Databaser

Mandatory assignment
Software Design

Computer Science

Carsten Lydeking

Zealand Business College

Zealand
Academy of Technologies and Business

Databaser

Mandatory assignment
Software Design

Carsten Lydeking

cal002@edu.zealand.dk

Teacher:	M. Lynggaard Krarup
Report Deadline:	03/05/24
Handed-in:	18/03/24
Faculty:	Computer Science

Cover:	Generated image of binary using DALL-E
Style:	ZBC template – created by Carsten Lydeking

Zealand
Academy of Technologies and Business

Contents

Nomenclature

0.1. Abbreviations and definitions

The following abbreviations and definitions are used throughout the report:

Abb.	Definition
DB	Database: A structured collection of data stored electronically.
RDB	Relational Database: A type of DB that stores and provides access to data points that are related to one another.
DBMS	Database Management System: Software that handles the storage, retrieval, and updating of data in a database.
RDBMS	Relational DBMS: A database management system based on the relational model introduced by E.F. Codd.
SQL	Structured Query Language: A programming language used to manage and manipulate relational databases.
CRUD	Create, Read, Update, Delete The four basic operations of persistent storage: adding, retrieving, modifying, and removing data.
GUI	Graphical User Interface A user interface that allows users to interact with electronic devices using graphical icons and visual indicators.
PK	Primary Key:A unique identifier for each record in a database table.
FK	Foreign Key: A field in a database table that links to the primary key of another table.
NML	Normalization: In relation to RDB design, the process of organizing the columns (attributes) and tables (relations) to minimize data redundancy.
1NF	First Normal Form: A stage of NML where each table has atomic (i.e. indivisible) values and each record needs to be unique.
2NF	Second Normal Form: A stage of NML where it meets all the requirements of the 1NF and does not have partial dependency.

3NF	Third Normal Form: A stage of NML where it meets all the requirements of the 2NF and has no transitive functional dependencies.
ERD	Entity Relationship Diagram: A graphical representation of entities and their relationships to each other, typically used in database design.
UML	Unified Modeling Language: A standardized modeling language used to specify, visualize, construct, and document the artifacts of software systems.
DDL	Data Definition Language: A subset of SQL used to define database structures, such as tables, schemas, and databases.
DML	Data Manipulation Language: A subset of SQL used for adding (inserting), deleting, and modifying (updating) data in a database.
DCL	Data Control Language: A subset of SQL used to control access to data in a database.
TCL	Transaction Control Language: A subset of SQL used to manage the changes made by DML statements.
XML	Extensible Markup Language: A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
JSON	JavaScript Object Notation: A lightweight data-interchange format that is easy for humans to read and write, and for machines to parse and generate.

1 Introduction

1.1. The Report
This report is written in LaTeXin VS Code with the LaTeXWorkshop extension. A template has also been prepared for use by other students at Zealand. It is expected that this will be made available on GitHub at a later date. As a strong opposer of *Danglish*, I have chosen to write in english, as it is the base language for the applied software scripts, programming languages and terminology.

1.2. Used Tools
- LaTeX- For writing the report
- Visual Studio Code v. 1.60.2 - With LaTeXWorkshop extension for editing
- Visual Studio 2022 Enterprise Edition v. 17.8.0 - For application development
- GitHub - For version control
- Microsoft SQL Server Management Studio v. 19.3 - For database development
- Microsoft SQL Server 2022 - For hosting the database

1.3. Purpose
This report has been prepared as part of the curriculum for Software Design at Zealand. The report deals with a case where a database is to be developed for a fictitious company. In addition, an application is to be developed that can access the database. Furthermore an application is to be developed that can access the database.

1.4. The assigmnet
Has been copied verbatim from Moodle and is as follows:

1.4.1. Afleveringsformat
Følgende uploades i WiseFlow:

- En rapport på PDF format
- En reference til jeres kode uploades sammen med rapporten enten som ZIP-fil eller med en reference til GitHub i rapporten.

1.4.2. Opgaven:

1. Lav en database model (Hint: beskriv attributes, relationer, multiplicitet, primary keys (PK) og foreign keys (FK)).
2. Redegør for, at dine tabeller er på 3je Normalform.
3. Lav et SQL script over, hvorledes du kan oprette de nødvendige tabeller.
4. Lav din database, så den indeholder tabellerne.
5. Indsæt nogle passende værdier.
6. Lav en Windows Form Applikation (eller console applikation), så den kan lave CRUD på faciliteter.

Besvarelsen skal indeholde de nødvendige oplysninger, så jeres lærer kan afprøve jeres applikation.

2
The database

2.1. Database Model

The logic behind the database is to keep track of reservations, customers, hotels, rooms and facilities. The database is designed to handle reservations of rooms at hotels, and to keep track of customer information and what facilities the hotels offer.

This results in the database consisting of the following tables:

- Hotels
- Customers
- Rooms
- Reservations
- Facilities
- HotelFacilities

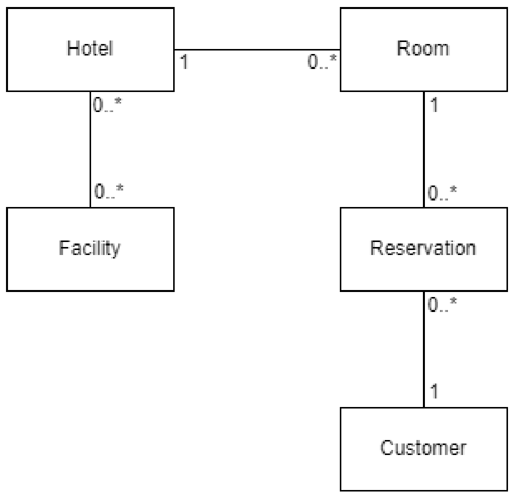
2.1.1. Relations and Multiplicity

- A Hotel can have many Rooms (1 to Many)
- A Customer can have many Reservations (1 to Many)
- A Room can be part of one Reservation at a time (1 to Many)
- Facilities are shared among Hotels through HotelFacilities (Many to Many)

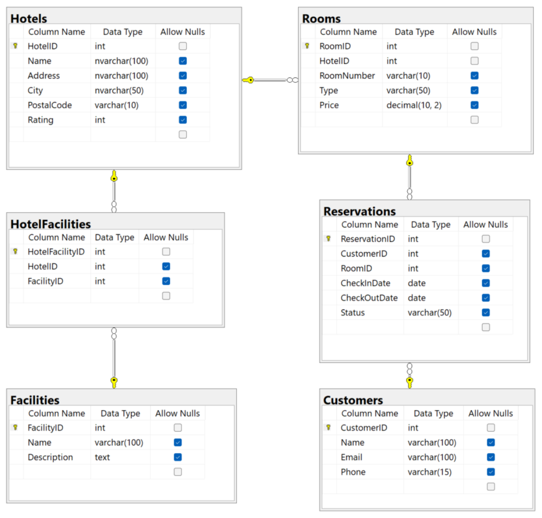
2.1.2. 3NF State

All tables are in 3NF because:

1. They are in 1NF as all attributes are *atomic* - no groups are repeated or contain more than one value. This can be seen, for example, by HotelFacilities being a relationship table between Hotel and Facility.
2. They are in 2NF as there is no *partial dependency* - Non-key columns depend solely on the PK. This can be seen, for example, by Hotel having a relationship to Facility instead of containing Facility as an attribute.
3. There are no *transitive dependencies* - Non-key columns depend only on the PK. This can be seen, for example, by Room having a relationship to Hotel instead of



(a) Domain Model Diagram



Hotels

	Column Name	Data Type	Allow Nulls
🔑	HotelID	int	☐
	Name	nvarchar(100)	✓
	Address	nvarchar(100)	✓
	City	nvarchar(50)	✓
	PostalCode	varchar(10)	✓
	Rating	int	✓
			☐

Rooms

	Column Name	Data Type	Allow Nulls
🔑	RoomID	int	☐
	HotelID	int	☐
	RoomNumber	varchar(10)	✓
	Type	varchar(50)	✓
	Price	decimal(10, 2)	✓
			☐

HotelFacilities

	Column Name	Data Type	Allow Nulls
🔑	HotelFacilityID	int	☐
	HotelID	int	✓
	FacilityID	int	✓
			☐

Reservations

	Column Name	Data Type	Allow Nulls
🔑	ReservationID	int	☐
	CustomerID	int	✓
	RoomID	int	✓
	CheckInDate	date	✓
	CheckOutDate	date	✓
	Status	varchar(50)	✓
			☐

Facilities

	Column Name	Data Type	Allow Nulls
🔑	FacilityID	int	☐
	Name	varchar(100)	✓
	Description	text	✓
			☐

Customers

	Column Name	Data Type	Allow Nulls
🔑	CustomerID	int	☐
	Name	varchar(100)	✓
	Email	varchar(100)	✓
	Phone	varchar(15)	✓
			☐

(b) Entity Relationship Diagram

Figure 2.1: Domain Model and Entity Relationship Diagrams

containing Hotel as an attribute.

Diagrams

The diagrams should be simple enough to be self-explanatory. The DMD is created first and based on the many-to-many relationship between hotels and facilities, it gives rise to an index table (HotelFacilities). In the DMD, HotelFacilities is not considered as an independent entity but rather as a way to manage the relationship between Hotel and Facilities.

2.1.3. Domain Model Diagram

See Figure 1. This DMD only contains entities, multiplicities, and relationships. This provides input on how the database should be designed based on the individual tables and their relationships to each other.

2.1.4. Entity-Relationship Diagram

See Figure 1. The ERD is generated from the tables in the database and shows the relationships between the tables, their PK, FK, and data types.

3
WinForm Application

3.1. Considerations regarding the application

3.1.1. Scope

Based on the task description and my interpretation of the intention behind the task, the application is the tertiary priority - implying that the database is the primary focus, and diagrams and arguments for documentation are secondary. Therefore, I have narrowed down the task to an application that can perform CRUD operations on the "Facilities" table in the database. This leads me to the following functionality:

- Connection to my local database
- Displaying the "Facilities" table
- Adding a record to "Facilities" using DB autoincrement
- Updating the Name and Description of a record in "Facilities"
- Deleting a record from "Facilities"

3.1.2. Out of scope

This leads me to the conclusion that I will not implement features such as adding a Facility to a Hotel through HotelFacilities. This is because it would require more work on the UI side, rather than on and in the database work.

3.1.3. Design

The application will be a simple WinForm application with a DataGridView for displaying the "Facilities" table, and a few buttons for the CRUD operations. Classic WinForm controls will be used, and the application will be designed to be as simple as possible. The reason is the same as the one for the scope of the application: the database is the primary focus, and the application is secondary.

3.1.4. However

I have added a DataGridView for the "ViewHotelFacilities", which is a view that I have created in the database. The script can be found in A.1.1. This is because I wanted to see how the relationship between the Hotel and the Facilities would be displayed in a DataGridView.

3.2. Implementation

4 Conclusion

A conclusion...

A

Example

A.1. Mock Data

This section contains the SQL code for inserting mock data into the tables. The mock data is in Danish.

A.1.1. SQL code

```sql
CREATE VIEW HotelsWithFacilities AS
SELECT
    h.HotelID,
    h.Name AS HotelName,
    f.FacilityID,
    f.Name AS FacilityName,
    f.Description
FROM
    Hotels h
JOIN
    HotelFacilities hf ON h.HotelID = hf.HotelID
JOIN
    Facilities f ON hf.FacilityID = f.FacilityID;

SELECT * FROM HotelsWithFacilities;
```