LIGHTSPEED: A BOOKING APPLICATION FOR TRAVEL AGENCIES

Steffen Breedijk, Ellert Colijn, Amelia Ibrahim, Mariëlle van de Pol

Project Databases 2004-2005, University of Amsterdam
steffen@balai.nl, e.colijn@hccnet.nl, a.ibrahim@student.uva.nl, mpol@science.uva.nl

This document presents an overview of the design and development process of the LIGHTSPEED booking application for travel agencies. A brief review of database history and design on databases is provided followed by the conceptual and physical implementation of the database. The GUI for the application has been written in JAVA with JBuilder.

Introduction

Today, database management systems (DBMS) are used in almost every company and organization in society. In the highly dynamic and competitive travel industry there is need for ICT solutions. The same requirements follow for the travel agency in this project, but at this agency it's only possible to book by phone or at the office. Because of the recent competition of on-line booking systems on the Internet, the automation system designed for this agency should be flexible and cost effective. The goal of this project is to design, build and implement a centralized relational database application for a travel agency. The system must be able store and retrieve information related to different pre-defined holiday packages.

De database design and development for this project takes place at two levels: the conceptual level and the physical level. The first part of this paper will discuss the design at conceptual level. In this section, the purpose of the system and the required functionalities will be described. Some background information about the development and architecture of databases in general is indispensable. The second part of this paper describes the physical level, how the conceptual model of the system has been implemented.

Part One - Conceptual Level

Architecture and Development

Architecture

The development of databases has evolved constantly in the past fifty years. The most simple en oldest system is a flat-file system. The database management systems evolved from this base system is relational, object-oriented or object-relational. The underlying architecture for these systems can be divided in three types. The first is a centralized database system, which is a stand-alone system. The second type is a distributed database system. This system is network-based and makes it possible to have geographical diffusion of sites. Third system type is a federated database system. This system is also network-based and very useful to support the heterogeneity and autonomy of sites [2]. The environment of the database also has evolved from a mainframe-based to a client server environment. The client server makes it possible to operate in a network environment. In this project it was obliged to design and develop a centralized relational database. A network environment is present at the travel agency. To support this there is chosen for client server architecture, because of the disadvantages a file server has. The specific type is database server architecture. Because of the early live stage of the application, the architecture is two-tier. There are not so many users at this moment and to make application server is too complex at this stage of the development process. The first tier is the client with the tasks, user interface and main business and processing logic. The second tier is the database server with the tasks of server side validation en database access.

Recent Developments

The latest development in the area of database research is the emerging of Collaborative Networks (CN). By the easily available and accessible communication networks worldwide the different forms of web-based collaboration are configured and growing. Web-based mechanisms are used to deal with the need to operate within distributed and collaborative environments. Building new support systems on top of the web for CN is a new challenge for database researchers. Some examples of CN's are Virtual Organizations (VO), Virtual Communities (VC) and Virtual Laboratories (VL). The design and development of VO's, VC's and VL's is still an open area which researchers are trying to tackle [4].

Conceptual design

The conceptual design represents informal requirements of the application in terms of a conceptual schema that refers to a conceptual data model. In this section the users and the requirements of the database will be discussed first. Second, the ER data model, the constraints, keys and functional dependencies will be explained. Third, the overall logical structure from the database is shown graphically by means

of an ER diagram. A relational schema diagram from the database is the fourth topic in this section. Finally, the security and authorization issues concerning the database will be discussed.

Users

The users of the DBMS are the employees of the travel agency. These are called clerks and their manager.

Requirements

The basis of the database consists of a module, called a service. Clients can book a package, a single service, or a combination of services. Packages are predefined combinations of several services. It is possible to add, delete and modify the services modules at any time.

The functional requirements that follow from the above are the following. There must be a functional separation between booking and creation of the associated packages.
While making a booking a Clerk/manager should have a view of all the predefined packages in the database. He must be able to interact with the application in order to create a deal for the customer, based on the service selection or a package. The selected packages for a deal are based on selected properties, like destination, accommodation type etc. It must be possible to book separately an accommodation, transportation and activity.

The manager is the one who predefines the packages which can be booked by customers. He is the one to moderate the database; to add, update, and delete holiday packages and services to the database system. He is authorized to add a package with discount in the system

ER data model

The following entity sets and relationship sets can be identified in the E-R data model for the travel agency. The entity sets in the database are: customer, deal, package, service, accommodation, transportation, activity. The relationship sets in the database are: has, part of, delivered on.

Constraints

The ER enterprise schema has constraints to which the contents of the database must conform. These are cardinalities and participation constraints. In the schema for the travel agency most cardinalities are many to many, except the relation between customer and deal. This relation is a one to many. This means that one customer can have zero or more deals, but a deal can only belong to one customer. Every package entity is related to at least one service trough the ‘part of’ relationship. This also goes for the deal. Every deal is related to at least one service also through a ‘part of’

relationship. Therefore the participation of package and deal in both the relationships sets ‘part of’ is total.

Primary and Foreign keys

To specify how the entities in the database can be distinguished from each other, the difference among them is expressed by their attributes. A key makes it possible to identify a set of attributes that adequate distinguish entities from each other. The same is true for the relationships in the database. The most important keys are the primary key and the foreign key. The primary key is the unique identifying attribute of an entity. In the E-R diagram the attributes that are a primary key are underlined. A foreign key is an attribute of a relationship that is the primary key of another entity. The foreign keys will be shown in the database schema in the next paragraph. The attribute ‘price’ from the entity package and the attribute ‘total price’ from the entity deal are derived attributes. They have dashed ellipses.

Functional Dependencies

The set of relation schemas makes it possible to store information without unnecessary redundancy and to retrieve information easily. To avoid anomalies the schemas must be in an appropriate *normal* form. The process of normalization decomposes the relations according to functional dependencies. In this database the relation schemas are in fourth normal form. Each record of the file consists of one main key, and some mutually independent field values. This means that the field values are not multi-valued, not partially dependent and not transitively dependent on the primary key. For the fifth normal form there must be no loss of decomposition.

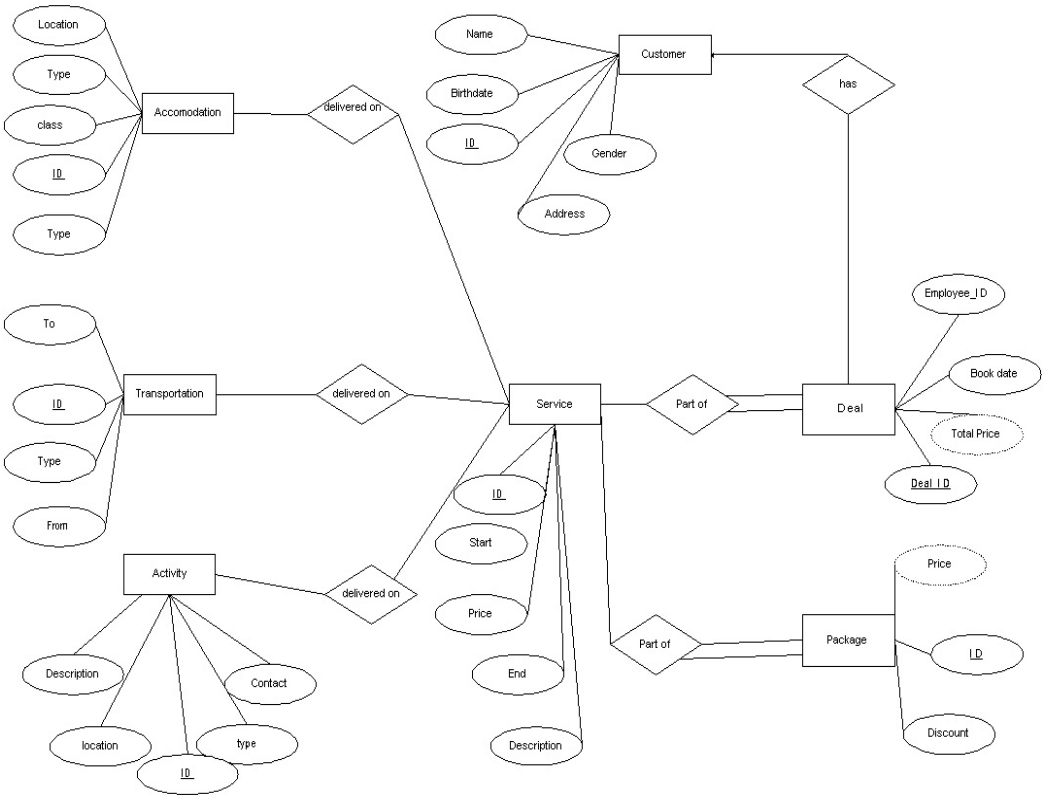


Figure 1 ER Diagram

Relational Database

The ER diagram above is the starting point for making a relational database for the travel agency. Based on the entities and relationships in the ER diagram a database schema can be made. This database schema gives a logical structure of the database. One of the tables of the database schema is shown. The other tables can be found in the appendix.

```
CREATE TABLE `accomodation` (
  `acc_id` int(11) NOT NULL auto_increment,
  `acc_name` varchar(64) NOT NULL default '',
  `acc_location` varchar(64) NOT NULL default '',
  `acc_type` varchar(64) NOT NULL default '',
  `acc_class` int(11) NOT NULL default '0',
  PRIMARY KEY  (`acc_id`)
)
```

The database schema with the primary keys and foreign keys is shown by the diagram below.

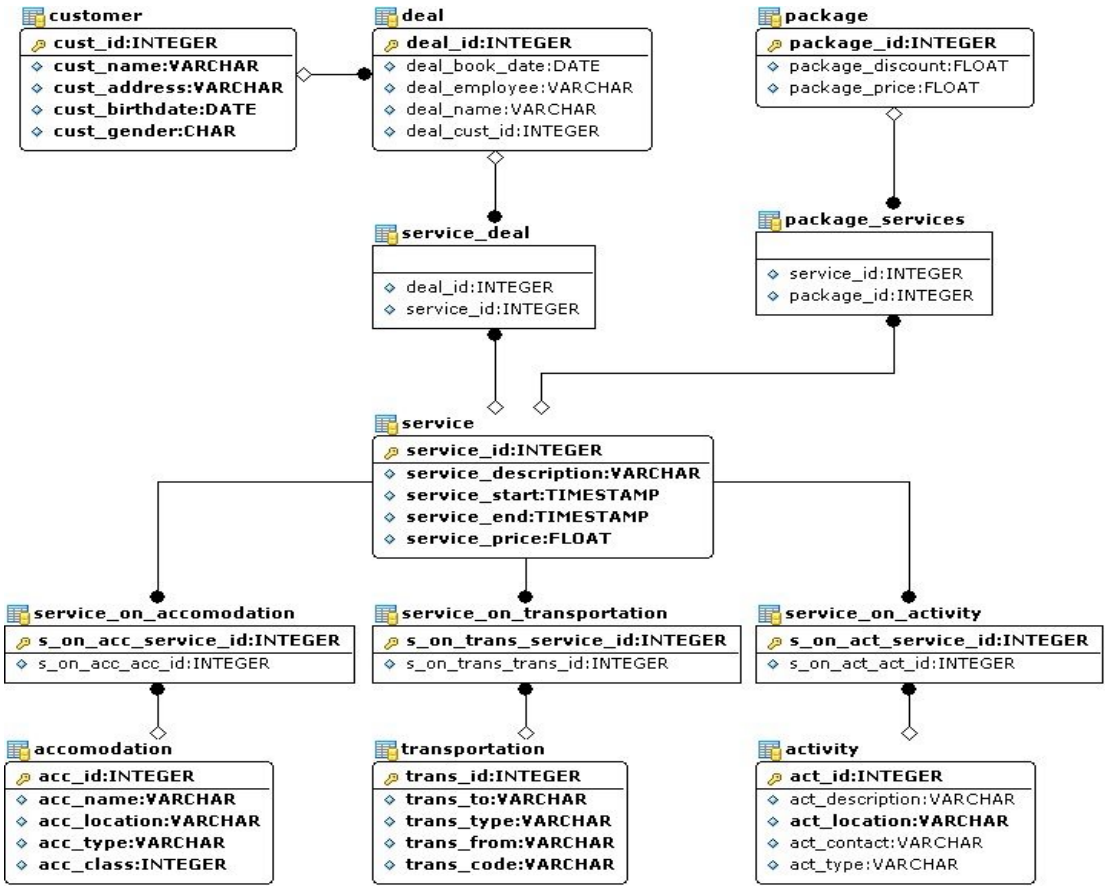


Figure 2: schema diagram

Security and Authorization

The client server architecture for the database has a network environment. For this network environment there are security issues on the levels of the network, operating system, web server, database, and data communication. First, at the network level important security issues consists of create an account with a user name and a password for each user, make sure the database server is away from other business systems, minimize the sharing of hard disks among the servers. Second, the operating system level is discussed. The most important security issues on this level involve unauthorized activity and unauthorized access. This can be addressed by using an Intrusion Detection System and disabling non-required services. Third, by the security issues of the web server attention must be given here to the following: restrict the number of users on the web server, restrict the access, and remove unneeded programs that load automatically when setting up the server. The fourth level is the database. The security issues on this level are all involved with the restriction of the access and manipulation rights of the users of the database. The most important are logins and passwords checks, creating user roles and views. On the level of the data communication the security issues exists of encryption and decryption processes, public and private keys, digital signature and virtual private network (VPN) [1]. Because of the nature of this database project it is not realistic to implement all the security and authorization issues described. The most important thing for this project was creating views to protect the consistency of the database. Users can be given authorization on views, without being given any authorization on the relations used in the view definition. Ability of views to hide data serves both to simplify usage of the system and to enhance security by allowing users access only to data they need for their job. The other security issues must be seen as a guideline for further development in the future [3].

Part Two Implementation

Development

Development environment

Due to the relative high learning curve for the Eclipse IDE, Borland JBuilder has been used for the implementation of the client.

Database choice

Initially, a MaxDB database was provided for the implementation of the Travel Agency project. Due to memory limitations of the server, connectivity problems and server stability the database schemas have been transferred to a MySQL 4.x database. Because this version does not support views MySQL 5.0 has been set up on a separate server. To ensure compatibility with both MySQL versions the views have not been actively used in the client . They are there just to demonstrate the usefulness of views for easy access of the data. Despite the fact that views are officially 'read-only', it can be shown that updates on a view are permitted in some circumstances.

Java code

To make an application interface the following JAVA code is developed. Here are some structures of the JAVA code.

Initiate database

```
Database database1 = new Database();
Connection database1 = DriverManager.getConnection
("jdbc:mysql://localhost/team5 user=root password=")
```

Next sample uses the data from the accommodation table in our MySQL-database.

Creating a dataset for:

```
accomodation.setQuery (new
com.borland.dx.sql.dataset.QueryDescriptor(
        database1, "SELECT
accomodation.acc_id,accomodation.acc_name,accomodation.acc_loc
ation,accomodation.acc_type,acco" +
        "modation.acc_class FROM team5.accomodation", null,
true, Load.ALL));
```

Explanation
We created a dataset for java. We make a selection of de items: acc_id , acc_name, acc_location en acc_type. This dataset is called “acco”.
For all data in the database it is possible to create a dataset like the above.

The graphical user interface

On the screen we create a text field:

Initialize
```
JdbTextField jdbTextField16 = new JdbTextField();

jdbTextField16.setFont(new java.awt.Font("Dialog", Font.PLAIN,
14));
```

\font and font size are defined.

```
 jdbTextField16.setPreferredSize(new Dimension(100, 25));
```
\size of the textfield is set

```
jdbTextField16.setText("Prijs");
```
\a name is given to the field for faster and easier refering.

```
 jdbTextField16.setColumnName("service_price");
```
\referring to the database column name.

```
 jdbTextField16.setDataSet(Accomodation);
```
/using the dataset accommodation.

Database updating using a jdbc toolbar:

```
JdbNavToolBar jdbNavToolBar1 = new JdbNavToolBar();
jdbNavToolBar1.setDataSet(accomodation);
```

uses internal jdbNavToolbar class for updating and inserting data.

Graphical user interface

Most users interact with databases using form interfaces with graphical interaction facilities. Such facilities are provided by GUI (Graphical user interfaces). GUI is a program interface that uses a computer's graphics capabilities to make the program easier to use. Graphical interfaces use a pointing device to select objects, including icons, menus, text boxes, etc. A GUI includes standard formats for representing text and graphics. With use of form interfaces, which provide mechanisms to check the correctness of user input, and automatically fills in fields we can connect a database to such interfaces. Forms and GUI allow users to perform some operations (enter data, search, retrieve etc.) or enter values that complete pre-defined queries; GUI provide an easy-to-use way to interact with a database system.
While designing an interface it is very important to design an interface which is user friendly for the end users, and is efficient in use in order to accomplish the task which the user has to employ. In order to make an interface user friendly one should normally take a usability test with the end users. The usability is the quantitative and qualitative measure to which a user interface can achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [1]. For this

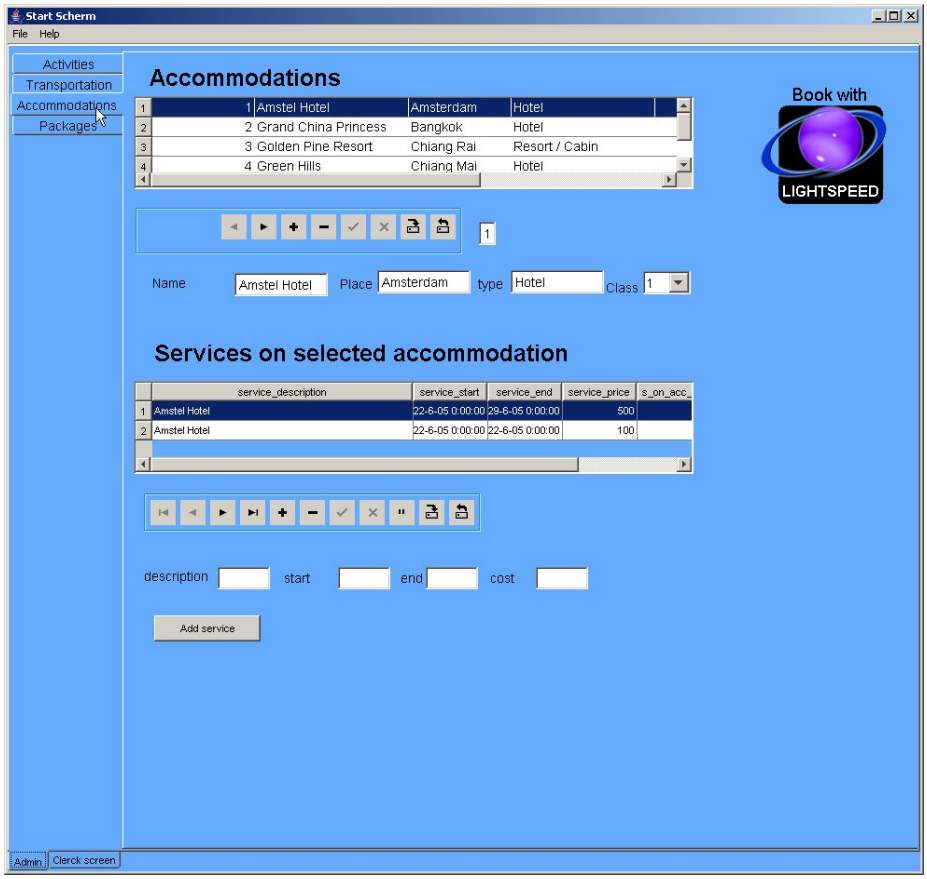
assignment we do not have enough time to evaluate the interface nor to do a usability test. Keeping in mind that the interface should be effective en efficient to work with we designed the further shown interface for the managers and the clerks.
For designing an interface for this assignment we have chosen for an interface which makes it easy for a clerk to make bookings for the customers. We also designed some tabs for the manager who can easily moderate the database by using the GUI. In the next section we will give some screenshots of the designed GUI with a scenario.

Results: Scenarios and Screenshots

Administration-related scenarios

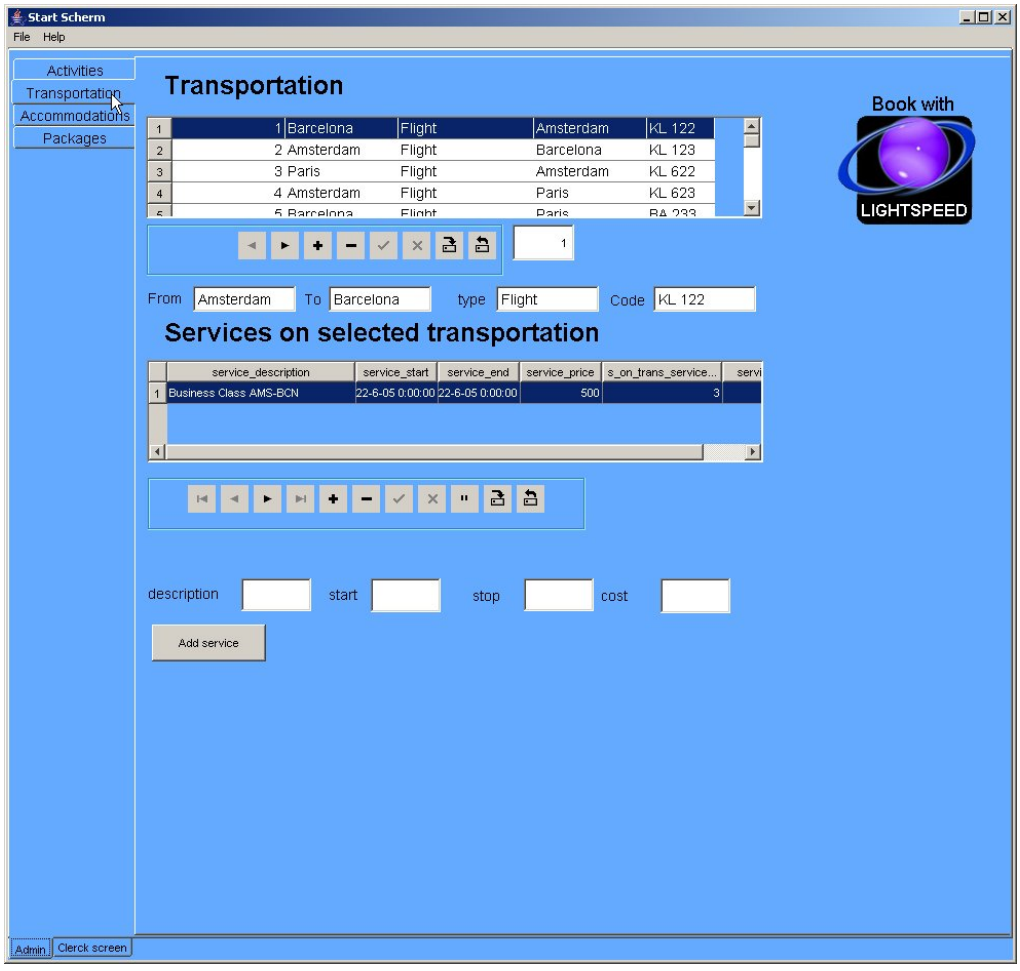
Scenario 1: Adding facilities to the database

Manager wants to add a new Hotel to the database. The manager selects accommodation tab and adds the name, place, type and class. Finally he submits the form.



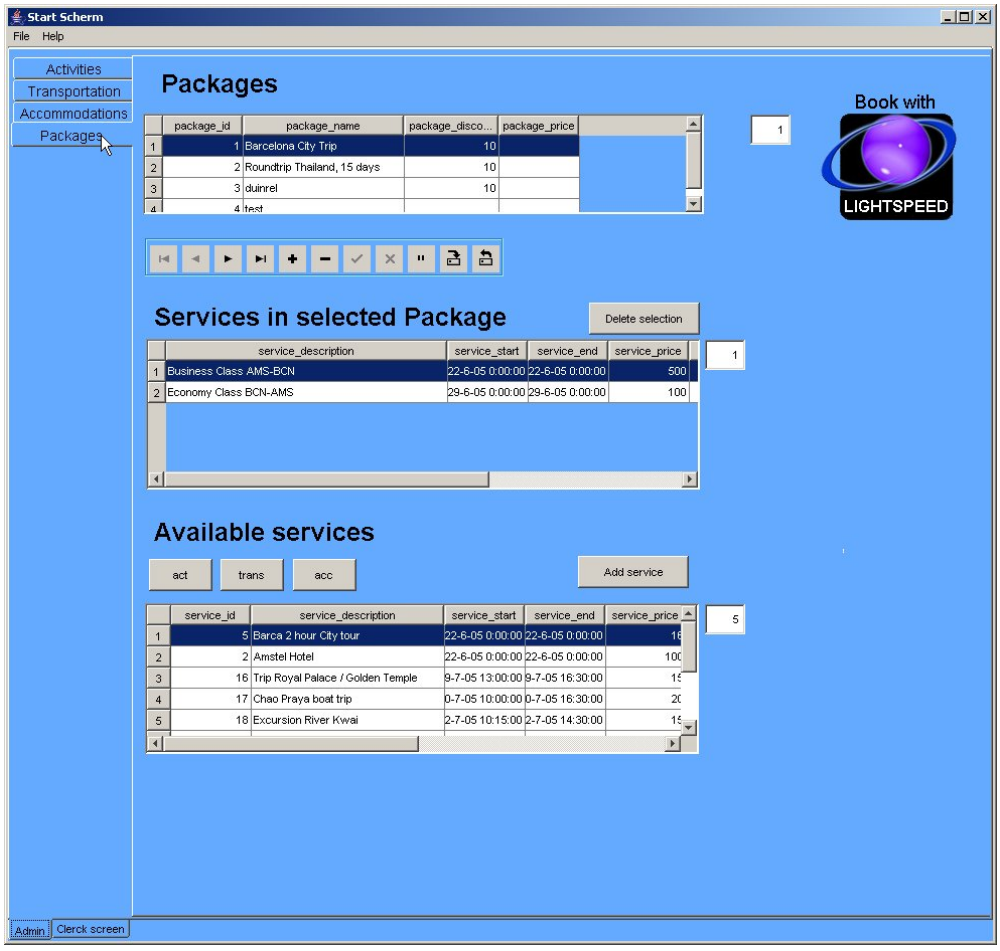
Scenario 2: Delivering services on a facility

Manager wants to specify when transportation is available for booking.
The manager selects a facility. Then he selects a transportation type to offer services for. For example, a service can be added for a certain period and costs. The accommodations can have multiple services offered on them for different periods using different pricing strategies.



Scenario 3: Creating a package

Manager wants to create a new package and provide a discount on the services included. The manager goes to the package tab and creates a new package. Then services can be selected and added to the package. Finally a discount can be set.



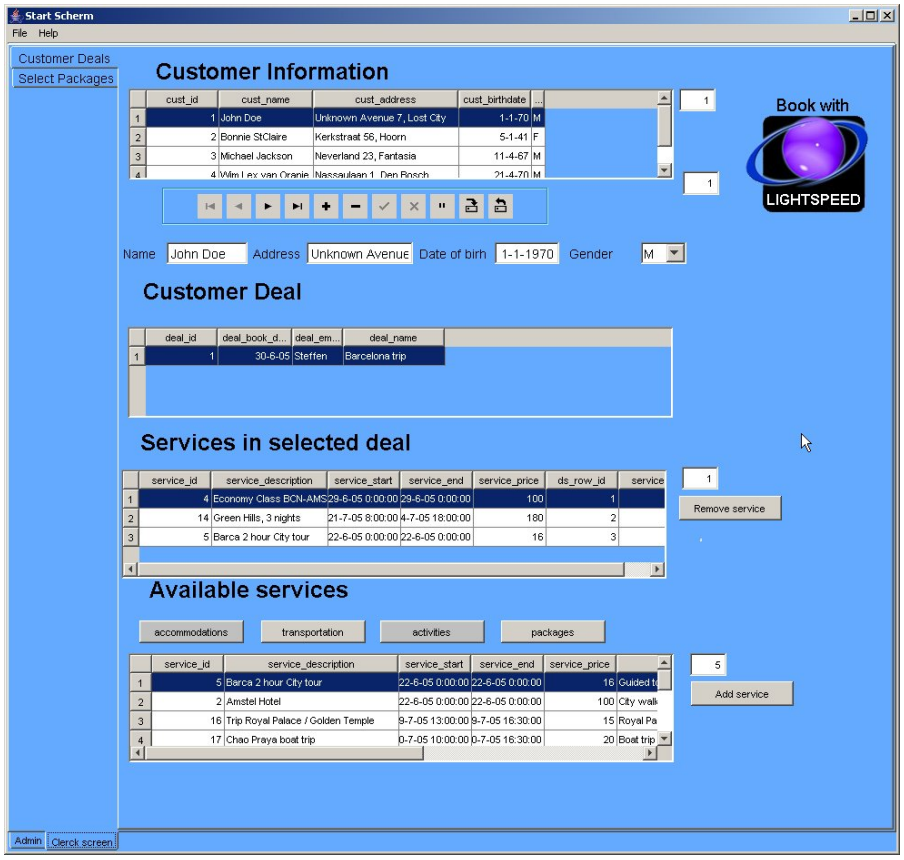
Clerk interaction with the customer

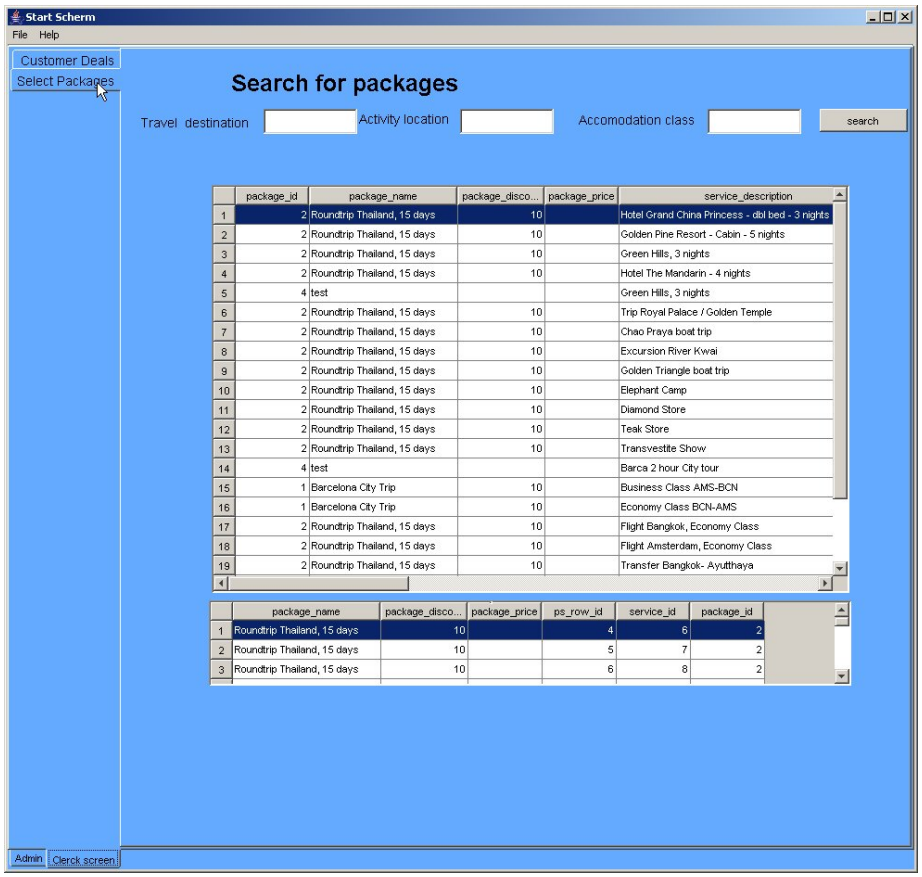
Scenario 4: Making a booking

The customer calls the travel agency and will be helped by the clerk who will help the customer to select a predefined package or make a customized deal.
Suppose the customer wants to book a trip to Spain, wants to reside in a luxurious hotel and likes to see as much from the city as possible.

The clerk goes to the customer/booking tab where all the packages are shown in a list. By selecting location or class from the dropdown field the package list will be show all predefined packages matching these requirements. By selecting a package details on the services will be shown in the list at the bottom of the form.

When the customer wants to buy the package the services of the package will be transferred to the deal tab. In this view it will be possible to make some final adjustments by adding more service or removing them.





	package_id	package_name	package_disco...	package_price	service_description
1	2	Roundtrip Thailand, 15 days	10		Hotel Grand China Princess - dbl bed - 3 nights
2	2	Roundtrip Thailand, 15 days	10		Golden Pine Resort - Cabin - 5 nights
3	2	Roundtrip Thailand, 15 days	10		Green Hills, 3 nights
4	2	Roundtrip Thailand, 15 days	10		Hotel The Mandarin - 4 nights
5	4	test			Green Hills, 3 nights
6	2	Roundtrip Thailand, 15 days	10		Trip Royal Palace / Golden Temple
7	2	Roundtrip Thailand, 15 days	10		Chao Praya boat trip
8	2	Roundtrip Thailand, 15 days	10		Excursion River Kwai
9	2	Roundtrip Thailand, 15 days	10		Golden Triangle boat trip
10	2	Roundtrip Thailand, 15 days	10		Elephant Camp
11	2	Roundtrip Thailand, 15 days	10		Diamond Store
12	2	Roundtrip Thailand, 15 days	10		Teak Store
13	2	Roundtrip Thailand, 15 days	10		Transvestite Show
14	4	test			Barca 2 hour City tour
15	1	Barcelona City Trip	10		Business Class AMS-BCN
16	1	Barcelona City Trip	10		Economy Class BCN-AMS
17	2	Roundtrip Thailand, 15 days	10		Flight Bangkok, Economy Class
18	2	Roundtrip Thailand, 15 days	10		Flight Amsterdam, Economy Class
19	2	Roundtrip Thailand, 15 days	10		Transfer Bangkok- Ayutthaya

	package_name	package_disco...	package_price	ps_row_id	service_id	package_id
1	Roundtrip Thailand, 15 days	10		4	6	2
2	Roundtrip Thailand, 15 days	10		5	7	2
3	Roundtrip Thailand, 15 days	10		6	8	2

Conclusion

This project description illustrate that is it possible to implement a relational centralized database application for a travel agency with JAVA. The following steps have been taken to develop and design the database. The first step consists of making the conceptual design. The conceptual level describes what the purpose of the system must be and the required functionalities. Following from the requirements and the constraints an ER diagram represented a graphically logical structure for the database. The second step is conversion of the conceptual model of the system into a physical structure, this is development phase. The ER diagram was the starting point for making a relational database. The tables of the database schema are the structure for the database. The database schemas have been transferred to a MySQL 4.x database. To make an application interface JAVA code has been developed.
Finally, to make it possible for a clerk and manager to communicate with the database we designed a GUI.. Through the GUI it is possible for the clerk to make bookings for a customer. The GUI enables the manager to easily moderate the database. The GUI will provide the clerks and managers an efficient and effective working environment.

Recommendations

Database

Though the MySQL database is free of charge for educational purposes in some cases licenses are required for use in a business environment. For professional use the application could benefit from more mature database features like views or stored procedures which MySQL does not offer in their stable (4.x) branch. Therefore, either Postgres (completely free) or Oracle (commercial) are recommended for their functionality and stability for application in a business environment.

Database design

For each entity a limited amount of attributes has been implemented. In stead of trying to be overly complete on e.g. customer attributes, focus has mainly been on the functionality. In future releases of the application the number of attributes can be easily extended.

Literature

1. Afsarmanesh, H. Slides Lecture 2 Project Databases
2. Afsarmanesh, H. Slides Lecture 3.2 Project Databases
3. Silberschatz A., Korth H.F., Sudarshan S. Database System Concepts 4th ed. McGraw-Hill 2002
4. Guevara-Masis V, Afsarmanesh H, Hertzberger L.O., Ontology-based automatic data structure generation for collaborative networks. IFIP 2004.

Other

www.mysql.com
www.oracle.com
www.postgres.org
www.borland.com

Appendix A: Database Definition

```
#
# Structure for the `accomodation` table :
#

CREATE TABLE `accomodation` (
  `acc_id` int(11) NOT NULL auto_increment,
  `acc_name` varchar(64) NOT NULL default '',
  `acc_location` varchar(64) NOT NULL default '',
  `acc_type` varchar(64) NOT NULL default '',
  `acc_class` int(11) NOT NULL default '0',
  PRIMARY KEY  (`acc_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `activity` table :
#

CREATE TABLE `activity` (
  `act_id` int(11) NOT NULL auto_increment,
  `act_description` varchar(64) default NULL,
  `act_location` varchar(64) NOT NULL default '0',
  `act_contact` varchar(64) default NULL,
  `act_type` varchar(64) default NULL,
  PRIMARY KEY  (`act_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `customer` table :
#

CREATE TABLE `customer` (
  `cust_id` int(11) NOT NULL auto_increment,
  `cust_name` varchar(64) NOT NULL default '',
  `cust_address` varchar(64) NOT NULL default '',
  `cust_birthdate` date NOT NULL default '0000-00-00',
  `cust_gender` char(1) NOT NULL default '',
  PRIMARY KEY  (`cust_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `deal` table :
#

CREATE TABLE `deal` (
  `deal_id` int(11) NOT NULL auto_increment,
  `deal_book_date` date default NULL,
  `deal_employee` varchar(64) default NULL,
  `deal_name` varchar(64) default NULL,
  `deal_cust_id` int(11) default NULL,
  PRIMARY KEY  (`deal_id`),
  KEY `deal_cust_id` (`deal_cust_id`),
  CONSTRAINT `deal_ibfk_1` FOREIGN KEY (`deal_cust_id`) REFERENCES
`customer` (`cust_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `service` table :
#
```

```
CREATE TABLE `service` (
  `service_id` int(11) NOT NULL auto_increment,
  `service_description` varchar(64) NOT NULL default '',
  `service_start` timestamp NOT NULL default '0000-00-00 00:00:00',
  `service_end` timestamp NOT NULL default '0000-00-00 00:00:00',
  `service_price` float(9,2) NOT NULL default '0.00',
  PRIMARY KEY  (`service_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `deal_services` table :
#

CREATE TABLE `deal_services` (
  `deal_id` int(11) default NULL,
  `service_id` int(11) default NULL,
  KEY `SERVICE_ID` (`service_id`),
  KEY `DEAL_ID` (`deal_id`),
  KEY `service_deal_ibfk_1` (`service_id`),
  KEY `service_deal_ibfk_2` (`deal_id`),
  CONSTRAINT `deal_services_ibfk_1` FOREIGN KEY (`service_id`)
REFERENCES `service` (`service_id`),
  CONSTRAINT `deal_services_ibfk_2` FOREIGN KEY (`deal_id`) REFERENCES
`deal` (`deal_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `package` table :
#

CREATE TABLE `package` (
  `package_id` int(11) NOT NULL auto_increment,
  `package_name` varchar(64) NOT NULL default '',
  `package_discount` float(31,30) default NULL,
  `package_price` float(31,30) default NULL,
  PRIMARY KEY  (`package_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `package_services` table :
#

CREATE TABLE `package_services` (
  `service_id` int(11) default NULL,
  `package_id` int(11) default NULL,
  KEY `SERVICE_ID` (`service_id`),
  KEY `PACKAGE_ID` (`package_id`),
  KEY `package_services_ibfk_1` (`service_id`),
  KEY `package_services_ibfk_2` (`package_id`),
  CONSTRAINT `package_services_ibfk_1` FOREIGN KEY (`service_id`)
REFERENCES `service` (`service_id`),
  CONSTRAINT `package_services_ibfk_2` FOREIGN KEY (`package_id`)
REFERENCES `package` (`package_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `service_on_accomodation` table :
#

CREATE TABLE `service_on_accomodation` (
```

```
  `s_on_acc_service_id` int(11) NOT NULL default '0',
  `s_on_acc_acc_id` int(11) default NULL,
  PRIMARY KEY  (`s_on_acc_service_id`),
  KEY `s_on_acc_service_id` (`s_on_acc_service_id`),
  KEY `s_on_acc_acc_id` (`s_on_acc_acc_id`),
  CONSTRAINT `service_on_accomodation_ibfk_3` FOREIGN KEY
(`s_on_acc_service_id`) REFERENCES `service` (`service_id`) ON DELETE
CASCADE,
  CONSTRAINT `service_on_accomodation_ibfk_2` FOREIGN KEY
(`s_on_acc_acc_id`) REFERENCES `accomodation` (`acc_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `service_on_activity` table :
#

CREATE TABLE `service_on_activity` (
  `s_on_act_service_id` int(11) NOT NULL default '0',
  `s_on_act_act_id` int(11) default NULL,
  PRIMARY KEY  (`s_on_act_service_id`),
  KEY `s_on_act_act_id` (`s_on_act_act_id`),
  KEY `s_on_act_service_id` (`s_on_act_service_id`),
  CONSTRAINT `service_on_activity_ibfk_2` FOREIGN KEY
(`s_on_act_service_id`) REFERENCES `service` (`service_id`) ON DELETE
CASCADE,
  CONSTRAINT `service_on_activity_ibfk_1` FOREIGN KEY
(`s_on_act_act_id`) REFERENCES `activity` (`act_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `transportation` table :
#

CREATE TABLE `transportation` (
  `trans_id` int(11) NOT NULL auto_increment,
  `trans_to` varchar(64) NOT NULL default '',
  `trans_type` varchar(64) NOT NULL default '',
  `trans_from` varchar(64) NOT NULL default '',
  `trans_code` varchar(64) NOT NULL default '',
  PRIMARY KEY  (`trans_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Structure for the `service_on_transportation` table :
#

CREATE TABLE `service_on_transportation` (
  `s_on_trans_service_id` int(11) NOT NULL default '0',
  `s_on_trans_trans_id` int(11) default NULL,
  PRIMARY KEY  (`s_on_trans_service_id`),
  KEY `s_on_trans_service_id` (`s_on_trans_service_id`),
  KEY `s_on_trans_trans_id` (`s_on_trans_trans_id`),
  CONSTRAINT `service_on_transportation_ibfk_3` FOREIGN KEY
(`s_on_trans_service_id`) REFERENCES `service` (`service_id`) ON
DELETE CASCADE,
  CONSTRAINT `service_on_transportation_ibfk_2` FOREIGN KEY
(`s_on_trans_trans_id`) REFERENCES `transportation` (`trans_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#
# Definition for the `new_view4` view :
```

```
#

CREATE ALGORITHM=UNDEFINED VIEW `new_view4` AS (
  select
    `package`.`package_id` AS `package_id`,
    `package`.`package_name` AS `package_name`,
    `package`.`package_discount` AS `package_discount`,
    `package`.`package_price` AS `package_price`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`,
    `accomodation`.`acc_name` AS `acc_name`,
    `accomodation`.`acc_location` AS `acc_location`,
    `accomodation`.`acc_type` AS `acc_type`,
    `accomodation`.`acc_class` AS `acc_class`,
    NULL AS `act_description`,
    NULL AS `act_location`,
    NULL AS `act_contact`,
    NULL AS `act_type`,
    NULL AS `trans_to`,
    NULL AS `trans_type`,
    NULL AS `trans_from`,
    NULL AS `trans_code`
  from
    `package` join `package_services` join `service` join
`service_on_accomodation` join `accomodation`
  where
    ((`package`.`package_id` = `package_services`.`package_id`) and
(`package_services`.`service_id` = `service`.`service_id`) and
(`service`.`service_id` =
`service_on_accomodation`.`s_on_acc_service_id`) and
(`service_on_accomodation`.`s_on_acc_acc_id` =
`accomodation`.`acc_id`))) union (
  select
    `package`.`package_id` AS `package_id`,
    `package`.`package_name` AS `package_name`,
    `package`.`package_discount` AS `package_discount`,
    `package`.`package_price` AS `package_price`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`,
    NULL AS `acc_name`,
    NULL AS `acc_location`,
    NULL AS `acc_type`,
    NULL AS `acc_class`,
    `activity`.`act_description` AS `act_description`,
    `activity`.`act_location` AS `act_location`,
    `activity`.`act_contact` AS `act_contact`,
    `activity`.`act_type` AS `act_type`,
    NULL AS `trans_to`,
    NULL AS `trans_type`,
    NULL AS `trans_from`,
    NULL AS `trans_code`
  from
    `service` join `package` join `package_services` join
`service_on_activity` join `activity`
  where
    ((`package`.`package_id` = `package_services`.`package_id`) and
(`package_services`.`service_id` = `service`.`service_id`) and
```

```
(`service`.`service_id` = `service_on_activity`.`s_on_act_service_id`)
and (`service_on_activity`.`s_on_act_act_id` = `activity`.`act_id`)))
union (
  select
    `package`.`package_id` AS `package_id`,
    `package`.`package_name` AS `package_name`,
    `package`.`package_discount` AS `package_discount`,
    `package`.`package_price` AS `package_price`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`,
    NULL AS `acc_name`,
    NULL AS `acc_location`,
    NULL AS `acc_type`,
    NULL AS `acc_class`,
    NULL AS `act_description`,
    NULL AS `act_location`,
    NULL AS `act_contact`,
    NULL AS `act_type`,
    `transportation`.`trans_to` AS `trans_to`,
    `transportation`.`trans_type` AS `trans_type`,
    `transportation`.`trans_from` AS `trans_from`,
    `transportation`.`trans_code` AS `trans_code`
  from
    `service` join `package` join `package_services` join
`service_on_transportation` join `transportation`
  where
    ((`package`.`package_id` = `package_services`.`package_id`) and
(`package_services`.`service_id` = `service`.`service_id`) and
(`service`.`service_id` =
`service_on_transportation`.`s_on_trans_service_id`) and
(`service_on_transportation`.`s_on_trans_trans_id` =
`transportation`.`trans_id`)));

#
# Definition for the `v_accomodation_services` view :
#

CREATE ALGORITHM=UNDEFINED VIEW `v_accomodation_services` AS
  select
    `accomodation`.`acc_id` AS `acc_id`,
    `accomodation`.`acc_name` AS `acc_name`,
    `accomodation`.`acc_location` AS `acc_location`,
    `accomodation`.`acc_type` AS `acc_type`,
    `accomodation`.`acc_class` AS `acc_class`,
    `service`.`service_id` AS `service_id`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`
  from
    `accomodation` join `service_on_accomodation` join `service`
  where
    ((`service`.`service_id` =
`service_on_accomodation`.`s_on_acc_service_id`) and
(`service_on_accomodation`.`s_on_acc_acc_id` =
`accomodation`.`acc_id`));

#
# Definition for the `v_activity_services` view :
```

```
#

CREATE ALGORITHM=UNDEFINED VIEW `v_activity_services` AS
  select
    `service`.`service_id` AS `service_id`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`,
    `activity`.`act_id` AS `act_id`,
    `activity`.`act_description` AS `act_description`,
    `activity`.`act_location` AS `act_location`,
    `activity`.`act_contact` AS `act_contact`,
    `activity`.`act_type` AS `act_type`
  from
    `service` join `service_on_activity` join `activity`
  where
    ((`service`.`service_id` =
`service_on_activity`.`s_on_act_service_id`) and
(`service_on_activity`.`s_on_act_act_id` = `activity`.`act_id`));

#
# Definition for the `v_customer_deals` view :
#

CREATE ALGORITHM=UNDEFINED VIEW `v_customer_deals` AS
  select
    `customer`.`cust_id` AS `cust_id`,
    `customer`.`cust_name` AS `cust_name`,
    `customer`.`cust_address` AS `cust_address`,
    `customer`.`cust_birthdate` AS `cust_birthdate`,
    `customer`.`cust_gender` AS `cust_gender`,
    `deal`.`deal_id` AS `deal_id`,
    `deal`.`deal_book_date` AS `deal_book_date`,
    `deal`.`deal_employee` AS `deal_employee`,
    `deal`.`deal_name` AS `deal_name`
  from
    `deal` join `customer`
  where
    (`deal`.`deal_cust_id` = `customer`.`cust_id`);

#
# Definition for the `v_deal_contents` view :
#

CREATE ALGORITHM=UNDEFINED VIEW `v_deal_contents` AS
  select
    `deal`.`deal_id` AS `deal_id`,
    `deal`.`deal_book_date` AS `deal_book_date`,
    `deal`.`deal_employee` AS `deal_employee`,
    `deal`.`deal_name` AS `deal_name`,
    `deal`.`deal_cust_id` AS `deal_cust_id`,
    `service`.`service_id` AS `service_id`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`
  from
    `service` join `deal` join `deal_services`
  where
```

```
    ((`deal`.`deal_id` = `deal_services`.`deal_id`) and
(`deal_services`.`service_id` = `service`.`service_id`));

#
# Definition for the `v_package_contents` view :
#

CREATE ALGORITHM=UNDEFINED VIEW `v_package_contents` AS
  select
    `package`.`package_id` AS `package_id`,
    `package`.`package_name` AS `package_name`,
    `package`.`package_discount` AS `package_discount`,
    `package`.`package_price` AS `package_price`,
    `service`.`service_id` AS `service_id`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`
  from
    `service` join `package` join `package_services`
  where
    ((`package`.`package_id` = `package_services`.`package_id`) and
(`package_services`.`service_id` = `service`.`service_id`));

#
# Definition for the `v_transpratation_services` view :
#

CREATE ALGORITHM=UNDEFINED VIEW `v_transpratation_services` AS
  select
    `transportation`.`trans_id` AS `trans_id`,
    `transportation`.`trans_to` AS `trans_to`,
    `transportation`.`trans_type` AS `trans_type`,
    `transportation`.`trans_from` AS `trans_from`,
    `transportation`.`trans_code` AS `trans_code`,
    `service`.`service_id` AS `service_id`,
    `service`.`service_description` AS `service_description`,
    `service`.`service_start` AS `service_start`,
    `service`.`service_end` AS `service_end`,
    `service`.`service_price` AS `service_price`
  from
    (`service` join (`service_on_transportation` join
`transportation`))
  where
    ((`service`.`service_id` =
`service_on_transportation`.`s_on_trans_service_id`) and
(`service_on_transportation`.`s_on_trans_trans_id` =
`transportation`.`trans_id`));
```