JIMOH MORUFU OLAIDE

MySQL DATABASE

Bachelor's Thesis
Information Technology

JUNE 2010

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	Date of the bachelor's thesis **June 2010**
Author(s) Morufu Jimoh	**Degree programme and option** Information Technology

Name of the bachelor's thesis

MySQL Database

Abstract

Objectives

The main objectives of this thesis were to show how it is much easier and faster to find required information from computer database than from other data storage systems or old fashioned way. We will be able to add, retrieve and update data in a computer database easily.

Using computer database for company to keep the information for their customer is the objective of my thesis. It is faster which makes it economically a better solution. The project has six tables which are branch, staff, customer, car, type and hirecontract.

It shows how six tables that contain information were connected with the help of primary and foreign keys. The new system can reduce the time it takes to find a specific piece of information. The system will allow all the staff within the company get access to the data.

Subject headings, (keywords)

MySQL Database

Pages	**Language**	**URN**
35	English	http://www.urn.fi/URN:NBN:fi:amk-2010060811776

Remarks, notes on appendices

Tutor	**Employer of the bachelor's thesis**
Reijo Vuohelainen	

CONTENTS

1 INTRODUCTIONS

The idea of this thesis is to create a database for a hypothetical company that has five branches located in different countries. It is a car rental company. This will allow each branch to be able to add, update and retrieve data in database. This database will be based on the relational model. In many industries today relational model is widely used. The benefits of a computerised database for the company are various. It is faster to retrieve information from a database on a computer than to search it from an old fashioned card or other filing system. Also adding new data is faster, making it an economically better solution. A computer database is also more reliable. Chances of making mistakes, for example double entries, are smaller. It is more secure, because it can be used only by the people who are authorized and have the correct password. A database is accessible from anywhere, any branch of the company, as far as you have the necessary applications. Microsoft Access, Oracle and visual FoxPro are the main relational products used to create a database.

The aim of this thesis is to show how it is much faster and easier to find required information from a database than from other data storage systems. How can we calculate the average sales within the company by using the database? How can we find a member of staff by the letter with which their name either begins or ends? How can we find out which contracts have a cost more than a certain amount? These questions will be later solved with the MySQL language.

SQL (Structured Query Language) is an industry standard language particularly designed to enable people to create databases, add new data to databases, insertion, updating and deletion of data. SQL was initially developed to operate on data in databases that follow the relational model. SQL is a programming language for querying and modifying data and managing data. [3]

A database management system also includes management and administrative functions, which use a command-line interface that allows the entry and execution of language commands. A database is a structure made to keep information that is valuable to the user. A database management system is the tool used to build the structure of the database and in operating on the data contained within it.

The project will have 6 tables. Which contain staff, branch, customer, car, type and hire contract data. I will also give examples of some useful queries that can be done with the database.

2 INTRODUCTION TO MySQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. In MySQL the beginning My of the name comes from the daughter of the Finnish developer[9]. SQL was initially developed to operate on data in databases that follow the relational model. It is a programming language for querying, modifying and managing data.

MySQL is the most common open source database tool. It is considered an easy and reliable program compared to other database software. MySQL offers various different programs that are database related. The most famous one is MySQL Enterprise.

2.1 Defining databases

Database is a set of tables that can be manipulated in accordance with the relational model of data. It is a collection of integrated records. A record is a representation of some conceptual object that has multiple attributes like name, address and telephone number of a customer.

A database consists of both data and metadata. Metadata is data that describes the structure of the data within a database. They store metadata in an area called the data dictionary, which describes the tables, columns, index constraints, and other items that make up the database. [1] Databases come in all sizes, from a simple collection of a few records to millions of records.

2.2 Database design consideration

A database is a representation of a physical or conceptual structure, such as an organization, an automobile assembly, or the performance statistics of all the major-league baseball clubs. The accuracy of the representation depends on the level of detail of the database design. The amount of effort that you put into database design should depend on the type of information you want to get out of the database. It is always good to make a database plan and to be able to estimate the size of the database you need. A plan can vary from a single paper drawing to a hundred page document describing every possible thing to do with the database, usually when the plan is bigger a more complex database is needed. Decide how much detail you need now and how much you may need in future and then provide exactly that level of detail in your design. Databases are perfect for applications that have many users because coordination between all users is easy.

Maintenance and security questions should also be properly thought before designing anything. Database should be designed to be as small as possible and to avoid not useful information. It might be not so easy to do bigger changes in your database later. Normalization rules should always be followed and different kind of applications might have different performance needs. Database can also be over normalized, this means that before normalization it has lots of small tables that form a complicated network of relations which can affect performance and should be avoided if possible. Performance also depends on the amount of users so you should consider what changes might happen to the amount of users of your database in future and what are the effects. Data integrity is in danger if you do not correctly limit the values entered to fields, it reduces the quality of data.

2.3 Comparing database models

The relational model, as expressed through relational calculus and relational algebra, does not distinguish between primary keys and other kinds of keys. Primary keys were added to the SQL standard mainly as a convenience to the application programmer.

Hierarchical database model uses parent/child way of representing data, if drawn it would be tree shaped. Each parent record can have more than one child records. Entity types are all in one-to-many form. Probably the most well known example of a hierarchical database is the windows registry.

Network model is more flexible than the hierarchical structure because it allows each record to have one or more parent and/or child records. The relational model allows whole tables of records to be related to each others. There is also an object oriented database model which is not that commonly used but it works well in some specific uses for example with molecular biology.

All these database models are in use but the relative model is most common because it is most flexible of these. In this thesis I have decided to use the relational database model because it allows the user to make queries and retrieve information more freely and gives more possibilities for detailed searches.

3 DATA TYPE

Different SQL implementations support a variety of data types. Some SQL recognizes only six general types; exact numerics, approximate numerics, character strings, bit strings, datetime, and intervals.

There are SQL implementations that support one or more data types that SQL-92 specification doesn’t describe, by avoiding these un-described data types your database can be kept more portable.

3.1 Character String

Databases store many different types of data, including graphic images, sound and animation. A series of characters manipulated as a group. A character string differs from a name in that it does not represent anything, a name stands for some other object. A character string is often specified by enclosing the characters in single or double quotes. For example, MIKKELI would be a name, but 'MIKKELI' and "MIKKELI" would be character strings. The length of a character string is usually the number of characters in it. For example, the character string "MIKKELI" has a length of 7 (the quote marks are not included). Some programs, however, mark the beginning or end of a character string with an invisible character, so the length might actually be one greater than the number of intended characters.

There are two main types of character data: fixed character data (CHARACTER or CHAR) and varying character data (CHARACTER VARYING or VARCHAR). We also have two variants of these types of character data: NATIONAL CHARACTER and NATIONAL CHARACTER VARYING.

3.2 Character data types

Character data as the name says can be data of any kind of characters. It includes capital and small alphabet, number digits and punctuation marks (.,?/@ and many more). These different kinds of characters can be in any order, for example numbers mixed with capital letters. The number of characters in a column can be specified by using the syntax character (X), where x is the number of characters. If we specify a column’s data type as CHARACTER (40), the maximum length of any data you can enter in the column is 40 characters.

3.3 Integer Data type

Data of the integer type has no fraction part, and its precision depends on the specific SQL implementation. This data type is basically whole numbers (1,2,3…), the values jump, meaning that there is nothing between 1 and 2. This type of data in a database can’t have negative values. It is good to have this integer data type although it is restrictive. There are lots of things that can be counted only in whole numbers, for example you can’t rent half a car.

3.4 Date and Time

The date type stores year, month, and day values of a date, in that order. The year value is four digits long and the month and day values are both two digits long. A date value can represent any date from the year 0001 to 9999. So the length of a date is ten positions as in 2009-05-29. The time data type stores hour, minute, and second values of time. The hours and minutes occupy exactly two digits. The second value may be only two digits but may also expand to include an optional fraction part. Wrong date information can also be added on some occasions, for example if you do not know an exact birth date it is possible to put it with zeros instead of months or days, for example 2005-00-00.

3.5 Numerical and decimal

Numerical data can have a fraction component in addition to its integer component. We can specify both the precision and the scale of numerical data. The scale of a number is the number of digits in its fraction part. The scale of a number can’t be negative or larger than that number’s precision. Specifications such as how many digits the fraction part can have vary in different versions of programs. Numericals can be signed or unsigned which means they have either + or – in front of them or nothing. If they are unsigned they are always positive.

4 TABLE

All databases are made of one or more tables. They are the structure of database that holds the information, the very foundation of any relational database. Tables are divided in rows and columns (also often called fields). Rows contain the actual data while columns describe and limit the quality and form of data put there. Each table must have its own unique name and so do every column of the table. When viewing a table the first row usually shows the names of all columns. Each column has its own pre-defined data type, meaning that the data entry is limited by certain requirements, for example character, text, date or timestamp. There are also other conditions to what kind of data the fields can accept such as not null or unique.

4.1 Creating tables

Table is a two dimensional array made up of rows and columns. It can be created by using SQL’s CREATE TABLE command. Within the command, you specify the name and data type of each column. After creating a table, next thing to do is to start loading the table with data. Loading data is a DML (Data manipulation language), not a DDL (Data definition language), function. The table’s structure can be changed after you have created it by using the ALTER TABLE command but in MYSQL you can just click change button to change what ever you want. In some circumstances you might want to delete the whole table by using DROP command or click DROP button in MYSQL. Create, Alter and Drop commands they all make up SQL’s DDL. [1]

When building tables for the database it is desirable that you do not leave any mistakes, because when you make updates over time this could lead to problems. Each table created contains columns that correspond to attributes that are tightly linked to each other.

Below is the table needed for this database which shows the number of records and their sizes.

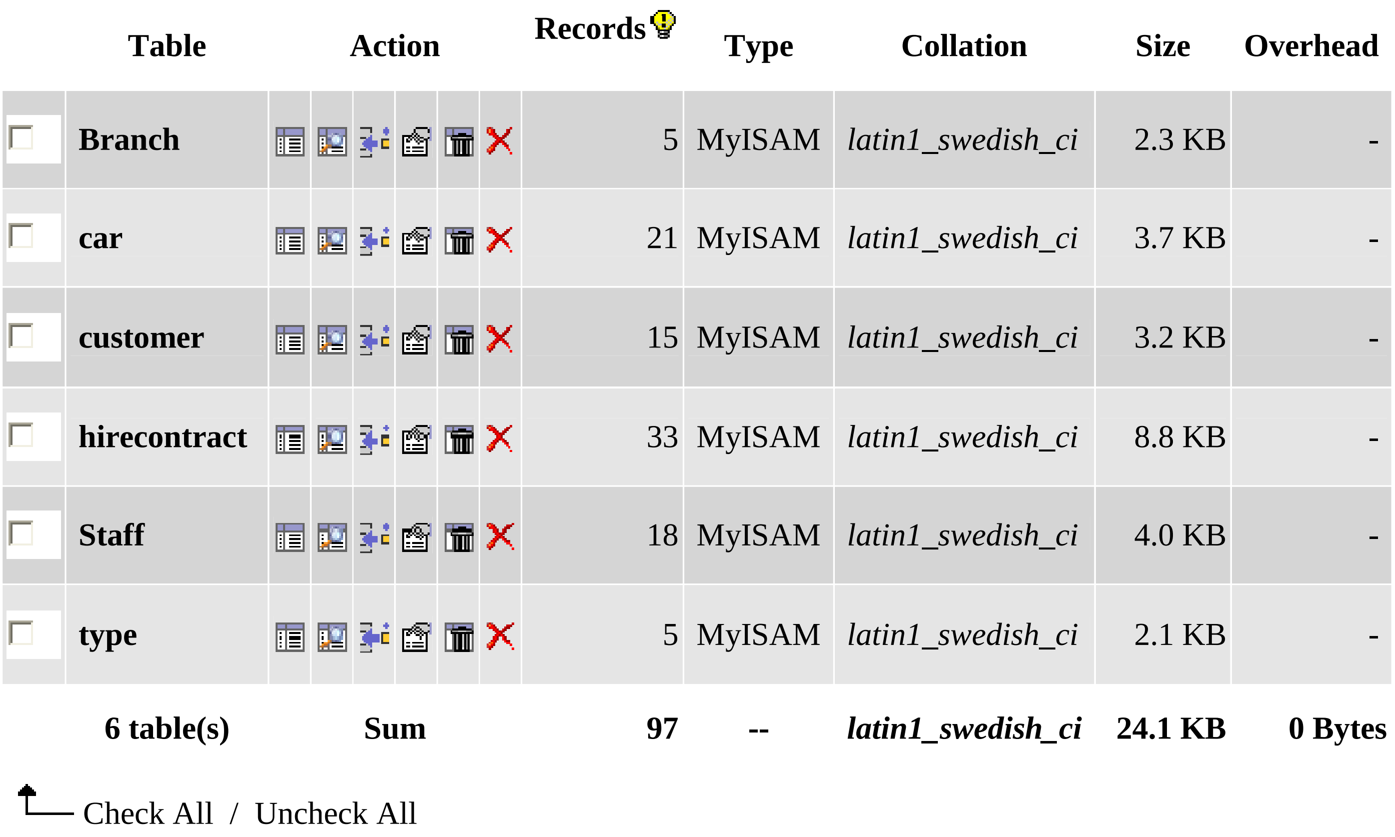


	Table	Action	Records	Type	Collation	Size	Overhead
☐	**Branch**		5	MyISAM	*latin1_swedish_ci*	2.3 KB	-
☐	**car**		21	MyISAM	*latin1_swedish_ci*	3.7 KB	-
☐	**customer**		15	MyISAM	*latin1_swedish_ci*	3.2 KB	-
☐	**hirecontract**		33	MyISAM	*latin1_swedish_ci*	8.8 KB	-
☐	**Staff**		18	MyISAM	*latin1_swedish_ci*	4.0 KB	-
☐	**type**		5	MyISAM	*latin1_swedish_ci*	2.1 KB	-
	6 table(s)	**Sum**	**97**	**--**	***latin1_swedish_ci***	**24.1 KB**	**0 Bytes**

Check All / Uncheck All

4.2 Entity integrity

All the tables in a database correspond to an entity in the real world. That entity can be physical or conceptual, but in some sense, the entity's existence is independent of the database. A table has entity integrity if the table is entirely consistent with the entity that it models. To have entity integrity, a table must have a primary key. The primary key uniquely identifies each row in a table. Without a primary key, you can not be sure that the row retrieved is the one you want. So in order to maintain entity integrity, we need to specify that the column or group of columns that compromise the primary key are NOT NULL. It should be UNIQUE. [1]

A key is a specialized type of index that might be used for referential integrity. An index is just like a key in all respects, other than referential integrity and in that index can’t be constructed at the same time as a table is created. Index can be created on any field or combination of fields. The exception to this rule applied in most database engines is that an index can’t be created on a field or combination of fields, for which an index already exists.

4.3 Primary key

A unique key is nearly the same as primary key. It is a candidate key to uniquely identify each row in a table. A unique key or primary key comprises a single column or set of columns. No two distinct rows in a table can have the same value or combination of values in those columns. Depending on its design, a table may have arbitrarily many unique keys but at most one primary key. A unique key must uniquely identify all possible rows that exist in a table and not only the currently existing rows. Examples of unique keys are Social Security numbers associated with a specific person or ISBNs associated with a specific book. Telephone books and dictionaries cannot use names or words or Dewey Decimal system numbers as primary keys because they do not uniquely identify telephone numbers or words.

A primary key is a special case of unique keys. The major difference is that for unique keys the implicit NOT NULL constraint is not automatically enforced, while for primary keys it is. Thus, the values in a unique key column may or may not be NULL. Another difference is that primary keys must be defined using another syntax. Unique keys as well as primary keys can be referenced by foreign keys. [10]

4.4 Foreign key

A foreign key is linking two tables together, it must be a primary key in one table. It identifies a column or a set of them in one table that refers to a column or columns in another referenced table. The key should uniquely identify a column or columns in the referenced table but it is not necessary for it to be unique itself.

The values in one row of the referencing columns must occur in a single row in the referenced table. Thus, a row in the referencing table can’t contain values that don't exist in the referenced table (except potentially NULL). This way references can be made to link information together and it is an essential part of database normalization. Multiple rows in the referencing table may refer to the same row in the referenced table. Most of the time, it reflects the one (master table, or referenced table) to many (child table, or referencing table) relationship. The referencing and referenced table may be the same table, i.e. the foreign key refers back to the same table. Such a foreign key is known in SQL:2003 as a self-referencing or recursive foreign key. [11]

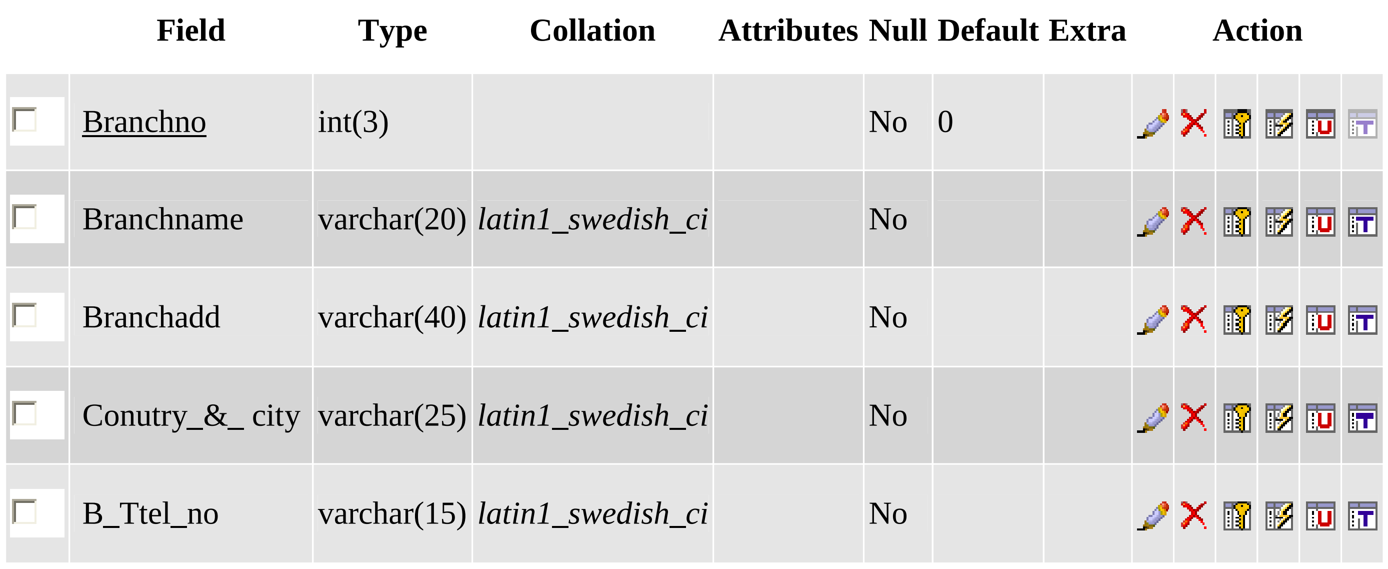
A table can have more than one foreign keys, which then can have a different referenced table. Foreign key depends on the primary key in the referential table, that’s why cascading links between tables can be made with foreign keys. If these relationships are not used correctly it can lead to serious problems in the database.

4.5 Referential integrity

Referential integrity requires that the values of a column or columns in one table match the values of a column or columns in another table. We refer to the columns in the first table as the foreign key and the columns in the second table as the primary key or unique key. For example we may declare that the column branchno in staff table is a foreign key that references the branchno column of a branch table. This matchup ensures that if we record a staff in the staff table as working in branch 2, a row appears in the branch table where branchno is 2. Tables 1-6 are the tables I have created for my example database.

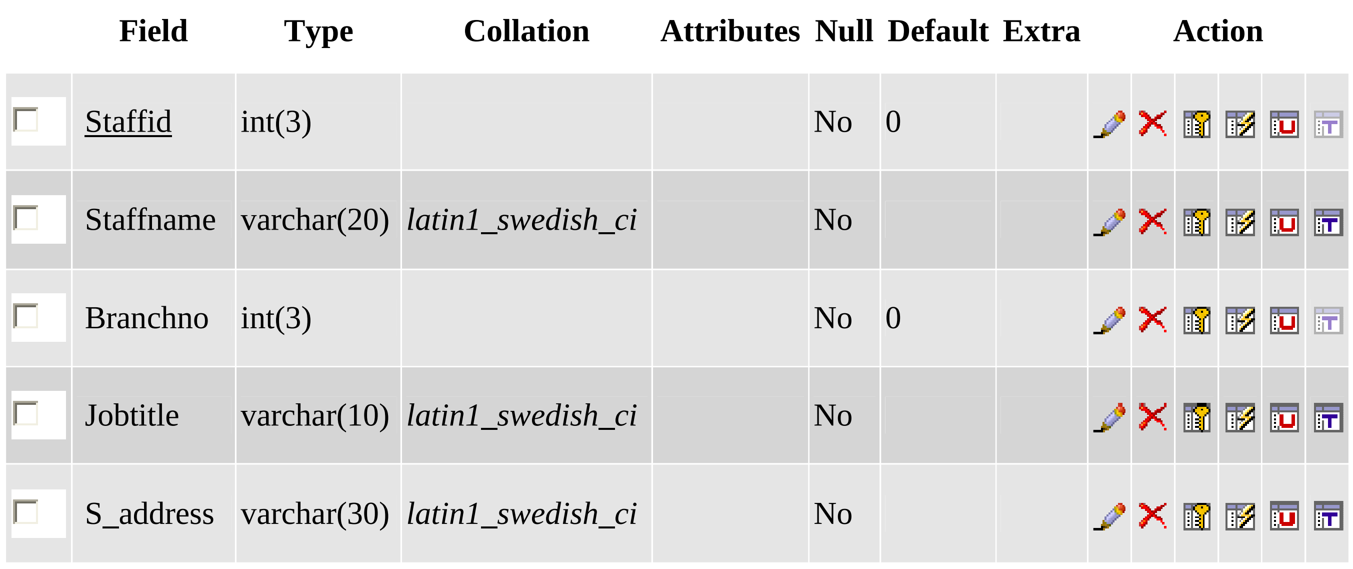
Database THESIS has been created. SQL-query:
CREATE DATABASE `THESIS` ;

Table 1. Branch Table



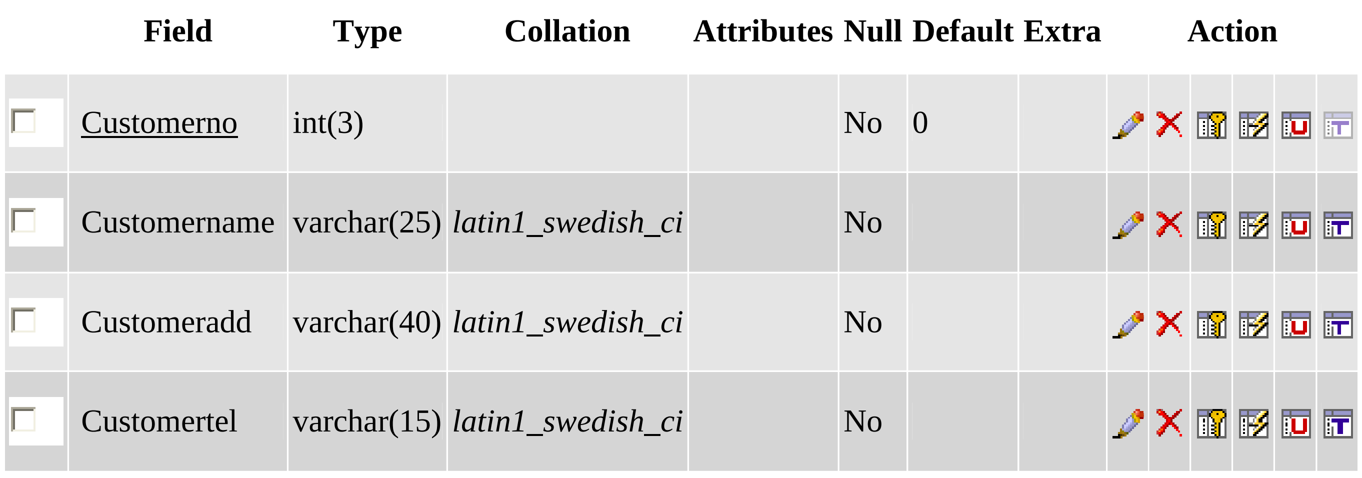
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Branchno	int(3)			No	0		
☐	Branchname	varchar(20)	*latin1_swedish_ci*		No			
☐	Branchadd	varchar(40)	*latin1_swedish_ci*		No			
☐	Conutry_&_ city	varchar(25)	*latin1_swedish_ci*		No			
☐	B_Ttel_no	varchar(15)	*latin1_swedish_ci*		No			

Table 2. Staff Table



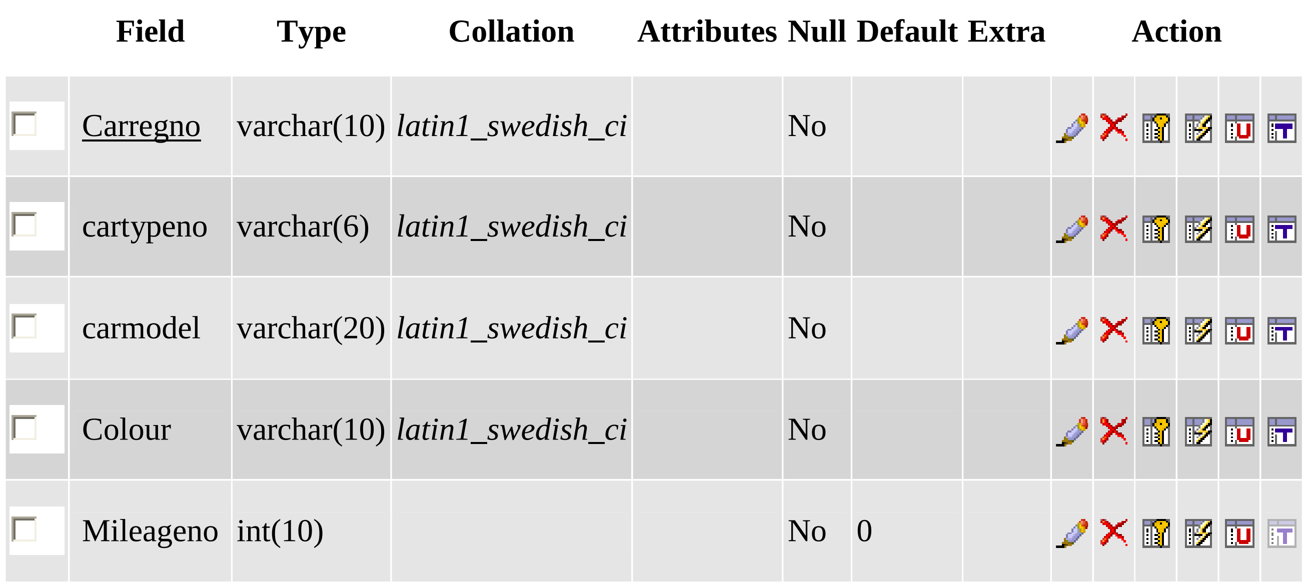
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Staffid	int(3)			No	0		
☐	Staffname	varchar(20)	*latin1_swedish_ci*		No			
☐	Branchno	int(3)			No	0		
☐	Jobtitle	varchar(10)	*latin1_swedish_ci*		No			
☐	S_address	varchar(30)	*latin1_swedish_ci*		No			

Table 3. Customer Table



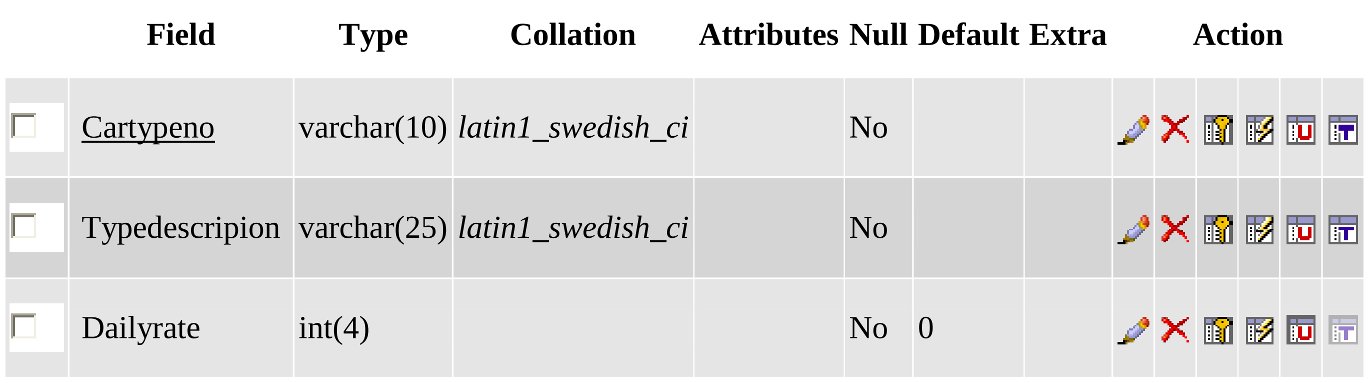
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Customerno	int(3)			No	0		
☐	Customername	varchar(25)	*latin1_swedish_ci*		No			
☐	Customeradd	varchar(40)	*latin1_swedish_ci*		No			
☐	Customertel	varchar(15)	*latin1_swedish_ci*		No			

Table 4. Car Table



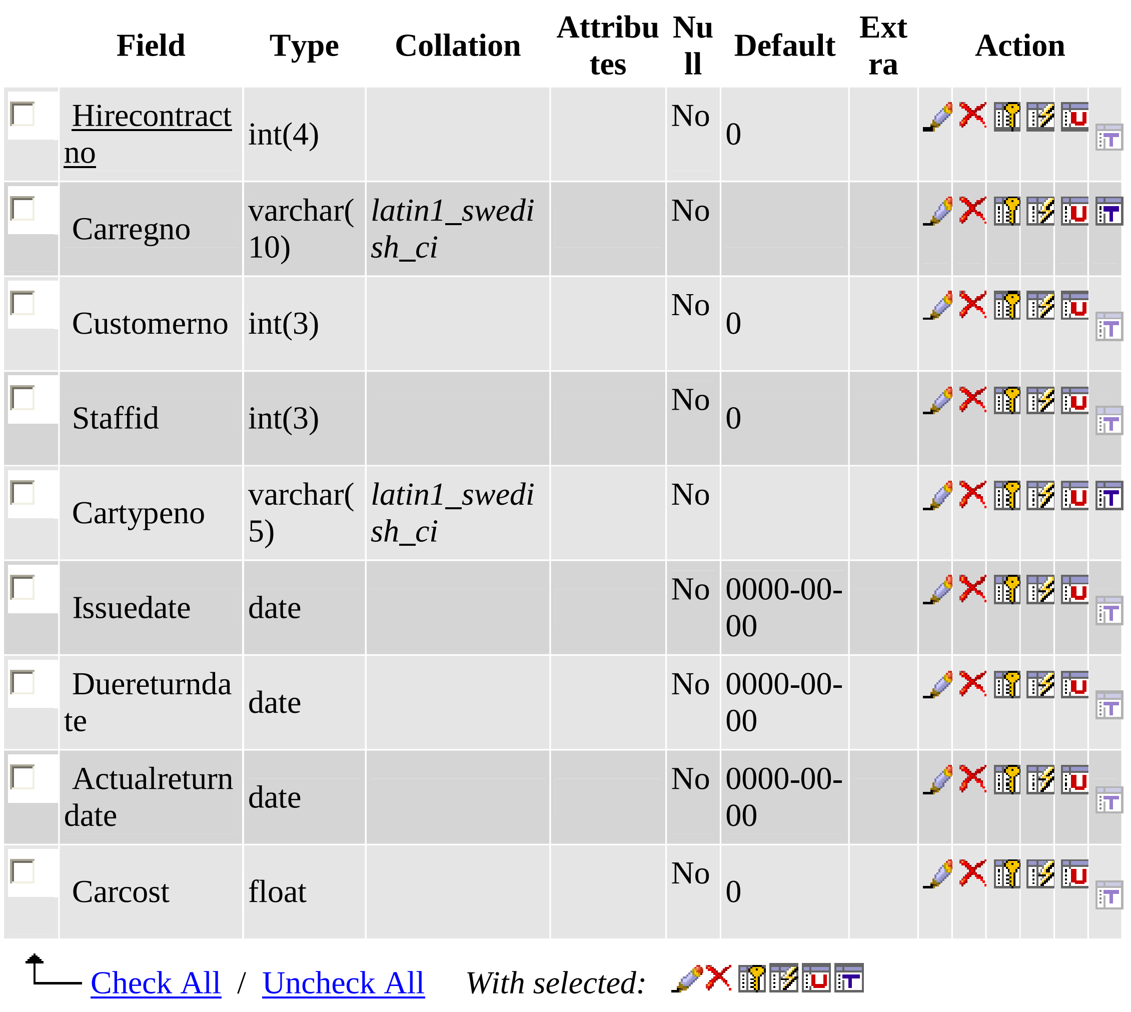
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Carregno	varchar(10)	*latin1_swedish_ci*		No			
☐	cartypeno	varchar(6)	*latin1_swedish_ci*		No			
☐	carmodel	varchar(20)	*latin1_swedish_ci*		No			
☐	Colour	varchar(10)	*latin1_swedish_ci*		No			
☐	Mileageno	int(10)			No	0		

Table 5. Type Table



	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Cartypeno	varchar(10)	*latin1_swedish_ci*		No			✎ ✗ ▦ ▦ ▦ ▦
☐	Typedescripion	varchar(25)	*latin1_swedish_ci*		No			✎ ✗ ▦ ▦ ▦ ▦
☐	Dailyrate	int(4)			No	0		✎ ✗ ▦ ▦ ▦ ▦

Table 6. Hirecontract Table



	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	Hirecontractno	int(4)			No	0		✎ ✗ ▦ ▦ ▦ ▦
☐	Carregno	varchar(10)	*latin1_swedish_ci*		No			✎ ✗ ▦ ▦ ▦ ▦
☐	Customerno	int(3)			No	0		✎ ✗ ▦ ▦ ▦ ▦
☐	Staffid	int(3)			No	0		✎ ✗ ▦ ▦ ▦ ▦
☐	Cartypeno	varchar(5)	*latin1_swedish_ci*		No			✎ ✗ ▦ ▦ ▦ ▦
☐	Issuedate	date			No	0000-00-00		✎ ✗ ▦ ▦ ▦ ▦
☐	Duereturndate	date			No	0000-00-00		✎ ✗ ▦ ▦ ▦ ▦
☐	Actualreturndate	date			No	0000-00-00		✎ ✗ ▦ ▦ ▦ ▦
☐	Carcost	float			No	0		✎ ✗ ▦ ▦ ▦ ▦

Check All / Uncheck All *With selected:* ✎ ✗ ▦ ▦ ▦ ▦

5 JOINING TABLES

This is a way by which two or more table records can be combined together in a database if there is a need to retrieve information from more than one table. This new combined table can either be saved as a new table or used as it is. If there is no possibility to join tables together some data would be repeated many times without any purpose. If there are no common values in the join attributes it is not possible to join the tables. If you want to join more than two tables you need more than one join conditions, one will only specify the link between two tables. There are various different kinds of joins but the most common is the inner join. This kind of query returns rows in which there is at least one match in both tables.

6 NORMALIZATION

Is a way of systematically ensuring that database structure is well suitable for normal querying and free of most common anomalies in undesirable characteristics insertion, update and deletion, which could lead to a loss of data integrity.

When an attempt is made to modify (update, insert into, or delete from) a table, undesired side-effects may follow. Not all tables can suffer from these side-effects; rather, the side-effects can only arise in tables that have not been sufficiently normalized.

There are up to six different normal forms but mainly the first three of these are in use. This level is usually enough to ensure the database is well normalized and doesn’t have any anomalies. If the database adheres to the 3rd normal form it is common for it to automatically be in 4th and 5th normal form too.

6.1 First normal form

Is to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic**.** Querying and manipulating the data within an un-normalized data structure, involves more complexity than is really necessary [12] and the same data might be repeated often. There are no commonly agreed exact rules as to what the first normal form should or shouldn’t contain but there are certain things that are desired. For example if you want to put more than one phone number in a database for a person it can be done in many ways, by adding two values in the same field or by creating two separate columns for first and second numbers, but the best way to do it is to create a separate table for them, this is the only way the normalization can be kept. Having two values in the same field means these two values can’t be separately found which affects the accuracy of data.

Table must have the following attributes to be in first normal form

Make sure that the table is a two-dimensional table, with rows and columns.

- Each column contains data for a single attribute of the thing it’s describing.
- Each row contains data that pertains to some thing or portion of a thing
- Each column must have a unique name.
- Entries in any column must all be of the same kind
- No two rows may be identical.
- Each cell of the table must have only a single value. [1]

6.2 Second normal form

Is a higher than the first form used in database normalization. Second normal form (2NF) was defined by E.F. Codd in 1971[12], he also introduced the first normal form a year earlier. A table normalized according to the criteria of first normal form (1NF) must maintain all that is required of 1NF and meet some additional criteria to qualify for 2NF. In order to deal with second normal form we must understand the idea of functional dependency. The definition in Wikipedia for 2NF is “A 1NF table is in 2NF if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it[12].” This means that the candidate key should identify all the attributes related to it. For example basic personal data held in a database about a person has a candidate key which is their social security number. No one can have the same social security number as this person and all other information about him can be verified with this candidate key. There can not be any duplicate of the candidate key.

A functional dependency is a relationship between or among attributes. One attribute is functionally dependent on another if the value of the second attribute determines the value of the first attribute. [1]

Every table in the first normal form must have a unique primary key. That key may consist of one or more than one column. A key consisting of more than one column is called a composite key. To be in second normal form, all non-key attributes (columns) must depend on the entire key. Thus, every relation that is in first normal form with single attribute key is automatically in second normal form. If a relation has a composite key, all non-key attributes must depend on all components of the key. If we have a table where some non-key attributes don’t depend on all components of the key. [1]

6.3 Third normal form

Third normal form is the same as second normal form except that it only refers to tables that have a single field as their primary key. In other words, each non-key field in the table should be a fact about the primary key. Either of the preceding two tables act as an example of third normal form since all the fields in each table are necessary to describe the primary key. Once all the tables in a database have been taken through the third normal form, we can begin to set up relationships

7 MANIPULATING DATABASE DATA

Database manipulation is really simple and very fast compared to old fashioned ways of storing data. Understanding how to add data to a table is not difficult. We can add data either one row at a time or in a batch. Deleting, changing and retrieving table rows are also easy in practice. The only challenge lies in selecting the rows that you want to change, delete or retrieve. We can specify which rows we want to change by using SELECT statement command. The computer will do all the searching.

7.1 Retrieving data

In many databases the task that people perform frequently is retrieving selected information. Whether we want to retrieve the contents of one specific row out of thousands in a table, retrieve all the rows that satisfy a condition or want to retrieve all rows in the table, one simple SQL statement, the SELECT statement, performs all these tasks. The retrieved data that you get is called query object or record set.

7.1.1 Query

A database query is a request for information from a database. In SQL, you can use a database query from the console to find information in the database or to target a job to SQL with particular characteristics. All queries are intended to return a list of information arranged in data rows.

You can query the database by selecting a saved query or by creating a new query using search criteria provided by SQL. Each query you create and save is registered in the database. Unlike searches, you can name and save queries in the database as registered queries and then use them in the future.

Named queries are used to obtain a list of rows that match search criteria. The resulting list of information is used for targeting jobs, filtering the display of rows and other tasks. It is limited to obtaining a list of information that matches the search criteria and nothing more.

From the list of queries you can also submit a job based on a selected query and display rows associated with a query. The management tasks you can perform on a registered query include displaying query properties and deleting a query from the database.

7.1.2 Select command

The SQL SELECT statement returns a result set of records from one or more tables. It retrieves zero or more rows from one or more base tables, temporary tables, or views in a database. In most applications SELECT is the most commonly used Data Manipulation Language (DML) command. As SQL is a non-procedural language, SELECT queries specify a result set, but do not specify how to calculate it. SELECT can also have some subqueries or UNION statements include. WHERE clause identifies the conditions wanted in selecting the correct data from database. If there are no conditions about the kind of data the SELECT statement should return it will show all of it.

7.2 Average

The AVG(Average) function calculates and returns the average of the values in the specific column. We can use the AVG function only on columns that contain numeric data. In chapter 9 query number 11 is the example of how to use AVG function. With this query I wanted to find out the average cost of hire contracts for each branch.

7.3 View

A view is a virtual table. In most implementations a view has no independent physical existence. View retrieves some specific information from the tables in which you don’t want to look at everything, only some specific columns and rows. We need view to be able to mix the information from one or more tables and create it into a new single temporary table.

7.4 Updating views

After we have created a table, that table is automatically capable of accommodating insertions, deletions and updates. Views, on the other hand, don’t necessarily exhibit the same capability. If we update a view, we are actually updating its underlying table, not the temporary view table alone.

7.5 Adding new data

Every database table starts out empty. After creating a table, either by using SQL’s DDL or a RAD (Rapid application development) tool, that table is nothing but a structured shell, containing no data. To make the table useful, you must put some data into it. You may or may not have that data already stored in digital form. You might have to use keyboard to enter the data one record at a time

8 PROVIDING DATABASE SECURITY

When dealing with databases it is necessary to protect them from harm or misuse. If unauthorized person gets access to a database and deletes or changes information there it will be permanently lost to everyone. The person in charge of a database has the power to determine who can access that database and also the level of access a user receives. That person can selectively grant and revoke access to certain aspects of the system and can even grant to and revoke from someone else the right to grant and revoke such access privileges. If you use them correctly, the security tools that SQL provides are powerful protectors of important data.

With databases you should also remember all the basic security tips and rules that apply to all computer users. Passwords shouldn’t be easy to break and you should have a proper firewall. There are also many security issues that concern only databases. For example you should never give anyone else access to the user table. If a person gets more access than they need they could for example accidentally or by purpose delete a whole table with the drop command. All data types entered in the database should be protected. Even if data in the database is meant for all the public to view it needs to be protected against attacks that could stop other users from using it, another type of attack this kind of database could face is someone trying to change the data in it.

There are various ways to check and test whether the security of your database is good enough. It is also possible to check if the data sent by MySQL through internet is unencrypted, if this is the case it is visible for anyone who has the ability and time to use it in their own purposes.

Used incorrectly security measures can also be frustrating impediments to the effort of legitimate users trying to do their work. For example If you as a database administrator don’t remember to give access to all users that need it for all the database tables they require it will make their work difficult. MySQL has its own security based on Access Control Lists for all operations the users do.

8.1 Database administrator

The highest authority for a database is the database administrator. The database administrator has all right and permission to all aspects of the database. Being a database administrator can give you real power, but the position is also a great responsibility. With all that power you can easily mess up your database and destroy hundreds of hours of work. Database administrators must think clearly and carefully about the consequences of every action they perform.

The best way to become a database administrator is to install the database management system yourself. If you do, the installation manual gives you an account or login, and a password. That password identifies you as a specially privileged user. Sometimes the system calls this privileged user the database administrator, sometimes system administrator. As your first official act after logging in, you should change your password. If you don’t change the password, anyone who reads the manual can also log in with full database administrator authority. After you have changed the password only people who know the new password can log in as a database administrator.

It is advisable that we log in as database administrator if we have database administrator permission only if there is a need to perform a specific task that requires database administrator permission. After you have done it log out. For a specific limited task, log in by using your own personal login id and password. This method can save us from making mistakes that may have serious consequences for other users’ tables and for your own as well.

8.2 Database object owners

Another class of permitted users, along with the DBA, is the database object owner. Tables and views are examples of database objects. Any user who creates such an object can specify its owner. A table owner enjoys every possible access associated with that table, including the privilege to grant access of that table to other people. But these object owners have no automatic right to manipulate or enter to objects owned by other people unless they've especially been given permission to do so. Database administrator has greater rights than the object owner.

8.3 Granting privilege to users

Database administrator has all privileges on all objects in the database. The person who creates an object definitely has all privileges to that object. No one else has any privileges with respect to any object, unless someone who already has those privileges and the authority to pass them on specially grants those privileges to another person. We give permission to someone else to access the databases by using the GRANT statement. REVOKE statement can remove any privileges given to someone. SHOW GRANTS statement will show which users have privileges to parts or all of the database. You can also limit the operations a user does with the database, for example you can limit the amount of queries someone does in an hour.

9 PROJECT

In this project chapter I will show some example queries for all the tables at the database and show the data they contain. Queries 1-6 retrieve all the rows in each of the six tables I have created. Queries 7-10 are more complicated queries from the database. In some of them I have shortened the names of the tables to just their first letters. In query 7. I have retrieved details about the structure of hirecontract table. Query 8. shows all the staff that have “staffname” beginning with an S that work in the Liverpool branch. Query 9. is a list of all the staff that have a name ending in an S.

Query 10. retrieves all the hirecontracts where the overall cost is higher than a certain amount. This number could be any, I have chosen 118,13 as an example.

Query 1. All the rows from branch table.

Showing rows 0 - 4 (5 total, Query took 0.0008 sec)

SQL-query:

```
SELECT *
FROM `branch`
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh

←T→			Branchno	Branchname	Branchadd	Conutry_&_ city	B_Ttel_no
☐	✎	✗	1	BEST HIRE	36, RAVIRADANTIE	FINLAND, MIKKELI	0235474948
☐	✎	✗	2	COMFORT HIRE	23, JIM ROAD	NIGERIA, LAGOS	38497679292
☐	✎	✗	3	LOWRATE HIRE	78, BOOM STREET	UK, LONDON	73658837
☐	✎	✗	4	LUCKY HIRE	67, NOKIA STREET	USA, NEW YORK	326826738
☐	✎	✗	5	EASY HIRE	45, OLOWO RAOD	SPAIN, BARCELONA	63674883929

Query 2. All the rows from staff table.

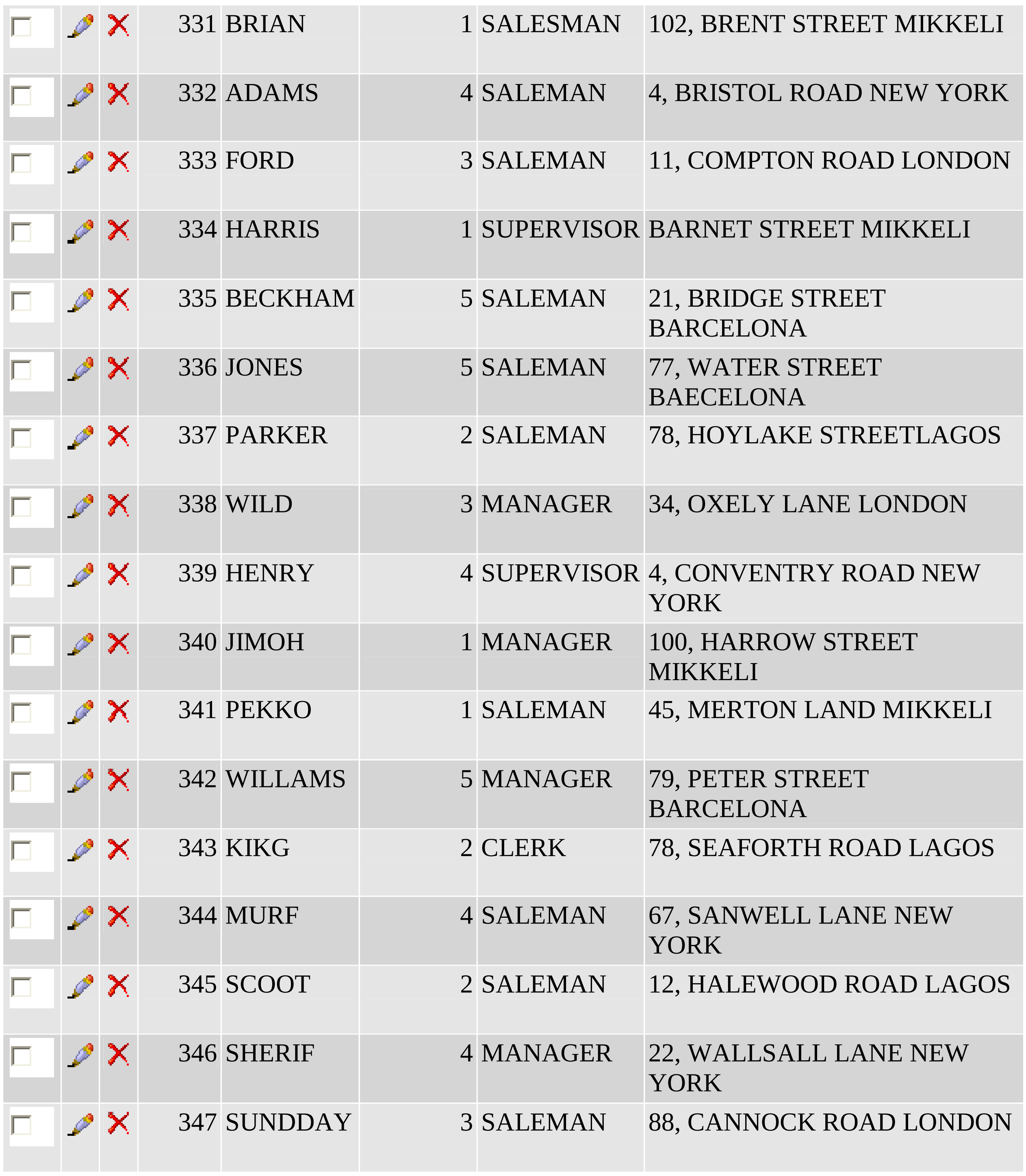
Showing rows 0 - 17 (18 total, Query took 0.0287 sec)

SQL-query:

```
SELECT *
FROM `staff`
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			Staffid	Staffname	Branchno	Jobtitle	S_address
☐	✎	✗	330	SMITH	2	MANAGER	78, WALLASEY STREET LAGOS



☐	✎	✗	331	BRIAN	1	SALESMAN	102, BRENT STREET MIKKELI
☐	✎	✗	332	ADAMS	4	SALEMAN	4, BRISTOL ROAD NEW YORK
☐	✎	✗	333	FORD	3	SALEMAN	11, COMPTON ROAD LONDON
☐	✎	✗	334	HARRIS	1	SUPERVISOR	BARNET STREET MIKKELI
☐	✎	✗	335	BECKHAM	5	SALEMAN	21, BRIDGE STREET BARCELONA
☐	✎	✗	336	JONES	5	SALEMAN	77, WATER STREET BAECELONA
☐	✎	✗	337	PARKER	2	SALEMAN	78, HOYLAKE STREETLAGOS
☐	✎	✗	338	WILD	3	MANAGER	34, OXELY LANE LONDON
☐	✎	✗	339	HENRY	4	SUPERVISOR	4, CONVENTRY ROAD NEW YORK
☐	✎	✗	340	JIMOH	1	MANAGER	100, HARROW STREET MIKKELI
☐	✎	✗	341	PEKKO	1	SALEMAN	45, MERTON LAND MIKKELI
☐	✎	✗	342	WILLAMS	5	MANAGER	79, PETER STREET BARCELONA
☐	✎	✗	343	KIKG	2	CLERK	78, SEAFORTH ROAD LAGOS
☐	✎	✗	344	MURF	4	SALEMAN	67, SANWELL LANE NEW YORK
☐	✎	✗	345	SCOOT	2	SALEMAN	12, HALEWOOD ROAD LAGOS
☐	✎	✗	346	SHERIF	4	MANAGER	22, WALLSALL LANE NEW YORK
☐	✎	✗	347	SUNDDAY	3	SALEMAN	88, CANNOCK ROAD LONDON

Query 3. All the rows from customer table.

Showing rows 0 - 14 (15 total, Query took 0.0007 sec)

SQL-query:
```
SELECT *
FROM `customer`
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			**Customerno**	**Customername**	**Customeradd**	**Customertel**
☐	✎	✗	11	E ELENIYAN	11, VICTORIA LANE LONDON	576674654760
☐	✎	✗	12	M MAFOLUKU	84, PARK STREET WOLVERHAMPTON	7655333426657
☐	✎	✗	13	R SAW	283, COMPTON LANE WOLVERHAMPTON	24356876890
☐	✎	✗	14	S SENIOR	29, SNOW HILL LIVERPOOL	643658875886
☐	✎	✗	15	P PETER	4, MAPLE LIVERPOOL	6444217899090
☐	✎	✗	16	L KAI	6, FLAT8 RIVER ROAD MANCHESTER	546546765875
☐	✎	✗	17	Y PEKKO	50, LEE ROAD NORWICH	235676557758
☐	✎	✗	18	A OLALEYE	19 CLERK LANE LONDON	445765476657
☐	✎	✗	19	M BEN	65 OLOWODE NEW YORK	33576544644
☐	✎	✗	20	J OKOCHA	40, OAK STREET WOLVERHAMPTON	455687654537
☐	✎	✗	21	F PIN	89, MAIN ROAD WALSALL	546778890001
☐	✎	✗	22	W COLE	95, LEAH LANE LIVERPO	657546554767
☐	✎	✗	23	N NOKIA	20, NEW STREET BIRMINGHAM	546546587655
☐	✎	✗	24	G LEKE	120, ALCORN AVENUE BIRMINGHAM	1232475769866
☐	✎	✗	25	D EMANUEL	55, PENHILL MANCHESTER	354565664846

Query 4. All the rows from car table.

Showing rows 0 - 20 (21 total, Query took 0.0006 sec)

SQL-query:

```
SELECT *
FROM `car`
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			Carregno	cartypeno	carmodel	Colour	Mileageno
☐	✎	✗	WA7 BOP	30	FOCUS	SILVER	57367
☐	✎	✗	N34 SAV	20	HUMMER	WHITE	127587
☐	✎	✗	Z72 ONK	20	HUMMER	BLUE	475739
☐	✎	✗	K77 BLS	50	ASTAL	BLUE	675946
☐	✎	✗	HH33 HKJ	10	TRANSIT	BLUE	647836
☐	✎	✗	JI56 SJY	10	TRANSIT	RED	778773
☐	✎	✗	MO5 SOE	30	FOCUS	BLACK	684763
☐	✎	✗	SQ78 JUR	50	ASTRA	RED	348637
☐	✎	✗	L98 DTQ	40	L200	YELLOW	66746
☐	✎	✗	M89 HYT	30	PUMA	BLACK	43537
☐	✎	✗	OR45 SSP	10	BRAVO	RED	65775
☐	✎	✗	AC34 UUT	30	CROMER	WHITE	57469
☐	✎	✗	E57 BLT	40	OMEGA	GREEN	63638
☐	✎	✗	PT23 RIS	40	OMEGA	RED	647944
☐	✎	✗	Y19 TDD	10	GALAXY	BLACK	225735
☐	✎	✗	PT2 RIS	40	OMEGA	RED	118848
☐	✎	✗	OI9 TDD	10	GALAXY	BLACK	43683
☐	✎	✗	WW99 EST	50	ASTRA	BLUE	257847
☐	✎	✗	EE60 TTR	30	PUMA	RED	123768

☐	✎	✗	Z34 SOK	40	L200	BLUE	87262
☐	✎	✗	S77 AAI	30	FOCUS	RED	54674

Query 5. All the rows from type table.

Showing rows 0 - 4 (5 total, Query took 0.0005 sec)

SQL-query:
SELECT *
FROM `type`
LIMIT 0 , 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			Cartypeno	Typedescription	Dailyrate
☐	✎	✗	10	VAN	99.99
☐	✎	✗	20	LIMOUSINE	90.78
☐	✎	✗	30	CAR	50.6
☐	✎	✗	40	PICK UP	60.98
☐	✎	✗	50	LORRY	70.7

Query 6. All the rows from hirecontract table.

SQL-query:
SELECT *
FROM `hirecontract`
LIMIT 0 , 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			Hirecontractno	Carregno	Customerno	Staffid	Cartypeno	Issueddate	Duereturndate	Actualreturndate	Carcost
☐	✎	✗	211	SQ78 JUR	14	345	50	2009-04-06	2009-04-07	2009-04-07	60.99
	✎	✗	111	OI9	25	341	10	2008-	2008-01-	2008-01-18	299.

☐				TDD				01-15	18		97
☐	✎	✗	511	L98 DTQ	12	336	40	2008-08-27	2008-08-28	2008-08-28	60.98
☐	✎	✗	411	OR45 SSP	18	332	10	2008-05-30	2008-06-07	2008-06-07	799.92
☐	✎	✗	412	N34 SAV	21	332	20	2008-05-18	2008-05-20	2008-05-20	181.56
☐	✎	✗	311	K77 BLS	22	333	50	2008-10-01	2008-10-05	2008-10-05	282.8
☐	✎	✗	112	M89 HYT	11	341	30	2008-09-10	2008-09-11	2008-09-11	50.6
☐	✎	✗	413	MO5 SOE	15	332	30	2008-05-08	2008-05-13	2008-05-13	253
☐	✎	✗	212	E57 BLT	23	345	40	2008-05-13	2008-05-17	2008-05-17	243.92
☐	✎	✗	201	PT2 RIS	23	330	40	2008-10-13	2008-10-15	2008-10-15	121.96
☐	✎	✗	209	E57 BLT	25	343	40	2008-05-27	2008-05-28	2008-05-29	121.96
☐	✎	✗	208	AC34 UTT	14	330	30	2008-05-02	2008-05-03	2008-05-03	50.6
☐	✎	✗	207	SQ78 JUR	23	337	50	2008-04-18	2008-04-21	2008-04-21	212.1
☐	✎	✗	206	JI56 SJY	14	337	10	2008-05-30	2008-05-31	2008-05-31	99.99
☐	✎	✗	205	Z72 ONK	25	337	20	2008-12-22	2008-12-23	2008-12-23	90.78
☐	✎	✗	310	HH33 HKJ	22	333	10	2008-08-13	2008-08-20	2008-08-20	699.93
☐	✎	✗	309	K77 BLS	22	338	50	2008-01-08	2008-01-09	2008-01-09	70.7
☐	✎	✗	308	WA7 BOP	13	333	30	2008-07-03	2008-07-13	2008-07-13	506
☐	✎	✗	307	K77 BLS	19	333	50	2008-09-13	2008-09-14	2008-09-15	141.4
☐	✎	✗	306	WA7 BOP	13	338	30	2008-05-03	2008-05-04	2008-05-04	50.6
☐	✎	✗	305	HH33	13	347	10	2008-	2008-05-	2008-05-13	299.

				HKJ				05-10	13		97
☐	✎	✗	410	N34 SAV	16	346	20	2008-11-15	2008-11-16	2008-11-16	90.78
☐	✎	✗	409	OR45 SSP	18	339	10	2008-05-12	2008-05-13	2008-05-13	99.99
☐	✎	✗	408	MO5 SOE	16	346	30	2008-05-26	2008-05-29	2008-05-30	202.4
☐	✎	✗	407	MO5 SOE	21	344	30	2008-05-08	2008-08-13	2008-08-13	253
☐	✎	✗	406	N34 SAV	18	339	20	2008-10-10	2008-10-11	2008-10-11	90.78
☐	✎	✗	405	OR45 SSP	15	344	10	2008-01-03	2008-01-04	2008-01-04	99.99
☐	✎	✗	510	L98 DTQ	12	336	40	2008-05-03	2008-05-05	2008-05-05	121.96
☐	✎	✗	509	Y19 TDD	17	342	10	2008-05-18	2008-05-19	2008-05-19	99.99
☐	✎	✗	508	WW99 EST	24	336	50	2008-04-23	2008-04-28	2008-04-28	353.5

Query 7. Structure of hirecontract table.

SQL-query:
DESC HIRECONTRACT

[Edit] [Create PHP Code]

Field	Type	Null	Key	Default	Extra
Hirecontractno	int(4)		PRI	0	
Carregno	varchar(10)		MUL		
Customerno	int(3)		MUL	0	
Staffid	int(3)		MUL	0	
Cartypeno	varchar(5)		MUL		
Issuedate	date			0000-00-00	
Duereturndate	date			0000-00-00	
Actualreturndate	date			0000-00-00	
Carcost	float			0	

Query 8. Staff whose name begins with an S that work in the Liverpool branch.

SQL-query:

```
SELECT staffid, staffname, jobtitle, city
FROM staff, branch
WHERE staff.branchno = branch.branchno
AND staffname LIKE 'S%'
AND city = 'NIGERIA, LAGOS'
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

staffid	staffname	jobtitle	city
330	SMITH	MANAGER	NIGERIA, LAGOS
345	SCOOT	SALEMAN	NIGERIA, LAGOS

Query 9. A list of staff with “staffname” ending in an ‘S’ in all departments.

SQL-query:

```
SELECT staffid, staffname, jobtitle, city
FROM staff s, branch b
WHERE s.branchno = b.branchno
AND staffname LIKE '%S'
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

staffid	staffname	jobtitle	city
334	HARRIS	SUPERVISOR	FINLAND, MIKKELI
332	ADAMS	SALEMAN	USA, NEW YORK
336	JONES	SALEMAN	SPAIN, BARCELONA
342	WILLAMS	MANAGER	SPAIN, BARCELONA

Query 10. List of all hire agreements where the overall cost is more than £118.13.

SQL-query:

```
SELECT Hirecontractno, Issuedate, Staffid, Carcost AS OVERALL_COST
FROM hirecontract
WHERE Carcost > 118.13
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

←T→			Hirecontractno	Issuedate	Staffid	OVERALL_COST
☐	✎	✗	111	2008-01-15	341	299.97
☐	✎	✗	411	2008-05-30	332	799.92
☐	✎	✗	412	2008-05-18	332	181.56
☐	✎	✗	311	2008-10-01	333	282.8
☐	✎	✗	413	2008-05-08	332	253
☐	✎	✗	212	2008-05-13	345	243.92
☐	✎	✗	201	2008-10-13	330	121.96
☐	✎	✗	209	2008-05-27	343	121.96
☐	✎	✗	207	2008-04-18	337	212.1
☐	✎	✗	310	2008-08-13	333	699.93
☐	✎	✗	308	2008-07-03	333	506
☐	✎	✗	307	2008-09-13	333	141.4
☐	✎	✗	305	2008-05-10	347	299.97
☐	✎	✗	408	2008-05-26	346	202.4
☐	✎	✗	407	2008-05-08	344	253
☐	✎	✗	510	2008-05-03	336	121.96
☐	✎	✗	508	2008-04-23	336	353.5

Query 11. Counts the average cost of contracts in each branch.

SQL-query:

```
SELECT CITY, AVG( Carcost ) AS AVGCOST_OF_HIRECONTRACT
FROM hirecontract H, staff S, branch B
WHERE H.Staffid = S.Staffid
AND S.branchno = B.branchno
GROUP BY City
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

CITY	AVGCOST_OF_HIRECONTRACT
FINLAND, MIKKELI	87.642499923706
NIGERIA, LAGOS	125.28749990463
SPAIN, BARCELONA	127.2859992981
UK, LONDON	293.05713871547
USA, NEW YORK	230.15777418349

10 CONCLUSIONS

I supposed that this is a work for an automobile dealership. I wanted to create an easier way to keep their data and to get it more accessible for all the staff. This included an inventory of all the vehicles they have in stock. They want to track such facts as registration, colour, model, and options on each vehicle so that they can know what there is available for rent. The database I created works well for the purposes of this company. I worked hard to create all the necessary tables and to put all the data in them. I connected all the six tables together with suitable primary keys. Once this creation of database was done it is easy to add and change information there. Before they had all their important information in old fashioned folders in paper form, the new system can reduce greatly the time it takes to find a specific piece of information or documentation. Once staff learns to use the database correctly they make less mistakes than before. Because this is a global company they could share information about each branch faster and it can reduce the cost of the business.

This project might have been done better if I had had the opportunity to use Oracle 11g which is the newest version of the language. There are some statements which MySQL does not understand but if used in Oracle would have made my work better.

11 REFERENCES

(1) **Taylor, Allen,** SQL For Dummies, 3rd edition, IDG Books Worldwide, 1998

(2) **Powell, Gavin,** Beginning Database Design, Wiley Publishing, Inc. 2006.

(3) **SQL**, [Web page], <http://en.wikipedia.org/wiki/SQL/>, 12.06.2009

(4) **SQL Introduction,** [Web page], <http://www.w3schools.com/SQL/sql_intro.asp/>, 30.06.2009

(5) **Databasics: A Database Dictionary**, [Web page], 2008, <http://www.geekgirls.com/database_dictionary.htm/>, 01.02.2010

(6) **SQL Tutorial**, [Web page], 2004, <http://www.sql-tutorial.net/sql-database-table.asp/>, 01.02.2010

(7) **Integer data type,** [Web page], <http://cnx.org/content/m18654/latest/>, 09.03.2010

(8) **MySql reference manual,** [Online], <http://dev.mysql.com/doc/refman/5.5/en/index.html>, 11.04.2010

(9) **Wikipedia – MySQL**, [Web page], <http://en.wikipedia.org/wiki/Mysql>, 01.06.2010

(10) **Wikipedia – Primary Key**, [Web page], <http://en.wikipedia.org/wiki/Primary_key>, 16.04.2010

(11) **Wikipedia – Foreign Key,** [Web page], <http://en.wikipedia.org/wiki/Foreign_key>, 16.04.2010

(12) **Wikipedia – Normalization,** [Web page], <http://en.wikipedia.org/wiki/Database_normalization>, 16.04.2010

(13) **MySQL, [Web page],** <http://www.mysql.com/about/>, 20.05.2010

Unpublished source

(14) **Study guide 2006 - 2007.** University of Wolverhampton England.