Vaasan yliopisto
UNIVERSITY OF VAASA

TEEMU MÄENPÄÄ

Utilization of adjacency model in graph analysis

ACTA WASAENSIA 335

COMPUTER SCIENCE 14

Reviewers

Professor Ismo Hakala
Kokkola University Consortium Chydenius
Talonpojankatu 2 B
P.O.Box 567
FI-67701 Kokkola

Professor Eljas Soisalon-Soininen
Aalto University
Department of Computer Science and Engineering
P.O. Box 15400
FI-00076 Aalto, Finland

Julkaisija	**Julkaisupäivämäärä**	
Vaasan yliopisto	Syyskuu 2015	
Tekijä(t) Teemu Mäenpää	**Julkaisun tyyppi** Monografia	
	Julkaisusarjan nimi, osan numero Acta Wasaensia, 335	
Yhteystiedot Vaasan yliopisto Teknillinen tiedekunta Tieto- ja tietoliikennetekniikka PL 700 65101 Vaasa	**ISBN** 978-952-476-642-5 (painettu) 978-952-476-643-2 (verkkojulkaisu)	
	ISSN 0355-2667 (Acta Wasaensia 335, painettu) 2323-9123 (Acta Wasaensia 335, verkkojulkaisu) 1455-7339 (Acta Wasaensia. Tietotekniikka 14, painettu) 2342-0693 (Acta Wasaensia. Tietotekniikka 14, verkkojulkaisu)	
	Sivumäärä 163	**Kieli** englanti

Julkaisun nimike
Vierekkyysmallin hyödyntäminen graafien analysoinnissa

Tiivistelmä

Puolirakenteinen tieto oli laajasti tutkittu aihe vuosituhannen vaihteen molemmin puolin. Monimutkaiset ja jäsentymättömät tietorakenteet ovat nousseet tutkimusteemaksi tekniikoiden kuten web 2.0:n ja big datan yleistyessä. Tässä tutkimuksessa etsitään keinoja, joiden avulla monimutkaiset tietorakenteet voidaan esittää relaatiotietokannan muodossa ja siten saada niiden tietosisällöt helpommin hyödynnettävään muotoon.

Vierekkyysmalli on malli tiedon esittämiseen. Mallin keskeinen komponentti on vierekkyysrelaatiojärjestelmä, joka on matemaattinen kuvaus joukkoihin kuuluvien elementtien vierekkyyksistä. Vierekkyysrelaatiojärjestelmä voidaan kuvata graafina, jossa elementit esitetään graafin solmuina ja solmuja yhdistävä sivu kuvaa elementtien välistä vierekkyyttä. Tämän tutkimuksen tavoitteena on kehittää menetelmä tällaisten graafien käsittelyyn ja analysointiin.

Vierekkyyden käsitettä on sovellettu muun muassa geometrisen mallintamisen tutkimuksessa. Vierekkyysmalli mahdollistaa vierekkyyden käsitteen hyödyntämisen relaatiotietokantojen mallintamisessa. Aiemmat tutkimukset ovat osoittaneet relaatiomallin ja vierekkyysmallin käsitteelliset samankaltaisuudet.

Tässä tutkimuksessa hyödynnetään graafiteorian peruskäsitteistöä vierekkyysmallin ja relaatiomallin samankaltaisuuksien analysoinnissa. Analysoitavat tietokannat esitetään vierekkyysrelaatiojärjestelmänä ja graafeina. Analyysitulosten perusteella graafeista tunnistetaan relaatiotietokantojen keskeisiä elementtejä kuvaavat osat ja niiden ominaisuudet.

Tutkimuksessa määritetään kriteeristö tietokannan osien tunnistamiseen sitä esittävästä graafista. Kriteeristö mahdollistaa graafin muuntamisen takaisin tietokannaksi. Lisäksi tutkimuksessa pohditaan monitavoiteoptimoinnin hyödyntämistä graafien analysoinnissa.

Asiasanat
tiedon mallinnus, relaatiomalli, vierekkyysmalli, graafianalyysi

Publisher	**Date of publication**	
University of Vaasa	September 2015	
Author(s) Teemu Mäenpää	**Type of publication** Monograph	
	Name and number of series Acta Wasaensia, 335	
Contact information University of Vaasa Faculty of Technology Computer Science P.O. Box 700 FI-65101 Vaasa Finland	**ISBN** 978-952-476-642-5 (painettu) 978-952-476-643-2 (verkkojulkaisu)	
	ISSN 0355-2667 (Acta Wasaensia 335, print) 2323-9123 (Acta Wasaensia 335, online) 1455-7339 (Acta Wasaensia. Computer Science 14, print) 2342-0693 (Acta Wasaensia. Computer Science 14, online)	
	Number of pages 163	**Language** English
Title of publication Utilization of adjacency model in graph analysis		

Abstract

Data without fixed schema was widely studied topic in late 90s and in early 2000s. The introduction of concepts such as web 2.0 and big data foregrounded complex and loosely structured data as a research theme. This study examines ways to process complex data structures to make them operable for relational databases.

Adjacency model is a model representing data. Key concept of the model is adjacency relation system, which is mathematical representation of adjacency between elements belonging to certain sets. Adjacency relation system can be visualized as a graph. The goal of this study is to develop framework for graph analysis. Early applications for the concept of adjacency focused on the boundary structures. The adjacency model aimed to generalize the concept of adjacency. Studies have shown that adjacency model and relational model have conceptual similarities. Based on the similarities a method for modeling relational databases with adjacency model was developed.

This study utilizes the graph theory in the analysis of the similarities between adjacency model and relational model. Databases were modeled with adjacency model and represented as adjacency relation system based graphs. The graphs were analyzed and typical properties for the database elements in the adjacency model were defined. This study focused on adjacency model and the expression of the concepts of relational model in the model. This study provided properties for database elements in adjacency model and a method for converting adjacency model into database was given. Also, the usage of multi-objective optimization in manipulation of graphs was discussed.

Keywords

data modeling, relational model, adjacency model, graph analysis

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisor Professor Merja Wanne for her guidance and support throughout the writing process. I also would like to thank Dr. Jari Töyli for his insightful comments regarding the topic of the thesis. I would like to thank Professor Ismo Hakala and Professor Eljas Soisalon-Soininen for reviewing this thesis and their valuable feedback and remarks about the work.

I would like to express my appreciation to my past and present colleagues, Simo, Vesa, Hannu, Johanna, Jouni, Laura, Teemu S., and many others for support, insights, fruitful discussions, and encouragement.

Finally, I would like to thank my wife Suvi. Her encouragement, support, and love were the solid foundation that made this dissertation possible.

Table of Contents

Figures

Tables

1 INTRODUCTION

Data models for unstructured data were vastly and intensively studied topic from mid-90s to early 2000s. Unstructured or semistructured data is data that is not raw data, but it is not strictly typed, and it does not have an accurate schema (Abiteboul 1997, Suciu 1998). The traditional data models do not handle unstructured data properly, so the research efforts strived to integrate such data and to develop efficient structures and models for unstructured data. Typically such data is depicted with graph-based representation methods such as the Object Exchange Model (Papakonstantinou, Garcia-Molina & Widom 1995) and deterministic data model (Buneman, Davidson & Suciu 1995).

Early applications of the concept of adjacency focused on the boundary structures (Weiler 1985; Ni & Bloor 1994). Wanne (1998) introduced a structure called Adjacency Relation Systems (ARS). It is a mathematical structure for data representation, and it also provides graph visualizations of data. ARS is based on the adjacency of elements that are members of certain sets of entity types. Wanne (1998) applied ARS in the field of planar graphs. ARS was further developed into Adjacency Model (AM). It is a framework for such concepts as adjacency relation system, adjacency defining type and relation combination (Wanne and Linna 1999; Töyli 2002). Furthermore, research works done with the AM has produced methods and models, which made it possible to utilize ARS in modeling of unstructured data.

After the introduction of AM, both ARS and AM have been utilized in various application domains. In Töyli, Linna and Wanne (2002a and 2002b) and Töyli (2002 and 2006) adjacency relation systems were applied in modeling relational data and semistructured data. Adjacency relation systems have also been used in modeling wind power production and distributed energy production (Heikkinen & Linna 2004; Nyrhilä, Mäenpää, Linna & Antila 2005a and 2005b). The adjacency relation systems have also been employed in modeling semantic link networks (Mäenpää & Nyrhilä 2013a and 2013b). Besides, the modeling usage, the concept of adjacency has been utilized in algorithm development for applications in computational geometry (Zadravec, Brodnik, Mannila, Wanne & Zalik 2008).

After the emergence of Web 2.0 and especially Semantic Web, the graph-based models and structures, such as Resource Description Framework (Manola & Miller 2004) and Semantic Link Network (Zhuge 2004), have become more widely utilized. Furthermore, the elements of graph-structures are more interconnected, and thus the structures are more complex. In order to make data more actionable and usable, for example, systems based on the relational data

model, there should be methods and frameworks for analyzing graph-structures and data. Based on the result generated by the framework researchers and developers could determine if a given graph is a suitable foundation, for example, for a database.

The methods for graph processing discussed in this work continue the research work started by Wanne (1999) and Töyli (2002 and 2006). Töyli's framework for modeling relational data with adjacency model, and as well Wanne's work both suggest that elements of the adjacency relation system correspond to the elements of relational database. To widen the utilization potential of the adjacency model to analyzing graphs, there should be precise criteria and systematical method for recognizing the database elements from the ARS based graph. The previous studies have provided a method for modeling relational databases with adjacency model, this study adds the method for reconstructing the database from the ARS based graph into the Adjacency Model framework.

This work aims to broaden the utilization areas of the Adjacency Model. The work focuses primarily on the possibilities that are tied to the similarities and analogies between adjacency model and relational model. This work seeks answers to the following questions.

1. What are the requirements under which adjacency model can be used for analyzing, structuring and representing heterogeneous data?
2. What are the solutions that enhance the data processing capabilities of the adjacency model?
3. What are the means to improve the adjacency model's capabilities to convey information about the relationships between data elements in a given structure?

For data gathering and analyzing purposes a framework that combines the principles of the adjacency model, the relational model, graph theory and multi-objective optimization (Price, Storm & Lampinen 2005) were developed. In this work will be given identification criteria for recognizing the essential elements of the relational database from an adjacency relation system based graph. Moreover, the reconstruction method from an adjacency relation system to the database is given. In addition, some adjustments to definitions within the adjacency model are proposed.

2 RESEARCH PROCESS

The research overall approach to this study combines the features of theory creating case research and design research. Järvinen (2012: 66) states that the theory creating research is suitable for situations where the research is fairly new, and there is little knowledge about the phenomenon. The research process utilizes the guidelines (figure 1) for the case study provided by Pan and Tan (2011: 164).

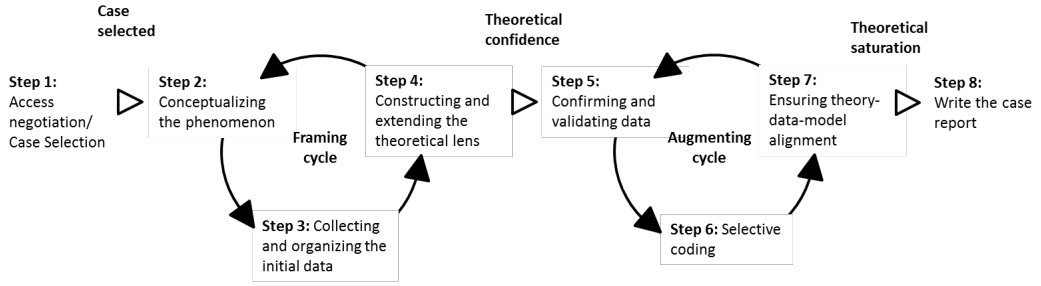


Figure 1. A structured-pragmatic-situational approach for case studies (Pan & Tan 2011: 164).

Pan and Tan (2011: 165) emphasize that their framework has a practical viewpoint. So, at the start of the research process it is typically more pragmatic to select an interesting case, and then define the research problem and questions. Pan and Tan state that usually, the selected case clarifies and guides the research problem definition. After the case is selected, the framework proceeds to the framing cycle, which consists of steps 2, 3 and 4. The second step builds a mental concept of the case phenomenon by reviewing the theoretical background of the phenomenon.

The purpose of the third step is to validate and modify the mental concept. The actions of this step include data collecting and organizing it into themes by comparisons, examinations, and categorizations. The fourth step builds the preliminary theory. Pan and Tan suggest that the key theories of the research should be recognized and broken into components in order to construct a theoretical lens. The lens guides the data collection and analysis. The framing cycle iterations end when the theoretical confidence is reached. Theoretical confidence means that the essence of the phenomenon is captured, and the contributions of the study can be derived from the data. (Pan & Tan 2011:167-169; Strauss & Corbin 1998: 101; Walsham 2006: 325.)

The augmenting cycle begins with confirming and validating data (Step 5). During this step, theoretical lens is transformed into the theory, and the validity of the data is ensured. Pan and Tan (2011: 169) underline that the sufficiency of

data and validity of data must be ensured. The transformation of the theoretical lens to theory is continued in the selective coding step. Selective coding corroborates and integrates the conceptual categories of the theoretical lens with the case data. The last step of the augmenting cycle ensures the theory-data-model alignment by recursive iterations between existing theories, data, and the emergent model. This phase answers to the following questions (Pan & Tan 2011: 171).

(1) Do the existing theories explain the data?

(2) Does the data support the new model and

(3) Do the existing theories support the new model?

The augmenting cycle ends when the theoretical saturation is gained. The decision, whether the study is mature enough depends on the researcher. The final step of the SPS –frameworks writing the case report, i.e. the key phases and finding of the study are reported in a systematical way. (Pan & Tan 2011: 169-172; Strauss & Corbin 1998: 143.)

2.1 Design research approach

Design research is a feasible approach for research, which aims to build innovations by utilizing the results of basic research (Järvinen 2012: 98). Van Aken (2004: 224) states that the goal of design research is to construct new solutions and artefacts based prior knowledge. Design research aims to improve the performance of existing entities.

Simon (1996: 129) has characterized the design research process with the "The Generate/Test Cycle" shown in figure 2.

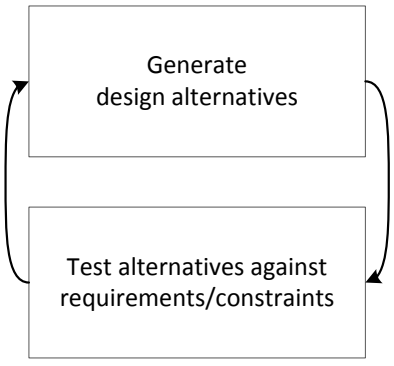


Figure 2. The Generate/Test Cycle (Simon 1996: 129; Hevner, March, Parka & Ram: 89).

March and Smith (1995: 255) have provided a research framework for design research. The framework describes the key research outputs and research activities (Table 1) of design research.

Table 1. The design research framework.

		Research activities			
		Build	Evaluate	Theorize	Justify
Research outputs	Constructs				
	Model				
	Method				
	Instantiation				

Constructs give the vocabulary and concepts for problem description within the domain, and they provide solutions for the problems. Relationships between constructs are represented by the model. Model can be seen as statement for problems and solutions. Respectively, the method provides guidelines or algorithms for performing the given task. An instantiation according to March and Smith is the realization of an artifact in its context.

March and Smith (2004: 258) state that the design research builds innovation for a particular purpose and evaluates how the innovation works. At theorizing phase of the framework, the features of the artifact and its interaction with the environment are analyzed. Generalizations and theories need to be justified, which means that evidence must be gathered to test and explain the theories. (March and Smith 2004: 259.)

The relationships between the selected research approaches and the conducted study are discussed in the next section.

2.2 An overview of the study

This section discusses confluences between the research process of this study and the SPS –framework for theory creating case studies. Because the main research efforts of this study focus on Adjacency Model (AM) and the framework of concepts build around it, this study can be characterized as case research. In addition, this study combines features of design research and theory-creating research.

The concepts and techniques of AM and multi-objective optimizations are utilized in the construction of tools for data collecting and analysis. Study also produces additions to the AM.

In the first step, the case was selected, and a preliminary draft of the research problem was given. A data representation method called Adjacency Model was selected as the primary case for this study. The preliminary research task at this point was to develop and enhance the adjacency model and its data representation capabilities.

In the second step base theories and supporting theories of the study were reviewed. Furthermore, the research problem clarified, and it focused on the analogies between the relational model and adjacency model. Is it possible to reconstruct a database modeled as an adjacency relation system into a relational database and if so can the finding be applied at general level in the analysis of the different graph-based data representations?

At the third step data collecting and analyzing criteria and tools were crafted. The crafting process can be seen as design research since the tools and criteria utilize existing theories such as graph theory, data models, multi-objective optimization and adjacency model. Crafting process conforms Simon's generate/test cycle (see figure 2).

At the fourth step data was collected and analyzed by the tools and criteria defined in the previous step. The tools and criteria were refined in this step according to comparisons between collected data and the key theories.

After iterations of the framing cycle, the elements of relational database can be identified from an ARS- based graph representing the database. Moreover, the graph representing a database can be reconstructed as a relational database.

The augmenting cycle begins with testing the results of the framing cycle. Test cases were simplified databases as well as real-life databases. Test material contained databases that represented different features of the relational model such as dependencies and simple constraints.

In the sixth step, the results of the fifth step were evaluated against the theories of the study. In addition, a method for ARS to database reconstruction was defined. Furthermore, some additions were made to the definitions of the adjacency model framework.

At the seventh step was noticed that the case data supported the analogies between adjacency model and relational model. Case data also shows that the con-

cepts of the relational model can be identified from the ARS based graph representing the database.

Finally, by combining the key features of relational model and adjacency model with the results of this research, a database can be modeled as ARS. Furthermore, the database can be reconstructed from the ARS. Moreover, the general applicability of the method introduced in this research was discussed in this step. Aim of the eighth step is to represent and report the research process and its findings.

3 GRAPH THEORY CONCEPTS

In this section the essential graph theory concepts that are utilized in this work are introduced. Graph is ordered pair (V, E), where V represents the finite nonempty set of elements called vertices. E consists of two-element subsets of V, such that $\{p, q\} \in E$, and $p, q \in V$. The elements of E are called edges. Edges join the elements of V, denoted by $p \overset{e}{-} q$. It is said that the vertices are incident to the edge and the elements joined by an edge are said to be adjacent or neighbors. Thus, an edge e can be denoted by $e = pq$. The graph G is said to be complete if each vertex is adjacent to each other i.e. each vertex in G are connected by an edge with each other vertex of G. When a certain part of a graph is examined, it is useful to construct a subgraph of given graph. A subgraph G' of G is defined as follows $G' = (V', E')$ where $V' \subset V$ and $E' \subset E$. (Savolainen 1978: 20; Foulds 1992:9-12; Jungnickel 2013: 2-3.)

Example 1. Let G be a graph (V, E) where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{v_1v_2, v_2v_3, v_3v_4, v_4v_1, v_2v_4\}$. A subgraph G' contains $V' = \{v_1, v_2, v_3\}$ and $E' = \{v_1v_2, v_2v_3\}$. Graphs G and G' are depicted in figures 3 and 4.

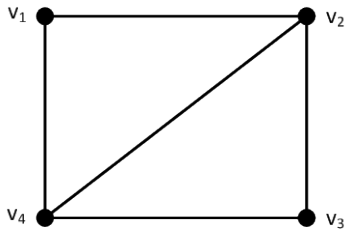


Figure 3. Graph G.

Figure 4. Graph G' represtents a subgraph of G.

3.1 Connectivity, walks, and paths

A graph is said to be connected if it has a connection, called walk, between any two vertices. Walk in a graph G is a sequence of edges and vertices. For example $(e_1, \dots, e_n)$ is an edge sequence in a graph. If there exist vertices $v_0, \dots, v_n$ such that $e_i = v_{i-1}v_i$ where $i = 1, \dots, n$, the edge sequence is called a walk. The walk is closed if $v_0 = v_n$. A closed walk containing at least three different vertices is called a cycle. If all the edges in a walk are unique the walk is called a trail. If the walk does not visit a vertex that it has visited before it is called path (or self-avoiding path). The length of a walk is the number of edges in it. Walk W from v_0 to v_n can be notated as follows $\langle v_0, v_1, \dots, v_n \rangle$. (Foulds 1992:17-18; Jungnickel 2013: 5-6; Goodaire & Parmenter 2006: 304-306; Newman 2010: 136.)

For example a walk from v_1 to v_4 in figure 3 can be formed as $\langle v_1, v_2, v_3, v_2, v_4 \rangle$ and respectively a path from v_1 to v_4 can be stated as $\langle v_1, v_2, v_4 \rangle$.

3.2 Components in undirected graph

A graph can be divided into subgraphs A and B. If there does not exist a connection between vertices in graphs A and B, the subgraphs are called components. Furthermore, within a component there must be at least one path from each vertex to each other vertex. (Newman 2010: 142.)

Example 2. Let G be a graph $G = (V, E)$ where $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and $E = \{v_1v_5, v_2v_5, v_4v_5, v_3v_7, v_6v_7\}$. Graph G is depicted in figure 5.

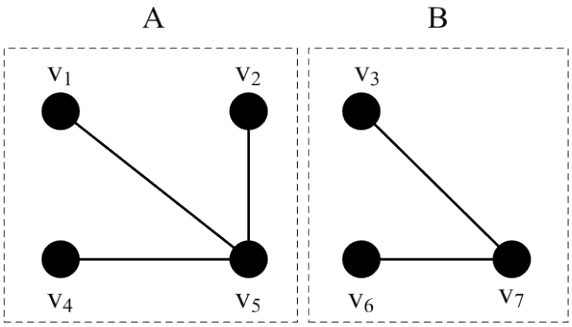


Figure 5. Graph G with components A and B.

The graph G in figure 5 has two components because, the vertices of subgraph A cannot be reached from subgraph B.

3.3 Tree

A tree is a connected acyclic graph (figure 6), and the edges of the tree are called branches (Foulds 1992: 27). Foulds (1992: 29) states the following alternative definitions for a tree:

1. *A tree is a connected graph with n vertices and (n-1) edges.*
2. *A tree is an acyclic graph with n vertices and (n-1) edges.*
3. *A tree is a graph in which there is exactly one path between every pair of its vertices.*
4. *A tree is an acyclic graph which has the property that if any two of its vertices that are not adjacent are joined directly by an edge then the resulting graph possesses exactly one cycle.*

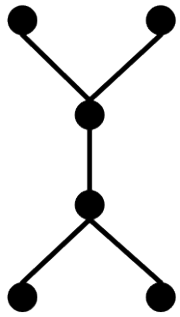


Figure 6. A tree with six vertices and five edges.

3.4 Matrix representation for a graph

The structure of a vertex labeled graph G can be represented with an adjacency matrix. The adjacency matrix is an n-by-n –matrix $A = (a_{ij})$. If $a_{ij} = 1$ then the vertex v_i is adjacent to vertex v_j, otherwise $a_{ij} = 0$. The graph of figure 3 can be represented with the adjacency matrix shown below. (Foulds 1992: 76; Newman 2010: 110-111.)

$$A = \begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 0 & 1 \\ v_2 & 1 & 0 & 1 & 1 \\ v_3 & 0 & 1 & 0 & 1 \\ v_4 & 1 & 1 & 1 & 0 \end{bmatrix}$$

According to Foulds (1992:76-77) adjacency matrix A depicting graph G has the following properties.

1. *A is symmetric*
2. *The sum of entries in each row i of A equals the degree of v_i.*

3. *There is a one-to-one correspondence between labeled graphs with n vertices and $n \times n$ symmetric binary matrices with all entries on the leading diagonal equal to zero.*
4. *G is connected if and only if there is no labeling of the vertices of G such that its adjacency matrix is block diagonal matrix.*
5. *If A_1 and A_2 are adjacency matrices which correspond to different labelings of the same graph G, then for some permutation matrix P, $A_1 = P^{-1}A_2P$.*
6. *The (i,j) entry of A^m is the number of walks of length m for vertex v_i to v_j, in G. This means that,*

– *if $i \neq j$, the (i,j) entry of A^2 is equal to the number of paths containing exactly two edges from v_i to v_j. The (i,i) entry of A^2 is the degree of v_i and that of A^3 is equal to twice the number of triangles containing the v_i.*
– *if G is connected, the distance between its vertices v_i and v_j, for $i \neq j$, is the least integer m, for which the (i,j) entry of A^m is nonzero.*

3.5 Adjacency matrix for directed graph

In a directed graph (or digraph) each edge has a direction. Edges are represented as lines with arrows pointing from one vertex to another. An adjacency matrix for directed graph has the following properties. The value of matrix cell of A_{ij} is set 1 if there exist an edge from i to j, otherwise the value is 0. Consider the graph shown in figure 7 (Jungnickel 2013: 40-41).

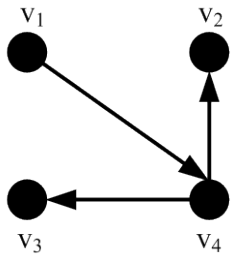


Figure 7. Directed graph.

The adjacency matrix A for the graph of figure 7 is constructed as follows.

$$A = \begin{bmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 0 & 0 & 1 \\ v_2 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 0 & 0 \\ v_4 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

3.6 Vertex – dominating set of a graph

In this work the concept of dominating set is utilized in the identification of vertices representing the values of the key attributes of relational database in a graph G. A vertex in a graph G dominates the vertices in G that are adjacent to it. In the graph $G = (V, E)$, $U \subset V$, U is said to be the vertex dominating set of graph G, if the every vertex of V belongs to U or is covered (dominated) by vertex of U. Dominating number of G is the cardinality of the vertex dominating set with the least number of elements. Dominating number is denoted by $\sigma_0(G)$ or σ_0. A dominating set of a graph G is *minimal* if none of its proper subsets are dominating. The dominating set of G that has the smallest number of elements among all the dominating sets of the G is said to be *minimum*. Moreover, if G contains such set of vertices that it does not contain any adjacent vertices it is called a vertex independent set of G. (Savolainen 1978: 81-82, Foulds 1992: 130.)

Foulds (1992:131) states the following about the vertex dominating set of a graph G.

1. *A vertex – dominating set for G exists.*
2. *If G is complete the* $\sigma_0 = 1$.
3. *If G is connected with at least one internal vertex, then every minimum vertex – dominating set for G does not contain any dependent vertices.*
4. *It is possible to remove a subset of vertices (possibly empty) from any vertex – dominating set for in order to create a minimal (but not necessarily minimum) vertex –dominating set* G.
5. *A minimal vertex – dominating set for G is not necessarily a vertex independent set of* G.
6. *Every maximal vertex independent set of G is a vertex – dominating set for* G.

Example 3. Let G be a graph (V, E) where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{v_1v_2, v_2v_3, v_3v_4, v_4v_1, v_2v_4, v_3v_5\}$. Graphs G is depicted in figure 8.

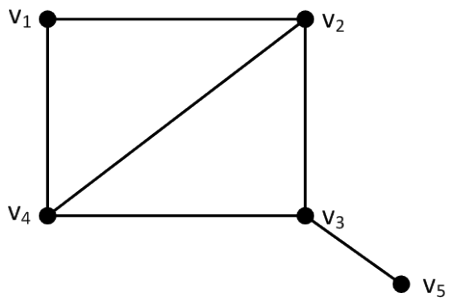


Figure 8. Graph G for example 3.

For the graph G shown in figure 8 $\sigma_0(G) = 2$ and the minimum vertex –dominating set D contains vertices v_2, v_3. It is worth noticing that also vertices v_3, v_4 form a minimum vertex –dominating set for graph G.

4 DATA STRUCTURES AND MODELS

Hislop (2006:15-16) defines data as a result of observation. Data can be, for example, raw numbers, words or images. Information can be seen as a refined form of data. Data is refined into information by organizing it into a meaningful pattern. So, information always includes some intellectual inputs. Both data and information are basic building blocks of knowledge. Moreover, application, analysis, and use of data and information are seen as source of knowledge. Hislop (2006: 15) states that "*knowledge is data or information with a further layer of intellectual analysis added, where it is interpreted, meaning is attached, and is structured and linked with existing systems of beliefs and bodies of knowledge.*"

The successful refinement of data into information and eventually into knowledge as well the efficient utilization of knowledge requires competent structures and models. In the following sections, the concepts of data structure and data models are introduced. Data models are covered at general level. The entity-relationship and relational models have importance for this study, so they are discussed in more detailed level.

The purpose of the following sections is to provide a general understanding of the data models and data representation techniques. The ideas of the models discussed in the following are utilized later in this work. For example, how to represent data in a structured form. Typically, data is organized into records, which are built around elements that identify individual records. Furthermore, the records and record collections can be visualized as graphs, that is as vertices and edges connecting them.

4.1 Data structures, data models, and ontologies

Data consist of single items called elements. Elements can be, for example, values, codes, and symbols. Williams (1971: 1) states that by storing the elements in an organized manner they are given a structure. The structure of data enables the preservation of the relationships between elements. Data structure also provides access from one element to another. (Williams 1971:1; Falley 2007: 148.)

Data structure refers to how the data is stored in the memory. Main function of data structure is to ensure algorithm efficiency and conceptual unity (Black 2004). More formally the data structure is a 4-tuple $< D, F, S, A >$, where D represents the domain or domains defined by the data structure. F is a set of function definitions, which provide operations on values in D. The variables and their

relationships are defined in storage structure S. The set of algorithms (A) realizes the functions (F) by processing the storage structure. (Hansen 1981: 89.)

Data model provides concepts for describing the structure of the database. The structure of the database refers to the data types, relationships, and constraints for the data. Typically, data models are divided into three categories, which are conceptual data models, physical data models, and representational data models. Conceptual level models describe the data as users perceive it whereas physical data models describe how the data is stored in the computer. Representational data models provide concepts that can be understood by users, but also they are close to the machine-understandable presentation of the data and, therefore, can be implemented on a computer system. (Elmasri & Navathe 2007: 30-31.)

Ontology can be considered as a conceptual level data model. It defines a set of concepts with which the domain of knowledge can be modeled. The concepts are classes, attributes, and relationships between elements. Ontologies can be seen as semantic level abstractions about the data where database schemas provide logical or physical level models of the data. Key application areas of ontologies are the integration of heterogeneous databases, to enable interoperability between different systems, and specifications of interfaces to independent, knowledge-based services. (Gruber 2009.)

4.2 Network data model

The network data model is a logical level data model. Network database is a collection of records, and a record is a collection of attributes. Relationships between records are represented with links. (Silberschatz, Korth & Sudarshan 2010: D1.)

Consider a database that has two types of records *supplier* and *product*. *Supplier* record contains attributes *supplier_id* and *supplier_name*, and respectively *product* record has attributes *product_id*, *product_name* and *units_in_stock*. Formal record definitions are listed below.

```
type supplier = record
                    supplier_id
                    supplier_name
            end

type product = record
                       product_id
                       product_name
                       units_in_stock
```

```
end
```

Figure 9 represents database with data stored according to definitions of supplier record and product record. Note that supplier *Sons* & *Sons* delivers two products *P*1 and *P*2. The properties of the relationships between records can be represented with the data-structure diagram (figure 10).

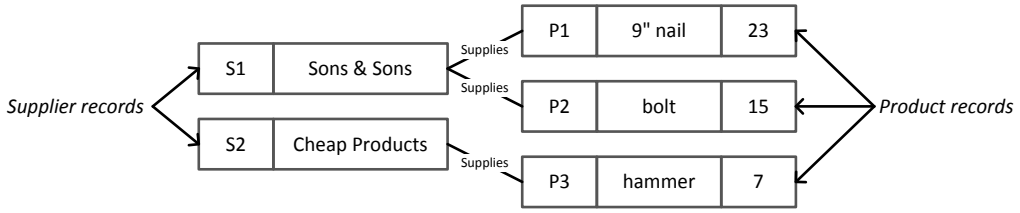


Figure 9. Sample database for supplier-product data.

Schema for the sample database is represented by the data-structure diagram. The nature of the relationship between records can be specified with directed arrows. Many-to-many relations are depicted with an undirected arrow; one-to-many relationships are depicted with directed arrows, and bidirectional arrows are used for denoting one-to-one relationships (figure 10). The relationships between records can also be named. For example, in this case the relationship could be named as *supplies*. (Bachman 1969: 4-5; Silberschatz et al. 2010: D2-D5.)

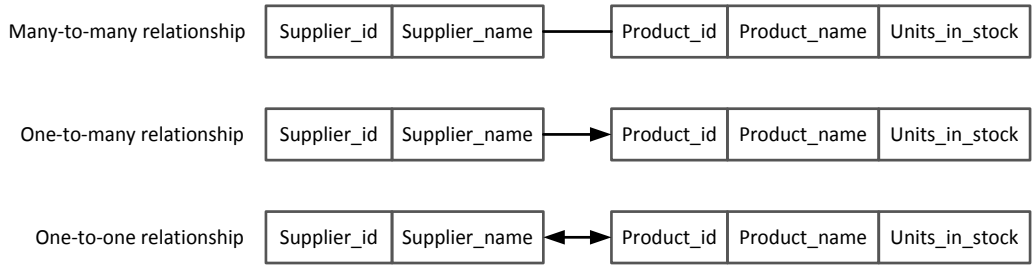


Figure 10. Data-structure diagram for a sample database.

4.3 Hierarchical data model

A hierarchical database is a collection of records connected by links. Similarly, to the network data model records are collections of single-valued attributes. Association between two records is represented as a link. Hierarchical databases are typically represented as a collection of rooted trees called database trees. Figure 11 depicts a small database for products and their suppliers, where *Sons* & *Sons* supplies two products. In the schema level the relationship type is indicated by directed arrows (figure 12). (Silberschatz et al. 2010: E1-E2.)

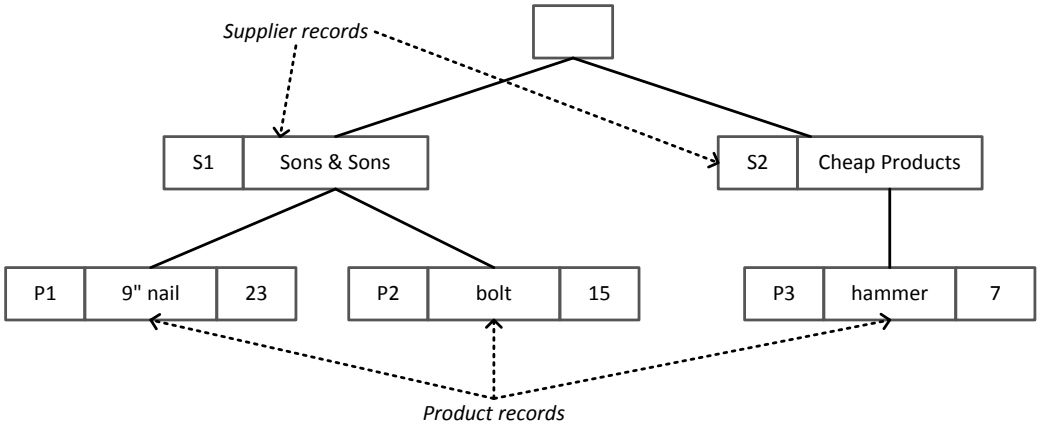


Figure 11. Database tree.

Hierarchical data model is considered quite rigid (Elmasri & Navathe 2000: 805-808). For example, situation where a product is supplied by multiple suppliers, can only be described with record replication that leads to (Silberschatz et al. 2010: E2):

1. data inconsistency when updating and
2. waste of space.

Schema for the hierarchical database is represented with a tree-structure diagram. The nature of relationship between records can be specified with directed arrows (figure 12). One-to-many and one-to-one relationships can be represented by single arrows. One-to-many relationships are represented by a directed arrow. One-to-one relationship is depicted with a bidirectional arrow. (Silberschatz et al. 2010: E2-E4.)

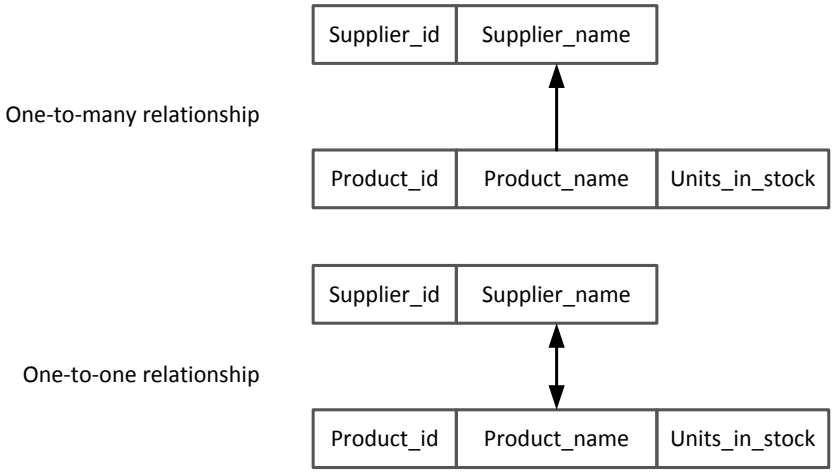


Figure 12. Tree-structure diagrams.

Many-to-many relationship is depicted with two tree-structure diagrams T_1 and T_2 (figure 13). The diagrams represent one-to-many relationships between supplier record and product record, and vice versa. This kind of expression is needed in situations where products have multiple suppliers and suppliers supply multiple products. (Silberschatz et al. 2010: E5.)

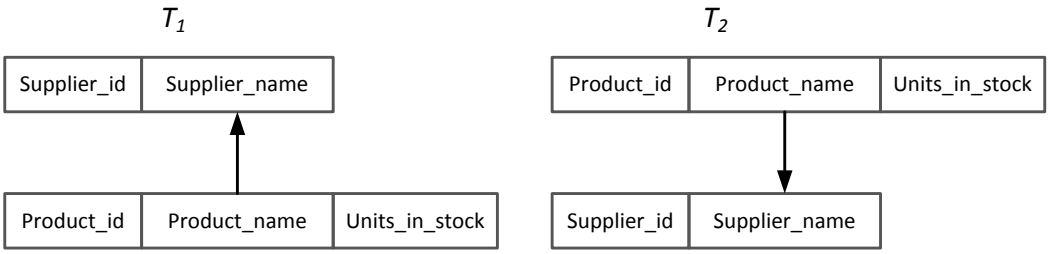


Figure 13. Expression of many-to-many relationship.

4.4 Entity-Relationship data model – ER Model

The ER model is an abstract representation of the database. The elements of the model are entity sets, attributes, and relationships. An entity represents an object in the real world, and similar entities form a group called an entity set. Within an entity set the entities have their own individual values for the given properties or attributes. Each entity within an entity type is unique, and it is identified by attribute or set of attributes called keys. (Chen 1976: 10-11; Garcia-Molina, Ullman & Widom: 2002: 24-25; Elmasri & Navathe 2007:61-62, 65-66.)

Attributes describe the properties of an entity. In the ER model, attributes can be composite or atomic, single-valued or multivalued and stored or derived. Composite attribute is an attribute that can be divided into attributes having independent meanings. An attribute that cannot be divided is called atomic attribute. An attribute that has only one value for an entity is called single-valued, and an attribute having set of values for the same entity is called multivalued. Some attributes are derivable from attributes, and they are called stored attributes. (Garcia-Molina et al. 2002: 25; Elmasri et al. 2007; 63-64.)

Chen (1976: 12) states that attribute is a function F that connects (figure 14) an entity set with a value set $f: E_i \rightarrow V_i$.

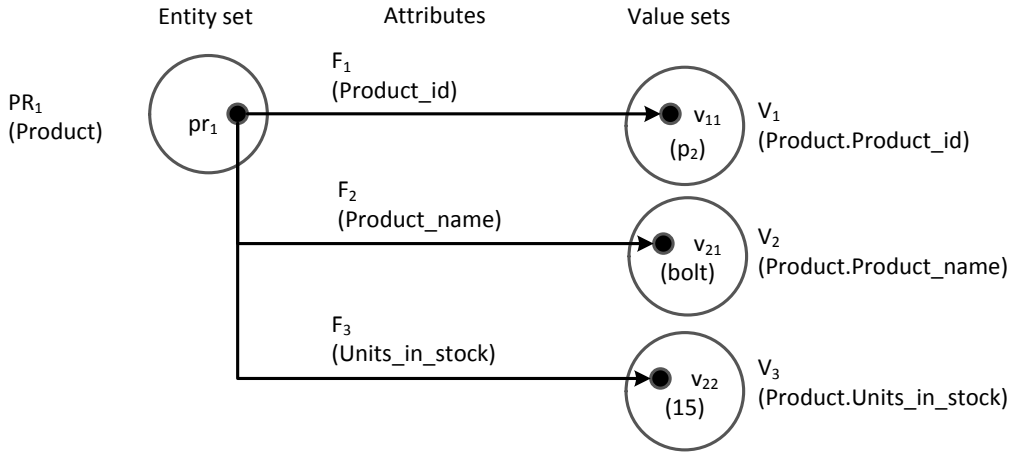


Figure 14. Attribute mappings for *Product* entity set.

Connections between entity sets are called relationships. Relationship types R are used for defining associations between n entity types $E_1, E_2, \dots, E_n$. According to Elmasri and Navathe (2007: 70) R is a set of relationship instances r_i, where each r_i associates n individual entities $(e_1, e_2, \dots e_n)$, and each entity e_j in r_i belongs to an entity type $E_j, 1 \leq j \leq n$. The type of relationship defined on mathematical relation $E_1, E_2, \dots, E_n$ and each entity type participates in relationship type R. Respectively entities $e_1, e_2, \dots e_n$ participate in the relationship instance $r_i = (e_1, e_2, \dots e_n)$.

The schema of ER model is represented with Entity-Relationship diagrams (figure 15), where entity sets are depicted with rectangles, attributes are portrayed with ovals. Respectively the relationships are represented as diamonds (figure 16). (Elmasri & Navathe 2007: 80.)

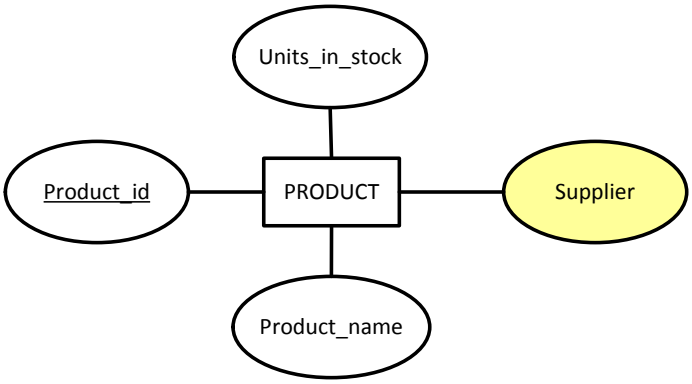


Figure 15. Entity-relationship diagram for the Product entity type.

Relationships between different entity types in ER diagram are not usually represented as an attribute. For example, in figure 15 PRODUCT entity has an

attribute called Supplier which implies that there should be a relationship between entities PRODUCT and SUPPLIER. An attribute depicting such association is converted into a relationship type (figure 16). The relationship shown in figure 16 is a binary relationship. Sometimes a relationship can have more participants these kind relationships are called multiway relationships. (Garcia-Molina et al. 2002: 28; Elmasri & Navathe 2007:70.)

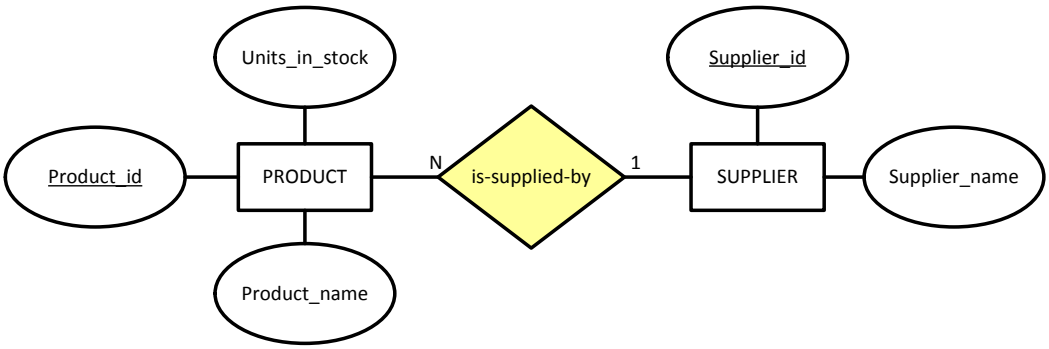


Figure 16. Relationship between Product and Supplier entities.

Entity-Relationship diagram contains information about identifying attributes called keys (underlined attributes in figure 16) and cardinality ratios for binary relationships. Other constraints such as referential constraints and domain-specific constraints can be expressed in ER model. (Garcia-Molina et al. 2002: 49-53; Elmasri & Navathe 2007:74-76.)

4.5 Relational data model

In the relational model, the data is represented as a collection of relations. Each relation is represented as a two-dimensional table. The elements of the table are tuple (a table row) and attribute (column header). The values of each column are determined by the domain of possible values. A domain represents a collection of atomic (indivisible) values. Each domain has predefined data type. (Elmasri & Navathe 2007:142-143.)

The structure of relation R is described by a relation schema. Relation schema $R(A_1, A_2, \dots, A_n)$ consists of the relation name R and an attribute list $A_1, A_2, \dots, A_n$. The names of the attributes are determined by the role they have in a certain domain, denoted by $dom(A_i)$. For example a relation for products can be formed as follows $PRODUCT(Product\_id, Product\_name, Units\_in\_stock)$. (Codd, 1970: 379; Elmasri & Navathe 2007:143.)

The instance or state of the relation r of the relation schema, $r(R)$, is set of n-tuples $r = \{t_1, t_2, \dots, t_k\}$. N-tuple is defined as an ordered set of values $t =< v_1, v_2, \dots, v_n >$. Each value v_i is an element of $dom(A_1)$. An instance of a relation r can be defined more formally as follows (Elmasri & Navathe 2007:144):

$$r(R) \subseteq \left(dom(A_1) \times\ dom(A_2) \times \dots \times\ dom(A_n)\right)$$

Codd (1970: 379) states that a table representing relation R has the following properties:

1. *Each row represents a tuple of R.*
2. *The ordering of rows is immaterial.*
3. *All rows are distinct.*
4. *The order of columns must correspond the ordering in the relation schema $R(A_1, A_2, \dots, A_n)$.*
5. *The significance of each column is partially conveyed by labeling it with the corresponding domain.*

By definition, each tuple in relation must be different (see above item 3), which means that tuples in relation cannot have exactly same combination values of attributes. The uniqueness of a relation is ensured by forming a superkey of the relation. Superkey is a subset of values that are different for each tuple in relation of R. Typically, superkey may contain redundant attributes, in these cases it is useful to assign a non-redundant key to ensure the uniqueness of the tuples in relation r of R. A key attribute must obey the following rules. (Elmasri & Navathe 2007:150-151.)

1. Tuples in the relation cannot have same values for all attributes in the key.
2. The key must be minimal superkey. In other words, no attributes can be removed from it without conflicting the uniqueness constraint of item 1.

Relation schema usually has more than one key. All the keys are called a candidate key, and one of these keys is defined as the primary key (PK) of the relation. Primary key identifies each (unique) tuple in the relation. (Codd, 1970: 380; Elmasri & Navathe 2007:150-151.)

Sometimes it is necessary to refer from one relation to another (figure 17). To maintain referential integrity, the concept of foreign key (FK) is introduced. A foreign key is an attribute set of the relation schema R_1 that references to a relation schema R_2. The foreign key has to satisfy the following conditions. (Codd 1970: 380; Elmasri & Navathe 2007:153-154.)

1. The attributes in the foreign key of R_1 have the same domain as the attributes of primary key in R_2.

2. A value of foreign key in tuple t_1 of the relation $r_1(R_1)$ occurs as value of PK for some tuple in t_2 in the relation $r_2(R_2)$. If $t_1[FK] = t_2[PK]$, then the tuple t_1 is said to reference the tuple t_2.

Product			
Product_id (PK)	Supplier_id (FK)	Product_name	Units_in_stock
P1	S1	9” nail	23
P2	S1	bolt	15
P3	S2	hammer	7

Foreign key

Primary key

Supplier	
Supplier_id (PK)	Supplier_name
S1	Sons & Sons
S2	Cheap Products
S3	Buy, buy!

Figure 17. Referential relationship between *Product* and *Supplier* relations.

The schema of relational database can be represented as a schema diagram (figure 18). The schema contains information keys and attributes and also information about possible cardinalities, referential and semantic constraints.

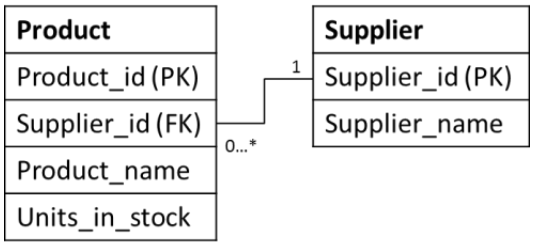


Figure 18. Database schema diagram.

The ER model and the relational model have correspondences, which are listed in table 2. These equivalences are utilized throughout this study when the properties of vertices and edges in an ARS-based-graphs are examined.

Table 2. Correspondences between ER model and relational model (according to Elmasri & Navathe 2007: 224).

ER model	Relational model
Entity type	Entity relation
1:1 or 1:N relationship type	Foreign key (or relationship relation)
M:N relationship type	Relationship relation and two foreign keys
n-ary relationship type	Relationship relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary or secondary key

4.6 Semantic link network – SLN

Semantic link network is flexible and associative method for representing semantic data. It is a directed graph (figure 19) which consists of nodes and links between nodes. Link between nodes is a labeled pointer, called alpha link. The label contains semantic properties that are derived from the domain. Any semantic relationship between nodes is described by a property or a combination of properties. SLN can be denoted $r \xrightarrow{\alpha} r'$ where r and r' represents real world concepts and α is a semantic factor connecting the concepts. (Zhuge 2003: 89-90; Zhuge 2005: 40)

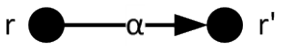


Figure 19. Semantic link network.

According to Zhuge (2011: 990) the use of SLN can be justified as follows:

1. It supports intelligent applications by assigning semantic indicators and rules to links and enabling relational, analogical, inductive, and complex reasoning.
2. It explores the laws of semantic linking. It pursues diversity and user experience of linking and exploring rather than the correctness.
3. It provides a light-weight semantic networking approach for peer-to-peer knowledge sharing.

The semantic expressiveness and reasoning capabilities of SLN is based semantic primitives (table 3). Reasoning rules are constructed by combining semantic links. (Zhuge 2005: 41-43; Zhuge, Yun, Jia & Liu: 2005: 228-229.)

Table 3. Semantic link primitives.

Link type	**Characteristics**
Cause-effect	Transitive link that indicates causality between two items.
Implication	Transitive link that means that the semantics of predecessor implies to its successor.
Subtype	Transitive link indicates that the successor is part of its predecessor.
Similar-to	Intransitive link that describes the similarity in semantics between successor and predecessor.
Instance	Link showing that the successor is an instance of the predecessor.
Sequential	Transitive link that indicates that the content of the item A is a successor of the content of item B. Also links can be connected in a sequential chain.
Reference	Transitive link that means that item A is an explanation of item B.
Equal-to	Link showing that two items are identical in meaning.
Empty	Link showing that two items are irrelevant to each other.
Null or unknown	Link that indicates unknown or uncertain relation between two items.
Non-α relation	Link that shows that there is no semantic relationship between two items.
Reverse relation operation	If there exists semantic relation from A to B, then there also exists relation from B to A.

5 ADJACENCY MODEL

Adjacency Model (AM), is a model for data representation. It is based on the concepts of adjacency relation system ARS, adjacency relation system with adjacency defining sets (ARST), the unique ARST and the valid ARST. ARST is a model that can be used to describe adjacencies between sets of elements belonging to collections called types. (Wanne 1998: 9-12; Töyli 2002: 39.)

The concept of the adjacency relation system was introduced by Wanne (1998) in "*Adjacency Relation Systems*". Adjacency relation system (ARS) is a pair of sets and relations (A, R). Set $A = \{A_1, A_2, \dots, A_n\}$, $n \geq 1$, is a set containing pairwise disjoint finite nonempty sets and $R = \{R_{ij} | i, j \in \{1, 2, \dots, n\}\}$ is a set of relations, where each R_{ij} is a relation on $A_i \times 2^{A_j}$, where 2^{A_j} denotes the power set of A_j. (Wanne 1998:9.)

Consider relation R_{ij} containing pairs $(x, y_1), (x, y_2), \dots, (x, y_m)$. Since each pair has x as the first component of the pair, thus elements $y_k (k = 1,2, \dots, m)$ are said to be adjacent to the element x. This is denoted as $Ad_j(x)$. (Wanne 1998: 9; Wanne & Linna 1999: 40.)

Example 4. Consider an adjacency relation system (A, R), where $A = \{A_1, A_2, A_3\}$, $A_1 = \{x_1, x_2, x_3\}$, $A_2 = \{y_1, y_2, y_3\}$, $A_3 = \{z_1, z_2\}$ and R contains relations:

$R_{11} = \{(x_2, \{x_1\})\}$

$R_{12} = \{(x_1, \{y_1\}), (x_2, \{y_3\}), (x_3, \{y_1\})\}$

$R_{13} = \{(x_2, \{z_1\})\}$

$R_{21} = \emptyset$

$R_{22} = \{(y_2, \{y_3\})\}$

$R_{23} = \{(y_3, \{z_2\})\}$

$R_{31} = \{(z_2, \{x_2\})\}$

$R_{32} = \emptyset$

$R_{33} = \{(z_1, \{z_2\})\}.$

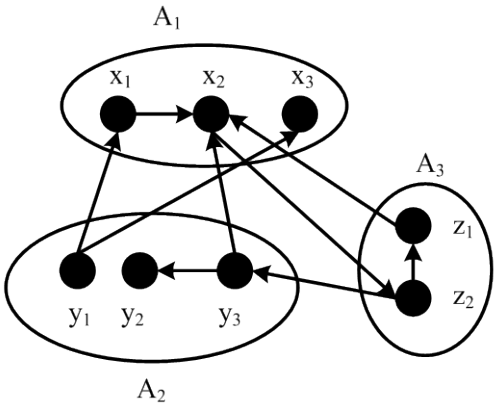


Figure 20. Graph representation of adjacency relation system.

The adjacency relation system shown in figure 20 is not symmetric because, for example, the element y_3 is adjacent to element x_2 $(Ad_2(x_2) = \{y_3\})$ but not vice versa.

Let us introduce the concept of a symmetric adjacency relation system. ARS is symmetric if for each pair $x \in A_i, y \in A_j$ holds that $x \in Ad_i(y)$ if and only if $y \in Ad_j(x)$ and for each $i, 1 \leq i \leq n$. (Wanne 1998: 10). Note that, in this work the adjacency relation systems depicting a relational databases considered symmetric.

Example 5. Change the definitions of the relations $R_{11}, R_{13}, R_{21}, R_{22}, R_{31}, R_{32}$ *and* R_{33} as follows:

$R_{11} = \{(x_1, \{x_2\}), (x_2, \{x_1\})\}$

$R_{13} = \{(x_2, \{z_1, z_2\})\}$

$R_{21} = \{(y_1, \{x_1, x_3\}), (y_3, \{x_2\})\}$

$R_{22} = \{(y_2, \{y_3\}), (y_3, \{y_2\})\}$

$R_{31} = \{(z_1, \{x_{2,}\}), (z_2, \{x_{2,}\})\}$

$R_{32} = \{(z_2, \{y_3\})\}$

$R_{33} = \{(z_1, \{z_2\}), (z_2, \{z_1\})\}.$

The refined ARS considered is symmetric, and it is represented by an undirected graph (figure 21). (Wanne 1998:10-11.)

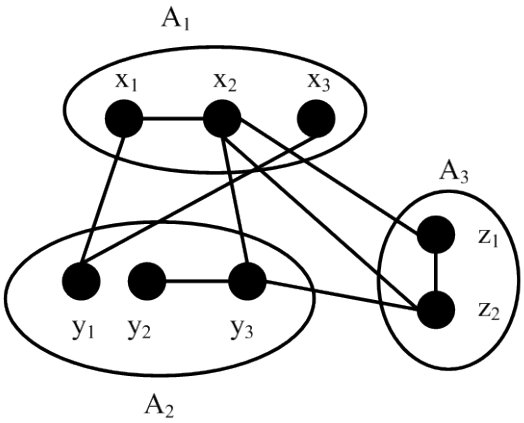


Figure 21. Symmetric ARS portrayed by undirected graph.

The adjacency between elements of certain entity types is expressed by relations. If the adjacency of elements depends on the definition of relations, it is said to be weak. The adjacency of elements is said to be strong if it is defined with respect to a set of entity types. The definition for adjacency defining sets if given next. (Töyli et al. 2002a: 303; Töyli et al. 2002b: 285.)

In an adjacency relation system each element set A_i, $i = 1,2,\dots,n$, represents a certain entity type denoted as $T_i, i = 1,2,\dots,n$. In addition, associate with each index pair $i,j \in \{1,2,\dots,n\}$ a set of indices $K \subseteq \{1,2,\dots,n\} - \{i,j\}$ and also a set of entity types $\tilde{T}_{ij} = \{T_k | k \in K\}$. The set $\tilde{T}_{ij}$ gives the entity types which determine the adjacency between the elements of A_i and A_j. (Wanne 1998: 11.)

The adjacency defining set $\tilde{T}_{ij}$ is defined as follows. The elements $x \in A_i, y \in A_j$ where $i,j \in \{1,2,\dots n\}$ and $x \neq y$, are considered to be adjacent with respect to a set of entity types $\tilde{T}_{ij} = \{T_k | k \in K\} \neq \emptyset$ if for each $k \in K$ there is an element $z \in A_k$ such that $x \in Ad_i(z)$ and $y \in Ad_j(z)$. (Wanne 1998:11; Töyli et al. 2002b: 284.)

In the ARS introduced in example 4 the elements y_3 ja z_1 are adjacent to element x_2 so their adjacency is defined by set $\tilde{T}_{23} = \{T_1\}$. For example, in the figure 20 the adjacency between elements y_1 and x_1 is considered weak because their adjacency is depends only on the relation definitions. On the other hand, the adjacency between elements y_3 and z_2 depends on the entity type T_1 ($\tilde{T}_{23} = \{T_1\}$), it is considered to be strong. The concepts of weak and strong adjacency are combined together in the concept of unique the adjacency. (Töyli et al., 2002a: 303.)

In addition to the adjacency defining sets Töyli (2002:47-48) introduced the idea of transitive adjacency which, is sequence of elements $x = x_1, x_2, \dots x_m = y$ such that $x_i \in Ad(x_{i+1})$, $i = 1,\dots,m-1$. Transitive adjacency is denoted by $x \in Ad_{tr}(y)$

For example, in figure 22 element y_1 is said to be transitively adjacent to z_2, $y_1 \in Ad_{tr}(z_2)$. The transitive adjacency of elements y_1 and z_2 is determined by sequence y_1, x_1, x_2, z_2 (figure 22). Töyli (2002: 47) points out that some of the elements $x_1, x_2, \ldots x_m$ can be of the same type.

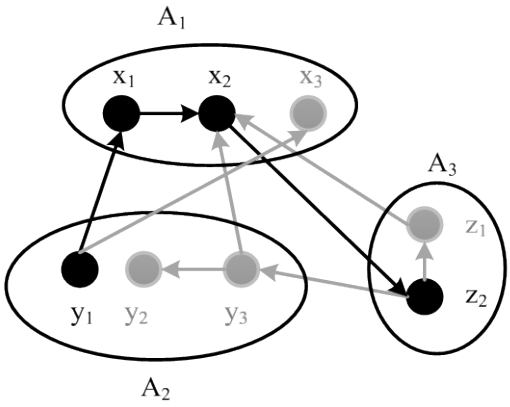


Figure 22. Sequence of elements depicting transitive adjacency between elements y_1 and z_2.

Adjacency relation system with adjacency defining sets (ARST) can be denoted by (A, R, τ), where τ means the set of adjacency defining sets. ARST is unique if for each pair $i, j = \{1,2, \ldots, n\}$ of integers the adjacency defining set $\tilde{T}_{ij}$ is nonempty for all elements $x \in A_i, y \in A_j$ and x and y are adjacent if and only if they are adjacent with respect to $\tilde{T}_{ij}$. (Wanne 1998: 12.)

Example 5. Consider the ARS defined in example 4. Let us add a new relation $R_{32} = \{(z_1, \{y_3\})\}$. Now we have the set of relations:

$R_{11} = \{(x_2, \{x_1\})\}$

$R_{12} = \{(x_1, \{y_1\}), (x_2, \{y_3\}), (x_3, \{y_1\})\}$

$R_{13} = \{(x_2, \{z_1\})\}$

$R_{21} = \emptyset$

$R_{22} = \{(y_2, \{y_3\})\}$

$R_{23} = \{(y_3, \{z_2\})\}$

$R_{31} = \{(z_1, \{x_2\})\}$

$R_{32} = \{(z_1, \{y_3\})\}$

$R_{33} = \{(z_1, \{z_2\})\}$.

and the adjacency defining set

$\tilde{T}_{23} = \{T_1\}$,

$\tilde{T}_{11} = \tilde{T}_{12} = \tilde{T}_{13} = \tilde{T}_{21} = \tilde{T}_{22} = \tilde{T}_{31} = \tilde{T}_{32} = \tilde{T}_{33} = \emptyset$.

In the figure 23 z_1 and y_3 are adjacent with respect to relation R_{32} (dashed arrow) and they are also adjacent according to $\tilde{T}_{23} = \{T_1\} \neq \emptyset$. If there are no other non-empty adjacency defining set the *ARST* is said to be unique.

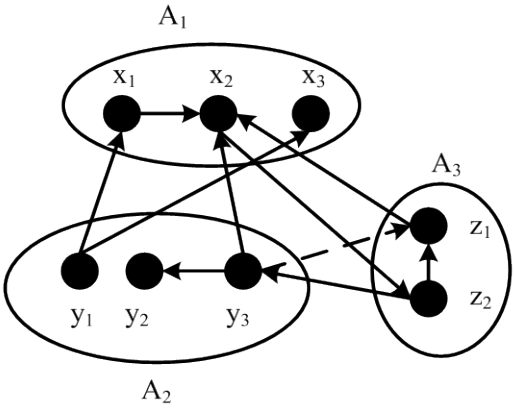


Figure 23. Unique ARS.

The notation $T_i \rightarrow T_j$ for a relation type indicates that the relations R_{ij} are defined in $A_i \times 2^{A_j}$ The set of relation types $\{T_i \rightarrow T_j | (i,j) \in S\}$, where $S \subseteq \{1,2,\dots,n\} \times \{1,2,\dots,n\}$, is called a relation combination. The relation combination for an ARST (A, R, τ) the restriction of R on a given relation combination is determined by $S \subseteq \{1,2,\dots,n\} \times \{1,2,\dots,n\}$ is denoted by $R|S$, i.e. $R|S = \{R_{ij} \in R | (i,j) \in S\}$. (Wanne 1998:13.)

The relation combination is defined as follows. Given the entity types $T_1, \dots, T_n$, a set $S \subseteq \{1,2,\dots,n\} \times \{1,2,\dots,n\}$, and adjacency defining sets τ. A relation combination $\{T_i \rightarrow T_j | (i,j) \in S\}$ is said to be valid, if for any unique ARST (A, R, τ) there is no unique ARST (A, R', τ) such that $R|S = R'|S$. Otherwise the relation combination is considered to be non-valid. The relation combination plays key role in query optimizations because it enables the restriction of the search space and so limits the search time used by the query. (Wanne 1998:13; Töyli et al. 2002a: 304.)

5.1 Adjacency schema

In the adjacency model, the concept of type represents a group of elements. Adjacency schema (AdSchema) was originally developed for representing semistructured data. AdSchema focuses on the adjacencies between the types contrary to adjacency relation systems which concerns adjacencies of individual elements of the types. (Töyli 2006: 61.)

AdSchema is a pair (T_A, R), where $T_A = \{T_1, T_2, \dots, T_n\}$, is a set of types, and $R = \{R_i | i \in 1,2, \dots, n\}$, where each $R_i = \big(T_i, Ir(T_i)\big)$, and $Ir(T_i)$ denotes a subset of T_A. If $R_i = \big(T_i, \{T_{i_1}, T_{i_2}, \dots, T_{i_m}\}\big) \in R$, then each type T_{i_k} $(k = 1,2, \dots, m)$ is said to be interrelated to the type T_i. The set of interrelated types $\{T_{i_1}, T_{i_2}, \dots, T_{i_m}\}$ is denoted by $Ir(T_i)$. (Töyli 2006: 61-62.)

The interrelationship between types T_i and T_j is denoted with $T_i \rightarrow T_j$. Assume that a data structure contains types T_i and T_j and if the elements of these types are adjacent with each other, then the notion $T_i \rightarrow T_j$ can be used. This implies that, $T_j = Ir(T_i)$ and the interrelationship of the elements can be represented with pair of elements. (Wanne 1998: 12-14; Töyli 2006: 62.)

Töyli (2006: 62) states that relations of the AdSchema are considered to be symmetric. In symmetric AdSchema, there is no predefined order between types in the schema, and it is possible to traverse the given data structure in both directions. AdSchema (T_A, R) is symmetric if for each pair of types it holds that $T_i = Ir\big(T_j\big)$ and $T_j = Ir(T_i)$. (Töyli 2006: 63.)

Interrelationship defining type can be used for defining the relationship between types. More specific, types $T_i, T_j \in T_A$ are interrelated with respect to a type T_k, $T_k \in T_A$, $k \in \{1,2, \dots, n\} - \{i, j\}$, if $T_k = Ir(T_i)$ and $T_k = Ir\big(T_j\big)$. Moreover, T_i and T_j, $i \neq j$ are said to be transitively interrelated if there are one or more consecutive intermediate types between them. (Töyli 2006: 63-64.)

The number of interrelationships in AdSchema, i.e. the size of the schema can be minimized with the concept interrelationship combination. Based on the minimal and valid combination all the rest of the interrelationships can be derived. (Töyli 2006: 65.)

Example 6. Let (T_A, R) be a pair, where $T_A = \{T_1, T_2, T_3, T_4, T_5, T_6\}$, R contains the relations $\{R_1, R_2, R_3, R_4, R_5, R_6\}$ and the relations defined as follows $R_1 = (T_1, \{T_2, T_4\})$, $R_2 = (T_2, \{T_1, T_3, T_4\})$, $R_3 = (T_3, \{T_2\})$, $R_4 = (T_4, \{T_1, T_2, T_5, T_6\})$, $R_5 = (T_5, \{T_4\})$ and $R_6 = (T_6, \{T_4\})$. The AdSchema is portrayed in figure 24.

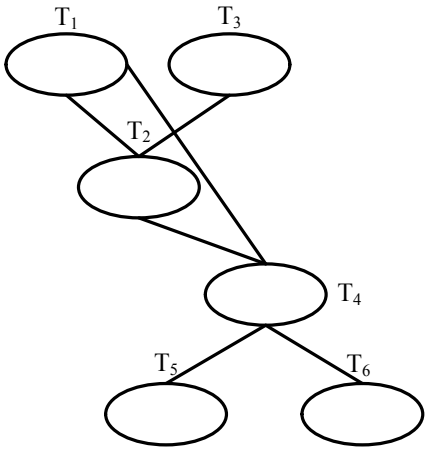


Figure 24. AdSchema.

The minimal interrelationship combination for the AdSchema of shown in figure 24 is R_2 and R_4. All the rest of the interrelationship can be derived based on these two relations.

5.2 Modeling adjacency relation systems with AdSchema

The basic structure of an ARS, i.e. interrelationships between types, can be represented by AdSchema. However, this requires some adjustments to the definitions discussed in the previous section. In the previous section was stated that types T_i and T_j are interrelated if their elements are adjacent to each other. In order to represent the basic structure of an ARS, it must be assumed that types T_i and T_j are interrelated if there exist some adjacent elements x and y, such that $x \in T_i$ and $y \in T_j$.

Since the AdSchema provides information about the basic structure of an ARS, it is a useful tool for analyzing adjacency relation systems. Especially in cases where ARS depicts a relational database an ARS contains multiple components, which have the similar structure. Below is a list of steps needed for representing the structure of an ARS with AdSchema. The conversion method is applied in example 7.

1. Select a set (A_i) from adjacency relation system.
2. Create a vertex for the corresponding type (T_i).
3. Repeat steps 1 and 2 until every set is covered.
4. Take and adjacency defining set and define it as in interrelationship defining type.
5. Repeat step 4 until all interrelationship defining types are defined.

6. Create interrelationship (edge) between type and its interrelationship defining type.
7. Repeat step 6 until all interrelationships are defined
8. Create all other interrelationships according to the original adjacency relation system

Example 7. Let (A, R) be symmetric adjacency relation (figure 25) system where $A = \{A_1, A_2, A_3, A_4\}$ and $A_1 = \{x_1, x_2, x_3\}$, $A_2 = \{y_1, y_2, y_3\}$, $A_3 = \{z_1, z_2\}$ and $A_4 = \{w_1, w_2\}$ and R consists of relations

$R_{11} = \{(x_1, \{x_2\}), (x_2, \{x_1\})\}$

$R_{12} = \{(x_1, \{y_1\}), (x_2, \{y_3\})\}$

$R_{13} = \{(x_2, \{z_1\}), (x_3, \{z_2\})\}$

$R_{14} = \{(x_3, \{w_1\})\}$

$R_{21} = \{(y_1, \{x_1\}), (y_3, \{x_2\})\}$

$R_{22} = \{(y_2, \{y_3\}), (y_3, \{y_2\})\}$

$R_{31} = \{(z_1, \{x_2\}), (z_2, \{x_3\})\}$

$R_{41} = \{(w_1, \{x_3\})\}$

$R_{44} = \{(w_1, \{w_2\}), (w_2, \{w_1\})\}$

and the adjacency defining sets are

$\tilde{T}_{23} = \tilde{T}_{34} = \{T_1\}$

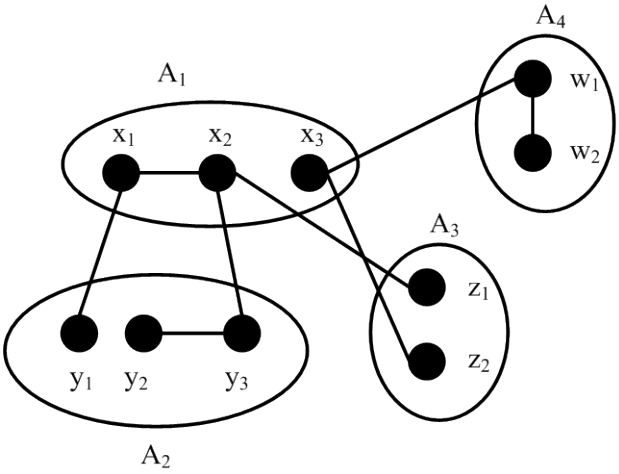


Figure 25. Graph representation for ARS of example 7.

The AdSchema for the ARS of example 7 can now be stated as follows. Let (T_A, R) be an AdSchema where $T_A = \{T_1, T_2, T_3, T_4\}$ and R consists of relations.

$R_1 = (T_1, \{T_2, T_3, T_4\})$

$R_2 = (T_2, \{T_1\})$

$R_3 = (T_3, \{T_1\})$

$R_4 = (T_4, \{T_1\})$

The AdSchema depicting the ARS of example 7 is shown in figure 26. It is worth noticing that, in this case, the minimal interrelationship combination is R_1. This means that the rest of relations can be derived based on relation R_1.

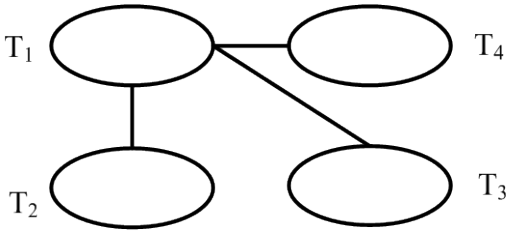


Figure 26. AdSchema for example 7.

6 UTILIZATION OF ADJACENCY MODEL IN DATAMODELING

This section discusses the utilization of AM and AdSchema in data modeling. Discussion covers modeling of graph structures such as OEM and semantic link network. This section also discusses briefly modeling relational data with graphs as an introduction for the usage of AM and AdSchema in modeling relational data. Finally, and an adjustment for the AdSchema is proposed in section "*Extended AdSchema*".

6.1 Modeling graphs with Adjacency Model

Adjacency model can be utilized in modeling graphs such as OEM –graph (McHugh, Abiteboul, Goldman, Quass & Widom: 1997: 54-57) and semantic link network. OEM –graph is used typically modeling unstructured or semistructured data. In OEM –graph (figure 27) the data is stored and represented with labeled graphs. McHugh et al. (1997: 55) state that the model does not have fixed schema and that the information of the labels can change dynamically. Its main intention is to handle incomplete and unstructured data.

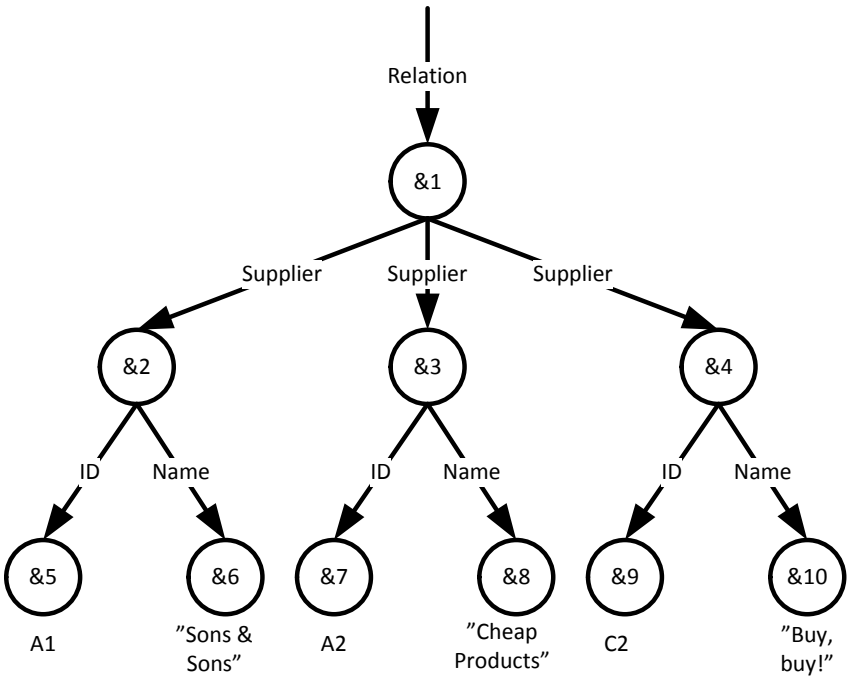


Figure 27. Supplier relation as OEM-graph.

Töyli et al. (2002b: 286-287) have developed a process for converting OEM –graph into adjacency relation system. The conversion is done from top to bottom, i.e. depth-first so that all the edges are systematically handled. The method also reduced data redundancy because the edges having the same label are considered as one type (Töyli 2006: 56). The conversion process has the following phases.

1. If there is an incoming edge into the root node, create a type of it.
2. Select an edge from the graph and create a new type of the edge.
3. Repeat step 2 until all the distinct edges are handled.
4. Select a type that has two or more distinct types adjacent to it and define it as an adjacency defining type.
5. Repeat step 4 until all such types are handled.
6. Create an adjacency between an element of an atomic type and an element of its corresponding adjacency defining type.
7. Repeat step 6 until all the elements of atomic types are handled.
8. Draw the adjacencies between the elements of the adjacency defining types according to the original graph.

The OEM –graph shown in figure 27 is converted into adjacency relation system shown in figure 28.

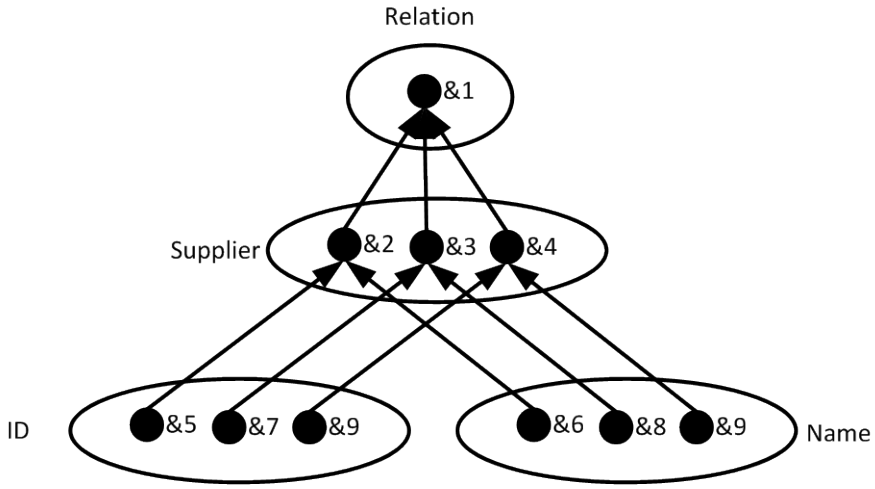


Figure 28. Adjacency relation system representation of OEM -graph.

Semantic link network is a graph-based model for representing semantic information. Semantic link network consists of nodes and edges connecting them (Zhuge 2003: 89-90). Adjacency relation system and semantic link network have some similarities that make it possible to model convert SLN into adjacency relation system. The conversion process it quite straightforward:

1. Select a node and create type of it.
2. Select an alpha link type and create type of it.

3. Assign all link elements to their corresponding types. Identify links with indices.
4. Repeat steps 1 and 2 until all the distinct nodes and link types are handled.
5. Define the alpha links as an adjacency defining type.
6. Repeat step 5 until all the links are defined as adjacency defining type.
7. Create adjacencies between elements of types and their corresponding adjacency defining types.
8. Repeat step 7 until all the adjacencies based on adjacency defining types are created.
9. If needed, create other adjacencies between elements according the original semantic link network.

Note that because, by definition, the edges of an SLN directed it would be nearly impossible to create an adjacency relation system with alpha links as adjacency defining types. However, there exists a semantic link primitive called *Reverse relation operation* (Zhuge 2005: 41-43; Zhuge, Yun, Jia & Liu: 2005: 228-229), which makes it possible to draw the symmetric adjacency relation system. So the SLN based ARS is symmetric.

Example 8. Consider a semantic link network shown in figure 29.

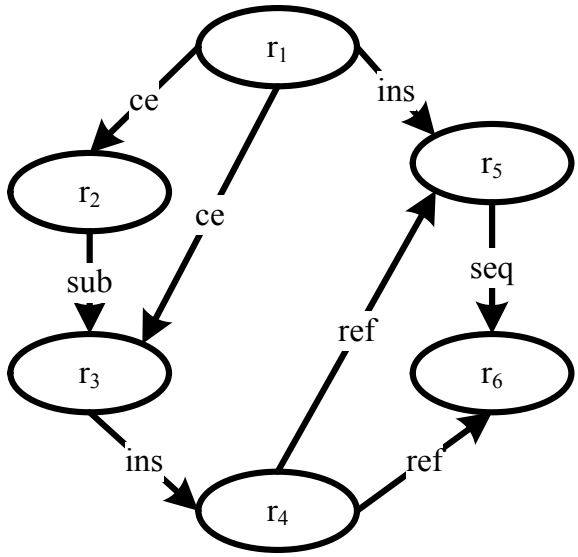


Figure 29. A simple SLN (Zhuge et al. 2005: 229).

In the first step types for each node of the graph are defined. The method reduces redundancy by creating types for unique nodes. In the second step types for the links are created. In the third step, it is necessary to represent all the links in the adjacency relation system. Each link is assigned to its corresponding type. If the SLN contains two or more links of the same type, the links must be identified

with indices. In the fourth step the adjacency defining types are defined, the nature of SLN dictates that the alpha links should be defined as adjacency defining sets. The purpose of the sixth step is to draw adjacencies between types and the corresponding element of the adjacency defining type. The adjacencies between elements are represented with undirected edges. Finally, all adjacencies are defined. Adjacency relation system representing the SLN of figure 29 is portrayed in figure 30.

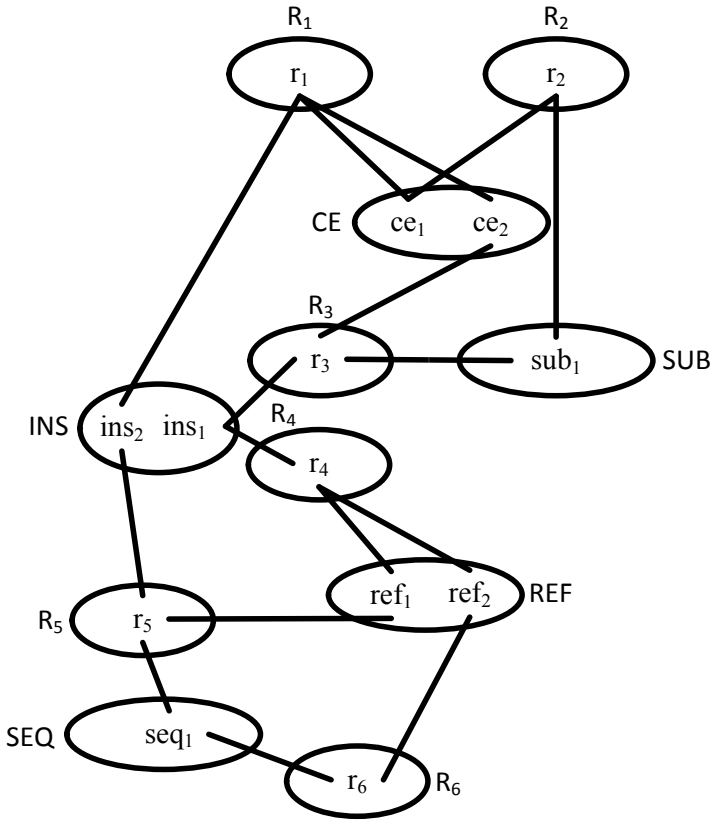


Figure 30. ARS representation of example SLN.

6.2 Graph-based modeling of the relational data

Buneman et al. (1996:1-2) presented a deterministic data model for modeling relational and semistructured data. Data model consists of labeled edges connecting the nodes of the graph. All the information is represented in the edges connecting the nodes. Edges contain information such as the name of the relation, attributes and their values. Each tuple of a relation is represented by a subgraph. For example, the *Supplier* relation can be depicted by the graph shown in figure 31.

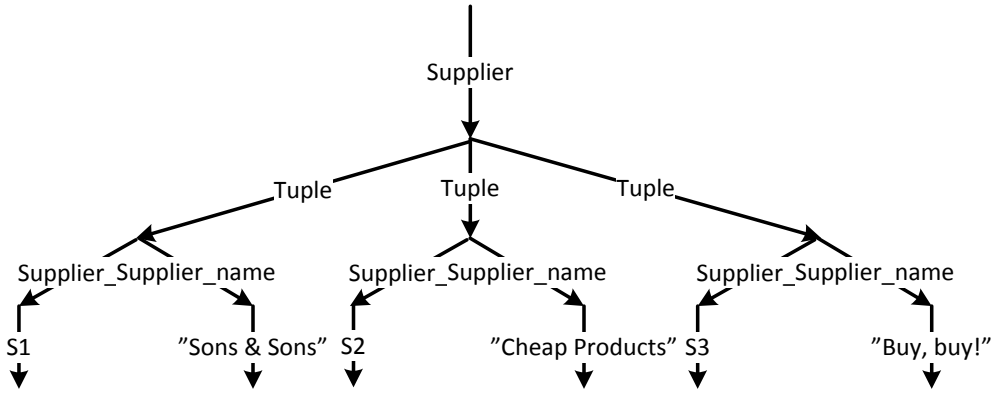


Figure 31. Supplier relation as a graph.

6.3 Modeling relational data with adjacency model

As stated in the previous section, Buneman et al. (1996:1-2) showed that, relational data can be modeled as trees. Töyli et al. (2002a: 304-305; Töyli 2006: 46-53) introduced a method for modeling relational data with adjacency model. The original design consisted of nine steps is accompanied in this work with a new step, step 3 "Assign attribute values as elements to the corresponding type".

1. Select an attribute from a table.
2. Create a corresponding type.
3. Assign attribute values as elements to the corresponding type.
4. Repeat steps 1, 2 and 3 until all the distinct tables and attributes are handled.
5. Remove the types that have been created from the foreign keys.
6. Take a primary key and define it as an adjacency defining type.
7. Repeat step 6 until all the distinct primary keys are defined as adjacency defining types.
8. Create an adjacency between an element of type and an element of its corresponding adjacency defining type (i.e. between an attribute primary key of the table).
9. Repeat step 8 until all the adjacencies are created between the elements and their adjacency defining sets.
10. Create all other wanted adjacencies between the types. For example, create adjacencies between adjacency defining sets, in order to represent dependencies.

The method produces an adjacency relation system representation of relational database. The method is applied to a simple database consisting of two relations

$PRODUCT(product\_id, product\_name, unit\_in\_stock)$ and $SUPPLIER(supplier\_id, supplier\_name)$. The Supplier-product database is shown in figure 32.

Product			
Product_id (PK)	Supplier_id (FK)	Product_name	Units_in_stock
P1	S1	9" nail	23
P2	S1	bolt	15
P3	S2	hammer	7

Supplier	
Supplier_id (PK)	Supplier_name
S1	Sons & Sons
S2	Cheap Products
S3	Buy, buy!

Figure 32. Supplier-product database.

By following Töyli's (2002 and 2006) modeling procedure, the Supplier-product database is built as an ARS. The ARS constructed by this method are considered to be symmetric. The ARS a pair (A, R) where

$A = \{product\_id, product\_name, units\_in\_stock, supplier\_id, supplier\_name\}$,

$product\_id = \{P1, P2, P3\}, product\_name = \{9" \, nail. \, bolt, hammer\}$,

$units\_in\_stock = \{23, 15, 7\}$,

$supplier\_id = \{S1, S2, S3\}$,

$supplier\_name = \{Sons\&Sons, Cheap\ Products, Buy, buy!\}$,

and R consists of relations.

$R_{product\_id,units\_in\_stock} = \{(P1, \{23\}), (P2, \{15\}), (P3, \{7\})\}$

$R_{product\_id,product\_name} = \{(P1, \{"9" \, nail"\}), (P2, \{"bolt"\}), (P3, \{"hammer"\})\}$

$R_{units\_in\_stock,product\_id} = \{(23, \{P1\}), (15, \{P2\}), (7, \{P3\})\}$

$R_{product\_name,product\_id} = \{("9" \, nail", \{P1\}), ("bolt", \{P2\}), ("hammer", \{P3\})\}$

$R_{supplier\_id,supplier\_name}$
$= \{(S1, \{"Sons \ \& \ Sons"\}), (S2, \{"Cheap\ Products"\}), (S3, \{"Buy, buy!\ "\})\}$

$R_{product\_id,supplier\_id} = \{(P1, \{S1\}), (P2, \{S1\}), (P3, \{S2\})\}$

$R_{supplier\_id,product\_id} = \{(S1, \{P1, P2\}), (S2, \{P3\})\}$

Adjacency defining type represents the key attributes of the database. The adjacency defining types for the ARS are defined as follows.

$\tilde{T}_{units\_in\_stock,product\_name} = \tilde{T}_{units\_in\_stock,supplier\_id} = \tilde{T}_{product\_name,supplier\_id} = \{T_{product\_id}\}$ and

$\tilde{T}_{supplier\_name,product\_id} = \{T_{supplier\_id}\}$

The results of the modeling method can be visualized as an ARS graph. The ARS representing the Supplier-product database is depicted by the graph shown in figure 33. In the ARS, each attribute (column) is represented as an entity set and the values of the attributes are depicted by the elements of the corresponding sets. Dependencies or relationships are represented as edges connecting the elements. The expressions of relational database elements in ARS graph will be discussed in more detailed level in "*7 ANALYZING THE ADJACENCY RELATION SYSTEM REPRESENTATION OF RELATIONAL DATABASE*".

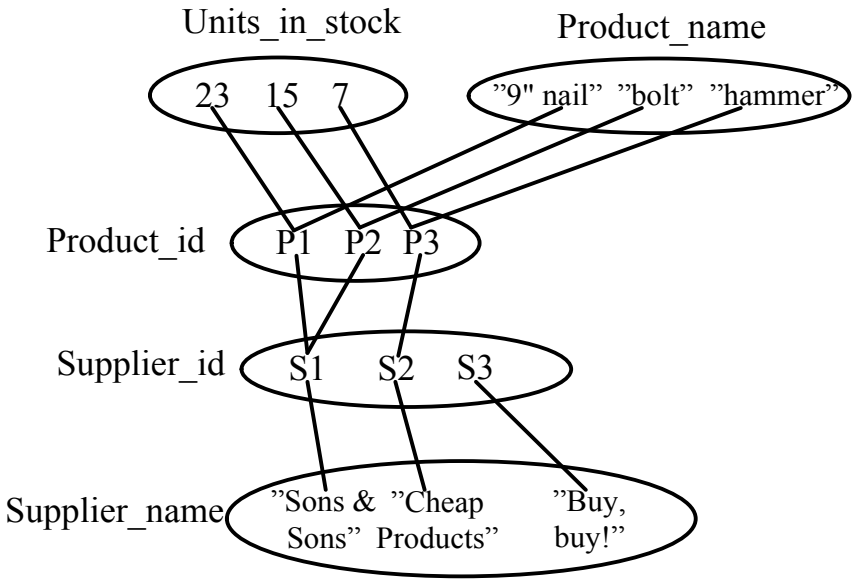


Figure 33. An ARS representation of the supplier-product database.

6.4 Modeling relational database schema with AdSchema

In the previous section, a modeling method (Töyli 2002 and 2006) from a relational database to an adjacency relation system was discussed. This study aims to extend the application areas of Adjacency Model so that it could be possible to analyze graphs and determine if a given graph contains features suggesting that the graph represents relational database or has such features that it could be used as a foundation for a data repository construction.

Relational databases can be visualized as an adjacency relation system. Typically, this kind of ARS contains multiple components and the basic structure of the components can be depicted with an AdSchema. If the modeled database contained dependencies such as one-to-many or many-to-many, it is possible that graph components have some variations in their structures. For example, the ARS based graph shown in figure 33 is made up of three components $\{"Buy, buy!", S3\}$, $\{"Cheap\ Products", S2, P3, 7, "hammer"\}$ and $\{"Sons\ \&\ Sons", S1, P1, P2, 23, 15, "9"\ nail", "bolt"\}$. In the case of ARS depicting relational database the components tend to have different structures.

In section *"5.2 Modeling adjacency relation systems with AdSchema"* was proposed a method, which makes it possible to simplify the structure of a graph representing relational database. The basic structure of an ARS, i.e. AdSchema can be produced from database schema (figure 34). The conversion process has nine steps. The result of the conversion process is portrayed in figure 35.

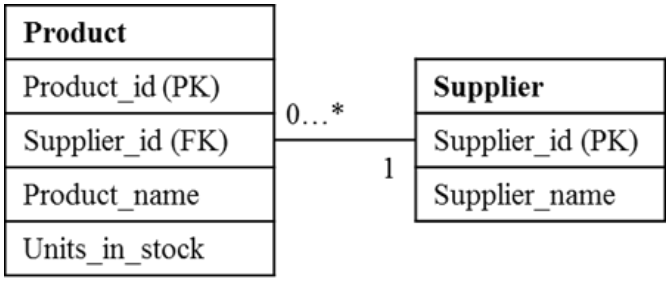


Figure 34. Database schema.

The nine phased method is an adaptation of Töyli's (2002 and 2006) method.

1. Select an attribute from a relation schema.
2. Create a corresponding type.
3. Repeat steps 1 and 2 until all the distinct tables and attributes are handled.
4. Remove the types that have been created from the secondary keys.
5. Take a primary key and define it as an interrelationship defining type.
6. Repeat step 5 until all the distinct primary keys are defined as interrelationship defining types.
7. Create an adjacency between a type and its corresponding interrelationship defining type.
8. Repeat step 7 until all the interrelationships are created between the elements and their interrelationship defining sets.

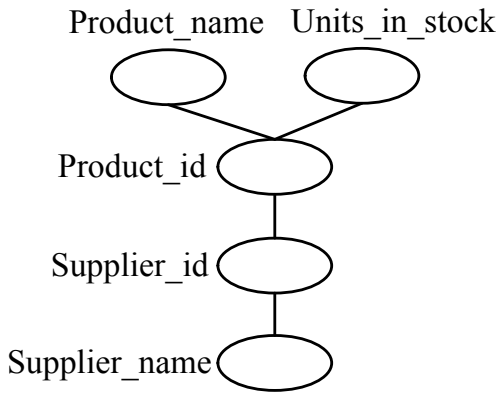


Figure 35. AdSchema representation of the database schema.

In figure 35 (above) is shown an AdSchema (T_A, R) where $T_A = \{product\_name, units\_in\_stock, product\_id, supplier\_id, supplier\_name\}$, R contains relations $\{R_1, R_2, R_3, R_4, R_5\}$. The relations are defined as follows.

$R_1 = (product\_id, \{product\_name, units\_in\_stock, supplier\_id\})$

$R_2 = (supplier\_id, \{supplier\_name, product\_id\})$

$R_3 = (product\_name, \{product\_id\})$

$R_4 = (units\_in\_stock, \{product\_id\})$

$R_5 = (supplier\_name, \{supplier\_id\})$

Note that the method described above produces an adjacency schema depicting only the relationship between types. It does not provide any information about dependencies and other constraints, that database schema might have expressed. By adopting the notations of the network data model, some information about dependencies between types could be expressed.

6.5 Extended AdSchema

The AdSchema approach considers only symmetric relationships between types. It allows traversing the data structure in both directions. Symmetric AdSchema does not contain any information about the relationships between the element types. (Töyli 2006: 62.)

If the AdSchema is used for representing the schemas of structured data such as relational data, there should be some method for conveying information about the nature of relationships between types. By assigning directions to the edges

depicting the interrelationships between the types, information about the nature of relationships can be represented. The directions and meanings for the edges are adopted from the network model.

In the network data model, one-to-one relationship is denoted by a bidirectional edge. One-to-many relationship is denoted by a directed edge, and many-to-many relationships are denoted by an undirected edge. Let us update the graph representation of the AdSchema by assigning the directions to the edges connecting the nodes representing types (figure 36).

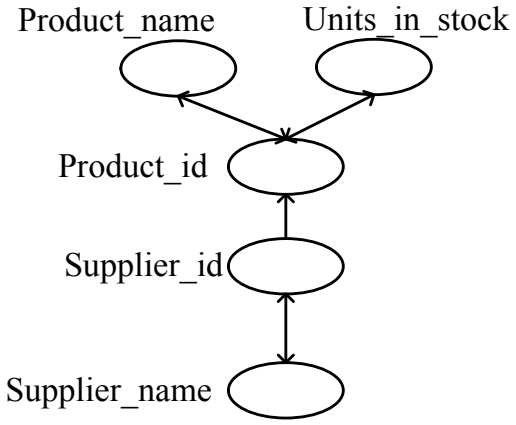


Figure 36. Extended AdSchema with directed edges.

Now the graph the graph conveys information about the database schema. The meanings of arrows can be read as follows. Each element in *supplier_id* is associated with exactly one element in *supplier_name* and vice versa. Furthermore, *supplier_id* type elements can be associated with multiple elements belonging to *product_id* type; however, the elements in *product_id* type are associated with only one element in *supplier_id* type. Finally, each element in *product_id* is associated with exactly one element in *product_name* as well with one element in *unit_in_stock* type and vice versa.

The relation notations of the AdSchema need to be revised. So, in the extended AdSchema the edges representing relationships between elements are defined as follows.

Bidirectional edge is represented with two relations both containing the elements incident to the edge. For example, consider the edge from *Product_id* to *Product_name*. It is portrayed by two relations R_1 and R_2. The relations are defined as follows.

$R_1 = (Product\_id, \{Product\_name\})$

$R_2 = (Product\_name, \{Product\_id\})$

Directed edge is represented by relation, which contains both elements associated with the edge. Furthermore, the element of the undirected end of the edge is assigned as the first element of the relation. Consider the edge from $Supplier\_id$ to $Product\_id$. It is depicted as relation R_5.

$R_5 = (Supplier\_id, \{Product\_id\})$

The relation definitions for the extended AdSchema of figure 36 are constructed as follows. Consider the extended AdSchema. It is a pair (T_A, R) where $T_A = \{Product\_name, Units\_in\_stock, Product\_id, Supplier\_id, Supplier\_name\}$, R contains relations $\{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$. The relations are defined as follows.

$R_1 = (Product\_id, \{Product\_name\})$

$R_2 = (Product\_name, \{Product\_id\})$

$R_3 = (Product\_id, \{Units\_in\_stock\})$

$R_4 = (Units\_in\_stock, \{ Product\_id\})$

$R_5 = (Supplier\_id, \{ Product\_id\})$

$R_6 = (Supplier\_id, \{Supplier\_name\})$

$R_7 = (Supplier\_name, \{Supplier\_id\})$

The many-to-many relationship is deprecated in the relational model. However, if an AdSchema contains an undirected edge, there should be a way to depict the relation. Let us change the edge connecting $Product\_id$ and $Supplier\_id$ to undirected one (figure 37). The relation representing the edge contains two element pairs both containing the elements incident to the edge.

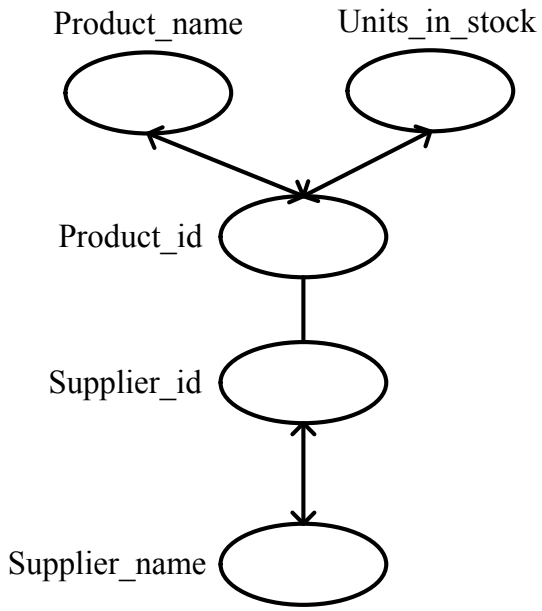


Figure 37. Extended AdSchema with undirected edge.

Now the relations for the AdSchema shown in figure 37 are defined as follows. The relation R_5 indicates the existence of many-to-many relationship between the elements of the relation.

$R_1 = (Product\_id, \{Product\_name\})$

$R_2 = (Product\_name, \{Product\_id\})$

$R_3 = (Product\_id, \{Units\_in\_stock\})$

$R_4 = (Units\_in\_stock, \{Product\_id\})$

$R_5 = (\{Supplier\_id, Product\_id\}, \{Product\_id, Supplier\_id\})$

$R_6 = (Supplier\_id, \{Supplier\_name\})$

$R_7 = (Supplier\_name, \{Supplier\_id\})$

7 ANALYZING GRAPH REPRESENTATION OF RELATIONAL DATABASE

In this section is introduced a framework for identifying the essential elements of a relational database from an ARS-based graph. The preliminary identification of the element is based on the ideas and definitions provided by Wanne (1998) and Töyli (2002 and 2006). Later on, the elements are identified by utilizing the concepts of graph theory.

Wanne (1998: 5) stated that the dependencies between the relations in relational databases are implemented with keys. The keys correspond to the adjacency defining entity types. Furthermore, it is stated that type represents the attributes and elements of the types represent the values of the attributes. The paths of adjacencies provide some view on tuples of the relational model. (Töyli 2002: 55-59.)

Let us identify the instances of attributes and attribute values from an adjacency relation system based graph, which represents Supplier -relation. The relation is modeled as an ARS by the Töyli's (2002 and 2006) method (figure 38).

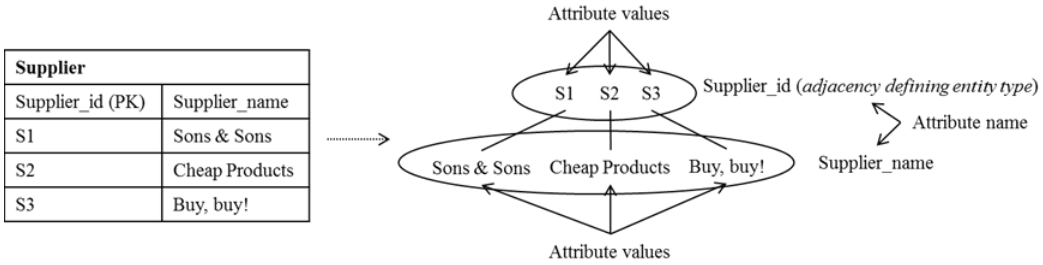


Figure 38. Attributes and their values in ARS based graph.

In the figure above it is shown a database table and an ARS-based graph representing it. The attributes and their values are marked in the figure. The tuples of the table are relatively easy to recognize from the ARS by utilizing the concept of walk. In this case, the ARS has three distinct walks $\langle "S1", "Sons\&Sons" \rangle$, $\langle "S2", "Cheap\ Product" \rangle$ and $\langle "S3", "Buy, buy!" \rangle$. The walks are ordered such that the elements of the adjacency defining type are the first elements of the walks.

Typically the ARS is made up of multiple types with multiple elements assigned to them. Obviously, this means that ARS based graphs contain multiple vertices, and the vertices representing the values of key attributes have multiple leaves. This means that the walks, depicting the tuples, visit the key vertices several times. So, a tuple should be seen as a set of distinct vertices belonging to the walk from the key vertex to its leaves.

7.1 Identifying dependencies, tuples, and relations

Wanne (1998) and Töyli (2002 and 2006) have stated in their works that adjacency model and relational model have conceptual similarities, which made possible to utilize the adjacency model in modeling relational databases. Furthermore, certain elements of the relational model have identifiable features in an adjacency relation system. However, previous studies did not provide systematic tools for the element identification. Information about the dependencies (relationships) between relations of the database is carried out with key attributes. The adjacency defining entity types and their elements represent the key attributes and their values. Thus, it can be assumed that in the case of ARS representing relational database, the elements of a regular type are associated with one element of the adjacency defining entity type.

Since the adjacencies between elements of adjacency defining types represent the relationships (dependencies) of relational database, it can be assumed that the adjacencies in ARS can be more complex. For example, in figure 39 element $S1$ is adjacent to $P1$ and $P2$ while $S3$ does not have adjacent elements. Obviously, when the number of elements in ARS grows the adjacencies have more diverse and complex forms.

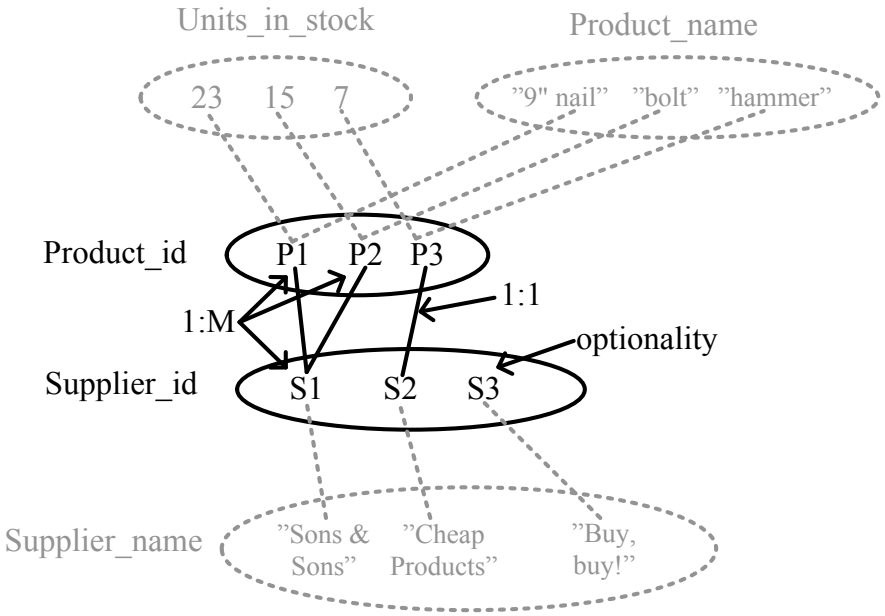


Figure 39. Expressions of dependencies in ARS.

In the figure 39 is shown an ARS that represents a simple database for products and suppliers. The types *product_id* and *supplier_id* are defined as adjacency defining types. The adjacencies between elements of these types contain infor-

mation about the dependencies of the database tables. Consider the relation $R_{supplier\_id,product\_id} = \{(S_1, \{P_1, P_2\}), (S_2, \{P_3\})\}$. The analysis of the relation and graph reveal the following.

- $(S_1, \{P_1, P_2\})$ imply the existence of one-to-many relationship
- $(S_2, \{P_3\})$ imply the existence of one-to-one relationship
- the element S_3 does not have adjacenct elements in type $Product\_id$, which implies that in the database the relationship is optional

In the case of an ARS representing relational database, the tuples of the database are made up of the distinct elements belonging to the walk from the element of the adjacency defining type to its leaf. The ARS shown in figure 39 contains the following tuples.

$\langle P_1, 23{,}9"\ nail \rangle$

$\langle P_2, 15, bolt \rangle$

$\langle P_3, 7, hammer \rangle$

$\langle S_1, Sons\&Sons \rangle$

$\langle S_2, Cheap\ Products \rangle$

$\langle S_3, Buy, buy! \rangle$

Finally, the tuples are assigned to relations. In this case tuples are assigned to the relations according to type definitions. So, the elements of *product_id* and their adjacent leaves form a relation and its tuples and respectively the elements of *supplier_id* and their adjacent leaves form the other relation and its tuples.

7.2 Extended analysis of ARS based graphs

In the previous section was proposed a simple method for identifying the basic concepts of relational database from an ARS. The method is applicable to relatively small and simple adjacency relation systems. The analyzes of the ARS based graph utilize graph theory concepts such as degree of a vertex, leaf vertex and vertex dominating set of the graph.

Since the ARS-graph is based on relational database the properties of the vertices, and especially the vertices representing the values of the primary keys, can be easily examined. Because the ARS graph (figure 39) consists of three components having different structures, it is useful to complement the analysis by ex-

amining the basic structure and relationships between types of the ARS. The AdSchema is a basic representation of the ARS. It describes all the types and their interrelationships. The AdSchema for the ARS is shown in figure 40.

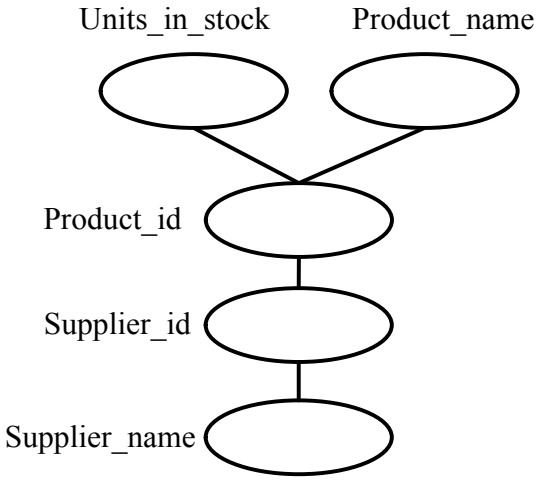


Figure 40. AdSchema of an ARS.

Tables 4 and 5 list the findings of the ARS analysis (figure 39) and AdSchema analysis (figure 40). The analyzes examined the degree of vertices, vertices that have leaves, and (minimal) vertex dominating set of the graphs.

Table 4. Properties for ARS vertices/elements.

Vertex/Element	Degree	Has leaves	Member of dominating set
23	1	No	No
15	1	No	No
7	1	No	No
9" nail	1	No	No
bolt	1	No	No
hammer	1	No	No
P1	3	Yes	Yes
P2	3	Yes	Yes
P3	3	Yes	Yes
S1	3	Yes	Yes
S2	2	Yes	Yes
S3	1	Yes	Yes
Sons & sons	1	No	No
Cheap Products	1	No	No
Buy, buy!	1	Yes	No

Table 5. Properties for AdSchema vertices/types.

Vertex/Type	Degree	Has leaves	Member of dominating set
Units_in_stock	1	No	No
Product_name	1	No	No
Product_id	3	Yes	Yes
Supplier_id	2	Yes	Yes
Supplier_name	1	No	No

Based on the results shown in the tables 4 and 5, it can be stated that vertices representing the values of key attributes and as well types representing key attributes have common properties, which are:

– they belong to the vertex dominating set of the graph,
– vertices also have leaves,
– and their degree is at least 2.

In the appendix 1 is depicted analysis results for properties of vertices in 15 different AdSchemas. Preliminary correspondences between the relational model and adjacency model were put together and listed in the table 6.

Table 6. Basic properties for relational model concepts expressed in Adjacency Model or AdSchema.

Relational model	**Adjacency model/AdSchema**
Database	Adjacency relation system
Database schema	AdSchema
Attribute	Entity type (AM and AdSchema)
Attribute values	Elements of the entity type (AM)
Key attribute	Entity type – adjacency defining entity type (AM) – interrelationship defining type (AdSchema) – has degree of at least 2 – has leaf nodes – dominates graph.
Relation	Set of distinct vertices that belong to a walk in AdSchema from the interrelationship defining entity type to adjacent leaf entity types.
Tuple	Set of distinct vertices that belong to a walk in ARS from the element of the adjacency defining entity type to the elements of the adjacent type.
One-to-one dependency	In the adjacency model, adjacency between the elements of the adjacency defining entity types. In the AdSchema, adjacency between interrelationship defining type.
One-to-many dependency	Adjacency between the elements of the adjacency defining entity types. For example, so that $x_1, x_2 \in Ad(y_1)$ ja $x_3 \in Ad(y_2)$. AdSchema does not convey information about complex dependencies.
Optionality	Adjacency defining entity type includes both elements that are adjacent to the element of other adjacency defining entity type and elements that are not adjacent to any elements belonging to any adjacency defining entity type. AdSchema does not convey information about optionality constraints.

7.3 Database reconstruction

In this section is proposed a framework for reconstructing relational database from an ARS graph. Reconstructing the database requires the identification of

the elements of the database but more importantly the different types of dependencies must be identified. The identification of dependencies such as one-to-many and many-to-many are covered with examples. The analysis and reconstruction process done in this section has the following features. Finally, a detailed example of database reconstruction is given in *7.3.3 Reconstructing the database.*

Preliminary steps

1. Model relational database with ARS
2. Create corresponding AdSchema

Analysis and reconstruction steps

3. Recognize types and elements representing key attributes and their values
4. Construct tuples
5. Construct relations
6. Define dependencies and possible constraints for relations.

In order to reconstruct the database tuples, relations and dependencies have to be identified. Consider the properties of the elements of ARS and types of AdSchema listed in tables 4 and 5. According to properties listed in table 6 assign elements representing the values of key attributes to set A and assign other elements to set B.

Set A contains elements $P1, P2, P3, S1, S2, S3$ and respectively set B consist of elements 23,15,7,"9" nail", "bolt", "hammer", "Sons&sons", "Cheap Products", "Buy, buy!". Furthermore, the sets A and B can divided into subsets, in this case the subset division conforms the type definition of adjacency relation system, A_1, A_2, where $A_1 = \{P1, P2, P3\}$ and $A_2 = \{S1, S2, S3\}$ and respectively subsets of B are defined as follows B_1, B_2, B_3 where $B_1 = \{23,15,7\}, B_2 = \{"9\ nail", "bolt", "hammer"\}$ and $B_3 = \{"Sons\ \&\ sons", "Cheap\ Products", "Buy, buy!"\}$. Now the neighboring elements x and y ($x \in A \wedge y \in B$) has to be recognized. Neighboring elements and subset division are shown in table 7.

Table 7. Adjacent elements in sets A and B.

		B_1	B_2	B_3
A_1	P1	23	9” nail	-
	P2	15	bolt	-
	P3	7	hammer	-
A_2	S1	-	-	Sons & sons
	S2	-	-	Cheap Products
	S3	-	-	Buy, buy!

After the identification of adjacencies between the elements in sets A and B the tuples are constructed. Based on the adjacencies listed in table 7 it can be said that the ARS representing the database contains six instances of tuples which are listed below. The tuples are organized so that elements of the set A are the first elements of tuples.

$t_1 = \{P1,23,"9"\ nail"\}$

$t_2 = \{P2,15,"bolt"\}$

$t_3 = \{P3,7,"hammer"\}$

$t_4 = \{S1,"Sons\ \&\ Sons"\}$

$t_5 = \{S2,"Cheap\ Products"\}$

$t_6 = \{S3,"Buy, buy!\ "\}$

Based on the type definitions provided by the adjacency relation system, the tuples are assigned to relations. The relations are composed as follows $r_1 = \{t_1, t_2, t_3\}, r_2 = \{t_4, t_5, t_6\}$. It is worth noticing that the type definitions are only available in the case of ARS representing a database.

If the type definitions are not available, tuples are assigned to relations based on their properties. The properties can be, for example, number of elements in tuple and element types.

The next step in database reconstruction is to analyze adjacencies between the elements subsets of A. For this purpose an adjacency matrix is built. The aim is to find adjacent elements x and x’ in set A $(x, x' \in A \ \wedge x \neq x')$. If the elements are adjacent the value of the cell is 1, otherwise the value is 0.

Table 8. Adjacency matrix for the elements of set A.

		A_1			A_2	
		P1	**P2**	**P3**	**S1**	**S2**
A_1	**P1**	0	0	0	1	0
	P2	0	0	0	1	0
	P3	0	0	0	0	1
A_2	**S1**	1	1	0	0	0
	S2	0	0	1	0	0
	S3	0	0	0	0	0

In table 8 elements $P1$, $P2$ and $P3$ belong to the set A_1 and elements $S1$, $S2$ and $S3$ are members of the set A_2. In table 9 the adjacencies between element belonging these set are analyzed in order to determine the type dependency between the elements. The row-wise examination provides information how many adjacent elements the elements of A_1 have in A_2. The column-wise examinations show the number of adjacent elements the elements of A_2 have in A_1. Based on the adjacencies the dependency type can be concluded.

Table 9. Adjacency of elements in A_1 and A_2.

		A_2			Dependency
		S1	**S2**	**S3**	
A_1	**P1**	1	0	0	1:1
	P2	1	0	0	1:1
	P3	0	1	0	1:1
Dependency		1:M	1:1	Optional	

The elements of A_1 represent the values of key attributes of relation r_1 and the elements of A_2 represent the values of key attributes of relation r_2.

In table 9 is shown that elements of A_1 are adjacent to exactly one element of A_2. So, it can be concluded that between relations r_1 and r_2 exists one-to-one relationship. Table 9 also shows that element $S1$ is adjacent to elements $P1$ and $P2$, and element $S2$ is adjacent to $P3$. So, there exist one-to-one and one-to-many dependencies between the relations. Moreover, the element $S3$ does not have any neighbors in set A_1 i.e. relation r_1, it can be assumed that the relationship between r_1 and r_2 is optional.

Based on the adjacencies discussed above, it can be concluded that the dependency between r_2 and r_1 (table 10) can be denoted as one-to-many with optionality constraint. One-to-many was chosen as prevailing dependency definition, since it does not exclude the one-to-one dependency.

Table 10. Dependency between r_2 and r_1.

r_2	r_1
1	0..*

Since there exists one-to-many dependency between given relations, the key attribute of the "one-end" of the dependency is assigned as a foreign to the "many-end" of the dependency. So, the key attribute of r_2 is duplicated as foreign key to r_1. Furthermore, in this case the optionality supports the foreign key definition.

The ARS depicted in figure 39 is converted into a relational database with six tuples, and two relations. Between the relations is one-to-many dependency with optionality constraint. The reconstructed database is shown in figure 41.

r_1			
P1	23	9" nail	S1
P2	15	Bolt	S1
P3	7	Hammer	S2

r_2	
S1	Sons & Sons
S2	Cheap Products
S3	Buy, buy!

Figure 41. Reconstructed database.

In this section was given a small example of converting an ARS into a relational database. The example database discussed did not contain complex dependencies. The future sections of this chapter will discuss the instances of many-to-many dependencies in ARS based graph representing a relational database.

7.3.1 Complex dependencies in adjacency relation system

This section discusses the instances of many-to-many dependencies in adjacency relation system based graph. The many-to-many dependencies (denoted as M:N) are deprecated in the relational model. The possible M:N –dependencies are implemented in relational model by relations called "relationship relations". Relationship relation contains two foreign keys (see, for example, figure 42., *Order_Product* -relation).

The instances of M:N –dependencies in ARS are recognized by analyzing the adjacencies between elements of different types. For the analyzing purposes, an adjacency matrix is constructed. Row sums and column sums of the matrix show information about the dependencies between the elements. If the row/column sum is greater than two, it can be read as an indication of M:N dependency, if the sum is one, it implies the existence of a one-to-one dependency and if the sum is zero, it suggests the that the dependency is optional. The following example provides a detailed examination of the instances of many-to-many dependencies in ARS.

Example 9. Consider a simple database for customer orders. There is a rule for the database that states that one product can be in several orders, and order may contain several products. Database contains the relations shown in figure 42.

Customer	
Customer_id (PK)	Customer_name
C1	Jack Smith
C2	Julia Smith
C3	John Doe
C4	Jane Doe

Product		
Product_id (PK)	Product_name	Units_in_stock
P1	9" nail	23
P2	bolt	15
P3	hammer	7
P4	bucket	9

Order		
Order_id (PK)	Customer_id (FK)	Order_date
O1	C1	5.12.2012
O2	C2	28.1.2013
O3	C4	3.2.2013
O4	C4	5.3.2013

Order_Product		
Order_id (FK)	Product_id (FK)	Units_per_order
O1	P1	7
O1	P2	8
O2	P1	5
O2	P2	7
O2	P3	2
O3	P4	3
O4	P1	5

Figure 42. Customer order database.

The database is converted into the adjacency relation system shown in figure 43. Note that the set *Units_per_order* contains similar elements (7 twice and 5 twice), because the conversion method does not remove similar attribute values, since they are assigned to different tuples of the modeled database. The interrelationships between the types of the adjacency relation system are portrayed with AdSchema (figure 44).

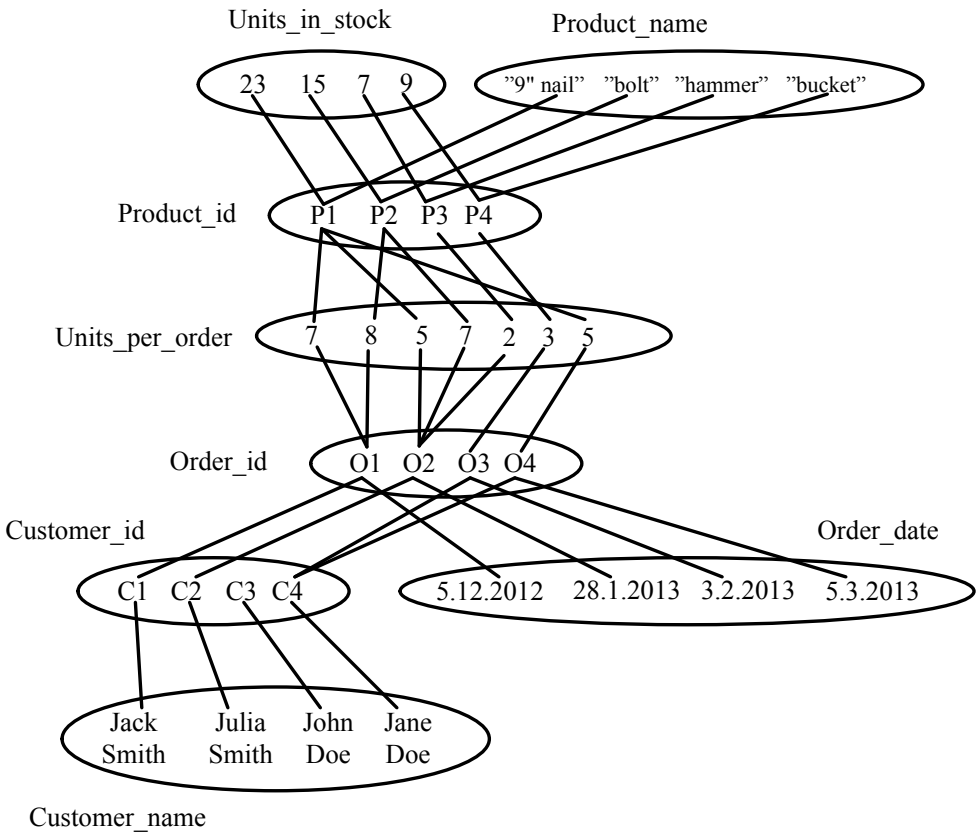


Figure 43. Adjacency relation system for Customer-order database.

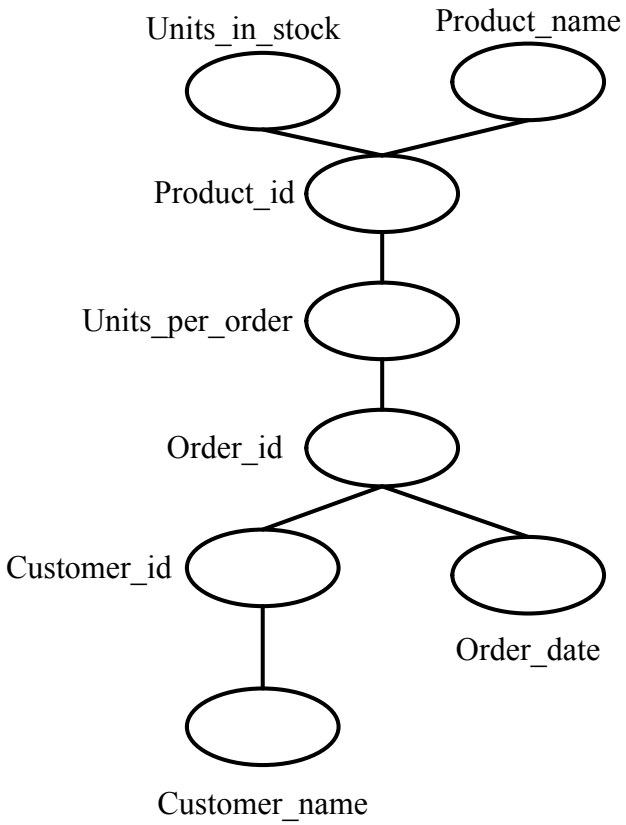


Figure 44. AdSchema for Customer-order database.

The ARS and AdSchema reveal that the type *Units_per_order* and its elements do not fully satisfy the criteria set for key attribute types and their values. Each element in the type *Units_per_order* has the following properties: element have degree of 2, element do not belong to vertex dominating set and element do not have leaves. Note that in the original database relation *Order_Product* has an attribute *Units_per_order*, which contains same values. When the database is converted into adjacency relation system, the duplicate values are not removed but they are modeled as elements of the given type.

Let us examine the adjacencies between the elements of type *Product_id* and *Units_per_order* and also *Order_id* and *Units_per_order*. The adjacencies between the elements are shown in tables 11 and 12. The row and column sums shown in tables indicate the type of dependency.

Table 11. Adjacency of elements between types *Product_id* and *Units_per_order*.

		Units_per_order								
		7	**8**	**5**	**7**	**2**	**3**	**5**	**Row sum**	**Dependency**
Product_id	**P1**	1	0	1	0	0	0	1	3	1:M
	P2	0	1	0	1	0	0	0	2	1:M
	P3	0	0	0	0	1	0	0	1	1:1
	P4	0	0	0	0	0	1	0	1	1:1
	Column sum	1	1	1	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1	1:1	1:1	1:1		

Table 12. Adjacencies between elements of types *Order_id* and *Units_per_order*.

		Units_per_order								
		7	**8**	**5**	**7**	**2**	**3**	**5**	**Row sum**	**Dependency**
Order_id	**O1**	1	1	0	0	0	0	0	2	1:M
	O2	0	0	1	1	1	0	0	3	1:M
	O3	0	0	0	0	0	1	0	1	1:1
	O4	0	0	0	0	0	0	1	1	1:1
	Column sum	1	1	1	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1	1:1	1:1	1:1		

Tables 11 and 12 show that there exists one-to-many dependencies between elements of *Product_id* and *Units_per_Order* and also between *Order_id* and *Units_per_Order*. Since it is known that database is defined so that it contains relationship relation, it can be concluded that the adjacencies represented in tables 11, and 12 express the many-to-many dependency. Furthermore, because the relational model deprecates the many-to-many dependency, it can be deduced that the reconstructed database should have relationship relation and that the relation would consist of *Product_id* and *Order_id* as foreign keys and *Units_per_Order* as an attribute of the relation.

Example 10. Consider a simple database for customer orders with a rule which states that one product can be in several orders, and order may include many products. In this example, the relationship relation (*Order_Product*) does not contain any extra information. It contains the foreign keys. Database consists of the relations shown in figure 45.

Customer	
Customer_id (PK)	Customer_name
C1	Jack Smith
C2	Julia Smith
C3	John Doe
C4	Jane Doe

Product		
Product_id (PK)	Product_name	Units_in_stock
P1	9" nail	23
P2	bolt	15
P3	hammer	7
P4	bucket	9

Order_Product	
Order_id (FK)	Product_id (FK)
O1	P1
O1	P2
O2	P1
O2	P2
O2	P3
O3	P4
O4	P1

Order		
Order_id (PK)	Customer_id (FK)	Order_date
O1	C1	5.12.2012
O2	C2	28.1.2013
O3	C4	3.2.2013
O4	C4	5.3.2013

Figure 45. Customer order database with changed Order_Product -relation.

ARS and AdSchema (figures 45 and 46) represent the elements and structure of the database of example 10.

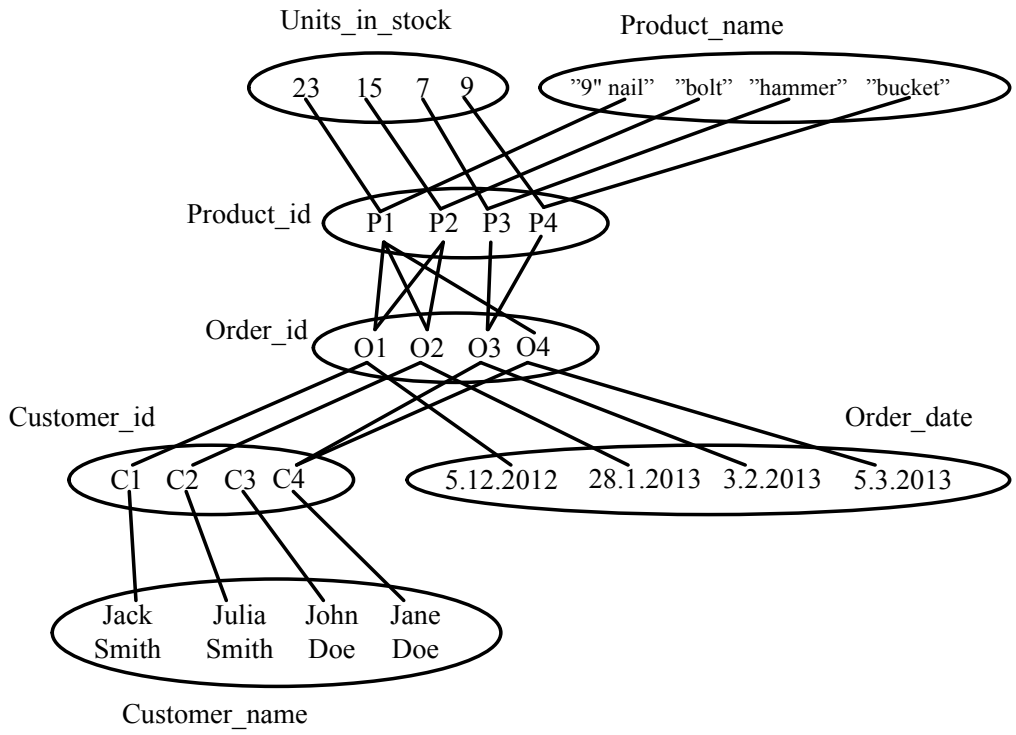


Figure 46. ARS for Customer-Order database.

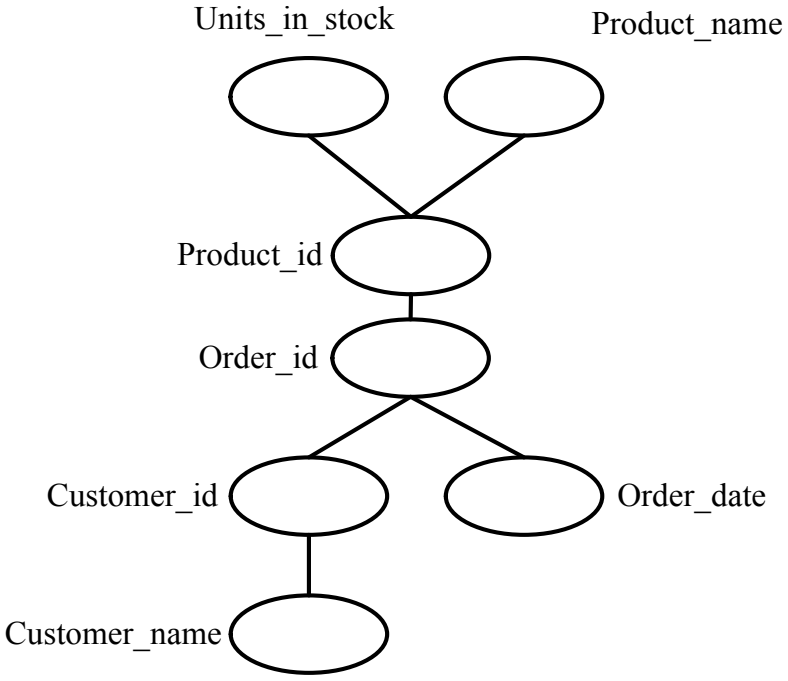


Figure 47. AdSchema for Customer-Order database.

Both ARS and AdSchema show that types *Customer_id*, *Product_id* and *Order_id* and their elements satisfy the requirements for the key attributes. From the fig-

ure 46 it can be seen that adjacencies between the elements of *Product_id* and *Order_id* differ from other adjacencies. Namely, the elements are more interconnected. Let us examine the adjacencies between the elements of type *Product_id* and *Order_id*. The adjacencies between the elements are shown in table 13.

Table 13. Adjacencies between *Product_id* and *Order_id*.

		Order_id					
		O1	**O2**	**O3**	**O4**	**Row sum**	**Dependency**
Product_id	**P1**	1	1	0	1	3	1:M
	P2	1	1	0	0	2	1:M
	P3	0	0	1	0	1	1:1
	P4	0	0	1	0	1	1:1
	Column sum	2	2	2	1		
	Dependency	1:M	1:M	1:M	1:1		

The row and column sums in table 13 reveal that multiple elements in types *Product_id* and *Order_id* are adjacent to each other. Since, by definition the database has relationship relation, it can be concluded that the observations on the table 13 express the many-to-many dependency. Since the many-to-many dependencies are deprecated in the relational model, the observations will lead to the creation of relationship relations, which contains *Product_id* and *Order_id* as foreign keys.

Example 11. Consider a simple database for customer orders (figure 48). There is a rule for the database that states that one product can be in several orders, and order may contain several products. Let us assume that the Order-relation contains only primary key and foreign key. The modeling rules state that the types created from foreign keys are removed from the ARS. This situation causes challenges when recognizing the elements of the database from ARS based graph.

Customer	
Customer_id (PK)	Customer_name
C1	Jack Smith
C2	Julia Smith
C3	John Doe
C4	Jane Doe

Product		
Product_id (PK)	Product_name	Units_in_stock
P1	9” nail	23
P2	bolt	15
P3	hammer	7
P4	bucket	9

Order	
Order_id (PK)	Customer_id (FK)
O1	C1
O2	C2
O3	C4
O4	C4

Order_Product		
Order_id (FK)	Product_id (FK)	Units_per_order
O1	P1	7
O1	P2	8
O2	P1	5
O2	P2	7
O2	P3	2
O3	P4	3
O4	P1	5

Figure 48. Customer order database.

ARS and AdSchema (figures 49 and 50) show that *Units_per_Order* and *Order_id* type do not meet the criteria set for the key attribute types, because they do not have leaves and do not belong to the vertex dominating set of the graph. By definition it is known that *Order_id* type should be considered as type for key attribute and type *Units_per_order* indicates the relationship relation. So, let us examine closely the adjacencies between types *Product_id*, *Units_per_order*, *Order_id*, *Customer_id* and *Customer_name*. The aim of the examination is to provide understanding how the exceptional dependencies of the database (figure 48) are expressed in an ARS. Note that the set *Units_per_order* contains similar elements (7 twice and 5 twice), because the conversion method does not remove similar attribute values, since they are assigned to different tuples of the modeled database.

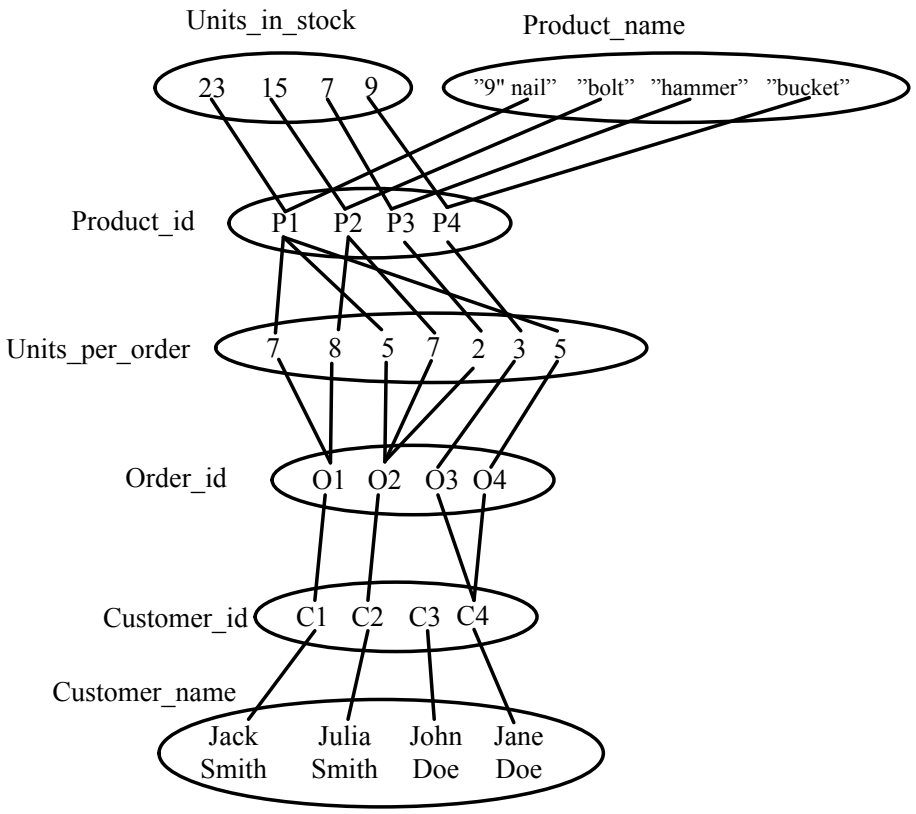


Figure 49. Adjacency relation system for Customer order database.

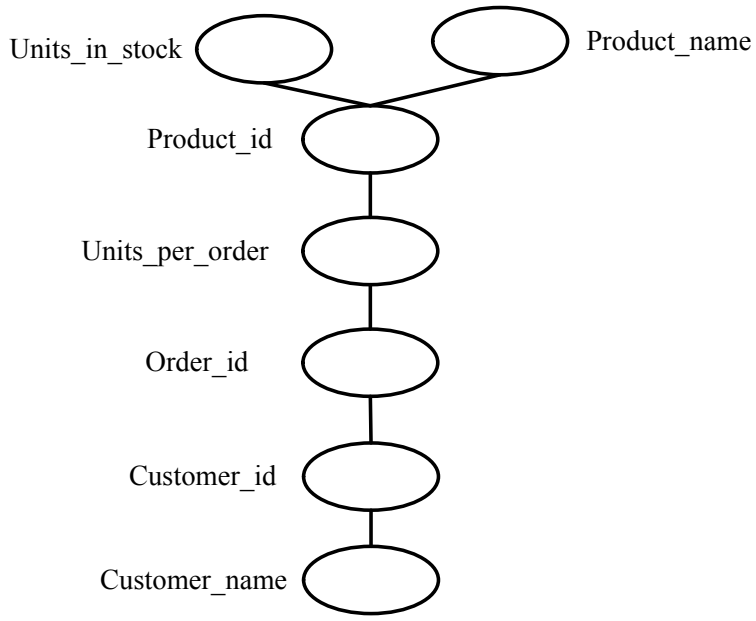


Figure 50. AdSchema for Customer order database.

The adjacencies between types *Product_id*, *Units_per_order* and *Order_id* are similar to adjacencies of example 9 (see table 13). So, it can be assumed that the

types form a relationship relation. Furthermore, type *Customer_id* should be considered as key type since it has leaf vertex (*Customer_name*).

Table 14. Adjacencies between the elements of types *Order_id* and *Customer_id*.

		Order_id					
		O1	**O2**	**O3**	**O4**	**Row sum**	**Dependency**
Custom-er_id	**C1**	1	0	0	0	1	1:1
	C2	0	1	0	0	1	1:1
	C3	0	0	0	0	0	Optional
	C4	0	0	1	1	2	1:M
	Column sum	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1		

The adjacencies (table 14) between the elements of *Order_id* and *Customer_id* indicate, that between the types there is one-to-many dependency with optionality constraint. These observations strongly suggest that type *Order_id* should be considered as key attribute type. Thus, types *Order_id* and *Customer_id* construct a relation in which *Order_id* and its elements represent key attributes and their values. *Customer_id* type represents the foreign key.

Example 12. Consider a simple database for customer orders. There is a rule for the database that states that one product can be in several orders, and order may include many products. In this example the *Order* relation contains primary key and foreign key and *Order_Product* relation contain only the foreign keys. Database is made up of the relations shown in figure 51.

Customer	
Customer_id (PK)	Customer_name
C1	Jack Smith
C2	Julia Smith
C3	John Doe
C4	Jane Doe

Product		
Product_id (PK)	Product_name	Units_in_stock
P1	9" nail	23
P2	bolt	15
P3	hammer	7
P4	bucket	9

Order	
Order_id (PK)	Customer_id (FK)
O1	C1
O2	C2
O3	C4
O4	C4

Order_Product	
Order_id (FK)	Product_id (FK)
O1	P1
O1	P2
O2	P1
O2	P2
O2	P3
O3	P4
O4	P1

Figure 51. Customer order database.

Database is portrayed as ARS and AdSchema shown in figures 51 and 52. From the AdSchema (figure 52) it can be seen that type *Order_id* does not belong to the vertex dominating set of the graph. Based on the observations represented in Table 13 it is known that between elements of *Product_id* and *Order_id* there are many-to-many relationships. Observation represented in Table 14 show that between the elements of *Customer_id* and *Order_id* exist one-to-many relationships. These observations suggest that the database should contain relationship relation with elements of *Product_id* and *Order_id* as foreign keys and it should contain also a relation which has *Order_id* as primary key and *Customer_id* as foreign key.

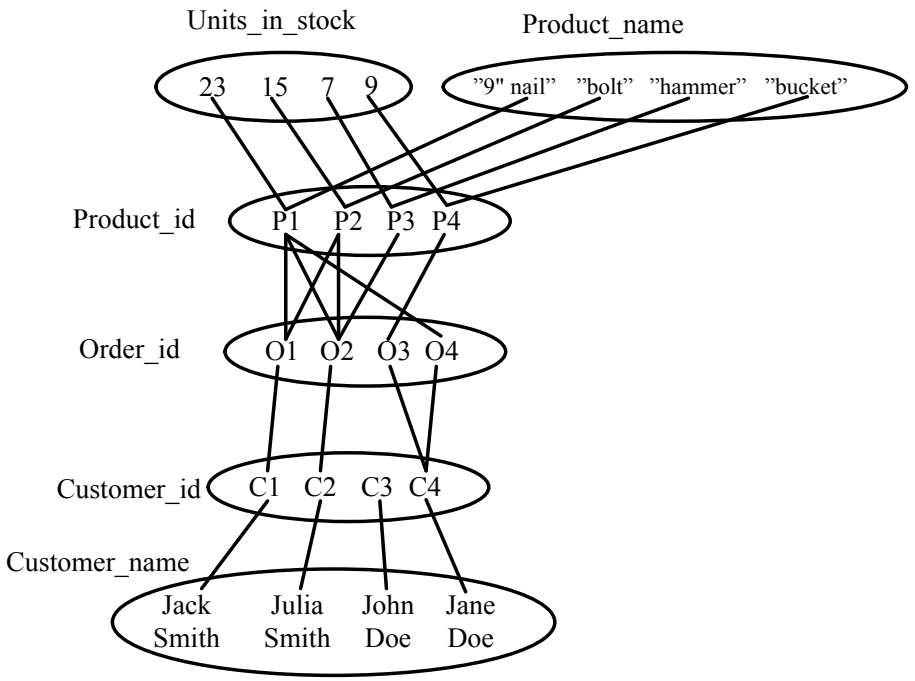


Figure 52. ARS for Customer order database.

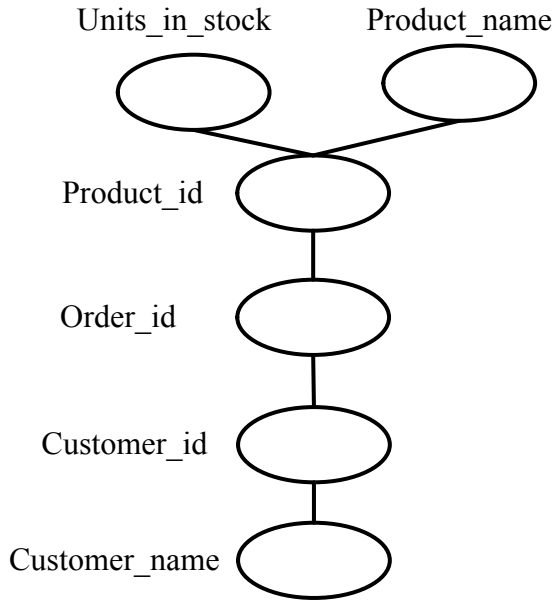


Figure 53. AdSchema for Customer order database.

7.3.2 Facts table in adjacency relation system

Data warehouse is a widely used solution for storing data that supports decision-making processes. Typically data warehouse is implemented as a relational database and its main purpose of use is query and analysis. The usual schema for data

warehouse is a star schema (figure 54). It typically consists of facts table and several dimension tables. Consider a warehouse for sales data that contains tables and data shown in figure 55. (Lane 2007:1-1,2-3; Golfarelli & Rizzi 2009: 5; Hovi, Karvonen & Koistinen 2009: 36-39)

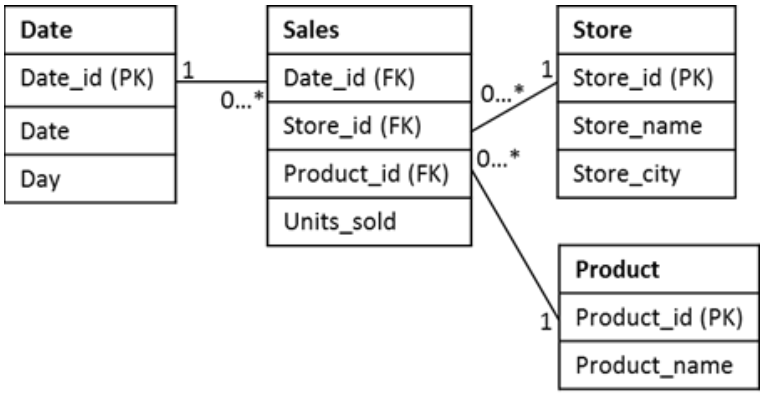


Figure 54. Star schema for sales data.

Date

Date_id	Date	Day
D1	21.1.2013	Monday
D2	22.1.2013	Tuesday
D3	23.1.2013	Wednesday
D4	24.1.2012	Thursday
D5	25.1.2013	Friday
D6	28.1.2013	Monday
D7	11.2.2013	Monday
D8	12.2.2013	Tuesday

Sales

Date_id	Store_id	Product_id	Units_sold
D1	S1	P1	100
D2	S1	P2	35
D3	S2	P1	71
D4	S3	P3	2
D5	S4	P2	50
D6	S4	P4	1
D7	S4	P5	2
D8	S5	P5	4

Product

Product_id	Product_name
P1	9" nail
P2	Bolt
P3	Hammer
P4	Bucket
P5	Saw

Store

Store_id	Store_name	Store_city
S1	AB Company	Harrisonburg
S2	City Hardware	Dauphin
S3	Sons & Sons	Loghill
S4	Cheap Products	Newington
S5	Buy, Buy!	Rockwell

Figure 55. Tables of sales data warehouse.

The data warehouse is modeled as ARS and AdSchema (figures 56 and 57). Because the modeling method removes all the types created from foreign keys, the challenge is to recognize these elements and thus construct the fact table of a star schema.

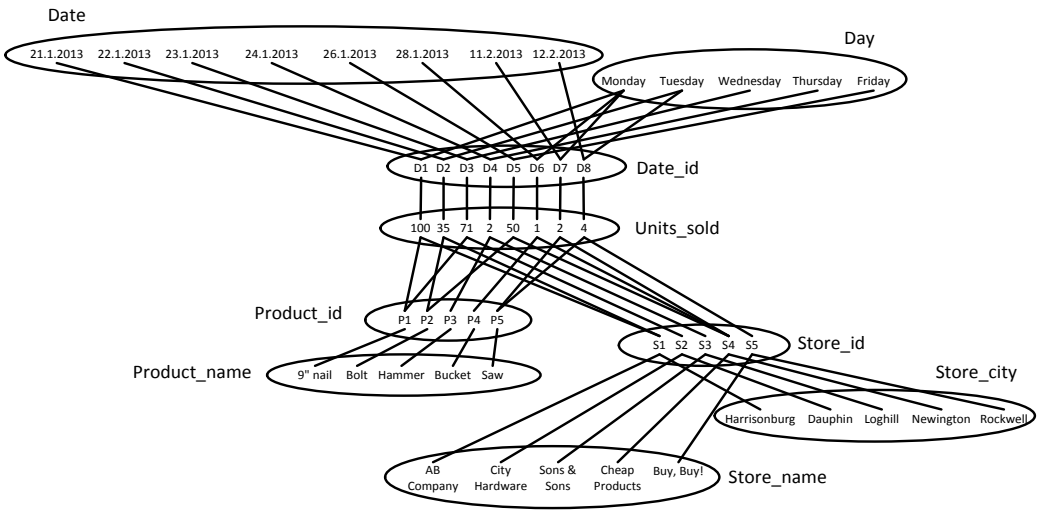


Figure 56. ARS for data warehouse.

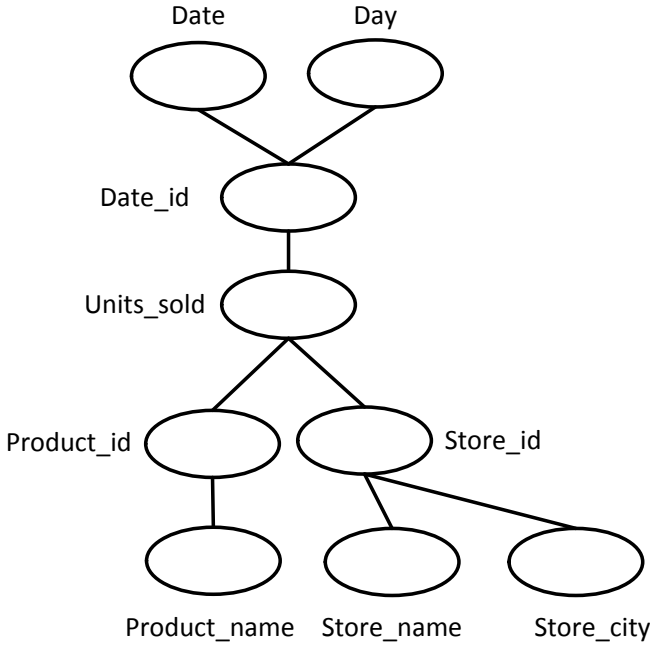


Figure 57. AdSchema for data warehouse.

In the ARS between the elements of types *Day* and *Date_id* appears to be one-to-many relationships because for example "*Monday*" is adjacent "*D1*", "*D6*" and "*D7*". However, by examining the AdSchema the type *Day* represents regular at-

tributes despite the one-to-many relationship. Next problematic type is *Units_sold*. Even though the elements of *Units_sold* have degree greater than two but because the elements are not members of vertex dominating set, and they do not have leaves. The type or its elements do not fully meet the criteria for key attribute type and its values. Let us examine (Tables 15, 16 and 17) the adjacencies between *Date_id*, *Units_sold*, *Product_id* and *Store_id*. These types satisfy the criteria for key attributes. Note that the set *Units_sold* contains similar elements (2 twice), because the conversion method does not remove similar attribute values, since they are assigned to different tuples of the modeled database.

Table 15. Adjacencies between *Date_id* and *Units_sold*.

		Units_sold									
		100	**35**	**71**	**2**	**50**	**1**	**2**	**4**	**Row sums**	**Dependency**
Date_id	**D1**	1	0	0	0	0	0	0	0	1	1:1
	D2	0	1	0	0	0	0	0	0	1	1:1
	D3	0	0	1	0	0	0	0	0	1	1:1
	D4	0	0	0	1	0	0	0	0	1	1:1
	D5	0	0	0	0	1	0	0	0	1	1:1
	D6	0	0	0	0	0	1	0	0	1	1:1
	D7	0	0	0	0	0	0	1	0	1	1:1
	D8	0	0	0	0	0	0	0	1	1	1:1
	Column sums	1	1	1	1	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1		

Table 16. Adjacencies between *Product_id* and *Units_sold*.

		Units_sold									
		100	**35**	**71**	**2**	**50**	**1**	**2**	**4**	**Row sums**	**Dependency**
Product_id	**P1**	1	0	1	0	0	0	0	0	2	1:M
	P2	0	1	0	0	1	0	0	0	2	1:M
	P3	0	0	0	1	0	0	0	0	1	1:1
	P4	0	0	0	0	0	1	0	0	1	1:1
	P5	0	0	0	0	0	0	1	1	2	1:M
	Column sums	1	1	1	1	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1		

Table 17. Adjacencies between *Store_id* and *Units_sold*.

		Units_sold									
		100	**35**	**71**	**2**	**50**	**1**	**2**	**4**	**Row sums**	**Dependency**
Store_id	**S1**	1	1	0	0	0	0	0	0	2	1:M
	S2	0	0	1	0	0	0	0	0	1	1:1
	S3	0	0	0	1	0	0	0	0	1	1:1
	S4	0	0	0	0	1	1	1	0	3	1:M
	S5	0	0	0	0	0	0	0	1	1	1:1
	Column sums	1	1	1	1	1	1	1	1		
	Dependency	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1		

Table 15 to 17 reveal one-to-many relationship between the types *Product_id* and *Units_sold* and as well between *Store_id* and *Units_sold*. Thus, the relation should have *Product_id*, *Store_id* as foreign keys accompanied with *Units_sold*. Moreover, AdSchema shows that *Units_sold* is adjacent to multiple types that represent key attributes. This implies that the database should have a star schema. So the *Date_id* type is also assigned as foreign key to the relation which is can be considered as facts table. If the facts table contains several regular attributes, each new type is adjacent to all primary key types (figure 58). Thus facts table is constructed with the same principles discussed before.

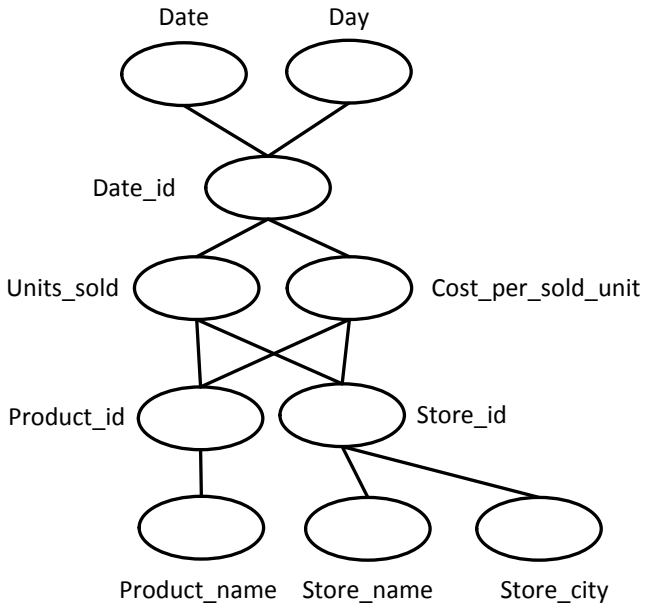
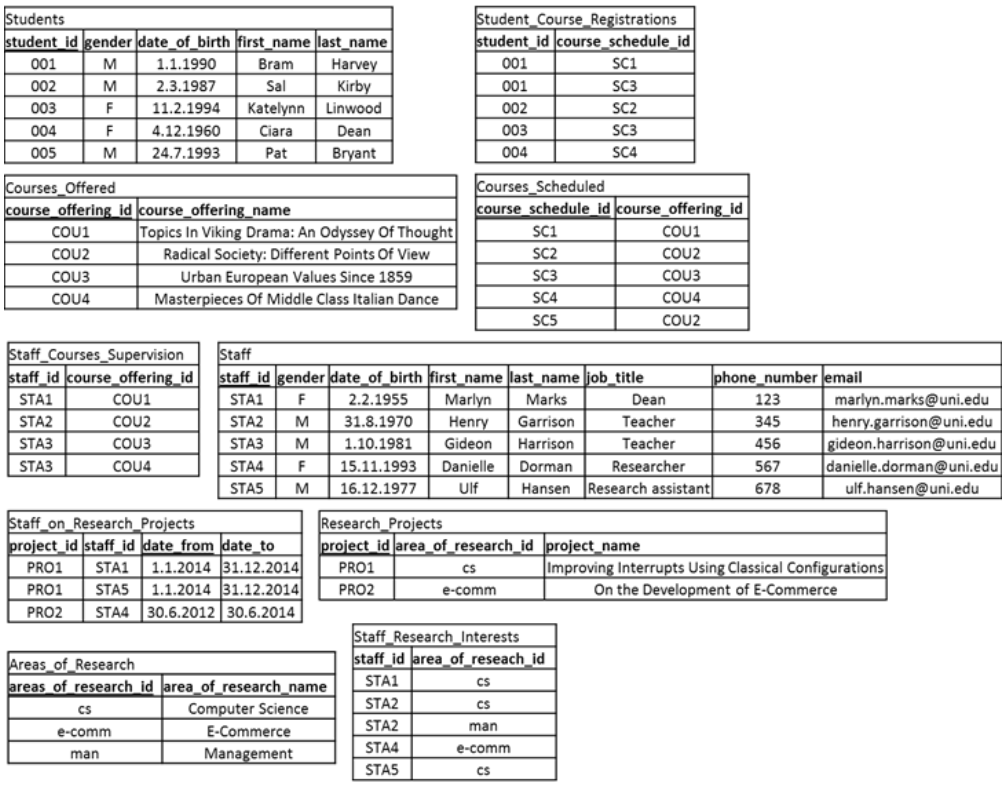


Figure 58. AdSchema with multiple attributes in facts table.

7.3.3 Reconstructing the database

This section the usage of methods discussed in this study for modeling relational database with adjacency model and adjacency relation system and as well reconstructing the database back into adjacency relation system.

Consider a database (figure 59) for the United Kingdom's Open University (Williams, 2012). Database contains information about university staff, students, courses, course schedules, course supervisors, research projects and staff's research interests.



Students

student_id	gender	date_of_birth	first_name	last_name
001	M	1.1.1990	Bram	Harvey
002	M	2.3.1987	Sal	Kirby
003	F	11.2.1994	Katelynn	Linwood
004	F	4.12.1960	Ciara	Dean
005	M	24.7.1993	Pat	Bryant

Student_Course_Registrations

student_id	course_schedule_id
001	SC1
001	SC3
002	SC2
003	SC3
004	SC4

Courses_Offered

course_offering_id	course_offering_name
COU1	Topics In Viking Drama: An Odyssey Of Thought
COU2	Radical Society: Different Points Of View
COU3	Urban European Values Since 1859
COU4	Masterpieces Of Middle Class Italian Dance

Courses_Scheduled

course_schedule_id	course_offering_id
SC1	COU1
SC2	COU2
SC3	COU3
SC4	COU4
SC5	COU2

Staff_Courses_Supervision

staff_id	course_offering_id
STA1	COU1
STA2	COU2
STA3	COU3
STA3	COU4

Staff

staff_id	gender	date_of_birth	first_name	last_name	job_title	phone_number	email
STA1	F	2.2.1955	Marlyn	Marks	Dean	123	marlyn.marks@uni.edu
STA2	M	31.8.1970	Henry	Garrison	Teacher	345	henry.garrison@uni.edu
STA3	M	1.10.1981	Gideon	Harrison	Teacher	456	gideon.harrison@uni.edu
STA4	F	15.11.1993	Danielle	Dorman	Researcher	567	danielle.dorman@uni.edu
STA5	M	16.12.1977	Ulf	Hansen	Research assistant	678	ulf.hansen@uni.edu

Staff_on_Research_Projects

project_id	staff_id	date_from	date_to
PRO1	STA1	1.1.2014	31.12.2014
PRO1	STA5	1.1.2014	31.12.2014
PRO2	STA4	30.6.2012	30.6.2014

Research_Projects

project_id	area_of_research_id	project_name
PRO1	cs	Improving Interrupts Using Classical Configurations
PRO2	e-comm	On the Development of E-Commerce

Areas_of_Research

areas_of_research_id	area_of_research_name
cs	Computer Science
e-comm	E-Commerce
man	Management

Staff_Research_Interests

staff_id	area_of_reseach_id
STA1	cs
STA2	cs
STA2	man
STA4	e-comm
STA5	cs

Figure 59. Database diagram for Open University (Williams, 2012).

Database has nine relations. Four of these relations are considered as relationship relations (*Student_Course_Registrations*, *Staff_Courses_Supervision*, *Staff_Reseach_Interests* and *Staff_on_ Research_Projects*). Töyli's (2002 and 2006) modeling method removes the foreign keys from the ARS. Only the relation *Staff_on_Research_Projects* has attributes that are not considered as foreign keys (*date_from* and *date_to*) and *date_from* was defined as part of relation's key. The type *course_schedule_id* does not fully satisfy the criteria set for types and elements representing the key attributes and their values. The ARS-based graph representation of the adjacency relation system for the database is shown in figure 60.

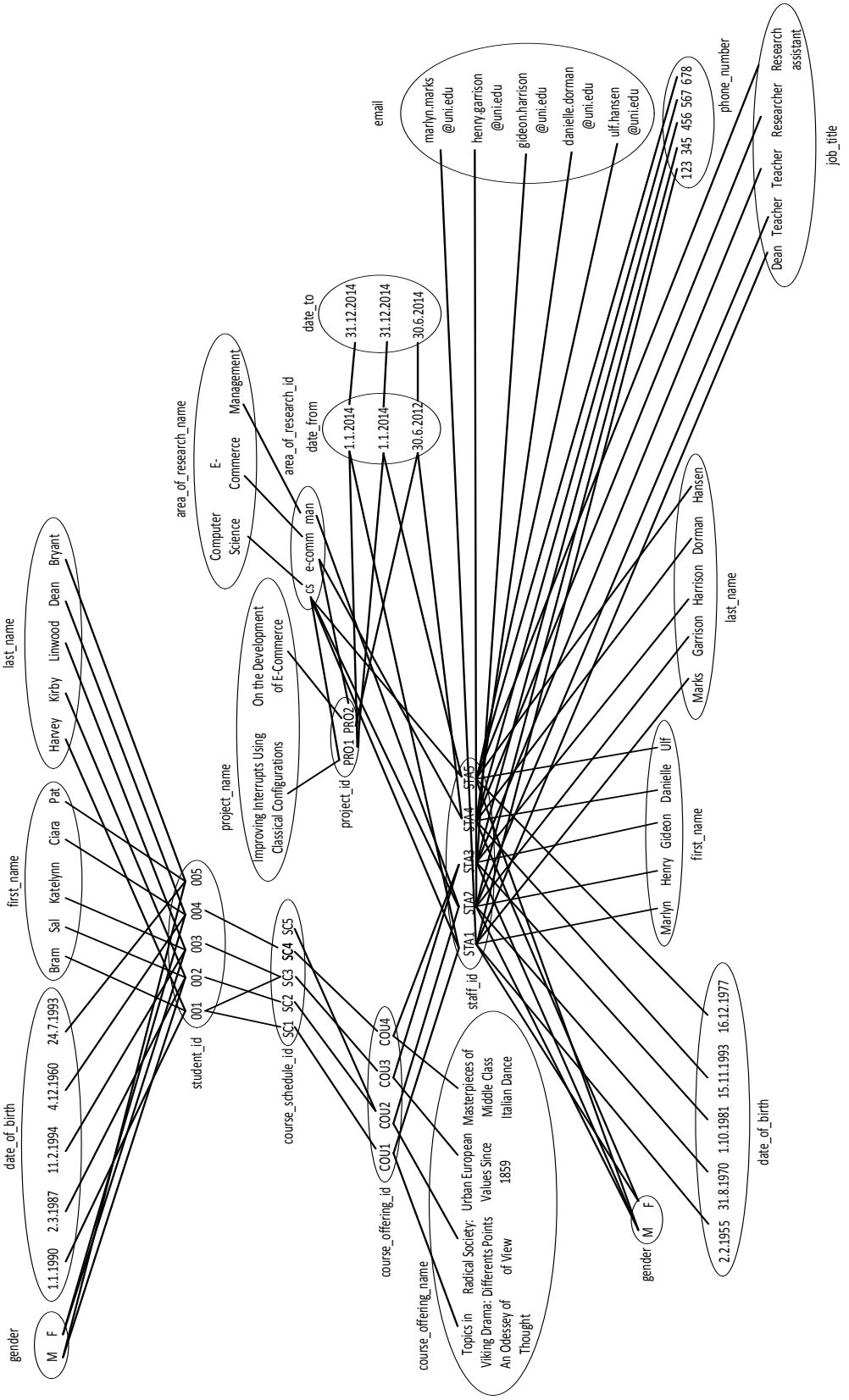


Figure 60. Adjacency relation system for the database.

The relationships between element types of ARS are represented by AdSchema shown in figure 61. AdSchema is a useful tool to complement the analysis of the ARS. In this case, for example, the attribute gender has two values M and F. In the ARS the elements M and F have multiple adjacent elements which, mean they have a degree greater than 1, and thus they are not considered to be leaves. However, the AdSchema reveals that the type gender is a leaf vertex and thus should be handled as a regular attribute.

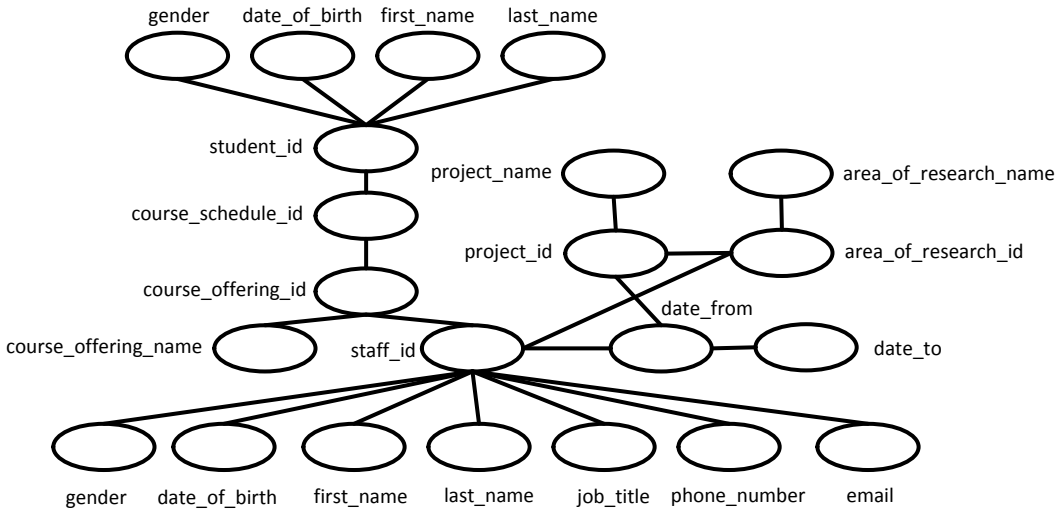


Figure 61. AdSchema of the ARS.

ARS and AdSchema are graphs representing the contents and the structure of the database. In this study was proposed that types and elements representing the key attributes and their values typically have three common properties – they have a degree at least two, they belong to vertex dominating set of the graph, and they have leaf vertices. Tables 18 and 19 portray properties for the vertices of ARS graph and AdSchema graph.

Table 18. Vertex properties in ARS graph.

Vertex name	Degree	Vertex dominating set	Parent node
M	3		
F	2		
1.1.1990	1		
2.3.1987	1		
11.2.1994	1		
4.12.1960	1		
24.7.1993	1		

Bram	1		
Sal	1		
Katelynn	1		
Ciara	1		
Pat	1		
Harvey	1		
Kirby	1		
Linwood	1		
Dean	1		
Bryant	1		
001	6	x	x
002	5	x	x
003	5	x	x
004	5	x	x
005	4	x	x
SC1	2		
SC2	2		
SC3	3		
SC4	2		
SC5	1		
COU1	3	x	x
COU2	4	x	x
COU3	3	x	x
COU4	3	x	x
Topics In Viking Drama: An Odyssey Of Thought	1		
Radical Society: Different Points Of View	1		
Urban European Values Since 1859	1		
Masterpieces Of Middle Class Italian Dance	1		
STA1	11	x	x
STA2	10	x	x
STA3	9	x	x
STA4	10	x	x
STA5	10	x	x
M	3		
F	2		
2.2.1955	1		
31.8.1970	1		
1.10.1981	1		
15.11.1993	1		
16.12.1977	1		
Marlyn	1		

Henry	1		
Gideon	1		
Danielle	1		
Ulf	1		
Marks	1		
Garrison	1		
Harrison	1		
Dorman	1		
Hansen	1		
Dean	1		
Teacher	1		
Teacher	1		
Researcher	1		
Research assistant	1		
123	1		
345	1		
456	1		
567	1		
678	1		
marlyn.marks@uni.edu	1		
henry.garrison@uni.edu	1		
gideon.harrison@uni.edu	1		
danielle.dorman@uni.edu	1		
ulf.hansen@uni.edu	1		
PRO1	6	x	x
PRO2	4	x	x
Improving Interrupts Using Classical Configurations	1		
On the Development of E-Commerce	1		
cs	5	x	x
e-comm	3	x	x
man	2	x	x
Computer Science	1		
E-Commerce	1		
Management	1		
1.1.2014	3	x	x
1.1.2014	3	x	x
30.6.2012	3	x	x
31.12.2014	1		
31.12.2014	1		
30.6.2014	1		

Table 19. Vertex properties in the AdSchema graph.

Vertex name	Degree	Vertex dominating set	Parent node
gender	1		
date_of_birth	1		
first_name	1		
last_name	1		
student_id	5	x	x
course_schedule_id	2		
course_offering_id	3	x	x
course_offering_name	1		
staff_id	11	x	x
gender	1		
date_of_birth	1		
first_name	1		
last_name	1		
job_title	1		
phone_number	1		
email	1		
date_from	3	x	x
date_to	1		
project_id	4	x	x
project_name	1		
areas_of_research_id	3	x	x
area_of_research_name	1		

Based on the observations listed in tables 18 and 19, types *student_id*, *course_offering_id*, *staff_id*, *date_from*, *project_id* and *areas_of_research_id* and their elements represent key attributes and their values. Each vertex, in ARS as well as in AdSchema, has degree equal or greater than 2. Also each type has leaf vertices, and each type is a member of vertex dominating set of the graph.

Relation in AdSchema is a set of distinct vertices of the walk from vertex representing key attribute type to its leaves. Based on the AdSchema (figure 61) it can be concluded, that database contains at least the relations listed below.

$R_1 = \{student\_id, gender, date\_of\_birth\_first\_name, last\_name\}$

$R_2 = \{course\_offering\_id, course\_offering\_name\}$

$R_3 = \{staff\_id, gender, date\_of\_birth, first\_name, last\_name, job\_title, phone\_number, email\}$

$R_4 = \{date\_from, date\_to\}$

$R_5 = \{project\_id, project\_name\}$

$R_6 = \{area\_of\_research\_id, area\_of\_research\_name\}$

Type *course_schedule_id* is not considered as vertex representing key attributes. This situation can be interpreted in two ways, it implies the existence of relationship relation or the type belongs to a relation, which originally contained only primary key and foreign key. At this phase database has 22 tuples which are:

$t_1 = \{001, M, 1.1.1990, Bram, Harvey\}$

$t_2 = \{002, M, 2.3.1987, Sal, Kirby\}$

$t_3 = \{003, F, 11.2.1994, Katelynn, Linwood\}$

$t_4 = \{004, F, 4.12.1960, Ciara, Dean\}$

$t_5 = \{005, M, 24.7.1993, Pat, Bryant\}$

$t_6 = \{COU1, Topics\ In\ Viking\ Drama: An\ Odyssey\ Of\ Thought\}$

$t_7 = \{COU2, Radical\ Society: Different\ Points\ Of\ View\}$

$t_8 = \{COU3, Urban\ European\ Values\ Since\ 1859\}$

$t_9 = \{COU4, Masterpieces\ Of\ Middle\ Class\ Italian\ Dance\}$

$t_{10} = \{STA1, F, 2.2.1955, Marlyn, Marks, Dean, 123, marlyn.marks@uni.edu\}$

$t_{11} = \{STA2, M, 31.8.1970, Henry, Garrison, Teacher, 345, henry.garrison@uni.edu\}$

$t_{12} = \{STA3, M, 1.10.1981, Gideon, Harrison, Teacher, 456, gideon.harrison@uni.edu\}$

$t_{13} = \{STA4, F, 15.11.1993, Danielle, Dorman, Researcher, 567, danielle.dorman@uni.edu\}$

$t_{14} = \{STA5, M, 16.12.1977, Ulf, Hansen, Research, assistant, 678, ulf.hansen@uni.edu\}$

$t_{15} = \{1.1.2014, 31.12.2014\}$

$t_{16} = \{1.1.2014, 31.12.2014\}$

$t_{17} = \{30.6.2012, 30.6.2014\}$

$t_{18} = \{PRO1, Improving\ Interrupts\ Using\ Classical\ Configurations\}$

$t_{19} = \{PRO2, On\ the\ Development\ of\ E-Commerce\}$

$t_{20} = \{cs, Computer\ Science\}$

$t_{21} = \{e-comm, E-Commerce\}$

$t_{22} = \{man, Management\}$

The dependencies between the relations of the database are determined by analyzing the adjacencies between the elements of the key types. Adjacency examinations reveal the primary dependencies such as one-to-one, one-to-many and many-to-many. The examinations also shows, if the relationship should be considered optional. At first adjacencies between elements of $course\_offering\_id$ and $staff\_id$ are analyzed. The adjacencies are depicted in table 20.

Table 20. Adjacencies between elements of the types $course\_offering\_id$ and $staff\_id$.

		staff_id					
		STA1	**STA2**	**STA3**	**STA4**	**STA5**	**Rows sums**
course_offering_id	**COU1**	1	0	0	0	0	1
	COU2	0	1	0	0	0	1
	COU3	0	0	1	0	0	1
	COU4	0	0	1	0	0	1
	Column sums	1	1	2	0	0	

The type of dependency can be determined by row sums and columns of the adjacency matrix. Row sums show, that each element in $course\_offering\_id$ is adjacent to one element in $staff\_id$. Moreover, the column sums show, that elements of $staff\_id$ can be adjacent to multiple elements of $course\_offering\_id$ and also the column sums show that some of the elements of $staff\_id$ do not have adjacent elements in $course\_offering\_id$.

Because the elements of $course\_offering\_id$ type represent the values of key attributes of relation R_2 and respectively the elements of $staff\_id$ represent the values of key attributes of relation R_3. The dependency between relations is defined as is one-to-many (Table 21).

Table 21. Dependency between relations R_3 and R_2.

R_3	R_2
1	0…*

The adjacencies between the elements of type $staff\_id$ and the elements of type $date\_from$ (Table 22) are analyzed in order to determine the nature of relationship between the relations R_3 and R_4.

Table 22. Adjacencies between elements of types $staff\_id$ and $date\_from$.

		date_from			
		1.1.2014	**1.1.2014**	**30.6.2012**	**Row sums**
staff_id	**STA1**	1	0	0	1
	STA2	0	0	0	0
	STA3	0	0	0	0
	STA4	0	0	1	1
	STA5	0	1	0	1
	Column sums	1	1	1	

The column sums of table 22 show that each element of $date\_from$ is adjacent to one element of type $staff\_id$. The rows sums indicate that elements *STA1*, *STA4* and *STA3* have adjacent elements in $date\_from$, on the other hand the elements *STA2* and *STA3* do not have adjacent elements in $date\_from$. Based on these factors the dependency (Table 23) between R_3 and R_4 defined as one-to-one with optionality constraint.

Table 23. Dependency between relations R_3 and R_4.

R_3	R_4
1	0…1

The adjacencies between the elements of type $project\_id$ and the elements of type $date\_from$ are analyzed (table 24) in order to determine the type of dependency between the relations R_5 and R_4.

Table 24. Adjacencies between elements of *types project_id* and *date_from*.

		date_from			
		1.1.2014	**1.1.2014**	**30.6.2012**	**Row sums**
project_id	**PRO1**	1	1	0	2
	PRO2	0	0	1	1
	Column sums	1	1	1	

The column sums of table 24 show that each element of *date_from* is adjacent to one element of type *project_id*. The row sums indicate that elements of *project_id* are adjacent to one or more elements of *date_from*. Thus, dependency between R_5 and R_4 is one-to-many (table 25).

Table 25. Dependency between relations R_5 and R_4.

R_5	R_4
1	1…*

Table 26 shows the adjacencies between types *project_id* and *area_of_research_id*. The column and row sums of the table indicate the type of dependency between relations R_5 and R_6.

Table 26. Adjacencies between elements of *types project_id* and *area_of_research_id*.

		area_of_research_id			
		cs	**e-comm**	**man**	**Row sums**
project_id	**PRO1**	1	0	0	1
	PRO2	0	1	0	1
	Column sums	1	1	0	

Table 26 shows that elements of *project_id* type have one adjacent element in *area_of_research_id*. Elements of *area_of_research_id* type have one adjacent element in *project_id* type, however, element *man* of *area_of_research_id* does not have any adjacent elements belonging to *project_id* type. So, the dependency between relations R_5 and R_6 is considered one-to-one with optionality constraint (Table 27).

Table 27. Dependency between relations R_6 and R_5.

R_6	R_5
1	0…1

Table 28 examines the adjacencies between the elements of types $staff\_id$ and $area\_of\_research\_id$. Elements of type $staff\_id$ represent the values of key attributes of relation R_3 and the elements of type $area\_of\_research\_id$ represent the values of key attributes of relation R_6.

Table 28. Adjacencies between the elements of types $staff\_id$ and $area\_of\_research\_id$.

		area_of_research_id			
		cs	**e-comm**	**man**	**Row sums**
staff_id	**STA1**	1	0	0	1
	STA2	1	0	1	2
	STA3	0	0	0	0
	STA4	0	1	0	1
	STA5	1	0	0	1
	Column sums	3	1	1	

Table 28 shows that elements of $staff\_id$ type have multiple adjacent elements in $area\_of\_research\_id$, yet, *STA3* element does not have adjacent element. Elements of type $area\_of\_research\_id$ have multiple adjacent elements in $project\_id$. So, the dependency between relations R_3 and R_6 can be seen as many-to-many (Tables 29 and 30).

Table 29. Dependency between relations R_3 and R_6.

R_3	R_6
1	0…*

Table 30 Dependency between relations R_6 and R_3.

R_6	R_3
1	1…*

Relational model does not favor many-to-many dependency, which means that a relationship relation has to be created. Relation R_{rel1} contains both types as foreign keys to corresponding relations. The dependency definitions for relations R_3, R_6 and R_{rel1} are shown in figure 62.

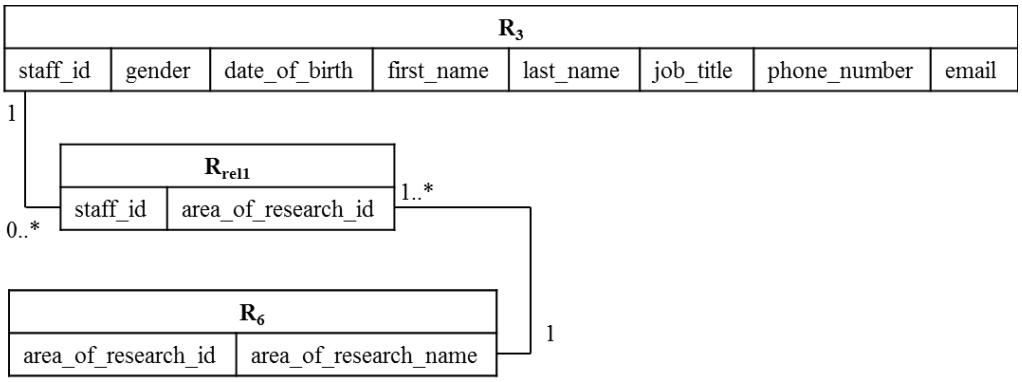


Figure 62. Relationship relation.

The type *course_schedule_id* in this case is not considered as a key type. However, based on the AdSchema and ARS graph the type should not be considered as a type that represents regular attribute. It is necessary to examine the adjacencies between the elements of types *student_id* and *course_schedule_id* as well the adjacencies between types *course_offering_id* and *course_schedule_id* (table 31).

Table 31. Adjacencies between the elements of types *student_id* and *course_schedule_id*.

		course_schedule_id					
		SC1	**SC2**	**SC3**	**SC4**	**SC5**	**Row sums**
student_id	**001**	1	0	1	0	0	2
	002	0	1	0	0	0	1
	003	0	0	1	0	0	1
	004	0	0	0	1	0	1
	005	0	0	0	0	0	0
	Column sums	1	1	2	1	0	

Table 31 shows that elements of *student_id* type have multiple adjacent elements in *course_schedule_id*, yet, element *005* does not have adjacent elements. Elements of type *course_schedule_id* have multiple adjacent elements in *student_id*, except the SC5 which does not have adjacent elements. So, the dependency between relations R_1 and *course_schedule_id* is seen as many-to-many.

Table 32. Dependency between relations R_1 and *course_schedule_id*.

R_1	*course_schedule_id*
1	0…*

Table 33. Dependency between relations *course_schedule_id* and R_1.

course_schedule_id	R_1
1	0…*

Because the relationship between R_1 and course *schedule_id* is many-to-many, a relationship relation R_{rel2} has to be constructed. It has both types as foreign keys to corresponding relations. At this point a new relation R7 is created it contains type *course_schedule_id* as primary key. The dependency definitions for relations R_1, *course_schedule_id* (denoted as relation R_7) and R_{rel2} are shown in figure 63.

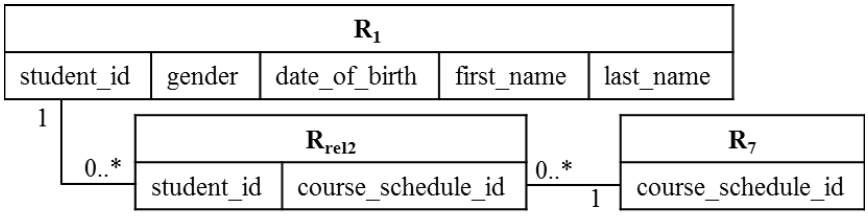


Figure 63. The relationship definitions for relations R_1, *course_schedule_id* and R_{rel2}.

According to AdSchema (see figure 61) type *course_schedule_id* is adjacent to type *course_offering_id*. The adjacency of elements of these types are shown in table 34.

Table 34. Adjacencies between *course_schedule_id* and type *course_offering_id*.

		course_schedule_id					
		SC1	**SC2**	**SC3**	**SC4**	**SC5**	**Row sums**
course_offering_id	**COU1**	1	0	0	0	0	1
	COU2	0	1	0	0	1	2
	COU3	0	0	1	0	0	1
	COU4	0	0	0	1	0	1
	Column sums	1	1	1	1	1	

Table 34 shows that elements of *course_schedule_id* type have one adjacent element in *course_offering_id*. Elements of type *course_offering_id* may have multiple adjacent elements in *course_schedule_id*. So, the dependency between relations R_2 and newly created R_7 (*course_schedule_id*) is seen as one-to-many (table 35).

Table 35. Dependency between relations R_2 and R_7.

R_2	R_7
1	1…*

After the dependencies and possible constraints are defined for the relations, the database schema is reconstructed (figure 64). Respectively the reconstructed database is shown in figure 65. The modeling method removes all the types created from foreign keys from the ARS. Thus, some of the keys must be duplicated and assigned as foreign keys to the corresponding relations.

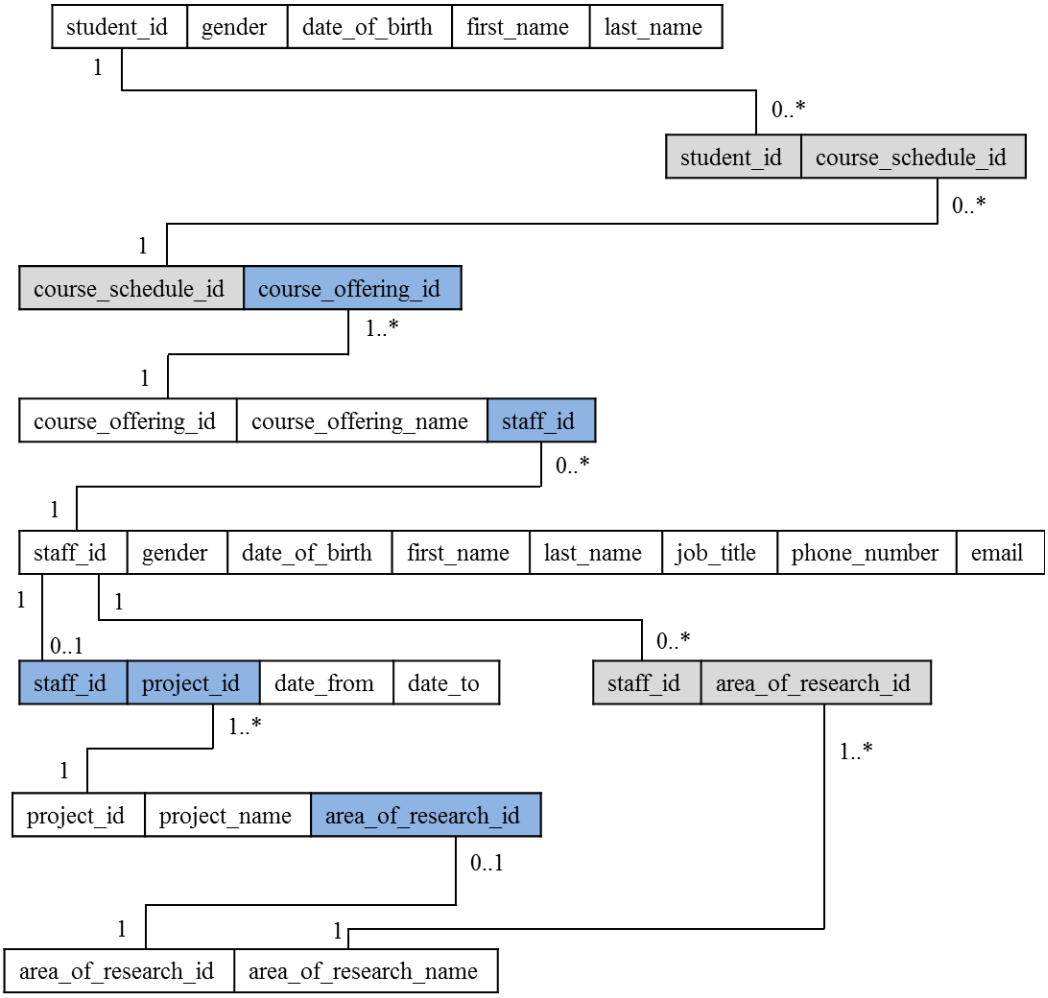


Figure 64. The schema of the reconstructed database.

Students

student_id	gender	date_of_birth	first_name	last_name
001	M	1.1.1990	Bram	Harvey
002	M	2.3.1987	Sal	Kirby
003	F	11.2.1994	Katelynn	Linwood
004	F	4.12.1960	Ciara	Dean
005	M	24.7.1993	Pat	Bryant

Student_Course_Registrations

student_id	course_schedule_id
001	SC1
001	SC3
002	SC2
003	SC3
004	SC4

Courses_Scheduled

course_schedule_id	course_offering_id
SC1	COU1
SC2	COU2
SC3	COU3
SC4	COU4
SC5	COU2

Courses_Offered

course_offering_id	staff_id	course_offering_name
COU1	STA1	Topics In Viking Drama: An Odyssey Of Thought
COU2	STA2	Radical Society: Different Points Of View
COU3	STA3	Urban European Values Since 1859
COU4	STA3	Masterpieces Of Middle Class Italian Dance

Staff

staff_id	gender	date_of_birth	first_name	last_name	job_title	phone_number	email
STA1	F	2.2.1955	Marlyn	Marks	Dean	123	marlyn.marks@uni.edu
STA2	M	31.8.1970	Henry	Garrison	Teacher	345	henry.garrison@uni.edu
STA3	M	1.10.1981	Gideon	Harrison	Teacher	456	gideon.harrison@uni.edu
STA4	F	15.11.1993	Danielle	Dorman	Researcher	567	danielle.dorman@uni.edu
STA5	M	16.12.1977	Ulf	Hansen	Research assistant	678	ulf.hansen@uni.edu

Staff_on_Research_Projects

project_id	staff_id	date_from	date_to
PRO1	STA1	1.1.2014	31.12.2014
PRO1	STA5	1.1.2014	31.12.2014
PRO2	STA4	30.6.2012	30.6.2014

Research_Projects

project_id	area_of_research_id	project_name
PRO1	cs	Improving Interrupts Using Classical Configurations
PRO2	e-comm	On the Development of E-Commerce

Areas_of_Research

areas_of_research_id	area_of_research_name
cs	Computer Science
e-comm	E-Commerce
man	Management

Staff_Research_Interests

staff_id	area_of_reseach_id
STA1	cs
STA2	cs
STA2	man
STA4	e-comm
STA5	cs

Figure 65. Reconstructed database.

The foreign key duplications are based on the adjacency examinations discussed in this section. Rule of thumb for key duplication is that the key of the one-end of the relationship is copied to the relation of the many-end of relationship (figure 64). In the case one-to-one relationship of either of the keys can be copied and assigned as a foreign key. If there is optionality constraint in the relationship the key of the non-optional end assigned as a foreign key to optional-end relation (figure 66).

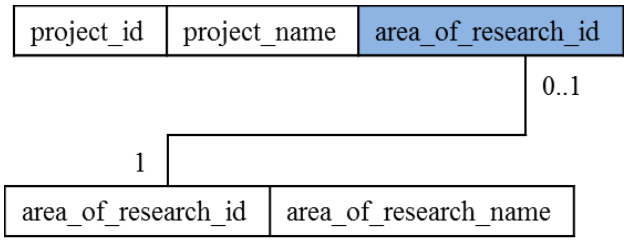


Figure 66. Foreign key duplication.

The ARS created from the original database is a snapshot of the database at the moment. Because the properties of the database are determined based on the stored data, i.e. the adjacencies between types and their elements, the reconstructed database might not fully conform the original database schema. For example the original database schema has relationship relation for staff course supervision ($Staff\_Courses\_Supervisions$), but the adjacency analysis showed only one-to-many relationship between staff and courses, which means that based on the data the relationship relation will not be created. The reconstructed database does not violate relational model, but it contradicts the original schema.

The database reconstruction at this point relies heavily on human insight. So, the process is not optimal. However, some general complexity considerations for the calculating the metrics of the graph can be given.

The complexity of an algorithm can be measured by running time. Running time tells how much effort and computing problem solving requires. Running time is a function of a measure of the amount of data that is needed to convey the problem to the computer. For example, running time can be given as a function of the size of the input matrix or the total number of vertices in a graph. For some problems there does not exist algorithms that could solve the problems in polynomial time. These problems are called NP-complete (or NP-hard). (Wilf 1994: 2, 104-105.)

The graph structure is stored as adjacency matrix, each row containing n elements, which causes degree calculations to take time $O(n)$ (Newman 2010: 309). Determining the vertex dominating set of the graph is NP-hard problem. The running times of the proposed solutions have variations. For example an exhaustive solution could take $O(2^{n^2})$ (Fernau 2010: 310). On the other hand Grandoni (2006: 209-210) has suggested a solution that takes $O(1{,}3424^{2n})$, where n is number of vertices in G. Determining the parent nodes of the graph utilizes the principles of finding nearest neighbors. Vaidya (1989: 102) has suggested a solution for solving the nearest neighbor problems that takes $O(n \log n)$ time.

The framework represented in this work enables the conversions from database to ARS and from ARS to database. In more general level, the framework could be considered as feasibility study that determines if a given graph structure is suitable foundation for the creation of relational database.

8 UTILIZATION OF MULTI-OBJECTIVE OPTIMIZATION IN GRAPH ANALYSIS

Previous sections of this report discussed the similarities between adjacency model and relational model. These similarities are used in modeling relational data with adjacency model (see for example Töyli 2002). Similarities made possible to reconstruct a database from an adjacency relation system, which is a graph visualization of the adjacency model.

The introduction of the reconstruction method revealed the fact that the database elements, represented as a graph, have certain features, which can be identified by utilizing graph theory concepts such vertex degree, vertex dominating set of the graph and leaf vertex. It was also noticed, that in a graph representation of database tuples and relations are built and de around the vertices representing the values of key attributes. Also, it was noticed that dependencies are expressed by adjacencies between these vertices. Moreover, when dealing with AdSchema representation of database schema relations and dependencies are built around vertices representing key attribute types.

The identification of the vertices can be done in moderate time by using tables if the ARS is relatively small. As the number of vertices and edges increases, it is useful to utilize the multi-objective optimization in the identification process.

8.1 Multi-objective optimization

Optimization aims to maximize desirable features of the system. Optimization goals represent the extreme values, which are achieved with optimization functions. Typically the goal is to minimize the value of the objective function. (Price et al. 2005: 1)

Optimization tasks usually involve a number of variables that affect the performance of the objective function. Parameters can be discrete or continuous, they might have different types, and parameters can be finite or infinite. Domain values may have a continuous range, or they can be a set of isolated and irregularly spaced values. Parameter quantization transforms the domain values into usable scale. (Price et al. 2005: 1,189-192).

Multi-objective optimization strives to minimize K individual objective functions. The purpose is to find $x = (x_0, x_1, \dots, x_{D-1})^T, x \in \Re^D$ such that it minimizes the function $f_k(x), k = 1, \dots, K, K \geq 2$. $\Re^D$ represents the D-dimensional space of real numbers. (Price et al. 2005: 244)

If the graph is large the analysis tables contain large number of elements (rows) and thus the row-wise examination of properties is time consuming. The set of potential vertices representing the values of key attributes can be produced with multi-objective optimization. The first step is to define evaluation scale for each property. This is called parameter quantization. After the parameter quantization, the objective functions for each property are defined. The aim is then to minimize the value of each function so that the set of potential key vertices is obtained. The result set is called pareto-optimal, it is a solution that satisfies all conditions. A vector of objective function values dominates other vectors if its values are not higher, and at least one is lower (Price et al. 2005: 246; Erfani, Tohid et al. 2011: 467-468).

Pareto-front is the set of optimal solutions. Lampinen (2000: 18) states the following about the Pareto-front.

1. It contains the Pareto-optimal solutions, and it divides the objective function space into two parts, which are non-optimal solutions and infeasible solutions.
2. Pareto-front is not necessarily continuous.

8.2 Recognizing keys with multi-objective optimization

This section discusses the utilization of multi-objective optimization in the recognition of vertices representing the key attributes and their values from an ARS-based graph. The process has the following steps.

1. Determine optimization parameters.
2. Define value mappings for the parameters.
3. Define goal functions and constraining functions.
4. Run the functions.
5. Analyze and compile the results in order to find the optimal solution set.
6. Apply constraining functions if necessary and repeat step 5.

The goal functions aim to minimize the values of three parameters p_1, p_2 and p_3. The value for parameter p_1 is defined by the degree of the graph vertex. The parameter p_2 gets its value by determining if a given vertex belongs to the vertex dominating set of the graph. The value for parameter p_3 is defined by the factor whether a vertex is parent or not i.e. has leaf vertices. The value mappings for the parameters are shown in tables 36 to 38.

Table 36. Value mappings for parameter p_1.

Vertex degree	>2	2	1	0
Value for p_1	0	1	2	3

Table 37. Value mappings for parameter p_2.

Member of vertex dominating set	Yes	No
Value for p_2	0	1

Table 38. Value mappings for parameter p_3.

Vertex is parent	Yes	No
Value for p_3	0	1

After the value scales for the parameters are set, the goal functions f_1, f_2 and f_3 are defined.

- First function (f_1) determines the degree of each vertex in graph G and returns values for the vertices of the G based on the value mappings for the parameter p_1.
- Function (f_2) determines the members vertex dominating set of the graph and returns values for each vertex based on the value mappings for the parameter p_2.
- Third function (f_3) finds the parent vertices of the graph G and produces values for the vertices in G based on the value mappings for the parameter p_3.

After the parameter and goal function definitions are done the functions are applied. The functions produce $n \times 3$ matrix, where n is total number of vertices of the graph. A row of the result matrix is called value vector. A value vector represents the values that the goal functions returned. In this case the desired value vector should read $(0, 0, 0)$. Vertices having zero vectors make up the pareto-optimal solution. The set represents key attributes when the analyzed graph is AdSchema based. Respectively, when the analyzed graph is ARS based the set represents the values of key attributes.

As shown in tables 36 to 38 parameters p_2 and p_3 have only two possible values (0 or 1) as the parameter p_1 has larger value scale. The larger scale originates from the situation where original database contains relationship relation which has for example two attributes, one regular and one foreign key. Since, the modeling rules state that foreign keys are not included in the ARS based graph representation of the database; the graph contains vertices that have degree of two. Yet these vertices do not have leaves and typically they do not belong to the vertex

dominating set. The larger scale eliminates these vertices from the solution. However, the larger scale also eliminates the vertices representing key attributes (and their values) of such relations which contain primary key accompanied with one regular attribute. To include these vertices in the solution a constraining function must be defined.

The following examples show how the multi-objective optimization is utilized for recognizing the types (i.e. vertices) representing key attributes in AdSchema.

Example 13. Let $G = (V, E)$ be an AdSchema (figure 67) representation of relational database schema where $V = \{x_1, x_2, x_3, x_4, y_1, z_1, z_2, v_1, v_2, v_3\}$ and $E = \{x_1x_2, x_1x_3, x_1x_4, x_1y_1, z_1y_1, z_1z_2, z_1v_1, v_1v_2, v_1v_3, v_1v_3\}$. Since the AdSchema depicts a relational database schema let us assume that vertices x_1, z_1 and v_1 represents the key attributes types.

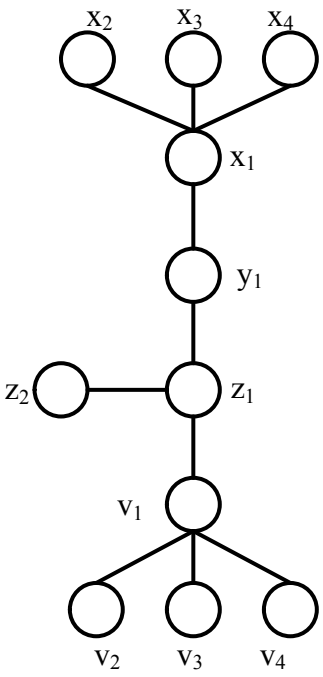


Figure 67. AdSchema representation of relational database.

Matrix M shows the values determined by objective functions for each vertex in AdSchema. Vertices that have smallest value vectors are potential set of vertices (types) representing key attributes.

$$M = \begin{array}{c|ccc} & \boldsymbol{f_1} & \boldsymbol{f_2} & \boldsymbol{f_3} \\ x_1 & 0 & 0 & 0 \\ x_2 & 2 & 1 & 1 \\ x_3 & 2 & 1 & 1 \\ x_4 & 2 & 1 & 1 \\ y_1 & 1 & 1 & 1 \\ z_1 & 0 & 0 & 0 \\ z_2 & 2 & 1 & 1 \\ v_1 & 0 & 0 & 0 \\ v_2 & 2 & 1 & 1 \\ v_3 & 2 & 1 & 1 \\ v_4 & 2 & 1 & 1 \end{array}$$

Matrix M shows that vertices x_1, z_1 and v_1 have smallest value vectors. These vertices are thus potential key attributes. The vertex set is a starting point for the reconstruction of relations, tuples and dependencies. However the value vector for y_1 is the smallest one of the remaining vectors (excluding the zero vectors), which means that in the database reconstruction phase adjacencies of such types and their elements should be examined more thoroughly.

Sometimes the graph representation of relational database may contain vertices whose degree is two, but they have leaf vertex. Therefore, it is necessary to define some constraint so that the optimal solution set includes such vertices. In the next example, a constraint function is defined and applied in determining the set of potential key types. (Price et al. 2005: 201-202)

Example 14. Let $G = (V, E)$ be an AdSchema (figure 68) representation of relational database where $V = \{x_1, x_2, y_1, z_1, z_2, v_1, v_2, v_3\}$ and $E = \{x_1x_2, x_1y_1, z_1y_1, z_1z_2, z_1v_1, v_1v_2, v_1v_3, v_1v_3\}$. Let us assume that vertices x_1, z_1 and v_1 represent key attributes.

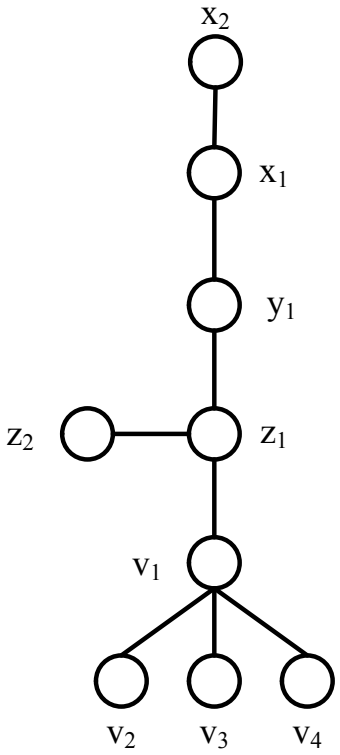


Figure 68. AdSchema representation of relational database.

The values for each vertex of objective functions are shown in the matrix M_1. From the matrix M_1it is seen that vertex x_1 does not belong to the pareto-optimal solution although it should be in it. To solve this situation a constraint is defined (Price et al. 2005: 201). Constraining function f_c minimizes the value of function f_1 if the functions f_2 and f_3 provided minimized values.

$$M_1 = \begin{array}{c} \\ x_1 \\ x_2 \\ y_1 \\ z_1 \\ z_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{ccc} f_1 & f_2 & f_3 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 0 & 0 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{array}$$

$$M_c = \begin{array}{c} \\ x_1 \\ x_2 \\ y_1 \\ z_1 \\ z_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{ccc} f_1 & f_2 & f_3 \\ 0 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 0 & 0 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{array}$$

The matrix M_c depicts the value vectors after the constraint is applied.

The factor that indicates that a constraining function should be considered is that element x_1 got minimized values from the functions f_2 and f_3 and f_1 did not produce the optimal value.

The three goal functions discussed in this section produce a set of vertices that represent key attributes in AdSchema. The AdSchema provides a graph visualization of the attributes that create relations and the database. An ARS can be seen as a graph that visualizes the values of attributes which form the tuples and eventually relations. So, the optimization functions can be applied to the analysis of ARS based graphs as well, in that case the solution set contains the elements representing the values of key attributes. The vertices in the solution set act as root vertex for the walks from root to the leaf vertices.

8.3 Utilization of goal functions in key identification

Consider the database introduced in section "*7.3.3 Reconstructing the database*". The database and the schema of the database were reconstructed by analyzing the properties of the vertices both in ARS based graph and AdSchema based graph.

The analysis is straightforward it first recognizes the keys from the graphs and then constructs tuples and relations around the keys. Finally, the dependencies are determined based on the adjacencies between key vertices.

The recognition of keys plays a vital role in database reconstruction. In this section, an example of usage goal functions is given. The goal functions aim to find vertices that satisfy the criteria set for key vertices.

1. Vertex degree must be greater or equal than 2.
2. Vertices must belong to vertex dominating set of the graph.

3. Vertices must have leaves.

The tables 39 and 40 lists vertices that minimize the functions i.e. produce zero vectors. Table 39 shows the value vectors for vertices in the ARS based graph. Respectively table 40 lists the value vectors for vertices of the AdSchema based graph.

Table 39. Value vectors for vertices representing the values of key attributes in ARS based graph.

	Goal function values		
Vertex name	f_1	f_2	f_3
001	0	0	0
002	0	0	0
003	0	0	0
004	0	0	0
005	0	0	0
COU1	0	0	0
COU2	0	0	0
COU3	0	0	0
COU4	0	0	0
STA1	0	0	0
STA2	0	0	0
STA3	0	0	0
STA4	0	0	0
STA5	0	0	0
PRO1	0	0	0
PRO2	0	0	0
cs	0	0	0
e-comm	0	0	0
man	*1*	*0*	*0*
1.1.2014	0	0	0
1.1.2014	0	0	0
30.6.2012	0	0	0

Table 40. Value vectors for vertices representing the key attributes in AdSchema based graph.

	Goal function values		
Vertex name	f_1	f_2	f_3
student_id	0	0	0
course_offering_id	0	0	0
staff_id	0	0	0
date_from	0	0	0
project_id	0	0	0
areas_of_research_id	0	0	0

Comparison of ARS and AdSchema graphs (see figures 60 and 61) with the potential key vertices shows that vertex named “man” should be included in the results. The vertex “man” belongs to type $areas\_of\_research\_id$, which according to AdSchema (see figure 61 and table 40) is considered as type representing the key attributes.

Let us examine the value vector of the vertex “man” (see table 39, row with gray background). It reads $(1, 0, 0)$, which indicates that function f_1 did not reach the desired goal. Let us apply the constraining function defined in previous section, now the value for the vertex reads $(0, 0, 0)$. After the application of goal and constraining functions the set of potential keys contains the vertices listed in table 39. Respectively, table 40 lists the set of potential keys of the database schema.

After the key attributes and their values are identified the database construction follows the steps introduced in “*7.3.3 Reconstructing the database*”, so here the detailed description of the process is omitted. Briefly described the reconstruction process does the following. The tuples and relations are built around elements of sets for potential keys, simply by traversing from key to its leaves. Moreover, the dependencies are determined by examining the adjacencies between elements belonging to these sets.

9 CONCLUDING REMARKS

This work examined the ways to expand the utilization potential of Adjacency Model (AM) and AdSchema. The main focus of the research efforts was on Adjacency Model and graph representations of relational databases. The work aimed to find the analogical features of AM and relational database. Moreover, the goal of this study was to provide definitions for the instances of relational database elements in AM representation of the database.

The key concepts of AM are adjacency relation system (ARS), adjacency defining sets and relation combination. ARS can be seen as a graph structure that is based on the concepts of type, element and adjacency. Each element represents a type and relationship between elements is called the adjacency (Wanne 1998:9-12.). Wanne (1998) and Töyli (2002, 2006) both have said that relational database and AM have analogical features. They both provided preliminary definitions for database concepts in AM. According to Wanne (1998), adjacency defining type corresponds with the key attributes of the relational database. Töyli (2002) pointed out that types of the AM represent the attributes of the database, and the elements of types represent the values of attributes. Töyli (2002) also stated that the concept of transitive adjacency gives an idea of a tuple in AM.

The definitions provided by Wanne and Töyli were a starting point the creation of identification criteria for the database elements in AM. In ARS an element representing the value of the key attribute is a member of adjacency defining type. In this work the properties of elements were examined from the graph theory point of view. An element representing the value of the key attribute typically has degree greater than two, it also has leaf vertices, and the element belongs to minimal vertex dominating set of the graph. Elements that are members of types that are not defined as adjacency defining types are considered as regular attributes. In ARS-graph regular attributes are leaf vertices. A concept of transitive adjacency gives an understanding about tuples in ARS. From a graph theory perspective, a tuple is a set of distinct elements that belong to the walk from vertex representing the value of the key attribute to it leaf vertices.

AdSchema provides information about the interrelationships between types of the AM. An AdSchema is utilized when relations are created, and tuples are assigned to relations. The types of AdSchema represent the key attributes and regular attributes of the database. In AdSchema key attribute is represented by the interrelationship defining type, and the other types represent regular attributes. An AdSchema graph can also be studied from graph theory perspective. Type representing key attribute has same properties as the elements representing the

values of key attributes. A relation is a set of distinct elements that belong to the walk from vertex representing the key attribute to it leaf vertices.

The AdSchema gives a view basic structure of the database. Dependencies between relations are defined by adjacencies between the elements of the key types. Information about the type the of dependency is achieved by examining the adjacency of elements of ARS-graph. Adjacency examinations reveal one-to-one, one-to-many dependencies, and many-to-many dependencies, which are deprecated in relational model. Thus, the need for relationship relation is also revealed by adjacency examinations.

The reconstruction process from ARS-based graph to relational database is based on the identification criteria outlined for the database elements. The reconstruction process is defined as follows. The first step of the process identifies types and elements that represent the key attributes and their values. The purposes of the second step is to form tuples based on ARS and define relations based on the AdSchema. After the tuples and relations are defined, tuples are assigned to relations either based on the ARS type definitions or based on the properties of the tuples, such as number of elements, type elements and adjacency of elements. Last step determines the dependencies between relations and defines foreign keys that implement the dependencies.

Since, the foreign keys were removed when the database was modeled by AM, the appropriate attributes are duplicated and assigned to relations. The last step also includes the creation of new relations, for example if ARS analysis indicates the existence of many-to-many dependency a relationship relation needs to be created.

The observations and methods presented in this work complement the Adjacency Model and AdSchema. Furthermore, the results of this work can be seen as framework of tools for analyzing graphs structures. The findings of this work can be used as a feasibility study which determines, if a given graph structure contains structures and features, which make is a suitable foundation for database construction.

The database construction process presented in this work has limitations. The ARS and AdSchema are static graph structures, so when the contents and structure of the database are represented with ARS and AdSchema, the representation is snapshot of the database. This may lead to situations where the reconstructed database does not correspond fully to the original one. For example, there is a rule which states that between two relations exists many-to-many dependency. The dependency is implemented by a relationship relation. Thus, if the ARS rep-

resentation of the database contains, at the time of the snapshot, only instances of one-to-many dependencies, then the reconstructed database does not have the relationship relation.

Key attribute recognition can be done easily, and the results are reliable, and both tuples and relations can be constructed with quite simple traversal algorithms. Even so, solutions for recognizing the exceptions, such as many-to-many or Boolean like attributes, rely mostly on human insight.

This work presents extensions for the definitions introduced by Wanne (1998) and Töyli (2002, 2006). Furthermore, in this work the ARS and AdSchema structures were analyzed from the graph theory aspect. The properties of dependencies between relations are determined from the adjacencies between the elements representing the values of key attributes. One future research area should aim to find ways to represent more information about the nature of the relationships directly in ARS. At this point, the identification of the database elements and database reconstruction are done manually. There is a need for algorithms that would automatize the identification and reconstruction process.

The first two future research areas focused mainly on the Adjacency Model. To support the future utilization of the results of this study in graph analysis, the possibilities of multi-objective optimization techniques should be studied more thoroughly. This means that the goal functions and their parameters have to be redefined. For example, the parameters defined in this work are quite plain. The combination of multi-objective optimization and community finding techniques (Newman & Girvan 2003) are fruitful future research area. Since a community in a graph and a tuple in an ARS-graph have certain similarities, tuples are communities that are formed around the key attribute, a successful combination of the techniques could provide tools for graph analysis.

References

Abiteboul, Serge (1997). Querying Semi-Structured Data. Proceedings of 6th International Conference Database Theory, pp. 1-8.

Black, Paul E. (2004) "data structure", in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. Cited 7.11.2011 Available from: http://www.nist.gov/dads/HTML/datastructur.html

Buneman, Peter, Susan Davidson and Dan Suciu (1995). Programming Constructs for Unstructured Data. Proceedings of the 1995 International Workshop on Database Programming Languages, pp. 1-12.

Buneman, Peter, Susan Davidson, Gerd Hillebrand and Dan Suciu (1996). A Query Language and Optimization Techniques for Unstructured Data. Technical Report MS-CIS 96-09.

Chen, Peter (1976). The Entity-Relationship Model Toward a Unified View of Data. ACM Transactions on Database Systems, 1:1, 9-36.

Codd, E., F. (1970). A Relational Model of Data for Large Shared Data Banks. In Communications of the ACM, 13:6, pp. 377-387.

Elmasri, Ramez and Shamkant B. Navathe (2000). Fundamentals of Database Systems. Reading: Addison-Wesley. ISBN 0-8053-1755-4.

Elmasri, Ramez and Shamkant B. Navathe (2007). Fundamentals of Database Systems. Boston: Addison Wesley. ISBN 0-321-41506-X.

Falley, Peter (2007). Categories of Data Structures. In Journal of Computing Sciences in Colleges - Papers of the Fourteenth Annual CCSC Midwestern Conference and Papers of the Sixteenth Annual CCSC Rocky Mountain Conference, 23:1, pp. 147-153.

Fernau, Henning (2010). Minimum dominating set of queens: A trivial programming exercise? In Discrete Applied Mathematics, 158:4, pp. 308-318.

Foulds, L., R. (1992). Graph Theory Applications. New York: Springer-Verlag.

Garcia-Molina, Hector, Jeffrey D. Ullman and Jennifer Widom (2002). Database Systems the Comple Book. Upper Saddle River, New Jersey: Prentice Hall.

Golfarelli, Matteo and Stefano Rizzi (2009). Data Warehouse Design: Modern Principles and Methodologies. New York: McGraw-Hill. ISBN 978-0-07-161039-1.

Goodaire, Edgar, G. and Michael M. Parmenter (2006). Discrete Mathematics with Graph Theory. Upper Saddle River, New Jersey: Pearson Prentice Hall. ISBN 978-0-13-167995-3.

Grandoni, Fabrizio (2006). A note on the complexity of minimum dominating set. In Journal of Discrete Algorithms, 4:2, pp. 209-214.

Gruber, Tom (2009). Ontology. In Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.). Springer-Verlag. Available from internet <URL: http://tomgruber.org/writing/ontology-definition-2007.htm>.

Hansen, Wilfred, J. (1981). The Structure of Data Structures. In ACM '81 Proceedings of the ACM '81 conference, pp. 89-95.

Hevner, Alan, R., Salvato T. March, Jinsoo Park and Sudha Ram (2004). Design Science in Information System Research. MIS Quarterly, 28:1, pp 75-105.

Heikkinen, Sari and Matti Linna (2004). The Adjacency Model and Wind Power. Proceedings of European Power and Energy Systems.

Hislop, Donald (2005). Knowledge management in organizations: a critical introduction. Oxford: Oxford University press.

Hovi, Ari, Henrikki Karvonen and Heikki Koistinen (2009). Tietovarastot ja business intelligence. Porvoo: WSOYpro/Docendo. ISBN 987-951-0-34792-8.

Jungnickel, Dieter (2002). Graphs, Networks and Algorithms (2nd ed). Algorithms and Computation in Mathematics vol 5. (eds E. Becker, M. Bronstein, H. Cohen, D. Eisenbud, R, Gilman). Berlin: Springer.

Jungnickel, Dieter (2013). Graphs, Networks and Algorithms (5th ed). Algorithms and Computation in Mathematics vol 5. (eds Manuel Bronstein, Arjeh M. Cohen, Henri Cohen, David Eisenbud, Bernd Sturmfels). Berlin: Springer.

Järvinen, Pertti (2012). On Research Methods. Tampere: Opinpajan Kirja. ISBN 978-952-99233-4-2.

Lampinen, Jouni (2000). Multiobjective Nonlinear Pareto-Optimization: A Pre-Investigation Reports. Lappeenranta: Lappeenranta University of Technology.

Lane, Paul (2007). Oracle Database - Data Warehousing Guide.

Manola, Frank and Eric Miller (2004). RDF Primer. [online] cited 6.9.2013. Available from the Internet: <URL: http://www.w3.org/TR/rdf-primer/>.

March, Salvatore, T. and Gerald F. Smith (1995). Design and Natural Science Research on Information Technology. Decision Support Systems, 15:4, pp. 251-266.

McHugh, J. S. Abiteboul, R. Goldman, D., Quass and J. Widom (1997). Lore: A Database Management System for Semistructured Data. ACM SIGMOD Record, 26:3, pp. 54-66.

Mäenpää, Teemu and Vesa Nyrhilä (2013a). Framework for Representing Semantic Link Network with Adjacency Relation System. Proceedings of the 2nd International Conference on Integrated Information, pp. 438-443.

Mäenpää, Teemu and Vesa Nyrhilä (2013b). Visualizing and Structuring Semantic Data. International Journal of Machine Learning and Computing, 3:2, pp. 209-213.

Newman, M., E., J. and M. Girvan (2003). Finding and Evaluating Community Structure in Networks. Physical review E, 69:2.

Newman, M., E., J. (2010). Networks – An Introduction. Oxford: Oxford University Press. ISBN 978-0-19-920665.

Ni, X. and S. Bloor (1994). Performance evaluation of boundary data structures. IEEE Computer Graphics and Applications 14:6, pp. 66-77.

Nyrhilä, Vesa, Teemu Mäenpää, Matti Linna and Erkki Antila (2005a). Information modeling in the case of distribution energy production. WSEAS Transactions on communications, 12:4.

Nyrhilä, Vesa, Teemu Mäenpää, Matti Linna and Erkki Antila (2005b). A novel information model for distribution energy production. Proceedings of the WSEAS conferences: 5th WSEAS Int. Conf. on Power Systems and electromagnetic compatibility (PSE ‘05).

Pan, Shan, L. and Barney Tan (2011). Demystifying case research: A structured-pragmatic-situational (SPS) approach to conducting case studies. Information and Organization, 21:3, pp. 161-167.

Papakonstantinou, Yannis, Hector Garcia-Molina and Jennifer Widom (1995). Object exchange across heterogeneous information sources. Proceedings of the 11th International Conference on Data Engineering, pp. 251-260.

Price, Kenneth, V., Rainer M. Storm and Jouni A. Lampinen (2005). Differential evolution – A practical approach to global optimization. Berlin: Springer-Verlag.

Savolainen, Vesa (1978). Verkkoteorian perusteet ja algoritmit. ISBN 951-662-237-2.

Silberschatz, Abraham, Henry Korth and S. Sudarshan (2010). Database System Concepts. New York: McGraw-Hill. ISBN 978-0-07-352332-3.

Simon, Herbert, A. (1996). The Sciences of the Artificial. 3rd ed. Cambridge, USA: The MIT Press. ISBN 9780262193740.

Strauss, Anselm L. and Juliet M. Corbin (1998). Basic of Qualitative Research – Techniques and Procedures for Developing Grounded Theory. 2nd edition. Thousand Oaks, CA, USA: Sage Publications. ISBN 9780585383323.

Suciu, Dan (1998). An Overview of Semistructured Data. ACM SIGACT News, vol. 29 no. 4, pp. 28-38.

Töyli, Jari (2002). Modeling semistructured data by the adjacency model. Vaasa: University of Vaasa.

Töyli, Jari, Matti Linna and Merja Wanne (2002a). Modeling Relational Data by the Adjacency Model. Proceedings of the Fifth Joint Conference on Knowledge-Based Software Engineering, pp. 301-306.

Töyli, Jari, Matti Linna and Merja Wanne (2002b). Modeling Semistructured Data by the Adjacency Model. Proceedings of the Fourth International Conference on Enterprise Information Systems, pp. 282-290.

Töyli, Jari (2006). AdSchema – a schema for semistructured Data. Acta Wasaensia, 157: Computer Science 5. Vaasa: Vaasan yliopisto. ISBN 952-476-131-9.

Vaidya, P., M. (1989). An O(n log n) Algorithm for the All-Nearest-Neighbors Problem. In Discrete & computational geometry, 4:2, pp. 101-116.

Van Aken, Joan, E. (2004). Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. Journal of Management Studies, 41:2, pp. 221-246.

Walsham, Geoff (2006). Doing interpretive research. European Journal of Information Systems, 15:3, pp. 320-330.

Wanne, Merja (1998). Adjacency relation systems. Acta Wasaensia, 60: Computer Science 1. Vaasa: Vaasan yliopisto. ISBN 951-683-703-4.

Wanne, Merja and Matti Linna (1999). A General Model for Adjacency. Fundamenta Informaticae, 38:1-2, pp. 39-50

Weiler, K. (1985). Edge-based data structures for solid modeling in curved-surface environments. IEEE Computer Graphics and Applications, 5:1, pp. 21-40.

Wilf, Herbert, S. (1994). Algorithms and Complexity. Philadelphia: University of Pennsylvania. Available from internet <URL: https://www.math.upenn.edu/~wilf/AlgoComp.pdf>.

Williams, Barry (2012). Data Model for UK's Open University. Available from the internet <URL: http://www.databaseanswers.org/data_models/open_university/index.htm>

Zadravec, Mirko, Andrej Brodnik, Markus Mannila, Merja Wanne and Borut Zalik (2008) A practical approach to the 2D incremental nearest-point problem suitable for different point distributions. Pattern Recognition 41:2, pp. 646-653.

Zhuge, Hai (2003). Active e-document framework ADF: model and tool. Information & Management, 41:1, pp. 87-97.

Zhuge, Hai (2004). The Knowledge Grid. Singapore: World Scientific. ISBN 981-256-140-4.

Zhuge, Hai (2005). The Knowledge Grid. Singapore: World Scientific. ISBN: 981-256-140-4.

Zhuge, Hai, Yunchuan Yun, Ruixiang Jia and Jie Liu (2005). Algebra model and experiment for semantic link network. International Journal of High Performance Computing and Networking, 3:4, pp. 227-238.

Zhuge, Hai (2011). Semantic linking through spaces for cyber-physical-socio intelligence: A methodology. Artificial Intelligence, 175:5-6, pp. 988-1019.

Appendix

Appendix 1. Analyses of the properties of the relational database primary key attributes in AdSchema based graph.

Database schema can be presented as AdSchema. AdSchema reveals the basic structure of the database, which makes it possible to examine the properties of the AdSchema elements. In this appendix, database schema diagrams are converted into AdSchema format. The following properties for the vertices of the AdSchema-based graph are examined.

1) degree of a vertex,

2) does vertex have leaves and

3) does vertex belong to the minimal vertex dominating set of the graph.

The observations for each case are represented in the tables following the AdSchema. The tables also contain information whether the node in AdSchema is defined as the primary key in database schema diagram. The observations reveal that the nodes representing the primary keys have a certain pattern in their properties.

The last three cases cover situation when the AdSchema is produced from a smaller database i.e. tables contain smaller number of attributes. In these cases, some of the primary keys do not have leaves, and they do not belong to the dominating set of the graph. The observations regarding all the key nodes in AdSchemas are shown in the next table.

Table 41. Properties for vertices representing key element types in AdSchemas.

Number of AdSchemas	Number of types representing keys attributes	Average vertex degree	Vertices with d>2	Parent vertices	Members of dominating set
15	120	4,04	65 %	98 %	98 %

Case 1. Patient monitoring system (Williams 2012a).

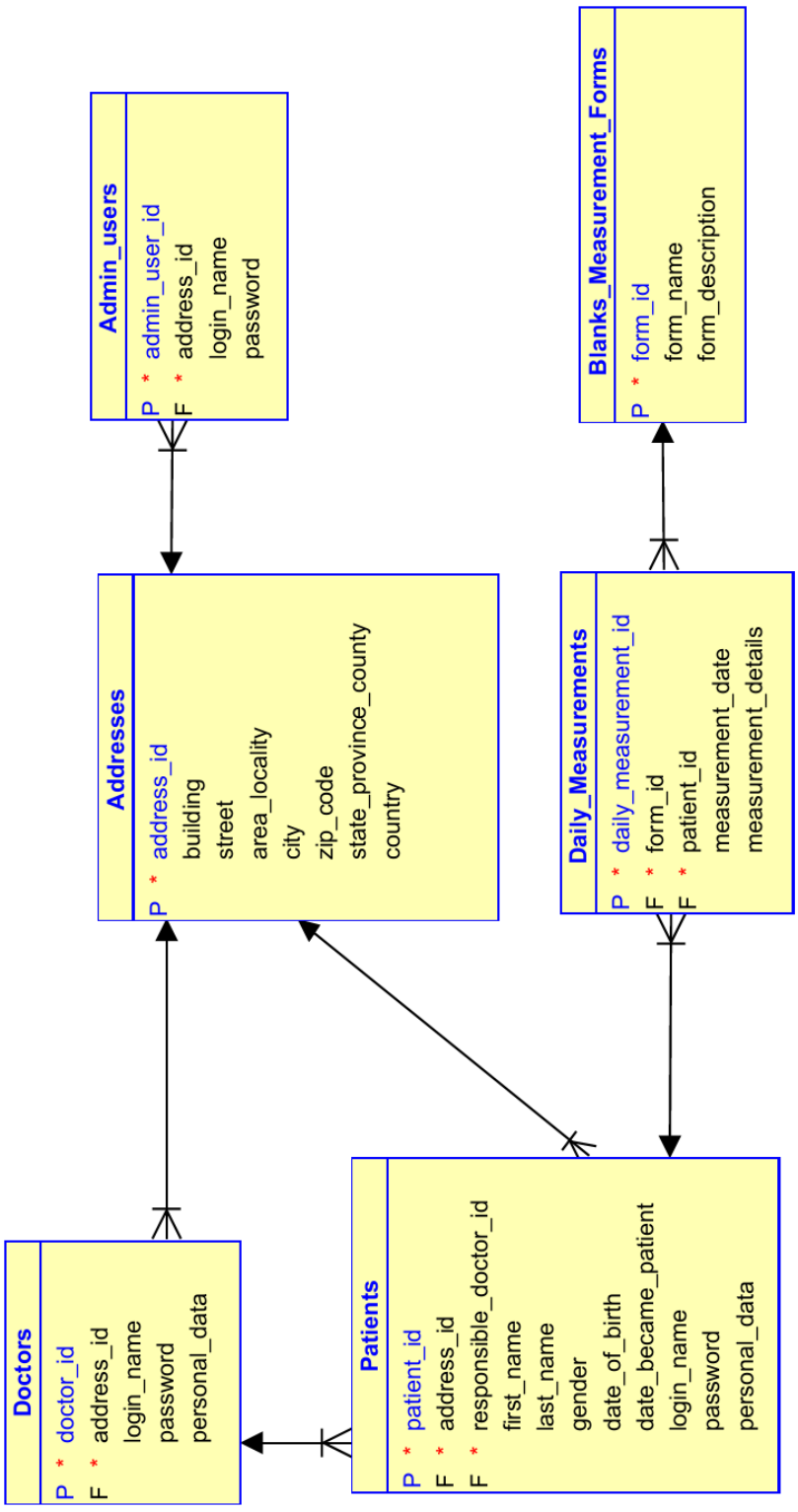


Figure 69. Database diagram for patient monitoring system.

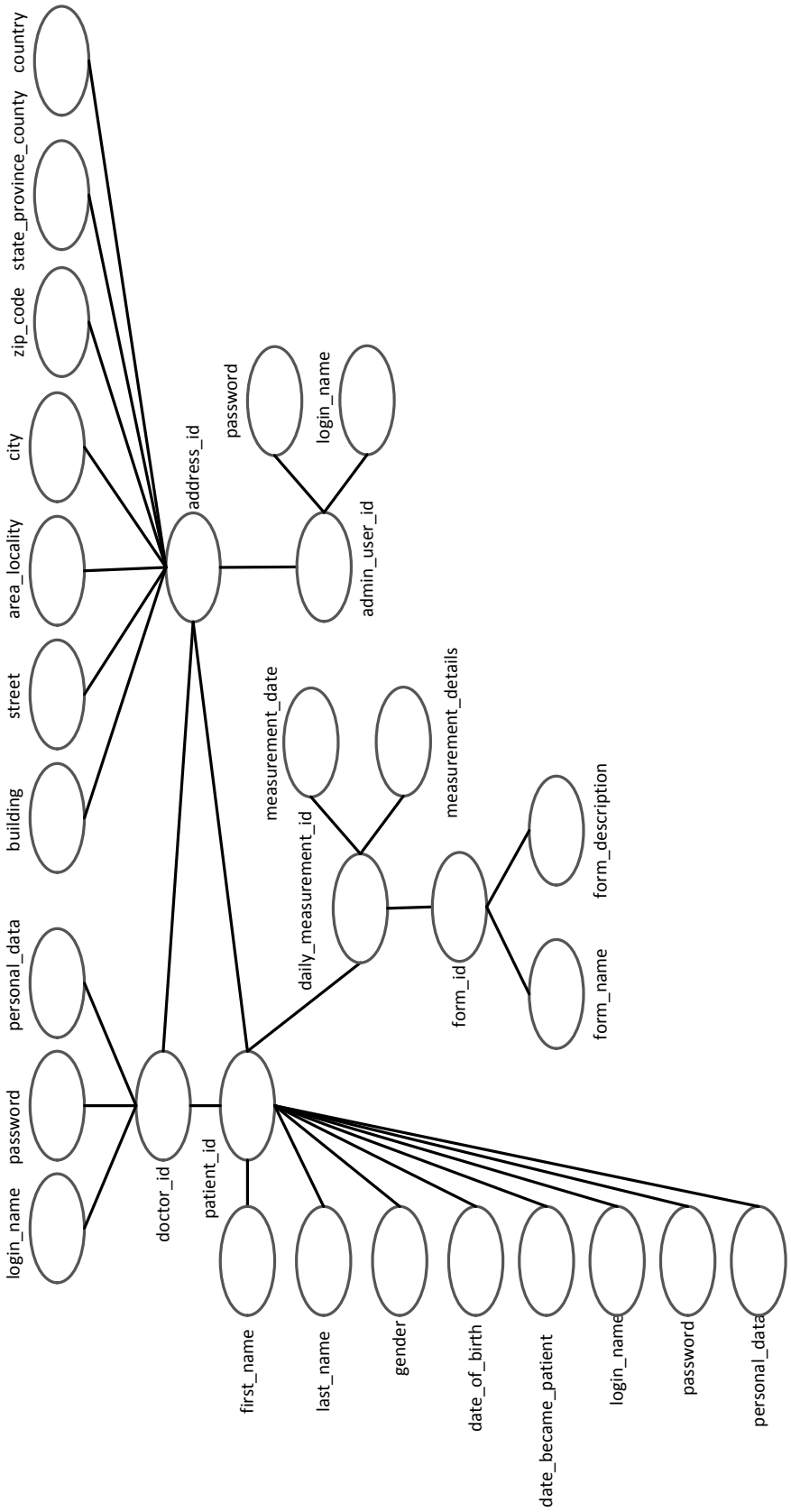


Figure 70. AdSchema for patient monitoring system.

Table 42. Properties for vertices in AdSchema.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	login_name		1		
2	password		1		
3	personal_data		1		
4	doctor_id	x	5	x	x
5	building		1		
6	street		1		
7	area_locality		1		
8	city		1		
9	zip_code		1		
10	state_province_county		1		
11	country		1		
12	address_id	x	10	x	x
13	patient_id	x	11	x	x
14	first_name		1		
15	last_name		1		
16	gender		1		
17	date_of_birth		1		
18	date_became_patient		1		
19	login_name		1		
20	password		1		
21	personal_data		1		
22	daily_measurement_id	x	4	x	x
23	measurement_date		1		
24	measurement_details		1		
25	form_id	x	3	x	x
26	form_name		1		
27	form_description		1		
28	admin_user_id	x	3	x	x
29	password		1		
30	login_name		1		

Case 2. Afghanistan rainfall surveying system (Williams 2004a).

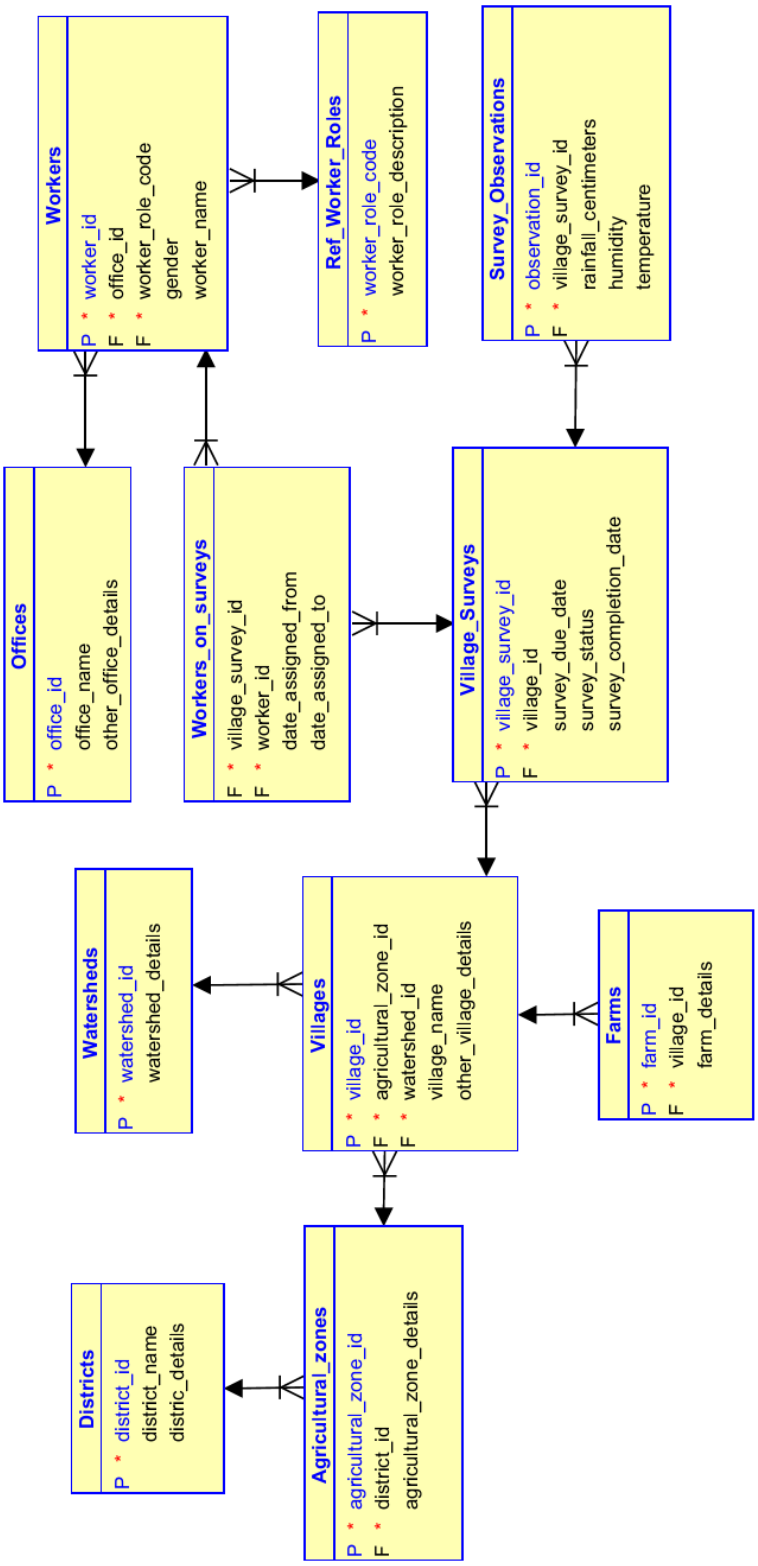


Figure 71. Database schema diagram for Afghanistan rainfall surveying system.

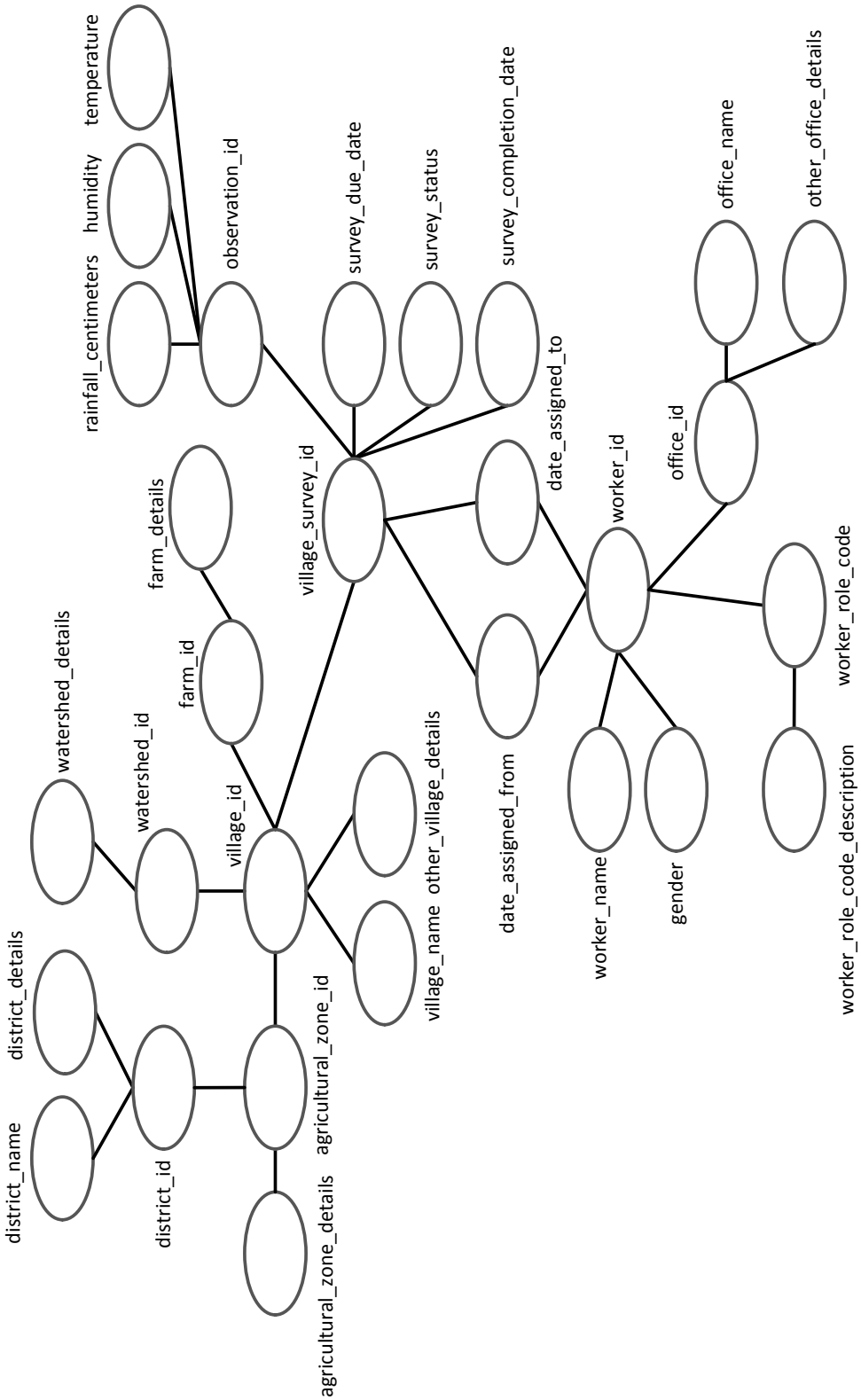


Figure 72. AdSchema for Afghanistan rainfall surveying system.

Table 43. Properties for vertices in AdSchema.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	agricultural_zone_details		1		
2	agricultural_zone_id	x	3	x	x
3	date_assigned_from		2		
4	date_assigned_to		2		
5	distric_details		1		
6	district_id	x	3	x	x
7	district_name		1		
8	farm_details		1		
9	farm_id	x	2	x	x
10	gender		1		
11	humidity		1		
12	observation_id	x	4	x	x
13	office_id	x	3	x	x
14	office_name		1		
15	other_office_details		1		
16	other_village_details		1		
17	rainfall_centimeters		1		
18	survey_due_date		1		
19	survey_completion_date		1		
20	survey_status		1		
21	temperature		1		
22	village_id	x	6	x	x
23	village_name		1		
24	village_survey_id	x	7	x	x
25	watershed_details		1		
26	watershed_id	x	2	x	x
27	worker_id	x	6	x	x
28	worker_name		1		
29	worker_role_code	x	2	x	x
30	worker_role_description		1		

Case 3. Longterm accommodation (Williams 2001a).

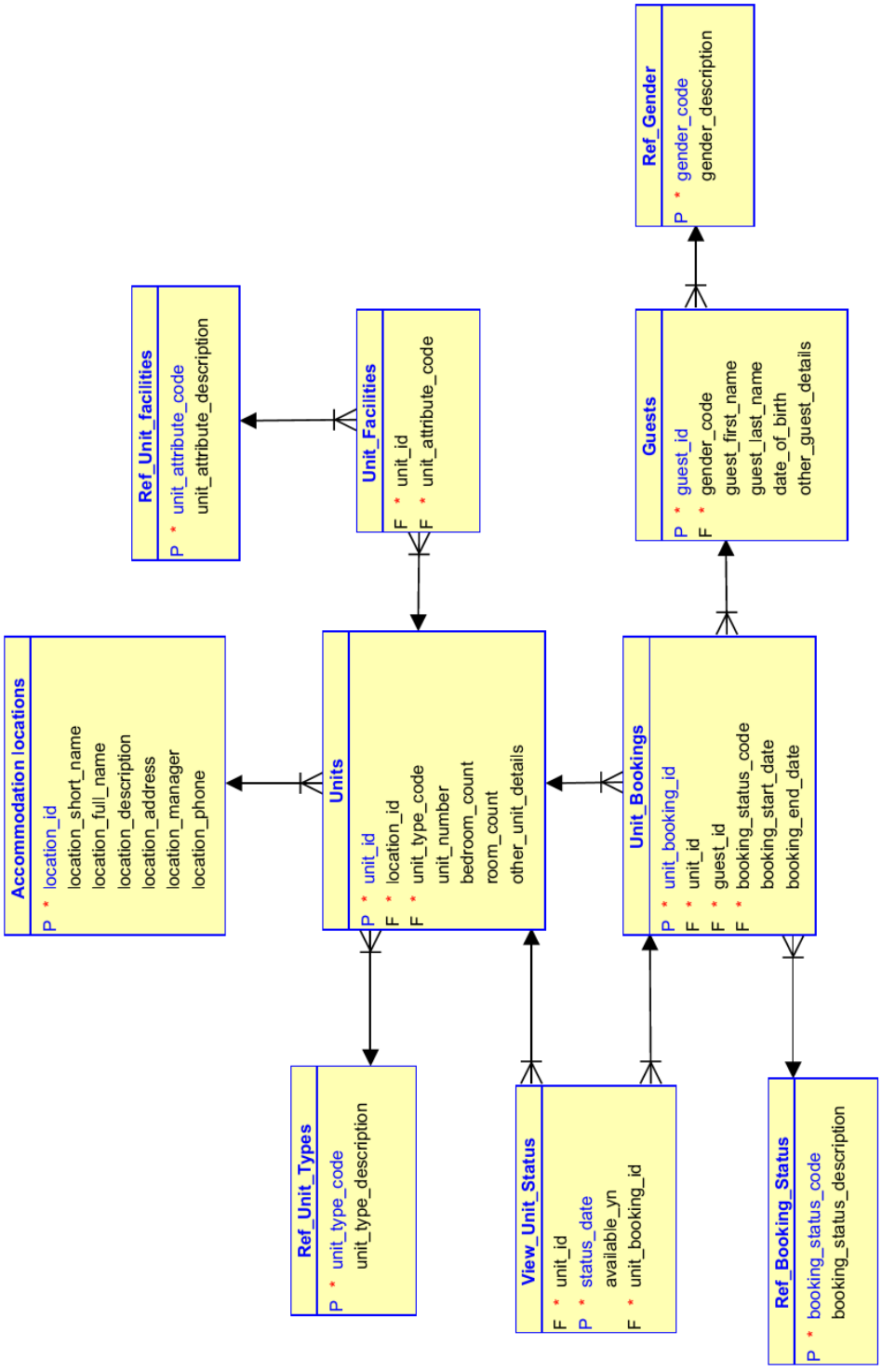


Figure 73. Database schema diagram for longterm accommodation system.

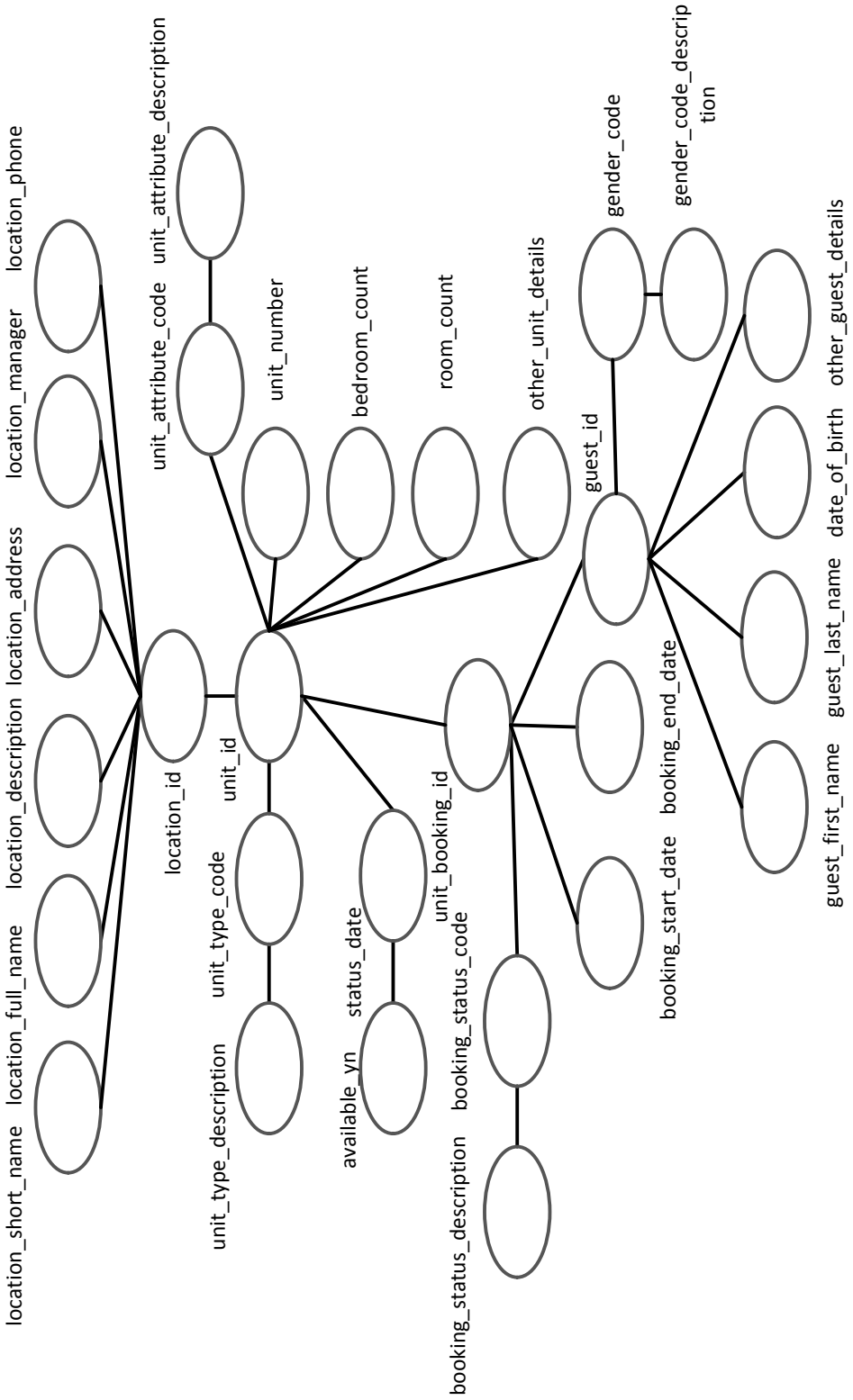


Figure 74. . AdSchema for longterm accommodation system.

Table 44. Properties of vertices in AdSchema graph.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	bedroom_count		1		
2	booking_end_date		1		
3	booking_start_date		1		
4	booking_status_code	x	2	x	x
5	booking_status_description		1		
6	date_of_birth		1		
7	guest_first_name		1		
8	guest_id	x	6	x	x
9	guest_last_name		1		
10	location_address		1		
11	location_description		1		
12	location_full_name		1		
13	location_id	x	7	x	x
14	location_manager		1		
15	location_phone		1		
16	location_short_name		1		
17	other_guest_details		1		
18	other_unit_details		1		
19	room_count		1		
20	status_date	x	2	x	x
21	unit_attribute_code	x	2	x	x
22	unit_attribute_description		1		
23	unit_booking_id	x	5	x	x
24	unit_id	x	9	x	x
25	unit_number		1		
26	unit_type_code	x	2	x	x
27	unit_type_description		1		
28	gender_code	x	2	x	x
29	gender_description		1		
30	available_yn		1		

Case 4. Accidents at work (Williams 2001b).

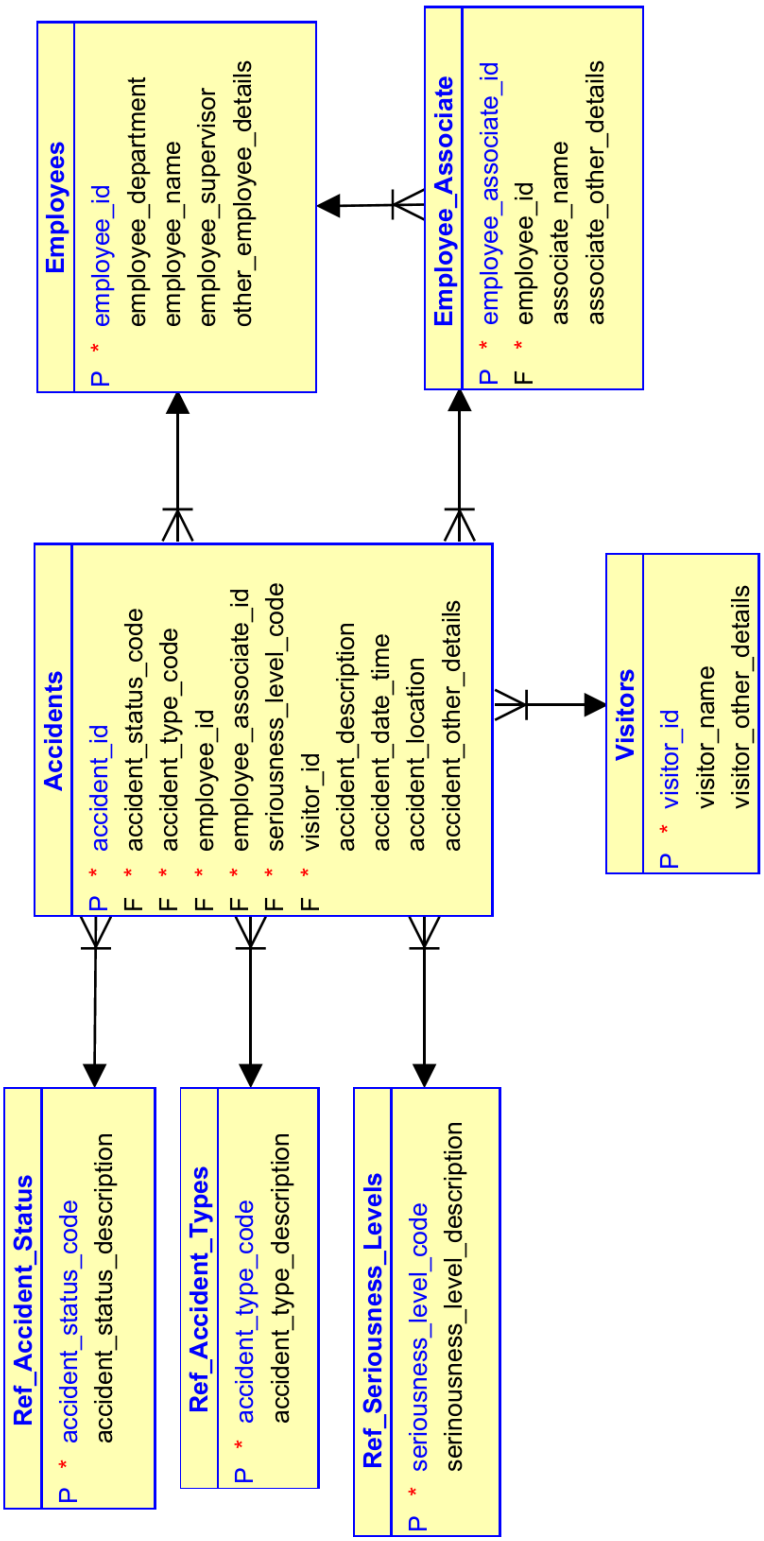


Figure 75. Database schema diagram for Accidents at work.

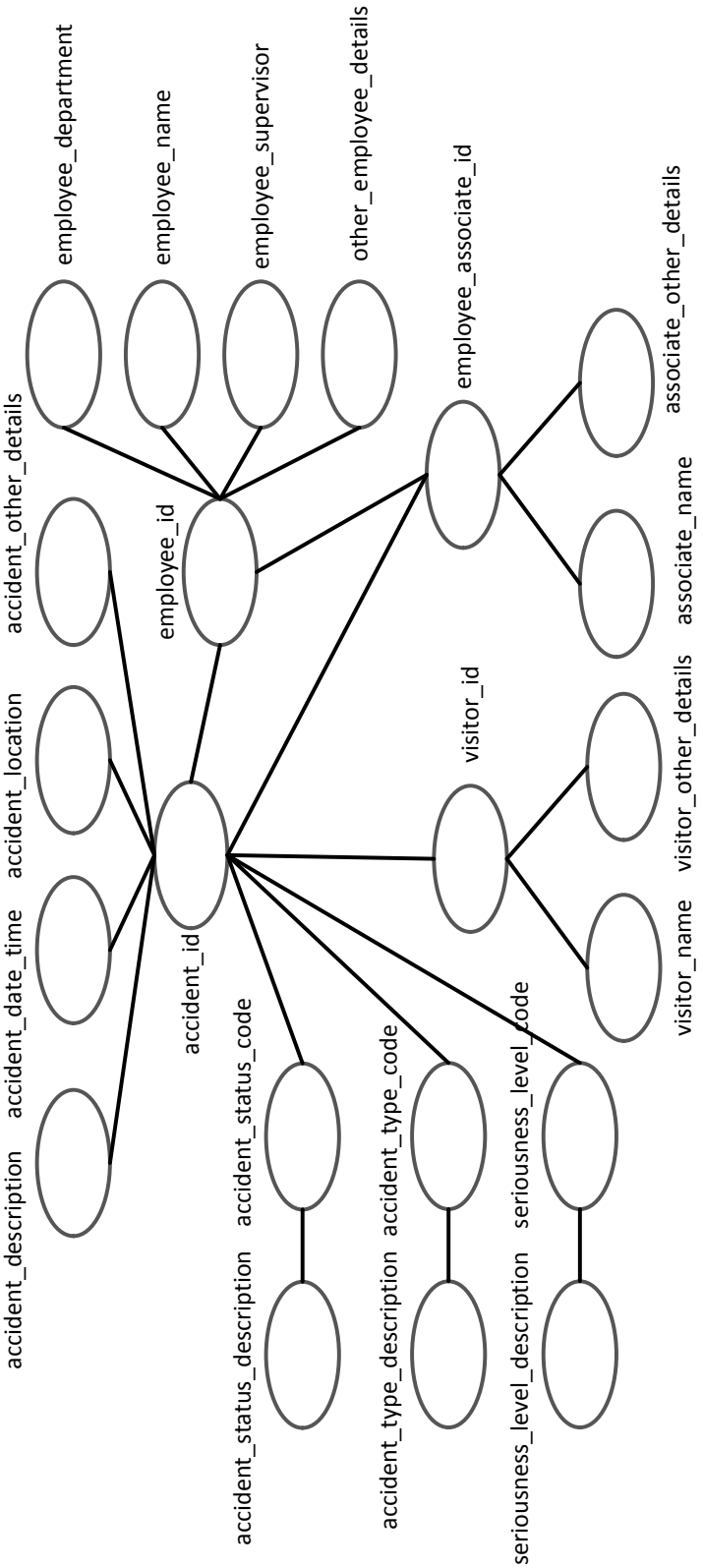


Figure 76. AdSchema for Accidents at work.

Table 45. Properties of attribute nodes in AdSchema.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	accident_date_time		1		
2	accident_description		1		
3	accident_id	x	10	x	x
4	accident_location		1		
5	accident_other_details		1		
6	accident_status_code	x	2	x	x
7	accident_status_description		1		
8	accident_type_code	x	2	x	x
9	accident_type_description		1		
10	associate_name		1		
11	associate_other_details		1		
12	employee_associate_id	x	4	x	x
13	employee_department		1		
14	employee_id	x	6	x	x
15	employee_name		1		
16	employee_supervisor		1		
17	other_employee_details		1		
18	seriousness_level_description		1		
19	seriousness_level_code	x	2	x	x
20	visitor_id	x	3	x	x
21	visitor_name		1		
22	visitor_other_details		1		

Case 5. Customer deliveries system (Williams 2011).

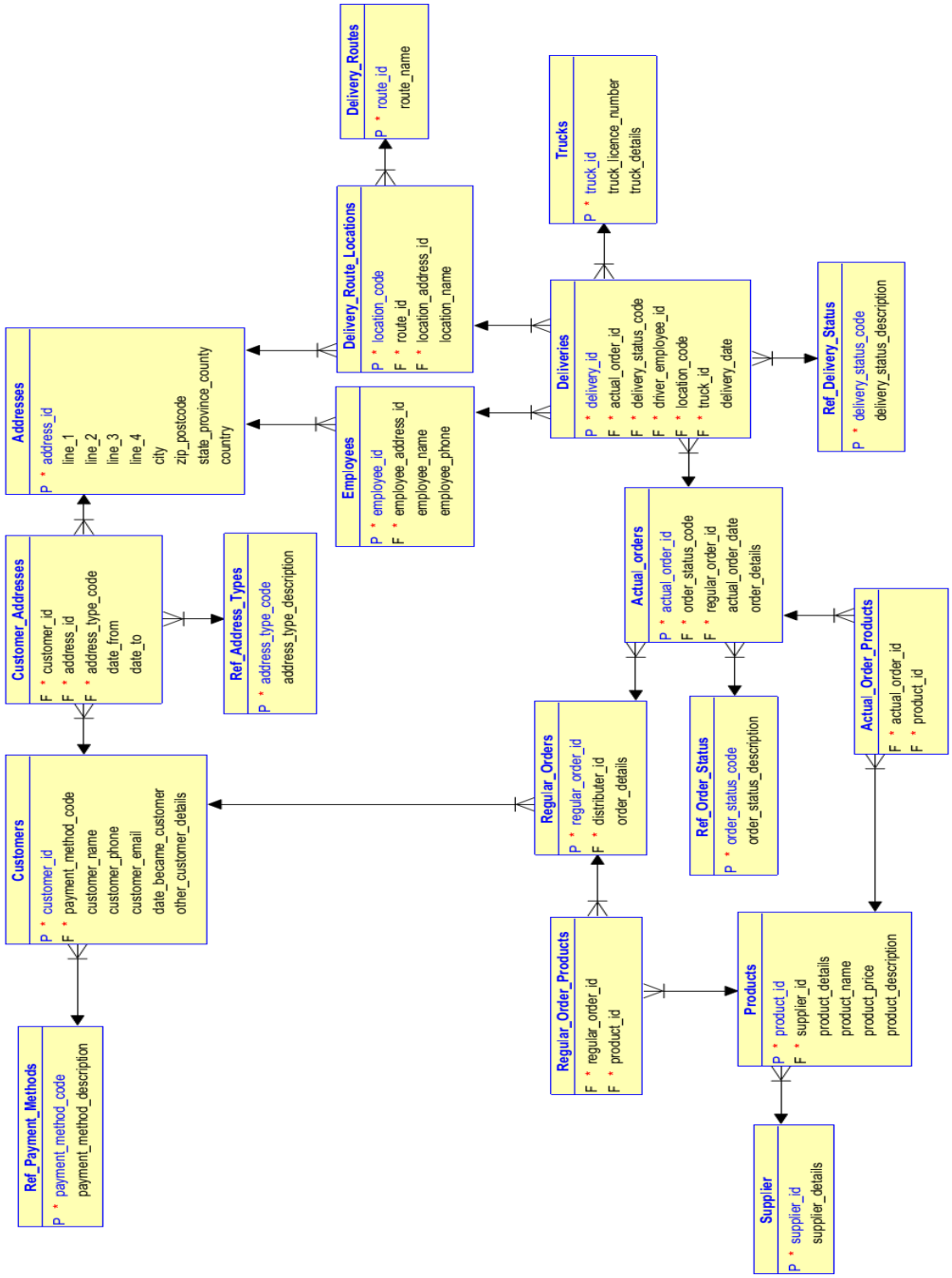


Figure 77. Database schema diagram for customer deliveries system.

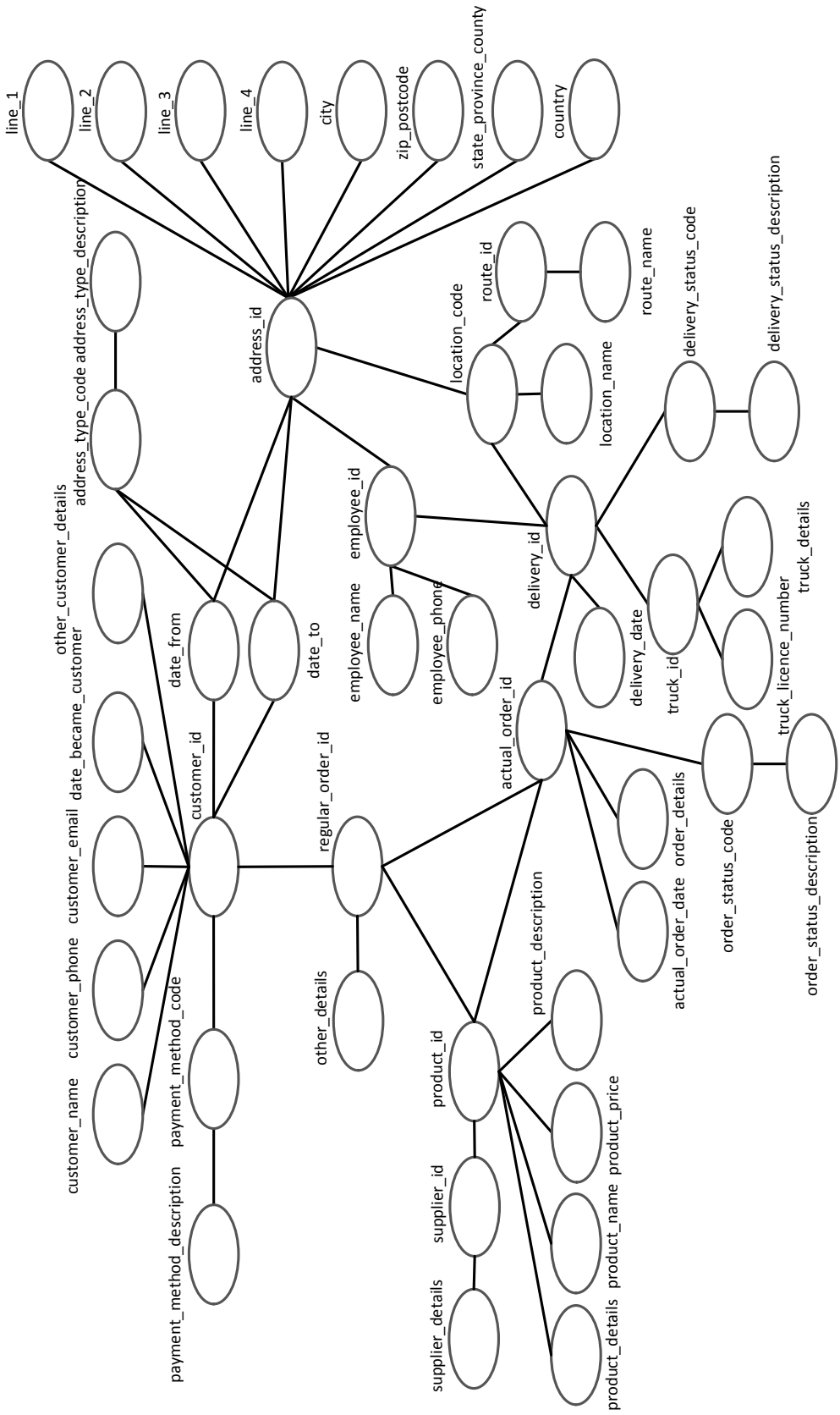


Figure 78. AdSchema for customer deliveries system.

Table 46. Vertex properties in AdSchema.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	actual_order_date		1		
2	actual_order_id	x	6	x	x
3	address_id	x	12	x	x
4	address_type_code	x	3	x	x
5	address_type_description		1		
6	city		1		
7	country		1		
8	customer_email		1		
9	customer_id	x	9	x	x
10	customer_name		1		
11	customer_phone		1		
12	date_became_customer		1		
13	date_from		3		
14	date_to		3		
15	delivery_date		1		
16	delivery_id	x	6	x	x
17	delivery_status_code	x	2	x	x
18	delivery_status_description		1		
19	employee_id	x	4	x	x
20	employee_name		1		
21	employee_phone		1		
22	line_1		1		
23	line_2		1		
24	line_3		1		
25	line_4		1		
26	location_code	x	4		x
27	location_name		1		
28	order_details		1		
29	order_details		1		
30	order_status_code	x	2	x	x
31	order_status_description		1		
32	other_customer_details		1		
33	payment_method_code	x	2	x	x
34	payment_method_description		1		
35	product_description		1		
36	product_details		1		
37	product_id	x	7	x	x

38	product_name		1		
39	product_price		1		
40	regular_order_id	x	4	x	x
41	route_id	x	2	x	x
42	route_name		1		
43	state_province_county		1		
44	supplier_id	x	2	x	x
45	supplier_details		1		
46	truck_details		1		
47	truck_id	x	3	x	x
48	truck_licence_number		1		
49	zip_postcode		1		

Case 6. School management system (Williams 2004b).

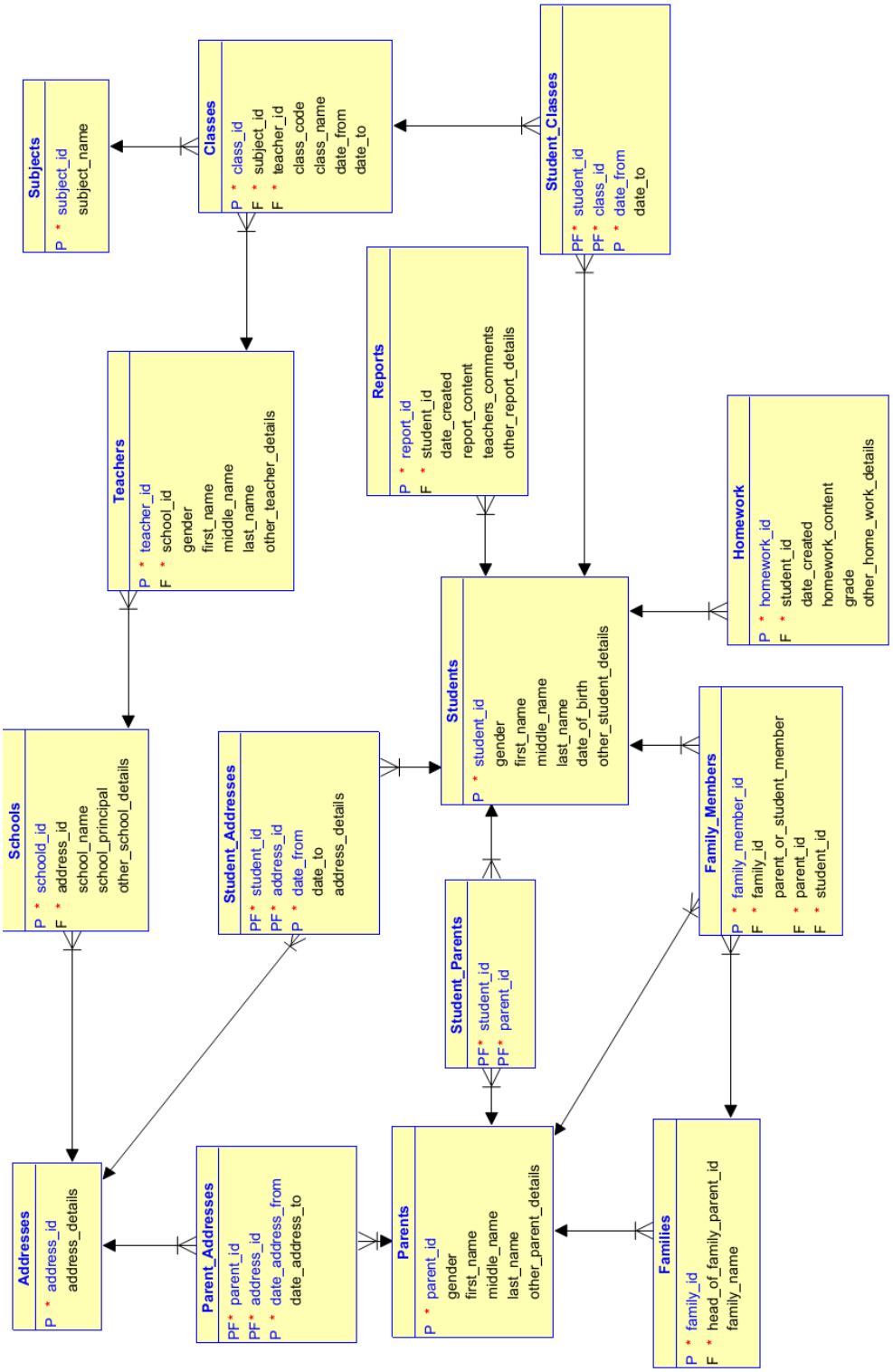


Figure 79. Database schema diagram for school management system.

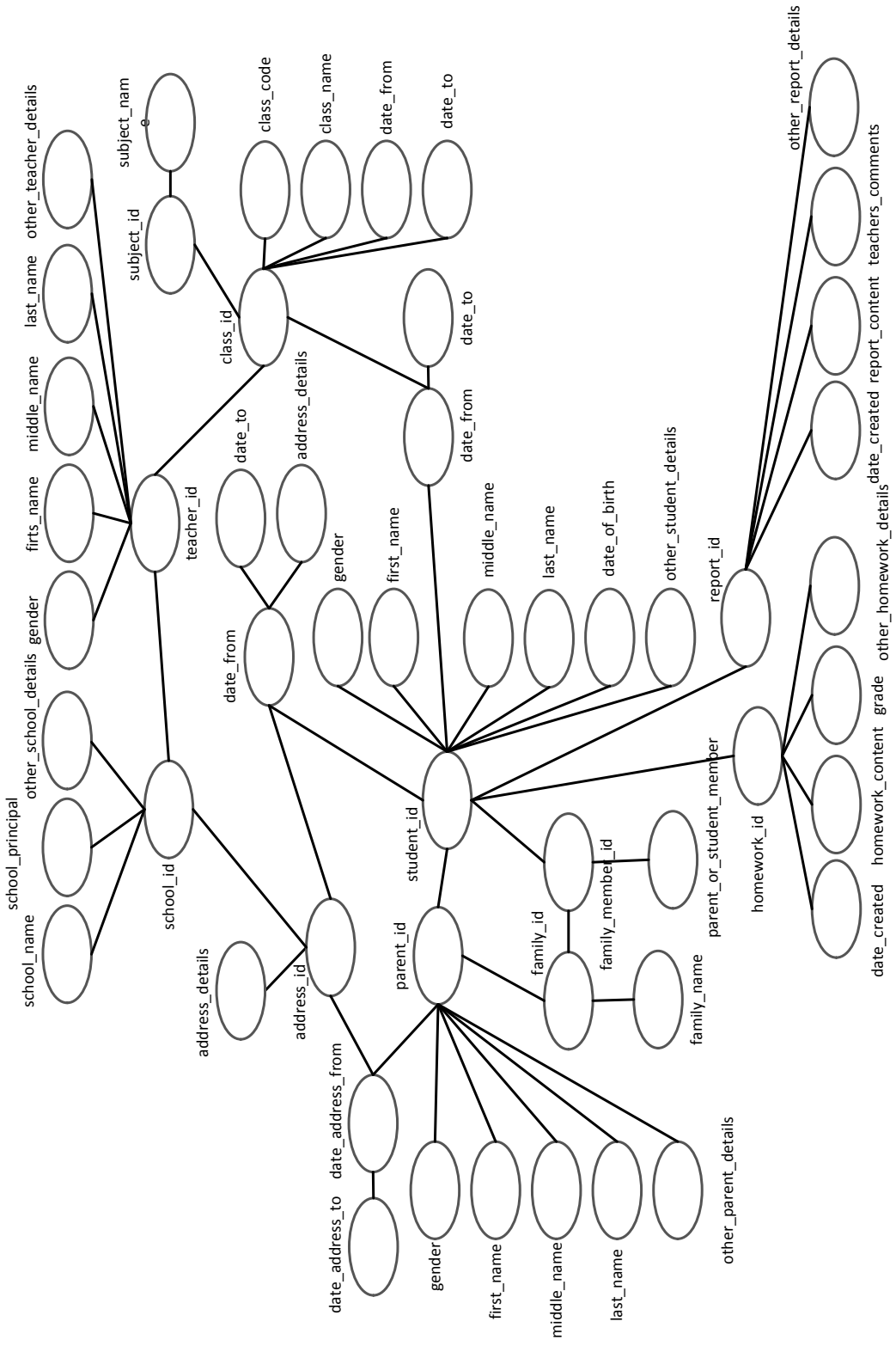


Figure 80. AdSchema for school management system.

Table 47. Vertex properties.

Node index	Node name	AdSchema properties			
		Primary key in database	Degree	Has leaves	Member of dominating set
1	address_id	x	4	x	x
2	address_details		1		
3	address_details		1		
4	class_code		1		
5	class_id	x	7	x	x
6	class_name		1		
7	date_address_from	x	3	x	x
8	date_address_to		1		
9	date_created		1		
10	date_created		1		
11	date_from	x	4	x	x
12	date_from		1		x
13	date_from	x	3	x	x
14	date_of_birth		1		
15	date_to		1		
16	date_to		1		
17	date_to		1		
18	family_id	x	3	x	x
19	family_member_id	x	3	x	x
20	family_name		1		
21	first_name		1		
22	first_name		1		
23	first_name		1		
24	gender		1		
25	gender		1		
26	gender		1		
27	grade		1		
28	homework_content		1		
29	homework_id	x	5	x	x
30	last_name		1		
31	last_name		1		
32	last_name		1		
33	middle_name		1		
34	middle_name		1		
35	middle_name		1		
36	other_home_work_details		1		
37	other_parent_details		1		

38	other_report_details		1		
39	other_school_details		1		
40	other_student_details		1		
41	other_teacher_details		1		
42	parent_id	x	8	x	x
43	parent_or_student_member		1		
44	report_content		1		
45	report_id	x	5	x	x
46	school_id	x	5	x	x
47	school_name		1		
48	school_principal		1		
49	student_id	x	12	x	x
50	teacher_id	x	7	x	x
51	teachers_comments		1		
52	subject_id	x	2	x	x
53	subject_name		1		

Case 7. Apartment rental system (Williams 2004c).

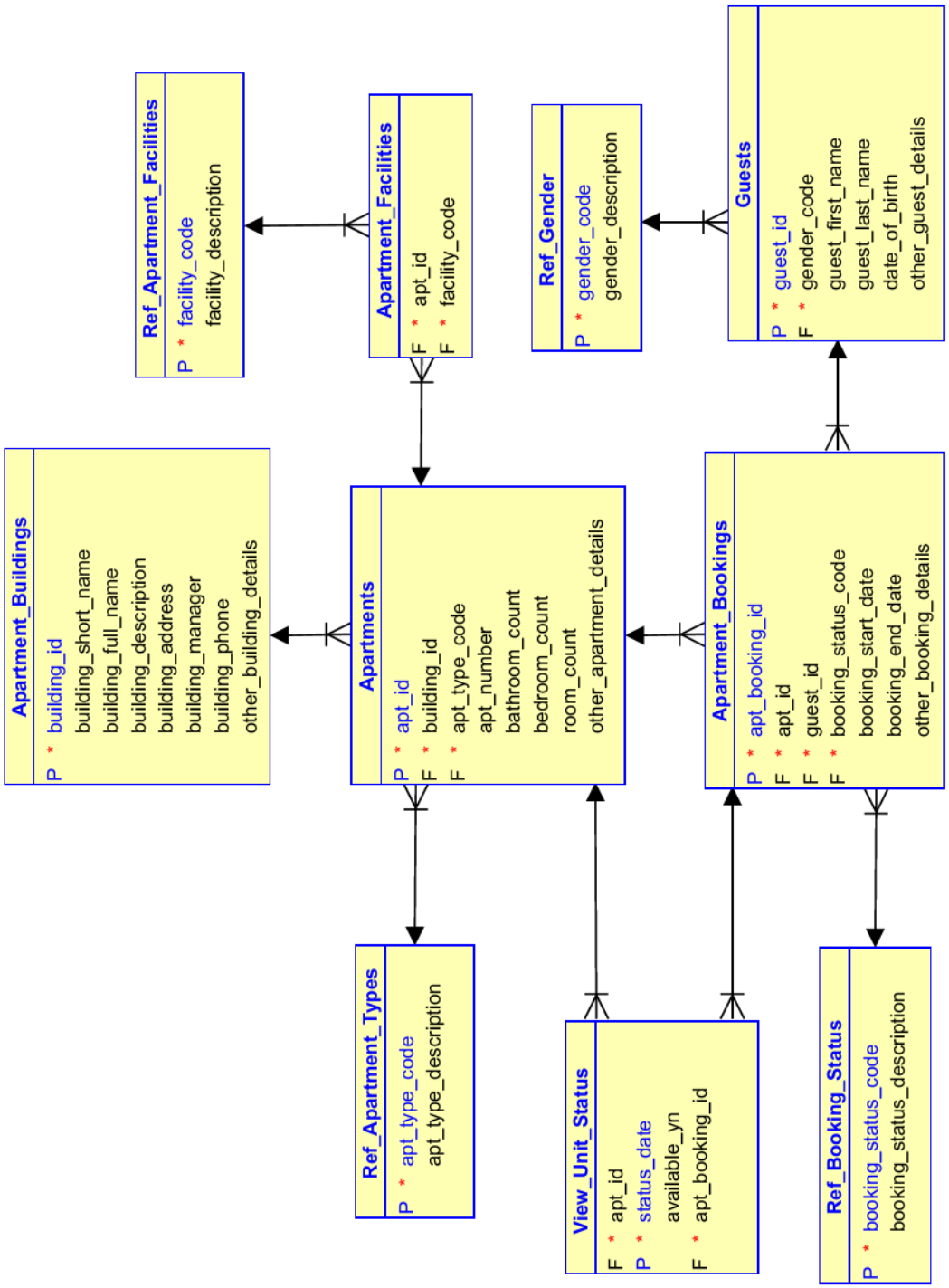


Figure 81. Database schema diagram for apartment rental system.

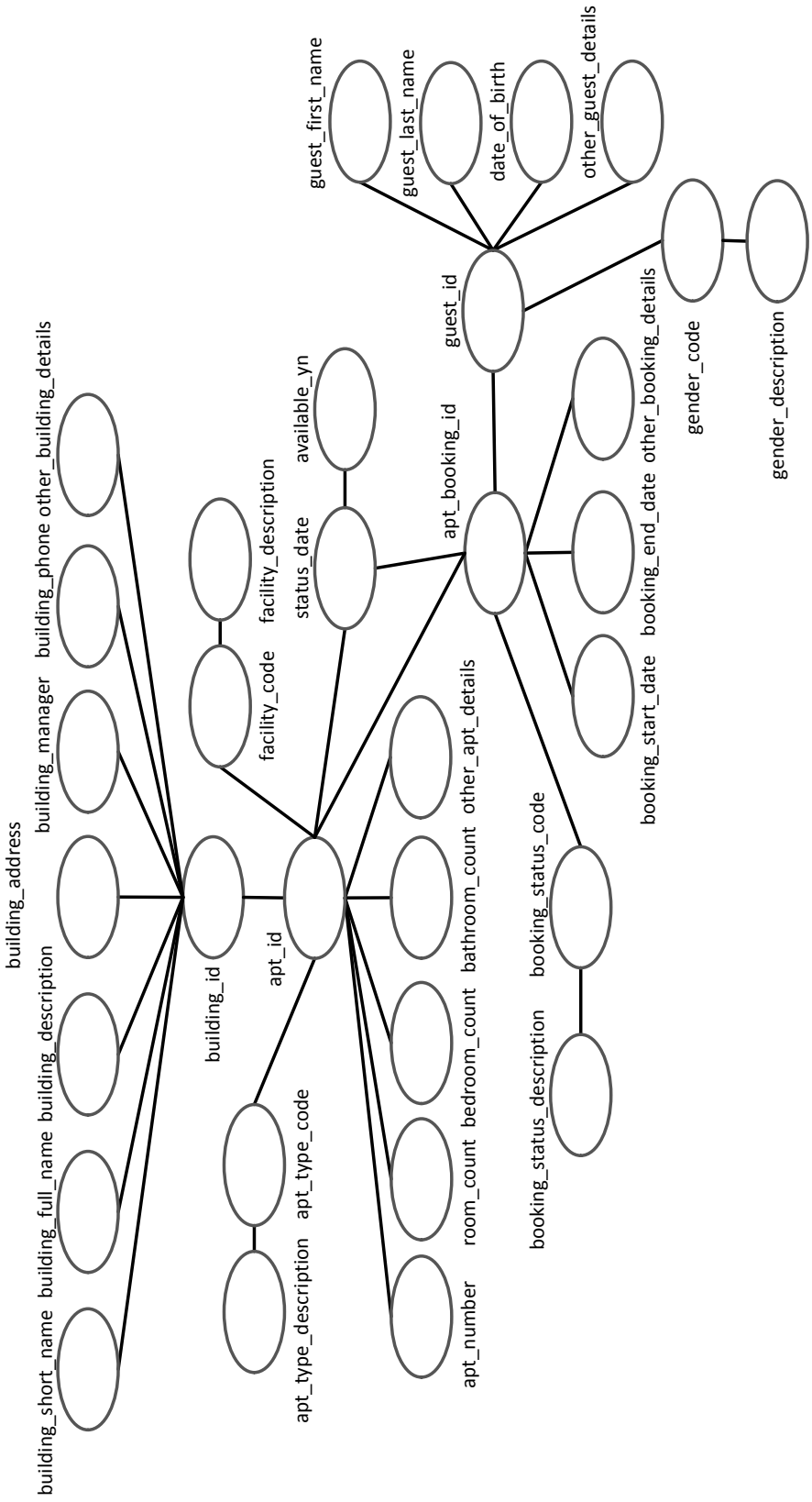


Figure 82. AdSchema diagram for apartment rental system.

Table 48. Vertex properties.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	apt_booking_id	x	7	x	x
2	apt_id	x	10	x	x
3	apt_number		1		
4	apt_type_code	x	2	x	x
5	apt_type_description		1		
6	available_yn		1		
7	bathroom_count		1		
8	bedroom_count		1		
9	booking_end_date		1		
10	booking_start_date		1		
11	booking_status_code	x	2	x	x
12	booking_status_description		1		
13	building_address		1		
14	building_description		1		
15	building_full_name		1		
16	building_id	x	8	x	x
17	building_manager		1		
18	building_phone		1		
19	building_short_name		1		
20	date_of_birth		1		
21	facility_code	x	2	x	x
22	facility_description		1		
23	gender_code	x	2	x	x
24	guest_first_name		1		
25	guest_id	x	6	x	x
26	guest_last_name		1		
27	other_apartment_details		1		
28	other_booking_details		1		
29	other_building_details		1		
30	other_guest_details		1		
31	room_count		1		
32	status_date	x	3	x	x
33	gender_description		1		

Case 8. Occupational Health System (Williams 2014a).

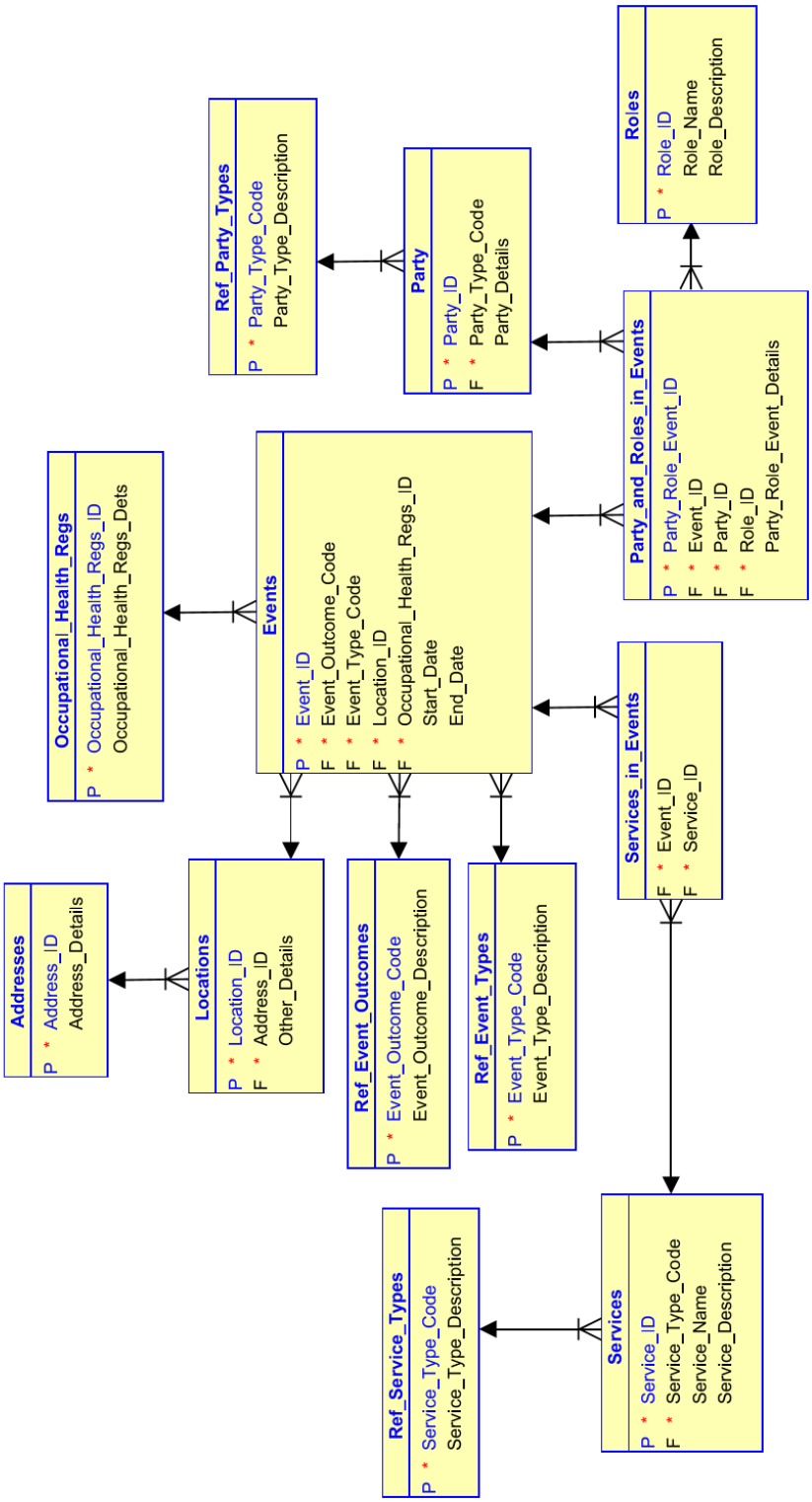


Figure 83. Datamodel for Occupational Health System.

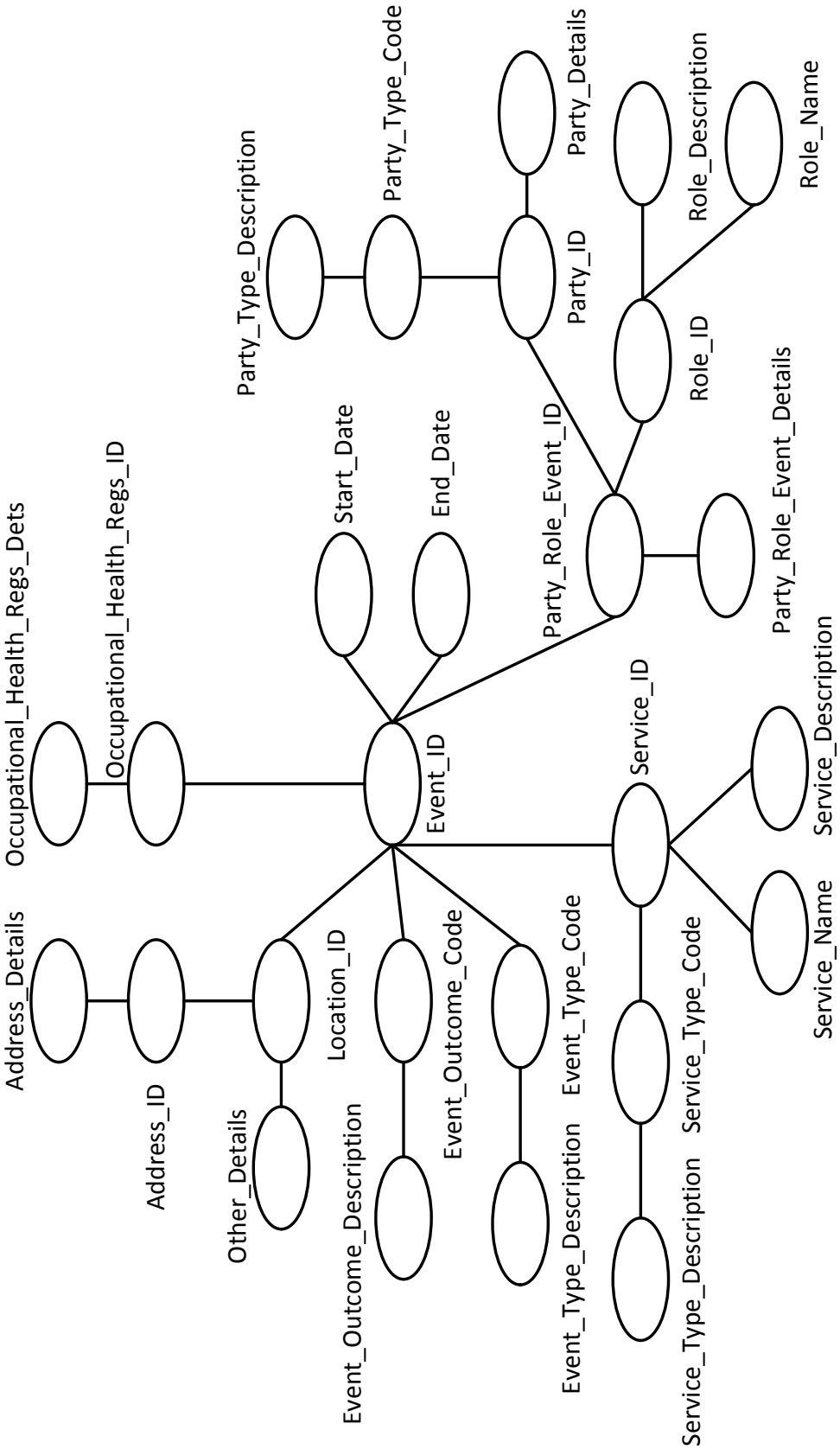


Figure 84. AdSchema for Occupational Health System.

Table 49. Vertex properties of AdSchema.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	Address_Details		1		
2	Address_ID	x	2	x	x
3	End_Date		1		
4	Event_ID	x	8	x	x
5	Event_Outcome_Code	x	2	x	x
6	Event_Outcome_Description		1		
7	Event_Type_Code	x	2	x	x
8	Event_Type_Description		1		
9	Location_ID	x	3	x	x
10	Occupational_Health_Regs_Dets		1		
11	Occupational_Health_Regs_ID	x	2	x	x
12	Other_Details		1		
13	Party_Details		1		
14	Party_ID	x	3	x	x
15	Party_Role_Event_Details		1		
16	Party_Role_Event_ID	x	4	x	x
17	Role_Description		1		
18	Role_ID	x	3	x	x
19	Role_Name		1		
20	Service_Description		1		
21	Service_ID	x	4	x	x
22	Service_Name		1		
23	Service_Type_Code	x	2	x	x
24	Service_Type_Description		1		
25	Start_Date		1		
26	Party_Type_Code	x	2	x	x
27	Party_Type_Description		1		

Case 9. Parking Ticket System (Williams 2009a).

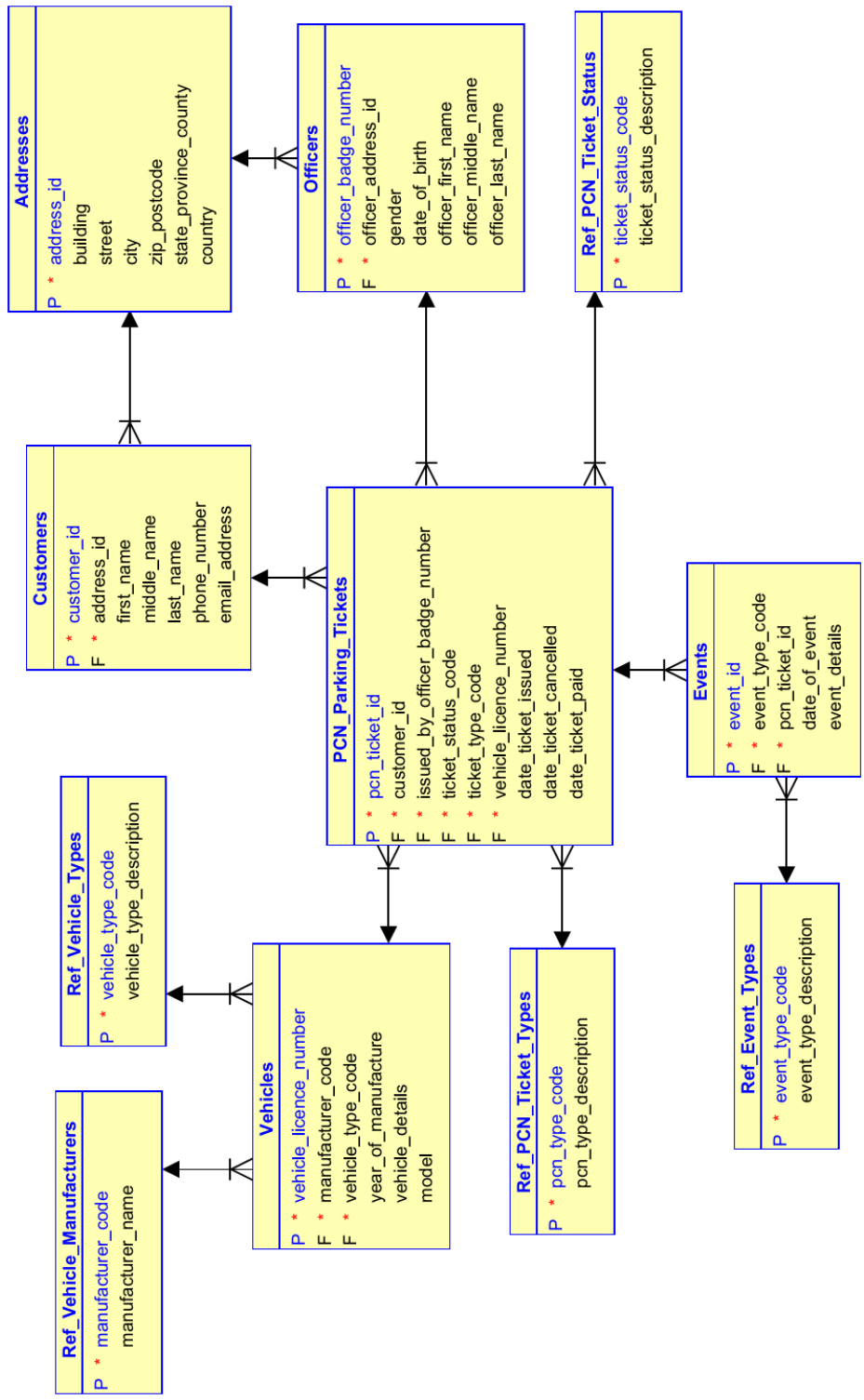


Figure 85. Parking ticket system.

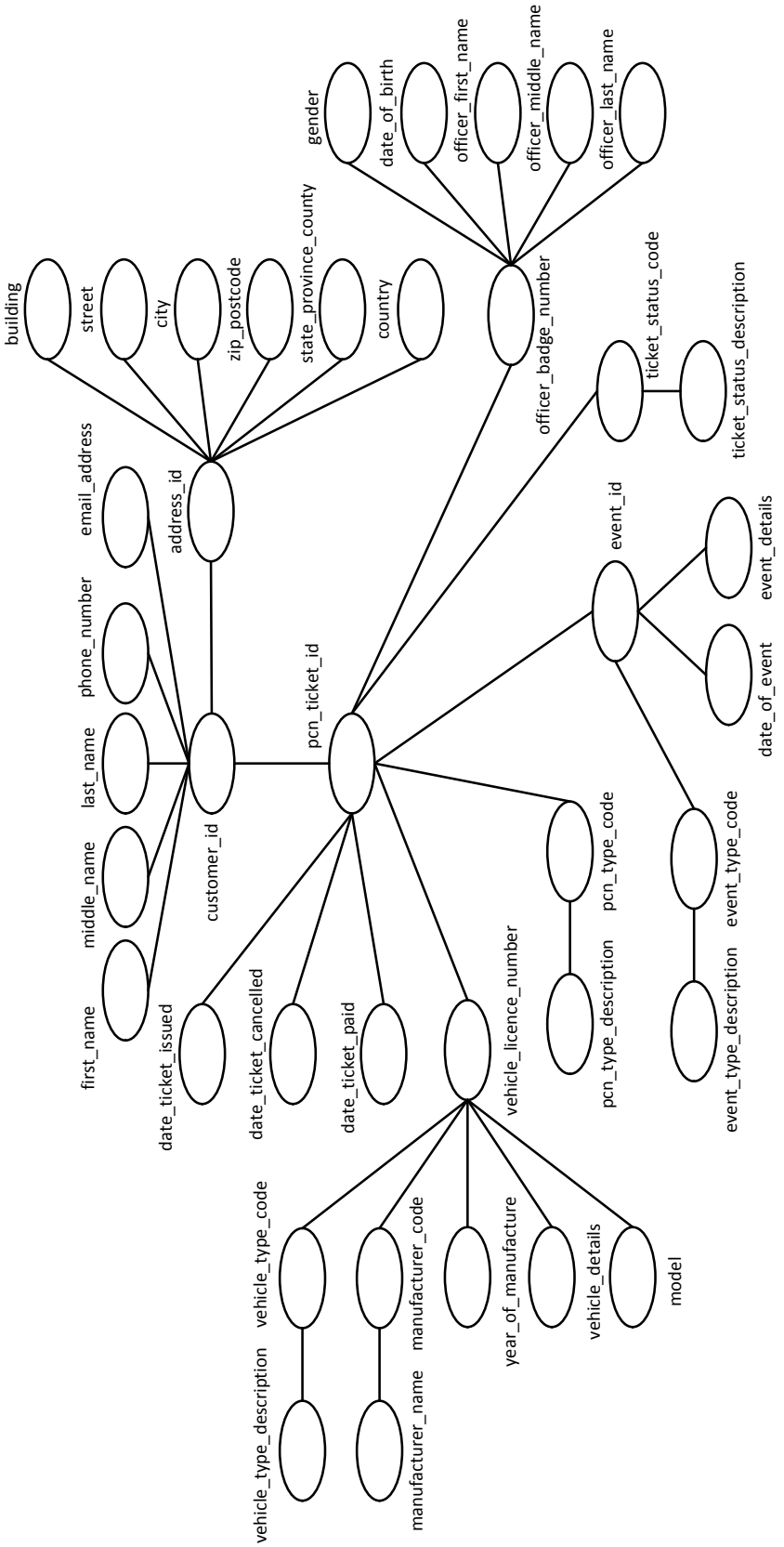


Figure 86. AdSchema for parking ticket system.

Table 50. Vertex properties of AdSchema.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	customer_id	x	7	x	x
2	date_of_birth		1		
3	date_of_event		1		
4	date_ticket_cancelled		1		
5	date_ticket_issued		1		
6	date_ticket_paid		1		
7	email_address		1		
8	event_details		1		
9	event_id	x	4	x	x
10	event_type_code	x	2	x	x
11	event_type_description		1		
12	first_name		1		
13	gender		1		
14	last_name		1		
15	manufacturer_code	x	2	x	x
16	manufacturer_name		1		
17	middle_name		1		
18	model		1		
19	officer_badge_number	x	7	x	x
20	officer_first_name		1		
21	officer_last_name		1		
22	officer_middle_name		1		
23	pcn_ticket_id	x	9	x	x
24	pcn_type_code	x	2	x	x
25	pcn_type_description		1		
26	phone_number		1		
27	ticket_status_code	x	2	x	x
28	ticket_status_description		1		
29	vehicle_details	x	1		
30	vehicle_licence_number	x	6	x	x
31	vehicle_type_code		2	x	x
32	vehicle_type_description		1		
33	year_of_manufacture		1		
34	address_id	x	8	x	x
35	building		1		
36	street		1		
37	city		1		

38	zip_postcode		1		
39	state_province_county		1		
40	country		1		

Case 10. Customer-order system (Williams 2014b).

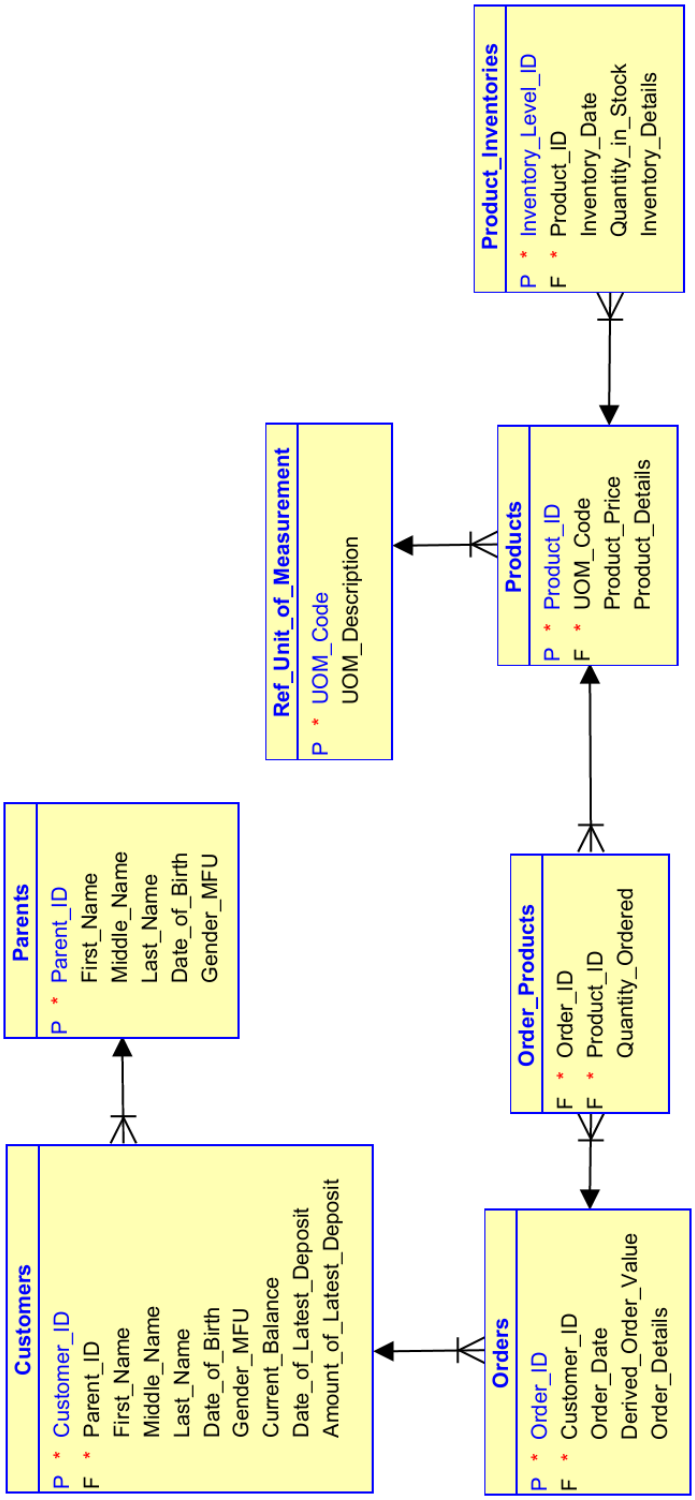


Figure 87. Datamodel for customer-order system.

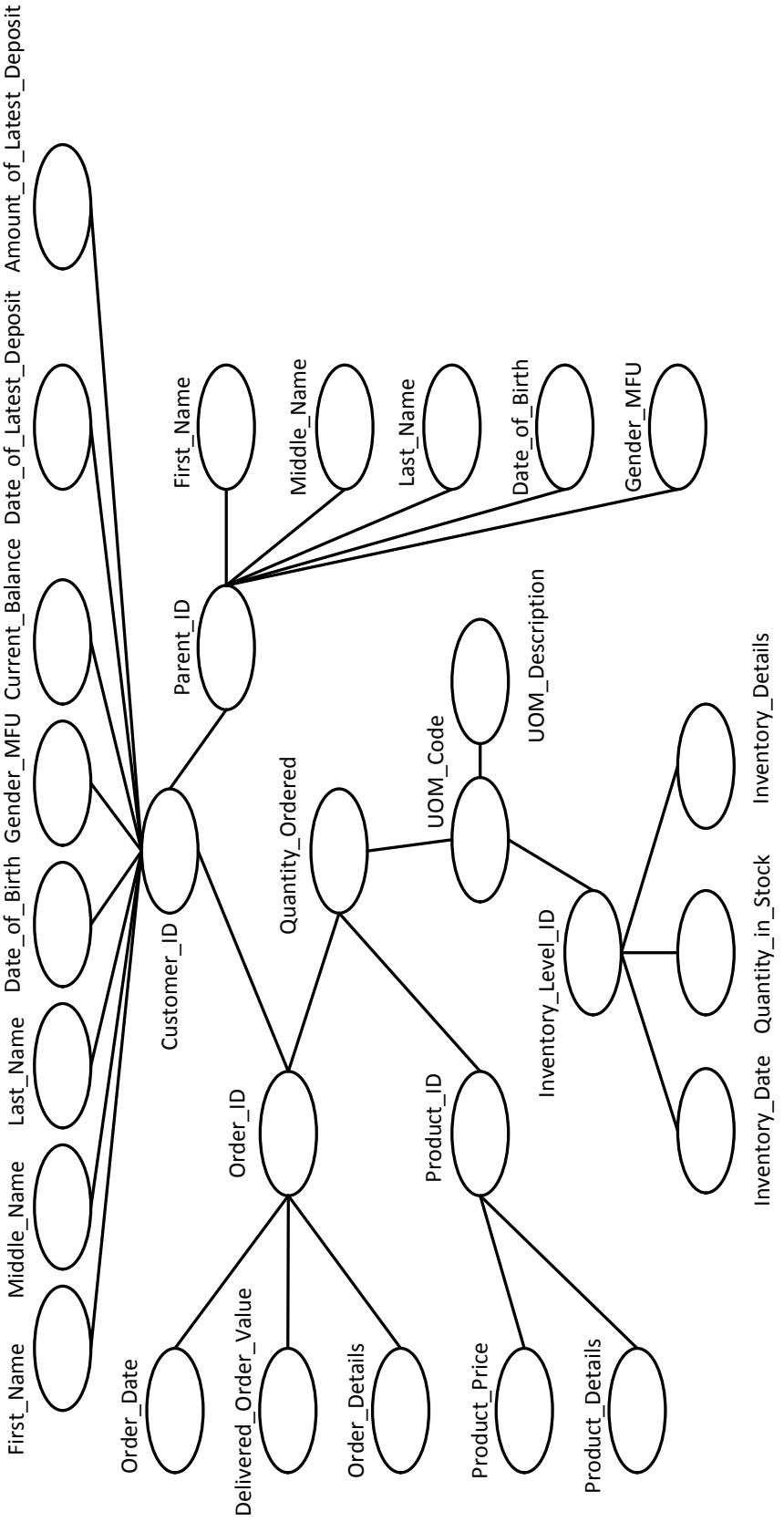


Figure 88. AdSchema for customer-order system.

Table 51. Vertex properties.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	Amount_of_Latest_Deposit		1		
2	Current_Balance		1		
3	Customer_ID	x	10	x	x
4	Date_of_Birth		1		
5	Date_of_Latest_Deposit		1		
6	Derived_Order_Value		1		
7	First_Name		1		
8	Gender_MFU		1		
9	Inventory_Date		1		
10	Inventory_Details		1		
11	Inventory_Level_ID	x	4	x	x
12	Last_Name		1		
13	Middle_Name		1		
14	Order_Date		1		
15	Order_Details		1		
16	Order_ID	x	5	x	x
17	Product_Details		1		
18	Product_ID	x	5	x	x
19	Product_Price		1		
20	Quantity_Ordered		2		
21	Quantity_in_Stock		1		
22	UOM_Code	x	2	x	x
23	UOM_Description		1		
24	Parent_ID	x	6	x	x
25	First_Name		1		
26	Middle_Name		1		
27	Last_Name		1		
28	Date_of_Birth		1		
29	Gender_MFU		1		

Case 11. Father of All Data Models (Williams 2009b).

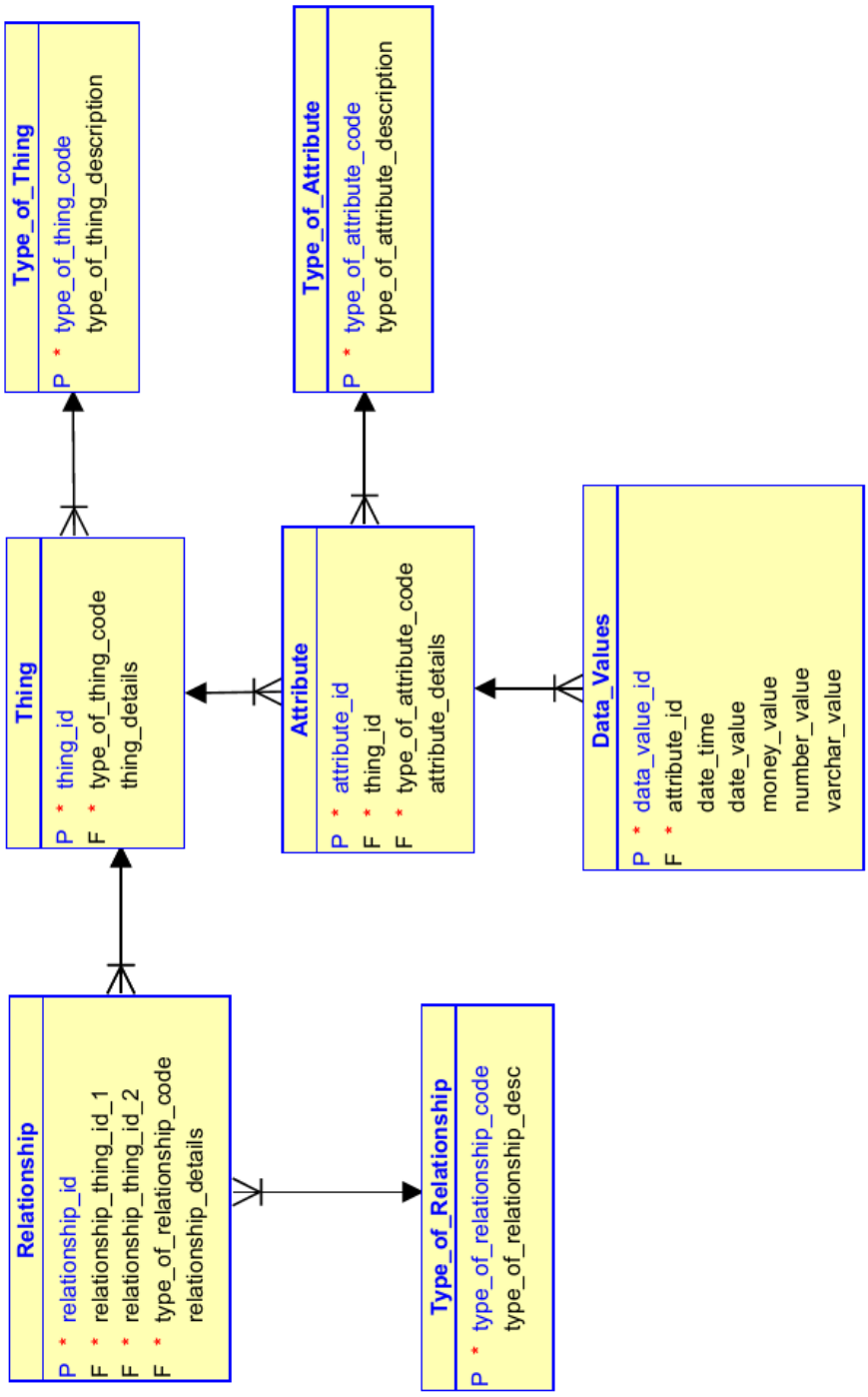


Figure 89. Father of all data models.

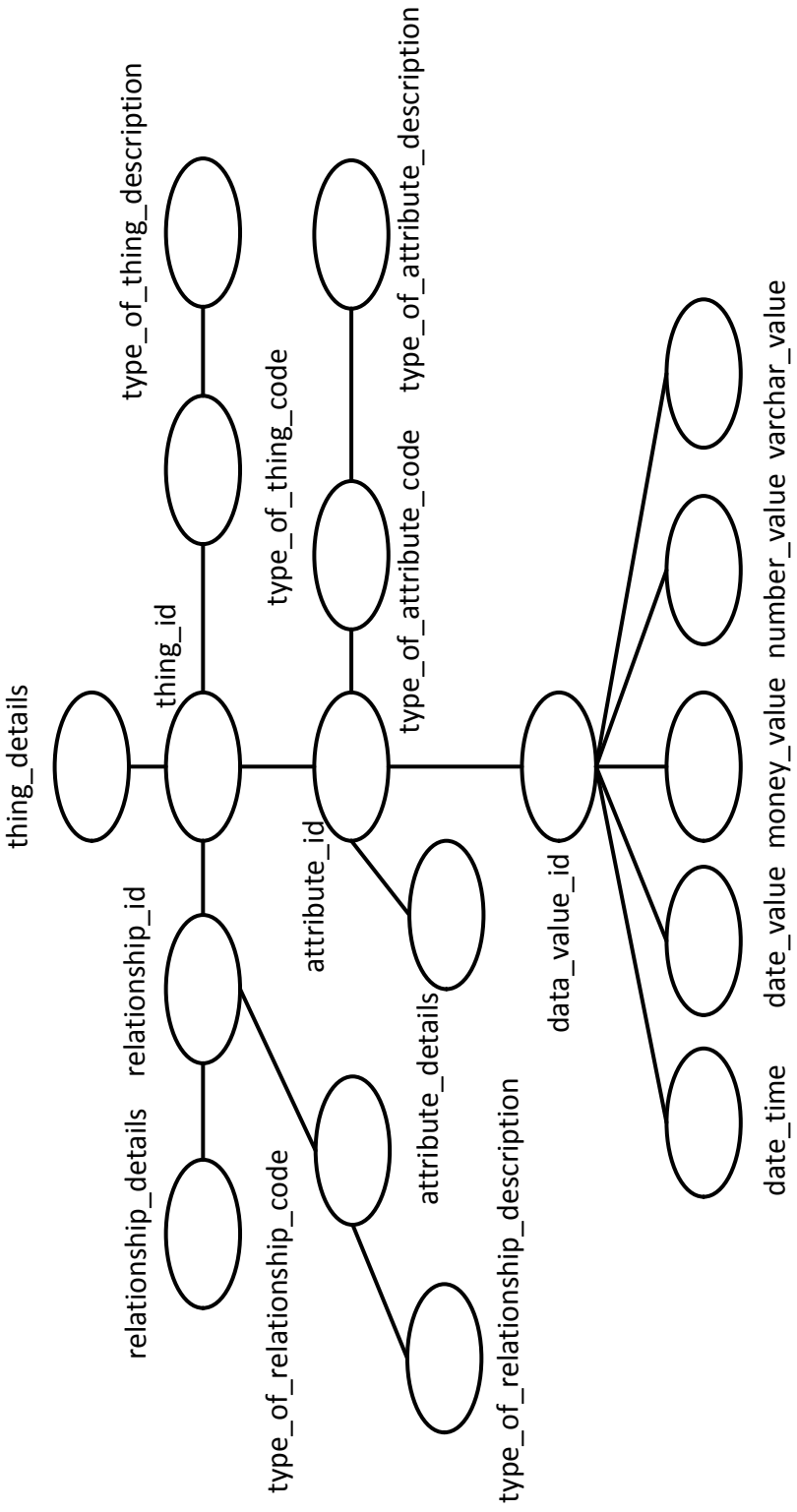


Figure 90. AdSchema for the data model.

Table 52. Vertex properties.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	attribute_details		1		
2	attribute_id	x	4	x	x
3	data_value_id	x	6	x	x
4	date_time		1		
5	date_value		1		
6	money_value		1		
7	number_value		1		
8	relationship_details		1		
9	relationship_id	x	3	x	x
10	thing_details		1		
11	thing_id	x	4	x	x
12	type_of_attribute_code	x	2	x	x
13	type_of_attribute_description		1		
14	type_of_thing_code	x	2	x	x
15	type_of_thing_description		1		
16	varchar_value		1		
17	type_of_relationship_code	x	2	x	x
18	type_of_relationship_desc		1		

Case 12. Sales force (Williams 2004d).

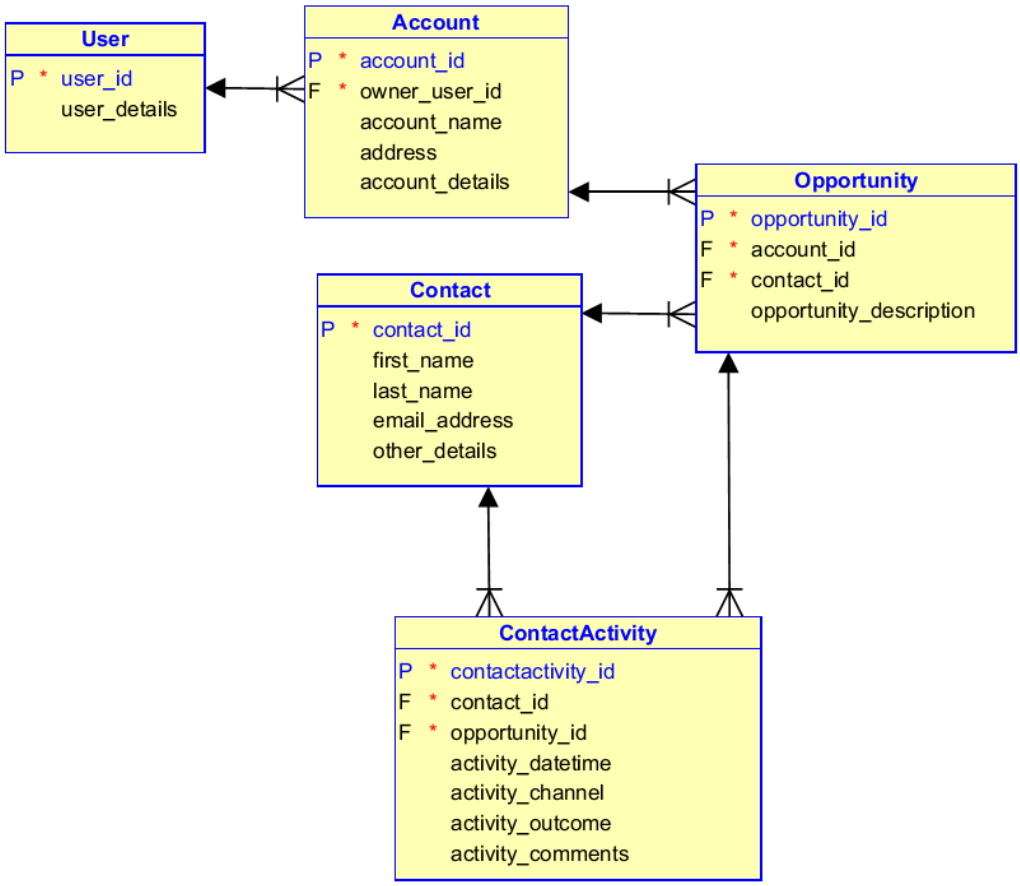


Figure 91. Data Model for SalesForce.com.

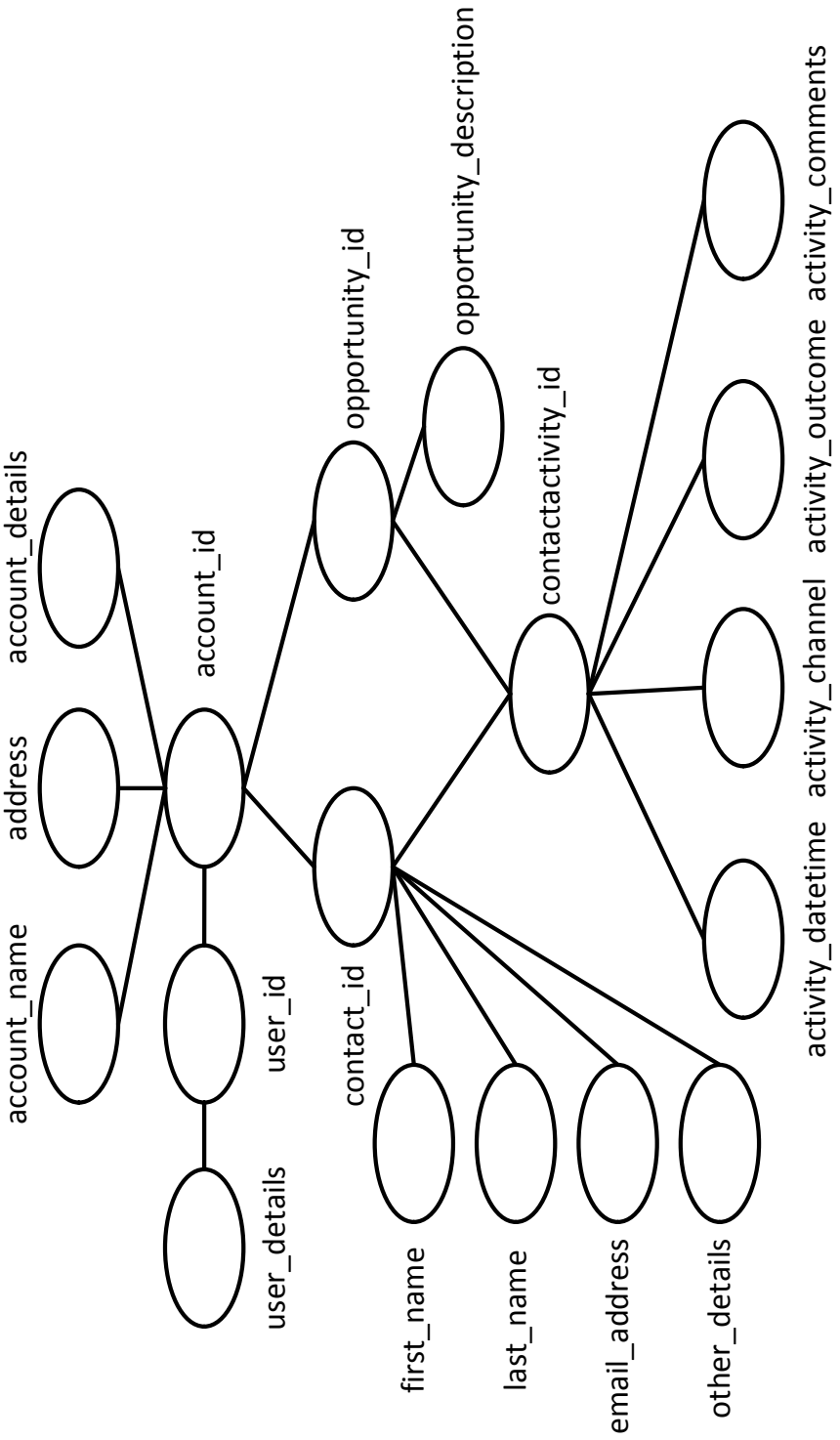


Figure 92. AdSchemafor the data model.

Table 53. Vertex properties.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	account_details		1		
2	account_id	x	6	x	x
3	account_name		1		
4	address		1		
5	contact_id	x	6	x	x
6	email_address		1		
7	first_name		1		
8	last_name		1		
9	opportunity_description		1		
10	opportunity_id	x	3	x	x
11	other_details		1		
12	user_details		1		
13	user_id	x	2	x	x
14	contactactivity_id	x	6	x	x
15	activity_datetime		1		
16	activity_channel		1		
17	activity_outcome		1		
18	activity_comments		1		

Case 13. Simplified customer order system

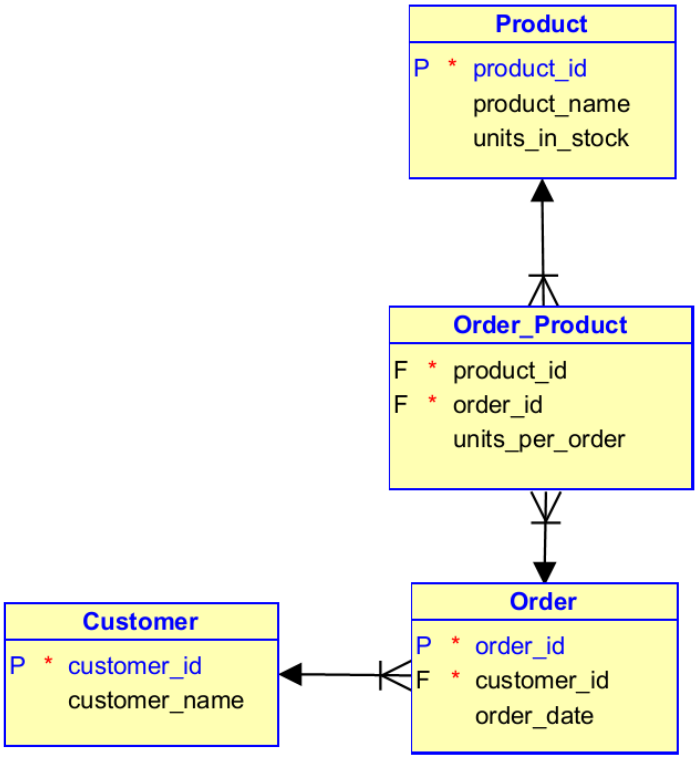


Figure 93. Database schema diagram for customer order system.

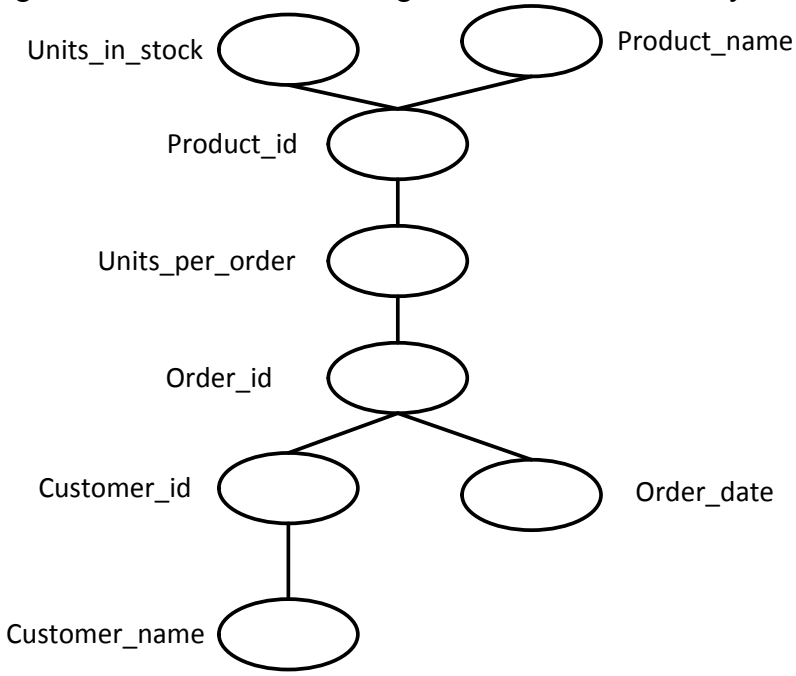


Figure 94. AdSchema for customer order system.

Table 54. Vertex properties in AdSchema graph.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	customer_id	x	2	x	x
2	customer_name		1		
3	order_date		1		
4	order_id	x	3	x	x
5	product_id	x	3	x	x
6	product_name		1		
7	units_in_stock		1		
8	units_per_order		2		

Case 14. Simplified customer order system

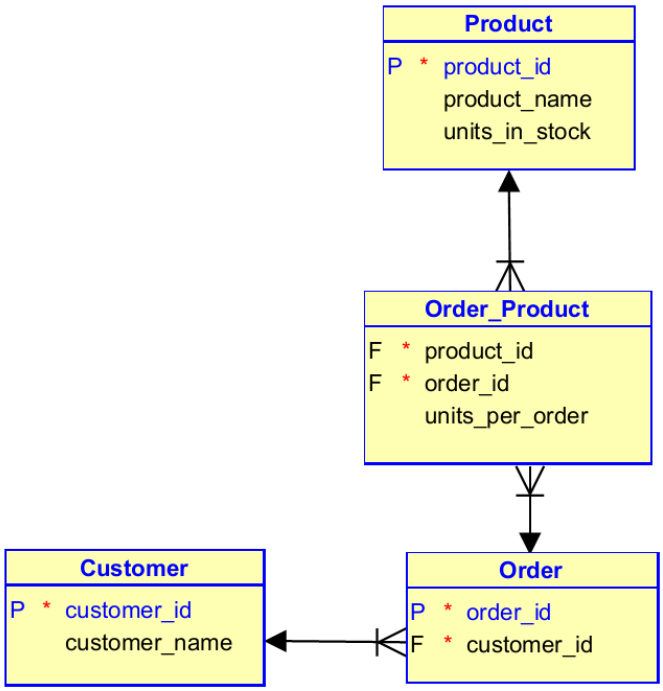


Figure 95. Database schema diagram for customer order system.

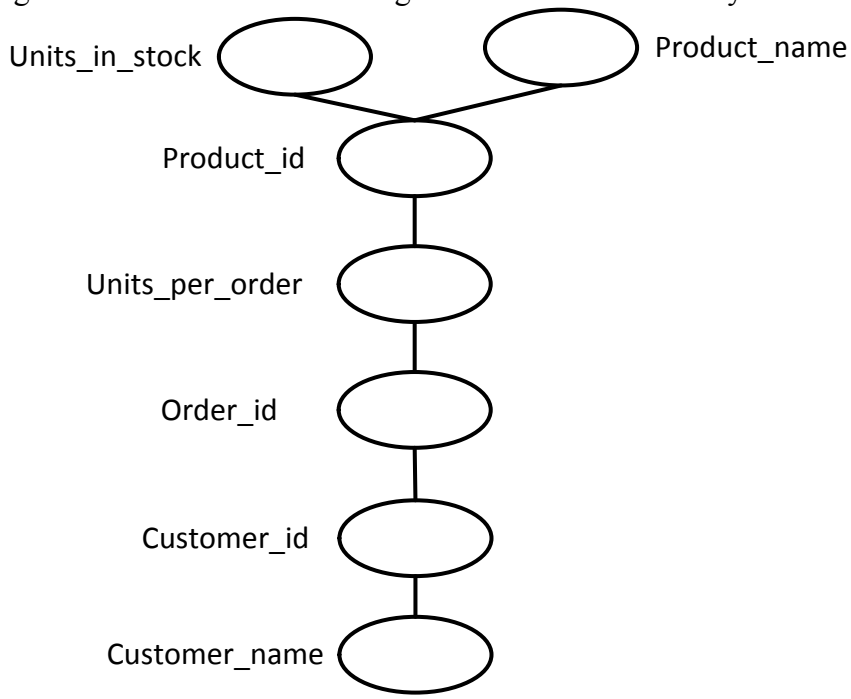


Figure 96. AdSchema for customer order system.

Table 55. Properties of attribute nodes in AdSchema.

			AdSchema properties		
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	customer_id	x	2	x	x
2	customer_name		1		
3	order_id	x	2		
4	product_id	x	3	x	x
5	product_name		1		
6	units_in_stock		1		
7	units_per_order		2		

Case 15. Simplified customer order system

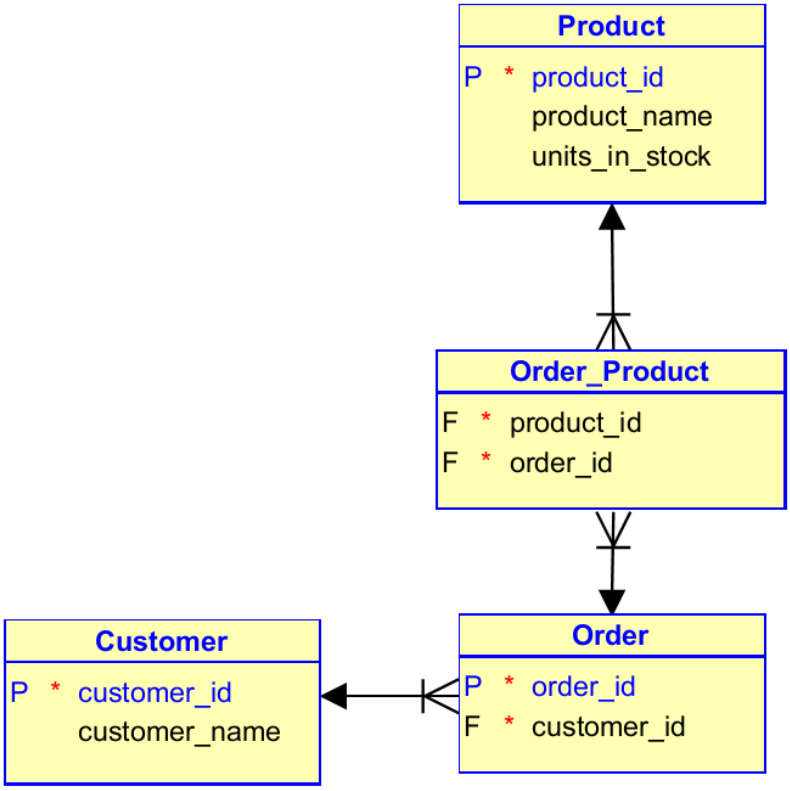


Figure 97. Database schema diagram for customer order system.

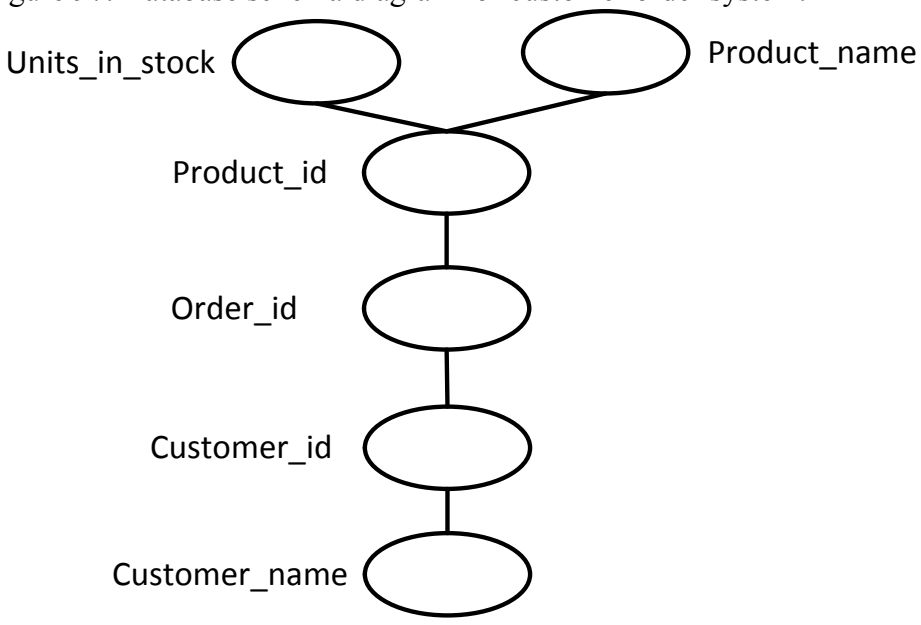


Figure 98. AdSchema for customer order system.

Table 56. Properties of attribute nodes in AdSchema.

		AdSchema properties			
Node index	Node name	Primary key in database	Degree	Has leaves	Member of dominating set
1	customer_id	x	2	x	x
2	customer_name		1		
3	order_id	x	2		
4	product_id	x	3	x	x
5	product_name		1		
6	units_in_stock		1		

REFERENCES

Williams, Barry (2001a). Accommodation Longterm - eg Employees. Available from the internet <URL:http://www.databaseanswers.org/data_models/ long-term_accommodation/index.htm>.

Williams, Barry (2001b). Physical Data model for Accidents. Available from the internet <URL: http://www.databaseanswers.org/data_models/accidents_at_work/ accidents_physical.htm>.

Williams, Barry (2004a). Data model for Afghanistan Rainfall. Available from the internet <URL: http://www.databaseanswers.org/data_models/ afghanistan_rainfall/index.htm>.

Williams, Barry (2004b). School Management System – A Physical Data Model. Available from the internet <URL:http://www.databaseanswers.org/data_models/ school_management_ systems/school_management_system_physical.htm>.

Williams, Barry (2004c). Apartment Rentals Data Model. Available from the internet <URL: http://www.databaseanswers.org/data_models/apartment_rentals/ index.htm>.

Williams, Barry (2004d). SalesForce.com – Four Major Entities. Available from the internet <URL: http://www.databaseanswers.org/data_models/ salesforce_dotcom/index.htm>.

Williams, Barry (2009a). Data model for Parking Tickets. Available from the internet <URL: http://www.databaseanswers.org/data_models/parking_tickets/ index.htm>.

Williams, Barry (2009b). The Father of All Data Models. Available from the internet <URL: http://www.databaseanswers.org/data_models/ father_of_all_models/index.htm>.

Williams, Barry (2011). Data model for Customer Deliveries. Available from the internet <URL:http:// www.databaseanswers.org/data_models/ customer_deliveries/index.htm>.

Williams, Barry (2012a). Data Model for a Patient Monitoring System. Available from the internet <URL:http://www.databaseanswers.org/data_models/ patient_monitoring_system/index.htm>.

Williams, Barry (2014a). Data Model for Occupational Health. Available from the internet <URL: http://www.databaseanswers.org/data_models/ occupational_health/index.htm>.

Williams, Barry (2014b). Customer, Inventory and POS. Available from the internet <URL: http://www.databaseanswers.org/data_models/ customers_inventory_and_pos/ index.htm>.