CMPE 450
SOFTWARE ENGINEERING
NORMALIZED DATABASE DESIGN DOCUMENT

Version 3.1

Project Name	:	Stock Follow Up System
Submitted to Instructor	:	Ayşe Başar Bener
Assistant	:	Gül Çalıklı, Onur Güngör
Submitted by	:	Database Design Group 3

Duygu İşler

Gülnur Derelioğlu

Boğaziçi University
Fall 2006

TABLE of CONTENTS

1. INTRODUCTION

In this document, as the Database Management group of the Stock Follow up System Project, we will first briefly explain our design principles. Then, our database design will be explained in detail including the properties, descriptions of the tables, key fields, the content of the fields, data types and the relations between the tables. Finally, we will end the document by an evaluation of our design from an objective perspective as much as possible.

1.1.Purpose

Our main purposes in this paper are;

- To define database tables according to the requirements of the Stock Follow up System to integrating with existing project SAS (Satın Alma Sistemi).
- This is Database Design Document Version 2.0
- To define fields in tables, their properties and restrictions on them if any exists.
- To explain the design trade-offs while forming tables.
- To define primary keys in tables and foreign keys between them.
- To define standards when naming fields, tables and stored procedures.
- To define the ways for accessing tables that will be used by other teams. In other words, to define stored procedures.
- To give general information about system or software properties that is used in the implementation.
- To define important factors in design approach.

1.2.System Properties

In this project, we will use MySQL to implement tables and procedures. The MySQL® database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used in more than 10 million installations ranging from large corporations to specialized embedded applications on every continent in the world. (Yes, even Antarctica!)

MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, which holds the copyright to most of the codebase. This is similar to the JBoss model and how the Free Software Foundation handles copyright in its projects, and dissimilar to how the Apache project does it, where the software is developed by a public community, and the copyright to the codebase is owned by its individual authors.

The company develops and maintains the system, selling support and service contracts, as well as proprietary-licensed copies of MySQL, and employing people all over the world who collaborate via the Internet. MySQL AB was founded by David Axmark, Allan Larsson, and Michael "Monty" Widenius.

Not only is MySQL the world's most popular open source database, it's also become the database of choice for a new generation of applications built on the LAMP stack (Linux, Apache, MySQL, PHP / Perl / Python.) MySQL runs on more than 20 platforms including Linux, Windows, OS/X, HP-UX, AIX, Netware, giving you the kind of flexibility that puts you in control.

1.3.Design Approach

1.3.1. Investigation

Our design approach is slightly changed because we distinguished later that our Stock Follow Up System is not a new design or Software it should maintain with last year project SAS (Satın Alma Sistemi). So we begin to design everything from beginning. With this approach it is not version 2.0 it is just a new design.

1.3.2. Scope

We tried to set realistic and affordable goals. Our main focus is functionality and compatibility with existing database SAS-Database. We concentrate mainly on completing database operations on time and correctly. After that performance issues are considered.

1.3.3. Design

Our main design goals, although they are certainly conflicting, are make requirement tables harmony with existing database SAS. We tried to take a balanced approach between those alternatives as much as possible. In order to improve space efficiency we sometimes divided the data into several tables, separating elements having different information. This enables us to save space as we do not have to keep null fields for elements of the system that do not bear the given information in a column. However, sometimes we decided that it would be better to have some space inefficiency rather than doing many join operations. These will be explained in more detail in the trade-offs and design evaluation section; overall, the main design issue for us was to balance space inefficiency versus the overhead of join operations.

1.3.4. Maintenance

Once the system was in place and stable, requests were implemented carefully in order not to disturb the operation of the system. Especially changes on the database, addition of new tables or especially the modification of existing ones, required considerable amount of maintenance work on the SPs because of the inflexibility introduced by MySQL regarding returning multiple columns. Sometimes types had to be defined from scratch. Also as the number of SPs increased, it became harder to spot errors.

1.4.Trade-offs

As mentioned above, the most important design choice we considered was whether to allow some space inefficiency in exchange for a performance increase gained by fewer join operations. During design and implementation, we usually favoured preserving space efficiency. We implemented extra tables to preserve space efficiency. This caused maintenance to be a problem because there were lots of tables. However, performance increased due to less number of rows.

Second, working with existing database and modification or addition to that database required an extra time and work to handle it. Because firstly we must understand SAS database than we could add extra table which needed.

A detailed evaluation of the database design is available in the design evaluation section.

1.5.Interface Design Standards

Database naming standards

- Table names must start with uppercase.
- All table names consisting of Upper cases. Ex: GENEL_DEPO_GIRIS
- Underscores will be used to separate meaningful words or phrases.
- Table column names must start with uppercase. Ex : Miktar
- If table column names consist of more than one word all word must begin with uppercase and between them there will be no space. Ex : UrunNo

All standards mentioned above are defined by our Design Group. In this document and also in our database implementation we use these standards.

2. TABLES

2.1.GENEL_DEPO_GIRIS

DepoGirisNo	UrunNo	TalepEdenBirimNo	RafNo	Miktar	FaturaNo	FaturaTarihi	Onaylandimi

2.1.1. Purpose

The main aim to create this table is to hold the list of all purchased goods that came to the depot and approved by the clerk. Table is updated if the goods are for administrative departments.

Our main purpose of creating this table is to define users' type numerical and seperated from other users' information thus we scan specify each user types.

2.1.2. Key Fields

“DepoGirisNo” field is unique for each record and the primary key field of the GENEL_DEPO_GIRIS table.

2.1.3. Referential Integrity Constraints

“UrunNo” field is the foreign key referencing the “UrunNo” field of the URUN table.
“TalepEdenBirimNo” field is the foreign key referencing the “BirimNo” field of the BIRIM table.
“RafNo” field is the foreign key referencing the “RafNo” field of the GENEL_DEPO_RAF table.
“FaturaNo” field is the foreign key referencing the “FaturaNo” field of the ISTEK_FISI table.

2.2.GENEL_DEPO_HARCAMA

DepoHarcamaNo	UrunNo	TalepEdenBirimNo	TalepEdenUstBirimNo	RafNo	Miktar	HarcamaTarihi	Aciklama

2.2.1. Purpose

The main aim to create this table is to hold the list of all goods that are removed from the general depot.

2.2.2. Key Fields

"DepoHarcamaNo" field is unique for each record and the primary key field of the GENEL_DEPO_HARCAMA table.

2.2.3. Referential Integrity Constraints

"UrunNo" field is the foreign key referencing the "UrunNo" field of the URUN table.

"TalepEdenBirimNo" field is the foreign key referencing the "BirimNo" field of the BIRIM table.

"RafNo" field is the foreign key referencing the "RafNo" field of the GENEL_DEPO_RAF table.

"TalepEdenUstBirimNo" field is the foreign key referencing the "UstBirimNo" field of the UST_BIRIM table.

2.3.GENEL_DEPO_RAF

RafNo#	Aciklama	Bosmu

2.3.1. Purpose

The main aim to create this table is to hold the list of shelfs in the general depot and either they are empty or not.

2.3.2) Key Fields :

“RafNo” field is unique for each record and the primary key field of the GENEL_DEPO_RAF table

2.3.3) Referential Integrity Constraints :

There are no referential integrity constraints

2.4.GENEL_DEPO_URUN_MIKTAR

UrunNo	AltLimit	Mevcut	Aciklama

2.4.1. Purpose

The main aim to create this table is to hold the list of all goods according to their types, quantities in the general depot and their allowed minimum numbers since the system will alert the user when a specific good decreases below a certain number.

2.4.2. Key Fields

“UrunNo” field is unique for each record and the primary key field of the GENEL_DEPO_URUN_MIKTAR table

2.4.3. Referential Integrity Constraints

"UrunNo" field is the foreign key referencing the "UrunNo" field of the URUN table.

2.5.BIRIM_ DEPO_GIRIS

<u>BirimNo</u>	<u>DepoGirisNo</u>	UrunNo	Miktar	FaturaNo	FaturaTarihi	Aciklama

2.5.1. Purpose

The main aim to create BIRIM_DEPO_GIRIS table is to hold the list of all purchased goods that came to the each department's depot and approved by the departmental clerk. Table is updated if the goods are for administrative and academic departments.

2.5.2. Key Fields

"BirimNo" field and DepoGirisNo field are unique only together and the multiple primary key field of the BIRIM_DEPO_GIRIS table.

2.5.3. Referential Integrity Constraints

"BirimNo" field is the foreign key referencing the "BirimNo" field of the BIRIM table.

"UrunNo" field is the foreign key referencing the UrunNo field of the URUN table.

"FaturaNo" field is the foreign key referencing the FaturaNo field of the ISTEK_FISI table.

2.6.BIRIM_ DEPO_HARCAMA

<u>DepoHarcamaNo</u>	UstBirimNo	<u>BirimNo</u>	UrunNo	Miktar	HarcamaTarihi	Aciklama

2.6.1. Purpose

The main aim to create BIRIM_DEPO_HARCAMA table is to hold the list of all goods that are spent from departments' depot.

2.6.2. Key Fields

"BirimNo" field and "DepoHarcamaNo" field are unique only together and the multiple primary key field of the BIRIM_DEPO_HARCAMA table.

2.6.3. Referential Integrity Constraints

"BirimNo" field is the foreign key referencing the "BirimNo" field of the BIRIM table.

"UrunNo" field is the foreign key referencing the "UrunNo" field of the URUN table.

"UstBirimNo" field is the foreign key referencing the “UstBirimNo" field of the UST_BIRIM table.

2.7.BIRIM_DEPO_URUN_MIKTAR

BirimNo	UrunNo	AltLimit	Mevcut

2.7.1. Purpose

The main aim to create BIRIM_DEPO_URUN_MIKTAR table is to hold the list of all goods that are used in the each administrative and academic department by that department staff.

2.7.2. Key Fields

"UrunNo" and "BirimNo” field are unique and the primary key field of the BIRIM_DEPO_URUN_MIKTAR table.

2.7.3. Referential Integrity Constraints

"UrunNo" field is the foreign key referencing the "UrunNo" field of the URUN table.

"BirimNo" field is the foreign key referencing the "BirimNo" field of the BIRIM table.

2.8.BEKLEYEN_URUNLER

FaturaNo	UrunNo	TalepEden BirimNo	TalepEdenUst BirimNo	FaturaTarihi	Miktar	Aciklama

2.8.1. Purpose

The main aim to create BEKLEYEN_URUNLER table is to hold the list of all goods that are not approved yet in the each administrative and academic department by that department's staff.

2.8.2. Key Fields

"FaturaNo" and "UrunNo" and "TalepEdenBirimNo" field are unique and the primary key field of the BEKLEYEN_URUNLER table.

2.8.3. Referential Integrity Constraints

"UrunNo" field is the foreign key referencing the "UrunNo" field of the URUN table.

"TalepEdenBirimNo" field is the foreign key referencing the "BirimNo" field of the BIRIM table.

"TalepEdenUstBirimNo" field is the foreign key referencing the "UstBirimNo" field of the UST_BIRIM table.

"FaturaNo" field is the foreign key referencing the "FaturaNo" field of the ISTEK_FISI table.

3. STORED PROCEDURES

Stored procedures are included to the database in order to ensure data integrity. Furthermore, these stored procedures are invoked from the client by a remote procedure calls.

AddStockBD_sp: Adds new item into departmental depot and increase the total number of the corresponding item in the storage.

AddStockGD_sp: Adds new item into general depot and increase the total number of the corresponding item in the storage.

AkademikDepoSorgusuKontrol_sp: Displays the storage information and item transaction in the departmental depot.

AktarilmamisUrunleriAl_sp: Returns the items that are already purchased but still has not been inserted into general depot.

AktarilmamisUrunuAktar_sp: Adds items into the general storage that are came to the general depot.

BirimDepoGoster_sp: Returns items stored in the corresponding departmental depot.

BirimDepoMevcut_sp: Returns existing items and the number of item stored in the corresponding departmental depot.

DepoCikisDDC_sp: This stored procedure inserts new data to BIRIM_DEPO_HARCAMA table.

DepoCikisGDCtoBDG_sp: Transfers input item from general depot to departmental depot given by the parameter.

DepoCikisGDC_sp: This stored procedure inserts new data to GENEL_DEPO_HARCAMA table.

DepoGirisDDC_sp: Adds input item into the given departmental depot.

GDepoRafGetir_sp: Displays the shelves`` number existing in the departmental depot.

GenelDepoGoster_sp: Displays the existing items in the general depot.

GenelDepoHarcama_sp: Consumes given item from general depot by the given amount.

GenelDepoMevcut_sp: Displays the existing items and the amount in the general depot

GetAdLimit_sp: This stored procedure returns the limit value of the goods listed in create BIRIM_DEPO_URUN_MIKTAR table.

GetAllBirimAdi_sp: Returns all department name in the system.

GetAllBirimNoVeAdi_sp: Returns all department name and their number in the system.

GetAllGrupAdi_sp: Returns all grup name in the system.

getAllGrupNoVeBirimAdi_sp: Returns all grup name and their number in the system.

GetAllUrunAdi_sp: Returns all item name in the system.

GetAllUrunNoVeAdi_sp: Returns all item name and their number in the system.

GetAllUrunNo_sp: Returns all item number in the system.

GetBdKalanID_sp: Returns the total number of the given product in the departmental storage.

GetBirimNo_sp: Returns the number of the department that name is given as parameter.

GetDepartments_sp: Returns the number of the grup that name is given as parameter.

getDepForDekan_sp: Returns the department related to given Dekan.

getDepNamesForDekan_sp: Returns the name of the department related to given Dekan.

getDepNoForDekan_sp: Returns the number of the department related to given Dekan.

GetExistingItemsAD_sp: returns the existing product and their amount that is stored in the given departmental depot.

GetExistingItemsGD_sp: returns the existing product and their amount that is stored in the given general depot.

GetGdLimit_sp: This stored procedure returns the limit value of the goods listed in create GENEL_DEPO_URUN_MIKTAR table.

GetGdKalanID_sp: Returns the total number of the given product in the general storage.

GetGdLimit_sp: Displays the minimum amount that can be stored in the general depot.

GetGroupName_sp: Displays the name of the given group in the system.
returns the number of the group corresponding the group name given as parameter.

getGroupNoByBirimIsmi_sp: Returns the number of the group corresponding to the department given as parameter.

GetGroups_sp: Displays the groups` information in the system.

GetGrupNo_sp: Returns the number of the group corresponding the group name given as parameter.

GetUrunNo_sp: This stored procedure returns the id of the good according to given good name by selecting appropriate values from URUN table.

LimitiAsanUrunSayisiAD_sp: Returns items that the departmental storage consists more then the minimum limit.

LimitiAsanUrunSayisiGD_sp: Returns items that the general storage consists more then the minimum limit.

LimitUrunAD_sp: Returns items in the departmental depot that has less then the minimum limit.

LimitUrunGD_sp : Returns items in the general depot that has less then the minimum limit.

ListPendingItems_sp: This stored procedure lists the goods which are entered to depots but not approved yet by selecting from BEKLEYEN_URUNLER table.

SetAdLimit_sp: This stored procedure sets a limit for the goods stored in BIRIM_DEPO_URUN_MIKTAR table or it inserts new data (good) to this table with given limit.

SetGdLimit_sp: This stored procedure sets a limit for the goods stored in GENEL_DEPO_URUN_MIKTAR table or it inserts new data (good) to this table with given limit.

SetPendingItems_sp: This stored procedure inserts new goods, which are entered to general depot but not approved yet, to BEKLEYEN_URUNLER table.

Set_PendingToStock_sp: This stored procedure transfers the approved items from BEKLEYEN_URUNLER table to GENEL_DEPO_GIRIS table.

SpendStockGDC_sp: This stored procedure decrements the value of a good in GENEL_DEPO_URUN_MIKTAR table, when the good is inserted to GENEL_DEPO_HARCAMA table.

SpendStock_sp: This stored procedure decrements the value of a good in BIRIM_DEPO_URUN_MIKTAR table, when the good is inserted to BIRIM_DEPO_HARCAMA table.

TumAkademikBirimAdiGoster_sp: Returns all academic departments.

TumAkademikBirimNoGoster_sp: This stored procedure returns all academic departments numbers.

TumAkademikGrupAdiGoster_sp: This stored procedure returns all academic departments names.

TumAkademikGrupGoster_sp: This stored procedure returns all academic groups.

TumAkademikGrupNoGoster_sp: This stored procedure returns all academic groups numbers.

TumAkademikUstBirimAdiGoster_sp: This stored procedure returns all academic high level departments names.

TumAkademikUstBirimNoGoster_sp: This stored procedure returns all academic high level departments numbers.

TumIdariBirimAdiGoster_sp: This stored procedure returns all managerial department names.

TumIdariBirimNoGoster_sp: This stored procedure returns all managerial department numbers.

TumIdariGrupAdiGoster_sp: This stored procedure returns all managerial groups names.

TumIdariGrupGoster_sp: This stored procedure returns all managerial groups.

TumIdariGrupNoGoster_sp: This stored procedure returns all managerial department numbers.

UrunAktar_sp: This stored procedure adds approved items into general depot.

4. DESIGN EVALUATION

This section aims to briefly evaluate our database design. We will try to be as objective as possible; we will talk about both weaknesses and strengths of our design.

First of all, we foresee that the main agents in this Stock Follow Up System or SAS are the users. Users have common information, which we store in user, as well as specific information according to their roles in the system such as manager, rector, staff etc. This specific information is kept in the appropriate table. This prevents data duplication and space inefficiency, however to retrieve all information about a user, join operations must be done. This is a small trade off compared to the inefficiency that would be introduced otherwise.

It is obvious that a user can have multiple roles, to allow such a design, role-user pairs are also kept in a separate table. This table tells with which table user is to be joined to retrieve info about that user. This system can also be extended to include a role based security model. At the beginning of the project, a user based security model was planned. However, it is difficult to determine each task in the system.

Also because it is a stock follow up system goods are as important as users. Especially due to Depot there will be huge density from and to Depot. So information that kept in the related table will change rapid.

We evaluate, comprehend, analyze the existing database and according to Requirement Analysis Group's Report we add tables to existing database.

4. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

CmpE	: Computer Engineering
RAD	: Requirements Analysis Document
Email	: Electronic Mail
ID	: Identification
SDD	: System Design Document
ODD	: Object Design Document
GUI	: Graphical User Interface
GDC	: General Depot Clerk
DDC	: Departmental Depot Clerks
DM	: Department Manager
GD	: General Depot
DD	: Departmental Depot

5. CONCLUSION

Our main goal was to implement tables that are missing or incomplete in previous working database SAS. So we begin to analyze SAS Database, work with that database to understand that database after that we begin to analyze RAD V2.0 and create tables those needed.

But the request-implement-response cycle did not work as planned. Although from the beginning told us that we implement and create a new system database, after one month and also after our V1.0 we became aware of that we should work with existing system. Of course this situation get tired us.

Overall, we were satisfied by the final design. Although it is a one term course project, it is not much different then a real application database. It is rather large, includes complex stored procedures and does not include data repetitions. We propose some enhancements to our design in the suggestion section.

6. REFERENCES

- Requirement Analysis Group Document V2.0
- Cmpe 321 Introduction to Database Systems Lecture Notes
- http://www.en.wikipedia.org/wiki/MySQL
- www.thinkdesign.com/approach
- www.cs.pitt.edu/~chang/156/11funcdep.html
- www.faqs.org/docs/ppbook/c208.htm
- www.geocities.com/mnhashmi/triggers.html
- www.datawarehouse.com/iknowledge/articles/article.cfm?ContentID=1846
- "Standards Document" released by Test and Security Group
- MySql Official Documentation:

  http://www.mysql.com/

7. GLOSSARY

Backup: This is an operation of storing current information and data of a database for further uses in a file.

Database: A database consists of some collection of persistent data that is used by the application systems of some given enterprise.

Database Management: This module includes the physical database design, and the necessary triggers and stored procedures to enforce data integrity. Since many different components will access the same database, it should be capable of meeting the needs and requirements of all these components without data duplication.

Database Management System (DBMS): is the software that handles all access to the database.

Foreign Key: A foreign key in one table is a set of one or more attributes whose values match the values, which exist in the primary key attributes of a related table.

Primary Key: The primary key of a table is the one or more attributes, which are used to distinguish every row of the table from every other row of that same table.

Normal Form: A relation is said to be in a particular normal form if it satisfies a certain prescribed set of conditions.

Determinant: A field, which functionally determines the other fields' values.

Stored Procedure: is a precompiled program that is stored at the server site. It is invoked from the client by a remote procedure call.

Functional Dependency: A functional dependency is a constraint between 2 sets of attributes from the database for each values of the first set there is a unique value of the second set.

Non-key Attribute: any attribute that does not participate in the primary key of the relation concerned.

Relational Database: A database consisting of more than one table. In a multi-table database, you not only need to define the structure of each table, you also need to define the relationships between each table in order to link those tables correctly.

Referential Integrity: Referential integrity is a concept that requires that a foreign key should be either wholly null or its value should match the value of the primary key in some tuple in the parent relation.

Restore: This is a recreation of a database from a backup file.

MySQL: MySQL is a multithreaded, multi-user, SQL Database Management System (DBMS) with more than six million installations. MySQL AB makes MySQL Server available as free software under the GNU General Public License (GPL), but they also offer the MySQL Enterprise subscription offering for business users and dual-license it under traditional proprietary licensing arrangements for cases where the intended use is incompatible with the GPL.

SQL: Structured Query Language (pronounced *sequel* in the US; *ess-queue-ell* elsewhere). A computer language designed to organize and simplify the process of getting information out of a database in a usable form, and also used to reorganize data within databases. SQL is most often used on larger databases on minicomputers, mainframes and corporate servers.

Stored Procedure: is a precompiled program that is stored at the server site. It is invoked from the client by a remote procedure call (RPC).

Trade-off: the exchange of one thing for another of more or less equal value, esp. to effect a compromise.