A Retail Ontology: Formal Semantics and Efficient Implementation

by

Maryam Fazel Zarandi

A thesis submitted in conformity with the requirements
for the degree of Masters of Science
Graduate Department of Computer Science
University of Toronto

Abstract

A Retail Ontology: Formal Semantics and Efficient Implementation

Maryam Fazel Zarandi

Masters of Science

Graduate Department of Computer Science

University of Toronto

2007

In today's competitive business environment, the ability to make effective decisions is of central importance to an organization's survival. In order to be successful, a modern enterprise should take advantage of all available data. Business intelligence systems can help provide the means to transform the available data into information and derive specific and timely knowledge about the domain.

The focus of this research is on building a customer-centric business intelligence system applied to retail. This work describes a retail ontology that can automatically deduce answers to retail queries based upon the system's general knowledge of online retailing and actual data. To be applicable in real world situations, the system should be able to deal efficiently with the huge amounts of data present in retail environments. To this end, this work also introduces a technique for reasoning efficiently with large datasets using state-of-the-art theorem provers or reasoners and existing database technology.

Dedication

This thesis is dedicated to my parents, Susan Bastani and Mohammad Hossein Fazel Zarandi, who taught me that the best kind of knowledge to have is that which is learned for its own sake.

Acknowledgements

There are many people to whom I owe a debt of thanks for their support over the last two years. First, I would like to express sincere gratitude to my research advisor, Professor Mark S. Fox, for providing inspiration, guidance, and support during this research and the preparation of this thesis, and for teaching me how to be a good researcher.

I thank Professor Michael Gruninger for his time and valuable comments which have greatly improved and clarified this work. I am also grateful to Scott Sanner and Professor Sheila McIlraith for their wonderful comments along the way. Furthermore, I am also thankful to all members of Novator Systems Ltd. for their support and beneficial comments.

I would also like to acknowledge the financial assistance provided by the Department of Computer Science at the University of Toronto.

Last but not least, many thanks are due to my family for their valuable moral support especially during times of frustration.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

In today's competitive business environment, the ability to make effective decisions is of central importance to an organization's survival. Companies need to continually assess and redirect their actions in order to stay on top of the markets they choose to serve. As marketing becomes more customer-centric, the accuracy of decisions with regards to which potential customers to engage in relationships with and how to retain a satisfied customer also gain importance.

In order to be successful, a modern enterprise should be able to take advantage of all available data. Business intelligence (BI) systems can help provide the means to transform the available data into information and derive specific and timely knowledge about customers, products and markets which in turn can help boost profits, reduce costs and support better and more effective management. BI is defined as "the acquisition, interpretation, collation, assessment, and exploitation of business-related information" [Chung et al., 2003] with the objective of supporting sound decision making [Marshall et al., 2004]. With the constantly increasing volume of data, both internal and external to the enterprise, gathering business intelligence can become a challenge.

The focus of this research is on building a customer-centric business intelligence system applied to (online) retail. This work describes a retail ontology that can automatically deduce answers to retail queries based upon the system's general knowledge of online retailing and actual data. An ontology is a data model that formally represents a set of entities and their properties, relationships, constraints, and behaviours within a domain [Fox et al., 1997]. The objectives of this research are:

1. To capture and represent business semantics in the online retailing domain in a flexible way that can be easily extended.

2. To provide the means for exploring the terminology and generating further knowledge by assuming deductive capability as provided by an inference engine.

3. To address the issue of reasoning efficiently with the large amounts of data present in retail systems.

4. To provide a framework in which managers and knowledge workers have the ability to compose information requests without programmer assistance.

5. To provide a scientific foundation for using historical data to improve future decision making with regards to which potential customers to engage in relationships with, which customers to retain, etc.

Thus, from the standpoint of artificial intelligence this work is interesting as it explores a practical application of ontologies and introduces an approach for dealing with large datasets. Also, this work is useful to the business intelligence community as it attempts to formalize retail knowledge and provides the means for automated query answering.

1.2 The Need for a Retail Ontology

In the early 90's, business intelligence (BI) was born within the industrial world as a means to answer the managers' request to better understand the situation of their

business and to improve the decision process [Golfarelli et al., 2004]. The academic world became interested in BI in the mid-90's and research came up with ways for analyzing the enterprise data efficiently and effectively [Golfarelli et al., 2004], and by using the results obtained vendors were successful in creating useful software solutions. Today, the most important benefits considered in the development of BI are better information, better strategies, better tactics and decisions, and more efficient processes [Gibson et al., 2004].

Current BI solutions are mostly characterized as a kind of data-driven decision support systems [Power, 2007] which provide information to managers in the form of reports or at most dashboard-like monitoring of various business processes [Azvine et al., 2005]. Although they offer the means to transform data into information and derive some sort of knowledge through analytical tools [Sell et al., 2005], they still fall short of what is desired. Specialists are still required to run data mining or statistical analysis processes to set up reports which can later be used by managers [Azvine et al., 2005]. Of course, this goes against the desire of any manager or knowledge worker to be able to compose information requests without programmer assistance [Hill and Scott, 2004] and stops actions from being propagated back into business processes [Azvine et al., 2005]. Also, due to the fact that current analytical tools have no support for the definition of business logic, they lack the inference power needed to solve the requests of decision makers in a flexible way [Sell et al., 2005].

Ontologies, on the other hand, can be used to capture and represent business semantics, and by assuming deductive capability as provided by an inference engine, it is possible to explore the terminology and generate further knowledge. In this research an inference engine capable of handling first-order logic (FOL) is assumed. Another advantage of this approach is that anyone with a basic knowledge of first-order logic would be able to query the system. Even for query processing that is reducible to "looking up" an answer that is explicitly represented in a database, writing first-order sentences is a much easier task and more comprehendible than writing somewhat sophisticated SQL

statements. Thus, from the standpoint of managerial implementation, it is a significant benefit that managers can compose their own first-order sentences to query the system using the existing terminology; thereby avoiding the need to rely on others and so saving both time and money.

As marketing becomes more customer-centric, the accuracy of decisions with regards to which potential customers to engage in relationships with is becoming more important. Because in customer-based marketing companies have to invest in relationships, they need information on the potential value of a relationship. The ability to accurately predict the value of a company's relationships can have a large impact on the ability to intelligently influence both business process policies and IT related decisions pertaining to a company [Etzion et al., 2005]. By incorporating concepts such as customer lifetime value, customer profitability, expected transaction value, etc. the ontology provides a scientific foundation for using historical data to improve future decision making with regards to which potential customers to engage in relationships with, which customers to retain, whom to admit to a loyalty program, etc.

Therefore, in this regard this work differs from other attempts in its use of 1) first-order logic for the formalization of retail knowledge and business semantics which allows for automated query answering; and 2) customer-based marketing concepts which help improve future decision making.

1.3 Issues and Challenges

To be applicable in real world situations, the system should be able to deal efficiently with the huge amounts of data present in retail environments. In the context of the retail ontology, data become instances of concepts and roles in the ontology. State-of-the-art theorem provers and reasoners are still incapable of dealing efficiently with such large amounts of instances. On the other hand, relational databases have an established record

of storing and querying large amounts of data efficiently and are used by most companies on everyday basis. Thus, due to the size of the ontology, in order to be able to answer queries effectively it is essential to use database technology alongside the inference engine.

A closer look at the nature of instances in a domain like retailing reveals that it is not always necessary to reason over all existing instances of a particular concept since in most cases they are very similar to each other. Taking this fact into account and by using existing databases, this work introduces a technique for using both state-of-the-art theorem provers and/or reasoners and database technology for dealing efficiently with large amounts of instances that exist in these domains.

1.4 Evaluation and Design Criteria for the Ontology

Evaluation is an important step in the process of developing ontologies. The following criteria have been proposed as a basis for evaluating an ontology [Fox et al., 1997]:

- **Generality**: Is it possible to use the model in different applications?
- **Competence**: How well does the ontology represent the information necessary to support a task?
- **Perspicuity**: Is the model descriptive enough to be easily understood by the users?
- **Granularity**: Is it possible to use the representation to do reasoning at different levels of abstraction?
- **Efficiency**: How efficient is the reasoning process?
- **Scalability**: How well will the solution work when the size of the problem increases?
- **Extensibility**: How easy is it to extend the model?

In this research, the competency requirement has been chosen for evaluation. This is done by specifying a set of competency questions which the ontology should be able to represent and answer using a necessary and sufficient set of axioms, if it contains the relevant information. It is worth mentioning that the specification of the competency questions and the creation of the ontology is an iterative process in which the competency questions drive the development of the ontology which in turn results in the modification of the competency questions. The following steps are taken in the development of the ontology:

1. Define the span of the model by a set of competency questions.
2. Create an ontology for online retail.
3. Implement the definitions and constraints as axioms in a theorem prover or a reasoner.

Following are a subset of questions considered in the creation of the retail ontology.

- How much is the total revenue earned in a given period?
- Who are the customers?
- What is the purchase behaviour of a customer?
- How much is the acquisition profit (loss) in the acquisition year?
- How much is the total retention cost over a period of n years?
- What is the retention rate over a period specified by the starting time point s and ending time point t?
- How do buying trends compare across geographies?
- What is the average order value of each customer?

- What is the expected average transaction value associated with a customer?
- Which items have been purchased by customers who bought SKU x?
- Which customers add a cross-sell product to their order?
- How many visitors are converting into customers?
- Which products are making the most profit?
- Who are the most profitable customers?
- Which suppliers provide a product?
- Which suppliers are delivering the best value for a product?
- What is the minimum order size for a product?
- How much order lead-time is required to take shipment of a product?
- How effective was a promotion?
- Which orders have been canceled in a time period specified by the starting time point s and the ending time point e?
- Which payments have not been authorized by the assigned payment provider in a time period specified by the starting time point s and the ending time point e?
- What is the current physical count of an item in the inventory?
- Which items of stock on hand have reached a specified reorder point?
- Are there any product shortages?
- Which items of stock on hand have the highest margin?
- Which items of stock on hand have the lowest margin?

1.5 Overview

The document proceeds as follows: In Chapter 2 a review of related research is presented. The online retail ontology is discussed in detail in Chapters 3, 4, 5, and 6, which are respectively on product, purchasing, customer, and marketing. Chapter 7 is concerned with implementation issues and includes an approach for answering first-order expressive queries where the relations are populated by databases that potentially contain hundreds of millions of instances. Finally, Chapter 8 concludes the document with a discussion of the contributions and areas of future work.

Chapter 2

Related Research

2.1 Introduction

In today's competitive business environment, the ability to make effective decisions is of central importance to an organization's survival. Decision making is defined as the cognitive process of generating and evaluating alternatives and making choices among them [Druzdzel and Flynn, 2002], and is considered as the central aspect of most management and business activities including strategic planning, marketing management, and investment.

Making informed and effective decisions concerning complex systems, however, is often a difficult process. The difficulty is in part due to the complexity of decisions, the presence of uncertainty, and the existence of multiple and sometimes opposing objectives in these environments [Mezher et al., 1997]. Since the accuracy and quality of decisions is important in many situations, using information systems to aid the process of decision making and to improve the effectiveness of the decision maker has been a major focus of information systems research for many years.

Decision support systems (DSS), which are interactive, computer-based systems with a focus on supporting and improving managerial decision-making [Arnott and Pervan,

2005], were introduced in the mid-1960's [Power, 2007] as a solution for improving the decision process. There are a number of different approaches to implementing DSS, with each approach representing a different philosophy of support, system scale, level of investment, and potential organizational impact [Arnott and Pervan, 2005]. DSS are typically categorized into communications-driven, data-driven, document-driven, knowledge-driven and model-driven decision support systems [Power, 2007]. Arnott and Pervan [2005] group different decision support systems according to contemporary professional practice into personal decision support systems, group support systems, negotiation support systems, intelligent decision support systems, knowledge management-based systems, executive information systems, business intelligence systems, and data warehousing.

The focus of this research is on building a customer-centric business intelligence system applied to online retail in the form of a retail ontology that can automatically deduce answers to retail queries based upon the system's general knowledge of online retailing and actual data. The objectives are:

1. To capture and represent business semantics in the online retailing domain in a flexible way that can be easily extended.

2. To provide the means for exploring the terminology and generating further knowledge by assuming deductive capability as provided by an inference engine.

3. To address the issue of reasoning efficiently with the large amounts of data present in retail systems.

4. To provide a framework in which managers and knowledge workers have the ability to compose information requests without programmer assistance.

5. To provide a scientific foundation for using historical data to improve future decision making with regards to which potential customers to engage in relationships with, which customers to retain, etc.

The remainder of this chapter is organized as follows: in Section 2.2 the existing literature on business intelligence systems and their role in retail enterprises are reviewed. Section 2.3 presents existing retail data models and Section 2.4 is concerned with the definition and models of customer lifetime value, a useful measure of the value of a company's relationships. Section 2.5 briefly describes existing ontologies that are used in creating the retail ontology. Finally, the approach taken in this research is presented in Section 2.6.

2.2 Business Intelligence Systems

Perhaps the first time the term *Business Intelligence System* appeared in literature was in a paper by Luhn published in 1958. There, Luhn defines business intelligence (BI) as "the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal" in a specified business. He also describes a technique for selective dissemination of information and envisions a day when efficient data storage and retrieval will ultimately provide an effective answer to business intelligence problems.

Despite the fact that the first paper was published about fifty years ago, little academic research exists on BI [Negash and Gray, 2003][Gibson et al., 2004][Arnott and Pervan, 2005]. Nevertheless, business intelligence systems are widely used in industry and a large body of vendor and industry focused literature can be found [Gibson et al., 2004]. In this literature different software vendors and consulting organizations have defined BI differently to suit their products [Arnott and Pervan, 2005] and mostly tend to view BI as the query, reporting and analysis functions of decision support systems [Gibson et al., 2004].

Although it is widely acknowledged that business intelligence is not a well-defined concept [Azvine et al., 2005] and that it has been defined in literature from several perspectives, almost all the definitions share the same focus and they all agree in that 1)

the core of BI is information gathering, analysis and use [Lonnqvist and Pirttimaki, 2006] and 2) the goal is to support the decision making process [Marshall et al, 2004]. In this research, BI is considered as a broad umbrella concept which includes "the acquisition, interpretation, collation, assessment, and exploitation of business-related information" [Chung et al., 2003] with the aim of supporting decision making.

In this regard, two classes of BI tools can be defined. The first class consists of tools that are used to manipulate massive operational data and to extract essential business information from them [Chung et al., 2002]. Online analytical processing (OLAP) tools and data warehouses (DW) are examples of such tools. The second class, also known as competitive intelligence tools, consists of tools that are used to collect and analyze information from the competitive environment [Chung et al., 2002]. The goal of an organization in using these tools is to gain competitive advantage by utilizing information gathered from public sources such as the Web to assist organizational decision making [Soper, 2005]. This review focuses on the first class of tools.

The most important components of current BI systems are [Olszak and Ziemba, 2006]:

- Data warehouses (DW), used to store detailed summary data and metadata. The data is gathered from different sources through ETL tools.

- Extraction, transformation and loading tools (ETL). Extraction involves obtaining access to data originating from different sources and usually storing them in a relational database. Data transformation involves data unification, calculation of necessary aggregates, and identification of missing data or duplication of data and is usually performed by means of traditional programming languages, script languages or the SQL language. Data loading involves providing data warehouses with data that are aggregated and filtered.

- Analytical tools, used to analyze data and derive insights. Examples of such tools are online analytical processing (OLAP) tools which are mainly aimed at interac-

tive report generation according to users pre-defined criteria, and optimization of searching huge data files by means of automatic generation of SQL queries. Data mining techniques are also used to discover various patterns, generalizations, regularities and rules in data resources in order to either describe the current state of the business or predict the future.

A closer look at BI tools reveals the existence of some critical issues, making their use less effective in the process of decision making. In general, the existing analysis capabilities of many BI tools are still weak [Chung et al., 2002], and most come with a limited set of exploratory functionalities and do not provide scalable ways for extension of these functionalities [Sell et al., 2005]. Thus, thorough analysis is not provided by many BI tools and they only provide different views of the collected information [Chung et al., 2002]. In addition, specialists are still required to run data mining or statistical analysis processes to set up reports which can later be used by managers [Azvine et al., 2005]. Another shortcoming of current BI solutions is that since almost all of them depend on pre-built data warehouses, user selectable data sources and real-time data integration are rarely supported by these systems [Azvine et al., 2005]. Also, due to the fact that current analytical tools have no support for the definition of business logic, they lack the inference power needed to solve the requests of decision makers in a flexible way [Sell et al., 2005].

Some solutions have been proposed to address the deficiencies of BI tools. In [Sell et al., 2005], the authors argue that business semantics should be applied as the backbone for contextualization and integration of data and services in organizations and propose an architecture for business intelligence which uses semantic web technology based on *IRS-III* (Figure 2.1). In their approach, they process queries over the data sources by using a domain ontology (i.e. the formal specification of the terminology of the business domain that is being modeled by the system) to support the rewrite of conditions in order to broad the results of a query and to support inferences over the results of the

queries; and make the assumption that most of the data that could be useful to allow inferences and query rewrite are stored in the dimensions of a DW. One disadvantage of this model is that the specification of the domain ontology is left to the user, who needs to be familiar with the way data warehouses function and how analytical functionalities can be extended. Also, although they state that by defining symmetric and transitive relations between business concepts one can support and extend the exiting functionalities of analytical tools, such as *drill-down*, *drill-up*, and *drill-across*, it is not clear how this can be achieved in their system.

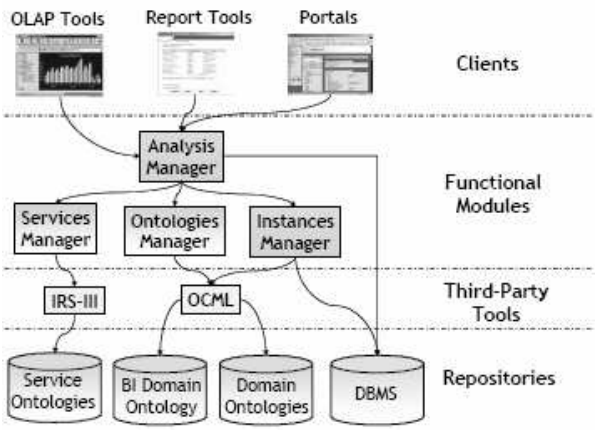

Figure 2.1: OntoDSS system architecture

Ou and Peng [2006] propose another architecture for BI by presenting a knowledge based business intelligence system which can be used for facilitating process model reuse. Their system is composed of five main components: an inference engine (composed of a case-based and a rule-based reasoning engine), a model base management system (composed of a process and a mathematical model-base), a data storage management system (composed of databases, data warehouses, and other data sources), a knowledge base management system (for storing business rules and related knowledge), and a workflow management system. Two types of tasks can be performed in this system: (1) the task can be a question query in which case domain or problem specific knowledge will be extracted from the knowledgebase to answer the question, and (2) the task can be the

process of finding solutions for a certain complex problem, in which case the most similar process will be retrieved form the process model base. The users can also make their own processes with the aid of the model base management system and the inference engine. One thing that can be problematic with this work is that the authors do not state how the issue of data integration is addressed, i.e., whether data is replicated as instances in the knowledgebase or customized engines are used. This is an important issue in domains where large amount of data is present. Also, one potential disadvantage of using this approach is that the knowledge discovery process described based on data mining techniques can take a long time, making the model inefficient in real world applications. In any case, since they do not provide any evaluation results, it is not clear how the system would perform in real world situations.

The literature review on BI also reveals that much benefit can be derived from using BI solutions [Lonnqvist and Pirttimaki, 2006], among which better information, better strategies, better tactics and decisions, and more efficient processes [Gibson et al., 2004] are considered to be the most important. However, few empirical studies can be found [Petrini and Pozzebon, 2003] [Gibson et al., 2004] to show that benefits are actually occurring in practice [Lonnqvist and Pirttimaki, 2006]. This is in part due to the fact that most benefits in BI are intangible [Gibson et al., 2004] which make BI measurement a difficult task to carry out.

2.2.1 Business Intelligence in Retail Enterprises

The retail environment is highly complex and competitive. Retailers need to continually assess and redirect their actions in order to stay on top of the markets they choose to serve. As a result of technological developments in data collection and data processing, retailers have vast amounts of data at their disposal - everything from supply chains to sales information to customer transaction data. But with the constantly increasing volume of data, it is hard to keep track of important information and even to know which

information is valuable.

BI solutions are aimed at giving retailers the capability to analyze the vast amounts of information they already have to identify sales and profit growth opportunities, to help in understanding customers' buying patterns, and to improve overall decision making [Hill and Scott, 2004].

Olszak and Ziemba [2006] outline the following objectives for which BI systems can be used in the retail industry:

- **Forecasting**: To forecast demand in order to better define inventory requirements.
- **Ordering and replenishment**: To make faster decisions about items to order and to determine optimum quantities.
- **Marketing**: To provide analyses of customer transactions.
- **Merchandising**: To define the right merchandise for the market at any point in time which helps in planning store level and refining inventory.
- **Inventory planning**: To help identify the inventory needed level, and ensure a given grade of service.

2.3 Existing Retail Ontologies

An ontology is a formal description of a set of objects, concepts, and other entities that are assumed to exist in a domain of interest along with their properties and the relationships that hold among them [Gruber, 1993]; it forms a shared terminology for the objects in that domain, along with definitions for the meaning of each of the terms [Fadel, 1994]. Different approaches to ontologies exist depending on the manner of specifying the meaning of terms, resulting in a continuum of kinds of ontologies [Uschold and Gruninger, 2004]. Moving from left to right, the amount of meaning specified, the degree of formality,

and the support for automated reasoning increase [Uschold and Gruninger, 2004]. In this research, the term 'ontology' is used in the narrow sense restricted to formal logic-based models.

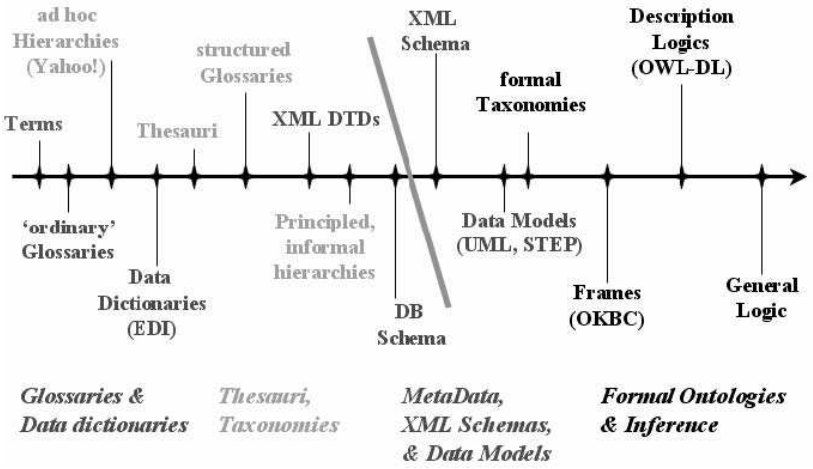

Figure 2.2: Kinds of ontologies [Uschold and Gruninger, 2004]

To our knowledge, no ontology (in the narrow sense) has been developed for (online) retailing. However, many retail data models have been introduced. A data model is an abstract model that describes how data is represented and used. The function of a data model is to clearly convey data, data relationships, data attributes, data definitions and the business rules that govern data. The relationships between ontologies and data models have been addressed in [Uschold and Gruninger, 2004] and [El-Ghalayini et al., 2006]. In what follows, an overview of some existing retail data models (ARTS, Teradata, Claraview, and ADRM) is presented.

2.3.1 ARTS Retail Data Model

The Association for Retail Technology Standards[1] (ARTS) is an international membership organization dedicated to reducing the costs of technology through standards. To achieve the objective of fully integrating the retail applications of both the retailers and

[1] http://www.nrf-arts.org

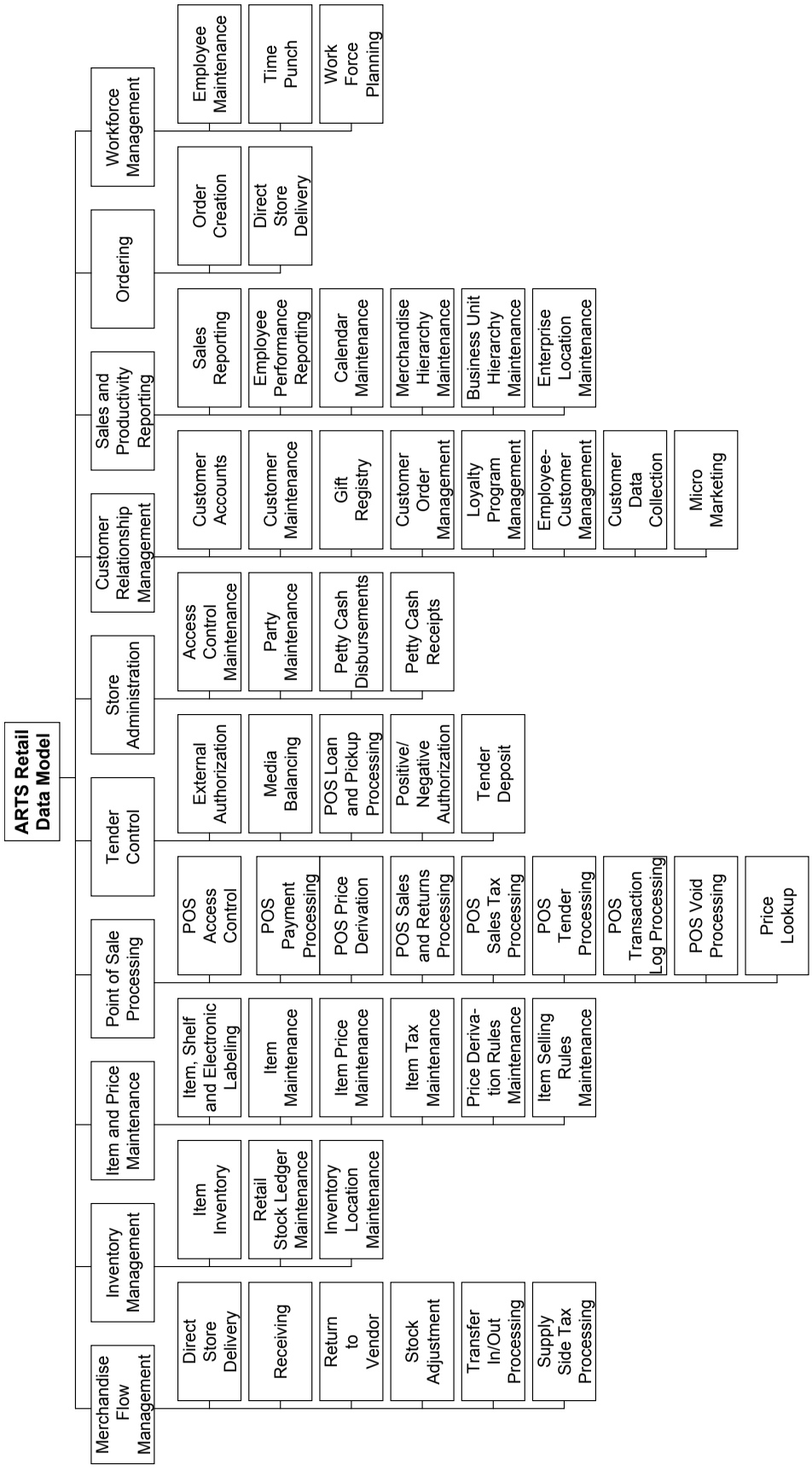

Figure 2.3: ARTS Retail Data Model - Business Functions

their partners, ARTS has introduced a retail data model in XML which identifies the entities in the retail enterprise and specifies data associated with each. The ARTS Retail Data Model provides two distinct perspectives across the retail business, namely:

- Enterprise context which provides an insight into the retail enterprise via three levels within the retail operation. (Home Office Level, Distribution Centre (Warehouse) Business Level and the Retail Store Business Level)

- Subject area composition which provides an insight into the retail enterprise via the subject areas, which cut across all three levels of retail operation.

The data model is organized into ten business functions, with each business function being broken down into a further set of subject areas. The current business functions and their subject areas are illustrated in Fig. 2.3.

2.3.2 Teradata Retail Logical Data Model

The Teradata[2] Retail Logical Data Model (RLDM) is a logical data model in Third Normal Form (3NF). A logical data model (LDM) is an abstract construct that is physically realized in the database or data warehouse. The logical data model provides an architecture for the information that will be included in the warehouse. The database provides the physical realization of that architecture in a form that can be efficiently maintained and used.

The Teradata Retail LDM reflects the operating principles and policies of the retail industry and provides the underlying structure for the data imported into the data warehouse, in the following ways:

- It serves as a road map for achieving cross-functional data integration in an organization.

[2]http://www.teradata.com/t/

Table 2.1: Teradata Retail Logical Data Model - Subject Areas

Address/Geography	Location	Shipping Order
Agreement	Model Score and Forecast	Shipping Transport
Associate Labor	Multimedia Component	Shipping Plan
Catalog	Payment Account	Shipping Regulation
Customer/Party	Retail Pharmacy	Quality Feedback
Demographics	Plan-o-gram	Time Period
Financial Management	Point-of-Sale Register	Vendor Purchase Order
Item Definition	Privacy	Web Operations/Server
Item Pricing	Promotion	Web Site
Inventory (External)	RFID/Serialized Item Tracking	Web Visit/Browsing
Inventory (Internal)	Sales (External)	Vendor Purchase Receipts
Invoice	Sales (Internal)/Fulfillment	

- It shows interlocking and interdependent data relationships that allow for extension and expansion for future Teradata Retail Solutions.
- It assists in application development through common definitions and standards.
- It aids the communication between technical and business constituencies.

Subject areas that the Teradata Retail LDM help support are given in Table 2.1.

2.3.3 Claraview Retail Data Model

The Claraview[3] Retail Data Model is an enterprise data model that encompasses transaction-level operational and planning data and integrates with the ARTS operational data

[3]http://www.claraview.com/

model. The major features of Claraview RDM include:

- It is expandable to meet individual retailer's requirements.
- It is designed for implementation on any of the leading data warehouse platforms, including IBM DB2, Microsoft SQL Server, NCR Teradata, Netezza, and Oracle, and using any of the leading data integration platforms, including Informatica and IBM WebSphere DataStage.

2.3.4 ADRM Retail Data Environment

Applied Data Resource Management[4] (ADRM) develops and markets industry-specific data models for enterprise planning, data warehouse, data mart and applications development.

The ADRM Retail Data Environment consists of a set of integrated data models developed for companies developing, marketing and supplying consumer retail products and services. It provides a comprehensive data architecture that can be immediately applied to business data requirements across a variety of industry segments, such as clothing and apparel, home furnishing, consumer electronics, electrical equipment, furniture, personal products, leisure and recreational products, office equipment, and appliances. The Retail data models address business-wide data and reporting issues. The ADRM Architecture consists of:

- The *Enterprise Model* incorporates and depicts the integrated data requirements of the organization in a single logical data model. It is the primary data model and foundation data model for strategic planning, communicating information requirements throughout the organization, implementing integrated systems and organizing data in the lower-level Business Area, Data Warehouse and Data Mart models. Each major 'subject' in the Enterprise Data Model has additional information and

[4]http://www.adrm.com/7_adrm.htm

detail that cannot be addressed completely in the Enterprise Data Model due to lack of space and an inappropriate level of detail for a general audience. That additional information is contained in one or more related Business Area Models.

- *Business Area Models* describe functional business or subject areas found in many industries or developed for a specific industry. Each Business Area Model is constructed from a common set of entities from the corresponding industry Enterprise Model, insuring that they will have common keys, attributes and definitions throughout the data architecture. Business Area Models contain the greatest level of detail and provide the lowest level of data granularity in the model hierarchy. Individual models can easily be combined or integrated to create other models.

- *Data Warehouse and Data Mart Models* are derived from the Enterprise and Business Area Models in either 3NF or star schema format.

2.4 Customer Lifetime Value

As marketing becomes more customer-centric, the accuracy of decisions with regards to which potential customers to engage in relationships with is becoming more important. Because in customer-based marketing companies have to invest in relationships, they need information on the potential value of a relationship. The ability to accurately predict the value of a company's relationships can have a large impact on the ability to intelligently influence both business process policies and IT related decisions pertaining to a company [Etzion et al., 2005]. A useful measure for this is the concept of customer lifetime value (CLV). CLV can be used for estimating the strength of a customer relationship, and for making decisions such as how much to invest in which (potential) customers, whom to admit to a loyalty program, etc.

2.4.1 Definition and Models

There are several definitions of CLV present in the direct marketing literature. The definition we assume is the one given in [Pfeifer et al., 2005]: CLV is the present value of the future cash flows associated with a customer. Closely linked to the definition of CLV is its formalization in terms of a procedure or a formula. Many different approaches exist for calculating CLV. The differences in the calculations range from the components that constitute "cash flow" mentioned in the definition of CLV to the procedures that are proposed to arrive at outcomes [Hoekstra and Huizingh, 1999].

The difficulty in calculating CLV arises from the fact that it is very challenging to model future cash flows appropriately, specially when the time at which a customer becomes inactive is unobserved [Fader et al., 2005]. Therefore, in order for a CLV model to be complete, it must identify the set of attributes of customers future behavior, and specify a method for predicting the customer value based on these attributes [Etzion et al., 2005].

In [Berger and Nasr, 1998], the authors present a series of mathematical models for calculating CLV based on different underlying assumptions. Their first two models are based on the restrictive assumption that revenues per customer are constant. For direct retailing applications, this seems like an over simplification. The other models presented in their paper relax this assumption, thereby addressing situations in which gross contribution and promotional cost are non-constant over time (by estimating the customers profit function), and allow continuous cash flow rather than discrete cash flow. Although they give an example of a function that can be used as an approximation to the profit curve, they do not comment on which attributes of customer behavior should be used to estimate the values of parameters needed for the calculation of the profit function, and only suggest that historical data can be used to infer those values.

The CLV model in [Hoekstra and Huizingh, 1999] considers both past and future contribution of customers in terms of four perspectives of lifetime value, namely, customer

quality and potential, and supplier quality and potential. Quality refers to the value of a customer/supplier as manifested by their behavior in the past, whereas potential reflects the future value of a customer/supplier. In this work, the customer share, which presents the amount spent by the customer with regards to his/her total spending, is also taken into account. The main drawback of this model is that it lacks a mathematical formalization of quality and potential, and therefore, does not provide a method for deriving the models parameters.

There are also CLV models in which it is assumed that the customers past behavior is a good indicator of their future behavior. These models summarize customers prior behavior in terms of their recency (time of most recent purchase), frequency (number of prior purchases), and monetary value (average purchase amount per transaction), three attributes which are known as RFM characteristics in the direct marketing literature. The CLV model described in [Fader et al., 2005] is one such model. It uses a forward-looking procedure that features behavioral assumptions rather than relying on only prior purchase data as a mirror for the future. The behavioral assumptions are formalized mathematically using the Pareto/NBD model. Examples of these general assumptions include the fact that a customer is active for some period of time and then becomes permanently inactive, and that purchase rates and drop out rates vary independently across customers. The most significant benefit of this model is that its inputs are nothing more than each customers RFM characteristics.

Another RFM-based model is presented in [Etzion et al., 2005], in which customer relations are represented using Markov Chain Models (MCM). In this work, RFM variables along with other domain specific attributes are used for the definition of MC states, where a state is defined for each distinct set of possible values for these variables. A learning algorithm is also given which derives the transition probabilities between states. One disadvantage of this model is that because MCM is a finite state machine, only a finite number of values are allowed for each variable, hence, all continuous variables must be

discretized which is really a problem. Another potential disadvantage is that depending on the variables chosen for the definition of MCMs states and the possible range of values they can take, the number of states present can be very large, thus making the model inefficient in some applications and domains.

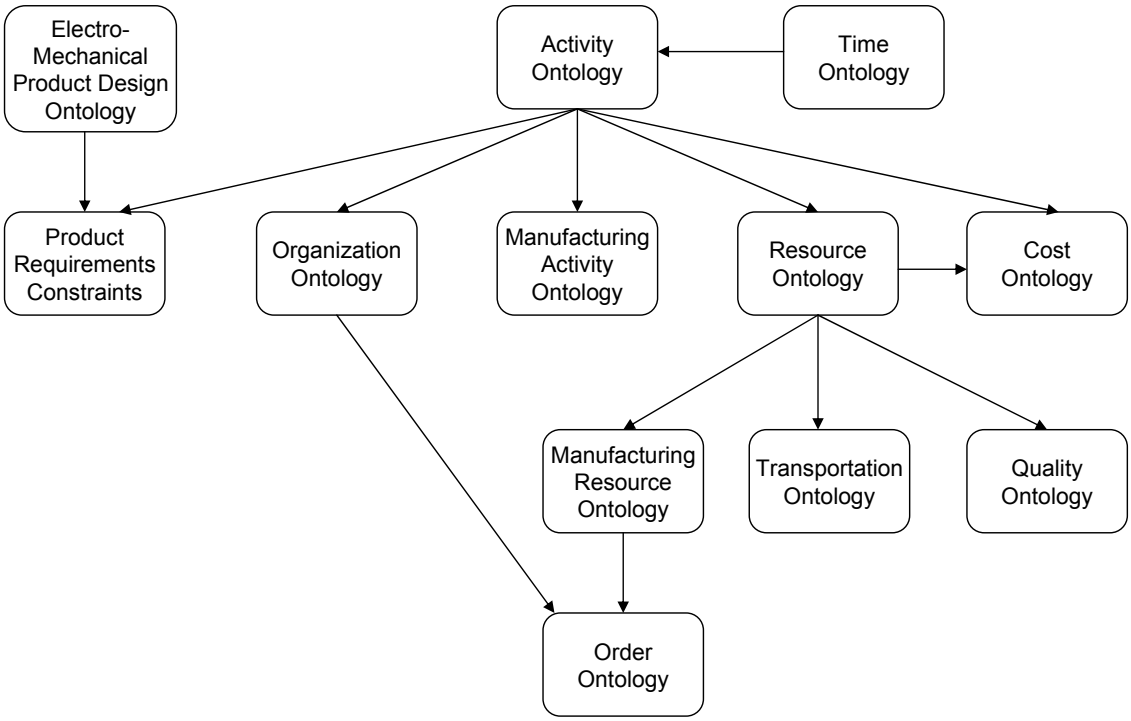

Figure 2.4: Toronto Virtual Enterprise Ontologies

2.5 Related Ontologies

The ontologies developed as part of the Toronto Virtual Enterprise (TOVE) project [Fox and Gruninger, 1998] are used in the development of the retail ontology. TOVE ontologies were developed in cooperation with several companies with the goal of providing a basis for enterprise modeling. The project currently spans knowledge of activity, time, and causality [Fox and Gruninger, 1994], resources [Fadel, 1994] [Fadel et al., 1994], cost

[Tham et al., 1994], quality [Kim et al., 1999], organization structure [Fox et al., 1997], product [Lin et al., 1996] and agility [Gruninger et al., 2000]. This section briefly describes the activity, time, resource, cost, and organization structure ontologies. The complete set of axioms for all the ontologies discussed in this chapter can be found in Appendix B.

2.5.1 The Activity-State Ontology

In TOVE, an activity is the basic transformational action primitive with which processes and operations can be represented. An enabling state defines what has to be true of the world in order for the activity to be performed. A caused state defines what will be true of the world once the activity has been completed. An activity along with its enabling and caused states is called an *activity cluster* (Figure 3.2) and is used to represent an action.

Figure 2.5: Activity-State model

2.5.1.1 States

As already stated, states in TOVE define what holds to be true before and after an activity is performed. There are two types of states: *terminal* and *non-terminal*. Terminal states associate resources with activities through the four types of states which reflect the four ways in which a resource is related to an activity: use, consume, release, and produce. Non-terminal states enable the boolean combination of states.

The predicates $uses(s,\ r), consumes(s,\ r), produces(s,\ r), releases(s,\ r)$ express the relation between a terminal state s and a resource r (defined later). The *quanitiy_needed* relation refers to the quantity of objects needed in the particular resource class.

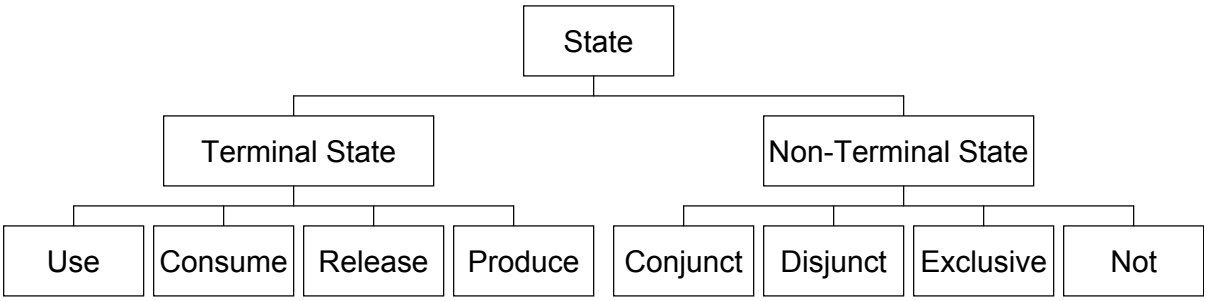

Figure 2.6: State taxonomy

The status of a state, and any activity, is dependent on the status of the resources that the activity uses or consumes. All states are assigned a status with respect to a point in time. Status can have one of the following values:

- **committed**: a unit of the resource that the state consumes or uses has been reserved for consumption.
- **enabled**: a unit of the resource that the state consumes or uses is being consumed.
- **disenabled**: a unit of the resource that the state consumes or uses has become unavailable.
- **reenabled**: a unit of the resource that the state consumes or uses is re-available.
- **completed**: a unit of the resource that the state consumes or uses has been consumed or used and is no longer needed.

2.5.1.2 Activities

An activity specifies a transformation on the world. Its status is reflected in an attribute called status. The domain of an activitys status is a set of linguistic constants:

- **dormant**: the activity is idle and has never been executing before.
- **executing**: the activity is executing.

- **suspended**: the activity was executing and has been forced to an idle state.

- **reExecuting**: the activity is executing again.

- **completed**: the activity has finished.

The status of an activity is defined by the status of its enabling and caused states. The complete logical definitions of the status of activities and states can be found in the TOVE manual [Fox et al., 1994].

In TOVE, activity clusters may be also aggregated to form multiple levels of abstraction in order to define new activities. The predicate *has_sub_activity* is used to denote that an activity is a subactivity of an aggregate activity.

2.5.2 Time

In TOVE, time is represented by points and periods on a continuous time line. A time-point represents an instance and lies within an interval. A time-period is bounded by a start and end time-point and has a duration. Allen's temporal relations are used to describe the relationships between time-points and/or time-periods.

By combining the ontology of time with the activity-state ontology, the notion of duration can be defined, which is essential for scheduling and the analysis of activities in time-based competition. The duration of a state is defined as the time period beginning at the time that the state is enabled and ending at the time that the state is completed. Similarly, the duration of an activity is defined as the time period beginning at the time that activity begins the status of executing and ending at the time that the activity begins the status of terminated. The duration of a state is represented by the predicate *state_duration*, while the duration of an activity is represented by the predicate *activity_duration*.

2.5.3 Resource

"Being a resource", as viewed in TOVE, is a property that is derived from the role an object plays with respect to an activity. Examples of resources include machines, raw materials, human skills, and information. Following are a subset of questions the resource ontology is capable of answering.

- Divisibility: Can the resource be divided and still be usable?
- Quantity: What is the stock level at any time point?
- Location: Where is resource R?
- Consumption: Is the resource consumed by the activity? If so, how much?
- Commitment: Which activities is the resource committed to at any time?
- Structure: What are the subparts of resource R?
- Capacity: Can the resource be shared with other activities?

In order to answer these questions, the resource ontology includes the concepts of a resource being divisible, quantifiable, consumable, reusable, a component of, and committed to an activity. Also, as already stated, states associate resources with activities through *use*, *consume*, *release*, and *produce* terminal states.

TOVE's resource ontology is used for describing retail resources including financial, physical, human, virtual, legal, organizational, and informational resources. Figure 2.7 shows the taxonomy for retail resources.

2.5.4 Activity Cost

Activity based costing (ABC) is a methodology that assigns costs to activities rather than products or services. This enables resource and overhead costs to be more accurately

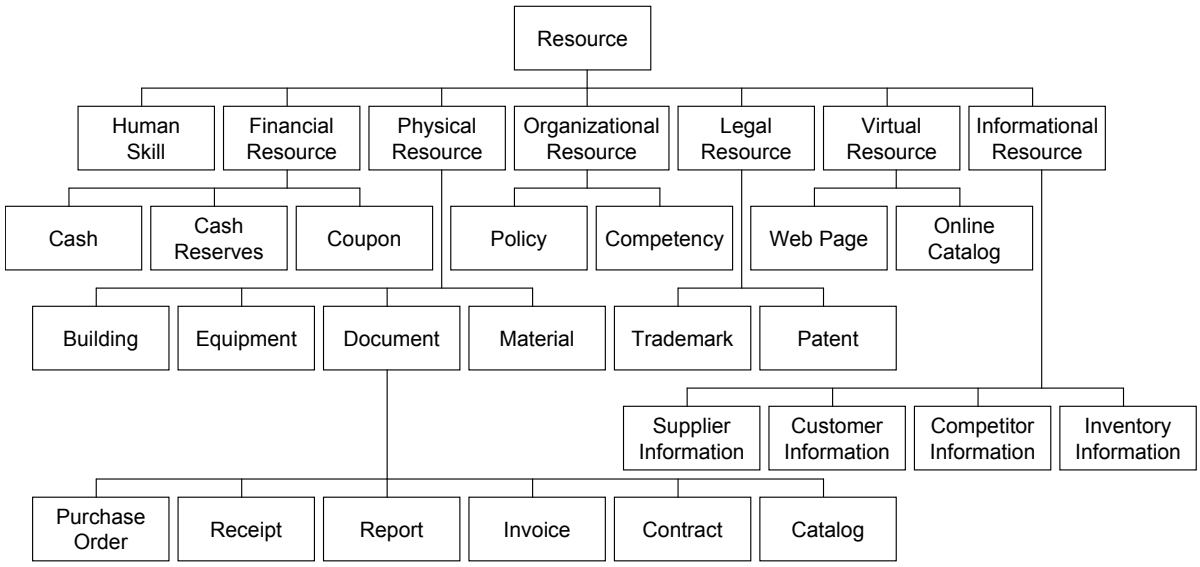

Figure 2.7: Retail resource taxonomy (sub-class relationships)

assigned to the products and the services that consume them. In retailing, some costs are further described based on their timing or their applicability. For example, the cost associated with acquiring a new customer is referred to as the customer acquisition cost.

In TOVE, an activity cost is defined as the entity which represents the temporal fiscal or monetary dimension, attribute, or characteristic of an activity. Hence, the activity cost ontology includes concepts of cost value of an activity associated with a required resource, the aggregate cost value of an activity, resource cost units, etc., with the objective of answering questions such as:

- What is the instantaneous and cumulative cost of a resource used in an activity at a given time point?
- What is the instantaneous and cumulative cost of an activity at a given time point?
- What is the cumulative cost of the class of activities a?

It is worth mentioning that the quantification of cost is treated constant with time until a status change occurs, i.e., costs will only change whenever the status of states and

activities change. Also, it is assumed that resource cost units are known or given for the enterprise modeled.

2.5.5 Organization

An organization, as viewed in TOVE, is a set of constraints on the activities performed by agents. An Organization consists of a set of divisions and subdivisions, a set of agents, a set of roles that the members play in the organization, and a set of goals that the members try to achieve.

2.6 The Approach in this Research

The focus of this research is on building a customer-centric business intelligence system applied to online retail in the form of a retail ontology that can automatically deduce answers to retail queries based upon the system's general knowledge of online retailing and actual data. An ontology is a formal description of a set of objects, concepts, and other entities that are assumed to exist in a domain of interest along with their properties and the relationships that hold among them [Gruber, 1993]; it forms a shared terminology for the objects in that domain, along with definitions for the meaning of each of the terms [Fadel, 1994] which are normally organized in a taxonomy [Fridman and Hafner, 1997]. The system is designed with the following objectives:

1. To capture and represent business semantics in the online retailing domain in a flexible way that can be easily extended;

2. To provide a scientific foundation for using historical data to improve future decision making with regards to which potential customers to engage in relationships, which customers to retain, etc.;

3. To provide the means for exploring the terminology and generating further knowledge by assuming deductive capability as provided by an inference engine;

4. To provide a framework in which managers and knowledge workers have the ability to compose information requests without programmer assistance.

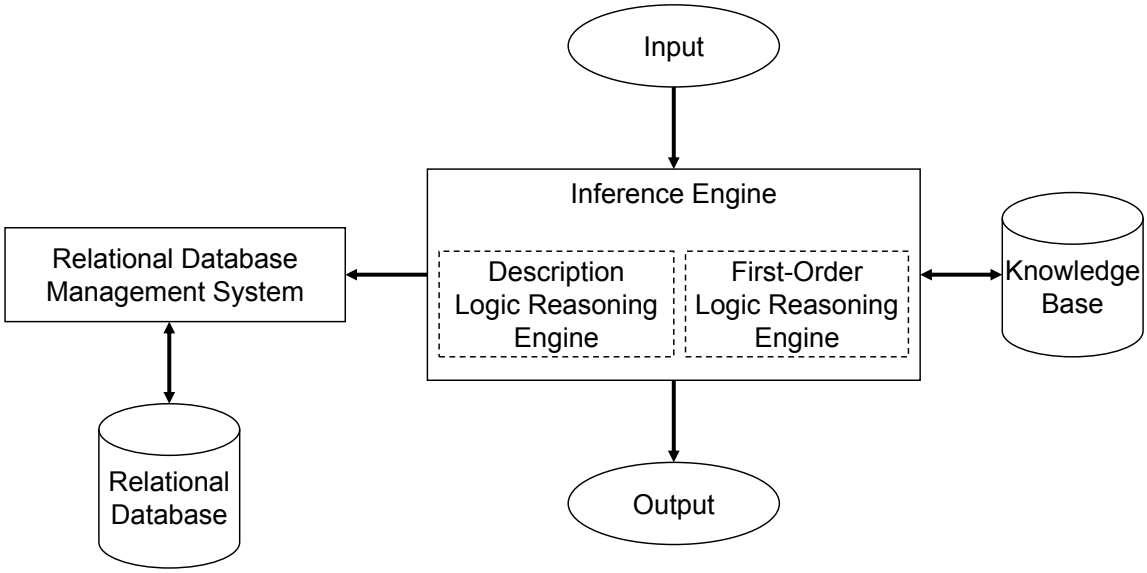

Figure 2.8: Retail business intelligence system architecture

The ontology is defined in the following way. First, the objects - the basic entities in the ontology - in the domain of discourse are defined. There are two types of objects: concept and instance. Concepts are used for specifying generalized types or categories of objects and are represented by unary predicates, whereas instances are used for denoting specific members of a class and are represented by constants or variables. Second, the properties of the specified concepts and the relations that exist over them are defined and represented by predicates, hereafter referred to as roles. Next, a set of axioms in first-order logic are defined to represent the constraints over the concepts and roles in the ontology. This set of axioms provides a declarative specification for the various definitions in the terminology. Finally, in order to provide a declarative semantics for the system, a necessary and sufficient set of axioms are defined. These axioms are needed

to represent and solve a set of problems, called competency questions, that are used in order to prove the competency of the ontology. In addition to defining the types of tasks that the representation can be used in, competency questions drive the development of the ontology. They do not only represent simple retrievals from the knowledgebase, but also entail deduction [Fadel, 1994]. The retail ontology modules and their relationships with respect to each other are illustrated in Figure 3.1.

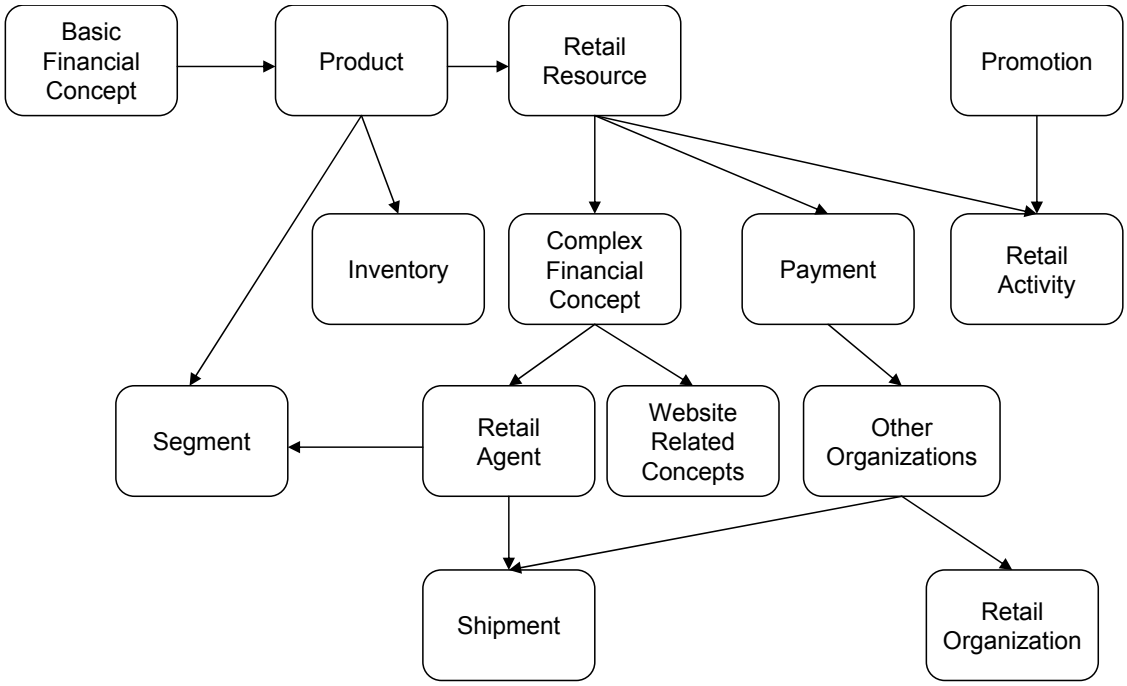

Figure 2.9: Retail Ontology Modules

We chose the CLV model given in [Fader et al., 2005] for calculating customer lifetime value in the e-retailing domain for the following reasons:

1. The context of non-contractual purchasing (i.e., a setting where the time at which customers become inactive is unobserved) assumed in the model is right for this domain;

2. The behavioral assumptions made are very general and beautifully formalized using the Pareto/NBD model; and

3. The model is only based on customers RFM characteristics which can be derived from the available customer transaction data.

In their model, Fader, Hardie, and Lee assume that monetary value is independent of the underlying transaction process and show in their experiments that although a slight positive correlation between average transaction value and the number of transactions exist, the value is such that it can be ignored. Taking this assumption into account, the model for CLV can be divided into two submodels: a submodel for the number of discounted expected transactions (DET), and a separate submodel for expected revenue per transaction.

$$\text{CLV} = \text{margin} \times \text{revenue/transaction} \times \text{DET}. \tag{2.1}$$

The only customer-level information that the DET submodel requires is recency and frequency which is represented using the notation (X = x, t_x, T), where x is the number of transactions observed in the time interval (0, T] and t_x ($0 < t_x \leq T$) is the time of the last transaction. DET is then computed over an infinite time horizon, i.e. standing at time T, the present value of the expected future transaction stream for a customer with purchase history (X = x, t_x, T) with continuous compounding at rate of interest δ is computed. Thus, the DET expression for the Pareto/NBD model is given by:

$$\text{DET}(\delta \mid r, \alpha, s, \beta, X = x, t_x, T) = \frac{\alpha^r \beta^s \delta^{s-1} \Gamma(r + x + 1) \Psi[s, s; \delta(\beta + T)]}{\Gamma(r)(\alpha + T)^{r+x+1} L(r, \alpha, s, \beta \mid X = x, t_x, T)}, \tag{2.2}$$

where r, α, s, and β are Pareto/NBD parameters; $\Psi(.)$ is the confluent hypergeometric function of the second kind; and L(.) is the Pareto/NBD likelihood function:

$$\text{L}(r, \alpha, s, \beta \mid X = x, t_x, T) = \frac{\Gamma(r + x)\alpha^r \beta^s}{\Gamma(r)} \times \{\frac{1}{(\alpha + T)^{r+x}(\beta + T)^s} + (\frac{s}{r + s + x})A_0\}, \tag{2.3}$$

where, for $\alpha \geq \beta$,

$$\mathrm{A}_0 = \frac{1}{(\alpha + t_x)^{r+s+x}}\ {}_2F_1(r+s+x, s+1; r+s+x+1; \frac{\alpha-\beta}{\alpha+t_x}) - \frac{1}{(\alpha+T)^{r+s+x}}\ {}_2F_1(r+s+x, s+1; r+s+x+1; \frac{\alpha-\beta}{\alpha+T}),$$

and for $\alpha \leq \beta$,

$$\mathrm{A}_0 = \frac{1}{(\beta + t_x)^{r+s+x}}\ {}_2F_1(r+s+x, r+x; r+s+x+1; \frac{\beta-\alpha}{\beta+t_x}) - \frac{1}{(\beta+T)^{r+s+x}}\ {}_2F_1(r+s+x, r+x; r+s+x+1; \frac{\beta-\alpha}{\beta+T}).$$

In the above equations, ${}_2F_1$(a, b; c; z) is the Gauss hypergeometric function that can be computed using numerical integration. The four Pareto/NBD model parameters can be estimated using the method of maximum likelihood.

For their submodel of monetary value, they assume that the underlying average transaction values follow a gamma distribution across the population. The goal of this submodel is to make inferences about the true underlying average transaction value E(M) given the customers average observed transaction value m_x. For a given customer with x transactions, if z_1, z_2, ..., z_x represent the dollar value of each transaction, then $\mathrm{m}_x = \sum_{i=1}^{x} z_i/x$. Assuming that the Z_i are i.i.d. gamma variables with shape parameter p and scale parameter ν, and that values of ν are distributed across the population according to a gamma distribution with shape parameter q and scale parameter γ, it can be shown that the expected average transaction value for a customer with an average spend of m_x across x transactions is

$$\mathrm{E}(\mathrm{M} \mid p, q, \gamma, m_x, x) = (\frac{q-1}{px+q-1})\frac{\gamma p}{q-1} + (\frac{px}{px+q-1})m_x. \tag{2.4}$$

Having submodels for DET and monetary value, by substituting Equation 2.2 and Equation 2.4 into Equation 2.1 along with a value for the gross contribution margin, the expected CLV for customers can be calculated.

Chapter 3

Product

3.1 Introduction

A key challenge for any retailer is inventory management since too much inventory means unnecessary cost and too little increases the risk of losing sales. Effective inventory management enables an organization to meet or exceed customers' expectations of product availability while maximizing net profits or minimizing costs. Thus, the primary objective of inventory management is determining which items to order, and in what quantity. Other objectives include increasing inventory turnover without sacrificing the service level, maintaining a wide assortment of stock, and keeping stock low without sacrificing performance.

This chapter presents the entities, attributes and relations needed for inventory management. Relevant competency questions are presented in Section 4.2, followed by the basic terminology of the *Product*, *Inventory*, and *Basic Financial Concept* modules of the Retail Ontology in Sections 4.3, 4.4, and 4.5. Section 4.6 is concerned with defining *Catalog* and *Online Catalog* which are subclasses of physical and virtual resources described in the previous chapter. Section 4.7 provides answers for the competency questions, and Section 4.8 concludes this chapter with a summary.

3.2 Competency Questions

Assume a scenario which deals with the issue of determining which products to order. The scenario is initiated when the purchasing manager (PM) analyses the inventory and identifies the items that may be reordered. The analysis is done through asking the following questions.

- What is the current physical count of an item in the inventory?
- Are there any product shortages?
- Which items of stock on hand have reached their specified reorder point?
- Which items of stock on hand have the highest margin?
- Which items of stock on hand have the lowest margin?
- What are the most profitable products in a specified time period? (This question will be discussed later in Chapter 6)

Once the items are identified, the PM sends a message to the product managers (TM) informing them of the need to reorder the items specified. The product managers then conduct a cost/benefit analysis and assign priorities to items and quantities in which they should be reordered. The resulting list is sent to the resource manager (RM) who checks to see if resources are available for ordering the products identified (using the resource ontology). Finally, the RM informs the purchase manager of the decisions made with a message to buy the items chosen.

The rest of this chapter presents the entities, attributes and relations needed for answering the above competency questions.

3.3 Product

A product refers to an item for sale and is represented by a stock keeping unit (SKU) code for inventory purposes. Product is a concept in the ontology. p is a product if the following is asserted in the knowledgebase:

- $Product(p)$.

The SKU is used to identify an individual product, and therefore each product must have a SKU, and each SKU must be unique.

$$(\forall\, p)\ Product(p) \supset$$
$$((\exists\, x)\ SKU(x) \wedge hasSKU(p, x) \wedge (\neg\exists\, y)\ hasSKU(p, y) \wedge y \neq x) \quad \text{(FOL a-1)}$$

Categories are groups of products and are used as organizing tools. Products are organized into categories and categories are organized relative to each other. Examples of categories include kids shoes and Halloween costumes. Category is also a concept in the ontology.

$$(\forall\, cat,\ p)\ ProductCategory(cat) \wedge Product(p) \supset$$
$$(hasProduct(cat,\ p) \equiv productOf(p,\ cat)). \quad \text{(FOL a-2)}$$

$$(\forall\, cat,\ par\_cat)\ ProductCategory(cat) \wedge ProductCategory(par\_cat) \supset$$
$$(hasChildCaterogry(par\_cat,\ cat) \equiv childCategoryOf(cat,\ par\_cat)). \quad \text{(FOL a-3)}$$

In addition, a product also has the following attributes which will be discussed later:

- $hasCatalog(p,\ cg)$, where cg is a Catalog ID.
- $hasCurrentPrice(p,\ pr)$, where pr is a Price ID.
- $hasCostOfGoodsSold(p,\ c)$, where c is a decimal value.

Different products can be related to each other in different ways. For example, a product can be an add-on or supplement to another. To capture this information, product-product maps are defined and can be used to recommend products to customers. A *ProductProductMap* is defined between two products, *hasProduct* and *hasParentProduct*, and has a mapping type (more on product-product maps is presented later in Chapter 6). The parent product refers to the main product being mapped.

$$(\forall\, ppm)\; ProductProductMap(ppm) \supset (\exists\, par\_p,\; p,\; m\_type)$$
$$Product(par\_p) \wedge hasParentProduct(ppm,\; par\_p) \wedge$$
$$Product(p) \wedge hasProduct(ppm,\; p) \wedge hasMappingType(ppm,\; m\_type), \tag{FOL a-4}$$

where $m\_type \in$ {add-on, cross-sell, up-sell, ...}.

Products can be returned or exchanged after they have been purchased by customers if they are returnable:

- $isReturnable(p,\; x)$, where x is either true or false.

3.4 Inventory

A retailer's inventory of goods for sale consists of all the products held available in stock that it has purchased from suppliers. In this ontology, inventory is considered to be a set of inventory items, each holding the availability information of a product. The reason for having the concept of inventory and not just associating the availability information with the product is that a retailer may be stocking the same product in different warehouses to be sold at different stores.

- $Inventory(inv)$,

- $InventoryItem(inv\_item)$,

- $hasItem(inv,\ inv\_item)$,
- $hasProduct(inv\_item,\ p)$, where p is a Product ID.

In addition, each $InventoryItem$ also has the following attributes:

- $hasQuantity$: total available amount of the product.
- $hasQuantitySold$: amount of the product sold (payment has been received).
- $hasQuantityDeferred$: amount of the product in deferred order. Deferred orders refer to items which have been purchased, but the customer is allowed to pay for them in a later date, with no interest charge.
- $hasQuantityBuffer$: amount of the product that should be kept, in case there are damages or exchanges.
- $hasQuantitySoldTotal$: total sold amount of the product.

$$(\forall\ inv\_item,\ qs,\ qd,\ \exists\ tot)\ InventoryItem(inv\_item) \wedge$$
$$hasQuantitySoldTotal(inv\_item,\ tot) \equiv hasQuantitySold(inv\_item,\ qs) \wedge$$
$$hasQuantityDeferred(inv\_item,\ qd) \wedge tot = qs + qd. \qquad \text{(FOL a-5)}$$

- $hasReorderLevel$: threshold for reordering.
- $hasReorderDatetime$: time when the stock was reordered.

When the values of quantity and reorder level are equal for an inventory item, the item has reached its reorder point. In this case management has to decide on whether the item should be reordered or not.

$$(\forall\ inv\_item,\ q,\ rl)\ InventoryItem(inv\_item) \wedge hasQuantity(inv\_item,\ q) \wedge$$
$$hasReorderLevel(inv\_item,\ rl) \wedge q = rl \supset$$
$$reachedReorderLevel(inv\_item, true). \qquad \text{(FOL a-6)}$$

3.5 Basic Financial Concepts

3.5.1 Price

Price is the sum or amount of money for which a product is bought, sold, or offered for sale. Price is a concept in the ontology with the following two attributes:

- $hasCurrency(pr,\ cur)$, where cur is a Currency ID.
- $hasValue(pr,\ val)$, where val is a decimal value.

Every product has a price assigned to it:

$$(\forall\, p)\ Product(p) \supset (\exists\, pr)\ Price(pr) \wedge hasCurrentPrice(p,\ pr). \qquad \text{(FOL a-7)}$$

A product may have a discounted price assigned to it as a result of sales promotions. This price would be valid as long as the promotion is running.

- $hasDiscountPrice(p,\ dpr)$, where dpr is a DiscountPrice ID.
- $hasDiscountPriceStartDate(dpr,\ start\_date)$, where $start\_date$ is a Datetime ID.
- $hasDiscountPriceEndDate(dpr,\ end\_date)$, where end_date is a Datetime ID.

In order to keep track of a product's original price, the following predicate is also kept in the knowledgebase:

- $hasBasePrice(p,\ bpr)$, where bpr is a Price ID.

Thus, at any time a product's current price is either equal to its base price or a discounted price assigned to it.

$$\begin{aligned}(\forall\, p,\ dpr,\ start\_date,\ end\_date) & Product(p) \wedge hasDiscountPrice(p,\ dpr) \wedge \\ & hasDiscountPriceStartDate(dpr,\ start\_date) \wedge \\ & hasDiscountPriceEndDate(dpr,\ end\_date) \wedge \\ & CurrentDatetime(cur\_dt) \wedge before(start\_date,\ cur\_dt) \wedge \\ & before(cur\_date,\ end\_dt) \supset hasCurrentPrice(p,\ dpr). \qquad \text{(FOL a-8)}\end{aligned}$$

$$(\forall\, p,\ bpr,\ dpr,\ start\_date,\ end\_date) Product(p) \wedge hasDiscountPrice(p,\ dpr) \wedge$$
$$hasDiscountPriceStartDate(dpr,\ start\_date) \wedge$$
$$hasDiscountPriceEndDate(dpr,\ end\_date) \wedge$$
$$currentDatetime(cur\_dt) \wedge (before(cur\_date,\ start\_date) \vee$$
$$before(end\_date,\ cur\_dt)) \wedge hasBasePrice(p,\ bpr) \supset$$
$$hasCurrentPrice(p,\ bpr). \qquad \text{(FOL a-9)}$$

3.5.2 Cost of Goods Sold

The cost of goods sold (COGS) in retailing is a term used to refer to the price paid for the product plus any additional costs necessary to get the merchandise into inventory and ready for sale. Every product has a cost of goods sold assigned to it:

$$(\forall\, p)\ Product(p) \supset (\exists\, cogs)\ hasCostOfGoodsSold(p,\ cogs) \qquad \text{(FOL a-10)}$$

3.5.3 Tax

Tax refers to a fee charged by a government on a product. Different geographic zones may have different tax rates assigned to them. Tax is a concept in the ontology with the following properties:

$hasZone(tx,\ tx\_zone)$, where *tx_zone* is a Zone ID defined below,

$hasRate(tx,\ tx\_rate)$, where *tx_rate* is a floating point number,

$hasDescription(tx,\ tx\_desc)$, where $tx\_desc \in$ {provincial, federal, ecological, ...}

Zone (*zone*) is a concept in the ontology with the following properties:

- $hasParentZone(zone,\ par\_zone)$,
- $hasPostalCode(zone,\ pcode)$,
- $hasLevel(zone,\ z\_level)$: the level of the zone with respect to its parents,
- $hasCountry(zone,\ country)$.

3.5.4 Gross Product Margin

A product's margin is a measure of profitability and is calculated using the following formula:

$$margin = \frac{sales - cost}{sales},$$

where, *sales* is the selling price of a product, and *cost* refers to the cost of goods sold associated with that product. Thus,

$$\begin{aligned}(\forall\, p,\, \exists\, m)\ & Product(p) \wedge hasGPMargin(p,\, m) \equiv \\ & (\forall\, pr,\, c,\, val)\ Price(pr) \wedge hasCurrentPrice(p,\, pr) \wedge hasValue(pr,\, val) \\ & hasCostOfGoodsSold(p,\, c) \wedge m = 1 - c/val. \end{aligned} \quad \text{(FOL a-11)}$$

Using this information, it is possible to group products according to their margin by defining GPM groups.

- $GPMGroup(gpm\_group)$,
- $hasGPMMin(gpm\_group,\, gpm\_min)$,
- $hasGPMMax(gpm\_group,\, gpm\_max)$.

A product belongs to a GPM group if its gross product margin falls in that group's range:

$$\begin{aligned}(\forall\, p,\, m,\, & gpm\_group,\, gpm\_min,\, gpm\_max)\ Product(p) \wedge hasGPMargin(p,\, m) \\ & \wedge GPMGroup(gpm\_group) \wedge hasGPMMin(gpm\_group,\, gpm\_min) \wedge \\ & hasGPMMax(gpm\_group,\, gpm\_max) \wedge m \geq gpm\_min \wedge m \leq gpm\_max \supset \\ & belongsToGPMGroup(p,\, gpm\_group). \end{aligned} \quad \text{(FOL a-12)}$$

The two most important GPM groups are those which refer to the highest and lowest margins. Assuming that the groups don't overlap, the highest GPM group can be defined as the group which has a minimum value greater than the minimum value of all other groups that include at least one product:

$$
\begin{aligned}
&(\forall\ gpm\_group_1,\ ...,\ gpm\_group_n,\ min\_value_1,\ ..., min\_value_n)\\
&\quad HighestGPMGroup(gpm\_group_1) \equiv GPMGroup(gpm\_group_1) \wedge\\
&\quad hasGPMMin(gpm\_group_1,\ min\_value_1) \wedge\\
&\quad hasGPMMax(gpm\_group_1,\ max\_value_1) \wedge ... \wedge\\
&\quad GPMGroup(gpm\_group_n) \wedge hasGPMMin(gpm\_group_n,\ min\_value_n) \wedge\\
&\quad hasGPMMax(gpm\_group_n,\ max\_value_n) \wedge\\
&\quad \{(\exists\ p)\ Product(p) \wedge belongsToGPMGroup(p,\ gpm\_group_1)\} \wedge\\
&\quad \{(min\_value_1 \geq min\_value_2) \vee (min\_value_1 < min\_value_2 \wedge\\
&\qquad (\neg\exists\ p')\ Product(p') \wedge belongsToGPMGroup(p',\ gpm\_group_2)\} \wedge ... \wedge\\
&\quad \{(min\_value_1 \geq min\_value_n) \vee (min\_value_1 < min\_value_n \wedge\\
&\qquad (\neg\exists\ p')\ Product(p') \wedge belongsToGPMGroup(p',\ gpm\_group_n)\}
\end{aligned}
$$

(FOL a-13)

Similarly, the lowest GPM group can be defined as the group which has a minimum value less than the minimum value of all other groups that include at least one product:

$$
\begin{aligned}
&(\forall\ gpm\_group_1,\ ...,\ gpm\_group_n,\ min\_value_1,\ ..., min\_value_n)\\
&\quad LowestGPMGroup(gpm\_group_1) \equiv GPMGroup(gpm\_group_1) \wedge\\
&\quad hasGPMMin(gpm\_group_1,\ min\_value_1) \wedge\\
&\quad hasGPMMax(gpm\_group_1,\ max\_value_1) \wedge ... \wedge\\
&\quad GPMGroup(gpm\_group_n) \wedge hasGPMMin(gpm\_group_n,\ min\_value_n) \wedge\\
&\quad hasGPMMax(gpm\_group_n,\ max\_value_n) \wedge\\
&\quad \{(\exists\ p)\ Product(p) \wedge belongsToGPMGroup(p,\ gpm\_group_1)\} \wedge\\
&\quad \{(min\_value_1 \leq min\_value_2) \vee (min\_value_1 > min\_value_2 \wedge\\
&\qquad (\neg\exists\ p')\ Product(p') \wedge belongsToGPMGroup(p',\ gpm\_group_2)\} \wedge ... \wedge\\
&\quad \{(min\_value_1 \leq min\_value_n) \vee (min\_value_1 > min\_value_n \wedge\\
&\qquad (\neg\exists\ p')\ Product(p') \wedge belongsToGPMGroup(p',\ gpm\_group_n)\}
\end{aligned}
$$

(FOL a-14)

3.6 Catalog

Definition: Catalog

A catalog is an organized, detailed, descriptive list of a company's items for sale arranged systematically in order for customers to view. A catalog (cg) is comprised of products and categories, and must contain at least one product.

$$(\forall\, cg)\ Catalog(cg) \supset (\exists\, p)\ Product(p) \wedge containsProduct(cg,\ p). \qquad \text{(FOL a-15)}$$

The currency of a catalog defines the currency of the prices of products in that catalog.

$$(\forall\, cg,\ \exists\, cur)\ Catalog(cg) \wedge Currency(cur) \wedge hasCurrency(cg,\ cur) \equiv$$
$$(\forall\, p,\ pr)\ Product(p) \wedge containsProduct(cg,\ p) \wedge$$
$$Price(pr) \wedge hasCurrentPrice(p,\ pr) \wedge hasCurrency(pr,\ cur). \qquad \text{(FOL a-16)}$$

It is assumed that every company (described in the next chapter) has at least one catalog. This is shown by the following axiom:

$$(\forall\, com)\ Company(com) \supset (\exists\, cg)\ Catalog(cg) \wedge hasCatalog(com,\ cg) \qquad \text{(FOL a-17)}$$

3.6.1 Online Catalog

Definition: Online Catalog

Online catalogs are special kinds of catalogs that provide shoppers with the ability to navigate and analyze product information online. An *OnlineCatalog* (online_cg)is a subclass of *Catalog* with the following additional property:

- $hasWebsite(online\_cg,\ wsite)$, where, $wsite$ is a Website ID (described later).

3.7 Competency Questions Revisited

In this section we repeat the competency questions and follow each with the query that would provide the answer. Note that parameters that are preceded with a "?" are

variables. If a query is not existentially quantified, then it is a query for free variable bindings.

- What is the current physical count of an item in the inventory?

  - **Query 1:**

    $$InventoryItem(?inv) \wedge Product(?p) \wedge hasProduct(?inv,\ ?p) \wedge hasQuantity(?inv,\ ?q).$$

- Are there any product shortages?

  - **Query 2:**

    $$(\exists\ ?inv)\ InventoryItem(?inv) \wedge hasQuantity(?inv,\ ?q) \wedge ?q < 1.$$

- Which items of stock on hand have reached their specified reorder point?

  - **Query 3:**

    $$InventoryItem(?inv) \wedge reachedReorderLevel(?inv,\ true).$$

- Which items of stock on hand have the highest margin?

  - **Query 4:**

    $$HighestGPMGroup(?g) \wedge Product(?p) \wedge belongsToGPMGroup(?p,\ ?g).$$

- Which items of stock on hand have the lowest margin?

  - **Query 5:**

    $$LowestGPMGroup(?g) \wedge Product(?p) \wedge belongsToGPMGroup(?p,\ ?g).$$

3.8 Summary

Effective inventory management enables an organization to meet or exceed customers' expectations of product availability while maximizing net profits or minimizing costs. In this chapter, useful entities, attributes, and relations for inventory management were introduced, and a set of competency questions were presented to demonstrate the use of the terminology.

Chapter 4

Purchasing

4.1 Introduction

Logistics and distribution is the strategic process that links organizations with their suppliers and customers and refers to the activity of getting products from the source of production into the hands of customers. Distribution refers to the movement of physical goods between stages in a supply chain, and logistics to both material flow and information flow [Harrison and White, 2006]. Logistics and distribution involves inventory management, purchasing, transport, warehousing, and the organizing and planning of these activities.

Purchase is the most common type of financial transaction, where an item is changed for money. In commerce, a retailer purchases goods and services the company needs either to resell to end-user customers or for the establishment's own use from manufacturers or importers, either directly or through a wholesaler. Products can be purchased either at stores or shops, or ordered via mail, telephone or online without having been examined physically but instead in a catalog, on television or on a website.

When choosing suppliers and merchandise, purchasing professionals consider price, quality, availability, reliability, and technical support. They try to get the best deal for

their company, meaning the highest quality goods and services at the lowest possible cost to their companies. The following are some competency questions related to purchasing.

- Which suppliers provide a product?
- Which suppliers are delivering the best value for a product?
- What is the minimum order size for a product?
- How much order lead-time is required to take shipment of a product?
- Which orders have been canceled in a time period specified by the starting time point s and the ending time point e?
- Which payments have not been authorized by the assigned payment provider in a time period specified by the starting time point s and the ending time point e?
- Where do customers enter the site and where do they exit?

This chapter presents the entities, attributes and relations needed for answering the above competency questions. Sections 5.2 and 5.3 present the basic terminology needed for representing retail organizations and suppliers. Section 5.4 is concerned with the definition of orders and introduces customer and purchase orders (customer order will be used to answer competency questions in later chapters). The concepts of payment and payment provider are described in Sections 5.5 and 5.6, followed by the definitions of shipment and shipping courier in Sections 5.7 and 5.8. Section 5.9 provides answers for the competency questions, and Section 5.10 concludes this chapter with a summary.

4.2 Retail Organization

Retailing consists of those business activities involved in the sale of goods to consumers for their personal, family, or household use. In commerce, a retailer purchases products in

large quantities from manufacturers or importers, either directly or through a wholesaler, and then sells individual items or small quantities to the general public or end user customers.

A company, in the retail ontology, is a business organization concerned with retailing activities. A company may belong to a larger corporation or may have smaller child companies.

$$(\forall\ com,\ par\_com)\ Company(com) \wedge Company(par\_com) \supset$$
$$(hasParentCompany(com,\ par\_com) \equiv parentCompanyOf(par\_com,\ com)) \quad \text{(FOL b-1)}$$

A division or operating unit of a company is called a business unit. Common business units are accounting, human resources, marketing and sales, purchasing, administration, and management.

$$(\forall\ com,\ bu)\ Company(com) \wedge Division(bu) \supset$$
$$(hasDivision(com,\ bu) \equiv divisionOf(bu,\ com)). \quad \text{(FOL b-2)}$$

A store is referred to the physical location where merchandise is offered for sale. The virtual version of a store, where products are offered over the internet, is called a webstore (a.k.a. online shop, e-shop, online store). Every company may have zero or more stores or webstores, but in order to be considered as a retail entity, it must have at least one store or webstore.

$$(\forall\ com,\ st)\ Company(com) \wedge Store(st) \supset$$
$$(hasStore(com,\ st) \equiv storeOf(st,\ com)). \quad \text{(FOL b-3)}$$

$$(\forall\ st)\ Store(st) \supset (\exists\ st\_loc)\ hasLocation(st,\ st\_loc). \quad \text{(FOL b-4)}$$

$$(\forall\ com,\ wstore)\ Company(com) \wedge Webstore(wstore) \supset$$
$$(hasWebstore(com, wstore)\ \equiv webstoreOf(wstore,\ com)). \quad \text{(FOL b-5)}$$

$$(\forall\, com)\; Company(com) \supset (\exists\, x)\; (Store(x) \wedge hasStore(com,\; x)) \vee$$
$$(Webstore(x) \wedge hasWebstore(com,\; x)). \qquad \text{(FOL b-6)}$$

Webstores operate through websites. A website is a collection of web pages, images, videos and other digital assets and is hosted on a particular domain or subdomain on the World Wide Web.

$$(\forall\, wsite,\; wstore) Website(wsite) \wedge Webstore(wstore) \supset$$
$$(websiteOf(wsite,\; wstore) \equiv hasWebsite(wstore,\; wsite)). \qquad \text{(FOL b-7)}$$

For a website to be viewed, it must be stored on a server that is connected to the internet. An organization that provides this service is known as a host, and the service it provides is hosting. A website host is a concept in the ontology (*WebsiteHost(host)*) with the following attributes:

- $hasHostName(host,\; h\_name)$,
- $hasSecureHostName(host,\; h\_sname)$,
- $hasURI(host,\; uri)$.

Thus, if a website can be viewed then it must be associated with a host:

$$(\forall\, wsite)\; Website(wsite) \wedge isAvailableOnline(wsite,\; true) \supset$$
$$(\exists\, host)\; WebsiteHost(host) \wedge hasWebsiteHost(wsite,\; host). \qquad \text{(FOL b-8)}$$

A retail organization's top level general goals include, but are not limited to:

1. Satisfying the needs and expectations of its target market, employees, and management efficiently and effectively;
2. Informing and convincing consumers to buy products offered by the company;
3. Improving the quality of customers' shopping experience;
4. Increasing profits and reducing costs.

4.2.1 Website Related Concepts

4.2.1.1 Web Page

A web page is a document on the world wide web and can be accessed through a web browser.

$$(\forall\, wpage)\, WebPage(wpage)$$
$$\supset (\exists\, wsite)\, Website(wsite) \wedge belongsTo(wpage,\, wsite). \qquad \text{(FOL b-9)}$$

Web pages are organized with respect to each other:

$$(\forall\, wpage,\, p\_wpage)\, WebPage(wpage) \wedge WebPage(p\_wpage) \supset$$
$$(hasParentWebPage(wpage,\, p\_wpage) \equiv$$
$$parentWebPageOf(p\_wpage,\, wpage)). \qquad \text{(FOL b-10)}$$

4.2.1.2 Session

A session is the time spent by a single user at a website. A new session is created for each visitor upon their first visit. If the visitor comes back to the site within a specified time period after a session is created, then the session is still valid. If the visitor returns to the site after the allotted time period has expired, then a new user session is created. The recording of the path taken through the site is referred to as session tracking. Session tracking can be used to store user generated information or help in refining site structure by recording behavior patterns of users. A session is a concept in the ontology with the following attributes:

- $Session(s)$,
- $hasEntryTime(s,\, c\_dt)$,
- $hasClientIPAddress(s,\, ip)$,
- $hasEntryURL(s,\, entry\_url)$,

- $hasExitURL(s,\ exit\_url)$,
- $hasIdentifier(s,\ iden)$, where $iden$ is a random number,
- $purchasedIn(s,\ x)$, where x is either true or false.

4.2.1.3 Page View

An instance of viewing a webpage is called a page view. Each page view stores information about the webpage viewed, the previous webpage viewed, view time, the session in which the page was viewed, and a purchase order in case the user has added items to the shopping cart.

- $PageView(pview)$,
- $hasWebpage(pview,\ wpage)$,
- $hasHitDatetime(pview,\ v\_dt)$,
- $hasSession(pview,\ s)$,
- $hasCustomerOrder(pview,\ co)$.

If a customer order (defined later in thish chapter) with status *completed* is associated with the page view, then the user has purchased in the session in which the page was viewed:

$$(\forall\ pview,\ s,\ po)\ PageView(pview) \wedge Session(s) \wedge CustomerOrder(co) \wedge$$
$$hasSession(pview,\ s) \wedge hasCustomerOrder(pview,\ co) \wedge$$
$$hasStatus(co,\ \text{`}completed\text{'}) \supset purchasedIn(s,\ true) \qquad \text{(FOL b-11)}$$

4.3 Supplier

A company which supplies products or services to another company is called a supplier. The products of a supplier may be incorporated into a company's catalog or another supplier's catalog.

$$(\forall\ com,\ supp)\ Company(com) \wedge Supplier(supp) \supset$$
$$(hasSupplier(com,\ supp) \equiv supplies(supp,\ com)). \qquad \text{(FOL b-12)}$$

Each supplier has at least one catalog. A supplier's catalog is represented by the concept *SupplierCatalog* which is a subclass of *Catalog*:

$$(\forall\ scg)\ SupplierCatalog(scg) \supset Catalog(scg). \qquad \text{(FOL b-13)}$$
$$(\forall\ supp)\ Supplier(supp) \supset (\exists\ scg)\ SupplierCatalog(scg) \wedge$$
$$hasSupplierCatalog(supp,\ scg). \qquad \text{(FOL b-14)}$$

In addition to the attributes inherited from *Catalog*, a supplier's catalog also contains the availability information of products in that catalog:

- $AvailabilityInfo(a\_info)$,
- $hasAvailabilityInfo(scg,\ a\_info)$.

The availability information consists of information lines, each linked to a product in the catalog and having the following attributes:

- $AvailabilityInfoLine(a\_info\_line)$,
- $hasInfoLine(a\_info,\ a\_info\_line)$,
- $hasProduct(a\_info\_line,\ p)$,
- $hasQuantity(a\_info\_line,\ q)$,
- $hasMinOrderSize(a\_info\_line,\ min\_size)$, where *min_size* is a Price ID. Orders under this value will not be accepted.

- $includesDelivery(a\_info\_line,\ x)$, where x is either true or false.
- $hasOrderProcessTime(a\_info\_line,\ op\_time)$, where op_time refers to the amount of time it takes to process an order.

A supplier delivers the best value for a product if no other supplier providing a better offer for the same product exists:

$$
\begin{aligned}
&(\forall\ supp,\ scg,\ p,\ pr,\ cur,\ val,\ ai,\ ai\_line)\ Supplier(supp) \wedge \\
&\quad SupplierCatalog(scg) \wedge hasSupplierCatalog(supp,\ scg) \wedge \\
&\quad AvailabilityInfo(ai) \wedge hasAvailabilityInfo(scg,\ ai) \wedge \\
&\quad AvailabilityInfoLine(ai\_line) \wedge hasInfoLine(ai,\ ai\_line) \wedge \\
&\quad hasProduct(ai\_line,\ p) \wedge hasCurrentPrice(p,\ pr) \wedge \\
&\quad hasCurrency(p,\ cur) \wedge hasValue(p,\ val) \wedge \{(\neg\exists\ supp',\ scg', \\
&\quad p',\ pr',\ cur',\ val',\ ai',\ ai\_line')\ Supplier(supp') \wedge supp \neq supp' \wedge \\
&\quad SupplierCatalog(scg') \wedge hasSupplierCatalog(supp',\ scg') \wedge \\
&\quad AvailabilityInfo(ai') \wedge hasAvailabilityInfo(scg',\ ai') \wedge \\
&\quad AvailabilityInfoLine(ai\_line') \wedge hasInfoLine(ai',\ ai\_line') \wedge \\
&\quad hasProduct(ai\_line',\ p') \wedge p = p' \wedge hasCurrentPrice(p',\ pr') \wedge \\
&\quad hasCurrency(p',\ cur') \wedge cur = cur' \wedge hasValue(p',\ val') \wedge val' < val\} \supset \\
&\quad deliversBestValue(ail,\ true) \qquad \text{(FOL b-15)}
\end{aligned}
$$

4.4 Order

An order is a request for materials or services. From a sales perspective, it is a customer's request (actual or forecasted) for an enterprise's goods produced for sale.

According to [Dobler and Burt, 1996], a Purchase Order (PO) is a commercial document issued by a buyer to a seller, indicating the type, quantities and agreed prices for products or services that the seller will provide to the buyer. Sending a PO to a supplier

constitutes a legal offer to buy products or services. Acceptance of a PO by a seller usually forms a once-off contract between the buyer and seller so no contract exists until the PO is accepted.

In TOVE, four generic and identifiable types of orders are defined: customer, internal, forecast, and purchase orders. The customer and purchase orders are of interest to the Retail Ontology. A customer order is a request for items by a customer that is external to the organization of the enterprise, whereas a purchase order is a request for parts that are not produced within a company.

Within a retail environment, purchase occurs at two different levels:

1. Retailers purchase products in large quantities from suppliers. These purchases are represented by purchase orders.

2. Customers purchase individual items or small quantities from retailers. These purchases are represented by customer orders.

In the Retail Ontology, similar to TOVE, an order consists of one or more order line items and contains information about the purchase time, shipping and destination information, and shipping and handling costs that apply to the order.

- $Order(ord)$,
- $OrderLineItem(oli)$,
- $hasOrderLineItem(ord,\ oli)$,
- $hasCreateDatetime(ord,\ p\_dt)$,
- $hasShipment(ord,\ sh)$, (defined later),
- $hasStatus(ord,\ p\_status)$.

The status of an order can have one of the following values:

1. Updating: The order is in this state while items are being added to the order.
2. Executing: The order is in this state while payment is being authorized and fraud is being handled.
3. Held: The order is held for future authorization, fraud check or payment receive.
4. Canceled: The whole order was canceled.
5. Executed: Ready to be exported to the merchant's order processing system (in the case of online purchasing).
6. Completed: Payments have been received and the order is completed; all items have been shipped.

Each order line refers to an individual item and has the following properties:

- $hasProduct(oli,\ p)$,
- $hasQuantity(oli,\ q)$,
- $hasLineBaseTotal(oli,\ line\_btot)$,
- $hasTax(oli,\ line\_tax)$,
- $hasLineCost(oli,\ cost)$,

$$(\forall\ oli,\ p,\ c,\ q)\ OrderLineItem(oli) \wedge Product(p) \wedge hasProduct(oli,\ p) \wedge hasCostOfGoodsSold(p,\ c) \wedge hasQuantity(oli,\ q) \supset hasLineCost(oli,\ MUL(c,\ q)). \quad \text{(FOL b-16)}$$

A product belongs to an order if it belongs to one of the order line items in that order.

$$(\forall\ ord,\ pol,\ p)\ Order(ord) \wedge Product(p) \wedge hasProduct(ord,\ p) \equiv OrderLineItem(pol) \wedge hasOrderLineItem(ord,\ pol) \wedge hasProduct(pol,\ p). \quad \text{(FOL b-17)}$$

The base total amount of an order line, *hasLineBaseTotal*, is equal to the price of the item the line refers to multiplied by the quantity specified.

$$(\forall\ oli,\ p,\ pr,\ q)\ OrderLineItem(oli) \wedge Product(p) \wedge hasProduct(oli,\ p) \wedge$$
$$Price(pr) \wedge hasPrice(p,\ pr) \wedge hasValue(pr,\ val) \wedge hasQuantity(oli,\ q) \supset$$
$$hasLineBaseTotal(oli,\ MUL(val,\ q)). \qquad \text{(FOL b-18)}$$

The total value of an order line after applying taxes, *hasLineTaxedTotal*, is calculated as follows:

$$(\forall\ oli,\ b\_tot, \exists\ tot)\ OrderLineItem(oli) \wedge hasLineBaseTotal(oli,\ b\_tot) \wedge$$
$$hasLineTaxedTotal(oli,\ tot) \equiv (\forall\ tax_1,\ ...,\ tax_n,\ rate_1,\ ...,\ rate_n)$$
$$Tax(tax_1) \wedge hasTax(oli,\ tax_1) \wedge hasRate(tax_1,\ rate_1) \wedge ... \wedge Tax(tax_n) \wedge$$
$$hasTax(oli,\ tax_n) \wedge hasRate(tax_n,\ rate_n) \wedge$$
$$tot = (rate_1 + ... + rate_n + 1) \times b\_tot. \qquad \text{(FOL b-19)}$$

The total value of an order before applying shipping costs, *hasBaseTotal*, is equal to the sum of the taxed total values of each line in that order.

$$(\forall\ ord,\ \exists\ tot)\ Order(ord) \wedge hasBaseTotal(ord,\ tot) \equiv$$
$$(\forall\ pol_1,\ ...,\ pol_n,\ line\_btot_1,\ ...,\ line\_btot_n)\ OrderLineItem(pol_1) \wedge ... \wedge$$
$$OrderLineItem(pol_n) \wedge hasOrderLineItem(ord,\ pol_1) \wedge$$
$$hasLineTaxedTotal(pol_1,\ line\_btot_1) \wedge ... \wedge hasOrderLineItem(ord,\ pol_n) \wedge$$
$$hasLineTaxedTotal(pol_n,\ line\_btot_n) \wedge tot = line\_btot_1 + ... + line\_btot_n.$$
$$\text{(FOL b-20)}$$

The total value of an order, *hasTotal*, after adding shipping costs will be given when *Shipment* is presented later in this chapter.

4.4.1 Customer Order

A customer order is a request for items by a customer that is external to the organization of the enterprise. A *CustomerOrder* is a subclass of *Order* and is associated with a

customer account (defined later).

$$(\forall\ co)\ CustomerOrder(co) \supset (\exists\ ca)\ CustomerAccount(ca) \wedge$$
$$hasCustomerAccount(co,\ ca) \wedge (\neg\exists\ ca')\ CustomerAccount(ca') \wedge$$
$$hasCustomerAccount(co,\ ca') \wedge ca \neq ca' \qquad \text{(FOL b-21)}$$

In addition to the attributes inherited from *Order*, a customer order also has the following attributes:

- $hasCoupon(co,\ cp)$, (defined later),
- $hasTotalAfterCoupon(co,\ c\_tot)$, the formula for calculating this value is given later in Chapter ... when discussing promotions and coupons,
- $hasProductCost(co,\ cost)$. The total cost of goods sold associated with a customer order equals the sum of the costs of each order lines:

$$(\forall\ co,\ \exists\ cost)\ CustomerOrder(co) \wedge hasProductCost(co,\ cost) \equiv$$
$$(\forall\ pol_1,\ ...,\ pol_n,\ line\_cost_1,\ ...,\ line\_cost_n)\ OrderLineItem(pol_1) \wedge ... \wedge$$
$$OrderLineItem(pol_n) \wedge hasOrderLineItem(co,\ pol_1) \wedge$$
$$hasLineCost(pol,\ line\_cost_1) \wedge ... \wedge hasOrderLineItem(co,\ pol_n) \wedge$$
$$hasLineCost(pol,\ line\_cost_n) \wedge cost = line\_cost_1 + ... + line\_cost_n.$$
$$\text{(FOL b-22)}$$

4.4.2 Purchase Order

A purchase order is a request for parts that are not produced within a company. In the context of the Retail Ontology, this refers to the items purchased for resell to end-user customers. A *PurchaseOrder* is a subclass of *Order* and is associated with a supplier.

$$(\forall\ po)\ PurchaseOrder(po) \supset (\exists\ supp)\ Supplier(supp) \wedge hasSupplier(po,\ supp) \wedge$$
$$(\neg\exists\ supp')\ Supplier(supp') \wedge hasSupplier(po,\ supp' \wedge supp \neq supp')$$
$$\text{(FOL b-23)}$$

Time it takes for products to be ordered and received so that selling can begin is referred to as *order lead-time*. To every purchase order is assigned an order lead-time:

- $hasOrderLeadTime(po,\ ol\_time)$.

The formula for calculating this time is given in Section 5.7.

4.5 Payment

A payment is the transfer of wealth from buyer to the seller, made in exchange for the provision of goods and services. Common means of payment by an individual include money, check, debit, credit, or bank transfer. Payment (pay) is a concept in the ontology. A payment is made either by a company or a customer and has an amount and a status.

- $pays(x,\ pay)$, where x is either a CustomerAccount ID (defined later) or a Company ID.

$$(\forall\ pay,\ x)\ Payment(pay) \wedge pays(x,\ pay) \supset$$
$$CustomerAccount(x) \vee Company(x). \qquad \text{(FOL b-24)}$$

- $hasAmount(pay,\ pay\_amount)$,
- $hasStatus(pay,\ pay\_status)$.

The Status of a payment can have one of the following values:

1. New: The payment is in this state when it is generated and saved in the system. The payment can be either correct or incorrect. If the new payment is correct it can be confirmed and passed for execution, edited or deleted. If the new payment is incorrect it can be edited or deleted.
2. Executing: The payment is in this state while it is being authorized and fraud is being handled.

3. Canceled: The whole payment was canceled.

4. Rejected: The payment is rejected by the payment provider (described in the next section).

5. Accepted: The payment is authorized.

Each payment is associated with only one order:

$$(\forall\, pay)\ Payment(pay) \supset (\exists\, ord)\ Order(ord) \wedge paymentFor(pay,\ ord) \\ \wedge \neg(\exists\, ord')\ Order(ord') \wedge paymentFor(pay,\ ord') \wedge ord \neq ord'. \quad \text{(FOL b-25)}$$

Each payment has a payment type. Possible values for payment type are credit-card, debit-card, gift-certificate, store-credit, cheque, certified-cheque, money-order, and cash.

$$(\forall\, pay)\ Payment(pay) \supset (\exists\, pay\_type)\ hasType(pay,\ pay\_type). \quad \text{(FOL b-26)}$$

When payment type is not cash, the payment is linked to some payment provider who authorizes it:

$$(\forall\, pay)\ Payment(pay) \wedge \neg hasType(pay, `cash') \supset \\ (\exists\, pay\_pro)\ PaymentProvider(pay\_pro) \wedge\ hasProvider(pay,\ pay\_pro). \quad \text{(FOL b-27)}$$

4.6 Payment Provider

A payment provider offers retailers online services for accepting electronic payments by credit card or other payment methods such as payments based on online banking. The basic job of the payment provider is to collect a customer payment details in a secure manner, examine them for validity then ensure that the necessary payments are transferred securely to a bank or merchant account.

A payment provider is an organization independent of the retail system, who supports specified payment types in order to authorize payments.

$$(\forall\, pay\_pro,\ pay\_type)\ PaymentProvider(pay\_pro) \supset$$
$$supportsType(pay\_pro,\ pay\_type). \qquad \text{(FOL b-28)}$$

If a company operates a webstore, then it must have at least one payment provider assigned to that webstore:

$$(\forall\, x)\ Webstore(x) \supset$$
$$(\exists\, y)\ PaymentProvider(y) \wedge hasPaymentProvider(x,\ y). \qquad \text{(FOL b-29)}$$

4.7 Shipment

An instance of shipping goods is called a shipment. Every shipment must have an order and an address attached to it.

$$(\forall\, sh,\ ord)\ Shipment(sh) \wedge Order(ord) \supset$$
$$(hasOrder(sh,\ ord) \equiv hasShipment(ord,\ sh)). \qquad \text{(FOL b-30)}$$

$$(\forall\, sh)\ Shipment(sh) \supset (\exists\, ord)\ Order(x) \wedge hasOrder(sh,\ ord). \qquad \text{(FOL b-31)}$$
$$(\forall\, sh,\ \exists\, adr)\ Shipment(sh) \supset Address(adr) \wedge hasAddress(sh,\ adr). \qquad \text{(FOL b-32)}$$

In addition, shipment has the following attributes:

- $hasExpectedShipDate(sh,\ e\_dt)$,
- $hasRequestedDeliveryDate(sh,\ r\_dt)$,
- $hasShippingZone(sh,\ sh\_zone)$, where sh_zone is a Zone ID,
- $hasTotalQuantity(sh,\ q)$,
- $hasHandlingTotal(sh,\ h\_tot)$,
- $hasShippingTotal(sh,\ s\_tot)$,

- $hasHandlingTax(sh,\ h\_tax)$,
- $hasShippingTax(sh,\ s\_tax)$,
- $hasShippingCourier(sh,\ s\_c)$, (defined in the next section).

The total shipment cost, *hasShippingTotal*, is equal to the sum of the total handling and shipping costs plus taxes applied:

$$(\forall\ sh,\ h\_tot,\ s\_tot \exists\ tot)\ Shipment(sh) \wedge hasHandlingTotal(sh,\ h\_tot) \wedge hasShippingTotal(sh,\ s\_tot) \equiv (\forall\ tax\_h_1, ...,\ tax\_h_n,\ rate\_h_1, ...,\ rate\_h_n,\ tax\_s_1, ...,\ tax\_s_n,\ rate\_s_1, ..., rate\_s_n)\ Tax(tax\_h_1) \wedge hasHandlingTax(sh,\ tax\_h_1) \wedge hasRate(tax\_h_1,\ rate\_h_1) \wedge ... \wedge Tax(tax\_h_n) \wedge hasHandlingTax(sh,\ tax\_h_n) \wedge hasRate(tax\_h_n,\ rate\_h_n) \wedge Tax(tax\_s_1) \wedge hasShippingTax(sh,\ tax\_s_1) \wedge hasRate(tax\_s_1,\ rate\_s_1) \wedge ... \wedge Tax(tax\_s_n) \wedge hasShippingTax(sh,\ tax\_s_n) \wedge hasRate(tax\_s_n,\ rate\_s_n) \wedge tot = (rate\_h_1 + ... + rate\_h_n + 1) \times h\_tot + (rate\_s_1 + ... + rate\_s_n + 1) \times s\_tot. \quad \text{(FOL b-33)}$$

Having calculated the shipping total, an order's total cost can be calculated as follows:

$$(\forall\ ord,\ sh,\ b\_tot,\ s\_tot)\ Order(ord) \wedge Shipment(sh) \wedge hasShipment(ord,\ sh) \wedge hasBaseTotal(ord,\ b\_tot) \wedge hasShippingTotal(sh,\ s\_tot) \supset hasTotal(ord,\ ADD(b\_tot,\ s\_tot)). \quad \text{(FOL b-34)}$$

4.8 Shipping Courier

For online ordering or when products are ordered via mail or telephone, once a payment has been accepted the products can be delivered in the following ways.

1. Shipping: The product is shipped to the customer's address.

2. In-store pickup: The customer orders online, finds a local store, and picks the product up at the closest store.

3. Download: This is the method often used for digital media products such as software, music, movies, or images.

When the delivery type is shipping, usually a courier is employed to deliver packages to customers.

A shipping courier is an organization independent of the retail system, whose job is to deliver products after they have been ordered and payment has been accepted.

- $ShippingCourier(scour)$,
- $hasContactInfo(scour,\ c\_info)$, where *c_info* is a ContactInfo ID,
- $hasShippingInfo(scour,\ s\_info)$, where *S_info* is a ShippingInfo ID.

Shipping information is a concept used to keep information about available shipping speeds, their price, minimum and maximum delivery times, and the zones the information applies to.

- $ShippingInfo(s\_info)$,
- $hasCurrency(s\_info,\ cur)$,
- $hasMaxDays(s\_info,\ max\_days)$,
- $hasMinDays(s\_info,\ min\_days)$,
- $hasZone(s\_info,\ s\_zone)$, where *s_zone* is a Zone ID,
- $hasShippingSpeed(s\_info,\ s\_speed)$,
- $hasPerItemHandlingPrice(s\_info,\ ih\_pr)$,
- $hasPerItemMinHandlingPrice(s\_info,\ ih\_min\_pr)$,

- $hasPerItemShippingPrice(s\_info,\ is\_pr)$,
- $hasPerItemMinShippingPrice(s\_info,\ is\_min\_pr)$,
- $hasPerShipmentHandlingPrice(s\_info,\ sh\_pr)$,
- $hasPerShipmentMinHandlingPrice(s\_info,\ sh\_min\_pr)$,
- $hasPerShipmentShippingPrice(s\_info,\ ss\_pr)$,
- $hasPerShipmentMinShippingPrice(s\_info,\ ss\_min\_pr)$.

Shipper's contact information has the following attributes:

- $ContactInfo(c\_info)$,
- $hasContactPersonName(c\_info,\ c\_name)$,
- $hasAddress(c\_info,\ c\_adr)$,
- $hasEmail(c\_info,\ email)$,
- $hasTelephoneNumber(c\_info,\ tel)$.

Thus, an purchase order's lead-time is equal to the maximum order processing time of the products in that order plus the time it takes to ship the products, and can be calculated as follows:

$$
\begin{aligned}
&(\forall\ po,\ \exists\ ol\_time)\ PurchaseOrder(po) \wedge hasOrderLeadTime(po,\ ol\_time) \equiv \\
&\quad (\forall\ supp,\ scg,\ ai,\ ai\_line_1,\ ...,\ ai\_line_n,\ p_1,\ ..., p_n\ op\_time_1,\ ...,\ op\_time_n, \\
&\quad sh,\ scour,\ max\_time)\ Supplier(supp) \wedge hasSupplier(po,\ supp) \wedge \\
&\quad SupplierCatalog(scg) \wedge hasSupplierCatalog(supp,\ scg) \wedge \\
&\quad AvailabilityInfo(ai) \wedge hasAvailabilityInfo(scg,\ ai) \wedge \\
&\quad AvailabilityInfoLine(ai\_line_1) \wedge hasInfoLine(ai,\ ai\_line_1) \wedge \\
&\quad Product(p_1) \wedge hasProduct(po,\ p_1) \wedge hasProduct(ai\_line_1,\ p_1)
\end{aligned}
$$

$$hasOrderProcessTime(ai\_line_1,\ op\_time_1) \wedge ... \wedge$$
$$AvailabilityInfoLine(ai\_line_n) \wedge hasInfoLine(ai,\ ai\_line_n) \wedge$$
$$Product(p_n) \wedge hasProduct(po,\ p_n) \wedge hasProduct(ai\_line_n,\ p_n)$$
$$hasOrderProcessTime(ai\_line_n,\ op\_time_n) \wedge op\_time_1 \geq op\_time_2 \wedge ... \wedge$$
$$op\_time_1 \geq op\_time_n \wedge Shipment(sh) \wedge$$
$$hasShipment(po,\ sh) \wedge ShippingCourier(scour) \wedge$$
$$hasShippingCourier(sh,\ scour) \wedge hasMaxDays(scour,\ max\_time) \wedge$$
$$ol\_time = op\_time_1 + max\_time \qquad \text{(FOL b-35)}$$

4.9 Competency Questions Revisited

In this section we repeat the competency questions and follow each with the query that would provide the answer. Note that parameters that are preceded with a "?" are variables. If a query is not existentially quantified, then it is a query for free variable bindings.

- Which suppliers provide a product p?

  - **Query 1:**

    $$Supplier(?s) \wedge SupplierCatalog(?c) \wedge hasSupplierCatalog(?s,\ ?c) \wedge$$
    $$Product(\mathbf{p}) \wedge containsProduct(?c,\ \mathbf{p}).$$

- Which suppliers are delivering the best value for a product p?

  - **Query 2:**

    $$Supplier(?s) \wedge AvailabilityInfo(?inf) \wedge hasAvailabilityInfo(?s, ?inf)$$
    $$\wedge AvailabilityInfoLine(?ail) \wedge hasInfoLine(?inf,\ ?ail) \wedge$$
    $$Product(\mathbf{p}) \wedge hasProduct(?ail,\ \mathbf{p}) \wedge deliversBestValue(?ail,\ true).$$

- What is the minimum order size for a product?

- **Query 3:**

$$AvailabilityInfoLine(?ail) \wedge Product(textbfp) \wedge hasProduct(?ail,\ \mathbf{p}) \wedge hasMinOrderSize(?ail,\ ?x).$$

- How much order lead-time is required to take shipment of a product?

- **Query 4:**

$$PurchaseOrder(?po) \wedge Product(\mathbf{p}) \wedge hasProduct(?po,\ \mathbf{p}) \wedge hasOrderLeadTime(?po,\ ?otime).$$

- Which orders have been canceled in a time period specified by the starting time point s and the ending time point e?

- **Query 5:**

$$CustomerOrder(?co) \wedge hasStatus(?co,\ `canceled').$$

- Which payments have not been authorized by the assigned payment provider in a time period specified by the starting time point s and the ending time point e?

- **Query 6:**

$$Payment(?pay) \wedge hasStatus(?po,\ `rejected').$$

- Where do customers enter the site and where do they exit?

- **Query 7:**

$$Session(?s) \wedge hasEntryURL(?s,\ ?entry\_url) \wedge hasExitURL(?s,\ ?exit\_url).$$

4.10 Summary

Purchasing professionals try to get the best deal for their company, meaning the highest quality goods and services at the lowest possible cost to their companies. In this chapter, useful entities, attributes, and relations for effective purchasing were introduced, and a set of competency questions were presented to demonstrate the use of the terminology.

Chapter 5

Customer

5.1 Introduction

In customer-based or relationship-oriented marketing, a customer is placed at the center of the organization's universe and companies have to decide with which potential customers to engage in relationships [Hoekstra and Huizingh, 1999]. Cusomter relationship management (CRM) is a business strategy with the goal of improving customer satisfaction, profitability, and retention, as well as increasing company's revenue and reducing costs.

In this chapter the entities, attributes and relations needed for effective CRM are presented and the answers to following competency questions that are important for customer relationship management are provided.

- Who are the customers?
- Who are the most profitable customers?
- What is the purchase behaviour of a customer?
- How do buying trends compare across geographies?
- Which items have been purchased by customers who bought SKU x?

- What is the conversion rate in a time period specified by the starting time point s and the ending time point e (for a webstore)?
- What is the average order value of each customer?
- What is the expected average transaction value associated with a customer during a specified time period?

The organization of this chapter is as follows. In Section 6.2 and 6.3 the terminology of customer and customer account are presented. Section 6.4 is concerned with the definition of some useful concepts for CRM, namely, customer value, conversion rate, and customer lifetime value. Customer segmentation techniques are described in Section 6.5. Section 6.6 provides answers to the competency questions, and Section 6.7 concludes this chapter with a summary.

5.2 Customer

An organization-agent was defined in [Fox et al., 1997] as an individual member in the organization who is a member of some division, plays one or more roles in the organization, can perform activities, and communicates with other agents using communication-links (refer to TOVE's Organization Ontology for full axiomatization).

This section extends the organization-agent to capture the meaning of customers in a retail environment. A Customer is an person who uses the website to gain information, shop, manage account, purchase and handle previous purchases.

A person is a human being regarded as an individual[1] and is a concept in the ontology. p is a person if the following is asserted in the knowledgebase:

- $Person(p)$.

[1]Compact Oxford English Dictionary

Each person has the following attributes:

- $hasFirstName(p,\ f\_name)$,
- $hasLastName(p,\ l\_name)$,
- $hasGender(p,\ gen)$,
- $hasAddress(p,\ add)$,
- $hasEmail(p,\ email)$,
- $hasBirthdate(p,\ bd)$,
- $hasTelephoneNumber(p,\ tel)$.

where *f_name* and *l_name* are string values, *gen* is either female or male, and *add*, *email*, *bd*, and *tel* are an *Address*, *Email*, *Datatime* and *TelephoneNumber* IDs respectively.

In the online retailing environment, a customer is represented by the concpet *CustomerAccount* (defined in the next section) that is associated with a *Person*:

$$(\forall\ ca)\ CustomerAccount(ca) \supset (\exists\ p)\ Person(p) \wedge hasPerson(ca,\ p) \wedge$$
$$(\neg\exists\ p2)\ Person(p2) \wedge hasPerson(ca,\ p2) \wedge p \neq p2. \qquad \text{(FOL c-1)}$$

5.3 Customer Account

Customers can create customer accounts either in the process of placing their first order, or anytime by requesting to do so. Each *CustomerAccount* has the following attributes:

- $hasUserName(ca,\ u\_name)$,
- $hasPassword(ca,\ u\_pass)$,

where *u_name* and *u_pass* are string values. Every purchase a customer makes is identified by a *customer order*:

- $hasCustomerOrder(ca,\ co)$, where co is a CustomerOrder ID.

Thus, a customer has purchased a SKU if the corresponding item is in one of the completed orders belonging to that customer:

$$(\forall\ co,\ p,\ ca,\ x)\ CustomerOrder(co) \wedge Product(p) \wedge CustomerAccount(ca) \wedge SKU(x) \wedge hasSKU(p,\ x) \wedge hasProduct(co,\ p) \wedge hasCustomerOrder(ca,\ co) \supset purchasedSKU(ca,\ x). \quad \text{(FOL c-2)}$$

Two SKUs are linked through *purchasedBySameCustomer* if they are purchased by the same customer. This can be useful in identifying products to offer to a customer (more on this in the next chapter).

$$(\forall\ x,\ y)\ SKU(x) \wedge SKU(y) \wedge purchasedBySameCustomer(x,\ y) \equiv (\exists\ ca)\ CustomerAccount(ca) \wedge purchasedSKU(ca,\ x) \wedge purchasedSKU(ca,\ y) \wedge x \neq y. \quad \text{(FOL c-3)}$$

The FOL statements for calculating a customer's total amount purchased, purchase frequency, and average order value are given below.

$$(\forall\ ca,\ \exists\ tot,\ freq)\ CustomerAccount(ca) \wedge hasTotalAmountPurchased(ca,\ tot) \wedge hasPurchaseFrequency(ca,\ freq) \equiv (\forall\ po_1,\ ...,\ po_n,\ x_1,\ ...,\ x_n)\ CustomerOrder(po_1) \wedge ... \wedge CustomerOrder(po_n) \wedge hasCustomerOrder(ca,\ po_1) \wedge hasBaseTotalAfterCoupons(po_1,\ x_1) \wedge ... \wedge hasCustomerOrder(ca,\ po_n) \wedge hasBaseTotalAfterCoupons(po_n,\ x_n) \wedge tot = x_1 + ... + x_n \wedge freq = n. \quad \text{(FOL c-4)}$$

$$(\forall\ ca,\ tot,\ freq)\ CustomerAccount(ca) \wedge hasTotalAmountPurchased(ca, tot) \wedge hasPurchaseFrequency(ca,\ freq) \supset hasAverageOrderValue(ca,\ DIV(tot,\ freq)). \quad \text{(FOL c-5)}$$

A customer's time of last purchase is equal to the time of the last completed order issued to that customer:

$$(\forall\ ca,\ po,\ \exists\ po\_dt)\ CustomerAccount(ca) \wedge CustomerOrder(po) \wedge$$
$$hasCustomerOrder(ca,\ po) \wedge hasStatus(po,\ `completed') \wedge$$
$$hasCreateDatetime(po,\ po\_dt) \wedge \{(\forall\ po',\ po\_dt')CustomerOrder(po') \wedge$$
$$po \neq po' \wedge hasCustomerOrder(ca,\ po') \wedge\ hasCreateDatetime(po',\ po\_dt') \supset$$
$$before(po\_dt',\ po\_dt)\} \supset hasTimeOfLastPurchase(ca,\ po\_dt). \qquad \text{(FOL c-6)}$$

5.4 Concepts for Customer Relationship Management

5.4.1 Customer Value

Customer value, also referred to as customer profitability, is the difference between the revenues earned from and the costs associated with the customer relationship during a specified period [Pfeifer et al., 2005]. If individual customer profitability and the drivers of customer profitability are well understood by the retailer, then it is possible to take a variety of actions to transform unprofitable relationships into profitable ones.

In the Retail Ontology, customer profitability is represented by the concept *CustomerValue* which is a subclass of *Complex Financial Concept*, and is assigned to every customer:

$$(\forall\ ca,\ c\_value)\ CustomerAccount(ca) \wedge CustomerValue(c\_value) \supset$$
$$(valueOf(c\_value,\ ca) \equiv hasCustomerValue(ca,\ c\_value)). \qquad \text{(FOL c-7)}$$

CustomerValue has the following attributes:

- $hasStartDatetime(c\_value,\ s\_dt)$,
- $hasEndDatetime(c\_value,\ e\_dt)$,

- $hasAmountPurchased(c\_value,\ cv\_pur)$: specifies the revenue earned from sales to one customer and is calculated as follows:

$$
\begin{aligned}
&(\forall\ ca,\ c\_value,\ s\_dt,\ e\_dt,\ \exists\ tot)\ CustomerAccount(ca) \wedge \\
&\quad CustomerValue(c\_value) \wedge valueOf(c\_value,\ ca) \wedge \\
&\quad hasStartDatetime(c\_value,\ s\_dt) \wedge hasEndDatetime(c\_value,\ e\_dt) \wedge \\
&\quad hasAmountPurchased(c\_value,\ tot) \equiv \\
&\quad (\forall\ po_1,\ ...,\ po_n,\ x_1,\ ...,\ x_n,\ p\_dt_1,\ ...,\ p\_dt_n)\ CustomerOrder(po_1) \wedge \\
&\quad hasCustomerOrder(ca,\ po_1) \wedge hasStatus(po_1,\ `completed') \wedge \\
&\quad hasBaseTotalAfterCoupons(po_1,\ x_1) \wedge hasCreateDatetime(po_1,\ p\_dt_1) \wedge \\
&\quad before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge ... \wedge CustomerOrder(po_n) \wedge \\
&\quad hasCustomerOrder(ca,\ po_n) \wedge hasStatus(po_n,\ `completed') \wedge \\
&\quad hasBaseTotalAfterCoupons(po_n,\ x_n) \wedge hasCreateDatetime(po_n,\ p\_dt_n) \wedge \\
&\quad before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge tot = x_1 + ... + x_n. \qquad \text{(FOL c-8)}
\end{aligned}
$$

- $hasCost(c\_value,\ cv\_c)$: refers to the cost of acquisition and retention associated with one customer. This cost is defined later when acquisition and retention marketing strategies are discussed.

- $hasValue(c\_value,\ c)$: equals to the difference between the revenues earned from and the costs associated with the customer:

$$
\begin{aligned}
&(\forall\ c\_value,\ cv\_pur,\ cv\_c)\ CustomerValue(c\_value) \wedge \\
&\quad hasAmountPurchased(c\_value,\ cv\_pur) \wedge hasCost(c\_value,\ cv\_c) \supset \\
&\quad hasValue(c\_value,\ SUB(cv\_pur,\ cv\_c)). \qquad \text{(FOL c-9)}
\end{aligned}
$$

5.4.2 Conversion Rate

Conversion rate is the ratio of the number of visitors who place an order to the total number of visitors in a specified period of time. A high conversion rate depends on

several factors, namely, the interest level of the visitor, the attractiveness of the offer, and the ease of the process, all of which must be satisfactory to yield the desired results. Calculating conversion rate can especially be useful when measuring the effectiveness of a promotion by comparing the conversion rate before and after running the promotion.

ConversionRate (con_rate) is a concept in the ontology and has the following properties:

- $hasStartDatetime(con\_rate,\ s\_dt)$,
- $hasEndDatetime(con\_rate,\ e\_dt)$,
- $hasNumberOfVisitors(con\_rate,\ num\_vis)$,

$$
\begin{aligned}
&(\forall\ con\_rate,\ s\_dt,\ e\_dt,\ \exists\ num)\ ConversionRate(con\_rate) \wedge \\
&\quad hasStartDatetime(con\_rate,\ s\_dt) \wedge hasEndDatetime(con\_rate,\ e\_dt) \wedge \\
&\quad hasNumberOfVisitors(con\_rate,\ num\_vis) \equiv \\
&\quad (\forall\ s_1,\ ...,\ s_n,\ c\_dt_1,\ ...,\ c\_dt_n)\ Session(s_1) \wedge hasEntryTime(s_1,\ c\_dt_1) \wedge \\
&\quad ... \wedge Session(s_n) \wedge hasEntryTime(s_n,\ c\_dt_n) \wedge before(s\_dt,\ c\_dt_1) \wedge \\
&\quad before(c\_dt_1,\ e\_dt) \wedge ... \wedge before(s\_dt,\ c\_dt_n) \wedge before(c\_dt_n,\ e\_dt) \wedge \\
&\quad num\_vis = n. \qquad \text{(FOL c-10)}
\end{aligned}
$$

- $hasNumberOfBuyers(con\_rate,\ num\_po)$

$$
\begin{aligned}
&(\forall\ con\_rate,\ s\_dt,\ e\_dt,\ \exists\ num\_po)\ ConversionRate(con\_rate) \wedge \\
&\quad hasStartDatetime(con\_rate,\ s\_dt) \wedge hasEndDatetime(con\_rate,\ e\_dt) \wedge \\
&\quad hasNumberOfBuyers(con\_rate,\ num\_po) \equiv \\
&\quad (\forall\ pview_1,\ ...,\ pview_n,\ s_1,\ ...,\ s_n,\ v\_dt_1,\ ...,\ v\_dt_n)\ PageView(pview_1) \wedge \\
&\quad Session(s_1) \wedge hasSession(pview_1,\ s_1) \wedge purchasedIn(s_1,\ true) \wedge \\
&\quad hasHitDatetime(pview_1,\ v\_dt_1) \wedge ... \wedge Session(s_n) \wedge \\
&\quad hasSession(pview_n,\ s_n) \wedge purchasedIn(s_n,\ true) \wedge
\end{aligned}
$$

$$hasHitDatetime(pview_n,\ v\_dt_n) \wedge before(s\_dt,\ v\_dt_1) \wedge before(v\_dt_1, e\_dt)$$
$$\wedge ... \wedge before(s\_dt,\ v\_dt_n) \wedge before(v\_dt_n,\ e\_dt) \wedge num\_po = n. \quad \text{(FOL c-11)}$$

The value of a conversion is defined by:

$$(\forall\ con\_rate,\ num\_vis, num\_cus)\ ConversionRate(con\_rate) \wedge$$
$$hasNumberOfVisitors(con\_rate,\ num\_vis) \wedge$$
$$hasNumberOfBuyers(con\_rate,\ num\_cus) \supset$$
$$hasValue(con\_rate,\ DIV(num\_cus,\ num\_vis)). \quad \text{(FOL c-12)}$$

5.4.3 Customer Lifetime Value

Customer lifetime value (CLV) is the present value of the future cash flows associated with a customer [Pfeifer et al., 2005]. CLV was discussed in detail in Chapter 2. In the Retail Ontology, CLV is a concept that can be assigned to an individual customer.

$$(\forall\ ca,\ clv)\ CustomerAccount(ca) \wedge CustomerLifetimeValue(clv) \supset$$
$$(lifetimeValueOf(clv,\ ca) \equiv hasCLV(ca,\ clv)). \quad \text{(FOL c-13)}$$

CLV has the following attributes:

- $hasStartDatetime(clv,\ s\_dt)$,
- $hasEndDatetime(clv,\ e\_dt)$,
- $hasRecency(clv,\ rec)$,

$$(\forall\ clv,\ \exists\ rec)\ CustomerLifetimeValue(clv) \wedge hasRecency(clv,\ rec) \equiv$$
$$(\forall\ ca,\ r)\ CustomerAccount(ca) \wedge lifetimeValueOf(clv,\ ca) \wedge$$
$$hasTimeOfLastPurchase(ca,\ r) \wedge rec = r. \quad \text{(FOL c-14)}$$

- $hasFrequency(clv,\ freq)$,

$$(\forall\ clv,\ \exists\ freq)\ CustomerLifetimeValue(clv) \wedge hasFrequency(clv,\ freq) \equiv$$
$$(\forall\ ca,\ f)\ CustomerAccount(ca) \wedge lifetimeValueOf(clv,\ ca) \wedge$$
$$hasPurchaseFrequency(ca,\ f) \wedge freq = f. \qquad \text{(FOL c-15)}$$

- $hasMonetaryValue(clv,\ mon)$,

$$(\forall\ clv,\ \exists\ mon)\ CustomerLifetimeValue(clv) \wedge$$
$$hasMonetaryValue(clv,\ mon) \equiv (\forall\ ca,\ aov)\ CustomerAccount(ca) \wedge$$
$$lifetimeValueOf(clv,\ ca) \wedge hasAverageOrderValue(ca,\ aov) \wedge$$
$$mon = aov. \qquad \text{(FOL c-16)}$$

- $hasExpectedTransactionValue(clv,\ etv)$, etv is calculated using formula 2.4 given in Chapter 2.

- $hasValue(clv,\ clv\_value)$, this value is calculated using formula 2.1 given in Chapter 2.

5.5 Segment

Segmentation is an essential strategy in customer oriented marketing. The concept of market segmentation was first introduced by Smith in 1956, and it involves partitioning a heterogeneous market into a number of smaller, separate and distinct homogeneous markets [Wedel and Kamakura, 2000]. A good segmentation model would help managers in understanding the status of current and potential customers, and enable them to reach the market effectively and gain appropriate customer response [Lemon and Mark, 2006].

Customer segmentation is a managerially imposed market structure and not a natural phenomenon − it can be used to improve the effectiveness and efficiency of marketing strategies such as product development and pricing and tactical decisions like customer acquisition and retention [Lemon and Mark, 2006]. In this regard, two types of segmentation techniques can be defined: descriptive segmentation and predictive segmentation

[Olszak and Ziemba, 2006]. Descriptive segmentation techniques are useful when a set of variables can be identified in advance based on their importance for making marketing decisions. Segementation can then be achieved by grouping similar individuals together based upon the values of the variables selected. Descriptive segmentation can be divided into [Olszak and Ziemba, 2006]:

- demographic segmentation on the basis of data including customer sex, education, age, etc.;
- behavioural segmentation based on variables that represent measurable behavior of individuals such as transaction information on frequency, quantity and type of products purchased in a retail store, etc.; and
- motivational segmentation based on variables that relate to why customers make purchases.

Predictive segmentation methodologies, on the other hand, are useful for understanding what variables distinguish best customers. Predictive segmentation starts with the choice of one variable usually related to profitability that is key indicator of good customers, after which other variables that greatly influence the initial variable are determined. In addition to identifying the most profitable customers, this technique also allows the analyses of customers' migration between segments.

Definition: Segment

A customer segment is one of the groups into which customers are divided. A segment is a concept in the ontology. S is a segment if the following is asserted in the knowledgebase:

- $Segment(S)$.

There are many different ways in which customers can be grouped. In what follows, some criteria for segmenting customers are discussed. Of course, it is possible to combine these criteria to achieve a richer segmentation.

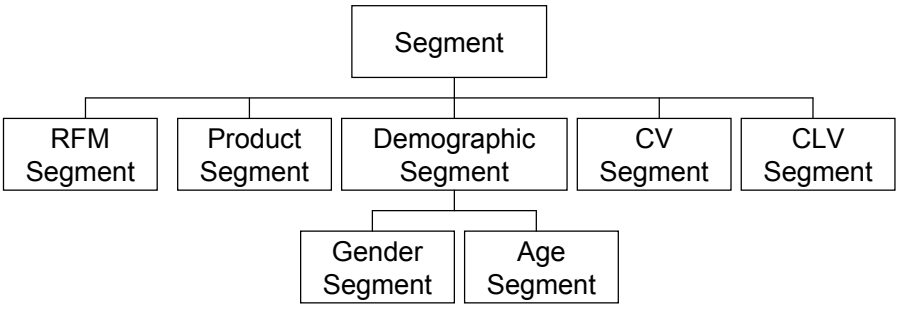

Figure 5.1: Segment taxonomy (sub-class relationship)

5.5.1 Customer Segmentation Based on Transactional Data

Definition: RFMSegment

In the direct marketing literature customers' prior behavior is usually summarized in terms of their recency (time of most recent purchase), frequency (number of prior purchases), and monetary value (average purchase amount per transaction), commonly known by the acronym RFM. The RFM characteristics can be used as the basis for customer segmentation. Since recency and monetary value are continuous variables, RFM segments are defined by specifying the minimum and maximum values each characteristic can have.

A RFM segment is represented by the concept $RFMSegment$ which is a subclass of $Segment$, and has the following properties:

- $RFMSegment(rfm\_seg)$,
- $hasRecencyMin(rfm\_seg, rmin\_value)$,
- $hasRecencyMax(rfm\_seg, rmax\_value)$,
- $hasFrequency(rfm\_seg, freq)$,
- $hasMonetaryValueMin(rfm\_seg, mmin\_value)$,
- $hasMonetaryValueMax(rfm\_seg, mmax\_value)$.

A customer belongs to a RFM segment if her/his time of last purchase, purchase frequency, and average order value fall into the ranges that define the segment:

$$(\forall\ ca,\ t,\ aov,\ freq,\ rfm\_seg,\ rmin\_value,\ rmax\_value,\ f,\ mmin\_value,\ mmax\_value)\ CustomerAccount(ca) \wedge hasTimeOfLastPurchase(ca,\ t) \wedge hasAverageOrderValue(ca,\ aov) \wedge hasPurchaseFrequency(ca,\ freq) \wedge RFMSegment(rfm\_seg) \wedge hasRecencyMin(rfm\_seg,\ rmin\_value) \wedge hasRecencyMax(rfm\_seg,\ rmax\_value) \wedge hasFrequency(rfm\_seg,\ f) \wedge hasMMin(rfm\_seg,\ mmin\_value) \wedge hasMMax(rfm\_seg,\ mmax\_value) \wedge t \geq rmin\_value \wedge t \leq rmax\_value \wedge aov \geq mmin\_value \wedge aov \leq mmax\_value \wedge freq = f \supset belongsToSegment(ca,\ rfm\_seg).$$

(FOL c-17)

Definition: CVSegment

Customer value can also be used as the basis for customer segmentation. This segmentation results in the identification of the most profitable customers. A customer value segment is represented by the concept $CVSegment$ and is a subclass of $Segment$ and has the following properties:

- $CVSegment(cv\_seg)$,
- $hasCVMin(cv\_seg, min\_value)$,
- $hasCVMax(cv\_seg, max\_value)$.

In order for a customer to belong to a CV segment, the customer value associated with that person must be in the range [min_value, max_value]:

$$(\forall\ ca,\ cv,\ cv\_amount,\ cv\_seg, min\_value,\ max\_value)\ CustomerAccount(ca) \wedge CustomerValue(cv) \wedge hasCustomerValue(ca,\ cv) \wedge hasValue(cv,\ cv\_value) \wedge CVSegment(cv\_seg) \wedge hasCVMin(cv\_seg, min\_value) \wedge hasCVMax(cv\_seg, max\_value) \wedge$$

$$cv\_value \geq min\_value \wedge cv\_value < max\_value \supset$$
$$belongsToSegment(ca, cv\_seg). \quad \text{(FOL c-18)}$$

Assuming that the segments don't overlap, the most profitable segment can be defined as the segment which has a minimum value greater than the maximum value of all the other segments that include at least one customer:

$$\begin{aligned}
&(\forall\ cv_seg_1,\ ...,\ cv_seg_n,\ min_value_1,\ max_value_2,\ ..., max_value_n)\\
&\quad MostProfitableSegment(cv_seg_1) \equiv CVSegment(cv_seg_1) \wedge\\
&\quad hasCVMin(cv_seg_1,\ min_value_1) \wedge hasCVMax(cv_seg_1,\ max_value_1) \wedge ... \wedge\\
&\quad CVSegment(cv_seg_n) \wedge hasCVMin(cv_seg_n,\ min_value_n) \wedge\\
&\quad hasCVMax(cv_seg_n,\ max_value_n) \wedge\\
&\quad \{(\exists\ ca)\ CustomerAccount(ca) \wedge belongsToSegment(ca,\ cv_seg_1)\} \wedge\\
&\quad \{(min_value_1 \geq max_value_2) \vee (min_value_1 < max_value_2 \wedge\\
&\qquad (\neg\exists\ ca')\ CustomerAccount(ca') \wedge belongsToSegment(ca', cv_seg_2)\} \wedge ... \wedge\\
&\quad \{(min_value_1 \geq max_value_n) \vee (min_value_1 < max_value_n \wedge\\
&\qquad (\neg\exists\ ca')\ CustomerAccount(ca') \wedge belongsToSegment(ca', cv_seg_n)\}
\end{aligned}$$
(FOL c-19)

5.5.2 Customer Segmentation Based on Customer Lifetime Value

Definition: CLVSegment

CLV-based segmentation is an approach in which customer lifetime value is the basis of customer segmentation. Each CLV segment is specified by a minimum and a maximum value. Clearly that this approach to segmentation and the RFM-based segmentation described in the previous section are in conflict since the latter rests on the assumption that profitability derives from large transactions that are recent and/or frequent while the goal of the former is to create a relationship with the customer and maintain it over time. These competing strategies simply reflect the reality that different businesses view

the consumer in different ways.

A CLV segment is represented by the concept $CLVSegment$ which is a subclass of $Segment$, and has the following two properties:

- $CLVSegment(clv\_seg)$,
- $hasCLVMin(clv\_seg, min\_value)$,
- $hasCLVMax(clv\_seg, max\_value)$.

In order for a customer to belong to a CLV segment, her/his lifetime value must be in the range [min_value, max_value]:

$$
\begin{aligned}
&(\forall\ ca,\ clv,\ clv\_amount,\ clv\_seg, min\_value,\ max\_value) \\
&\quad CustomerAccount(ca) \wedge CustomerLifetimeValue(clv) \wedge hasCLV(ca, clv) \wedge \\
&\quad hasValue(clv,\ clv\_value) \wedge CLVSegment(clv\_seg) \wedge \\
&\quad hasCLVMin(clv\_seg, min\_value) \wedge hasCLVMax(clv\_seg, max\_value) \wedge \\
&\quad clv\_value \geq min\_value \wedge clv\_value < max\_value \supset \\
&\quad belongsToSegment(ca, clv\_seg). \qquad \text{(FOL c-20)}
\end{aligned}
$$

5.5.3 Customer Segmentation Based on Product Data

Definition: ProductSegment

Customer segmentation based on product data is an approach in which customers are grouped according to the product categories they purchased. A product segment is represented by the concept $ProductSegment$ which is a subclass of $Segment$, and has the following property:

- $ProductSegment(pro\_seg)$,
- $hasProductCategory(pro\_seg,\ pcat)$, where $pcat$ is a ProductCategory ID.

In order for a customer to belong to a product segment, s/he must have purchased a product belonging to the category associated with that segment:

$$(\forall\ ca,\ po,\ p,\ pro\_seg,\ pcat)\ CustomerAccount(ca) \wedge CustomerOrder(po) \wedge$$
$$hasCustomerOrder(ca,\ po) \wedge Product(p) \wedge hasProduct(po,\ p) \wedge$$
$$ProductCategory(pcat) \wedge productOf(p,\ pcat) \wedge ProductSegment(pro\_seg) \wedge$$
$$hasProductCategory(pro\_seg,\ pcat) \supset belongsToSegment(ca,\ pro\_seg).$$

(FOL c-21)

5.5.4 Customer Segmentation Based on Demographic Data

Definition: DemographicSegment

Demographic segmentation is an approach to segmentation on the basis of the data including customer's age, sex, marital status, education, etc., and is considered to be a descriptive segmentation technique. A demographic segment is represented by the concept *DemographicSegment* and is a subclass of *Segment*.

Example:

Examples of demographic segments include *GenderSegment* and *AgeSegment*. The only attribute of a *GenderSegment* is $hasGender(gen\_seg,\ g)$, and an *AgeSegment* has the following properties:

- $hasAMin(age\_seg,\ min\_value)$,
- $hasAMax(age\_seg,\ max\_value)$.

It might also be useful to segment customers according to where they live. Using customers' postal codes, for instance, one can define *GeographicSegment* to group customers.

- $GeographicSegment(geo\_seg)$,
- $hasGSPostalCode(geo\_seg,\ p\_code)$.

Thus, a customer belongs to a geographic segment if s/he lives within the range associated with that segment:

$$(\forall\ ca,\ per,\ adr,\ geo\_seg,\ p\_code)\ CustomerAccount(ca) \wedge Person(per) \wedge$$
$$hasPerson(ca,\ per) \wedge Address(adr) \wedge hasAddress(per,\ adr) \wedge$$
$$hasPostalCode(adr,\ p\_code) \wedge GeographicSegment(geo\_seg) \wedge$$
$$hasGSPostalCode(geo\_seg, p\_code) \supset belongsToSegment(ca, geo\_seg).$$

(FOL c-22)

It is also possible to calculate the revenue that customers living in different geographic zones are generating in a period of time, in order to compare buying trends across geographies. To do this *hasGSRevenue(geo_seg, rev)* is defined as:

$$(\forall\ geo\_seg,\ \exists rev,\ s\_dt,\ e\_dt,\ tot)\ GeographicSegment(geo\_seg) \wedge Revenue(rev) \wedge$$
$$hasStartDatetime(rev,\ s\_dt) \wedge hasEndDatetime(rev,\ e\_dt) \wedge$$
$$hasAmount(rev,\ tot) \wedge hasGSRevenue(geo\_seg,\ rev) \equiv$$
$$(\forall\ ca_1,\ ...,\ ca_m,\ po_1,\ ...,\ po_n,\ x_1,\ ...,\ x_n,\ p\_dt_1,\ ...,\ p\_dt_n)$$
$$CustomerAccount(ca_1) \wedge belongsToSegment(ca_1, geo\_seg) \wedge ... \wedge$$
$$CustomerAccount(ca_m) \wedge belongsToSegment(ca_m, geo\_seg) \wedge$$
$$CustomerOrder(po_1) \wedge hasStatus(po_1,\ `completed') \wedge$$
$$hasBaseTotalAfterCoupons(po_1,\ x_1) \wedge hasCreateDatetime(po_1,\ p\_dt_1) \wedge$$
$$before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge \{hasCustomerOrder(ca_1,\ po_1) \vee ... \vee$$
$$hasCustomerOrder(ca_m,\ po_1)\} \wedge ... \wedge CustomerOrder(po_n) \wedge$$
$$hasStatus(po_n,\ `completed') \wedge hasBaseTotalAfterCoupons(po_n,\ x_n) \wedge$$
$$hasCreateDatetime(po_n,\ p\_dt_n) \wedge before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge$$
$$\{hasCustomerOrder(ca_1,\ po_n) \vee ... \vee hasCustomerOrder(ca_m,\ po_n)\} \wedge$$
$$tot = x_1 + ... + x_n.$$

(FOL c-23)

5.6 Competency Questions Revisited

In this section we repeat the competency questions and follow each with the query that would provide the answer. Note that parameters that are preceded with a "?" are variables. If a query is not existentially quantified, then it is a query for free variable bindings.

- What is the average order value of each customer?
  - **Query 1:**

    $CustomerAccount(?ca) \wedge hasAverageOrderValue(?ca,\ ?x)$.

- What is the purchase behaviour of a customer?
  - A Customer's prior behaviour is described in terms of her/his RFM characteristics.
  - **Query 2:**

    $CustomerAccount(\mathbf{ca}) \wedge hasTimeOfLastPurchase(\mathbf{ca},\ ?pd) \wedge$
    $hasPurchaseFrequency(\mathbf{ca},\ ?f) \wedge hasAverageOrderValue(\mathbf{ca},\ ?aov)$.

- Who are the customers?

  a) All segmentation techniques described in this chapter can be used to identify customers. In what follows two examples are given.

  b) Which customers are buying products from a specified category *cat*?
  - **Query 3:**

    $ProductSegment(?s) \wedge ProductCategory(\mathbf{cat}) \wedge$
    $hasProductCategory(?s,\ \mathbf{cat}) \wedge CustomerAccount(?ca) \wedge$
    $belongsToSegment(?ca,\ ?s)$.

  c) How can customers be characterized based on demographic variables?

- Demographic variables include sex, age, education, marital status, etc. *DemographicSegment* can be used to group customers according to the value of demographic variables. As an example, customer segmentation based on gender is given below.

- **Query 4:**

$$GenderSegment(?s) \wedge hasGender(?s,\ \mathbf{x}) \wedge$$
$$CustomerAccount(?ca) \wedge belongsToSegment(?ca,\ ?s).$$

- Who are the most profitable customers?

  - The most profitable customers are those who belong to the MostProfitable-Segment.

  - **Query 5:**

$$MostProfitableSegment(?s) \wedge CustomerAccount(?ca) \wedge$$
$$belongsToSegment(?ca,\ ?s).$$

- How do buying trends compare across geographies?

  - By knowing the answer to how much revenue is generated over a time period specified by the starting time point s and the ending time point e in a zone identified by *p_code*, it is possible to compare buying trends across geographies during different time periods.

  - **Query 6:**

$$GeographicSegment(?s) \wedge hasGSPostalCode(?s,\ \mathbf{p\_code}) \wedge$$
$$Revenue(?rev) \wedge hasGSRevenue(?s,\ ?rev) \wedge$$
$$hasAmount(?rev,\ ?amount) \wedge hasStartDatetime(?cr,\ \mathbf{s}) \wedge$$
$$hasEndDatetime(?cr,\ \mathbf{t}).$$

- Which items have been purchased by customers who bought SKU x?

- **Query 7:**

$$SKU(\mathbf{x}) \wedge SKU(?y) \wedge purchasedBySameCustomer(\mathbf{x},\ ?y).$$

- What is the conversion rate in a time period specified by the starting time point s and the ending time point e (for a webstore)?

  - **Query 8:**

$$ConversionRate(?cr) \wedge hasValue(?cr,\ ?val) \wedge$$
$$hasStartDatetime(?cr,\ \mathbf{s}) \wedge hasEndDatetime(?cr,\ \mathbf{t}).$$

- What is the expected average transaction value associated with a customer?

  - Assume g, p, and q are the value of parameters estimated for calculating the average expected transaction value for a customer using formula 2.4.

  - **Query 9:**

$$CustomerAccount(\mathbf{ca}) \wedge CustomerLifetimeValue(?clv) \wedge$$
$$hasCLV(\mathbf{ca},\ ?clv) \wedge hasExpectedTransactionValue(?clv,\ ?etv).$$

5.7 Summary

The entities, attributes and relations needed for effective customer relationship management were defined in this chapter. Also, a set of competency questions was presented to demonstrate the use of the terminology in retail environments. Other customer related competency questions that are important but not addressed in this research include:

- How many customers have previously visited an item page and bought the item at a later date in response to a special offer? (Impact of sale pricing on products)

- Which customer segments respond best to which marketing promotions?

- Which customers seem most amenable to product substitution? Which customers are more likely to impulse-buy?

Chapter 6

Marketing

6.1 Introduction

The main focus in retailing is to improve commercial performance by increasing organization's turnover and maximizing profitability. Achieving performance requires actions in one of the main area of store operations, human resources, finance, purchasing, customer care, marketing, logistics, information technology, administration and management.

Marketing consists of individual and organizational activities that facilitate exchange relationships in a dynamic environment through the creation, distribution, promotion and pricing of goods, services and ideas [Dibb et al., 2001]. Most businesses depend on marketing to provide an understanding of the marketplace in order to ensure their products and services satisfy the needs of customers and that they are competing effectively [Dibb et al., 2001]. The following are some competency questions that are important for effective marketing.

- How much is the total revenue earned in a given period?

- What is the retention rate over a period specified by the starting time point s and ending time point t?

- What are the most profitable products in a specified time period?
- Which items are likely to be purchased by customers who bought SKU x?
- Which customers add a cross-sell product to their order?
- How much is the acquisition profit (loss) in a specified period of time?
- How much is the total retention cost over a period specified by the starting time point s and ending time point t?
- How effective was a promotion?

The organization of this chapter is as follows. In Section 7.2 the basic terminology of promotion and coupon are presented. Section 7.3 is concerned with the *Complex Financial Concept* module of the Retail Ontology. Section 7.4 describes some marketing actions and Section 7.5 provides answers to the competency questions. Finally, Section 7.6 concludes this chapter with a summary.

6.2 Promotion

Promotion refers to the various methods of promoting a product, brand, or company. There are a number of promotional objectives, some of the most common being information dissemination, product demand, product differentiation, product highlights, and sales stabilization [McClintic and Gale, 2001]. Regardless of the promotional objective selected, the company's goal is to inform and convince consumers to buy the product [McClintic and Gale, 2001]. The most common components of the promotional mix are advertising, sales promotion, publicity, and personal selling [Karjaluoto et al., 2004]. Each element in the promotional mix has different capacities to communicate and to acheive different objectives [Karjaluoto et al., 2004].

- $Promotion(prom)$.

- $hasGoal(prom,\ obj)$, where obj is a Goal ID.
- $hasActivity(prom,\ a)$, where a is an Activity ID.
- $hasStartDatetime(prom,\ p\_sdt)$,
- $hasEndDatetime(prom,\ p\_edt)$,

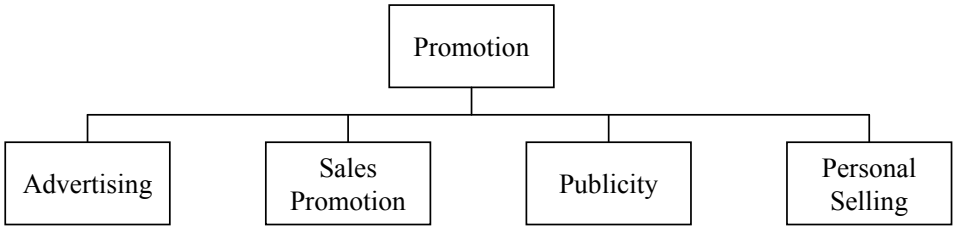

Figure 6.1: Promotional mix (sub-class relationships)

6.2.1 Coupon

A coupon is a ticket or document that can be exchanged for a financial discount or rebate when purchasing a product. Coupons are issued by retailers to be used in retail stores as a part of sales promotions. They are often widely distributed through mail, magazines, newspapers and the Internet.

- $Coupon(cp)$,
- $hasCurrency(cp,\ cp\_cur)$,
- $hasStartDatetime(cp,\ cp\_sdt)$,
- $hasEndDatetime(cp,\ cp\_edt)$,
- $hasValueAssigned(cp,\ val)$,, where val is a decimal value,
- $hasMaximumValue(cp,\ max\_val)$, where max_val is a decimal value,

- $hasMaximumUsage(cp,\ max\_use)$, where max_use is an integer value,
- $timesUsed(cp,\ use)$, where use is an integer value,
- $couponOf(cp,\ w\_store)$.

The total value of a customer order after applying coupons can be calculated as follows:

$$
\begin{aligned}
&(\forall\ co,\ p\_dt,\ b\_tot\ \exists\ tot)\ CustomerOrder(co) \wedge hasCreateDatetime(co,\ p\_dt) \wedge \\
&\quad hasBaseTotal(co,\ b\_tot) \wedge hasTotalAfterCoupon(co,\ tot) \equiv \\
&\quad (\forall\ cp_1,\ ...,\ cp_n,\ val_1,\ ...,\ val_n,\ cp\_sdt_1,\ ...,\ cp\_sdt_n,\ cp\_edt_1,\ ...,\ cp\_edt_n) \\
&\quad Coupon(cp_1) \wedge hasCoupon(co,\ cp_1) \wedge hasStartDatetime(cp_1,\ cp\_sdt_1) \wedge \\
&\quad hasEndDatetime(cp_1,\ cp\_edt_1) \wedge before(cp\_sdt_1,\ p\_dt) \wedge before(p\_dt,\ cp\_edt_1) \wedge \\
&\quad hasValueAssigned(cp_1,\ val_1) \wedge timesUsed(cp_1,\ use_1) \wedge \\
&\quad hasMaximumUsage(cp_1,\ max\_use_1) \wedge use_1 \leq max\_use_1 \wedge ... \wedge Coupon(cp_n) \wedge \\
&\quad hasCoupon(co,\ cp_n) \wedge hasStartDatetime(cp_n,\ cp\_sdt_n) \wedge \\
&\quad hasEndDatetime(cp_n,\ cp\_edt_n) \wedge before(cp\_sdt_n,\ p\_dt) \wedge before(p\_dt,\ cp\_edt_n) \wedge \\
&\quad hasValueAssigned(cp_n,\ val_n) \wedge timesUsed(cp_n,\ use_n) \wedge \\
&\quad hasMaximumUsage(cp_n,\ max\_use_n) \wedge use_n \leq max\_use_n \wedge \\
&\quad tot = b\_tot - val_1 - ... - val_n. \qquad \text{(FOL d-1)}
\end{aligned}
$$

6.3 Complex Financial Concepts

6.3.1 Revenue

Revenue is a term for the amount of money that a company receives from sales of products to customers in a given period. In this ontology, revenue is considered to be equal to price times quantity. It is worth emphasizing that revenue can only be presented in terms of a period and would be meaningless otherwise.

- $Revenue(rev)$,

- $hasStartDatetime(rev,\ s\_dt)$,
- $hasEndDatetime(rev,\ e\_dt)$,
- $hasAmount(rev,\ rev\_amount)$.

The revenue earned from sales (*SalesRevenue*) is calculated as follows:

$$
\begin{aligned}
&(\forall\ rev,\ s\_dt,\ e\_dt,\ \exists\ amount)\ SalesRevenue(rev) \wedge \\
&\quad hasStartDatetime(rev,\ s\_dt) \wedge hasEndDatetime(rev,\ e\_dt) \wedge \\
&\quad hasAmount(rev,\ amount) \equiv (\forall\ co_1,\ ...,\ co_n,\ tot_1,\ ...,\ tot_n,\ p\_dt_1,\ ...,\ p\_dt_n) \\
&\quad CustomerOrder(co_1) \wedge hasStatus(co_1,\ \text{`}completed\text{'}) \wedge \\
&\quad hasTotalAfterCoupons(co_1,\ tot_1) \wedge hasCreateDatetime(co_1,\ p\_dt_1) \wedge \\
&\quad before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge ... \wedge CustomerOrder(co_n) \wedge \\
&\quad hasStatus(co_n,\ \text{`}completed\text{'}) \wedge hasTotalAfterCoupons(co_n,\ tot_n) \wedge \\
&\quad hasCreateDatetime(co_n,\ p\_dt_n) \wedge before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge \\
&\quad amount = tot_1 + ... + tot_n. \qquad \text{(FOL d-2)}
\end{aligned}
$$

The revenue earned from sales to first-time customers is called the acquisition revenue. *AcquisitionRevenue* is a subclass of *Revenue* with the following additional attribute:

- $hasNumberOfCustomers(a\_rev,\ a\_num)$.

The amount of acquisition revenue can be calculated as follows:

$$
\begin{aligned}
&(\forall\ rev,\ s\_dt,\ e\_dt,\ \exists\ amount,\ a\_num)\ AcquisitionRevenue(rev) \wedge \\
&\quad hasStartDatetime(rev,\ s\_dt) \wedge hasEndDatetime(rev,\ e\_dt) \wedge \\
&\quad hasAmount(rev,\ amount) \wedge hasNumberOfCustomers(rev,\ a\_num) \equiv \\
&\quad (\forall\ ca_1,\ ..., ca_n,\ co_1,\ ...,\ co_n,\ tot_1,\ ...,\ tot_n,\ p\_dt_1,\ ...,\ p\_dt_n) \\
&\quad CustomerAccount(ca_1) \wedge hasPurchaseFrequency(ca_1,\ 1) \wedge \\
&\quad CustomerOrder(co_1) \wedge hasStatus(co_1,\ \text{`}completed\text{'}) \wedge \\
&\quad hasCustomerOrder(ca_1,\ co_1) \wedge hasTotalAfterCoupons(co_1,\ tot_1) \wedge
\end{aligned}
$$

$$hasCreateDatetime(co_1,\ p\_dt_1) \wedge before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge$$
$$... \wedge CustomerAccount(ca_n) \wedge hasPurchaseFrequency(ca_n,\ 1) \wedge$$
$$CustomerOrder(co_n) \wedge hasStatus(co_n,\ `completed') \wedge$$
$$hasCustomerOrder(ca_n,\ co_n) \wedge hasTotalAfterCoupons(co_n,\ tot_n) \wedge$$
$$hasCreateDatetime(co_n,\ p\_dt_n) \wedge before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge$$
$$amount = tot_1 + ... + tot_n \wedge a\_num = n. \quad \text{(FOL d-3)}$$

6.3.2 Gross Profit

Gross profit is the value obtained from subtracting all costs directly related to sales of products (cost of goods sold) from revenue generated by those sales.

- $GrossProfit(gp)$,
- $hasRevenue(gp,\ rev)$, where rev is a Revenue ID,
- $hasCost(gp,\ gp\_cost)$,
- $hasAmount(gp,\ gp\_amount)$.

The value of costs directly related to sales of products is calculated as follows:

$$(\forall\ gp,\ rev,\ s\_dt,\ e\_dt,\ \exists\ cost)\ GrossProfit(gp) \wedge hasCost(gp,\ cost) \wedge$$
$$Revenue(rev) \wedge hasRevenue(gp,\ rev) \wedge hasStartDatetime(rev,\ s\_dt) \wedge$$
$$hasEndDatetime(rev,\ e\_dt) \equiv (\forall\ po_1,\ ...,\ po_n,\ c_1,\ ...,\ c_n,\ p\_dt_1,\ ...,\ p\_dt_n)$$
$$PurchaseOrder(po_1) \wedge hasStatus(po_1,\ `completed') \wedge$$
$$hasCostOfGoodsSold(po_1,\ c_1) \wedge hasCreateDatetime(po_1,\ p\_dt_1) \wedge ... \wedge$$
$$PurchaseOrder(po_n) \wedge hasStatus(po_n,\ `completed') \wedge$$
$$hasCostOfGoodsSold(po_n,\ c_n) \wedge hasCreateDatetime(po_n,\ p\_dt_n) \wedge$$
$$before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge ... \wedge$$
$$before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge cost = c_1 + ... + c_n. \quad \text{(FOL d-4)}$$

Thus, the value of gross profit in a specified period is given by:

$$(\forall\ gp,\ rev,\ gp\_cost)\ GrossProfit(gp) \wedge Revenue(rev) \wedge hasRevenue(gp,\ rev) \wedge$$
$$hasAmount(rev,\ revenue) \wedge hasCost(gp,\ gp\_cost)$$
$$\supset hasAmount(gp,\ SUB(revenue,\ gp\_cost)) \qquad \text{(FOL d-5)}$$

6.3.3 Acquisition Profit

The acquisition profit can be defined as the product of the average order value of first time customers and the acquisition margin:

$$acquisition\ profit = \frac{acquisition\ revenue}{\#\ of\ first\ time\ customers} \times \frac{acquisition\ revenue - acquisition\ cost}{acquisition\ revenue}$$
$$= \frac{acquisition\ revenue - acquisition\ cost}{\#\ of\ first\ time\ customers}.$$

Acquisition profit is a concept in the ontology with the following attributes:

- $AcquisitionProfit(ap)$.
- $hasAcquisitionActivity(ap,\ act)$, where act is an AcquisitionActivity ID.
- $hasAcquisitionRevenue(ap,\ rev)$, where rev is an AcquisitionRevenue ID.
- $hasStartDatetime(ap,\ ap\_sdt)$,
- $hasEndDatetime(ap,\ ap\_edt)$,
- $hasAcquisitionCost(ap,\ ap\_cost)$. The acquisition cost can be defined using the Activity-Cost ontology as follows (*AcquisitionActivity* is discussed later in this chapter):

$$(\forall\ ap,\ \exists\ c)\ AcquisitionProfit(ap) \wedge hasAcquisitionCost(ap,\ c) \equiv$$
$$(\forall\ acq,\ acp)\ AcquisitionActivity(acq) \wedge hasAcquisitionActivity(ap,\ act) \wedge$$
$$ActivityCostPoint(acp) \wedge hasActivity(acp,\ acq) \wedge hasCost(acp,\ c).$$

(FOL d-6)

- $hasAmount(ap,\ ap\_amount)$. The value of acquisition profit is calculated using the formula given above:

$$(\forall\ ap,\ rev,\ cost,\ num,\ x)\ AcquisitionProfit(gp) \wedge$$
$$AcquisitionRevenue(rev) \wedge hasAcquisitionRevenue(ap,\ rev) \wedge$$
$$hasNumberOfCustomers(rev,\ num) \wedge hasAmount(rev,\ x) \wedge$$
$$hasAcquisitionCost(ap,\ cost) \supset$$
$$hasAmount(ap,\ DIV(SUB(x,\ cost),\ num)). \qquad \text{(FOL d-7)}$$

6.3.4 Retention Rate

The retention rate can be determined by tracking the number of newly acquired customers who return for purchase in a specified time period.

- $RetentionRate(ret\_rate)$,
- $hasStartDatetime(ret\_rate,\ s\_dt)$,
- $hasEndDatetime(ret\_rate,\ e\_dt)$,
- $hasNumberOfRetainedCustomers(ret\_rate,\ num\_ret\_cus)$. The number of retained customers is equal to the number of customers who have made their first purchase before s_dt and their next purchase during the specified time frame:

$$(\forall\ ret\_rate,\ s\_dt,\ e\_dt\ \exists\ num\_ret\_cus)\ RetentionRate(ret\_rate) \wedge$$
$$hasStartDatetime(ret\_rate,\ s\_dt) \wedge hasEndDatetime(ret\_rate,\ e\_dt) \wedge$$
$$hasNumberOfRetainedCustomers(ret\_rate,\ num\_ret\_cus) \equiv$$
$$(\forall\ ca_1,\ ...,\ ca_n,\ f_1,\ ...,\ f_n,\ \exists\ co_1,\ ....,\ co_n,\ p\_dt_1,\ ....,\ p\_dt_n)$$
$$CustomerAccount(ca_1) \wedge hasPurchaseFrequency(ca_1,\ f_1) \wedge f_1 > 1 \wedge$$

$$CustomerOrder(co_1) \wedge hasCreateDatetime(co_1,\ p\_dt_1) \wedge$$
$$hasCustomerOrder(ca_1,\ co_1) \wedge before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge$$
$$\{(\exists\ co'_1,\ p\_dt'_1)\ CustomerOrder(co'_1) \wedge hasCreateDatetime(co'_1,\ p\_dt'_1) \wedge$$
$$hasCustomerOrder(ca_1,\ co'_1) \wedge before(p\_dt'_1,\ s\_dt)\} \wedge ... \wedge$$
$$CustomerAccount(ca_n) \wedge hasPurchaseFrequency(ca_n,\ f_n) \wedge f_n > 1 \wedge$$
$$CustomerOrder(co_n) \wedge hasCreateDatetime(co_n,\ p\_dt_n) \wedge$$
$$hasCustomerOrder(ca_n,\ co_n) \wedge before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge$$
$$\{(\exists\ co'_n,\ p\_dt'_n)\ CustomerOrder(co'_n) \wedge hasCreateDatetime(co'_n,\ p\_dt'_n) \wedge$$
$$hasCustomerOrder(ca_n,\ co'_n) \wedge before(p\_dt'_n,\ s\_dt)\} \wedge num\_ret\_cus = n. \quad \text{(FOL d-8)}$$

- $hasNumberOfCustomers(ret\_rate,\ num\_cus)$. The number of customers in a specified time period is equal to the number of visitors who have made at least one purchase during the specified time period:

$$(\forall\ ret\_rate,\ s\_dt,\ e\_dt,\ \exists\ num)\ RetentionRate(ret\_rate) \wedge$$
$$hasStartDatetime(ret\_rate,\ s\_dt) \wedge hasEndDatetime(ret\_rate,\ e\_dt) \wedge$$
$$hasNumberOfCustomers(ret\_rate,\ num) \equiv$$
$$(\forall\ ca_1,\ ...,\ ca_n,\ \exists\ co_1,\ ....,\ co_n,\ p\_dt_1,\ ....,\ p\_dt_n)\ CustomerAccount(ca_1) \wedge$$
$$CustomerOrder(co_1) \wedge hasCreateDatetime(co_1,\ p\_dt_1) \wedge$$
$$hasCustomerOrder(ca_1,\ co_1) \wedge before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge$$
$$... \wedge CustomerAccount(ca_n) \wedge CustomerOrder(co_n) \wedge$$
$$hasCreateDatetime(co_n,\ p\_dt_n) \wedge hasCustomerOrder(ca_n,\ co_n) \wedge$$
$$before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge num = n. \quad \text{(FOL d-9)}$$

Thus, the retention rate value is defined by:

$$(\forall\ ret\_rate,\ num\_ret\_cus, num\_cus)\ RetentionRate(ret\_rate) \wedge$$
$$hasNumberOfRetainedCustomers(ret\_rate,\ num\_ret\_cus) \wedge$$

$$hasNumberOfCustomers(ret\_rate,\ num\_cus) \supset$$
$$hasValue(ret\_rate,\ DIV(num\_ret\_cus,\ num\_cus)). \quad \text{(FOL d-10)}$$

6.4 Marketing Actions

The TOVE ontologies are used to describe marketing actions. In this section some examples are briefly discussed.

6.4.1 Customer Acquisition

The primary means of growth in retailing for most businesses involves the acquisition of new customers. This could involve finding customers who previously were not aware of the products offered, were not candidates for purchasing the products (for example, baby diapers for new parents), or customers who in the past have bought from other providers. Customer acquisition involves identifying potential customers (Figure 6.2) and developing plans for establishing long-term relationships with them.

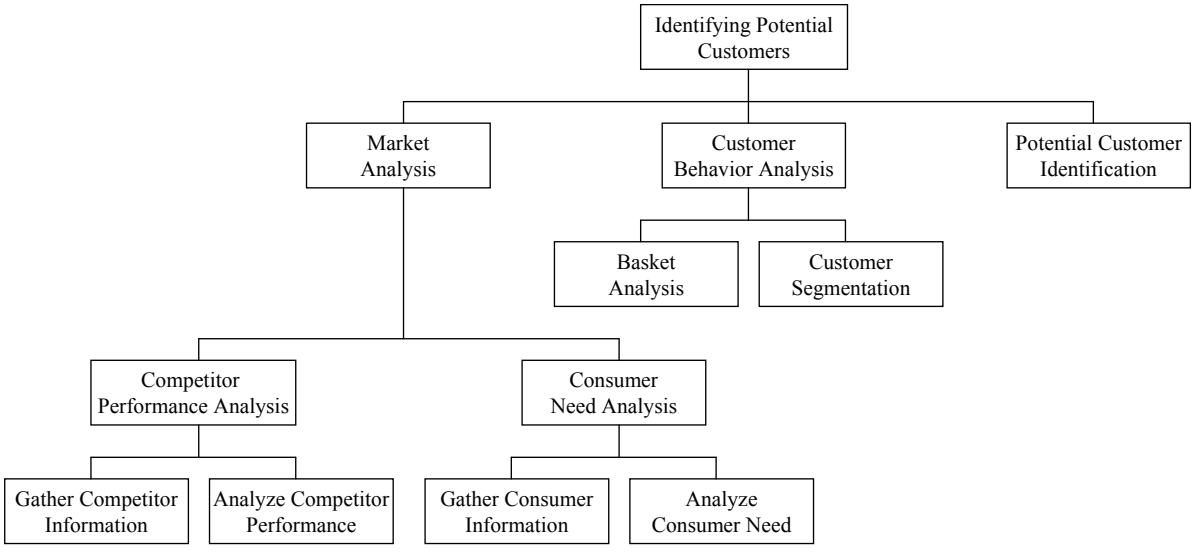

Figure 6.2: Identifying potential customers for customer acquisition (sub-activity relationships)

6.4.2 Customer Retention

Another important marketing activity is customer retention. Customer retention is a strategy whose objective is to keep a company's customers, prevent them from going to the competition, and retain their revenue contribution. The general belief among managers is that it costs less to keep an existing customer than to acquire a new one [Hughes, 1997]. Customer retention is the driving force behind customer relationship management, relationship marketing and loyalty marketing.

Cross-selling is the strategy of selling other products to a customer who has already purchased. It is designed to increase customer's trust in the company and reduce the risk of customer's switching to competition [Olszak and Ziemba, 2006]. In order to decide which products to offer to a customer, one can analyze the purchasing behavior of other previous customers who purchased the same items, and recommend other products that were purchased by these customers. It is also important to know if customers are actually buying items that the company has identified as cross-sell products. This helps in better understanding customer needs and improving recommendations. A product is mapped to another product if it is believed that they have cross-selling potentials.

$$(\forall\, ppm)\; CrossSellProductProductMap(ppm) \equiv ProductProductMap(ppm) \\ \wedge\, hasMappingType(ppm,\, `cross\text{-}sell'). \qquad \text{(FOL d-11)}$$

Thus, a product is cross-listed with another product if they both belong to a cross-sell mapping:

$$(\forall\, ppm,\; par\_p,\; p)\; CrossSellProductProductMap(ppm) \wedge Product(par\_p) \wedge \\ Product(p) \wedge hasParentProduct(ppm,\; par\_p) \wedge hasProduct(ppm,\; p) \supset \\ (isCrossListedWith(par\_p,\; p) \wedge isCrossListedWith(p,\; par\_p)). \qquad \text{(FOL d-12)}$$

A customer order is considered to contain cross-sell products if at least two items in that order belong to a cross-sell product map:

$$(\forall\ co,\ p1,\ p2)\ CustomerOrder(po) \wedge Product(p1) \wedge Product(p2) \wedge$$
$$hasProduct(co,\ p1) \wedge hasProduct(co,\ p2) \wedge isCrossListedWith(p1,\ p2) \supset$$
$$containsCrossSellProducts(co,\ true). \qquad \text{(FOL d-13)}$$

6.4.3 Product Selection

The product selection problem, in the retailing context, is concerned with determining what products to offer for sale with the aim of maximizing profitability subject to the constraints related to budget limitations and product demand from market sources. Products that should be chosen are those that are capable of generating revenues sufficient to cover the fixed and variable costs [Spence, 1976]. Figure 3.11 shows the breakdown of the product selection aggregate activity. In this case, the cost-benefit analysis process involves comparing direct and indirect costs and marketing expenses to projected sales for a proposed product in order to choose the most profitable option.

6.4.3.1 Product Segmentation Based on Profit

Product segmentation based on profit is an approach in which products are grouped according to the profit they make during a specified period of time, and results in the identification of the most profitable products. A product profit segment is represented by the concept $ProductProfitSegment$ and is a subclass of $Segment$ and has the following properties:

- $hasStartDatetime(pp\_seg,\ s\_dt)$,
- $hasEndDatetime(pp\_seg,\ e\_dt)$,
- $hasPPMin(pp\_seg,\ min\_value)$,
- $hasPPMax(pp\_seg,\ max\_value)$.

In order for a product to belong to a profit segment, the profit associated with that product is the specified time period must be in the range [min_value, max_value]:

$$
\begin{aligned}
&(\forall\ p,\ pr,\ val,\ m,\ pp\_seg, min\_value,\ max\_value,\ s\_dt,\ e\_dt,\ co_1, ...,\ co_n,\ p\_dt_1,\\
&\quad ...,\ p\_dt_n,\ oli_1,\ ...,\ oli_n,\ q_1,\ ...,\ q_n)\ Product(p) \wedge Price(pr) \wedge\\
&\quad hasCurrentPrice(p,\ pr) \wedge hasValue(pr,\ val) \wedge hasGPMargin(p,\ m) \wedge\\
&\quad CustomerOrder(co_1) \wedge hasStatus(co_1,\ `compeleted') \wedge OrderLineItem(oli_1)\\
&\quad \wedge hasOrderLineItem(co_1,\ oli_1) \wedge hasProduct(oli_1,\ p) \wedge\\
&\quad hasQuantity(oli_1,\ q_1) \wedge before(s\_dt,\ p\_dt_1) \wedge before(p\_dt_1,\ e\_dt) \wedge ... \wedge\\
&\quad CustomerOrder(co_n) \wedge hasStatus(co_n,\ `compeleted') \wedge OrderLineItem(oli_n)\\
&\quad \wedge hasOrderLineItem(co_n,\ oli_n) \wedge hasProduct(oli_n,\ p) \wedge\\
&\quad hasQuantity(oli_n,\ q_n) \wedge before(s\_dt,\ p\_dt_n) \wedge before(p\_dt_n,\ e\_dt) \wedge\\
&\quad ProductProfitSegment(pp\_seg) \wedge hasPPMin(pp\_seg, min\_value) \wedge\\
&\quad hasPPMax(pp\_seg, max\_value) \wedge \{(m \times val) \times (q_1 + ... + q_n)\} \geq min\_value\\
&\quad \wedge \{(m \times val) \times (q_1 + ... + q_n)\} < max\_value \supset belongsToSegment(p,\ pp\_seg).
\end{aligned}
$$

(FOL d-14)

Assuming that the segments don't overlap, the most profitable segment can be defined as the segment which has a minimum value greater than the maximum value of all the other segments that include at least one product:

$$
\begin{aligned}
&(\forall\ pp\_seg_1,\ ...,\ pp\_seg_n,\ min\_value_1,\ max\_value_2,\ ..., max\_value_n)\\
&\quad MostProfitableProductSegment(pp\_seg_1) \equiv\\
&\quad ProductProfitSegment(pp\_seg_1) \wedge hasPPMin(pp\_seg_1,\ min\_value_1) \wedge\\
&\quad hasPPMax(pp\_seg_1,\ max\_value_1) \wedge ... \wedge ProductProfitSegment(pp\_seg_n) \wedge\\
&\quad hasPPMin(pp\_seg_n,\ min\_value_n) \wedge hasPPMax(pp\_seg_n,\ max\_value_n) \wedge\\
&\quad \{(\exists\ p)\ Product(p) \wedge belongsToSegment(p,\ pp\_seg_1)\} \wedge\\
&\quad \{(min\_value_1 \geq max\_value_2) \vee (min\_value_1 < max\_value_2 \wedge\\
&\qquad (\neg\exists\ p')\ Product(p') \wedge belongsToSegment(p', pp\_seg_2)\} \wedge ... \wedge
\end{aligned}
$$

$$\{(min\_value_1 \geq max\_value_n) \vee (min\_value_1 < max\_value_n \wedge$$
$$(\neg\exists\, p')\ Product(p') \wedge belongsToSegment(p', pp\_seg_n)\} \qquad \text{(FOL d-15)}$$

6.4.4 Product Pricing

Pricing is the process of assigning prices to products [Figure 3.12]. The price assigned to a product is influenced by the type of distribution channel used, the type of promotions used, and the quality of the product. There are many different approaches to pricing, some of which include [Dibb et al., 2001]:

- Competition oriented pricing: A pricing method whereby a business considers costs and revenue to be secondary to competitors' prices.

- Cost oriented pricing: A pricing method whereby a monetary amount or percentage is added to the cost of a product.

- Cost plus pricing: A pricing method based on adding a specified amount or percentage to the seller's cost after that cost is determined.

- Demand oriented pricing: A pricing method based on the level of demand for a product, resulting in a high price when demand is strong and a low price when demand is weak.

- Experience curve pricing: A pricing policy in which a company expands its market share by fixing a low price that high cost competitors cannot match.

- Marketing oriented pricing: A pricing method whereby a company takes into account a wide range of factors including marketing strategy, competition, value to the customer, price-quality relationships, explicability, costs, product line pricing, negotiating margins, political factors and effect on distributors/retailers.

- Prestige pricing: A pricing method whereby prices are set at an artificially high level to provide prestige or a quality image.
- Promotional pricing: Pricing related to the short term promotion of a particular product.
- Special event pricing: Advertised sales or price cutting that is linked to a holiday, season or event to increase sales volume.

6.5 Competency Questions Revisited

In this section we repeat the competency questions and follow each with the query that would provide the answer. Note that parameters that are preceded with a "?" are variables. If a query is not existentially quantified, then it is a query for free variable bindings.

- How much is the total revenue earned in a given period?
  - **Query 1:**

    $$SalesRevenue(?rev) \wedge hasStartDatetime(?rev,\ \mathbf{s}) \wedge$$
    $$hasEndDatetime(?rev,\ textbfe) \wedge hasAmount(?rev,\ amount).$$

- What is the retention rate over a period specified by the starting time point s and ending time point t?
  - **Query 2:**

    $$RetentionRate(?rr) \wedge hasValue(?rr,\ ?val) \wedge hasStartDatetime(?rr,\ \mathbf{s}) \wedge$$
    $$hasEndDatetime(?rr,\ \mathbf{t}).$$

- What are the most profitable products in a specified time period?

- **Query 3:**

$$MostProfitableProductSegment(?seg) \wedge hasStartDatetime(?seg,\ \mathbf{s}) \wedge$$
$$hasEndDatetime(?seg,\ \mathbf{t}) \wedge Product?p) \wedge belongsToSegment(?p,\ ?seg).$$

- Which items are likely to be purchased by customers who bought SKU x?

  - **Query 4:**

$$CrossSellProductProductMap(?ppm) \wedge SKU(\mathbf{x}) \wedge Product(?p) \wedge$$
$$hasSKU(?p,\ \mathbf{x}) \wedge Product(?o\_p) \wedge isCrossListedWith(?p,\ ?o\_p) \wedge$$
$$SKU(?y) \wedge hasSKU(?o\_p,\ ?y).$$

- Which customers add a cross-sell product to their order?

  - **Query 5:**

$$CustomerAccount(?ca) \wedge CustomerOrder(?co) \wedge$$
$$hasCustomerOrder(?ca,\ ?co) \wedge containsCrossSellProducts(co,\ true).$$

- How much is the acquisition profit (loss) in a specified period of time?

  - **Query 6:**

$$AcquisitionProfit(?ap) \wedge hasStartDatetime(?ap,\ \mathbf{s}) \wedge$$
$$hasEndDatetime(?ap,\ \mathbf{t}) \wedge hasAmount(?ap,\ ?val).$$

- How much is the total retention cost over a period specified by the starting time point s and ending time point t?

  - **Query 7:**

$$RetentionActivity(?ret) \wedge ActivityCostPoint(?acp) \wedge$$
$$hasActivity(?acp,\ ?ret) \wedge hasCost(?acp,\ ?c) \wedge hasTime(?acp,\ \mathbf{t}).$$

- How effective was a promotion?

  - Conversion rate can be used to measure the effectiveness of a promotion by comparing the conversion rate before and after running the promotion.

  - **Query 8:**

$$Promotion(?prom) \wedge hasStartDatetime(?prom,\ ?s) \wedge$$
$$hasEndDatetime(?prom,\ ?t) \wedge ConversionRate(?cr) \wedge$$
$$hasValue(?cr,\ ?val) \wedge hasStartDatetime(?cr,\ ?s) \wedge$$
$$hasEndDatetime(?cr,\ ?t) \wedge ConversionRate(?cr1) \wedge$$
$$hasValue(?cr1,\ ?val1) \wedge hasStartDatetime(?cr1,\ ?s - (?t - ?s)) \wedge$$
$$hasEndDatetime(?cr1,\ ?s) \wedge ?val > ?val1.$$

6.6 Summary

The entities, attributes and relations needed for effective marketing were defined in this chapter. Also, a set of competency questions was presented to demonstrate the use of the terminology in retail environments. Other marketing related competency questions that are important but not addressed in this research include:

- How will the margins improve if a promotion is run?
- Which products should be recommended to customers?
- Is the pricing structure satisfactory?

An ontology for retail was presented in this chapter along with the previous three chapters, which consists of a number of ground assertions on top of which more complex terms are defined. The ontologies developed as part of the TOVE project are used in the development of the retail ontology. First order logic is used because of its expressive and declarative capability, i.e. by applying it in an inference engine the ontology will have

the deductive capability to answer queries. Further work can be done on completing the axiomatization of the ontology and extending it to capture additional concepts.

Chapter 7

Reasoning Over Large Datasets

7.1 Introduction

Ontologies are being successfully used in many applications and play a major role in the Semantic Web. Efforts have been made to develop various ontology languages, each providing a different expressive power and also computational complexity for reasoning [Zhang, 2005].

State-of-the-art ontology languages, such as DAML-OIL [van Harmelen et al., 2001] and OWL [Bechhofer et al., 2004], are based on expressive description logics (DLs). As well as giving a precise and unambiguous meaning to descriptions of the domain, this also allows for the development of reasoning algorithms that can be used to answer complex questions about the domain [Horrocks, 2007]. Examples of highly optimized DL reasoners include FaCT++ [Horrocks, 1998], Racer [Haarslev and Moller, 2003], and Pellet [Sirin et al., 2007]. It is widely recognized, however, that the expressive power of such languages is inadequate in some applications [Horrocks and Voronkov, 2006] (as is the case for the TOVE ontologies and the Retail Ontology). This has led to efforts to develop languages based on more expressive logics up to and including full first-order predicate logic [Horrocks and Voronkov, 2006].

Reasoning is essential in supporting both the design of high quality ontologies, and the deployment of ontologies in applications [Horrocks, 2007]. Designing ontologies is an extremely complex task, and modern ontology design tools typically use DL reasoners to provide feedback to the user about the logical implications of their design such as highlighting inconsistencies and redundancies. Reasoning is also important when ontologies are deployed in applications, in order to answer structural queries about the domain and to retrieve data.

The development of more expressive ontology languages requires the use of theorem provers able to reason with full first-order logic and even its extensions [Horrocks and Voronkov, 2006]. However, initial experimentation in [Tsarkov et al., 2004] suggests that an efficient DL reasoner can outperform a highly optimized FOL theorem prover on tasks such as classification. Recently, [Horrocks and Voronkov, 2006] have shown that a carefully engineered theorem prover can answer queries to ontologies having over 15,000 first-order axioms with equality.

To be applicable in real world situations, the system should be able to deal efficiently with the huge amounts of data usually present in applications such as bioinformatics and e-commerce. For example, in the context of the retail ontology, it must be capable of reasoning efficiently over billions of transactions and hundreds of thousands of customers. Although during the last few years efficient reasoning over large datasets by developing new scalable reasoning systems has been the focus of many studies, the ability to use an ontology with very large datasets is still a challenge to current systems.

For the foreseeable future, most data will continue to be stored in relational databases. Relational databases have an established record of storing and querying large amounts of data efficiently, but lack the ability to perform the inference sanctioned by entailments. To work with these data in ontology-based applications, tools and techniques that bridge the two models are required. Using database technology alongside the DL reasoner has been the focus of some research which have resulted in the development of systems such as

the OWL Instance Store [Bechhofer et al., 2005] that uses a combination of DL reasoning and relational database systems to deal with large volumes of instance data, and KAON2 [Hustadt et al., 2004] that reduces OWL ontologies to disjunctive datalog programs, and uses deductive database techniques to enable it to deal with very large datasets.

This chapter addresses the issue of reasoning efficiently with the large amounts of data present in retail systems and is structured as follows. Section 4.2 describes the approach taken in this research for implementing the ontology. In Section 4.3 a technique for reasoning efficiently with large datasets is introduced. Section 4.4 is concerned with preliminary empirical results and Section 4.5 concludes this chapter with a summary and a description of future work.

7.2 The Approach in this Research

As stated in the introduction, knowledge representation languages based on description logics are not sufficiently expressive to encode all existing axioms in the retail ontology, which in turn means that highly optimized DL reasoners can not be used to reason over the ontology. It also means that the new scalable reasoning systems such as the OWL Instance Store and KAON2 can not be used for efficient reasoning, either.

One way to overcome the first limitation is to use a FOL theorem prover such as Vampire [Riazanov and Voronokov, 2006] or Otter [McCune, 2003] to reason with full first-order logic. Initial experimentation in [Tsarkov et al., 2004], however, suggests that an efficient DL reasoner can outperform a highly optimized FOL theorem prover on tasks such as classification. Another approach would be to use a reasoner capable of reasoning efficiently with both DL and FOL axioms; in other words working with an FOL extension of an expressive DL, in which the DL handles the taxonomy related axioms of the theory, and the FOL handles the remaining axioms possibly relating to the DL axioms [Sanner and McIlraith, 2006].

Due to the size of the ontology, in order to be able to answer queries effectively it is essential to use database technology alongside the inference engine. Thus, to overcome the second limitation, an approach for answering first-order expressive queries where the relations are populated by databases that potentially contain hundreds of millions of instances is introduced in the next section.

7.3 Reasoning with Large Datasets

7.3.1 Representation

The implementation proposed here includes a knowledgebase (KB) that contains the definitions of concepts, roles and axioms, and a relational database (RDB) in which actual instances are kept (actual instances are not kept in the KB).

For each concept in the KB a table is created in the RDB. The concept's attributes and relations are then mapped to fields in the corresponding table. Each table also has a unique field ID as primary key to uniquely identify each row (instance) in that table. If an attribute or relation involves another concept, then the ID of that concept is considered as a foreign key in the initial table. For example, if there is a concept *Person* with properties *has_name*, *has_gender*, and *has_address* in the KB where *Address* is defined as a separate concept, then the schema for the relation *Person* would look like:

person(**person_ID**, name, gender, address_ID),

where *address_ID* is a reference to the *Address* table. Each table is then populated by direct instances of the corresponding concept. If storage is an issue, then tables can be created only for concepts that have at least one direct instance.

In addition to the tables for all the concepts, a separate table called the *virtual ontology table* is also kept in the RDB. This table contains the mapping of the ontology concepts and roles to the tables and their fields in the database (ontology-to-relational

mapping). The required information to be kept in this table can be described using the following fields: 1) concept/role name in the ontology, 2) type which indicates whether the row refers to a concept or a role, 3) the associated table name, and 4) the field(s) in the table the concept or role is mapped to. The relational schema of the virtual ontology can be described as:

VirtualOntology(**name**, type, table_name, domain, range).

For example, the rows corresponding to the example given above would be:

(Person, concept, person, person_ID, <NULL>,

(has_gender, role, person, person_ID, gender),

(has_name, role, person, person_ID, name),

(has_address, role, person, person_ID, address_ID).

Raw data are only kept in the RDB. For each group of similar instances in the database, one instance referred to as the *representative instance* along with its distinguishing properties are inserted into the KB. This can be done in a domain like retailing where it is usually not necessary to reason over all existing instances of a particular concept since in most cases they are very similar to each other. In the example above for instance, it might be useful to distinguish between people based on their gender. Accordingly, two sets of instances would be inserted into the KB, one corresponding to all females and the other to all males:

$Person(p1)$ $Person(p2)$

$has\_gender(p1, `female')$ $has\_gender(p2, `male')$

Other representative instances corresponding to the concept *Person* can also be inserted into the KB, for example those that distinguish people according to demographic properties. This significantly reduces the number of instances that are stored in the KB.

The SQL version of each representative instance and the instances it relates to are kept in a table called the *instance table* in the RDB. The SQL commands are generated

automatically from the representative instances and the virtual ontology table, and are used to retrieve all the instances a particular representative instance specifies (see section 4.3.4 for examples). The relational schema of the instance table is given below.

Instance(**name**, select_statement, from_statement, where_statement, additional).

The corresponding rows for *p1* and *p2* in the above example would be:

('p1', 'tp1.person_ID', 'person as tp1', 'tp1.gender=female', <NULL>),
('p2', 'tp2.person_ID', 'person as tp2', 'tp2.gender=male', <NULL>).

7.3.2 Query Answering

Applications of ontologies typically involve querying, by using a reasoner, to determine when an individual satisfies a query expression, or to retrieve all individuals satisfying a given query [Horrocks and Voronkov, 2006]. Since actual instances are not kept in the KB, another technique for instance retrieval has to be devised.

One way to go about it is to do the reasoning over representative instances, and when actual individuals are required construct a SQL command using representative instances and the virtual ontology and instance tables. Note that if an answer is not found, it does not necessarily mean that no instances satisfying the given query actually exist, but only implies that no representative instance satisfying the query can be found.

The process of instance retrieval used in this research is as follows. Start by querying the KB and finding all representative instances that satisfy a given query. Then, construct a SQL command from the answers returned by the theorem prover or reasoner in order to query the database. This can be done using the instance table. First, the SQL statements related to the representative instances in the answer sets are retrieved from the database. Then, for each answer set, the SQL statements for the instances in that set are combined to form a new SQL query (more description to follow). The final answer is the union of the SQL queries created for each answer set.

The SQL statements for the instances in an answer set are combined as follows. If the value of the *additional* field of all the retrieved instances from the instance table is equal to *<NULL>*, then the SQL statements are combined by concatenating the select, from, and where parts of all the SQL statements corresponding to each of the instances. If, on the other hand, the answer set contains one or more representative instances for which the value of the *additional* field of the corresponding rows in the instance table is not equal to *<NULL>*, then these instances are first removed from the answer set and a SQL statement for the remaining instances in the answer set is created in the manner mentioned above. Then for each of the removed instances, a separate SQL statement is created using the row itself and all the other rows it was related to using the same procedure as above. Finally, the two or more SQL statements created are joined on the columns they share (see section 4.3.4 for an example).

The queries which involve instance retrieval can be divided into two groups depending on whether they contain ground constants in the query statement or not. For example, *has_name(?x, 'Maryam')* contains the ground constant 'Maryam', but *has_name(?x, ?y)* doesn't (parameters that are preceded with a "?" denote variables). Since the ground constants are not directly kept in the knowledge base, in order to get correct results it is necessary to make changes to those queries which contain one or more ground constants. This can be achieved by replacing the ground constants in the query statement with unique variables and finding a solution for the new query in terms of the representative instances. Once the solution is found, the variables are exchanged with the ground constants they were representing (see section 4.3.4 for an example).

The procedure for query answering which involves instance retrieval can be summarized as follows:

1. Replace all ground constants in the query with new unique variables.

2. Query the theorem prover or reasoner.

3. If an answer is found, construct a SQL statement using the method described above.

4. Query the database using the SQL statement created and retrieve instances.

In order to make the system more efficient, for query processing that is reducible to "looking up" an answer that is explicitly represented in the database, there is no need to perform any reasoning and the SQL query can be constructed by using only the virtual ontology table. For example, assuming an ontology that includes the role *has_name* given in Section 4.3.1, and a query *has_name(?x,?y)*, as can be seen, all the information needed to construct the SQL query below is already contained in the row corresponding to this role in the virtual ontology.

SELECT DISTINCT $person\_ID$, $name$ FROM $person$.

7.3.3 Other Issues

Since actual instances are not kept in the KB, there are some common operations that can no longer be performed. This section discusses these operations and introduces methods for performing them.

7.3.3.1 Arithmetic Operations

Arithmetic operations can no longer be perfomed. In order to handle such operations, five new binary predicates are introduced: ADD, SUB, MUL, DIV, and POW. The system performs arithmetic operations by transforming each expression contained in the answer set the reasoner returns from prefix to infix notation with mathematical symbols, while constructing the SQL statement. In this way, the database engine would be able to compute the required expressions when retrieving instances.

As an example assume the reasoner returns { *x, ADD(y, z)* } as an answer to a query. Further, assume the following rows correspond to x, y, and z respectively (from the instance table):

('x', 'tx.x_ID', 'xTable as tx', <NULL>, <NULL>),

('y', 'tx.y_ID', 'xTable as tx', <NULL>, <NULL>),

('z', 'tx.z_ID', 'xTable as tx', <NULL>, <NULL>).

The first thing the system does is to transform the *ADD(y, z)* from prefix to infix notation with mathematical symbols, thus, $y + z$. Then, instead of inserting *tx.y_ID* and *tx.z_ID* in the select part of the SQL statement, it inserts *tx.y_ID + tx.z_ID*. The rest is the same as for any other answer set.

7.3.3.2 Functions

Some queries require counting the number of instances that satisfy a given statement. Others require summing up the values of an attribute over some instances or choosing the instance which has the maximum (or minimum) value for an attribute. Since these operations are performed over instances, the system is incapable of handling them while reasoning. Examples of such queries in the domain of online retailing include the number of orders placed by a customer or the total amount of all items a customer has purchased.

Some of these operations are done in SQL using aggregate functions COUNT(), SUM(), MAX(), and MIN(). The same keywords are used for the name of the predicates designated to represent such operations in the knowledgebase. The rows are then grouped using SQL GROUP BY.

Other user-defined functions can also be specified using the *user-defined function* UDF mechanism[1]. These functions can then be invoked in SQL statements just like native functions such as SUM().

[1]For more information on how the UDF mechanism works in MySQL visit http://dev.mysql.com/doc/refman/5.0/en/adding-functions.html.

7.3.4 Example

This section illustrates an example to help the reader in better understanding the procedures described in previous sections.

Consider the group of purchase orders that refer to a product and the following set of ground terms (representative instances) that represent this group in the knowledgebase:

$PurchaseOrder(po)$

$PurchaseOrderLine(pol)$

$hasPurchaseOrderLine(po,\ pol)$

$Product(p)$

$hasProduct(pol,\ p)$

Assume *Product*, *PurchaseOrder*, and *PurchaseOrderLine* concepts are mapped to *product*, *purchase_order*, and *purchase_order_line* tables with the following relational schema, respectively:

product(**Product_ID**),

purchase_order_line(**POLine_ID**, PO_ID, Product_ID),

purchase_order(**PO_ID**).

where *PurchaseOrderLine* has two attributes *belongsToPurchaseOrder* and *hasProduct* which are mapped to *PO_ID* and *Product_ID* fields in the associated table. The corresponding rows for each of the above instances in the instance table are shown in Table 4.1. The value of the *additional* field is <NULL> for all three instances.

Now consider the following two queries:

1. $\exists\ ?x,\ ?y,\ PurchaseOrder(?x) \bigwedge hasProduct(?x, ?y)$

2. $\exists\ ?x, PurchaseOrder(?x) \bigwedge hasProduct(?x, `chocolateChipCookie')$

Instance Name	**Select Statement**	**From Statement**	**Where Statement**
p	$tp.Product\_ID$	$product\ as\ tp$, $purchase\_order\_line\ as\ tpol$	$tp.Product\_ID = tpol.Product\_ID$
pol	$tpol.POLine\_ID$	$purchase\_order\_line\ as\ tpol$, $product\ as\ tp$, $purchase\_order\ as\ tpo$	$tp.Product\_ID = tpol.Product\_ID$, $tpol.PO\_ID = tpo.PO\_ID$
po	$tpo.PO\_ID$	$purchase\_order\ as\ tpo$, $purchase\_order\_line\ as\ tpol$	$tpol.PO\_ID = tpo.PO\_ID$

Table 7.1: SQL commands specifying instances in the example.

Since the second query contains the ground constant *chocolateChipCookie*, this term is first replaced with a new variable *?y*, making it equivalent to the first query. Executing the query would then result in the binding of *?x* to *po* and *?y* to *p*. Note that the axiom

$$(\forall\ po,\ pol,\ p)\ PurchaseOrder(po) \bigwedge hasPurchaseOrderLine(po,\ pol) \bigwedge hasProduct(pol,\ p) \supset hasProduct(po,\ p)$$

exists in the knowledgebase. To retrieve the instances corresponding to the first query from the database, the SQL queries for *po* and *p* from the instance table are simply merged together.

SELECT $tpo.PO\_ID,\ tp.Product\_ID$
FROM $purchase\_order\ as\ tpo,\ purchase\_order\_line\ as\ tpol,$
$product\ as\ tp$
WHERE $tpol.PO\_ID = tpo.PO\_ID$
AND $tp.Product\_ID = tpol.Product\_ID.$

For the second query, the statement *ti1.Product_ID='chocolateChipCookie'* is also added to the WHERE clause to enforce the constraint given in the query. The SQL statement associated with the second query is hence:

SELECT $tpo.PO\_ID$

FROM $purchase\_order\ as\ tpo$, $purchase\_order\_line\ as\ tpol$,

$product\ as\ tp$

WHERE $tpol.PO\_ID = tpol.PO\_ID$

AND $tp.Product\_ID = tpol.Product\_ID$

AND $tp.Product\_ID = `chocolateChipCookie'$.

7.4 Empirical Results

As an initial attempt to investigate the applicability of the retail ontology, we have implemented it using the DL-FOL reasoner of Sanner & McIlraith [Sanner and McIlraith, 2006] and the MySQL database management system. The DL-FOL reasoner is based on an ordered theory resolution calculus for hybrid reasoning in unrestricted FOL extensions of the DL *SHI*, which permits the integration of highly optimized FOL theorem provers and DL reasoners while maintaining soundness and refutational completeness [Sanner and McIlraith, 2006]. In this implementation, concepts, roles and axioms are represented using the DL-FOL syntax. The following DL-FOL notation is used to state queries:

$$(ask\ [cwa]\ [boolean]\ dlfol\ \ 'FOL - QUERY').$$

Here, brackets are used to indicate optional items. *cwa* indicates that the closed-world assumption should be used during reasoning and *boolean* indicates that the query should be a simple Boolean true (proved) or false query instead of a query for free variable bindings. The full DL-FOL syntax can be found in Appendix A.

The experiment was performed on a KB consisting of all the concepts, roles, and axioms described in Chapters 3, 4, 5, and 6 along with 65 representative instances. The

actual database consisted of 1123854 transactions performed by 915144 distinct customers over a period of three months. The DL-FOL reasoning time along with the number of clauses generated and proof length for some sample queries are given in Table 7.2.

Table 7.2: Time, number of clauses generated, and proof lenght for sample queries.

Query	**Time (sec)**	**Number of Clauses Generated**	**Proof Length**
$Person(?x)$	1.047	-	-
$(\forall\ ?x)\ Customer(?x) \supset Agent(?x)$	0.06	37	1
$hasTotal(?po,\ ?tot)$	1.188	-	-
$hasPurchaseOrder(?ca,\ ?po) \wedge hasTotal(?po,\ ?x)$	83.56	1472	4
$PurchaseOrder(?po) \wedge hasStatus(?po,\ `cancelled')$	107.41	1496	6
$POwithCrossSellProducts(?po)$	98.57	1559	12
$ProductSegment(?s) \wedge hasProduct(?s,\ xsd\#696833)$ $\wedge\ belongsToSegment(?ca,\ ?s)$	119.12	1656	10
$belongsToSegment(?ca,\ ?s) \supset$ $ProductSegment(?s)\ \vee\ GenderSegment(?s)$	235.89	1765	3
$InventoryItem(?inv) \wedge hasProduct(?inv,\ ?p)$ $\wedge\ hasQuantity(?inv,\ ?q)$	221.21	1798	4
$reachedReorderLevel(?inv,\ true)$	94.82	1492	6
$purchasedBySameCustomer(?x,\ ?y)$	394.52	2355	14
$ConversionRate(?r) \wedge hasStartDatetime(?r,\ ?s) \wedge$ $hasEndDatetime(?r,\ ?e) \wedge hasValue(?r,\ ?val)$	510.30	3568	21

All tests were made on Pentium 4, 3.20 GHz with 1.49 GB of RAM running Windows XP. We ran the DL-FOL reasoner with a clause limit of 5000 and a maximum iterations limit of 5000 per task. We used the technique described in Chapter 2 to estimate the CLV model parameters. The time recorded is solely due to DL-FOL reasoning and creating a SQL statement (the time spent on executing the SQL statement is not considered). As mentioned previously, for query processing that is reducible to "looking up" an answer that is explicitly represented in the database, there is no need to perform any reasoning and the SQL query can be constructed by using only the virtual ontology table.

Given that it was impossible to reason over the actual instances when all were placed in the KB, the query times seem very reasonable. Nevertheless, the reasoning still takes too long for some queries and the number of clauses generated for those queries is large. This could be in part due to the fact that the ontology itself is somewhat large. It is worth mentioning that the DL-FOL reasoner is not yet fully optimized. Using other optimized FOL theorem provers might further speed up the reasoning process. Also, since it has been shown that there is more than one way to represent the same knowledge with each approach having different complexity to answer queries [Fadel, 1994], using a different representation might also improve reasoning time.

7.5 Conclusions

To be applicable in real world situations, the system should be able to deal efficiently with the huge amounts of data present in retail systems. In this chapter implementation issues were addressed and an approach was introduced for answering first-order expressive queries where the relations are populated by databases that potentially contain hundreds of millions of instances. Future work can be done on automating the process of selecting representative instances using data mining techniques. It would also be interesting to implement the ontology using other theorem provers or reasoners that are sufficiently

expressive and to compare them according to their performance.

Chapter 8

Conclusions

Decision making is considered as the central aspect of most management and business activities. In order to make informed and effective decisions, managers in a modern enterprise should be able to take advantage of all available data. However, with the constantly increasing volume of data, both internal and external to the enterprise, the process can become a challenge. Since the accuracy and quality of decisions is important in many situations, using information systems to aid the process of decision making and to improve the effectiveness of the decision maker has been a major focus of information systems research for many years.

Business intelligence systems have been developed as a means to answer the managers' request to better understand the situation of their business and to improve the decision process by providing the means to transform the available data into information and derive specific and timely knowledge about customers, products and markets.

The focus of this research was on building an efficient customer-centric business intelligence system applied to online retail. The work described a retail ontology that could automatically deduce answers to retail queries based upon the system's general knowledge of online retailing and actual data, and presented a technique for dealing efficiently with the large amounts of data present in retailing systems. There are three main advantages

in using ontologies:

1. They can be used to capture and represent business semantics, which would in return result in using the same vocabulary across the enterprise.

2. By assuming deductive capability as provided by an inference engine, they make it possible to explore the terminology and generate further knowledge.

3. Managers can compose their own first-order sentences to query the system using the existing terminology; thereby avoiding the need to rely on others.

For the development of the ontology, the competency requirement was chosen as the focal point of the effort. Competency questions represent the starting point of the ontology development. They also define the types of tasks that the representation can be used in. Through the development of the retail ontology, we think we have fulfilled the following requirements:

- **Competence** is satisfied through the definition of the questions that the system should be able to answer.

- **Generality** is satisfied as the ontology can be used in different retailing environments.

- **Granularity** is satisfied through the implementation of the definitions and constraints in DL-FOL.

- **Efficiency** is partly satisfied through using a relational database alongside the reasoning engine. However, since it has been shown that there is more than one way to represent the same knowledge with each approach having different complexity to answer queries, this area requires the focus of more future work.

- **Perspicuity** is satisfied with the availability of the descriptive methodology.

I have presented a synopsis of my accomplishments in my attempt to present a retail ontology for the TOVE enterprise model. I have also looked at the problem of reasoning efficiently with the large amounts of data that exist in retail systems and presented an approach for incorporating a relational database alongside a DL-FOL reasoner to address the issue. Further work can be done on completing the axiomatization of the ontology and extending it to capture additional concepts.

An interesting topic of future research would be to evaluate the ontology according to knowledge representation criteria such as consistency and completeness, as well as systems performance criteria like extensibility and scalability which can only be evaluated through the consistent use of the representation in different applications.

Appendix A

DL-FOL Syntax

This appendix contains the logical notations from the description logic (DL) and first-order logic (FOL) reasoning systems used in the DL-FOL reasoner, as described in the *DL-FOL Reasoner Manual* written by Scott Sanner.

A.1 Description Logic Syntax

The DL notation used here is a traditional LISP-like notation. See here for an example Organization ontology in this format. The specification below uses $< >$ to delineate user-specified strings of symbols and to indicate 0 or more repeats of the preceding statement. Brackets [ ] are used to indicate optional items.

A.1.1 Concepts

Concepts are defined using

$$(concept \ < CONCEPT\text{-}NAME > \ ...)$$

where ... can be any of the following:

- $(: subClassOf \ < CONC\text{-}OR\text{-}RESTR\text{-}NAME >)^*$

- $(: disjointWith\ < CONC\text{-}OR\text{-}RESTR\text{-}NAME >)^*$
- A single optional definition chosen from the following:
  - $(: and\ < CONC\text{-}OR\text{-}RESTR\text{-}NAME\text{-}1 >\ ...\ < CONC\text{-}OR\text{-}RESTR\text{-}NAME\text{-}k >)$
  - $(: or\ < CONC\text{-}OR\text{-}RESTR\text{-}NAME\text{-}1 >\ ...\ < CONC\text{-}OR\text{-}RESTR\text{-}NAME\text{-}k >)$
  - $(: not\ < CONCEPT\text{-}NAME >)$
  - $(: one-of\ < INSTANCE\text{-}NAME\text{-}1 >\ ...\ < INSTANCE\text{-}NAME\text{-}k >)$

Pairwise disjointness can also be expressed using

$$(: disjoint\ < CONCEPT\text{-}NAME\text{-}1 >\ ...\ < CONCEPT\text{-}NAME\text{-}k >).$$

A.1.2 Roles

Roles are defined using

$$(role\ < ROLE\text{-}NAME > ...)$$

where ... can be any of the following:

- $(: domain\ < CONCEPT\text{-}NAME >)$
- $(: range\ < CONCEPT\text{-}NAME >)$
- $(: subrole\ < ROLE\text{-}NAME >)$
- $(: inverse\ < ROLE\text{-}NAME >)$
- Any of the following optional properties:
  - $: functional$
  - $: symmetric$
  - $: transitive$

A.1.3 Restrictions

Restrictions can be any of the following:

- (: *exists* $<$ *ROLE-NAME* $>$ $<$ *CONC-OR-RESTR-NAME* $>$)
- (: *all* $<$ *ROLE-NAME* $>$ $<$ *CONC-OR-RESTR-NAME* $>$)
- (: *has-value* $<$ *ROLE-NAME* $>$ $<$ *INSTANCE-NAME* $>$)
- (: *at-least* $<$ *INTEGER* $>$ $<$ *ROLE-NAME* $>$ $<$ *CONC-OR-RESTR-NAME* $>$)
- (: *at-most* $<$ *INTEGER* $>$ $<$ *ROLE-NAME* $>$ $<$ *CONC-OR-RESTR-NAME* $>$)
- (: *exactly* $<$ *INTEGER* $>$ $<$ *ROLE-NAME* $>$ $<$ *CONC-OR-RESTR-NAME* $>$)
- (: *le* $<$ *ROLE-NAME* $>$ $<$ *INTEGER* $>$)
- (: *ge* $<$ *ROLE-NAME* $>$ $<$ *INTEGER* $>$)

A.1.4 Instance Assertions

There are two types of instance assertions: instance type or role-filler:

- (*tell* (*<* *CONC-NAME* *>* *<* *INSTANCE-NAME* *>*))
- (*tell* (*<* *ROLE-NAME* *>* *<* *SUBJ-INSTANCE-NAME* *>* *<* *OBJ-INSTANCE-NAME* *>*))

A.1.5 Queries

Queries are expressed using the following notation:

$$(ask[cwa][boolean](otter|dlfol|kaon2)' < FOL - QUERY >')$$

where,

- *cwa* is optional and indicates that the closed-world assumption should be used during reasoning.

- *boolean* is optional and indicates that the query should be a simple boolean true (proved) or false (i.e. could not prove) query instead of a query for free variable bindings.

- *otter, dlfol, kaon2* is a required field which indicates whether to use the Otter theorem prover or the DL-FOL reasoner, or KAON2 reasoner.

- $'< FOL\text{-}QUERY >'$ is the query statement. See below for FOL syntax. Even DL queries should be stated in FOL, e.g., to see if pets#fido is an instance of pets#Dog, query for 'pets#Dog(pets#fido)'.

A.2 First-Order Logic Syntax

The FOL syntax is simply a standard first order logic notation with built-in predicates for equality and comparison.

A.2.1 Syntactic Elements

- Universal and existential quantifier: $!A, !E$

- Variable term: $? < VAR\text{-}NAME >$

- Constant term: $< CONST\text{-}NAME >$

- Function term: $< FUN\text{-}NAME > (< TERM\text{-}1 >, ..., < TERM\text{-}k >)$

- Infix equality and inequality predicates: $< TERM\text{-}1 >=< TERM\text{-}k >$, $< TERM\text{-}1 >\sim=< TERM\text{-}k >$

- Infix comparison predicates: $< TERM\text{-}1 > < < TERM\text{-}k >$, $< TERM\text{-}1 > > < TERM\text{-}k >$
- Predicate: $< PRED\text{-}NAME > (< TERM\text{-}1 >, ..., < TERM\text{-}k >)$
- Disjunction, conjunction, and negation connectives: |, $\wedge$, $\sim$
- Implication, equivalence connectives: =>, <=>
- Grouping: ( )

Rules are specified using the following notation. Each rule must contain an implication with a singular non-negated RHS.

$$(rule\ '...').$$

FOL statements can be any FOL axiom with equality, less than, greater than, as well as arithmetic operators:

$$(fol\ '...').$$

Appendix B

DL-FOL Formulations of the Online Retail Ontology

This appendix contains a list of all DL and FOL formulation of terms, axioms, implications, etc. included in the online retail ontology. The complete set of axioms can be found in the TOVE manual [Fox et al., 1994].

B.1 Time Ontology Axioms

(concept *Time* (:and (:le Time_hasHour 23) (:ge Time_hasHour 0)
(:le Time_hasMinute 59) (:ge Time_hasMinute 0)
(:le Time_hasSecond 59) (:ge Time_hasSecond 0)))

(concept *TimePoint* (:and (:exactly 1 TimePoint_hasLowerBound Time)
(:exactly 1 TimePoint_hasUpperBound Time)))

(concept *TimeInterval* (:and (:exactly 1 TimeInt_hasStartTimePoint TimePoint)
(:exactly 1 TimeInt_hasEndTimePoint TimePoint)))

(concept *Hour* (:subClassOf xsd#Integer))

(concept *Minute* (:subClassOf xsd#Integer))

(concept *Second* (:subClassOf xsd#Integer))

(disjoint *Time TimePoint TimeInterval*)

(disjoint *Hour Minute Second*)

(role *Time_hasHour* (:domain Time) (:range Hour))

(role *Time_hasMinute* (:domain Time) (:range Minute))

(role *Time_hasSecond* (:domain Time) (:range Second))

(role *TimePoint_hasIdentifier* (:domain TimePoint) (:range xsd#Integer))

(role *TimePoint_hasLowerBound* (:domain TimePoint) (:range Time))

(role *TimePoint_hasUpperBound* (:domain TimePoint) (:range Time))

(role *TimePoint_before* (:domain TimePoint) (:range TimePoint))

(role *TimePoint_equals* (:domain TimePoint) (:range TimePoint))

(role *TimeInt_hasStartTimePoint* (:domain TimeInterval) (:range TimePoint))

(role *TimeInt_hasEndTimePoint* (:domain TimeInterval) (:range TimePoint))

(role *TimeInt_overlaps* (:domain TimeInterval) (:range TimeInterval))

(role *TimeInt_sameAs* (:domain TimeInterval) (:range TimeInterval) :symmetric)

(role *TimeInt_meets* (:domain TimeInterval) (:range TimeInterval))

(role *TimeInt_during* (:domain TimeInterval) (:range TimeInterval))

(role *TimeInt_starts* (:domain TimeInterval) (:range TimeInterval) :transitive)

(role *TimeInt_ends* (:domain TimeInterval) (:range TimeInterval) :transitive)

(role *TimeInt_periodContains* (:domain TimeInterval) (:range TimePoint))

(rule 'TimeInt_hasEndTimePoint(?x, ?e) ∧ TimeInt_hasStartTimePoint(?y, ?s) ∧
TimePoint_before(?s, ?e) => TimeInt_overlaps(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?x, ?s1) ∧ TimeInt_hasEndTimePoint(?x, ?e1) ∧
TimeInt_hasStartTimePoint(?y, ?s2) ∧ TimeInt_hasEndTimePoint(?y, ?e2) ∧
TimePoint_equals(?s1, ?s2) ∧ TimePoint_equals(?e1, ?e2) =>
TimeInt_sameAs(?x, ?y)')

(rule 'TimeInt_hasEndTimePoint(?x,?e) ∧ TimeInt_hasStartTimePoint(?y,?s) ∧
(TimePoint_before(?e, ?s) | TimePoint_equals(?e, ?s)) => TimeInt_meets(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?x,?s1) ∧ TimeInt_hasEndTimePoint(?x,?e1) ∧
TimeInt_hasStartTimePoint(?y,?s2) ∧ TimeInt_hasEndTimePoint(?y,?e2) ∧
TimePoint_before(?s2, ?s1) ∧ TimePoint_before(?e1, ?e2) =>
TimeInt_during(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?x,?s1) ∧ TimeInt_hasEndTimePoint(?x,?e1) ∧
TimeInt_hasStartTimePoint(?y,?s2) ∧ TimeInt_hasEndTimePoint(?y,?e2) ∧
equal(?s1, ?s2) ∧ TimePoint_before(?e1, ?e2) => starts(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?x,?s1) ∧ TimeInt_hasEndTimePoint(?x,?e1) ∧
TimeInt_hasStartTimePoint(?y,?s2) ∧ TimeInt_hasEndTimePoint(?y,?e2) ∧
equal(?e1, ?e2) ∧ TimePoint_before(?s2, ?s1) => ends(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?x,?s1) ∧ TimeInt_hasEndTimePoint(?x,?e1) ∧
TimePoint(?y) ∧ TimePoint_before(?s1, ?y) ∧ TimePoint_before(?y, ?e1) =>
period_contains(?x, ?y)')

(rule 'TimePoint(?x) ∧ TimePoint(?y) ∧ TimePoint_hasUpperBound(?x, ?u) ∧
TimePoint_hasLowerBound(?y, ?l) ∧ Time_hasHour(?u, ?h1) ∧
Time_hasHour(?l, ?h2) ∧ ?h1 < ?h2 => TimePoint_before(?x, ?y)')

(rule 'TimePoint(?x) ∧ TimePoint(?y) ∧ TimePoint_hasUpperBound(?x, ?u) ∧
TimePoint_hasLowerBound(?y, ?l) ∧ Time_hasHour(?u, ?h1) ∧
Time_hasHour(?l, ?h2) ∧ ?h1 = ?h2 ∧ Time_hasMinute(?u, ?m1) ∧
Time_hasMinute(?l, ?m2) ∧ ?m1 < ?m2 => TimePoint_before(?x, ?y)')

(rule 'TimePoint(?x) ∧ TimePoint(?y) ∧ TimePoint_hasUpperBound(?x, ?u) ∧
TimePoint_hasLowerBound(?y, ?l) ∧ Time_hasHour(?u, ?h1) ∧
Time_hasHour(?l, ?h2) ∧ ?h1 = ?h2 ∧ Time_hasMinute(?u, ?m1) ∧
Time_hasMinute(?l, ?m2) ∧ ?m1 = ?m2 ∧ Time_hasSecond(?u, ?s1) ∧
Time_hasSecond(?l, ?s2) ∧ ?s1 < ?s2 => TimePoint_before(?x, ?y)')

(rule 'TimePoint(?x) ∧ TimePoint(?y) ∧ TimePoint_hasUpperBound(?x, ?u1) ∧
TimePoint_hasLowerBound(?x, ?l1) ∧ TimePoint_hasUpperBound(?y, ?u2) ∧
TimePoint_hasLowerBound(?y, ?l2) ∧ => equal(?x, ?y)')

(rule 'TimeInterval(?x) ∧ TimePoint(?y) ∧ TimeInt_hasStartTimePoint(?x, ?s) ∧
TimeInt_hasStartTimePoint(?x, ?e) ∧ TimePoint_before(?s, ?y) ∧
TimePoint_before(?y, ?e) => period_contains(?x, ?y)')

(rule 'TimeInt_hasStartTimePoint(?t,?s) ∧ TimeInt_hasEndTimePoint(?t,?e) =>
TimePoint_before(?s, ?e)')

B.2 Activity-State Ontology Axioms

(concept *Activity*)

(concept *AggregateActivity* (:and (:subClassOf Activity)
(at-least 1 Activity_hasInitialActivity Activity)
(at-least 1 Activity_hasFinalActivity Activity)))

(concept *SubActivity* (:subClassOf Activity))

(concept *State*)

(concept *TerminalState* (:and (:subClassOf State) (:disjointWith NonTerminalState)))

(concept *NonTerminalState* (:and (:subClassOf State) (:disjointWith TerminalState)))

(concept *UseState* (:subClassOf TerminalState))

(concept *ConsumeState* (:subClassOf TerminalState))

(concept *ReleaseState* (:subClassOf TerminalState))

(concept *ProduceState* (:subClassOf TerminalState))

(concept *AndState* (:and NonTerminalState (:at-least 2 State_hasSubState State)))

(concept *OrState* (:and NonTerminalState (:at-least 2 State_hasSubState State)))

(concept *XorState* (:and NonTerminalState (:at-least 2 State_hasSubState State)))

(concept *NotState* (:subClassOf NonTerminalState))

(concept *SubState* (:or TerminalState NonTerminalState))

(role *Activity_hasStatus* (:domain Activity)
(:range (:one-of dormant executing suspended terminated completed)))
(role *Activity_hasStatusAtTimePoint* (:domain Activity) (:range TimePoint))
(role *Activity_hasEnablingState* (:domain Activity) (:range State))
(role *Activity_causesState* (:domain Activity) (:range State))
(role *Activity_hasSubActivity* (:domain Activity) (:range SubActivity) :transitive)
(role *Activity_subActivityOf* (:domain SubActivity) (:range Activity)
(:inverse Activity_hasSubActivity) :transitive)
(role *Activity_hasElaboration* (:domain Activity) (:range AggregateActivity))
(role *Activity_hasInitialActivity* (:domain AggregateActivity) (:range Activity))
(role *Activity_initialActivityFor* (:domain Activity) (:range AggregateActivity)
(:inverse Activity_hasInitialActivity))
(role *Activity_hasFinalActivity* (:domain AggregateActivity) (:range Activity))
(role *Activity_finalActivityFor* (:domain Activity) (:range AggregateActivity)
(:inverse Activity_hasFinalActivity))
(role *Activity_hasNextActivity* (:domain SubActivity) (:range SubActivity))
(role *Activity_hasTimeInterval* (:domain Activity) (:range TimeInterval))
(role *Activity_hasSimultaneousUseRestriction* (:domain Activity)
(:range Activity) :symmetric)

(role *State_hasStatus* (:domain State)
(:range (:one-of not_possible possible committed enabled
disenabled re_enabled completed)))
(role *State_hasStatusAtTimePoint* (:domain State) (:range TimePoint))
(role *State_enablesActivity* (:domain State) (:range Activity)
(:inverse Activity_hasEnablingState))

(role *State_hasCausingActivity* (:domain State) (:range Activity)
(:inverse Activity_causesState))
(role *State_uses* (:domain UseState) (:range Resource))
(role *State_consumes* (:domain ConsumeState) (:range Resource))
(role *State_releases* (:domain ReleaseState) (:range Resource))
(role *State_produces* (:domain ProduceState) (:range Resource))
(role *State_quanitiyNeeded* (:domain State) (:range Quantity))
(role *State_hasSubState* (:domain NonTerminalState) (:range SubState) :transitive)
(role *State_subStateOf* (:domain SubState) (:range NonTerminalState)
(:inverse State_hasSubState))
(role *State_hasTimeInterval* (:domain State) (:range TimeInterval))
(role *State_isRelatedTo* (:domain State) (:range Activity) :symmetric)

B.3 Resource Ontology Axioms

(concept *Resource* (:and (:at-least 1 Resource_hasRole Role))
(concept *UnitID*)
(concept *Unit*)
(concept *Location*)
(concept *Rate* (:subClassOf xsd#Float))
(concept *Quantity* (:and xsd#Integer (:exists Quantity_hasUnit Unit)))
(concept *QuantityAtSpecifiedTime* (:and Quantity
(:exists QAST_hasCheckingTimePoint TimePoint)))
(concept *QuantityAtSpecifiedLocation* (:and QuantityAtSpecifiedTime
(:and (:exists QASL_hasCheckingLocation Location)))
(concept *PhysicalDivision* (:and (:exists divisionOf Resource)
(:has-value hasDivisionType physical)) (:disjointWith FunctionalDivision))

(concept *FunctionalDivision* (:and (:exists divisionOf Resource)
(:has-value hasDivisionType functional)) (:disjointWith PhysicalDivision))
(concept *Component* (:and Resource (:at-least 1 divisionOf Resource)))
(concept *PhysicalComponent* (:and Component (:has-value hasDivisionType physical))
(:disjointWith FunctionalComponent))
(concept *FunctionalComponent* (:and Component
(:has-value hasDivisionType functional)) (:disjointWith PhysicalComponent))
(concept *Configuration* (:and (:at-least 1 Configuration_hasActivity Activity))
(concept *AmountCommitted* (:subClassOf Quantity))

(role *divisionOf* (:domain owl#Thing) (:range Resource))
(role *hasDivisionType* (:domain owl#Thing) (:range (:one-of physical functional))
(role *Component_componentOf* (:domain Component) (:range Resource))
(role *Resource_hasRole* (:domain Resource) (:range Role))
(role *Resource_isMobile* (:domain Resource) (:range (:one-of true false)))
(role *Resource_isStationary* (:domain Resource) (:range (:one-of true false))
(:inverse Resource_isMobile))
(role *Resource_measuredBy* (:domain Resource) (:range UnitID))
(role *Resource_hasUnitOfMeasurement* (:domain Resource) (:range Unit))
(role *Resource_hasRate* (:domain Resource) (:range Rate))
(role *Resource_resourceAmount* (:domain Resource) (:range Quantity))
(role *Resource_hasConsumptionSpecQuantity* (:domain Resource) (:range Quantity))
(role *Resource_hasConsumptionSpecTimeInterval* (:domain Resource)
(:range TimeInterval))
(role *Resource_hasConsumptionSpecUnit* (:domain Resource) (:range Unit))
(role *Resource_isPhysicalDivisibleWithRespect* (:domain Resource) (:range Activity))
(role *Resource_isFunctionalDivisibleWithRespect* (:domain Resource) (:range Activity))

(role *Resource_isTemporalDivisibleWithRespect* (:domain Resource) (:range Activity))

(role *Resource_hasUseSpec* (:domain Resource) (:range Quantity))

(role *Resource_hasUseSpecTimeInterval* (:domain Resource) (:range TimeInterval))

(role *Resource_hasUseSpecUnit* (:domain Resource) (:range Unit))

(role *Resource_hasProduceSpec* (:domain Resource) (:range Quantity))

(role *Resource_hasProduceSpecTimeInterval* (:domain Resource) (:range TimeInterval))

(role *Resource_hasProduceSpecUnit* (:domain Resource) (:range Unit))

(role *Resource_hasReleaseSpec* (:domain Resource) (:range Quantity))

(role *Resource_hasReleaseSpecTimeInterval* (:domain Resource) (:range TimeInterval))

(role *Resource_hasReleaseSpecUnit* (:domain Resource) (:range Unit))

(role *Resource_isContinuousWithRespectTo* (:domain Resource) (:range Activity))

(role *Resource_isDiscreteWithRespectTo* (:domain Resource) (:range Activity)
(:inverse Resource_isContinuousWithRespectTo))

(role *Resource_rp* (:domain Resource) (:range QuantityAtSpecifiedTime))

(role *Resource_rpl* (:domain Resource) (:range QuantityAtSpecifiedLocation))

(role *Resource_rexists* (:domain Resource) (:range (:one-of true false))

(role *Resource_rexistsl* (:domain Resource) (:range (:one-of true false))

(role *Resource_hasUsageMode* (:domain Resource) (:range (:one-of continuous discrete))

(role *Resource_hasConfiguration* (:domain Resource) (:range Configuration))

(role *Resource_committedTo* (:domain Resource) (:range Activity))

(role *Resource_hasCommitmentType* (:domain Resource) (:range (:one-of partly fully)))

(role *Resource_hasCommitmentTimeInterval* (:domain Resource) (:range TimeInterval))

(role *Resource_hasCommitmentQuantity* (:domain Resource)
(:range AmountCommitted))

(role *Resource_hasCommitmentUnit* (:domain Resource) (:range Unit))

(role *Resource_hasSetupTime* (:domain Resource) (:range Time))

(role *Resource_hasResourceTrend* (:domain Resource)

(:range (:one-of decreasing increasing steady)))

(role *Resource_alternativeResourceFor* (:domain Resource) (:range Activity))

(role *Unit_measuringUnitFor* (:domain Unit) (:range UnitID))

(role *Quantity_hasUnit* (:domain Quantity) (:range Unit))

(role *QAST_hasCheckingTimePoint* (:domain QuantityAtSpecifiedTime)

(:range TimePoint))

(role *QASL_hasCheckingLocation* (:domain QuantityAtSpecifiedLocation)

(:range Location))

(role *Configuration_hasActivity* (:domain Configuration) (:range Activity))

B.4 Organization Ontology Axioms

(concept *Organization*)

(concept *Division* (:subClassOf Organization))

(concept *Subdivision* (:subClassOf Division))

(concept *Goal*)

(concept *SubGoal* (:subClassOf Goal))

(concept *Role*)

(concept *Skill*)

(concept *Authority*)

(concept *Constraint*)

(concept *Agent*)

(concept *Team*)

(concept *CommunicationLink*)

(concept *CommunicationWithAuthority* (:subClassOf CommunicationLink))

(concept *OrganizationOrTeam* (:or Organization Team))

(concept *Information*)

(concept *Empowerment*)

(role *Org_consistsOf* (:domain Organization) (:range Division) :transitive)

(role *Division_partOf* (:domain Division) (:range Organization)
(:inverse Org_consistsOf) :transitive)

(role *Division_hasSubDivision* (:domain Division) (:range Subdivision) :transitive)

(role *Division_subDivisionOf* (:domain Subdivision) (:range Division)
(:inverse Division_hasSubDivision):transitive)

(role *hasGoal* (:domain owl#Thing) (:range Goal) :transitive)

(role *Goal_goalOf* (:domain Goal) (:range owl#Thing) (:inverse hasGoal) :transitive)

(role *Goal_hasDecomposition* (:domain Goal) (:range SubGoal) :transitive)

(role *Goal_decompositionOf* (:domain SubGoal) (:range Goal)
(:inverse Goal_hasDecomposition) :transitive)

(role *Goal_achievedAt* (:domain Goal) (:range Time))

(role *Goal_dependsOn* (:domain Goal) (:range Goal))

(role *Role_hasActivity* (:domain Role) (:range Activity))

(role *Role_hasAuthority* (:domain Role) (:range Authority))

(role *Role_requiresSkill* (:domain Role) (:range Skill))

(role *Role_hasPolicy* (:domain Role) (:range Constraint))

(role *Role_hasResource* (:domain Role) (:range Resource) (:inverse Resource_hasRole))

(role *Role_superiorOf* (:domain Role) (:range Role) :transitive)

(role *Role_subordicateOf* (:domain Role) (:range Role)
(:inverse Role_superiorOf) :transitive)

(role *Role_generalizedRoleOf* (:domain Role) (:range Role) :transitive)

(role *Role_specializedRoleOf* (:domain Role) (:range Role)
(:inverse Role_generalizedRoleOf) :transitive)

(role *Agent_memberOf* (:domain Agent) (:range OrganizationOrTeam)
(:inverse OOT_hasMember))

(role *Agent_performs* (:domain Agent) (:range Activity))

(role *Agent_plays* (:domain Agent) (:range Role))

(role *Agent_hasAuthority* (:domain Agent) (:range Authority))

(role *Agent_hasCommunicationLink* (:domain Agent) (:range Agent)
(:inverse Agent_hasCommunicationLink) :symmetric)

(role *Agent_hasHomeDiv* (:domain Agent) (:range Division))

(role *Agent_isCommittedTo* (:domain Agent) (:range Goal))

(role *OOT_hasMember* (:domain OrganizationOrTeam) (:range Agent))

(role *CommLink_hasSendingAgent* (:domain CommunicationLink) (:range Agent))

(role *CommLink_hasSendingRole* (:domain CommunicationLink) (:range Agent))

(role *CommLink_hasReceivingAgent* (:domain CommunicationLink) (:range Agent))

(role *CommLink_hasReceivingRole* (:domain CommunicationLink) (:range Agent))

(role *CommLink_hasInterest* (:domain CommunicationLink) (:range Information))

(role *CommLink_willVolounteer* (:domain CommunicationLink) (:range Information))

(role *CWA_hasSupervisor* (:domain CommunicationWithAuthority) (:range Agent))

(role *CWA_hasSupervisee* (:domain CommunicationWithAuthority) (:range Agent))

(role *CWA_hasResource* (:domain CommunicationWithAuthority) (:range Resource))

(role *CWA_hasEmpowerment* (:domain CommunicationWithAuthority)
(:range Empowerment))

(role *CWA_hasRole* (:domain CommunicationWithAuthority) (:range Role))

(rule ‘hasGoal(?org,?g1) ∧ Goal_hasDecomposition(?g1,?g2) => hasGoal(?org,?g2)’)

(rule ‘CommLink_hasSendingAgent(?cl,?oa) ∧ CommLink_hasSendingRole(?cl,?r)
=> Agent_plays(?oa,?r)’)

(rule ‘CommLink_hasReceivingAgent(?cl,?oa) ∧ CommLink_hasReceivingRole(?cl,?r)
=> Agent_plays(?oa,?r)’)

(fol 'Role_subordicateOf(?r1,?r2) <=> (!A ?ath Role_hasAuthority(?r1,?ath)
=> Role_hasAuthority(?r2,?ath)) ∧ !E ?ath2 Role_hasAuthority(?r2,?ath2)
∧ ∼Role_hasAuthority(?r1,?ath2)')

(fol 'Goal_dependsOn(?g1,?g2) <=> (!A ?t Goal_achievedAt(?g1,?t)
=> Goal_achievedAt(?g2,t))')

(fol 'Role(?r) ∧ hasGoal(?r,?g) => (!E ?g2 Organization(?o)
∧ hasGoal(?o,?g2) ∧ Goal_hasDecomposition(?g,?g2))')

(fol 'Division(?d) ∧ hasGoal(?d,?g) ∧ Division_subDivisionOf(?d,?pd)
=> hasGoal(?pd,?g) | (!E ?g2 hasGoal(?pd,?g2)
∧ Goal_decompositionOf(?g,?g2))')

(fol 'Agent(?oa) ∧ hasGoal(?oa,?g) <=> !E ?r Agent_plays(?oa,?r) ∧ hasGoal(?r,?g)')

(fol 'Agent(?oa) ∧ Agent_hasAuthority(?oa,?a) <=> !E ?r Agent_plays(?oa,?r)
∧ Role_hasAuthority(?r,?a)')

(fol '!E ?d Agent_memberOf(?oa,?d)')

(fol 'Agent_hasHomeDiv(?oa,?d1) ∧ Agent_hasHomeDiv(?oa,?d2) ∧ ?d1=?d2')

(fol '!E ?oa1 !E ?oa2 ?oa1∼=?oa2
∧ Agent_memberOf(?oa1,?tm) ∧ Agent_memberOf(?oa2,?tm)')

(fol 'Agent(?oa) ∧ Agent_plays(?oa,?r) => (hasGoal(?r,?g)
=> Agent_isCommittedTo(?oa,?g))')

(fol 'CWA_hasResource(?cwa,?rs) => (!E ?r CWA_hasSupervisor(?cwa,?oa)
∧ Agent_plays(?oa,?r) ∧ Role_hasResource(?r,?rs))')

B.5 Retail Ontology Axioms

B.5.1 Concepts

(concept *Product*)

(concept *SKU* (:subClassOf xsd#Integer))

(concept *Category*)

(concept *ProductCategory* (:subClassOf Category))

(concept *ProductProductMap*)

(concept *CrossSellProductProductMap* (:and ProductProductMap
(:has-value PPM_hasMappingType cross-sell)))

(concept *Inventory*)

(concept *InventoryItem*)

(concept *BasicFinancialConcept*)

(concept *Currency* (:subClassOf BasicFinancialConcept))

(concept *Price* (:subClassOf BasicFinancialConcept))

(concept *DiscountPrice* (:subClassOf Price))

(concept *CostOfGoodsSold* (:subClassOf BasicFinancialConcept))

(concept *Tax* (:subClassOf BasicFinancialConcept))

(concept *Margin* (:subClassOf BasicFinancialConcept))

(concept *Zone*)

(concept *GPMGroup*)

(concept *HighestGPMGroup* (:subClassOf GPMGroup))

(concept *LowestGPMGroup* (:subClassOf GPMGroup))

(concept *Document* (:subClassOf Resource))

(concept *Catalog* (:subClassOf Resource))

(concept *OnlineCatalog* (:and Catalog VirtualResource))

(concept *SupplierCatalog* (:or Catalog OnlineCatalog))

(concept *Company* (:subClassOf Organization))

(concept *Store*)

(concept *WebStore*)

(concept *Website*)

(concept *WebsiteHost*)

(concept *URI* (:subClassOf InformationResource))

(concept *URL* (:subClassOf InformationResource))

(concept *VirtualResource* (:subClassOf Resource))

(concept *WebPage* (:subClassOf VirtualResource))

(concept *Session*)

(concept *PageView*)

(concept *Supplier* (:subClassOf Organization))

(concept *InformationResource* (:subClassOf Resource))

(concept *AvailabilityInfo* (:subClassOf InformationResource))

(concept *AvailabilityInfoLine*)

(concept *Order*)

(concept *PurchaseOrder* (:subClassOf Order))

(concept *CustomerOrder* (:subClassOf Order))

(concept *OrderLineItem*)

(concept *Total* (:subClassOf Price))

(concept *Payment*)

(concept *PaymentProvider*)

(concept *Shipment*)

(concept *ShippingCourier*)

(concept *ShippingInfo* (:subClassOf InformationResource))

(concept *ContactInfo* (:subClassOf InformationResource))

(concept *Address* (:subClassOf InformationResource))

(concept *IPAddress* (:subClassOf InformationResource))

(concept *PostalCode* (:subClassOf InformationResource))

(concept *Email* (:subClassOf InformationResource))

(concept *Telephone* (:subClassOf InformationResource))

(concept *Person*)

(concept *CustomerAccount*)

(concept *RelationshipManagementConcept*)

(concept *CustomerValue* (:subClassOf RelationshipManagementConcept))

(concept *AverageOrderValue* (:subClassOf RelationshipManagementConcept))

(concept *ConversionRate* (:subClassOf RelationshipManagementConcept))

(concept *CustomerLifetimeValue* (:subClassOf RelationshipManagementConcept))

(concept *Segment*)

(concept *RFMSegment* (:subClassOf Segment))

(concept *CVSegment* (:subClassOf Segment))

(concept *CLVSegment* (:subClassOf Segment))

(concept *ProductSegment* (:subClassOf Segment))

(concept *DemographicSegment* (:subClassOf Segment))

(concept *GenderSegment* (:subClassOf DemographicSegment))

(concept *AgeSegment* (:subClassOf DemographicSegment))

(concept *Promotion*)

(concept *Coupon* (:subClassOf Promotion))

(concept *ComplexFinancialConcept*)

(concept *Revenue* (:subClassOf ComplexFinancialConcept))

(concept *SalesRevenue* (:subClassOf Revenue))

(concept *AcquisitionRevenue* (:subClassOf Revenue))

(concept *Profit* (:subClassOf ComplexFinancialConcept))

(concept *GrossProfit* (:subClassOf Profit))

(concept *AcquisitionProfit* (:subClassOf Profit))

(concept *RetentionRate* (:subClassOf ComplexFinancialConcept))

(concept *ProductProfitSegment*)

(concept *MostProfitableProductSegment* (:subClassOf ProductProfitSegment))

B.5.2 Roles

(role *P_hasSKU* (:domain Product) (:range SKU))

(role *P_productOf* (:domain Product) (:range ProductCategory)
(:inverse PCat_hasProduct))

(role *P_hasCatalog* (:domain Product) (:range Catalog))

(role *P_hasCurrentPrice* (:domain Product) (:range Price))

(role *P_hasDiscountPrice* (:domain Product) (:range Price))

(role *P_hasBasePrice* (:domain Product) (:range Price))

(role *P_hasCostOfGoodsSold* (:domain Product) (:range xsd#Float))

(role *P_isReturnable* (:domain Product) (:range (:one-of true false)))

(role *P_hasGPMargin* (:domain Product) (:range xsd#Float))

(role *P_belongsToGPMGroup* (:domain Product) (:range GPMGroup))

(role *P_belongsToSegment* (:domain Product) (:range ProductProfitSegment))

(role *P_isCrossListedWith* (:domain Product) (:range Product))

(role *PCat_hasProduct* (:domain ProductCategory) (:range Product)
(:inverse P_productOf))

(role *PCat_childCategoryOf* (:domain ProductCategory) (:range ProductCategory)
(:inverse PCat_hasChildCategory) :transitive)

(role *PCat_hasChildCategory* (:domain ProductCategory) (:range ProductCategory)
(:inverse PCat_childCategoryOf) :transitive)

(role *PPMap_hasProduct* (:domain ProductProductMap) (:range Product))

(role *PPMap_hasParentProduct* (:domain ProductProductMap) (:range Product))

(role *PPMap_hasMappingType* (:domain ProductProductMap) (:range xsd#String))

(role *Inv_hasItem* (:domain Inventory) (:range InventoryItem))

(role *InvItem_hasProduct* (:domain InventoryItem) (:range InventoryProduct))

(role *InvItem_hasQuantity* (:domain InventoryItem) (:range Quantity))

(role *InvItem_hasQuantitySold* (:domain InventoryItem) (:range Quantity))

(role *InvItem_hasQuantityDeferred* (:domain InventoryItem) (:range Quantity))

(role *InvItem_hasQuantityBuffer* (:domain InventoryItem) (:range Quantity))

(role *InvItem_hasQuantitySoldTotal* (:domain InventoryItem) (:range Quantity))

(role *InvItem_hasReorderLevel* (:domain InventoryItem) (:range xsd#Integer))

(role *InvItem_hasReorderDatetime* (:domain InventoryItem) (:range Datetime))

(role *InvItem_reachedReorderLevel* (:domain InventoryItem) (:range (:one-of true false)))

(role *Price_hasCurrency* (:domain Price) (:range Currency))

(role *Price_hasValue* (:domain Price) (:range xsd#Float))

(role *DPrice_hasDiscountPriceStartDate* (:domain DiscountPrice) (:range Datetime))

(role *DPrice_hasDiscountPriceEndDate* (:domain DiscountPrice) (:range Datetime))

(role *Tax_hasZone* (:domain Tax) (:range Zone))

(role *Tax_hasRate* (:domain Tax) (:range xsd#Float))

(role *Tax_hasDescription* (:domain Tax) (:range xsd#String))

(role *Zone_hasParentZone* (:domain Zone) (:range Zone) :transitive)

(role *Zone_hasPostalCode* (:domain Zone) (:range PostalCode))

(role *Zone_hasLevel* (:domain Zone) (:range xsd#Integer))

(role *Zone_hasCountry* (:domain Zone) (:range xsd#String))

(role *GPM_hasMin* (:domain GPMGroup) (:range xsd#Float))

(role *GPM_hasMax* (:domain GPMGroup) (:range xsd#Float))

(role *Ctg_ContainsProduct* (:domain Catalog) (:range Product))

(role *Ctg_hasCurrency* (:domain Catalog) (:range Currency))

(role *OnlineCtg_hasWebsite* (:domain OnlineCatalog) (:range Website))

(role *Com_hasCatalog* (:domain Company) (:range Catalog))

(role *Com_parentCompanyOf* (:domain Company) (:range Company)
(:inverse Com_hasParentCompany) :transitive)

(role *Com_hasParentCompany* (:domain Company) (:range Company)
(:inverse Com_parentCompanyOf) :transitive)
(role *Com_hasStore* (:domain Company) (:range Store) (:inverse Store_storeOf))
(role *Com_hasWebstore* (:domain Company) (:range Webstore)
(:inverse WStore_webstoreOf))
(role *Com_hasSupplier* (:domain Company) (:range Supplier)
(:inverse Supp_supplies) :transitive)
(role *Com_pays* (:domain Company) (:range Payment))
(role *Store_hasLocation* (:domain Store) (:range Location))
(role *Store_storeOf* (:domain Store) (:range Company) (:inverse Com_hasStore))
(role *WStore_webstoreOf* (:domain Webstore) (:range Company)
(:inverse Com_hasWebstore))
(role *WStore_hasWebsite* (:domain Webstore) (:range Website)
(:inverse Website_websiteOf))
(role *Website_isAvailableOnline* (:domain Website) (:range (:one-of true false)))
(role *Website_hasWebsiteHost* (:domain Website) WebsiteHost))
(role *Website_websiteOf* (:domain Website) (:range Webstore)
(:inverse WStore_hasWebsite))
(role *WHost_hasHostName* (:domain WebsiteHost) (:range xsd#String))
(role *WHost_hasSecureHostName* (:domain WebsiteHost) (:range xsd#String))
(role *WHost_hasURI* (:domain WebsiteHost) (:range URI))
(role *WPage_belongsTo* (:domain WebPage) (:range Website))
(role *WPage_parentWebPageOf* (:domain WebPage) (:range WebPage)
(:inverse WPage_hasParentWebPage) :transitive)
(role *WPage_hasParentWebPage* (:domain WebPage) (:range WebPage)
(:inverse WPage_parentWebPageOf) :transitive)
(role *Session_hasEntryTime* (:domain Session) (:range Datetime))

(role *Session_hasClientIPAddress* (:domain Session) (:range IPAddress))

(role *Session_hasEntryURL* (:domain Session) (:range URL))

(role *Session_hasExitURL* (:domain Session) (:range URL))

(role *Session_hasIdentifier* (:domain Session) (:range xsd#Integer))

(role *Session_purchasedIn* (:domain Session) (:range (:one-of true false)))

(role *PView_hasWebPage* (:domain PageView) (:range WebPage))

(role *PView_hasHitDatetime* (:domain PageView) (:range Datetime))

(role *PView_hasSession* (:domain PageView) (:range Session))

(role *PView_hasCustomerOrder* (:domain PageView) (:range CustomerOrder))

(role *Supp_supplies* (:domain Supplier) (:range Company)
(:inverse Com_hasSupplier) :transitive)

(role *Supp_hasSupplierCatalog* (:domain Supplier) (:range SupplierCatalog))

(role *SCtg_hasAvailabilityInfo* (:domain SupplierCatalog) (:range AvailabilityInfo))

(role *AInfo_hasInfoLine* (:domain AvailabilityInfo) (:range AvailabilityInfoLine))

(role *AInfoLine_hasProduct* (:domain AvailabilityInfoLine) (:range Product))

(role *AInfoLine_hasQuantity* (:domain AvailabilityInfoLine) (:range Quantity))

(role *AInfoLine_hasMinOrderSize* (:domain AvailabilityInfoLine) (:range xsd#Integer))

(role *AInfoLine_includesDelivery* (:domain AvailabilityInfoLine)
(:range (:one-of true false)))

(role *AInfoLine_hasOrderProcessTime* (:domain AvailabilityInfoLine) (:range Datetime))

(role *Ord_hasOrderLineItem* (:domain Order) (:range OrderLineItem))

(role *Ord_hasCreateDatetime* (:domain Order) (:range Datetime))

(role *Ord_hasShipment* (:domain Order) (:range Shipment) (:inverse Sh_hasOrder))

(role *Ord_hasStatus* (:domain Order) (:range (:one-of updating executing held canceled
executed completed)))

(role *Ord_hasProduct* (:domain Order) (:range Product))

(role *Ord_hasBaseTotal* (:domain Order) (:range Total))

(role *Ord_hasTotal* (:domain Order) (:range Total))

(role *OrdLine_hasProduct* (:domain OrderLineItem) (:range Product))

(role *OrdLine_hasQuantity* (:domain OrderLineItem) (:range Quantity))

(role *OrdLine_hasLineBaseTotal* (:domain OrderLineItem) (:range Total))

(role *OrdLine_hasLineTaxedTotal* (:domain OrderLineItem) (:range Total))

(role *OrdLine_hasTax* (:domain OrderLineItem) (:range Tax))

(role *OrdLine_hasLineCost* (:domain OrderLineItem) (:range Total))

(role *OrdLine_* (:domain OrderLineItem) (:range))

(role *COrd_hasCustomerAccount* (:domain CustomerOrder) (:range CustomerAccount)
(:inverse CA_hasCustomerOrder))

(role *COrd_hasCoupon* (:domain CustomerOrder) (:range Coupon))

(role *COrd_hasTotalAfterCoupon* (:domain CustomerOrder) (:range Total))

(role *COrd_hasProductCost* (:domain CustomerOrder) (:range Total))

(role *COrd_containsCrossSellProducts* (:domain CustomerOrder)
(:range (:one-of true false)))

(role *POrd_hasSupplier* (:domain PurchaseOrder) (:range Supplier))

(role *POrd_hasOrderLeadTime* (:domain PurchaseOrder) (:range Datetime))

(role *Pay_hasAmount* (:domain Payment) (:range Total))

(role *Pay_hasStatus* (:domain Payment) (:range (:one-of new executing canceled
rejected accepted)))

(role *Pay_paymentFor* (:domain Payment) (:range Order))

(role *Pay_hasType* (:domain Payment) (:range (:one-of credit-card debit-card cash
store-credit cheque certified-cheque money-order gift-certificate)))

(role *Pay_hasProvider* (:domain Payment) (:range PaymentProvider))

(role *PayProvider_supportsType* (:domain PaymentProvider) (:range (:one-of credit-card
debit-card store-credit cheque certified-cheque money-order)))

(role *Sh_hasOrder* (:domain Shipment) (:range Order) (:inverse Ord_hasShipment))

(role *Sh_hasAddress* (:domain Shipment) (:range Address))

(role *Sh_hasExpectedShipDate* (:domain Shipment) (:range Datetime))

(role *Sh_hasRequestedDeliveryDate* (:domain Shipment) (:range Datetime))

(role *Sh_hasShippingZone* (:domain Shipment) (:range Zone))

(role *Sh_hasTotalQuantity* (:domain Shipment) (:range Quantity))

(role *Sh_hasHandlingTotal* (:domain Shipment) (:range Total))

(role *Sh_hasShippingTotal* (:domain Shipment) (:range Total))

(role *Sh_hasHandlingTax* (:domain Shipment) (:range Tax))

(role *Sh_hasShippingTax* (:domain Shipment) (:range Tax))

(role *Sh_hasShippingCourier* (:domain Shipment) (:range ShippingCourier))

(role *Sh_hasShippingTotal* (:domain Shipment) (:range Total))

(role *SCour_hasContactInfo* (:domain ShippingCourier) (:range ContactInfo))

(role *SCour_hasShippingInfo* (:domain ShippingCourier) (:range ShippingInfo))

(role *SInfo_hasCurrency* (:domain ShippingInfo) (:range Currency))

(role *SInfo_hasMaxDays* (:domain ShippingInfo) (:range xsd#Integer))

(role *SInfo_hasMinDays* (:domain ShippingInfo) (:range xsd#Integer))

(role *SInfo_hasZone* (:domain ShippingInfo) (:range Zone))

(role *SInfo_hasPerItemHandlingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerItemMinHandlingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerItemShippingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerItemMinShippingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerShipmentHandlingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerShipmentMinHandlingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerShipmentShippingPrice* (:domain ShippingInfo) (:range Price))

(role *SInfo_hasPerShipmentMinShippingPrice* (:domain ShippingInfo) (:range Price))

(role *CInfo_hasContactPersonName* (:domain ContactInfo) (:range xsd#String))

(role *CInfo_hasAddress* (:domain ContactInfo) (:range Address))

(role *CInfo_hasEmail* (:domain ContactInfo) (:range Email))

(role *CInfo_hasTelephoneNumber* (:domain ContactInfo) (:range Telephone))

(role *Per_hasFirstName* (:domain Person) (:range))

(role *Per_hasLastName* (:domain Person) (:range))

(role *Per_hasGender* (:domain Person) (:range))

(role *Per_hasAddress* (:domain Person) (:range))

(role *Per_hasEmail* (:domain Person) (:range))

(role *Per_hasBirthdate* (:domain Person) (:range))

(role *Per_hasTelephoneNumber* (:domain Person) (:range))

(role *CA_hasPerson* (:domain CustomerAccount) (:range Person))

(role *CA_hasUsername* (:domain CustomerAccount) (:range xsd#String))

(role *CA_hasPassword* (:domain CustomerAccount) (:range xsd#String))

(role *CA_hasCustomerOrder* (:domain CustomerAccount) (:range CustomerOrder)
(:inverse COrd_hasCustomerAccount))

(role *CA_purchasedSKU* (:domain CustomerAccount) (:range SKU))

(role *CA_pays* (:domain CustomerAccount) (:range Payment))

(role *CA_hasTotalAmountPurchased* (:domain CustomerAccount) (:range Total))

(role *CA_hasPurchaseFrequency* (:domain CustomerAccount) (:range xsd#Integer))

(role *CA_hasAverageOrderValue* (:domain CustomerAccount) (:range AverageOrderValue))

(role *CA_hasTimeOfLastPurchase* (:domain CustomerAccount) (:range Datetime))

(role *CA_hasCustomerValue* (:domain CustomerAccount) (:range CustomerValue)
(:inverse CV_valueOf))

(role *CA_belongsToSegment* (:domain CustomerAccount) (:range Segment))

(role *purchasedBySameCustomer* (:domain SKU) (:range SKU))

(role *CV_valueOf* (:domain CustomerValue) (:range CustomerAccount)
(:inverse CV_hasCustomerValue))

(role *CV_hasStartDatetime* (:domain CustomerValue) (:range Datetime))

(role *CV_hasEndDatetime* (:domain CustomerValue) (:range Datetime))

(role *CV_hasAmountPurchased* (:domain CustomerValue) (:range Total))

(role *CV_hasCost* (:domain CustomerValue) (:range Total))

(role *CV_hasValue* (:domain CustomerValue) (:range xsd#Float))

(role *CRate_hasStartDatetime* (:domain ConversionRate) (:range Datetime))

(role *CRate_hasEndDatetime* (:domain ConversionRate) (:range Datetime))

(role *CRate_hasNumberOfVisitors* (:domain ConversionRate) (:range xsd#Integer))

(role *CRate_hasNumberOfBuyers* (:domain ConversionRate) (:range xsd#Integer))

(role *CRate_hasValue* (:domain ConversionRate) (:range xsd#Float))

(role *CLV_hasStartDatetime* (:domain CustomerLifetimeValue) (:range Datetime))

(role *CLV_hasEndDatetime* (:domain CustomerLifetimeValue) (:range Datetime))

(role *CLV_hasRecency* (:domain CustomerLifetimeValue) (:range Datetime))

(role *CLV_hasFrequency* (:domain CustomerLifetimeValue) (:range xsd#Integer))

(role *CLV_hasMonetaryValue* (:domain CustomerLifetimeValue) (:range Price))

(role *CLV_hasExpectedTransactionValue* (:domain CustomerLifetimeValue) (:range Price))

(role *CLV_hasValue* (:domain CustomerLifetimeValue) (:range xsd#Float))

(role *RFMSeg_hasRecencyMin* (:domain RFMSegment) (:range xsd#Integer))

(role *RFMSeg_hasRecencyMax* (:domain RFMSegment) (:range xsd#Integer))

(role *RFMSeg_hasFrequency* (:domain RFMSegment) (:range xsd#Integer))

(role *RFMSeg_hasMonetaryValueMin* (:domain RFMSegment) (:range xsd#Float))

(role *RFMSeg_hasMonetaryValueMax* (:domain RFMSegment) (:range xsd#Float))

(role *CVSeg_hasCVMin* (:domain CVSegment) (:range xsd#Float))

(role *CVSeg_hasCVMax* (:domain CVSegment) (:range xsd#Float))

(role *CLVSeg_hasCLVMin* (:domain CLVSegment) (:range xsd#Float))

(role *CLVSeg_hasCLVMax* (:domain CLVSegment) (:range xsd#Float))

(role *ProductSeg_hasProductCategory* (:domain ProductSegment)

(:range ProductCategory))

(role *AgeSeg_hasAMin* (:domain AgeSegment) (:range xsd#Integer))

(role *AgeSeg_hasAMax* (:domain AgeSegment) (:range xsd#Integer))

(role *GSSeg_hasPostalCode* (:domain GeographicSegment) (:range PostalCode))

(role *GSSeg_hasGSRevenue* (:domain GeographicSegment) (:range Revenue))

(role *Prom_hasGoal* (:domain Promotion) (:range Goal))

(role *Prom_hasActivity* (:domain Promotion) (:range Activity))

(role *Prom_hasStartDatetime* (:domain Promotion) (:range Datetime))

(role *Prom_hasEndDatetime* (:domain Promotion) (:range Datetime))

(role *Cp_hasCurrency* (:domain Coupon) (:range Currency))

(role *Cp_hasStartDatetime* (:domain Coupon) (:range Datetime))

(role *Cp_hasEndDatetime* (:domain Coupon) (:range Datetime))

(role *Cp_hasValueAssigned* (:domain Coupon) (:range Price))

(role *Cp_hasMaximumValue* (:domain Coupon) (:range Price))

(role *Cp_hasMaximumUsage* (:domain Coupon) (:range xsd#Integer))

(role *Cp_timesUsed* (:domain Coupon) (:range xsd#Integer))

(role *Cp_couponOf* (:domain Coupon) (:range Webstore))

(role *Rev_hasStartDatetime* (:domain Revenue) (:range Datetime))

(role *Rev_hasEndDatetime* (:domain Revenue) (:range Datetime))

(role *Rev_hasAmount* (:domain Revenue) (:range xsd#Float))

(role *ARev_hasNumberOfCustomers* (:domain AcquisitionRevenue) (:range xsd#Integer))

(role *GProfit_hasRevenue* (:domain GrossProfit) (:range Revenue))

(role *GProfit_hasCost* (:domain GrossProfit) (:range Total))

(role *GProfit_hasAmount* (:domain GrossProfit) (:range xsd#Float))

(role *AProfit_hasAcquisitionActivity* (:domain AcquisitionProfit) (:range Activity))

(role *AProfit_hasAcquisitionRevenue* (:domain AcquisitionProfit) (:range Revenue))

(role *AProfit_hasStartDatetime* (:domain AcquisitionProfit) (:range Datetime))

(role *AProfit_hasEndDatetime* (:domain AcquisitionProfit) (:range Datetime))

(role *AProfit_hasAcquisitionCost* (:domain AcquisitionProfit) (:range Total))

(role *AProfit_hasAmount* (:domain AcquisitionProfit) (:range xsd#Float))

(role *RRate_hasStartDatetime* (:domain RetentionRate) (:range Datetime))

(role *RRate_hasEndDatetime* (:domain RetentionRate) (:range Datetime))

(role *RRate_hasNumberOfRetainedCustomers* (:domain RetentionRate)
(:range xsd#Integer))

(role *RRate_hasNumberOfCustomers* (:domain RetentionRate) (:range xsd#Integer))

(role *RRate_hasValue* (:domain RetentionRate) (:range xsd#Float))

(role *PPSeg_hasStartDatetime* (:domain ProductProfitSegment) (:range Datetime))

(role *PPSeg_hasEndDatetime* (:domain ProductProfitSegment) (:range Datetime))

(role *PPSeg_hasPPMin* (:domain ProductProfitSegment) (:range xsd#Float))

(role *PPSeg_hasPPMax* (:domain ProductProfitSegment) (:range xsd#Float))

B.5.3 FOL Statements

(rule 'InventoryItem(?inv_item) ∧ hasQuantity(?inv_item, ?q) ∧
hasReorderLevel(?inv_item, ?rl) ∧ ?q = ?rl =>
reachedReorderLevel(?inv_item, true)')

(rule 'Product(?p) ∧ DiscountPrice(?dpr) ∧ hasDiscountPrice(?p, ?dpr) ∧
hasDiscountPriceStartDate(?dpr, ?s_date) ∧
hasDiscountPriceEndDate(?dpr, ?e_date) ∧ CurrentDatetime(?cur_dt) ∧
before(?start_date, ?cur_dt) ∧ before(?cur_date, ?end_dt) =>
hasCurrentPrice(?p, ?dpr)')

(rule 'Product(?p) ∧ hasGPMargin(?p, ?m) ∧ GPMGroup(?gpm_group) ∧
hasGPMMin(?gpm_group, ?gpm_min) ∧ hasGPMMax(?gpm_group, ?gpm_max)
∧ ?m ≥ ?gpm_min ∧ ?m ≤ ?gpm_max =>
belongsToGPMGroup(?p, ?gpm_group)')

(rule ‘PageView(?pview) ∧ Session(?s) ∧ CustomerOrder(?co) ∧ hasSession(?pview, ?s)
∧ hasCustomerOrder(?pview, ?co) ∧ hasStatus(?co, completed) =>
purchasedIn(?s, true)’)

(rule ‘OrderLineItem(?oli) ∧ Product(?p) ∧ hasProduct(?oli, ?p) ∧
hasCostOfGoodsSold(?p, ?c) ∧ hasQuantity(?oli, ?q) =>
hasLineCost(?oli, ?c ∗ ?q)’)

(rule ‘OrderLineItem(?oli) ∧ Product(?p) ∧ hasProduct(?oli, ?p) ∧ Price(?pr) ∧
hasPrice(?p, ?pr) ∧ hasValue(?pr, ?val) ∧ hasQuantity(?oli, ?q) =>
hasLineBaseTotal(?oli, ?val ∗ ?q)’)

(rule ‘CustomerOrder(?co) ∧ Product(?p) ∧ CustomerAccount(?ca) ∧ SKU(?x) ∧
hasSKU(?p, ?x)∧ hasProduct(?co, ?p) ∧ hasCustomerOrder(?ca, ?co) =>
purchasedSKU(?ca, ?x)’)

(rule ‘CustomerValue(?c_value) ∧ hasAmountPurchased(?c_value, ?cv_pur) ∧
hasCost(?c_value, ?cv_c) => hasValue(?c_value, ?cv_pur - ?cv_c)’)

(rule ‘ConversionRate(?con_rate) ∧ hasNumberOfVisitors(?con_rate, ?num_vis) ∧
hasNumberOfBuyers(?con_rate, ?num_cus) =>
hasValue(?con_rate, ?num_cus / ?num_vis)’)

(rule ‘CustomerAccount(?ca) ∧ hasTimeOfLastPurchase(?ca, ?t) ∧
hasAverageOrderValue(?ca, ?aov) ∧ hasPurchaseFrequency(?ca, ?freq) ∧
RFMSegment(?rfm_seg) ∧ hasRecencyMin(?rfm_seg, ?rmin_value) ∧
hasRecencyMax(?rfm_seg, ?rmax_value) ∧ hasFrequency(?rfm_seg, ?f) ∧
hasMMin(?rfm_seg, ?mmin_value) ∧ hasMMax(?rfm_seg, ?mmax_value) ∧
?t >= ?rmin_value ∧ ?t <= ?rmax_value ∧ ?aov >= ?mmin_value ∧
?aov <= ?mmax_value ∧ ?freq = ?f => belongsToSegment(?ca, ?rfm_seg)’)

(rule ‘CustomerAccount(?ca) ∧ CustomerValue(?cv) ∧ hasCustomerValue(?ca, ?cv) ∧
hasValue(?cv, ?cv_value) ∧ CVSegment(?cv_seg) ∧
hasCVMin(?cv_seg, ?min_value) ∧ hasCVMax(?cv_seg, ?max_value) ∧

?cv_value >= ?min_value ∧ ?cv_value < ?max_value =>

belongsToSegment(?ca, ?cv_seg)')

(?rule 'CustomerAccount(?ca) ∧ CustomerLifetimeValue(?clv) ∧ hasCLV(?ca, ?clv) ∧

hasValue(?clv, ?clv_value) ∧ CLVSegment(?clv_seg) ∧

hasCLVMin(?clv_seg, ?min_value) ∧ hasCLVMax(?clv_seg, ?max_value) ∧

?clv_value >= ?min_value ∧ ?clv_value < ?max_value =>

belongsToSegment(?ca, ?clv_seg)')

(?rule 'CustomerAccount(?ca) ∧ CustomerOrder(?po) ∧

hasCustomerOrder(?ca,?po) ∧ Product(?p) ∧ hasProduct(?po,?p) ∧

ProductCategory(?pcat) ∧ productOf(?p, ?pcat) ∧ ProductSegment(?pro_seg) ∧

hasProductCategory(?pro_seg, ?pcat) => belongsToSegment(?ca, ?pro_seg)')

(?rule 'CustomerAccount(?ca) ∧ Person(?per) ∧

hasPerson(?ca, ?per) ∧ Address(?adr) ∧ hasAddress(?per, ?adr) ∧

hasPostalCode(?adr, ?p_code) ∧ GeographicSegment(?geo_seg) ∧

hasGSPostalCode(?geo_seg, ?p_code) => belongsToSegment(?ca, ?geo_seg)')

(rule 'GrossProfit(?gp) ∧ Revenue(?rev) ∧ hasRevenue(?gp, ?rev)∧

hasAmount(?rev, ?revenue) ∧ hasCost(?gp, ?gp_cost) =>

hasAmount(?gp, ?revenue − ?gp_cost)')

(rule 'AcquisitionProfit(?gp) ∧ AcquisitionRevenue(?rev) ∧

hasAcquisitionRevenue(?ap, ?rev) ∧ hasNumberOfCustomers(?rev, ?num) ∧

hasAmount(?rev, ?x) ∧ hasAcquisitionCost(?ap, ?cost) <=>

hasAmount(?ap, ?x / ?num ∗ ?x − ?cost / ?x)')

(rule 'RetentionRate(?ret_rate) ∧ hasNumberOfRetainedCustomers(?ret_rate, ?num_ret)

hasNumberOfCustomers(?ret_rate, ?num_cus) <=>

hasValue(?ret_rate, ?num_ret / ?num_cus)')

(rule 'CustomerOrder(?po) ∧ Product(?p1) ∧ Product(?p2) ∧

hasProduct(?co, ?p1) ∧ hasProduct(?co, ?p2) ∧ isCrossListedWith(?p1, ?p2) =>

containsCrossSellProducts(?co, true)')

(fol '!A ?p Product(?p) => (!E ?x SKU(?x) ∧ hasSKU(?p,?x) ∧ ∼!E ?y SKU(?y) ∧
?x ∼= ?y ∧ hasSKU(?p,?y))')
(fol '!A ?ppm ProductProductMap(?ppm) => !E ?par_p !E ?p !E ?m_type
Product(?par_p) ∧ hasParentProduct(?ppm, ?par_p) ∧ Product(?p) ∧
hasProduct(?ppm, ?p) ∧ hasMappingType(?ppm, ?m_type)')
(fol '!A ?inv_item !A ?qs !A ?qd !E ?tot InventoryItem(?inv_item) ∧
hasQuantitySoldTotal(?inv_item, ?tot) <=> hasQuantitySold(?inv_item, ?qs) ∧
hasQuantityDeferred(?inv_item, ?qd) ∧ tot = qs + qdc')
(fol '!A ?p Product(?p) => !E ?pr Price(?pr) ∧ hasCurrentPrice(?p, ?pr)')
(fol 'Product(?p) ∧ DiscountPrice(?dpr) ∧ hasDiscountPrice(?p, ?dpr) ∧
hasDiscountPriceStartDate(?dpr, ?s_date) ∧
hasDiscountPriceEndDate(?dpr, ?e_date) ∧ currentDatetime(?cur_dt) ∧
(before(?cur_date, ?s_date) | before(?end_date, ?cur_dt)) ∧
hasBasePrice(?p, ?bpr) => hasCurrentPrice(?p, ?bpr)')
(fol '!A ?p Product(?p) => !E ?cogs hasCostOfGoodsSold(?p, ?cogs)')
(fol '!A ?p !E ?m Product(?p) ∧ hasGPMargin(?p, ?m) <=> !A ?pr !A ?c !A ?val
Price(?pr) ∧ hasCurrentPrice(?p, ?pr) ∧ hasValue(?pr, ?val) ∧
hasCostOfGoodsSold(?p, ?c) ∧ ?m = 1 - ?c / ?val')
(fol '!A ?cg Catalog(?cg) => !E ?p Product(?p) ∧ containsProduct(?cg, ?p)')
(fol '!A ?cg !E ?cur Catalog(?cg) ∧ Currency(?cur) ∧ hasCurrency(?cg, ?cur) <=>
!A ?p !A ?pr Product(?p) ∧ containsProduct(?cg, ?p) ∧ Price(?pr) ∧
hasCurrentPrice(?p, ?pr) ∧ hasCurrency(?pr, ?cur)')
(fol '!A ?c Company(?c) => !E ?cg Catalog(?cg) ∧ hasCatalog(?c, ?cg)')
(fol '!A ?s Store(?s) => !E ?l Location(?l) ∧ hasLocation(?s, ?l)')
(fol '!A ?com Company(?com) => !E ?x (Store(?x) ∧ hasStore(?com, ?x)) |

(Webstore(?x) ∧ hasWebstore(?com, ?x))')

(fol '!A ?wsite Website(?wsite) ∧ isAvailableOnline(?wsite, true) =>
!E ?host WebsiteHost(?host) ∧ hasWebsiteHost(?wsite, ?host)')

(fol '!A ?wpage WebPage(?wpage) => !E ?wsite Website(?wsite) ∧
belongsTo(?wpage, ?wsite)')

(fol '!A ?s Supplier(?s) => !E ?cg SupplierCatalog(?cg) ∧ hasSupplierCatalog(?s, ?cg)')

(fol '!A ?supp !A ?scg !A ?p !A ?pr !A ?cur !A ?val !A ?ai !A ?ai_line Supplier(?supp) ∧
SupplierCatalog(?scg) ∧ hasSupplierCatalog(?supp, ?scg) ∧
AvailabilityInfo(?ai) ∧ hasAvailabilityInfo(?scg, ?ai) ∧
AvailabilityInfoLine(?ai_line) ∧ hasInfoLine(?ai, ?ai_line) ∧
hasProduct(?ai_line, ?p) ∧ hasCurrentPrice(?p, ?pr) ∧ hasCurrency(?p, ?cur) ∧
hasValue(?p, ?val) ∧ ∼(!E ?supp1 !E ?scg1 !E ?p1 !E ?pr1 !E ?cur1 !E ?val1
!E ?ai1 !E ?ai_line1 Supplier(?supp1) ∧ ?supp ∼= ?supp1 ∧
SupplierCatalog(?scg1) ∧ hasSupplierCatalog(?supp1, ?scg1) ∧
AvailabilityInfo(?ai1) ∧ hasAvailabilityInfo(?scg1, ?ai1) ∧
AvailabilityInfoLine(?ai_line1) ∧ hasInfoLine(?ai1, ?ai_line1) ∧
hasProduct(?ai_line1, ?p1) ∧ ?p = ?p1 ∧ hasCurrentPrice(?p1, ?pr1) ∧
hasCurrency(?p1, ?cur1) ∧ ?cur = ?cur1 ∧ hasValue(?p1, ?val1) ∧ ?val1 < ?val)
=> deliversBestValue(?ail, true) ')

(fol '!A ?ord !A ?pol !A ?p Order(?ord) ∧ Product(?p) ∧ hasProduct(?ord, ?p) <=>
OrderLineItem(?pol) ∧ hasOrderLineItem(?ord, ?pol) ∧ hasProduct(?pol, ?p)')

(fol '!A ?co CustomerOrder(?co) => !E ?ca CustomerAccount(?ca) ∧
hasCustomerAccount(?co, ?ca) ∧ ∼(!E ?ca1 CustomerAccount(?ca1) ∧
hasCustomerAccount(?co, ?ca1) ∧ ?ca ∼= ?ca1)')

(fol '!A ?po PurchaseOrder(?po) => !E ?su Supplier(?su) ∧ hasSupplier(?po, ?su) ∧
∼(!E ?su1 Supplier(?su1) ∧ hasSupplier(?po, ?su1) ∧ ?su ∼= ?su1)')

(fol '!A ?pay Payment(?pay) => !E ?ord Order(?ord) ∧ paymentFor(?pay, ?ord) ∧

∼(!E ?ord1 Order(?ord1) ∧ paymentFor(?pay, ?ord1) ∧ ?ord ∼= ?ord1)')

(fol '!A ?pay Payment(?pay) => !E ?t hasType(?pay, ?t) ')

(fol '!A ?pay Payment(?pay) ∧ ∼hasType(?pay, cash) => !E ?pro
PaymentProvider(?pro) ∧ hasProvider(?pay, ?pro)')

(fol '!A ?x Webstore(x) => !E ?y PaymentProvider(?y) ∧ hasPaymentProvider(?x, ?y)')

(fol '!A ?sh Shipment(?sh) => !E ?ord Order(?x) ∧ hasOrder(?sh, ?ord)')

(fol '!A ?sh Shipment(?sh) => !E ?adr Address(?adr) ∧ hasAddress(?sh, ?adr)')

(fol '!A ?ord !A ?sh !A ?b_tot !A ?s_tot Order(?ord) ∧ Shipment(?sh) ∧
hasShipment(?ord, ?sh) ∧ hasBaseTotal(?ord, ?b_tot) ∧
hasShippingTotal(?sh, ?s_tot) => hasTotal(?ord, ?b_tot + ?s_tot)')

(fol '!A ?ca CustomerAccount(?ca) => !E ?p Person(?p) ∧ hasPerson(?ca, ?p) ∧
∼(!E ?p2 Person(?p2) ∧ hasPerson(?ca, ?p2) ∧ ?p ∼= ?p2)')

(fol '!A ?x !A ?y SKU(?x) ∧ SKU(?y) ∧ purchasedBySameCustomer(?x, ?y) <=>
!E ?ca CustomerAccount(?ca) ∧ purchasedSKU(?ca, ?x) ∧
purchasedSKU(?ca, ?y) ∧ ?x ∼= ?y')

(fol '!A ?clv !E ?rec CustomerLifetimeValue(?clv) ∧ hasRecency(?clv, ?rec) <=>
!A ?ca !A ?r CustomerAccount(?ca) ∧ lifetimeValueOf(?clv, ?ca) ∧
hasTimeOfLastPurchase(?ca, ?r) ∧ ?rec = ?r')

(fol '!A ?clv !E ?freq CustomerLifetimeValue(?clv) ∧ hasFrequency(?clv, ?freq) <=>
!A ?ca !A ?f CustomerAccount(?ca) ∧ lifetimeValueOf(?clv, ?ca) ∧
hasPurchaseFrequency(?ca, ?f) ∧ ?freq = ?f')

(fol '!A ?clv !E ?mon CustomerLifetimeValue(?clv) ∧ hasMonetaryValue(?clv, ?mon)
<=>!A ?ca !A ?aov CustomerAccount(?ca) ∧ lifetimeValueOf(?clv, ?ca) ∧
hasAverageOrderValue(?ca, ?aov) ∧ ?mon = ?aov')

(fol '!A ?ap !E ?c AcquisitionProfit(?ap) ∧ hasAcquisitionCost(?ap, ?c) <=>
!A ?acq !A ?acp AcquisitionActivity(?acq) ∧ hasAcquisitionActivity(?ap, ?act)
∧ ActivityCostPoint(?acp) ∧ hasActivity(?acp, ?acq) ∧ hasCost(?acp, ?c)')

(fol ‘!A ?ppm !A ?par_p !A ?p CrossSellProductProductMap(?ppm) ∧ Product(?par_p)
∧ Product(?p) ∧ hasParentProduct(?ppm, ?par_p) ∧ hasProduct(?ppm, ?p)
=> (isCrossListedWith(?par_p, ?p) ∧ isCrossListedWith(?p, ?par_p))’)

Bibliography

[1] Arnott, D.; Pervan, G. (2005), "A Critical Analysis of Decision Support Systems Research", *Journal of Information Technology*, Vol. 20/2, 67-87.

[2] Azvine, B.; Cui, Z.; Nauch, D. D. (2005), "Towards Real-Time Business Intelligence", *BT Technology Journal*, Springer, Vol. 23/3, 214-225.

[3] Bechhofer, S.; Horrocks, I.; Turi, D. (2005), "The OWL Instance Store: System Description", in *Proceedings of the 20th International Conference on Automated Deduction (CADE-20), Lecture Notes in Artificial Intelligence*, Springer, 177-181.

[4] Bechhofer, S.; van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.L.; Patel-Schneider, P.F.; Stein, L.A. (2004), "OWL Web Ontology Language 1.0 Reference", W3C Recommendation, Available at http://www.w3.org/TR/owl-ref/.

[5] Berger, P. D., Nasr, N. I. (1998), "Customer Lifetime Value: Marketing Models and Applications, *Journal of Interactive Marketing*, Vol. 12, 17-30.

[6] Chung, W.; Chen, H.; Nunamaker, J.F. (2003), "Business Intelligence Explorer: A knowledge map framework for discovering business intelligence on the Web", in *Proceedings of the 36th Hawaii International Conference on System Sciences*, Hawaii, Big Island, USA, IEEE Computer Society.

[7] Cody, W. F.; Kreulen, J. T.; Krishna, V.; Spangler, W. S. (2002), "The Integration of Business Intelligence adn Knowledge Management", *IBM Systems Journal*, Vol. 41/4, 697-713.

[8] Dibb, S.; Simkin, L.; Pride, B.; Ferrell, O. C (2001), *Marketing: Concepts and Strategies, Fourth Edition*, Houghton Mifflin.

[9] Dobler, Donald W; Burt, David N (1996). *Purchasing and Supply Management, Text and Cases*, Sixth Edition, p70, Singapore: McGraw-Hill.

[10] Druzdzel, M. J.; Flynn, R. R. (2002), "Decision Support Systems", *Encyclopedia of Library and Information Science, Second Edition*, New York: Marcel Dekker, Inc.

[11] El-Ghalayini, H.; Odeh, M.; McClatchey, R. (2006), "Engineering Conceptual Data Models from Domain Ontologies: A Critical Evaluation", in *Proceedings of the fourth International Multiconference on Computer Science and Information Technology*, Vol. 3, 42-53.

[12] Etzion, O., Fisher, A., Wasserkrug, S. (2005), "e-CLV: A Modeling Approach for Customer Lifetime Evaluation in e-Commerce Domains, with an Application and Case Study for Online Auction, *Information Systems Frontiers*, Springer Science and Business Media, 7 (4/5), 421-434.

[13] Fadel, F. G. (1994), *A Resource Ontology for Enterprise Modelling*, M.A.Sc. Thesis, Enterprise Integration Laboratory, University of Toronto.

[14] Fadel, G. F.; Fox, M. S.; Gruninger, M. (1994), "A Generic Enterprise Resource Ontolog", in *Proceedings of the third IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 86-92, Morgantown, West Virginia.

[15] Fader, P. S., Hardie, B. G. S., Lee, K. L. (2005), "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis", *Journal of Marketing Research*, Vol. XLII, 415-430.

[16] Fader, P. S., Hardie, B. G. S., Lee, K. L. (2005b), "A Note on Implementing the Pareto/NBD Model in MATLAB", available at http://brucehardie.com/notes/008/.

[17] Fox, M.S.; Barbuceanu, M.; Gruninger, M.; Lin, J. (1997), "An Organization Ontology for Enterprise Modelling", in *Simulating Organizations: Computational Models of Institutions and Groups*, ed. M. Prietula, K. Carley, and L. Gasser, 131-152. Menlo Park, CA: AAAI/MIT Press.

[18] Fox, M. S.; Chionglo, J. F.; Fadel, F.; Gruninger, M. (1994), *TOVE Manual Second Version*, University of Toronto.

[19] Fox, M. S.; Gruninger, M. (1994), "Ontologies for Enterprise Integration", *Proceedings of the 2nd Conference on Cooperative Information Systems*, Toronto, Ontario.

[20] Fox, M. S.; Gruninger, M. (1998), "Enterprise Modelling", *AI Magazine*, 109-121, AAAI Press.

[21] Fridman, N.; Hafner, C. D. (1997), "The State of the Art in Ontology Design: a Survey and Comparative Review", *AI Magazine*, Vol. 18/3, 53-74.

[22] Gibson, M.; Arnott, D.; Jagielska, I. (2004), "Evaluating the Intangible Benefits of Business Intelligence: Review & Research Agenda", in *Proceedings of the 2004 IFIP International Conference on Decision Support Systems (DSS2004): Decision Support in an Uncertain and Complex World*, Prato, Italy, 295-305.

[23] Golfarelli, M.; Rizzi, S.; Cella, I. (2004), "Beyond Data Warehousing: What's Next in Business Intelligence?", in *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, Washington, DC, USA.

[24] Gruber, T. R. (1993), "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", in *KSL 93-04*, Knowledge Systems Laboratory, Stanford University.

[25] Gruninger, M.; Atefi, K.; Fox, M. S. (2000), "Ontologies to Support Process Integration in Enterprise Engineering", *Computational and Mathematical Organization Theory.*

[26] Haarslev, V.; Moller, R. (2003), *RACER User's Guide and Reference Manual, Version 1.7.7.*.

[27] Harrison, A.; White, A. (2006), "Intelligent distribution and logistics", *IEE Proceedings Intelligent Transport Systems*, Vol. 153/2, 167-180.

[28] Hill, J.; Scott, T. (2004), "A Consideration of the Roles of Business Intelligence and e-Business in Management and Marketing Decision Making in Kowledge-Based and High-Tech Start-Ups", *Qualitative Market Research: An International Journal*, Vol. 7/1, 48-57.

[29] Hoekstra, J. C., Huizingh, E. (1999), "The Lifetime Value Concept in Customers-Based Marketing, *Journal of Marketing Focused Management*, 257274.

[30] Horrocks, I. (1998), "Usinge an Expressive Description Logic: FaCT or Fiction?", in *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, ed. L. Schubert, and S. Shapiro, 636-647. San Francisco, CA: Morgan Kaufmann.

[31] Horrocks, I. (2007), "Semantic Web: The Story So Far", in *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, Vol. 225, ACM Press, New York, NY, 120-125.

[32] Horrocks, I.; Voronokov, A. (2006), "Reasoning Support for Expressive Ontology Languages Using a Theorem Prover", in *Proceedings of FoIKS'2006*, 201-218.

[33] Hughes, A. M. (1997), "Customer Retention: Integrating Lifetime Value into Marketing Strategies", *Journal of Database Marketing* Vol. 5/2, 171-178.

[34] Ustadt, U.; Motik, B.; Sattler, U. (2004), "Reducing SHIQ Description Logic to Disjunctive Datalog Programs", in *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, 152-162.

[35] Karjaluoto, H.; Leppaniemi, M.; Salo, J. (2004), "The Role of Mobile Marketing in Companies Promotion Mix: Empirical Evidence From Finland", *Journal of International Business and Economics*, Vol. 2/1, 111-116.

[36] Kim, H.; Fox, M. S; Bilgic, T. (1999), "An Ontology for Quality Management: Enabling Quality Problem Identification and Tracing", *BT Technology Journal*, Vol. 17/4.

[37] Lemon, K. N., Mark, T. (2006), "Customer Lifetime Value as the Basis of Customer Segmentation: Issues and Challenges, *Journal of Relationship Marketing*, 5 (2/3), 55-69.

[38] Lin, J.; Fox, M. S., Bilgic, T. (1996), "A Requirements Ontology for Concurrent Engineering", *Concurrent Engineering: Research and Applications*, 279-291.

[39] Lonnqvist, A.; Pirttimaki, V. (2006), "The Measurement of Business Intelligence", *Information Systems Management*, Vol. 23/1, 32-40.

[40] Luhn, H. P., (1958), "A Business Intelligence System", *IBM Journal of Research and Development*, Vol. 2/4, 314-319.

[41] Marshall, B.; McDonald, D.; Chen, H.; Chung, W. (2004), "EBizPort: Collecting and Analyzing Business Intelligence Information", *Journal of the American Society for Information Science and Technology*, 55(10):873891

[42] McDonald, K.; Wilmsmeier, A.; Dixon, D. C.; Inmon W.H. (2002), *Mastering the SAP Business Information Warehouse*, John Wiley & Sons, p.4.

[43] Mezher, T.; Abdul-Malak, M. A.; Maarouf, B. (1998), "Embedding Critics in Decision-Making Environments to Reduce Human Errors", *Knowledge-Based Systems*, Vol. 11, Elsevier Science.

[44] Motik, B.; Sattler, U. (2006), "A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes", in *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2006)*, Phnom Penh, Cambodia.

[45] Negash, S.; Gray, P. (2003), "Business Intelligence", *Ninth Americas Conference on Information Systems*, Tampa, Florida.

[46] Olszak, C. M.; Ziemba, E. (2006), "Business Intelligence Systems in the Holistic Infrastructure Development Supporting Decision Making in Organizations", *Interdisciplinary Journal of Information, Knowledge, and Management*, Informing Science Institute. Santa Rosa, USA.

[47] Ou, L.; Peng H. (2006), "Knowledge and Process Based Decision Support in Business Intelligence System", in *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 2*, IEEE Computer Society, Washington, DC, USA, 780-786.

[48] Petrini, M.; Pozzebon, M. (2003), "The Value of Business Intelligence in the Context of Developing Countries", in *Proceedings of the 11th European Conference on Information Systems, ECIS 2003*, Naples, Italy.

[49] Pfeifer, P. E., Haskins, M. E., Conroy, R. M. (2005), "Customer Lifetime Value, Customer Profitability, and the Treatment of Acquisition Spending, *Journal of Managerial Issues*, 17 (1), 11-25.

[50] Power, D. J. (2007), "A Brief History of Decision Support Systems - version 4.0", *DSSResources.COM*, World Wide Web, http://DSSResources.COM/history/dsshistory.html.

[51] McClintic Marion, A.; Gale, T. (2001), "Promotion", *Encyclopedia of Business and Finance*. eNotes.com. 2006. 25 Apr, 2007 ¡http://business.enotes.com/business-finance-encyclopedia/promotion¿

[52] McCune, W. (2003), "OTTER 3.3 Reference Manual", Technical Memorandum 263, Argonne National Laboratory.

[53] Riazanov, A.; Voronokov, A. (2002), "The Design and Implementation of Vampire", *AI Communications* Vol. 15(2-3), 91-110.

[54] Salonen, J.; Pirttimaki, V. (2005), "Outsourcing a Business Intelligence Function", *Frontiers of e-Business Research 2005*, 661-675.

[55] Sanner, S., McIlraith, S. (2006), "An Ordered Theory Resolution Calculus for Hybrid Reasoning in First-order Extensions of Description Logic", in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, ed. Patrick Doherty, John Mylopoulos, and Christopher Welty, 32-41. Menlo Park, CA: AAAI Press.

[56] Schmittlein, D. C., Morrison, D. G., Colombo, R. (1987), "Counting Your Customers: Who They Are and What Will They Do Next? *Management Science*, 33 (January), 124.

[57] Sell, D.; Cabral, L.; Motta, E.; Domingue, J.; Pacheco, R. (2005), "Adding Semantics to Business Intelligence", in *Proceedings of the 16th International Workshop on Database and Expert Systems Applications*, 543-547.

[58] Sell, D.; Cabral, L.; Motta, E.; Domingue, J.; Pacheco, R. (2005), "A Semantic Web Based Architecture for Analytical Tools", in *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, 347-354.

[59] Sirin, E.; Parsia, B.; Cuenca Grau, B.; Kalyanpur, A.; Katz, Y. (2007), "Pellet: A Practical OWL-DL Reasoner", *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 5/2,

[60] Soper, D. S. (2005), "A Framework for Automated Web Business Intelligence Systems", in *Proceedings of the 38th Hawaii International Conference on System Sciences*, Hawaii, Big Island, USA, IEEE Computer Society.

[61] Spence, M. (1976), "Product Selection, Fixed Costs, and Monopolistic Competition", *The Review of Economic Studies*, Vol. 43/2, 217-235.

[62] Tham, D.; Fox, M. S.; Gruninger, M. (1994), "A Cost Ontology for Enterprise Modelling", in *Proceedings of the Third Workshop on Enabling Technologies - Infrastructures for Collaborative Enterprises*, 111-117, West Virginia University.

[63] Tsarkov, D.; Riazanov, A.; Bechhofer, S.; and Horrocks, I. (2004), "Using Vampire to reason with OWL", in *Proceedings of the 3rd International Semantic Web Conference, volume 3298 of LectureNotes in Computer Science*, 471485.

[64] Uschold, M.; Gruninger, M. (2004), "Ontologies and Semantics for Seamless Connectivity", *SIGMOD Record* Vol. 33/4, 58-64.

[65] van Harmelen, F.; Patel-Schneider, P.F.; Horrocks, I. (2001), "Reference Description of the DAML+OIL Ontology Markup Language.

[66] Wedel, M.; Kamakura, W. (2000), *Market Segmentation: Conceptual and Methodological Foundations, Second Edition*, 3. Norwell, MA, Kluwer Academic Publishers.

[67] Zhang, Z. (2005), *Ontology Query Languages for the Semantic Web: A Performance Evaluation*, M.S. Thesis, University of Georgia.