FLEXIBLE DATABASE DESIGN FOR ENTERPRISE RESOURCE PLANNING (ERP) APPLICATIONS

A MASTER'S THESIS

in

Computer Engineering

Atılım University

by

KADİR KANSU ÖZTÜRK

SEPTEMBER 2006

FLEXIBLE DATABASE DESIGN FOR ENTERPRISE RESOURCE PLANNING (ERP) APPLICATIONS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

KADİR KANSU ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

SEPTEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Selçuk Soyupak
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İbrahim Akman
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Nergiz Çağıltay
Supervisor

Examining Committee Members

Prof. Dr. İbrahim Akman _____

Asst. Prof. Dr. Murat Erten _____

Asst. Prof. Dr. Nergiz Çağıltay _____

Asst. Prof. Dr. Çiğdem Turhan _____

Instructor Gül Tokdemir _____

ABSTRACT

ENTERPRISE RESOURCE PLANNING (ERP) APPLICATIONS

DATABASE DESIGN CONSIDERING FLEXIBILITY

Öztürk, Kadir Kansu

M.S., Computer Engineering Department

Supervisor: Asst. Prof. Dr. Nergiz Ercil Çağıltay

September 2006, 77 pages

Enterprise Resource Planning (ERP) software presents a frame for organizations to better utilize their process and business. Implementation of ERP software is the process of adaptation of the frame to the organization. Therefore, implementation process should be good managed in order to achieve the targeted results of ERP philosophy. To facilitate these implementation projects, ERP software should be flexible enough to meet extra data requirements of implementation domain. In this thesis, a two-phased case study is constructed to denote implementation problems arisen from inflexible design. Besides, this thesis proposes database design alternatives in order to have a flexible structure for ERP software. The implementation problems determined in the case study is also evaluated according to offered database design alternatives. It is observed that the generic database design steps should be extended to evaluate flexibility requirements.

Keywords: Enterprise Resource Planning (RP), Database Design, Application Domain, Implementation Domain, Flexibility

ÖZ

KURUMSAL KAYNAK PLANLAMA (ERP) UYGULAMALARI ESNEK VERİTABANI TASARIMI

Öztürk, Kadir Kansu

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Asst. Prof. Dr. Nergiz Ercil Çağıltay

Eylül 2006, 77 sayfa

Kurumsal Kaynak Planlama (ERP) yazılımı, organizasyonlara işlerini ve süreçlerini daha iyi kullanmaları için bir çerçeve sunar. ERP yazılımının uyarlanması süreci, bu çerçevenin organizasyona bağdaştırılmasıdır. Bu sebeple, uyarlama süreci, ERP felsefesinin hedeflediği sonuçlara ulaşmak için iyi yönetilmelidir. Uyarlama projelerinin kolaylaştırmak için, ERP yazılımının, uyarlama alanında ortaya çıkacak ek veri ihtiyaçlarını karşılayabilecek esneklikte olması gerekmektedir. Bu tezde, esnek olmayan tasarımdan kaynaklanan uyarlama problemleri, iki safhalı bir örnek olay kullanılarak sunulmuştur. Ayrıca, ERP yazılımına esnek bir yapı kazandırabilmek için veritabanı tasarım alternatifleri sunulmuştur. Örnek olayda karşılaşılan uyarlama problemleride önerilen tasarım alternatifleriyle değerlendirilmiştir. Standart veritabanı tasarım adımlarının esneklik gereksinimlerini dikkate alacak şekilde genişletilmesi gerekliliği gözlenmiştir.

Keywords: Kurumsal Kaynak Planlama (ERP), Veritabanı Tasarımı, Uygulama Alanı, Uyarlama Alanı, Esneklik

To my family,
Thanks for their endless support

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Asst. Prof. Dr. Nergiz Ercil Çağıltay for her guidance, insight and encouragement throughout the study.

I should also express my appreciation to examination committee members Prof. Dr. İbrahim Akman, Asst. Prof. Dr. Murat Erten, Asst. Prof. Dr. Çiğdem Turhan and Instructor Gül Tokdemir for their valuable suggestions and comments.

I would like to express my thanks to my manager in Roketsan A.Ş., Mr. Akay Kerim İnce and other colleagues for their assistance, encouragement and all members of my family for their patience, sympathy and support during this study.

TABLE OF CONTENTS

ABSTRACT iii
ÖZ iv
ACKNOWLEDGEMENTS vi
CHAPTER 1
1 INTRODUCTION 1
2 LITERATURE REVIEW 4
2.1 CONCEPT OF ENTERPRISE RESOURCE PLANNING (ERP) 4
2.1.1. Definition 4
2.1.2. History and Future Expectations 4
2.1.3. Importance and Difference of ERP Systems 8
2.1.4. The problem of ERP systems 9
2.1.5. Role of Database Design in ERP packages 11
2.2. DATABASE DESIGN STEPS 12
2.2.1. Classical Approach to Database Design 12
2.2.2. Different Approaches to Database Design 12
2.2.2.1 Evaluation of Flexibility in Generic Design Process 13
2.2.2.2 Adding a New Step to Generic Design Process 15
3 RESEARCH METHODOLOGY 17
3.1 APPLICATION DOMAIN 18
3.2. IMPLEMENTATION DOMAIN 18
3.2.1. Budget Application for Defense Industry: Case Study 1 19
3.2.2. Budget Application for Construction Industry: Case Study 2 19
3.3. ANALYSIS OF THE PROBLEMS 20
3.4 NEW DESIGN OFFER(S) 20
4 CASE STUDIES 21
4.1 BUDGET 21
4.2 BUDGET APPLICATION FOR DEFENSE INDUSTRY 22
4.2.1 Requirement Collection and Analysis 22
4.2.2 Summary of Requirements 24
4.2.3 Conceptual Design 25
4.2.4 Logical Design 28
4.2.5. Physical Design 29
4.3 BUDGET APPLICATION FOR CONSTRUCTION INDUSTRY 30
4.3.1 Requirement Collection and Analysis 30
4.4. DETERMINING IMPLEMENTATION PROBLEMS 32
5 DESIGN ALTERNATIVES 35

5.1 EXISTING DESIGN ALTERNATIVE - EXTRA ATTRIBUTES 36
5.1.1. Flexible Design of Budget System and Solution of the Problems of Implementation 43
5.2. PROPOSED DESIGN ALTERNATIVE - NEW TABLE FOR CUSTOMIZATION 46
5.2.1 Customization Tables 46
5.2.1. Flexible Design of Budget System and Solution of the Problems of Implementation 52
6 CONCLUSION.....55
REFERENCES 57
APPENDIX.....59

LIST OF TABLES

TABLES

Table 2.1 ERP Vendors Market Share [13] 8
Table 2.2 ERP System Selection Criteria 10
Table 4.1 Implementation Summary 33
Table 5.1 Expense Lines without implementation requirements....43
Table 5.2 Introduction of the empty attribute to the system. 44
Table 5.3 Value Set records defined for the custom attribute....44
Table 5.4 Value Set Member records. 45
Table 5.5 Expense Lines after the implementation requirements met. 45
Table 5.6 Expense Lines without implementation requirements met. 52
Table 5.7 Expense Lines without implementation requirements met. 52
Table 5.8 Introduction of the new attribute to the system. 53
Table 5.9 Value Set records defined for the custom attribute....53
Table 5.10 Value Set Member records. 54

LIST OF FIGURES

FIGURES

Figure 2.1 Security Layer in ERP Systems [12] 7
Figure 2.2 Database life-cycle including modularization [5] 14
Figure 2.3 Database life-cycle including Implementation Design Step [6] 16
Figure 4.1 Conceptual Design of the Budget System 27
Figure 4.2 Logical Design of the Budget System 28
Figure 5.1 Logical Design with Empty Attributes 37
Figure 5.2 Extra Attribute Management Tables and Relations 40
Figure 5.3 Design Alternative 1 with Management Schema 41
Figure 5.4 EXPENSE_LINES with its custom table 47
Figure 5.5 Design Alternative 2 – System with Custom Tables 48
Figure 5.6 – Proposed Design Process 49
Figure 5.7 – Design Alternative 2 with Management Schema 51

LIST OF ABBREVIATIONS

ERP	-	Enterprise Resource Planning
MRP	-	Material Requirements Planning
MRPII	-	Manufacturing Resource Planning
MIS	-	Management Information System
CRM	-	Customer Relationship Management
SCM	-	Supply Chain Management
KM	-	Knowledge Management
ERPII	-	New Generation ERP Systems
CC	-	Cost Center
UOM	-	Unit of Measure

CHAPTER 1

INTRODUCTION

In today's competitive market, organizations have to adapt themselves into the continuously changing and evolving conditions in order to survive and develop. Therefore, during the adaptation process the organization's decisions are critical. During that process, in order to take appropriate decisions, the organization should have a good understanding of its own system dynamics, its current place and state in the market as well as ever changing dynamics which impact and affect the world, the country, the sector and finally the organization itself. Management Information System (MIS) implementation throughout the organization would help them to better cope with those issues by the help of business intelligence and decision support systems. Within this scope, in order to implement the MIS, it is necessary to establish a common information system infrastructure and to integrate the corporation business workflows to this system. The software providing this infrastructure for MIS is called as Enterprise Resource Planning (ERP).

Main aims of an ERP application can be summarized as:

1. To expedite, integrate and maintain the standardization of business workflows
2. To provide the coordination, cooperation and integration among the functional units
3. To interfere the repetitive data entry within the organization

The organizations which implement an ERP system also accept main aims stated above. ERP has a distinction from other software systems, because modules (application

domains) in ERP packages are free from the implemented organizations. Thus, customizations are inevitable through the implementation. For example, an important step in implementation is to compare business processes embedded in the ERP package with organization's business processes. In order to manage this change, by bringing business processes to the same platform, organization should become aware of the organizational change and needs. Such a change leads customization. The short and not fitting aspects in the ERP package may increase risks in the success of implementation. Accordingly, risk management is an important factor for the success of such a project. Especially for the ERP package, risk management concentrates on the solutions and business processes that are available in the package but disregarded by the organizations. In that context, the stated items below are vital for an ERP package in order to manage the risk:

- Flexibility (allowing additional and new data entry not included in the package)
- Extendibility (localization programs not included in the standard package)

In order to increase the flexibility and extendibility of an ERP package, one should increase the flexibility available at the interface design and the conceptual design of the system. From the software engineering point of view, an ERP package should consist of two main components:

- Programs units that constitute the business intelligence
- Database that holds the setup and transaction information specific to the organization

Since the conceptual design of a system is the state of an information system where early implementation decisions are made, it is important to meet the requirements of a flexible implementation in that stage. Within this context, the database design which the ERP package is built on, should consider the employment of the package modules (application domains) considering requirements of different implementation domains.

As there is a need for building a flexible ERP application, thus flexible database design, it is clear that, we need new ideas for constructing database design in a flexible manner. The focus of this study is discussing the implementation problems that arise from non-flexible database design and offering techniques to solve them.

This study is organized as follows: Chapter 2 presents an overview of ERP concepts and historical details. Also importance and effect of database design for ERP systems and design considerations found in the literature are explained. In Chapter 3, methodology of research has been described and discussed. In Chapter 4, a two phase case study has been constructed to determine implementation problems. First, a database design for an ERP module (budget system) has been made according to the requirement analysis of a specific industry. Then, the designed system has been implemented to a different industry. The requirements are compared with the designed system and the implementation problems are listed. In Chapter 5, new design alternatives are offered to solve the problems while addressing the implementation flexibility.

CHAPTER 2

LITERATURE REVIEW

2.1 CONCEPT OF ENTERPRISE RESOURCE PLANNING (ERP)

2.1.1. Definition

GartnerGroup [2] described ERP systems as integrated modules for materials management, finance, accounting, sales and distribution, human resources and other business functions on the same architecture domain linking the enterprise to customers and suppliers. According to them, ERP systems are designed to streamline data flows within and between organizations, providing management and other organizational members with direct access to real-time operating information [2]. According to Satzinger-Jackson-Burd, ERP is a process in which an organization commits to using an integrated set of software packages for key information systems to improve the effectiveness and efficiency of the enterprise [7].

An ERP application is a software system that manages successfully the relations between various functional departments, the infrastructure of business intelligence and different reporting needs as well as the backbone of business processes and workflows in a corporation.

2.1.2. History and Future Expectations

Enterprise Resource Planning (ERP) is the last state of a system that has been released in 1960's, which have been called Material Requirements Planning (MRP). The

main aim of MRP systems was calculating the material requirements according to time stamps with the help of computerized systems. MRP was developed to calculate more efficiently the materials needed [15].

The improvements in the computer systems and operational needs that requires co-operation and co-ordination of various departments (quality control, finance, sales, production planning etc.) in a firm, have moved the MRP system one step further; Manufacturing Resource Planning (MRPII). MRPII was enough for firms until 1990's, the decade that globalism started to be effective on business processes. Firms became to sell and buy goods, services from everywhere in the world without considering the limits of distance. At that point, the MRPII functions had to be modified and re-designed for a firm that has different locations and branches around the world. Therefore, "Enterprise" concept has come to the scene. The planning function has to be deployed at the enterprise level. New software application then called Enterprise Resource Planning (ERP).

Early generation of ERP systems entirely focused on infrastructure renewal and creating efficiencies in back office operations around key functional areas such as finance and production management. These systems, while useful, did not address the mission critical issues such as supply chain management (SCM), customer relationship management (CRM), and knowledge management (KM). New generation systems have sought to address these short-comings in a variety of ways. Also, while early generation ERP systems were found mainly in large organizations, newer systems are now being targeted at small and medium enterprises as means of expanding the ERP market [2].

New concepts like e-business, e-government and e-commerce rely on ERP systems and force firms to implement ERP systems. Next step is the sharing of benefits gained from ERP or other special information systems, with trading partners. In this context, ERP systems will provide data exchange across firms to build supply chain systems and knowledge share. Therefore, a new link as ERPII will be added to the MRP-MRPII-ERP chain.

ERP systems should cover major business processes in an organization. Therefore, ERP software mainly consists of 7 parts [1]:

- Finance- Accounting

- Production Management
- Purchase Order Management
- Sales Management
- Inventory Management
- Quality Management
- Human Resources Management

The degree of relationship between these modules is derived from the implemented organization's requirements. For example, definition of an inventory item has been done in Inventory Management module. But its transaction about purchase and sales are have tracks on Purchase Order Management and Sales Managements modules. Also, these transactions have account balance changes in Accounting Management module.

There is a management module for ERP systems. Its main mission is to hold the features of modules, tables, columns and security data. Any of these objects should have been defined to this management schema in order to be recognized by the ERP system. As we mentioned ERP systems provide an authentication and authorization mechanism. This mechanism is in the management schema. This management schema covers all security needs of modules. It is similar to the metadata of a database system. Why ERP systems do not use database security mechanisms of database system that it is installed on? Because it is much simpler to install access protection measures on the level of the ERP itself as they must be integrated with other services in the ERP system [12].

In Figure 2.1 authentication and authorization of ERP systems are explained: in the most outer circle 'Security of ERP Layer', the user is logged on to the ERP software. Afterwards the second layer comes on the scene 'Application specific security', which determines the responsibility layer of the system. This layer assigns capability to the user about the chosen module. The third layer 'ERP Application Layer' authenticates users to execute reports and procedures with the help of above layers. At the last step, 'Integrity&Security Wrapper' layer, database security mechanisms and ERP security mechanisms are coordinated and combined together.

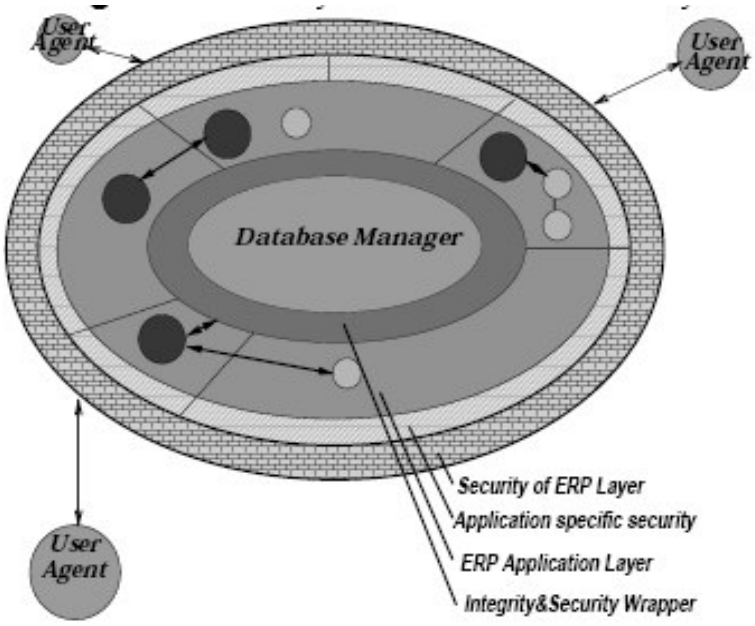


Figure 2.1 Security Layer in ERP Systems [12]

There are a lot of ERP software vendors in the market. The list of ERP vendors which has significant market share has listed in table 2.1. Table 2.1 shows us the leader of the ERP market is SAP. After the acquisition of PeopleSoft, Oracle has the second big share from the market. The sum of other ERP vendors' market share is smaller than the sum of SAP and Oracle market shares.

Table 2.1 ERP Vendors Market Share [13]

2004 Revenue Rank	Company Name	Revenue, 2003 ($M)	Revenue, 2004 ($M)	Revenue Forecast, 2005 ($M)	Revenue Share, 2003	Revenue Share, 2004	Revenue Share Forecast, 2005	Growth Rate, 2003–2004	Growth Rate Forecast, 2004–2005
1	SAP	7994	9372	10403	39%	40%	43%	17%	11%
2	PeopleSoft	2682	2880	0	13%	12%	0%	7%	-100%
3	Oracle*	2470	2465	4534	12%	10%	19%	-0%	84%
4	Sage Group	900	1243	1375	4%	5%	6%	38%	11%
5	Microsoft Business Solutions	683	775	891	3%	3%	4%	14%	15%
6	SSA Global	471	700	700	2%	3%	3%	49%	0%
7	Geac	431	445	445	2%	2%	2%	3%	0%
8	Intentia	361	388	407	2%	2%	2%	8%	5%
9	Infor Global Solutions	123	375	395	1%	2%	2%	205%	5%
10	Lawson	341	357	358	2%	2%	2%	5%	0%
Total (including other ERP vendors)		20711	23649	24288	100%	100%	100%	14%	3%

* Oracle acquired PeopleSoft on December 28, 2004.

Source: AMR Research, 2005

2.1.3. Importance and Difference of ERP Systems

Coordination between functional departments, defined and standardized business process and controlled data insertion enables firms to establish decision support systems (DSS) and take snapshot of firm at anytime with the help of ERP systems. These gains help firms to react effectively and timely in competitive environments. Only way of surviving in today's competitive market is, understanding of your own situation and determining new direction(s) according to the environment. Therefore, ERP system has to serve as mirror of the firms and should help mid-level and high level managers in their decision make process. ERP has always an important impact on the organization and on management practices, due in particular to the fact that it imposes the same treatment method whatever the organization's activity, as well as a standardized and rigorous implementation procedure [13].

ERP is mainly a packaged software. Packaged software is a system that is already built and can be purchased as a package. It is based on an underlying integrated database

that stores master and transactional data in a consistent way and with controlled redundancy [13]. The advantage of ERP systems is that, a new system can usually be obtained at a much lower cost and risk than in-house development. The cost is lower because 60 to 80 percent of the application already exists in the base system. Risk is lower because the base system is usually well developed and tested. Other organizations are already using it, so it has a track record of success. The disadvantage is that, ERP system may not do exactly what the organization needs even after the system has been customized. Frequently, there is a gap between the exact needs of the organization and the functionality provided. The company then must modify its internal processes and train its users to conform to the solution provided [7].

2.1.4. The problem of ERP systems

ERP selection and implementation is crucial for firms and flexibility is a critical success factor in these selection and implementation processes. ERP systems are huge and complex systems and warrant careful planning and execution to ensure their successful implementation [16]. There are significant numbers of failed ERP implementations which did not meet the expected results. Consequently, it has been estimated that at least 90% of ERP implementations end up late or over budget and almost half fail to achieve the desired results [16].

One reason for the failure of ERP implementation is the inflexibility of the software. Table 2.2 lists 29 of inquired ERP selection criteria have been identified by Bernroider and Koch [8]. This is a result of an empirical study concerning differences in characteristics of the ERP system selection process between small or medium and large sized organizations. As shown in this table, criteria “Adaptability and Flexibility of Software” is rated over 90% “Important” and “Very Important” by senior management of the IT department from 138 Austrian small/medium and large size organizations.

Table 2.2 ERP System Selection Criteria

	Very Important (percentage)	Important (percentage)	Rather unimportant (percentage)	lrrelevant (percentage)
Increased Transparency and Better Information Flow	65.8	30.8	1.7	1.7
Well Tried Software System	60.3	*36.2*	2.6	0.9
Good Support	56.0	40.5	3.4	0.0
Y2K Problem	54.2	22.9	14.4	8.5
Adaptability and Flexibility of Software	52.6	41.4	5.2	0.9
Shorter Cycle Time	52.1	39.3	7.7	0.9
Press Improvement	48.7	41.4	5.2	0.9
Currency Conversion (i.e. Euro)	47.0	29.1	13.7	10.3
Increased Organizational flexibility	46.2	39.3	11.1	3.4
Increased Customer Satisfaction	42.2	36.2	16.4	5.2
Internationality of Software	36.8	27.4	20.5	15.2
Other Strategic Considerations	36.2	35.3	22.4	6.0
Modular Architecture of Software	35.7	52.2	10.4	1.7
Higher Reliability	32.5	51.3	9.4	6.8
Market Position of Vendor	32.2	49.6	13.9	4.3
Implementation of Desired Business Processes	31.6	47.0	16.2	5.1
Short Implementation Time	31.0	52.6	12.5	4.3
Operating System Independency	28.4	37.1	28.4	6.0
Availability of Tools for Software-Adoption	27.7	45.5	21.4	5.4
Ergonomic Software	27.4	55.6	15.4	1.7
Availability of Special Solution or Branch of Badness	26.3	28.1	30.7	14.9
Improvement of Organizational Structure	22.0	45.8	29.7	2.5
Guidelines from a Controlling Company	21.9	19.3	14.9	43.9
Improved Innovation Capabilities	19.3	38.6	33.3	8.8
Increased Know-How	12.0	36.8	40.2	11.1
Customer and Supplier Needs	11.2	19.0	27.6	42.2
Better Application of Management-Style	10.4	40.0	40.9	8.7
Improved E-Commerce Support	5.1	23.1	43.6	28.2
Improved Internet Services	4.3	34.2	38.5	23.1

Below are the stated reasons that ERP applications fail or do not meet the required targets. These failures of implementations affect the happiness and motivation of the employees.

- Business processes used by firm that are not included in the ERP package
- Business processes that are included in the ERP package and used by the firm but extra data entrance are required
- Business processes that are included in the ERP package and used by the firm but firm insist on using its exact business process

These problems can be solved in 3 ways as stated below:

- Newly added program units (Extensions)
- Flexible design of ERP application in order to solve new data entrance requirements
- Acceptance of the know-how on the application domain and implement the entire ERP package business processes

The last one is the ideal situation, however it is not implementable, because ERP software cannot cover all aspects of business processes in all countries, sectors and firms. The first one is implemented if there is no workaround and it is inevitable, but its cost is too high. Also it is hard to manage and it creates problems in upgrade-migration phases. The second one, the research point of this thesis, gives us the solution in the package therefore, cost is lower and management is easy. Also there is no need to employ so many technical people and upgrade-migration is not problematic.

2.1.5. Role of Database Design in ERP packages

There are different sized firms in today's ERP applications software market. In order to be competitive in the market, ERP packages should be flexible enough to adopt requirements which are specific to implementation country, sector and firm. Our hypothesis is that, if we have had enough flexible database design and an interface

supports that design, without editing ERP applications source code, we could meet the extra requirements which appears during implementation and post implementation.

2.2. DATABASE DESIGN STEPS

2.2.1. Classical Approach to Database Design

A generic database design process according to Navathe, Elmasri [4] includes four phases, to be carried out in the following sequence:

- Requirements Collection and Analysis: consists of identifying and understanding the application's data and functional requirements.
- Conceptual Design: involves two parallel activities. The first consists of examining the data requirements resulting from the previous phase and, ignoring the details of implementation, producing the conceptual data schema. The second activity consists of examining the functional requirements defined in the previous phase and producing specifications for the defined transactions, regardless of the DBMS to be employed.
- Logical Design: consists of transforming the conceptual data schema built in the previous phase into a specific schema of a given DBMS called a logical data schema.
- Physical Design: consists of selecting specific storage and access structures for the DBMS to be used.

These steps are required and sufficient steps while designing generic database application software. But problem appears when flexibility consideration comes on the scene.

2.2.2. Different Approaches to Database Design

Adding a new step or evaluation of flexibility in a generic design step could be an alternative solution to our problem.

2.2.2.1 Evaluation of Flexibility in Generic Design Process

Adding new step(s) or evaluating needs in generic step of database design was mentioned in earlier studies for different aims. To evaluate the requirements of modularization, Ferreira and Busichia [5] have added two sub steps to the generic design process.

- Collection and Analysis of Modularization Requirements: aims to identify and understand the application's modularization requirements, defining the subsystems it is composed of and associating the transactions defined in the phase of Requirements Collection and Analysis to each one, according to its functions.
- Modularization Design: aims to modularize the database, associating the data to the subsystems that make up the application according to the transactions carried out by these subsystems, taking into consideration the need for information sharing and interoperability. Considering that, in order to be divided, the data schema has to have been built according to a data model attributing semantics to modelling; the Modularization Design must be made during the Conceptual Design, using the conceptual global schema to carry out partitioning. Because Modularization Design generates the conceptual data schemas of each module, the Logical Design phase must be applied to each of these conceptual schemas, generating logical data schemas for each module.

Figure 2.2 shows generic database design process with new sub-steps mentioned above. These sub-steps have changed the generic design process in order to ensure modularization requirements have been met. At Functional Requirements sub-step, the proposed design evaluates modularization requirements. Then, these requirements are considered in modularization design phase. Modularization design phase takes "Global Conceptual Design" as input. Than modularization requirements and "Global Conceptual Design" are evaluated together to setup conceptual design with modularization.

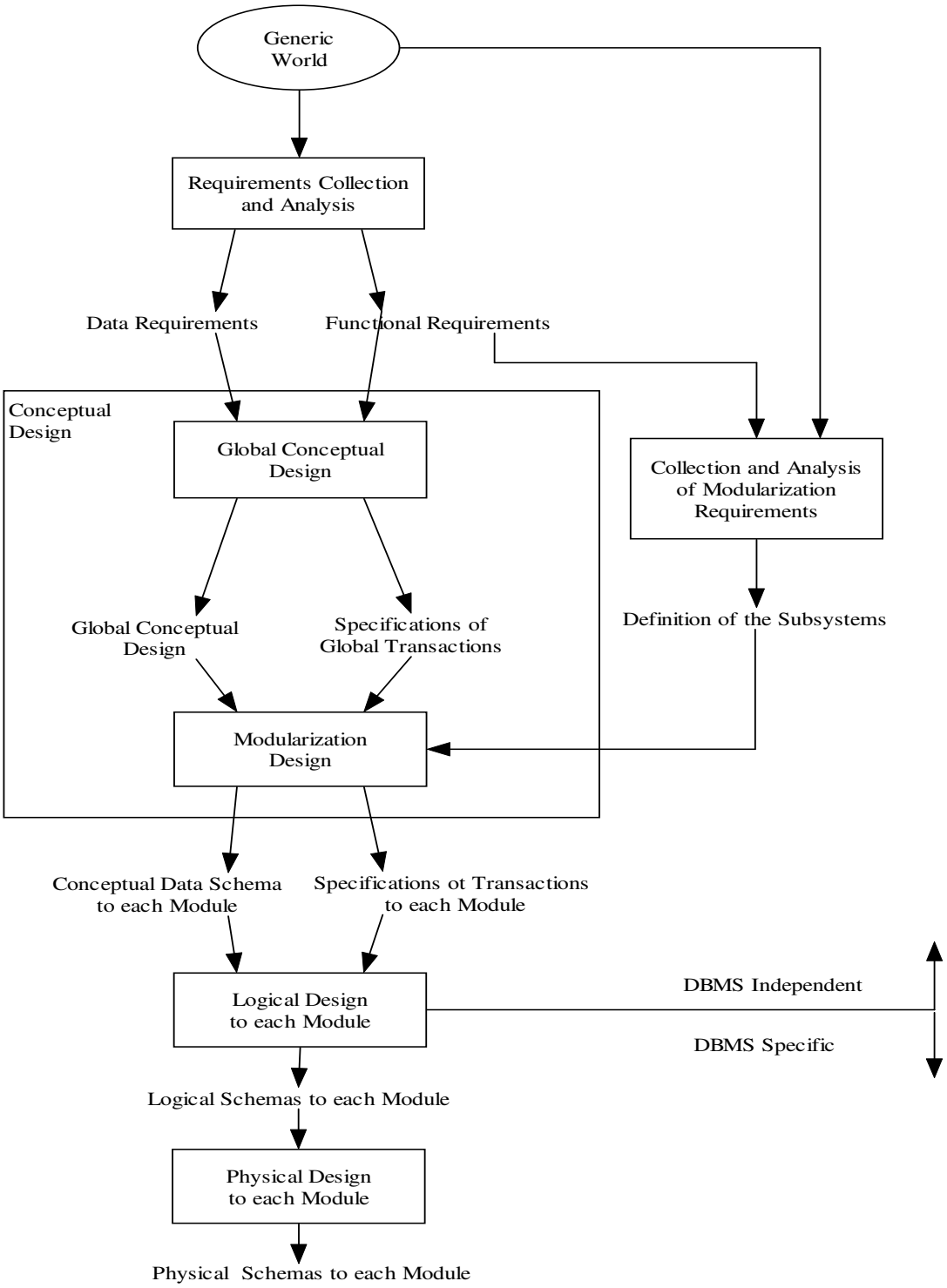


Figure 2.2 Database life-cycle including modularization [5]

2.2.2.2 Adding a New Step to Generic Design Process

Tim Martyn [6] was interested in database design process. His aim was adding a new step for obtaining a pragmatic version of the logical design. Thus, an extra step "implementation design" comes on the scene. According to him, implementation design techniques are very different from the design transformations applied during logical design, the techniques also differ from a physical design, because physical design does not consider internal, DBMS-specific facilities [6].

Figure-2.3 shows the generic database design process with a new step "Implementation Design" added. Like the three-step method, the process begins with conceptual design and moves to logical design. However, the logical model in logical design remains ideal. The ideal model becomes the input to implementation design, whose goal is to produce a pragmatic version of the ideal model; the implementation model. The implementation model then becomes the input to physical design. This process keeps all performance enhancements out of logical design and simplifies physical design [6].

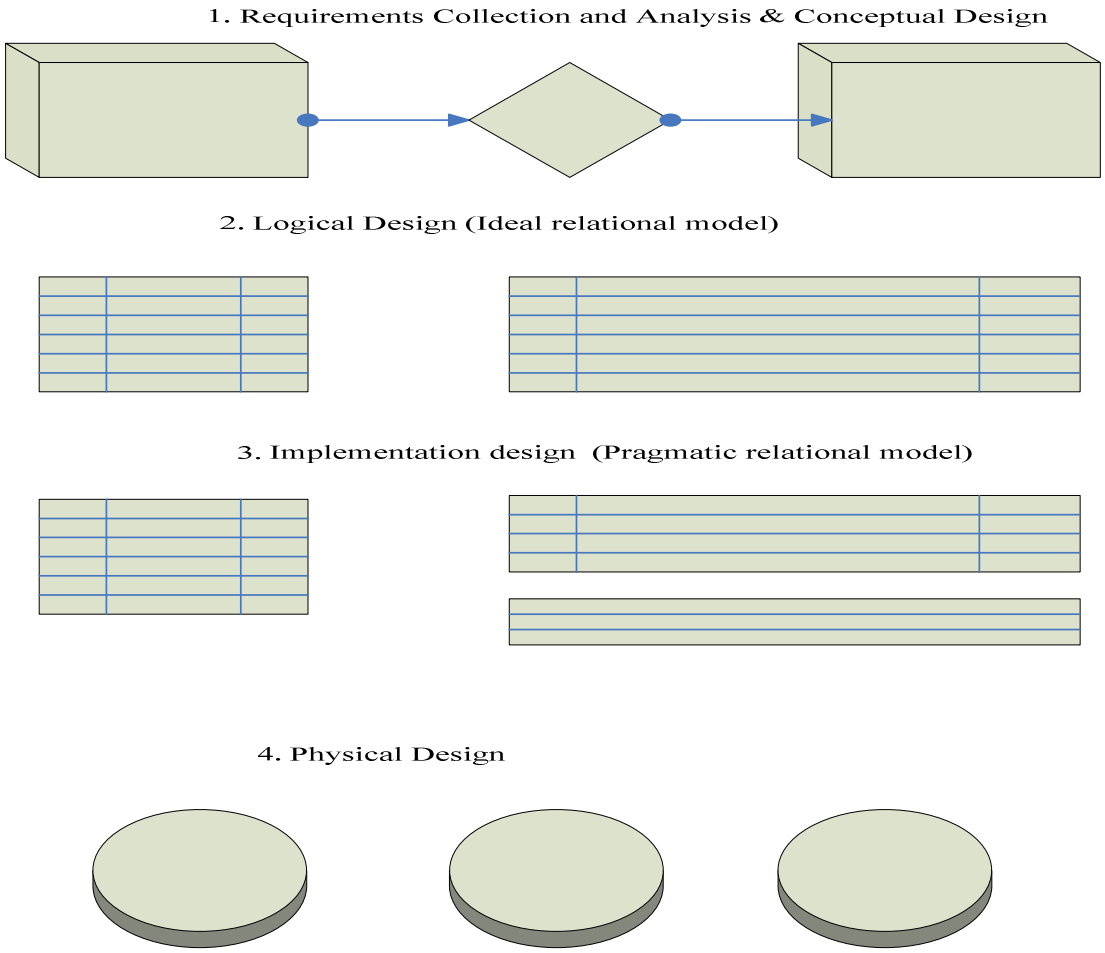


Figure 2.3 Database life-cycle including

Implementation Design Step [6]

As we understand from the literature, considering flexibility in ERP applications database design should be done in conceptual design phase. It is necessary to obtain requirements that ERP implementation step would need at first step of generic design. Then at conceptual design phase we have to develop new entities and attributes that provides flexibility to our package.

CHAPTER 3

RESEARCH METHODOLOGY

After the problem of this thesis is identified, we have to determine implementation problems that arise from inflexibility of database design. Thus, we have decided for a case study of implementation of a part (module) of an ERP system that would identify these problems. Therefore, we setup a two phased case study. Firstly, we have developed a real application for specific requirements of an organization and for a particular area. Then we implemented the developed system for the same area but for a different organization. Our aim was to capture more problems while we are implementing the application. We expected that, specific needs of an organization would not meet the requirements of a different organization at our second implementation. In this context, we can determine the implementation problems at second phase of our case study. Then we can offer a new design to prevent some of the implementation problems. Also our system would be a learning system, because when we extend domain knowledge of a chosen part of ERP system then our program would cover more business processes and business intelligence in itself.

We have used generic database design steps for defense industry budget application needs. Then requirements collection and analysis have been done for construction industry. According to these requirements, we have considered the design again. We listed the problems that could arise, if we would have used the first design.

After the determination of problems, we have considered the flexibility in generic database design process, especially in “conceptual design phase”.

Our method to identify the problems of implementation is to setup two case studies. Steps of our research methodology involve;

- Select an application domain
- Select two different implementation domains for each case
- Design of a database for requirements of first implementation domain (Case 1)
- Requirements collection of second implementation domain (Case 2)
- Analysis of database design according to the requirements of the second case. (2. Implementation domain)
- Determine and identify the problems that arise from inflexible design
- Propose a new design to prevent the implementation problems
- Analysis of the proposed design according to the requirements of case 2.

3.1 APPLICATION DOMAIN

We can define the application domain as a specific area that a software application interested in. Application domain knowledge is important for requirements engineering in order to establish and understand analysis of interested area. Also business process reengineering needs for a particular system can be addressed by application domain knowledge. In ERP systems, application domain usually refers to the module. We have selected budget application domain for this thesis.

Budget module mentioned in this thesis is basically used for listing the description of planned expenses in a period. Also, it is a road map of firm’s expenses in its valid period. Budget definition may cover project budget, departmental budget, and personnel budget. In this thesis we have interested in firms’ expense budget in a determined period.

3.2. IMPLEMENTATION DOMAIN

We have to find at least two different implementation domains in order to better recognize implementation problems. We suppose that different implementation domains

have different requirements. Our implementation domains are defense and construction industries.

Our first and elementary design is for a firm that is in defense industry. We designed a database for their specific needs and business processes. Our second and compared firm is in construction industry. We collected their requirements for budget application domain and compared their requirements with our first design.

3.2.1. Budget Application for Defense Industry: Case Study 1

We have used generic database design technique for the case study 1. This technique involves the below phases:

- Requirements Collection and Analysis
- Conceptual Design
- Logical Design
- Physical Design

At the first phase we have collected requirements by means of individual interviews. We have interviewed three groups of personnel;

- Administrators of Budget Department of the firm
- Users that prepare the budget for their departments and projects

At the second phase, we have created Entity Relationship (ER) diagrams according to the requirement analysis. We have identified entities, attributes and relations in order to create the diagrams.

At the third phase, we have identified primary and foreign keys and index structures for the entities identified in the conceptual phase. Also we have identified our commercial database system as Oracle 9i RDBMS.

At the fourth and last phase, we have created the tables according to the table creation and storage structures that are being used by Oracle RDBMS.

3.2.2. Budget Application for Construction Industry: Case Study 2

We only have collected and analyzed the requirements of construction industry. Then according to these requirements we have evaluated the design that was developed in the first case. We have collected requirements by means of individual interviews. We have interviewed with there groups of personnel;

- Administrators of Budget Department at centre of the firm
- Accountants of local construction areas
- Users that prepare the budget for their departments and projects

3.3. ANALYSIS OF THE PROBLEMS

First, we have listed the problems and analyzed them. Purpose of this step is to determine the problems that are about inflexible design of database for the budget module.

3.4 NEW DESIGN OFFER(S)

At this step, we have proposed new design alternatives to eliminate the implementation problems that we have analyzed. An ER diagram of new design is also constructed. The problems of implementation are solved according to new design alternatives. The results of the case study were mainly explained at this step.

CHAPTER 4

CASE STUDIES

There is a strict difference between a software development project and an ERP implementation. Software development is preparing your own cloth according to your needs. But implementing ERP is working within a well-defined design framework, which is already designed and developed by someone else [11]. Therefore, ERP software is standard software and there would be some problems while implementing it.

Our method to identify implementation problems of an ERP system is to setup a case study. This case study consists of implementation of an ERP module (application domain) in two related phases. At the first part, we have designed a database for chosen application domain (budget), in order to satisfy specific needs of an organization in defense industry. At the second part, we have collected the requirements of another organization in construction industry. Then we analyzed the requirements of the second organization with our first database design. We assume that, choice of two different industries (implementation domains) will help us to catch more problems. At that point we can analyze problems that arise from inflexible structure of database design.

4.1 BUDGET

Budget application domain has been chosen for application domain to express the implementation problems in different areas of industry. But what is budget, why it is important and why organizations desire to make a budget? Budget is a powerful management and planning tool to express what the organization will do in the related period [9]. It is important because it emphasizes detailed resource allocation according

to the business plan that is constructed in related period. Related period always means a year, therefore budgeting always mentions annual budget. Organizations make budgets and report expenses to identify performance of business plan and department managers. Good budgeting leads to good management, which leads to good business performance [9]. Budget preparation is directed by top management but whole organization would have contributions to it. Department managers must have important input in budgeting because they are directly involved and knowledgeable about their departments' activities and familiar with operations [10]. Budget and budgeting process may have differences according to countries, industries, organizations but the philosophy remains the same.

4.2 BUDGET APPLICATION FOR DEFENSE INDUSTRY

4.2.1 Requirement Collection and Analysis

We have chosen a sample firm to get requirements of budget applications in defense industry. First of all, we started to interview with specialists in the Budget and Finance Department. Notes based on the interview are as follows;

Budget of the firm consists of two main parts:

- Cost Center Budgets
- Project Budgets

There are various cost centers in the organizations. Each employee in the firm belongs to a cost center. Each cost center has its own budget to arrange expenses planned. There is a responsible person for preparing each cost center's budget. Therefore, owner of the budget is the cost center which has prepared it. Expenses of these cost centers are not related to any of the projects that the firm is concerning with.

Firm has projects for saving money. These projects are grouped as:

- Production Projects
- Research and Development Projects
- Service Projects

All projects have their own budget in order to identify and group the expenses. These expenses are also identified by cost centers with the coordination of project management departments. Project management departments are organized based on the type of project we mentioned above. But owner of these project budgets is not the cost

center that has prepared it. Project budgets are owned by Project Management Departments according to its group.

Firm budget is prepared by cost centers in three states:

Preparation State: During the budget preparation period each cost center and each project related with the cost center builds their own planned expense list. This is a draft version of the budget. Projects and cost centers validate and may add or extract some of the items into the list.

Freeze State: After the preparation is done, the budget would be ready for board of director's approval. As the Board of Director's approval is done, then the budget passes to the definite state. If any exception occurs budget returns to the preparation state.

Definite State: When the budget is approved, no insert or update is allowed. As the budget period starts, cost centers and projects can have purchase orders that are identified in the budget.

After the Definite State, (in which year the budget is valid) there may be new requirements or change of requirements which have not considered during the preparation of the budget. Under the control of Budget and Finance Department, these requirements are handled by adding new expense lines or transferring planned expense lines to each other. Cost Centers prepare the purchase orders from their budget or their budget line from related project budget.

Budgets are basically lines of expenses with amount and currency of money with its period. There are specific properties of these budget lines;

- Type of expense
- Description and Reason of expense
- Preparer Cost Center
- Currency code of expense

Also periods, total money and total quantity that you plan for this budget line will be entered. Firm also budgets its employees in numbers of personnel and work hours for each month. Personnel budget is prepared in detail of cost center, position and work hour budget is derived from personnel budget in monthly work hours. Work hour budget is also detailed in cost center, position and projects. Time sheet fill is a property for

positions and cost centers to calculate work hour. Work hour budget is only done for positions and cost centers which have time sheet fill property.

4.2.2 Summary of Requirements

We can summarize these requirements as follows:

- The company has a Budget and Analysis Department.
- Budget and Analysis Department has authority to control all budgets.
- Budget and Analysis Department observe the budgets of Cost Center and Projects.
- Budget and Analysis Department defines the budget of the firm.
- Budget definition consist of project, owner cost center, name ,status, related year and the type of the budget
- There are 3 states of budget definition; Preparation, Freeze, Definite
- During the Preparation state budget is determined and lines expenses are entered. During the preparation state cost centers prepare the lines of expense.
- During the freeze state no update or insert into budget is allowed. During the freeze state it is decided if the whole company budget is accepted or not.(until the beginning of the budget related year)
- During the definite state no update or insert into budget is allowed also requisitions from this budget starts. During the definite state related year's budget is published and became certain.
- Each Cost Center has specific name, code and time sheet fill.
- Each Project has specific name and code.
- Each position consists of position code, time sheet fill and position_name.
- New cost centers and projects would be added in life cycle of the firm.
- There are two types of budgets; Personnel and Expense.
- In the Personnel Budget the number of staff with positions for each department in relevant month and year is determined.
- For the related year's budget, work hour per month is calculated as follows: Each month's workday X work hour.

- Time sheet fill property assigned to positions and cost centers for calculating work hours.
- Yearly work hour budget for cost centers consist of work hours determined in the name of position and project for the relevant month.
- In the Expense Budget planned expense of the firm is determined.
- Expense Budget is divided into two categories. Project budgets and Cost Center budgets.
- Every expense is relevant with a cost center. This is required for project budgets. Project budgets are prepared by one or more cost centers.
- Budget and Analysis Department defines the other cost centers which can have expense items in the project budget.
- During the preparation time Project budget can be observed by Projects owner cost center but every cost center can only change its own written requirement.
- Expense consists of definition, reason, unit of measure, currency code, expense cost center, expense type and inventory item.
- Expense type consists of code, description and type.
- Each expense must have one or more expense period (year, month) and expense amount (quantity, money) related with this period.
- Cost Centers prepare the purchase orders from their budget or their budget line from related project budget.
- In the related year under the control of Budget and Analysis Department any expense line of a budget can be transferred into other expense line of budget in terms of money. History of the transferred information is reserved with the details of; from which budget, from which expense line, from which period and to which budget, to which expense line, to which period.

4.2.3 Conceptual Design

According to the requirements we can pass to the second phase of the design; conceptual design phase. First we will identify the entities of our mini-world.

- Entity: Project

  Attributes: NAME, CODE

- Entity: Cost Center

  Attributes: NAME, CODE, TIME SHEET FILL

- Entity: Purchase Orders

  Attributes: DATE, QUANTITY, UOM, DESCRIPTION, CURRENCY, REASON, STOCK_NO

- Entity: Expense Budget

  Attributes: NAME, DEFINITION, YEAR, PERIOD, CURRENCY CODE, AMOUNT, UOM, REASON, EXPENSE LINE TYPE (definition, group, code), INVENTORY ITEM, STATUS, CHANGE HISTORY (date , from budget, to budget , amount),

- Entity: Personnel Budget

  Attributes: YEAR, PERIOD, POSITION (name, code, time sheet fill), QUANTITY, WORKHOUR, PROJECT

Top-down strategy is used for the schema design. Top-down strategy is primitively a decomposition of an entity type into several entity types [4]. High level entity types are specified and then these entities are split into lower-level entity types and relations with the help of normalization rules. In Figure 4.1, ER schema of the budget system can be seen. According to the entities, attributes and relationships are identified during the requirements collection step, schema has been constructed.

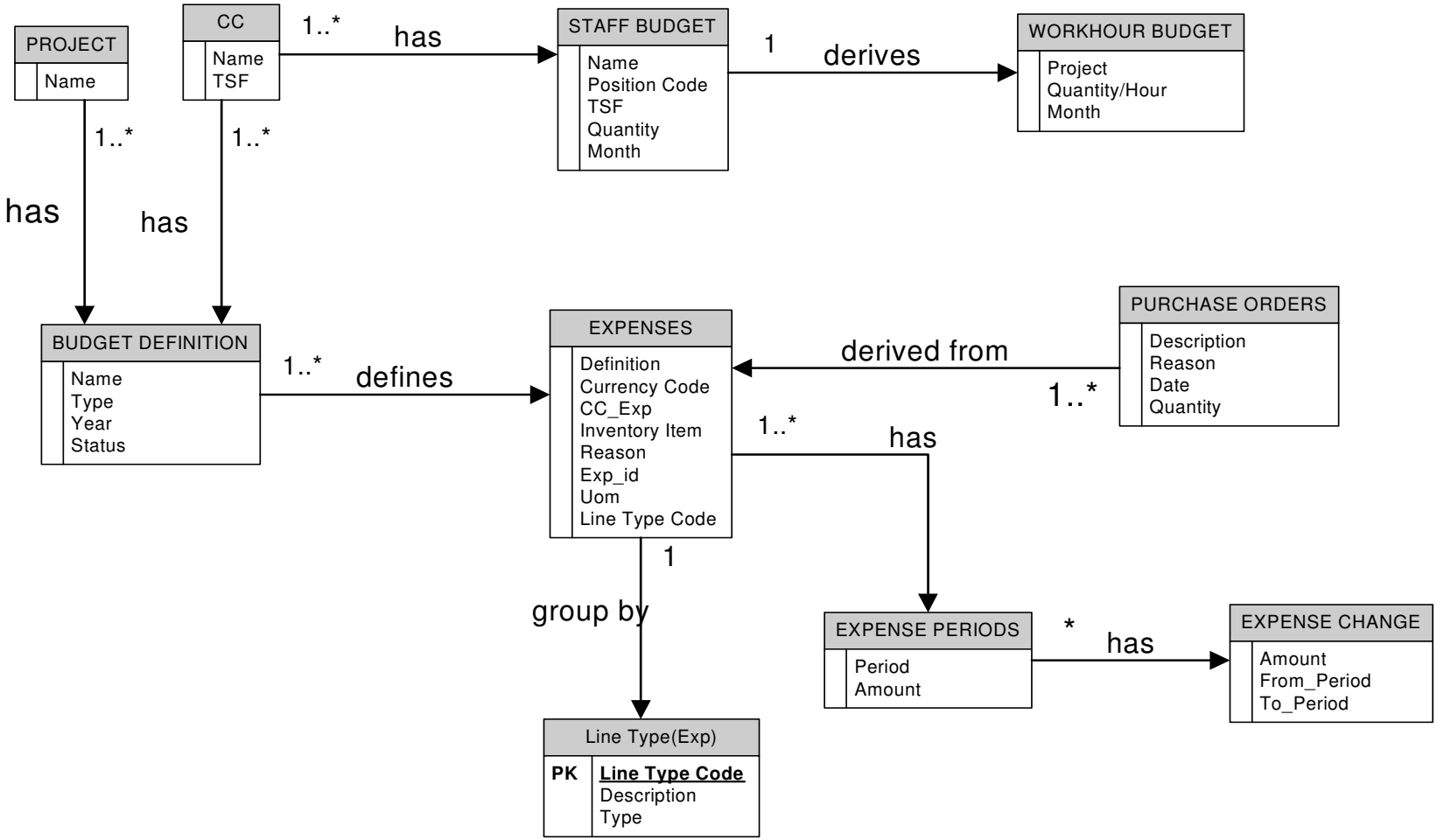


Figure 4.1 Conceptual Design of the Budget System (IDEFIX notation)

4.2.4 Logical Design

In the logical database design phase, the data model mapping of the system which is defined in the conceptual database design, is used. ORACLE database has been chosen for the system setup. Figure 4.3 shows logical design of the budget system, with relations and keys.

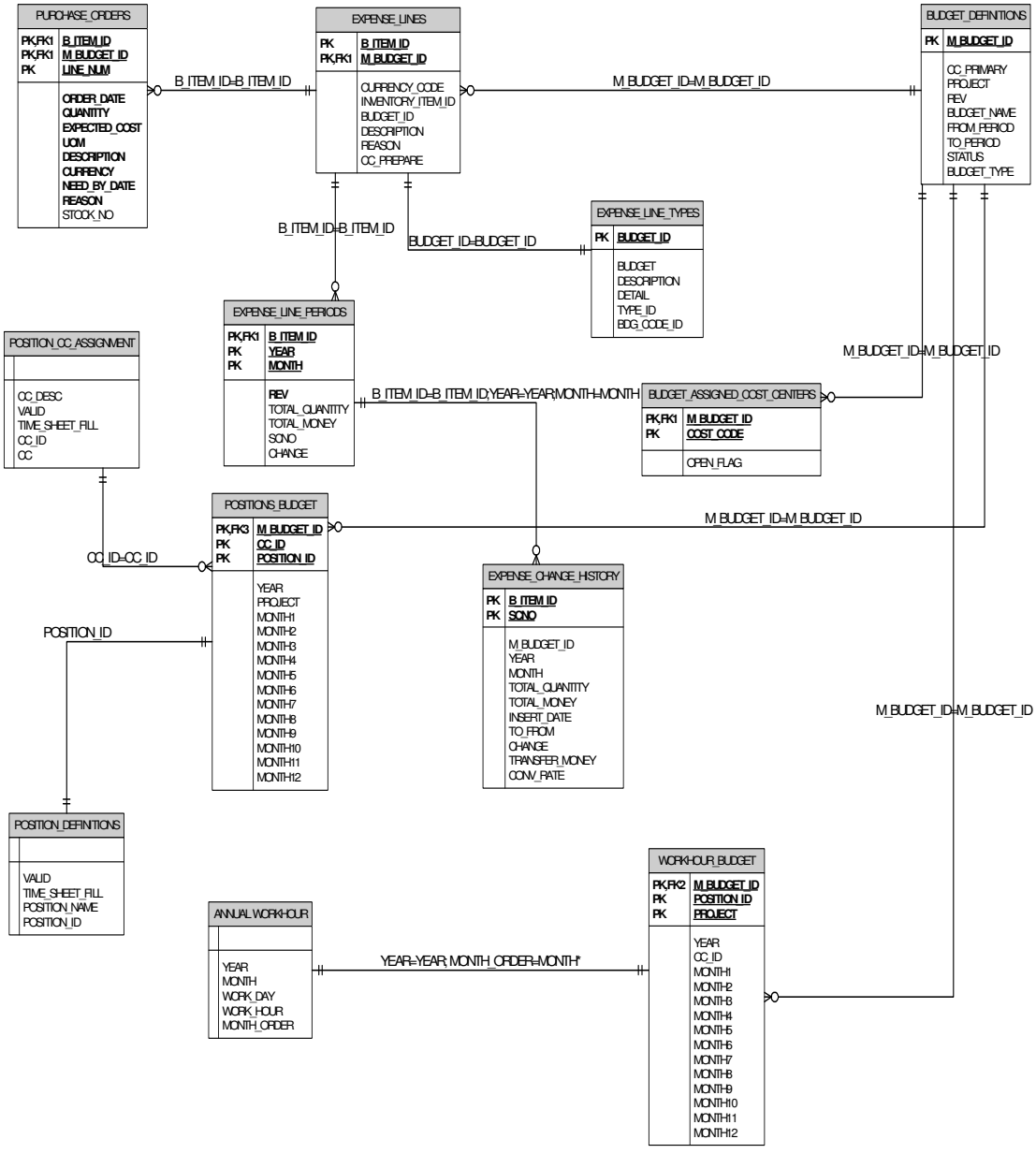


Figure 4.2 Logical Design of the Budget System (Crow's Foot)

4.2.5. Physical Design

In logical design, we have mentioned that commercial database management system was chosen as ORACLE. According to ORACLE SQL notation, physical designs (table scripts) are given in Appendix. Created tables and table definitions are below;

BUDGET_DEFINITIONS table

Budget definitions and management will be done by this table. This table manages the state and identifies the owner cost center of the budget. Also validation period and type of budget is determined.

BUDGET_ASSIGNED_COST_CENTERS table

This table defines the authorization to cost centers for budgets which have defined in the table BUDGET_DEFINITIONS. Cost centers authorization definitions could be disabled without deleting the record from the table; it can be done by column OPEN_FLAG.

EXPENSE_LINES table

This table stores the expense lines with relation of budget definitions table. (BUDGET_DEFINITIONS)

EXPENSE_LINE_TYPES table

This is the definition of expense line types. Expense line types are necessary for grouping expenses lines.

EXPENSE_LINE_PERIODS table

This table stores the detail information in period and amount according to expense lines with the relation of EXPENSE_LINES.

EXPENSE_CHANGE_HISTORY table

This table stores change history of expense lines after the budgets definite state.

POSITION_DEFINITIONS table

This table stores the definitions of positions.

POSITIONS_BUDGET table

This table stores budget details for positions defined in table POSITION_DEFINITIONS

POSITION_CC_ASSIGMENT table

This table stores the cost centers that will positions assigned to. Therefore, budgeting any position and any work hour for a cost center requires a setup in this table.

ANNUAL WORKHOUR table

This table stores amount of total work hour for the related month and year. This table records is determined by Budget and Analysis Department.

WORKHOUR_BUDGET table

This table stores work hour budget information according to project, position and cost center.

PURCHASE_ORDERS table

After the definite state of budget, purchase order starts. Purchase order records are stored in this table.

4.3 BUDGET APPLICATION FOR CONSTRUCTION INDUSTRY

4.3.1 Requirement Collection and Analysis

Since we would not develop a program for construction industry, analysis of this system is different from the first one. We will only implement the existing budget system which was prepared for the defense industry. Therefore, we only do the current state analysis for the construction industry. Also we will (if needed and wanted) process

re-engineering according to the designed system. We would and should not change the design of the first system (ERP philosophy entails this type of approach).

An interview is done with a Turkish construction company that has projects not only in Turkey but also in other countries. Firms' accounting department has helped us to analyze their budgeting system.

Firm has projects and construction areas. Construction areas are associated with projects. All construction areas have its own expenses and their expenses are in country's currency code of the construction area in. They have no budget for their employees, they write a constant expense line for employee outcomes as they do for their other expenses. Expenses are planned for a year and grouped monthly. They have categorized expenses to different codes. Therefore, they can report easily to the top management of firm. They also plan their annual sub-contractor budget. This means first they divide the planned jobs into two parts;

- Jobs that will be done by firms resources
- Jobs that will be done by sub-contractors' resources

Then the sub-constructor expenses are determined. Project budget is prepared after all construction areas finished their expense planning. Then center office of the firm makes its own budget for general expenses and project related expenses. All planned project expenses are controlled by project managers. Firm has grouped expense times into two categories;

- Time that the material will be received in(as a stock, as an entity, or as an service)
- Time that the money will be paid (cash)

Firm prepares expenses with description, reason, period and amount of the expense. Also firm pays custom commission to government for its imported goods and they calculate an expense for each imported entity. But construction areas only determine if the entity is international or local.

No budget states are used in this firm but in our program we have an opportunity for that. If they want to use this new attribute of this program their budget preparation would be more controllable.

In the related budget year budget transfer or change is not applicable for the construction industry. Existing database design of budget system provides the opportunity for the changes in live budget periods.

Purchase orders are derived from the prepared expense lines of budget. In construction industry, expense lines have extra data requirements in the purchase orders phase. The personnel who create the purchase order may enter the expense's alternative vendor name(s) and the building name (if possible) where the order is made for.

4.4. DETERMINING IMPLEMENTATION PROBLEMS

Budget for construction industry includes construction areas instead of cost centers. Therefore, construction area budget replaces cost center budget in the definition of budget. Project approach is same as our module. Thus project budget is the same as designed budget module.

Personnel budget in the construction industry is not applicable. There is no business process for personnel budget preparation. Personnel expenses will be treated as an expense line and will be handled by a line type code. As a new attribute of an expense line, type of expense resource (sub-contractor resource or firm resource) arises. Database design has no column for that extra data entry. A new concept in construction industry is that the expense line payment and delivery may appear in different budget periods. Therefore, a new attribute is needed for expense lines, payment or delivery flag. Another extra requirement is the place of expense, local or abroad. Also there are new purchase order attributes. Alternative vendors and building name which the purchase is made for.

Table 4.2 shows us summary of the implementation process and differences of the two industries according to main parts of the budget system.

Table 4.1 Implementation Summary

	DEFENSE	CONSTRUCTION	PROBLEM
PERSONEL BUDGET	OK	N/A	Since there is no need, no implementation problem
WORKHOUR BUDGET	OK	N/A	Since there is no need, no implementation problem
EXPENSE BUDGET	OK	"Place of Expense" and "Resource Type" attributes are missing. All other requirements are met.	Implementation problem, extra data entrance required.
PURCHASE ORDERS	OK	"Alternative Vendors" and "Building Name" attributes are missing All other requirements are met.	Implementation problem, extra data entrance required.
EXPENSE PERIODS	OK	Payment or Delivery information about planned expenses is missing. Other requirements are met.	Implementation problem, extra data entrance required.
BUDGET DEFINITION	OK	OK	Construction Areas treated as cost centers in the defense industry, thus, no implementation problem

According to logical design of budget system, the problematic implementation data needs are;

EXPENSE_LINES

Place of Expense, Resource Type

EXPENSE_LINE_PERIODS

Payment or Delivery Flag

PURCHASE_ORDERS

Alternative Vendors, Building Name

CHAPTER 5

DESIGN ALTERNATIVES

Chapter 4 of this thesis has identified the implementation problems that arise from inflexible ERP systems, thus inflexible database design of ERP systems. Then a question appears "How can we construct an ERP system that handles these implementation problems?" There are two extreme points in implementation of ERP systems. One is the complete fit of application domain (in the ERP system that implemented) to implementation domain. And the other one is complete difference of application domain to implementation domain. First extreme point has no problem of implementation about flexibility. Thus, this case is out of scope for this thesis. The second case needs full customization. So organization needs a custom solution for itself, a new program should be developed that has an interface with ERP system. This case is also out of scope for this thesis. We are concerned with the area that is between these two extreme points.

What is between these two extreme points? Answer of this question is, extra data (attribute) requirement(s) for each record in the implementation domain. This extra data requirement would be;

- Specific to country
- Specific to sector
- Specific to firm itself

How can we solve this problem at the ERP system's database design? There is a fact that the organization wants to keep the records of these extra data. Thus, we have to include these attributes in our design. Then do we have to re-design the ERP system?

For newer versions we can analyze the design for application domain according to feedbacks, but the answer is no. ERP design would not change for each specific implementation needs. The solution is a flexible design of database for ERP systems.

Nowadays, database design for ERP systems has a management schema like the commercial Relational Database Management Systems (RDBMS) has. (Metadata - sys schema) Data for table configurations, user definition, authorization and other management issues are handled in this management schema. Therefore, authentication and authorization are not mentioned in this thesis. In literature survey, we have mentioned about ERP security mechanisms. Therefore, it is assumed that ERP system management schema and programs provide the mechanisms of management. Module implementation of ERP systems is the main issue of this thesis.

5.1 EXISTING DESIGN ALTERNATIVE - EXTRA ATTRIBUTES

We can define new empty attributes at the database design step for probable implementation needs. If any extra data entry is needed, then these empty-created columns can be used. For example, expenses that the construction industry determined has an attribute for purchasing the goods from abroad or not, is required. Therefore, an extra attribute that is left empty intentionally can be used for the requirement above. Figure 5.1 shows us the new database design of the budget system with empty and extra attributes. No empty attributes are defined for definitive tables as EXPENSE_LINE_TYPES.

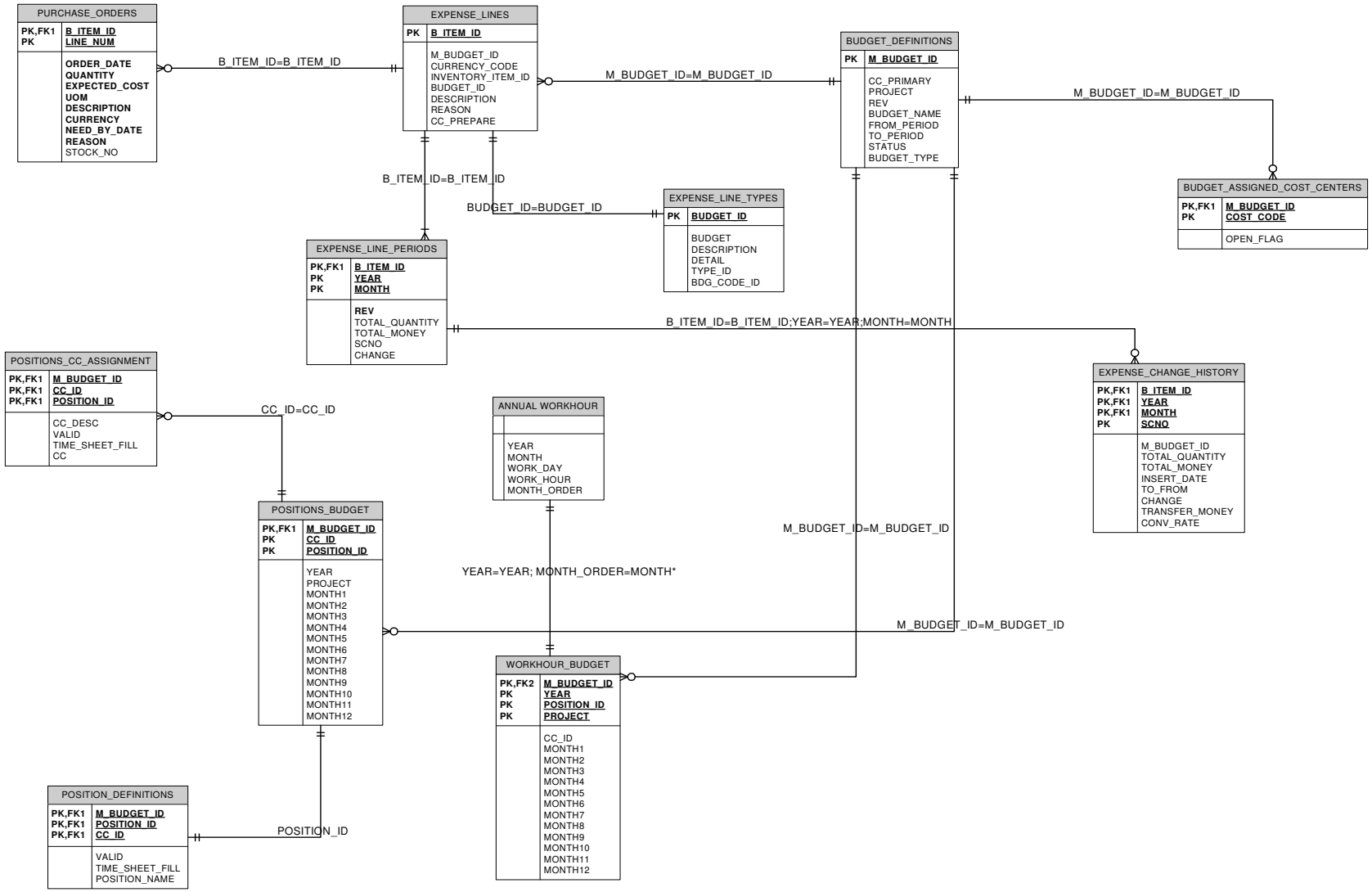


Figure 5.1 Logical Design with Empty Attributes (Crow's Foot)

But how can data entry be controlled and managed? Extra data requirement should be addressed to an appropriate empty column. The characteristics (name, type and size) of this requirement should be determined and recorded. Also, data entry to this column may be from a set of values which are determined in the implementation phase. The values of this set may be dynamic in the life-cycle of the ERP system.

For managing extra attributes which will be required in the implementation of the module, the new schema APP_MNG (Application Management) is created. In the new schema there are 3 tables to manage the extra attributes:

Table and Column definition of the new schema APP_MNG

- APP_MNG.CUSTOM_ATTRIBUTE_LIST
- APP_MNG.CUSTOM_LIST_SET
- APP_MNG.CUSTOM_LIST_VALUES

CUSTOM_ATTRIBUTE_LIST	
PK **PK**	**TABLE_NAME** **COLUMN_NAME**
FK1	**VALUE_LIST_SET_ID** **END_USER_COLUMN_NAME** **DESCRIPTION** ENABLED_FLAG REQUIRED_FLAG SIZE TYPE

CUSTOM_ATTRIBUTE_LIST:

Table_name: Name of table which needs extra attributes according to the implementation requirement.

Column_name: Name of the column which new data requirement will be entered

Value_list_set_id: Value set of extra attribute (If null free from any list)

End_user_column_name: Column name that end user interacts with (in the interface)

Description: Meaning/Reason of the extra attribute

Enabled_flag: Validation flag (to remove unused extra attribute)

Required_flag: Nulllable of extra attribute

Size: Size of the extra attribute (Limited by database column size)

Type: Type of the extra attribute (number, character, date)

CUSTOM_LIST_SET	
PK	**VALUE_LIST_SET_ID**
	VALUE_LIST_SET_NAME **DESCRIPTION** TYPE

CUSTOM_LIST_SET:

Value_list_set_id: Id of value set

Value_list_set_name: Name of value set

Description: Meaning/Reason of the value set

Type: Specified type of the value set members (number, character, date)

CUSTOM_LIST_VALUES	
PK,FK1 **PK**	**VALUE_LIST_SET_ID** **VALUE_LIST_ID**
	VALUE_LIST VALUE_LIST_MEANING DESCRIPTION ENABLED_FLAG

CUSTOM_LIST_VALUES:

Value_list_set_id: Id of value set

Value_list_id: Id of value list item

Value_list: Specified value (empty attribute will be filled with this value)

Value_list_meaning: Meaning of the value list item

Description: Meaning/Reason of the value list item

Enabled_flag: Validation flag (To remove unused items)

The table 'CUSTOM_ATTRIBUTE_LIST' contains address of new data requirement. The table and column name of the empty attribute required is recorded in this table. Also the specifications of the data are defined in this table. This process is re-

define and re-design of meta-data not in RDBMS but in the management tablespace of ERP system. Since empty attributes of tables would be meaningful according to implementation needs, then value list for these attributes should be assigned. Assigned value list sets to the specific columns of the tables which have new data requirement, are defined in CUSTOM_LIST_SET table. This table holds the master data of the value lists. CUSTOM_LIST_VALUES table stores the values of extra attribute records that are assigned to a value set. As we can see from Figure 5.2, the join of these tables is made by the VALUE_LIST_SET_ID columns.

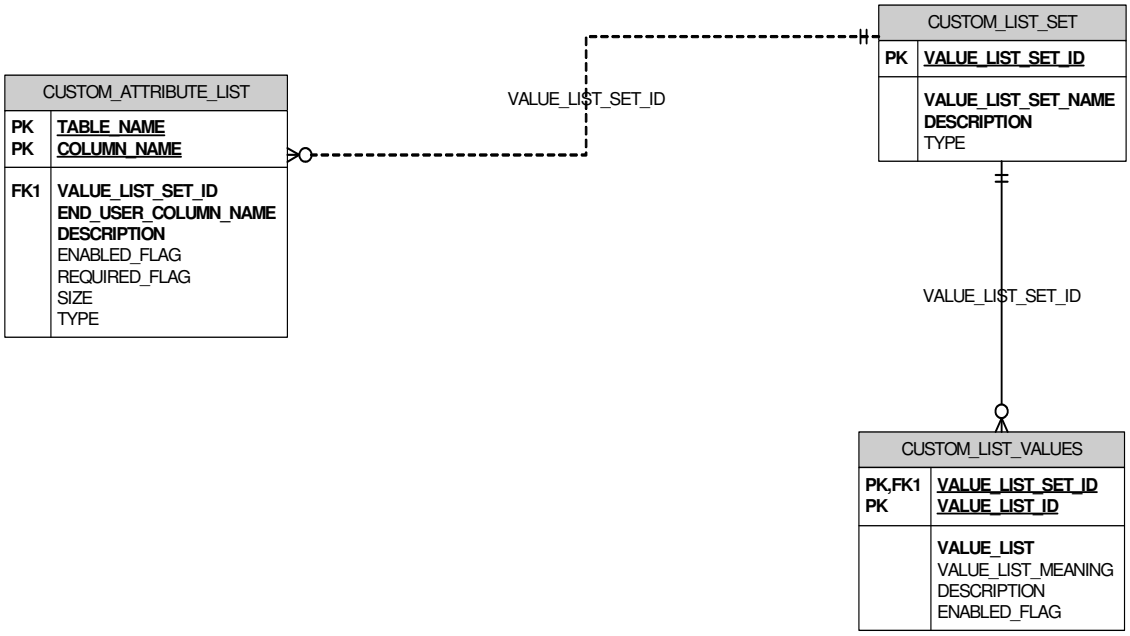


Figure 5.2 Extra Attribute Management Tables (Crow's Foot)

In Figure 5.3, the Budget System has shown with the management schema tables above.

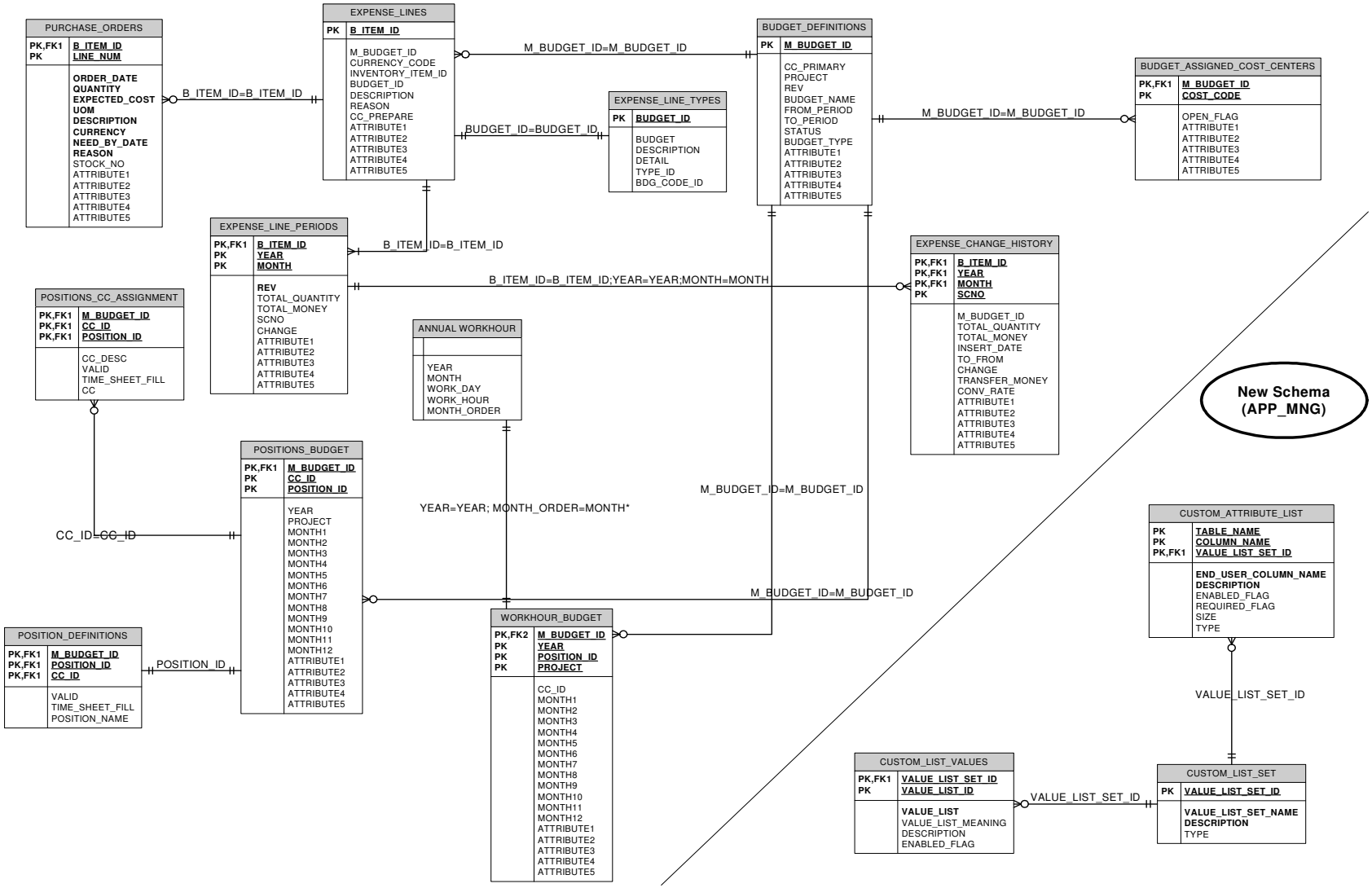


Figure 5.3 Design Alternative 1 with Management Schema (Crow's Foot)

Then at the implementation phase, these attributes will be used for extra data needs. But there are some dark points for this design alternative.

- What is the minimum and maximum number of extra attributes?
- Needed extra attribute and defined empty attribute may have different data types
- Size of the empty attribute is defined and cannot be changed
- Empty attributes contradict with normalization rules

As we have mentioned, extra management tables have been created for keeping the records of configuration and meanings of these extra attributes for the implementation domain. Also, definition of a new schema to hold the meta-data of ERP system has been made. This new schema solves reporting problem. Because it keeps the meaning of extra attributes for implementation domain. There is no minimum number of extra attributes (it would be zero) but there should be a maximum. This number would be identified by two extreme points of implementation that are explained above. If the number of extra data requirements is bigger than an acceptable number, then, full customization has to be done. In this thesis, acceptable number is taken as five but at real life examples it would be more than that.

Implementation requirements drive modifications to the empty attributes. An extra attribute that had been created as a character, can store every type of data if needed. But, reporting that data in the required format may need some code modifications in the reporting environment. Also the size issue may be a problem when the defined size is under the value of the required size at the implementation phase. Empty attributes violates normalization rules. First normal form eliminates null attributes and moves them into a separate table in a generic design. In fact empty attributes will be added after normalization rules applied to the database design of ERP system. Therefore normalization-denormalization order arises. This is a disadvantage of this solution alternative.

The design method described above is similarly being used by ERP vendors; SAP and ORACLE. As mentioned in the literature survey, these two ERP vendors have the greatest market share in the ERP market. Both of this ERP software is investigated. Two firms that are implementing the SAP and ORACLE were visited in order to understand

the solutions of these ERP vendors. SAP is using user-exits with include-statements. Oracle is using flexfields. But as we said before solutions of these two ERP vendors is similar to our first design alternative.

5.1.1. Flexible Design of Budget System and Solution of the Problems of Implementation

ERP module that had been designed for budget system was implemented and implementation problems identified. According to new design offer, table structure and sample records of this structure are below;

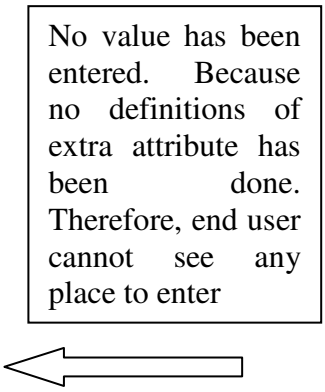
To demonstrate the situation, only the extra data requirement "Place of Expense" will be taken. An instance of records will be shown below the tables in order to illustrate the solution.

Table 5.1 shows us EXPENSE_LINES table before any extra data inserted. Insertion of records has been done as in Table 5.1.

Table 5.1 Expense Lines without implementation requirements

EXPENSE_LINES table

B_ITEM_ID	5544
M_BUDGET_ID	12
CURRENCY_CODE	USD
INVENTORY_ITEM_ID	NULL
BUDGET_ID	2
DESCRIPTION	CEMENT
REASON	BUILDING SURFACE
CC_PREPARE	175
EXTRA_ATTRIBUTE1	NULL
EXTRA_ATTRIBUTE2	NULL
EXTRA_ATTRIBUTE3	NULL
EXTRA_ATTRIBUTE4	NULL
EXTRA_ATTRIBUTE5	NULL



First the table and empty column requiring modification has to be introduced to the system. Table 5.2 shows us the metadata of the EXTRA_ATTRIBUTE1 column. We have defined data for related table, related column, name and characteristics of the

column. Then, the column has been enabled to insert and update data. Also, a value set was assigned to the related column.

Table 5.2 Introduction of the empty attribute to the system.

CUSTOM_ATTRIBUTE_LIST table

TABLE_NAME	EXPENSE_LINES
COLUMN_NAME	ATTRIBUTE1
VALUE_LIST_SET_ID	1
END_USER_COLUMN_NAME	Place of Expense
DESCRIPTION	Place of Expense
ENABLED_FLAG	Y
REQUIRED_FLAG	Y
SIZE	10
TYPE	VARCHAR2

Since, assignment of a value set to the custom attribute has been done; it has to be defined to the CUSTOM_LIST_SET table. Table 5.3 shows us value list characteristics of assigned value set.

Table 5.3 Value Set records defined for the custom attribute.

CUSTOM_LIST_SET Table

VALUE_LIST_SET_ID	1
VALUE_LIST_SET_NAME	Expense Place Type
DESCRIPTION	Expense Place Type
TYPE	VARCHAR2

Also, members of the assigned value set should be defined to the CUSTOM_LIST_VALUES table. Table 5.4 shows us two members for the assigned value set. These values are determined in the collection of requirements phase second phase of the case study. As table 5.4 shows, value and meaning would be different for different reporting needs.

Table 5.4 Value Set Member records.

CUSTOM_LIST_VALUES Table

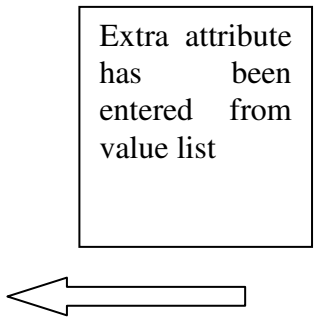
VALUE_LIST_SET_ID	1	1
VALUE_LIST_ID	1	2
VALUE_LIST	Local	Abroad
VALUE_LIST_MEANING	Domestic	International
DESCRIPTION	Local	Abroad
ENABLED_FLAG	Y	Y

Table 5.5 shows us the updated instance of the EXPENSE_LINES table. End user has chosen "Local" from the value list of "Domestic" and "Local". Therefore, implementation requirements are met for this extra data need.

Table 5.5 Expense Lines after the implementation requirements met.

EXPENSE_LINES table record example

B_ITEM_ID	5544
M_BUDGET_ID	12
CURRENCY_CODE	USD
INVENTORY_ITEM_ID	NULL
BUDGET_ID	2
DESCRIPTION	CEMENT
REASON	BUILDING SURFACE
CC_PREPARE	175
EXTRA_ATTRIBUTE1	Local
EXTRA_ATTRIBUTE2	NULL
EXTRA_ATTRIBUTE3	NULL
EXTRA_ATTRIBUTE4	NULL
EXTRA_ATTRIBUTE5	NULL



5.2. PROPOSED DESIGN ALTERNATIVE - NEW TABLE FOR CUSTOMIZATION

In the first design alternative, we have discussed empty attributes which have problems with the normalization issues. It cannot be guaranteed that all attributes, which have been left empty intentionally, will be used at the implementation. Therefore, repeating null values violates normalization.

The main problem is where the extra data (customization data) will be entered. If a suitable place (table) can be found to enter the customization data, then the problem would be solved. In the previous design alternative, the problem has been solved with intentionally left empty columns in the ERP system tables. Now, we are concerned with locating the extra place requirement to a different table. In this context, customization tables for extra data needs can be created.

5.2.1 Customization Tables

After the decision of moving customization data to a different table, we have faced new issues that have not been discussed. In first design, customization data has been managed and controlled by the data structure in the management tables. Also, there was no need to associate the customized attribute with the system table and relating customized data with system data.

Locating the customization data to a new table is the main aspect of our new design offer. But, this different table solution brings the question “Which custom table is for which real system table?” An identifier or a terminology may help about this problem. If we define customization table as “TABLE_CUST” for ERP system table “TABLE”, then the association between with two tables can be handled. In Figure 5.4, an example for this notation can be seen. The customization table for budget system table EXPENSE_LINES, has been created as EXPENSE_LINES_CUST.

EXPENSE_LINES_CUST for extra data requirements of EXPENSE_LINES table;

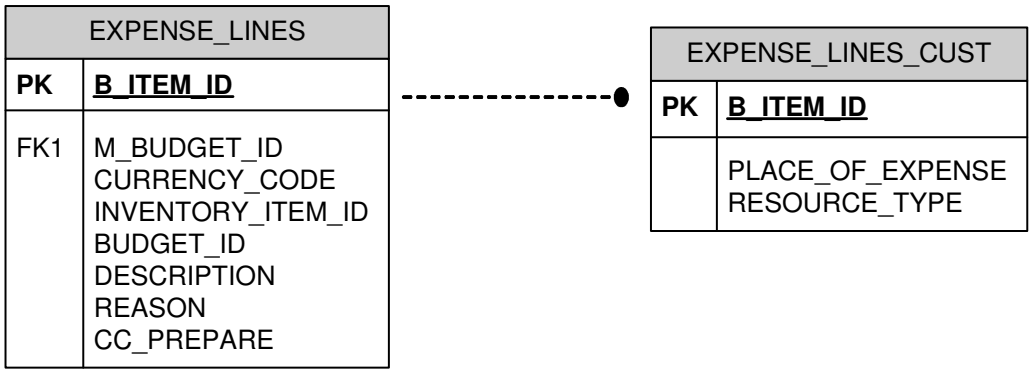


Figure 5.4 EXPENSE_LINES with custom table (Crow's Foot)

Also, in relational database manner these two tables, the ERP system table and its customization table should be related. In second design, two tables have no relational data connection. How can we connect the main data with the customization data?

When the creation of the custom table has been done, columns of the customization table can be determined as;

- Primary key(s) of the ERP system table
- Columns required for implementation

Therefore, same primary key(s) for different tables will help to setup the data relation between two tables. For example; primary key of the EXPENSE_LINES table is "b_item_id". Thus, table EXPENSE_LINES_CUST have been created with the primary key "b_item_id" and the extra attributes that are required for implementation. In Figure 5.4, ERP system table columns and customization requirement table columns can be seen together. The Budget System and its related custom tables according to the implementation requirements can be seen in Figure 5.5.

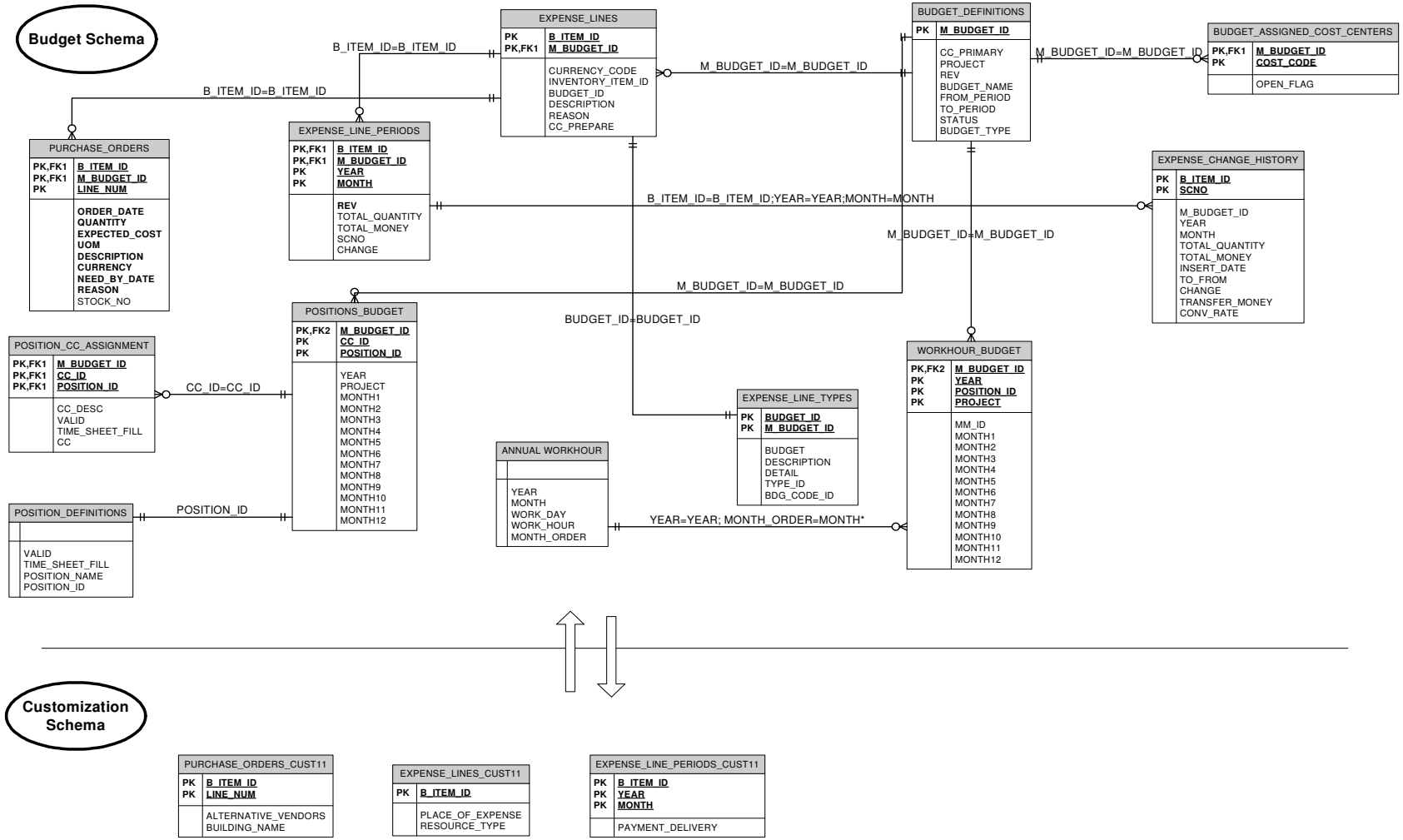


Figure 5.5 Design Alternative 2 – System with Custom Tables (Crow's Foot)

First design alternative have been discussed with its problematic sides. Empty attributes are determined at conceptual design phase of the ERP system. But implementation needs, arise latter. Therefore, conflicts between these two time stamps are the cause of the problems. At our new design alternative, problems are prevented by newly defined custom conceptual design, which has constructed according to the custom data requirements. Also, new design prevented the normalization problems. If there is no need for any custom data, no customization table is created. Therefore, repeating groups and null values are prevented. Also, characteristics of the attributes in the first design were a problem. The size, type and the name of the column can be defined without any limitation that has been faced in the first design. According to the new approach a new method can be constructed in order to ensure database design flexibility in ERP systems. In Figure 5.6, the new process of flexible database design for ERP systems can be seen.

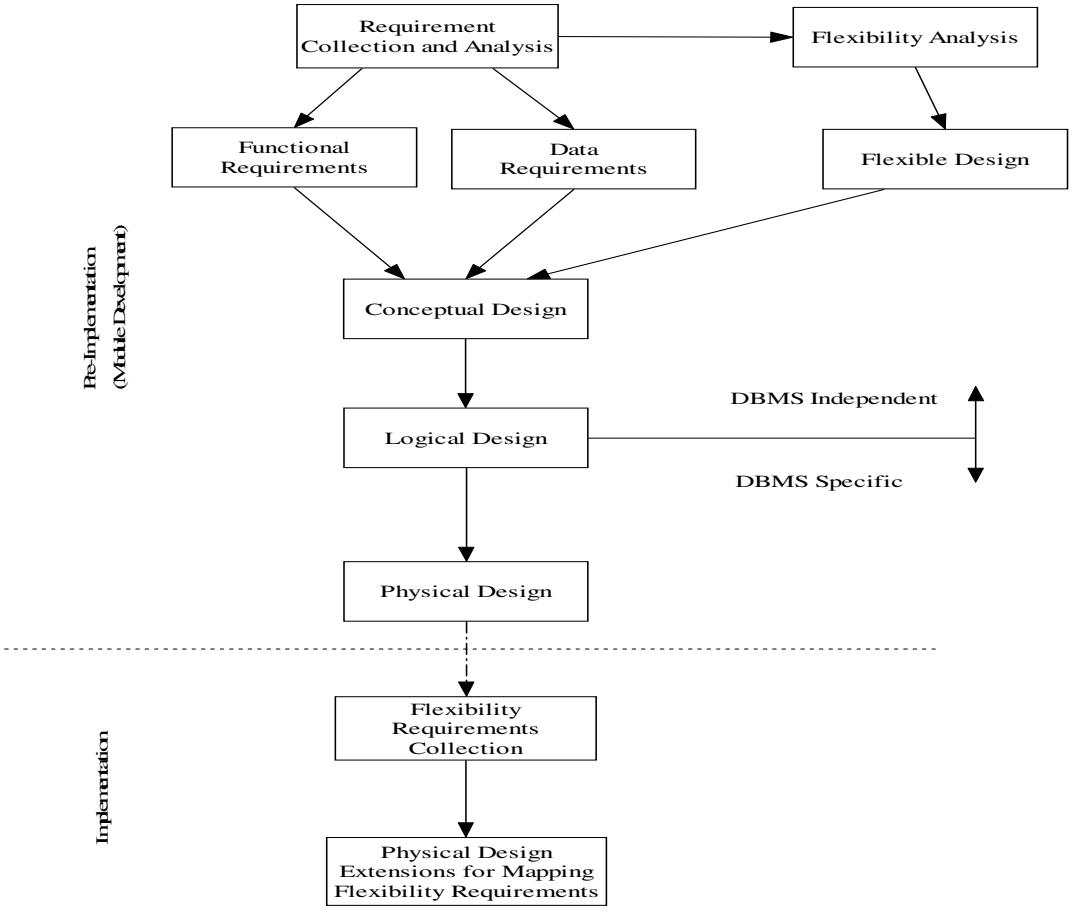


Figure 5.6 – Proposed Design Process

The proposed design for ensuring the flexibility of ERP system database also helped to generate the method. The method basically constructs the physical and semantic structure of flexibility requirements. Then, in the implementation phase, requirements that have been arised could have been added to the physical design of the existing database.

Management schema for managing and controlling the data insertion for custom data requirements have been mentioned in the first design alternative. Also, created custom table columns can use the solution which has been suggested in the first design. Therefore, assigning a value set for a custom table column and defining the value items of this set can be done. In Figure 5.7, three schemas for proposed design can be seen. First schema is budget system that is designed in the case study. Second schema is the custom requirements that are identified in the implementation. And the third schema, management schema, is for managing and controlling the data insertion.

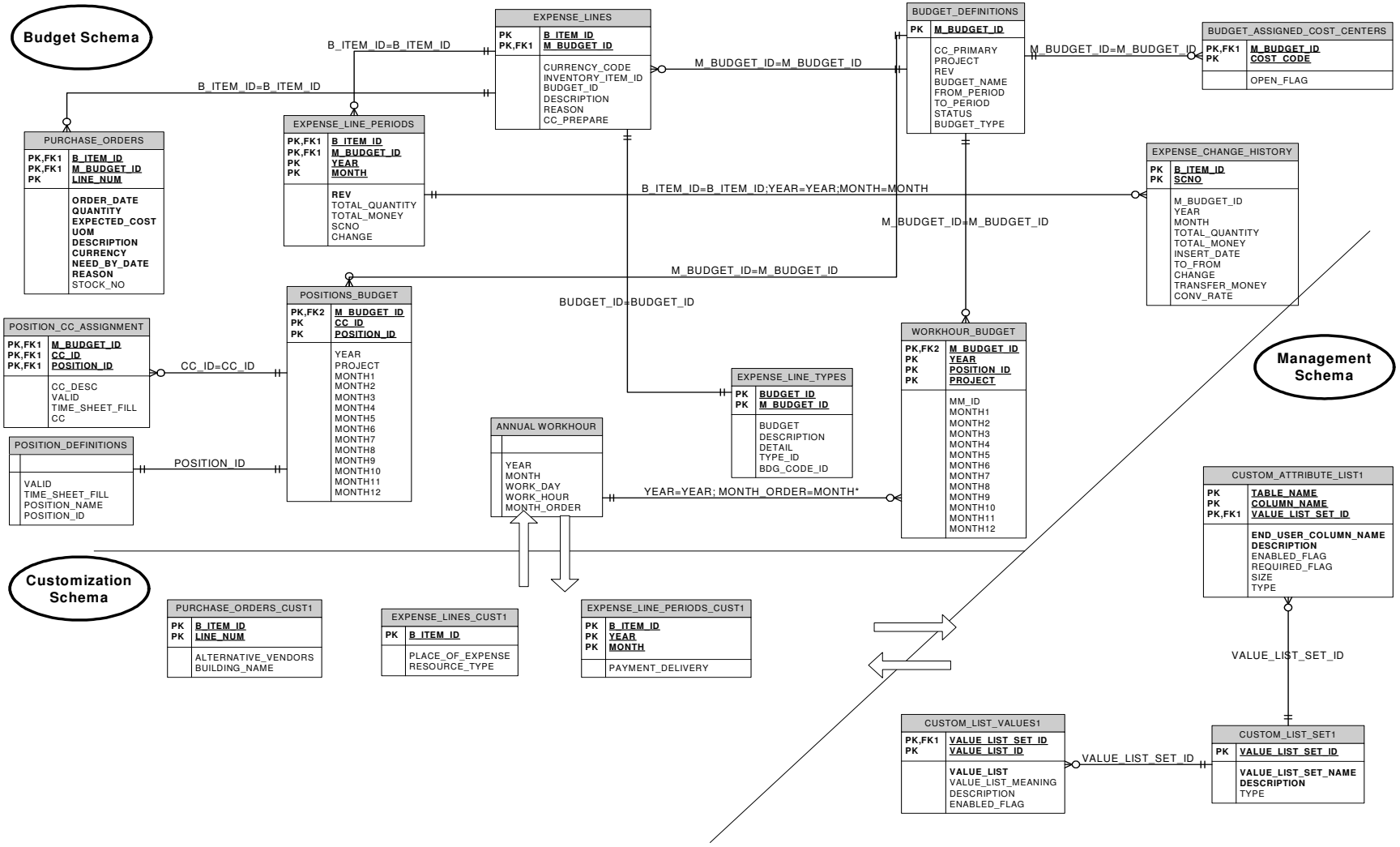


Figure 5.7 – Design Alternative 2 with Management Schema (Crow's Foot)

5.2.1. Flexible Design of Budget System and Solution of the Problems of Implementation

In Chapter 4, implementation problems for the budget system were identified. Proposed second design alternative has solutions to the problems of implementation. According to second design alternative, table structure and sample records for budget system implementation are below;

EXPENSE_LINES table has been taken to describe the solution. Extra data requirement "Place of Expense" will be recorded to a custom table which was created for implementation needs. An instance of records will be shown below the tables in order to illustrate the solution.

Table 5.6 shows us EXPENSE_LINES table without any custom data requirements.

Table 5.6 Expense Lines without implementation requirements met.

EXPENSE_LINES table

B_ITEM_ID	5544
M_BUDGET_ID	12
CURRENCY_CODE	USD
INVENTORY_ITEM_ID	NULL
BUDGET_ID	2
DESCRIPTION	CEMENT
REASON	BUILDING SURFACE
CC_PREPARE	175

After the custom data needs are identified, the custom table for the related table have been created. Table 5.7 shows an example of records for expense_lines.

Table 5.7 Expense Lines without implementation requirements met.
EXPENSE_LINES_CUST table

B_ITEM_ID	5544
PLACE_OF_EXPENSE	Local

First the custom table “EXPENSE_LINES_CUST” and “PLACE_OF_EXPENSE” column has to be introduced to the system. Table 5.8 shows us the assignment of a value set to the “PLACE_OF_EXPENSE” column. Then, column has been enabled to insert and update data.

Table 5.8 Introduction of the new attribute to the system.

CUSTOM_ATTRIBUTE_LIST table

TABLE_NAME	EXPENSE_LINES_CUST
COLUMN_NAME	PLACE_OF_EXPENSE
VALUE_LIST_SET_ID	1
END_USER_COLUMN_NAME	Place of Expense
DESCRIPTION	Place of Expense
ENABLED_FLAG	Y

Since, assignment of a value set to the custom attribute has been done; it has to be defined to the CUSTOM_LIST_SET table. Table 5.9 shows us value set definition of the assigned value set.

Table 5.9 Value Set records defined for the custom attribute.

CUSTOM_LIST_SET Table

VALUE_LIST_SET_ID	1
VALUE_LIST_SET_NAME	Expense Place Type
DESCRIPTION	Expense Place Type
TYPE	VARCHAR2

Also, CUSTOM_LIST_VALUES table has been filled with members of the assigned value set. Table 5.10 shows us the members for the assigned value set. This table is the same table that has been used in the first design alternative.

Table 5.10 Value Set Member records.

CUSTOM_LIST_VALUES

VALUE_LIST_SET_ID	1	1
VALUE_LIST_ID	1	2
VALUE_LIST	Local	Abroad
VALUE_LIST_MEANING	Domestic	International
DESCRIPTION	Local	Abroad
ENABLED_FLAG	Y	Y

CHAPTER 6

CONCLUSION

An organization's aim is dependent on its setup principles and its mission. But all organizations have to be effective and efficient. Enterprise Resource Planning (ERP) was born to be the infrastructure for helping organizations for their endless study to achieve this aim.

Implemention of ERP systems is a problematic process. ERP system is a well prepared suit, but this suit may not fit every organization. Thus, ERP software has to be flexible enough to meet the organization's requirements. ERP software consists of two major parts; program and database. Flexibility issue relies on both of these components. ERP software, first of all, should have a flexible database design, then, an interface that implements that design. Therefore, database design is crucial for flexibility of ERP software.

In this thesis, a two-phased case study is constructed to identify the implementation problems that arise from inflexible structure of database design. In the first phase of the case study, a database design has been done according to the specific needs of an industry. Then, the designed system has been implemented to a different industry to observe implementation problems. Analysis of implementation problems has been done. Extra data requirement problems are observed.

Afterwards, two database design alternatives have been proposed in order to have a flexible structure for ERP software. Discussion of both of the alternatives has been done. First alternative has been utilized by major ERP vendors in a similar way. The

implementation problems determined in the case study is also evaluated according to database design alternatives.

In this thesis, it is observed that the generic database design steps should be extended to evaluate the flexibility requirements. Importance of flexibility in the construction of new ERP software is emphasized in this thesis. Implementation diversity is inevitable; therefore, flexibility is mandatory for ERP software.

One possible future work may be evaluating the flexibility by object oriented databases. This thesis is based on a relational database system. When commercial object oriented databases would have been used for ERP systems then flexibility requirements would have been considered.

Another future work may be using XML data sets for "metadata" of ERP databases. Metadata for extra data requirements would be analyzed by adding extra definition lines or creating new XML files.

REFERENCES

[1]. Talu Ş., "SORULARLA KURUMSAL KAYNAK PLANLAMA" Publication Number :2004-27, İstanbul Ticaret Odası 2004

[2]. Grant G., "ERP& DATAWAREHOUSING IN ORGANIZATIONS: ISSUES AND CHALLENGES",2003

[3]. GartnerGroup Inc, http://www.gartner.com The GartnerGroup is the leading industry provider of research and analysis services.

[4]. Elmasri R., Navathe S., "Fundamentals of Database Systems", 3rd Edition, Addison-Wesley, 2000

[5]. Ferreira E. J., Busichia G., "Database Modularization Design for the Construction of Flexible Information Systems", Proceedings of the 1999 International Symposium on Database Engineering&Applications, IEEE Computer Society, 1999

[6]. Martyn T, "Implementation Design For Databases: The 'Forgotten' Step", IT Professional Volume 02 No:2,2000

[7]. Satzinger J.,Jackson R., Burd S., "System Analysis and Design", Course Technology Inc.,2000

[8]. Bernroider E., Koch S., "Differences in Characteristics of the ERP System Selection Process between Small or Medium and Large Organizations", Proceedings of the 6th Americas Conference on Information Systems pp 1022-1028, 2000

[9]. Finney Robert G., "Powerful Budgeting for Better Planning and Management." , American Management Association, 1993

[10]. Shim Jae K., Siegel Joel G. , "Budgeting Basics&Beyond",Prentice Hall, 1994

[11]. Ghosh S.,"Challenges on a Global Implementation of ERP software", IEEE 2002

[12]. Riet R., Janssen W, Gruijter P, "Security Moving from Database Systems to ERP Systems," *dexa*, p. 273, 9th International Workshop on Database and Expert Systems Applications (DEXA'98), 1998.

[13]. Gunson J., Blasis J., "The Place And Key Success Factors Of Enterprise Resource Planning (ERP) In The New Paradigms Of Business Management", Ecole des Hautes Etudes Commerciales, Universite de Geneve, Paper No: 2001.14,2001.

[14]. AMR Research Inc., www.amrresearch.com, "The Market Analysis Report: Enterprise Resource Planning, 2004-2009"

[15]. Klaus H., Rosemann M., Gable G., "What is ERP", Information Systems Frontiers, pp:141-162, 2000

[16]. Jarrar F.Y., Al-Mudimigh A., Zairi M., "Erp Implementation Critical Success Factors -The Role and Impact of Business Process Management", IEEE 2000.

APPENDIX

```
CREATE TABLE ANNUAL_WORKHOUR (
  YEAR         NUMBER (4)    NOT NULL,
  MONTH        VARCHAR2 (10)  NOT NULL,
  WORK_DAY     NUMBER (6,2)  NOT NULL,
  WORK_HOUR    NUMBER (6,2)  NOT NULL,
  MONTH_ORDER  NUMBER (2))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE BUDGET_ASSIGNED_COST_CENTERS (
  M_BUDGET_ID  NUMBER (15)   NOT NULL,
  COST_CODE    VARCHAR2 (4)  NOT NULL,
  OPEN_FLAG    VARCHAR2 (1),
  PRIMARY KEY ( M_BUDGET_ID, COST_CODE )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 40960 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS2147483645
```

```
 FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE BUDGET_DEFINITIONS (
 M_BUDGET_ID  NUMBER (15)   NOT NULL,
 CC_PRIMARY   VARCHAR2 (4),
 PROJECT      VARCHAR2 (10),
 REV          VARCHAR2 (5),
 BUDGET_NAME  VARCHAR2 (50),
 FROM_PERIOD  VARCHAR2 (4),
 TO_PERIOD    VARCHAR2 (4),
 STATUS       VARCHAR2 (15),
 BUDGET_TYPE  VARCHAR2 (15),
 CONSTRAINT BDG_DEFINITIONS_PK
 PRIMARY KEY ( M_BUDGET_ID )
  USING INDEX
   TABLESPACE BDG PCTFREE 10
   STORAGE ( INITIAL 65536 ))
  TABLESPACE BDG
  PCTFREE 10
  PCTUSED 40
  INITRANS 1
  MAXTRANS 255
 STORAGE (
  INITIAL 532480
  MINEXTENTS 1
  MAXEXTENTS 2147483645
  FREELISTS 1 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE EXPENSE_CHANGE_HISTORY (
 M_BUDGET_ID     NUMBER (10)   NOT NULL,
 B_ITEM_ID       NUMBER (15)   NOT NULL,
 YEAR            NUMBER        NOT NULL,
 MONTH           VARCHAR2 (2)  NOT NULL,
 TOTAL_QUANTITY  NUMBER,
 TOTAL_MONEY     NUMBER,
 INSERT_DATE     DATE,
 SCNO            NUMBER (5)    NOT NULL,
 TO_FROM         NUMBER        DEFAULT 0 NOT NULL,
 CHANGE          NUMBER        DEFAULT 0 NOT NULL,
 TRANSFER_MONEY  NUMBER        DEFAULT 0 NOT NULL,
 CONV_RATE       NUMBER,
 CONSTRAINT EXPENSE_CHANGE_HISTORY_PK
 PRIMARY KEY ( B_ITEM_ID, SCNO )
  USING INDEX
   TABLESPACE BDG PCTFREE 10
```

```
    STORAGE ( INITIAL 450560 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINES (
  B_ITEM_ID          NUMBER (15)   NOT NULL,
  M_BUDGET_ID        NUMBER (15)   NOT NULL,
  CURRENCY_CODE      VARCHAR2 (5),
  INVENTORY_ITEM_ID  NUMBER (15),
  BUDGET_ID          NUMBER (5)    NOT NULL,
  DESCRIPTION        VARCHAR2 (250),
  REASON             VARCHAR2 (250),
  CC_PREPARE         VARCHAR2 (5),
  CONSTRAINT EXPENSE_LINES_PK
  PRIMARY KEY ( B_ITEM_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 81920 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 8003584
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINE_PERIODS (
  B_ITEM_ID       NUMBER (15)   NOT NULL,
  YEAR            NUMBER        NOT NULL,
  MONTH           VARCHAR2 (2)  NOT NULL,
  REV             NUMBER,
  TOTAL_QUANTITY  NUMBER,
  TOTAL_MONEY     NUMBER,
  SCNO            NUMBER (5),
```

```
  CHANGE         NUMBER,
  CONSTRAINT EXPENSE_LINE_PERIODS_PK
  PRIMARY KEY ( B_ITEM_ID, YEAR, MONTH )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 450560 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINE_TYPES (
  BUDGET_ID    NUMBER (5)    NOT NULL,
  BUDGET       VARCHAR2 (5),
  DESCRIPTION  VARCHAR2 (100),
  DETAIL       VARCHAR2 (1000),
  TYPE_ID      NUMBER,
  BDG_CODE_ID  NUMBER (15),
  CONSTRAINT EXPENSE_LINE_TYPES_PK
  PRIMARY KEY ( BUDGET_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 40960 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE POSITIONS_BUDGET (
  M_BUDGET_ID  NUMBER (15)   NOT NULL,
  YEAR         NUMBER (4),
  CC_ID        NUMBER (3)    NOT NULL,
  POSITION_ID  NUMBER (5)    NOT NULL,
```

```
 MONTH1       NUMBER (5),
 MONTH2       NUMBER (5),
 MONTH3       NUMBER (5),
 MONTH4       NUMBER (5),
 MONTH5       NUMBER (5),
 MONTH6       NUMBER (5),
 MONTH7       NUMBER (5),
 MONTH8       NUMBER (5),
 MONTH9       NUMBER (5),
 MONTH10      NUMBER (5),
 MONTH11      NUMBER (5),
 MONTH12      NUMBER (5),
 CONSTRAINT POSITIONS_BUDGET_PK
 PRIMARY KEY ( M_BUDGET_ID, CC_ID, POSITION_ID )
  USING INDEX
   TABLESPACE BDG PCTFREE 10
   STORAGE ( INITIAL 65536 ))
 TABLESPACE BDG
 PCTFREE 10
 PCTUSED 40
 INITRANS 1
 MAXTRANS 255
STORAGE (
 INITIAL 532480
 MINEXTENTS 1
 MAXEXTENTS 2147483645
 FREELISTS 1 FREELIST GROUPS 1 )
 NOCACHE;

CREATE TABLE POSITION_CC_ASSIGNMENT (
 CC              VARCHAR2 (5)  NOT NULL,
 CC_DESC         VARCHAR2 (100),
 VALID           VARCHAR2 (1)  NOT NULL,
 TIME_SHEET_FILL  VARCHAR2 (1)  NOT NULL,
 CC_ID           NUMBER (3))
 TABLESPACE BDG
 PCTFREE 10
 PCTUSED 40
 INITRANS 1
 MAXTRANS 255
STORAGE (
 INITIAL 532480
 MINEXTENTS 1
 MAXEXTENTS 2147483645
 FREELISTS 1 FREELIST GROUPS 1 )
 NOCACHE;
```

```
CREATE TABLE POSITION_DEFINITIONS (
  POSITION_ID      NUMBER (5)    NOT NULL,
  VALID            VARCHAR2 (1)  NOT NULL,
  TIME_SHEET_FILL  VARCHAR2 (1)  NOT NULL,
  POSITION_NAME    VARCHAR2 (100),
  CONSTRAINT POSITION_DEFINITIONS_PK
  PRIMARY KEY ( POSITION_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE PURCHASE_ORDERS (
  LINE_NUM       NUMBER        NOT NULL,
  B_ITEM_ID      NUMBER        NOT NULL,
  ORDER_DATE     DATE          NOT NULL,
  QUANTITY       NUMBER        NOT NULL,
  EXPECTED_COST  NUMBER        NOT NULL,
  UOM            VARCHAR2 (6)  NOT NULL,
  DESCRIPTION    VARCHAR2 (80),
  CURRENCY       VARCHAR2 (5)  NOT NULL,
  NEED_BY_DATE   DATE,
  REASON         VARCHAR2 (240),
  STOCK_NO       VARCHAR2 (30),
  PRIMARY KEY ( B_ITEM_ID, LINE_NUM )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 16384
   MINEXTENTS 1
   MAXEXTENTS 2147483645
```

```
  FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE WORKHOUR_BUDGET (
 M_BUDGET_ID  NUMBER (15)   NOT NULL,
 YEAR         NUMBER (4),
 CC_ID        NUMBER (3)    NOT NULL,
 POSITION_ID  NUMBER (5)    NOT NULL,
 PROJECT      VARCHAR2 (5)  NOT NULL,
 MONTH1       NUMBER (9,2),
 MONTH2       NUMBER (9,2),
 MONTH3       NUMBER (9,2),
 MONTH4       NUMBER (9,2),
 MONTH5       NUMBER (9,2),
 MONTH6       NUMBER (9,2),
 MONTH7       NUMBER (9,2),
 MONTH8       NUMBER (9,2),
 MONTH9       NUMBER (9,2),
 MONTH10      NUMBER (9,2),
 MONTH11      NUMBER (9,2),
 MONTH12      NUMBER (9,2),
 CONSTRAINT WORKHOUR_BUDGET_PK
 PRIMARY KEY ( M_BUDGET_ID, POSITION_ID, PROJECT )
   USING INDEX
    TABLESPACE BDG PCTFREE 10
    STORAGE ( INITIAL 65536 ))
  TABLESPACE BDG
  PCTFREE 10
  PCTUSED 40
  INITRANS 1
  MAXTRANS 255
 STORAGE (
  INITIAL 532480
  MINEXTENTS 1
  MAXEXTENTS 2147483645
  FREELISTS 1 FREELIST GROUPS 1 )
  NOCACHE;

ALTER TABLE BUDGET_ASSIGNED_COST_CENTERS ADD  CONSTRAINT
BUDGET_ASG_COST_CENTERS_FK
 FOREIGN KEY (M_BUDGET_ID)
 REFERENCES BDG.BUDGET_DEFINITIONS (M_BUDGET_ID) ;

ALTER TABLE EXPENSE_CHANGE_HISTORY ADD  CONSTRAINT
EXPENSE_CHANGE_HISTORY
 FOREIGN KEY (B_ITEM_ID, YEAR, MONTH)
 REFERENCES BDG.EXPENSE_LINE_PERIODS (B_ITEM_ID, YEAR, MONTH) ;
```

```
ALTER TABLE EXPENSE_LINES ADD  CONSTRAINT EXPENSE_LINES_FK
 FOREIGN KEY (M_BUDGET_ID)
  REFERENCES BDG.BUDGET_DEFINITIONS (M_BUDGET_ID) ;

ALTER TABLE EXPENSE_LINES ADD  CONSTRAINT EXPENSE_LINES_FK2
 FOREIGN KEY (BUDGET_ID)
  REFERENCES BDG.EXPENSE_LINE_TYPES (BUDGET_ID) ;

ALTER TABLE EXPENSE_LINE_PERIODS ADD  CONSTRAINT
EXPENSE_LINE_PERIODS_FK
 FOREIGN KEY (B_ITEM_ID)
  REFERENCES BDG.EXPENSE_LINES (B_ITEM_ID) ;

ALTER TABLE POSITIONS_BUDGET ADD  CONSTRAINT
POSITION_DEFINITIONS_FK
 FOREIGN KEY (POSITION_ID)
  REFERENCES BDG.POSITION_DEFINITIONS (POSITION_ID) ;

ALTER TABLE PURCHASE_ORDERS ADD
 FOREIGN KEY (B_ITEM_ID)
  REFERENCES BDG.EXPENSE_LINES (B_ITEM_ID) ;

ALTER TABLE WORKHOUR_BUDGET ADD  CONSTRAINT
WORKHOUR_BUDGET_FK1
 FOREIGN KEY (M_BUDGET_ID)
  REFERENCES BDG.BUDGET_DEFINITIONS (M_BUDGET_ID) ;


CREATE TABLE ANNUAL_WORKHOUR (
  YEAR         NUMBER (4)    NOT NULL,
  MONTH        VARCHAR2 (10)  NOT NULL,
  WORK_DAY     NUMBER (6,2)  NOT NULL,
  WORK_HOUR    NUMBER (6,2)  NOT NULL,
  MONTH_ORDER  NUMBER (2))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;
```

```
CREATE TABLE BUDGET_ASSIGNED_COST_CENTERS (
  M_BUDGET_ID  NUMBER (15)   NOT NULL,
  COST_CODE    VARCHAR2 (4)  NOT NULL,
  OPEN_FLAG    VARCHAR2 (1),
  PRIMARY KEY ( M_BUDGET_ID, COST_CODE )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 40960 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE BUDGET_DEFINITIONS (
  M_BUDGET_ID  NUMBER (15)   NOT NULL,
  CC_PRIMARY   VARCHAR2 (4),
  PROJECT      VARCHAR2 (10),
  REV          VARCHAR2 (5),
  BUDGET_NAME  VARCHAR2 (50),
  FROM_PERIOD  VARCHAR2 (4),
  TO_PERIOD    VARCHAR2 (4),
  STATUS       VARCHAR2 (15),
  BUDGET_TYPE  VARCHAR2 (15),
  ATTRIBUTE1   VARCHAR2 (50),
  ATTRIBUTE2   VARCHAR2 (50),
  ATTRIBUTE3   VARCHAR2 (50),
  ATTRIBUTE4   VARCHAR2 (50),
  ATTRIBUTE5   VARCHAR2 (50),
  CONSTRAINT BDG_DEFINITIONS_PK
  PRIMARY KEY ( M_BUDGET_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
```

```
  MINEXTENTS 1
  MAXEXTENTS 2147483645
  FREELISTS 1 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE EXPENSE_CHANGE_HISTORY (
  M_BUDGET_ID     NUMBER (10)   NOT NULL,
  B_ITEM_ID       NUMBER (15)   NOT NULL,
  YEAR            NUMBER        NOT NULL,
  MONTH           VARCHAR2 (2)  NOT NULL,
  TOTAL_QUANTITY  NUMBER,
  TOTAL_MONEY     NUMBER,
  INSERT_DATE     DATE,
  SCNO            NUMBER (5)    NOT NULL,
  TO_FROM         NUMBER        DEFAULT 0 NOT NULL,
  CHANGE          NUMBER        DEFAULT 0 NOT NULL,
  TRANSFER_MONEY  NUMBER        DEFAULT 0 NOT NULL,
  CONV_RATE       NUMBER,
  ATTRIBUTE1      VARCHAR2 (50),
  ATTRIBUTE2      VARCHAR2 (50),
  ATTRIBUTE3      VARCHAR2 (50),
  ATTRIBUTE4      VARCHAR2 (50),
  ATTRIBUTE5      VARCHAR2 (50),
  CONSTRAINT EXPENSE_CHANGE_HISTORY_PK
  PRIMARY KEY ( B_ITEM_ID, SCNO )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 450560 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 1081344
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINES (
  B_ITEM_ID          NUMBER (15)   NOT NULL,
  M_BUDGET_ID        NUMBER (15)   NOT NULL,
  CURRENCY_CODE      VARCHAR2 (5),
  INVENTORY_ITEM_ID  NUMBER (15),
  BUDGET_ID          NUMBER (5)    NOT NULL,
  DESCRIPTION        VARCHAR2 (250),
```

```
  REASON            VARCHAR2 (250),
  CC_PREPARE        VARCHAR2 (5),
  ATTRIBUTE1        VARCHAR2 (50),
  ATTRIBUTE2        VARCHAR2 (50),
  ATTRIBUTE3        VARCHAR2 (50),
  ATTRIBUTE4        VARCHAR2 (50),
  ATTRIBUTE5        VARCHAR2 (50),
  CONSTRAINT EXPENSE_LINES_PK
  PRIMARY KEY ( B_ITEM_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 81920 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 8003584
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINE_PERIODS (
  B_ITEM_ID       NUMBER (15)   NOT NULL,
  YEAR            NUMBER        NOT NULL,
  MONTH           VARCHAR2 (2)  NOT NULL,
  REV             NUMBER,
  TOTAL_QUANTITY  NUMBER,
  TOTAL_MONEY     NUMBER,
  SCNO            NUMBER (5),
  CHANGE          NUMBER,
  ATTRIBUTE1      VARCHAR2 (50),
  ATTRIBUTE2      VARCHAR2 (50),
  ATTRIBUTE3      VARCHAR2 (50),
  ATTRIBUTE4      VARCHAR2 (50),
  ATTRIBUTE5      VARCHAR2 (50),
  CONSTRAINT EXPENSE_LINE_PERIODS_PK
  PRIMARY KEY ( B_ITEM_ID, YEAR, MONTH )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 450560 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
```

```
  MAXTRANS 255
 STORAGE (
  INITIAL 1081344
  MINEXTENTS 1
  MAXEXTENTS 2147483645
  FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE EXPENSE_LINE_TYPES (
 BUDGET_ID    NUMBER (5)    NOT NULL,
 BUDGET       VARCHAR2 (5),
 DESCRIPTION  VARCHAR2 (100),
 DETAIL       VARCHAR2 (1000),
 TYPE_ID      NUMBER,
 BDG_CODE_ID  NUMBER (15),
 CONSTRAINT EXPENSE_LINE_TYPES_PK
 PRIMARY KEY ( BUDGET_ID )
   USING INDEX
    TABLESPACE BDG PCTFREE 10
    STORAGE ( INITIAL 40960 ))
  TABLESPACE BDG
  PCTFREE 10
  PCTUSED 70
  INITRANS 3
  MAXTRANS 255
 STORAGE (
  INITIAL 1081344
  MINEXTENTS 1
  MAXEXTENTS 2147483645
  FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE POSITIONS_BUDGET (
 M_BUDGET_ID  NUMBER (15)   NOT NULL,
 YEAR         NUMBER (4),
 CC_ID        NUMBER (3)    NOT NULL,
 POSITION_ID  NUMBER (5)    NOT NULL,
 MONTH1       NUMBER (5),
 MONTH2       NUMBER (5),
 MONTH3       NUMBER (5),
 MONTH4       NUMBER (5),
 MONTH5       NUMBER (5),
 MONTH6       NUMBER (5),
 MONTH7       NUMBER (5),
 MONTH8       NUMBER (5),
 MONTH9       NUMBER (5),
 MONTH10      NUMBER (5),
```

```
  MONTH11      NUMBER (5),
  MONTH12      NUMBER (5),
  ATTRIBUTE1   VARCHAR2 (50),
  ATTRIBUTE2   VARCHAR2 (50),
  ATTRIBUTE3   VARCHAR2 (50),
  ATTRIBUTE4   VARCHAR2 (50),
  ATTRIBUTE5   VARCHAR2 (50),
  CONSTRAINT POSITIONS_BUDGET_PK
  PRIMARY KEY ( M_BUDGET_ID, CC_ID, POSITION_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE POSITION_CC_ASSIGNMENT (
  CC              VARCHAR2 (5)  NOT NULL,
  CC_DESC         VARCHAR2 (100),
  VALID           VARCHAR2 (1)  NOT NULL,
  TIME_SHEET_FILL  VARCHAR2 (1)  NOT NULL,
  CC_ID           NUMBER (3))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE POSITION_DEFINITIONS (
  POSITION_ID     NUMBER (5)    NOT NULL,
  VALID           VARCHAR2 (1)  NOT NULL,
  TIME_SHEET_FILL  VARCHAR2 (1)  NOT NULL,
  POSITION_NAME    VARCHAR2 (100),
```

```
  CONSTRAINT POSITION_DEFINITIONS_PK
  PRIMARY KEY ( POSITION_ID )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE PURCHASE_ORDERS (
  LINE_NUM       NUMBER        NOT NULL,
  B_ITEM_ID      NUMBER        NOT NULL,
  ORDER_DATE     DATE          NOT NULL,
  QUANTITY       NUMBER        NOT NULL,
  EXPECTED_COST  NUMBER        NOT NULL,
  UOM            VARCHAR2 (6)  NOT NULL,
  DESCRIPTION    VARCHAR2 (80),
  CURRENCY       VARCHAR2 (5)  NOT NULL,
  NEED_BY_DATE   DATE,
  REASON         VARCHAR2 (240),
  STOCK_NO       VARCHAR2 (30),
  ATTRIBUTE1     VARCHAR2 (50),
  ATTRIBUTE2     VARCHAR2 (50),
  ATTRIBUTE3     VARCHAR2 (50),
  ATTRIBUTE4     VARCHAR2 (50),
  ATTRIBUTE5     VARCHAR2 (50),
  PRIMARY KEY ( B_ITEM_ID, LINE_NUM )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 16384
   MINEXTENTS 1
   MAXEXTENTS 2147483645
```

```
  FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;

CREATE TABLE WORKHOUR_BUDGET (
  M_BUDGET_ID  NUMBER (15)   NOT NULL,
  YEAR         NUMBER (4),
  CC_ID        NUMBER (3)    NOT NULL,
  POSITION_ID  NUMBER (5)    NOT NULL,
  PROJECT      VARCHAR2 (5)  NOT NULL,
  MONTH1       NUMBER (9,2),
  MONTH2       NUMBER (9,2),
  MONTH3       NUMBER (9,2),
  MONTH4       NUMBER (9,2),
  MONTH5       NUMBER (9,2),
  MONTH6       NUMBER (9,2),
  MONTH7       NUMBER (9,2),
  MONTH8       NUMBER (9,2),
  MONTH9       NUMBER (9,2),
  MONTH10      NUMBER (9,2),
  MONTH11      NUMBER (9,2),
  MONTH12      NUMBER (9,2),
  ATTRIBUTE1   VARCHAR2 (50),
  ATTRIBUTE2   VARCHAR2 (50),
  ATTRIBUTE3   VARCHAR2 (50),
  ATTRIBUTE4   VARCHAR2 (50),
  ATTRIBUTE5   VARCHAR2 (50),
  CONSTRAINT WORKHOUR_BUDGET_PK
  PRIMARY KEY ( M_BUDGET_ID, POSITION_ID, PROJECT )
    USING INDEX
     TABLESPACE BDG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE BDG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

ALTER TABLE BDG1.BUDGET_ASSIGNED_COST_CENTERS ADD
CONSTRAINT BUDGET_ASG_COST_CENTERS_FK
 FOREIGN KEY (M_BUDGET_ID)
  REFERENCES BDG1.BUDGET_DEFINITIONS (M_BUDGET_ID) ;
```

```
ALTER TABLE BDG1.EXPENSE_CHANGE_HISTORY ADD  CONSTRAINT
EXPENSE_CHANGE_HISTORY
 FOREIGN KEY (B_ITEM_ID, YEAR, MONTH)
  REFERENCES BDG1.EXPENSE_LINE_PERIODS (B_ITEM_ID, YEAR, MONTH)
;

ALTER TABLE BDG1.EXPENSE_LINES ADD  CONSTRAINT
EXPENSE_LINES_FK
 FOREIGN KEY (M_BUDGET_ID)
  REFERENCES BDG1.BUDGET_DEFINITIONS (M_BUDGET_ID) ;

ALTER TABLE BDG1.EXPENSE_LINES ADD  CONSTRAINT
EXPENSE_LINES_FK2
 FOREIGN KEY (BUDGET_ID)
  REFERENCES BDG1.EXPENSE_LINE_TYPES (BUDGET_ID) ;

ALTER TABLE BDG1.EXPENSE_LINE_PERIODS ADD  CONSTRAINT
EXPENSE_LINE_PERIODS_FK
 FOREIGN KEY (B_ITEM_ID)
  REFERENCES BDG1.EXPENSE_LINES (B_ITEM_ID) ;

ALTER TABLE BDG1.POSITIONS_BUDGET ADD  CONSTRAINT
POSITION_DEFINITIONS_FK
 FOREIGN KEY (POSITION_ID)
  REFERENCES BDG1.POSITION_DEFINITIONS (POSITION_ID) ;

ALTER TABLE BDG1.WORKHOUR_BUDGET ADD  CONSTRAINT
WORKHOUR_BUDGET_FK1
 FOREIGN KEY (M_BUDGET_ID)
  REFERENCES BDG1.BUDGET_DEFINITIONS (M_BUDGET_ID) ;


CREATE TABLE EXPENSE_LINES_CUST (
  B_ITEM_ID        NUMBER (15)   NOT NULL,
  PLACE_OF_EXPENSE  VARCHAR2 (30),
  RESOURCE_TYPE     VARCHAR2 (30),
  CONSTRAINT EXPENSE_LINES_CUST_PK
  PRIMARY KEY ( B_ITEM_ID )
    USING INDEX
     TABLESPACE CUSTOM PCTFREE 10
     STORAGE ( INITIAL 81920 ))
   TABLESPACE CUSTOM
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
```

```
  STORAGE (
   INITIAL 8003584
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE EXPENSE_LINE_PERIODS_CUST (
  B_ITEM_ID         NUMBER (15)   NOT NULL,
  YEAR              NUMBER        NOT NULL,
  MONTH             NUMBER        NOT NULL,
  PAYMENT_DELIVERY  VARCHAR2 (30),
  CONSTRAINT EXPENSE_LINE_PERIODS_CUST_PK
  PRIMARY KEY ( B_ITEM_ID, YEAR, MONTH )
    USING INDEX
     TABLESPACE CUSTOM PCTFREE 10
     STORAGE ( INITIAL 81920 ))
   TABLESPACE CUSTOM
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 8003584
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 3 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE PURCHASE_ORDERS_CUST (
  LINE_NUM            NUMBER        NOT NULL,
  B_ITEM_ID           NUMBER        NOT NULL,
  ALTERNATIVE_VENDORS  VARCHAR2 (30),
  BUILDING_NAME       VARCHAR2 (30),
  PRIMARY KEY ( B_ITEM_ID, LINE_NUM )
    USING INDEX
     TABLESPACE CUSTOM PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE CUSTOM
   PCTFREE 10
   PCTUSED 70
   INITRANS 3
   MAXTRANS 255
  STORAGE (
   INITIAL 16384
   MINEXTENTS 1
   MAXEXTENTS 2147483645
```

```
  FREELISTS 3 FREELIST GROUPS 1 )
  NOCACHE;


CREATE TABLE CUSTOM_ATTRIBUTE_LIST (
 TABLE_NAME           VARCHAR2 (30)  NOT NULL,
 COLUMN_NAME          VARCHAR2 (30)  NOT NULL,
 VALUE_LIST_SET_ID    NUMBER,
 END_USER_COLUMN_NAME  VARCHAR2 (30),
 DESCRIPTION          VARCHAR2 (100),
 ENABLED_FLAG         VARCHAR2 (1),
 REQUIRED_FLAG        VARCHAR2 (1),
 COLUMN_SIZE          NUMBER,
 COLUMN_TYPE          VARCHAR2 (10),
 CONSTRAINT CUSTOM_ATTRIBUTE_LIST_PK
 PRIMARY KEY ( TABLE_NAME, COLUMN_NAME )
  USING INDEX
   TABLESPACE APPMNG PCTFREE 10
   STORAGE ( INITIAL 65536 ))
 TABLESPACE APPMNG
 PCTFREE 10
 PCTUSED 40
 INITRANS 1
 MAXTRANS 255
 STORAGE (
 INITIAL 532480
 MINEXTENTS 1
 MAXEXTENTS 2147483645
 FREELISTS 1 FREELIST GROUPS 1 )
 NOCACHE;

CREATE TABLE CUSTOM_LIST_SET (
 VALUE_LIST_SET_ID  NUMBER       NOT NULL,
 VALUE_SET_NAME     VARCHAR2 (30),
 DESCRIPTION        VARCHAR2 (100),
 TYPE               VARCHAR2 (1),
 CONSTRAINT CUSTOM_LIST_SET_PK
 PRIMARY KEY ( VALUE_LIST_SET_ID )
  USING INDEX
   TABLESPACE APPMNG PCTFREE 10
   STORAGE ( INITIAL 65536 ))
 TABLESPACE APPMNG
 PCTFREE 10
 PCTUSED 40
 INITRANS 1
 MAXTRANS 255
 STORAGE (
```

```
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

CREATE TABLE CUSTOM_LIST_VALUES (
  VALUE_LIST_SET_ID   NUMBER       NOT NULL,
  VALUE_LIST_ID       NUMBER       NOT NULL,
  VALUE_LIST          VARCHAR2 (30),
  VALUE_LIST_MEANING  VARCHAR2 (100),
  DESCRIPTION         VARCHAR2 (100),
  ENABLED_FLAG        VARCHAR2 (1),
  CONSTRAINT CUSTOM_LIST_VALUES_PK
  PRIMARY KEY ( VALUE_LIST_SET_ID, VALUE_LIST_ID )
    USING INDEX
     TABLESPACE APPMNG PCTFREE 10
     STORAGE ( INITIAL 65536 ))
   TABLESPACE APPMNG
   PCTFREE 10
   PCTUSED 40
   INITRANS 1
   MAXTRANS 255
  STORAGE (
   INITIAL 532480
   MINEXTENTS 1
   MAXEXTENTS 2147483645
   FREELISTS 1 FREELIST GROUPS 1 )
   NOCACHE;

ALTER TABLE APPMNG.CUSTOM_ATTRIBUTE_LIST ADD  CONSTRAINT
CUSTOM_ATTRIBUTE_LIST_FK
 FOREIGN KEY (VALUE_LIST_SET_ID)
  REFERENCES APPMNG.CUSTOM_LIST_SET (VALUE_LIST_SET_ID) ;

ALTER TABLE APPMNG.CUSTOM_LIST_VALUES ADD  CONSTRAINT
CUSTOM_LIST_VALUES_FK
 FOREIGN KEY (VALUE_LIST_SET_ID)
  REFERENCES APPMNG.CUSTOM_LIST_SET (VALUE_LIST_SET_ID) ;
```