage	23	10	17	954

Figure 35 – Example of applying a coverage measure

A.2 Automatic Exploration Methods

This class of methods is concerned in performing an unsupervised search over some data (patterns, in our context) trying to identify some interesting feature. One example is provided in Sathe and Sarawagi (2001). “The proposed operator can automatically generalize from a specific problem case in detailed data and return the broadest context in which the problem occurs.” (Sathe and Sarawagi, 2001). The idea of this operator can be applied in our context. As an example, the method can be adapted in order to identify the broadest context where an association rule is valid. Needless to say how significant this operation is when performing an analysis.

Automatically identifying local patterns is another interesting example. Calendar-based association rules (Li et al., 2001) is one example for such a local pattern. Providing efficient algorithms to explore the dimensional structure of the data warehouse of frequent itemsets in order to identify local patterns is an important issue.