**Table of Contents**

[**Conceptual Design in Entity Relationship Diagram** 3](#_Toc193815977)

[**Entities** 3](#_Toc193815978)

[**Relationships** 3](#_Toc193815979)

[**Relational Schema** 4](#_Toc193815980)

[**Relationship and Constraint** 4](#_Toc193815981)

[**Relationship Summary** 5](#_Toc193815982)

[**Integrity Constraints** 5](#_Toc193815983)

[**Entity Relationship Diagram** 5](#_Toc193815984)

[**Partial Entity Relationship Diagram** 6](#_Toc193815985)

[**Full Diagram with all Attributes** 6](#_Toc193815986)

[**Implementation using Microsoft Access** 7](#_Toc193815987)

[**What is Microsoft Access** 7](#_Toc193815988)

[**Why Microsoft Access** 7](#_Toc193815989)

[**Installation** 7](#_Toc193815990)

[**Tables Creation for Entities** 10](#_Toc193815991)

[**Database Normal Forms (Normalization)** 11](#_Toc193815992)

[**Clients Table** 12](#_Toc193815993)

[**Employees Table** 12](#_Toc193815994)

[**Service Catalogue Table** 12](#_Toc193815995)

[**Invoices Table** 13](#_Toc193815996)

[**ClientAppointments Table** 13](#_Toc193815997)

[**ServicesProvided Table** 14](#_Toc193815998)

[**InvoiceRows Table** 14](#_Toc193815999)

[**Populating Tables with Data** 15](#_Toc193816000)

[**Structured Query Language** 16](#_Toc193816001)

[**Queries** 17](#_Toc193816002)

[**Database Application** 21](#_Toc193816003)

[**Consideration on Privacy and Security** 27](#_Toc193816004)

[**Critical Evaluation** 27](#_Toc193816005)

[**Strengths and Weaknesses of the Database Design** 27](#_Toc193816006)

[**Effectiveness of Implementation** 27](#_Toc193816007)

[**Contributions:** 28](#_Toc193816008)

[**Personal Reflection on Learning Experience** 28](#_Toc193816009)

[**Areas for Improvement and Recommendations** 29](#_Toc193816010)

[**Conclusion:** 29](#_Toc193816011)

# **Conceptual Design in Entity Relationship Diagram**

In the database, there are total 7 tables combining 4 Entities and 3 Relationship tables.

## **Entities**

**Clients table**

1. ClientID = [PK]
2. ClientName
3. ClientAddress
4. ClientEmail
5. ClientMobile

**Employees Table**

1. EmployeeID = [PK]
2. EmployeePayrollNumber
3. EmployeeName
4. EmployeeMobile

**Service Catalogue Table**

1. ServiceID = [PK]
2. ServiceName
3. ServiceDescription
4. ServiceHourlyRate

**Invoices Table**

1. InvoiceID = [PK]
2. InvoiceNumber
3. InvoiceCost
4. InvoiceDiscount
5. InvoiceTotal
6. InvoiceDate

## **Relationships**

**Client Appointments Table**

1. ClientAppointmentID = [PK]
2. AppointmentDate
3. ServiceID = [FK]
4. ClientID = [FK]

**Services Provided Table**

1. ServiceID = [FK]
2. EmployeeID = [FK]
3. ClientAppointmentID = [FK]
4. Expenses

**Invoice Rows Table**

1. InvoiceRowID = [PK]
2. InvoiceID = [FK]
3. ClientAppointmentID = [FK]

## **Relational Schema**

1. **Clients** { ClientID, ClientName, ClientAddress, ClientEmail, ClientMobile }
2. **Employees** { EmployeeID, EmployeePayrollNumber, EmployeeName, EmployeeMobile }
3. **ServiceCatalogue** { ServiceID, ServiceName, ServiceDescription, ServiceHourlyRate }
4. **Invoices** { InvoiceID, InvoiceNumber, InvoiceCost, InvoiceDiscount, InvoiceTotal, InvoiceDate }
5. **ClientAppointments** { ClientAppointmentID, AppointmentDate, *ServiceID*, *ClientID*, *EmployeeID* }
6. **ServicesProvided** { *ServiceID*, *EmployeeID*, *ClientAppointmentID*, Expenses }
7. **InvoiceRows** { InvoiceRowID, *InvoiceID*, *ClientAppointmentID* }

Represents Primary key

Represents Foreign key

## **Relationship and Constraint**

1. **One to One (1:1) Relationship:** As there is no one-to-one (1:1) relation between the entities, all have either 1:M or M:N relation in the diagram.
2. **One to many (1:M) Relationship:** This happens when one record in one entity is related to multiple records in another.
   1. **Clients → ClientAppoinments:** One client can have many appointments(One ClientAppointments)
   2. **Employees** **→** **ClientAppoinments:** A single Employee can have multiple appointments on (ClientAppointments).
   3. **ServiceCatalogue** **→** **ClientAppoinments:** One or more appointments can be booked for a Service from the Service Catalogue.
   4. **Invoices** **→** **InvoiceRows:** This means that an invoice can have zero or many invoice rows (InvoiceRows).
   5. **ClientAppoinments** **→** **InvoiceRows:** One invoice row can be attached to multiple client appointments
3. **Many to One (M:1) Relationship:** In a way these are kind of the opposite of one-to-many (1:M) relationships.
   1. Multiple appointments are associated with one client
   2. Single employee has many appointments.
   3. One service can have many appointments.
   4. One invoice has many invoice rows.
   5. Multiple invoice rows point to one client appointments.
4. **Many to Many (M:N) Relationship:** These relationships are constructed using junction tables.
   1. **Employees ⇄ ServicesProvided ⇄ Service Catalogue:** A single employee is able to provide multiple services, and a single service is being provided by multiple employees. (ServicesProvided table)
   2. **Employees ⇄ ServicesProvided ⇄ ClientAppointments:** One employee can conduct many clients appointments, and one appointment can be for many employees. (ServicesProvided table)
   3. **Service Catalogue ⇄ ServicesProvided ⇄ ClientAppointments:** And a service maybe for multiple appointments and an appointment maybe for multiple services. (ServicesProvided table)
   4. **Invoices ⇄ InvoiceRows ⇄ ClientAppointments:** One or more invoices can be associated with one or more appointments; and one or more appointments can be billed in multiple invoices. (InvoiceRows table)

### **Relationship Summary**

|  |  |
| --- | --- |
| **Type of Relationships** | **Example** |
| **1:1 (One-to-One)** | No, 1:1 Relationships |
| **1:M (One-to-Many)** | Clients → ClientAppoinments, Employees → ClientAppoinments, ServiceCatalogue → ClientAppoinments, Invoices → InnvoiceRow, ClientAppoinments → InvoiceRows |
| **M:1 (Many-to-One)** | Opposite of One-to-Many |
| **M:N (Many-to-Many)** | Employees ⇄ ServicesProvided ⇄ ServiceCatalogue, Employees ⇄ ServicesProvided ⇄ ClientAppointments, ServiceCatalogue ⇄ ServicesProvided ⇄ ClientAppointments, Invoices ⇄ InvoiceRows ⇄ ClientAppointments |

### **Integrity Constraints**

An entity integrity constraint is a rule that ensures each record in a database table is unique, usually enforced by primary keys and UNIQUE constraints.

1. Foreign Key Constraints: Enforce referential integrity between related tables.

## **Entity Relationship Diagram**

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. They are used to design relational database management systems.

### **Partial Entity Relationship Diagram**

A group of white rectangles with black text

AI-generated content may be incorrect.

### **Full Diagram with all Attributes**

A black background with white text

AI-generated content may be incorrect.

# **Implementation using Microsoft Access**

In this Appointment Scheduling for a Service-Based Small & Medium Enterprise (SME) Application, we are using Microsoft Access to build the database, tables and a Graphical User Interface.

## **What is Microsoft Access**

Microsoft Access is a database management system (DBMS) from Microsoft that combines the relational Access Database Engine with a graphical user interface and software-development tools, allowing users to create and manage databases particularly for small to medium-sized businesses and individual users, offering a user-friendly interface and integration with other Microsoft Office applications.

### **Why Microsoft Access**

There are several advantages using it.

1. **Ease of Use:** Access is known for its easy-to-understand interface, making it relatively easy for non-technical users to create and manage databases, including tables, forms, queries, and reports.
2. **Cost-Effectiveness:** Compared to enterprise-level database systems like SQL Server or Oracle, Access is a more affordable option, especially for small businesses and individual users.
3. **Templates and Wizards:** Access offers pre-built templates and wizards that can be customized to fit specific needs, allowing users to quickly create databases and applications.

### **Installation**

Microsoft Access comes with Microsoft 365 Suite. So to install Access we need to install MS 365 suite. Goto [Microsoft 365 official website](https://www.microsoft.com/en-us/microsoft-365/download-office#download) and download the installation file for Windows/MacOS.

A black rectangular object with a line

AI-generated content may be incorrect.

Once it is downloaded, we need to open the file to start the installation process. The installation process needs administrator privileges, so make sure you have the admin permissions.

A screenshot of a computer

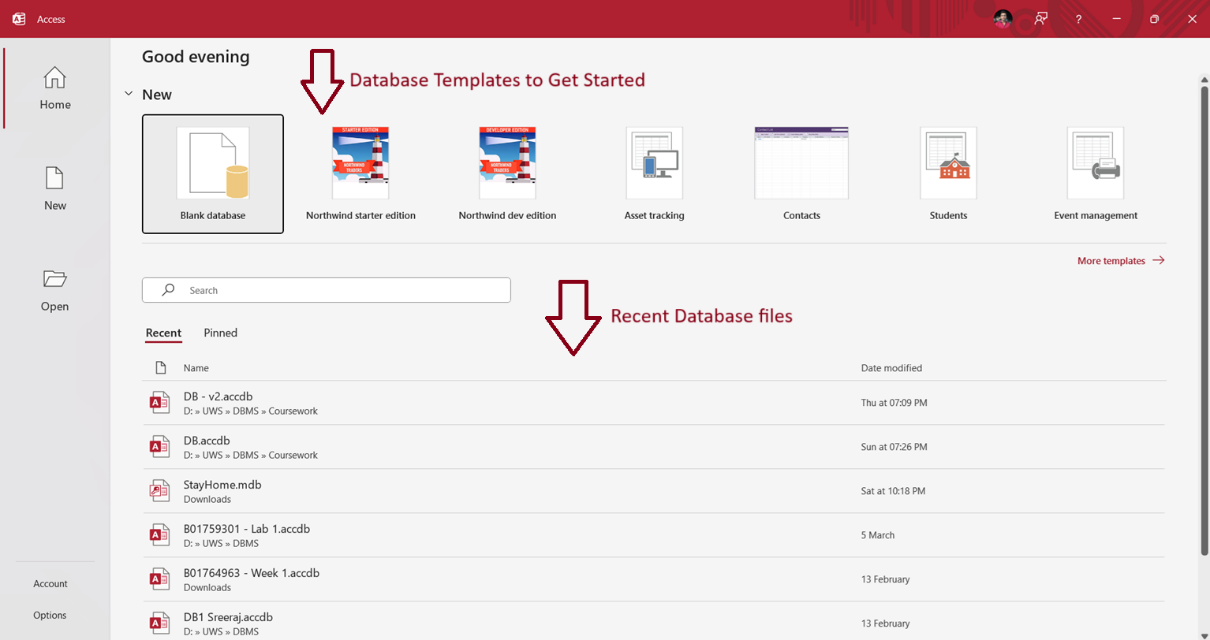
AI-generated content may be incorrect.

Once the installation file opened, please wait for couple of minutes until the installation finishes. After completion of installation, we are ready to use Access.

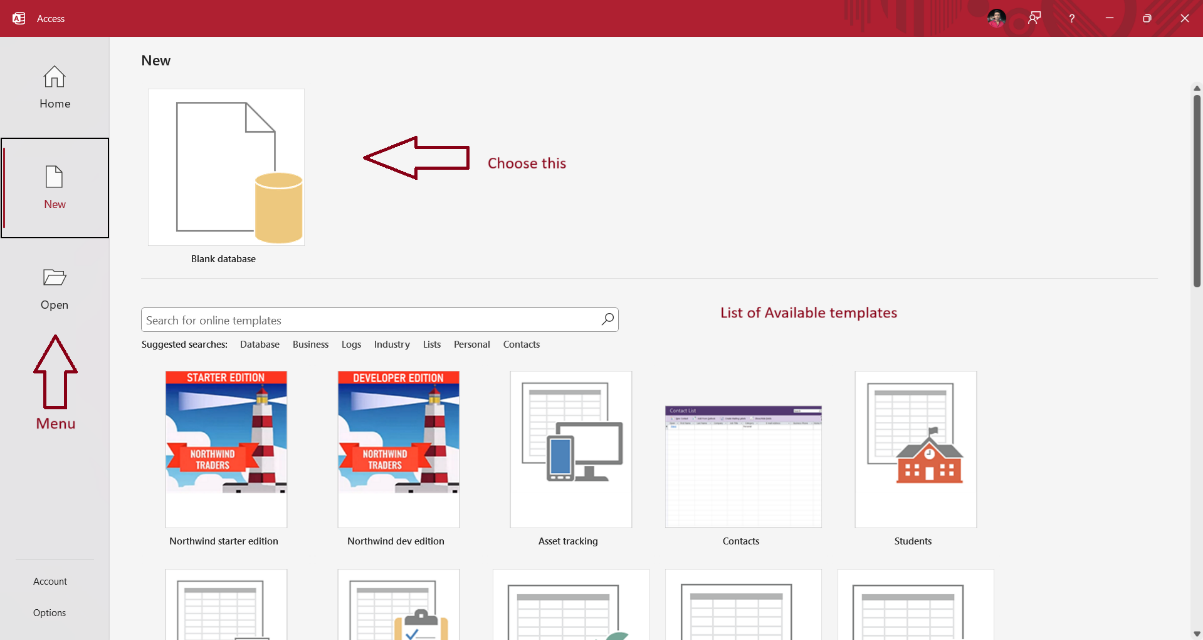
A screenshot of a computer

AI-generated content may be incorrect.

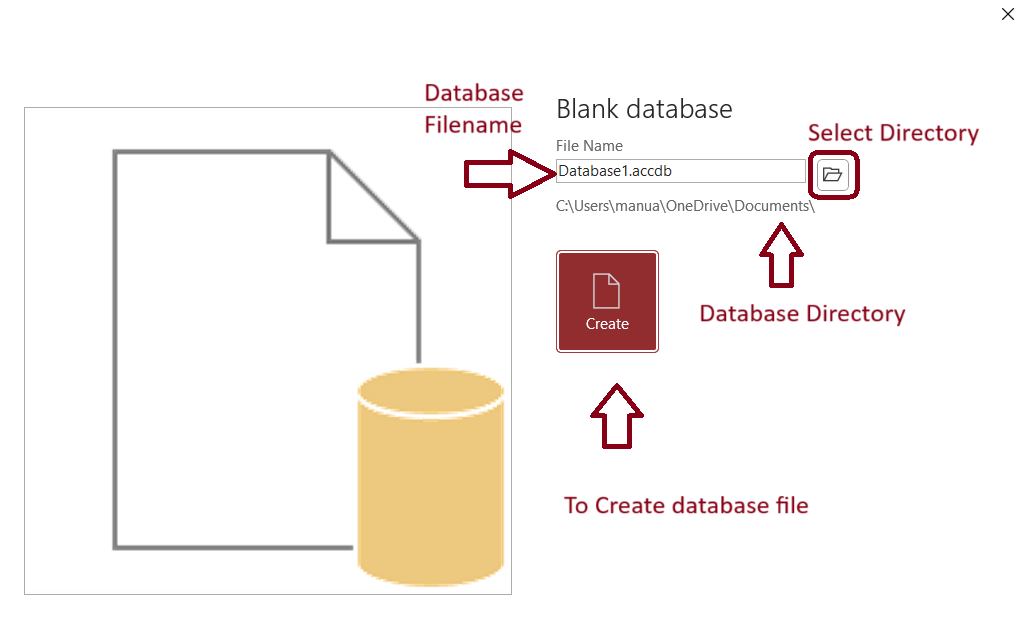
This is the icon for Microsoft Access, after opening you are presented with this user interface.



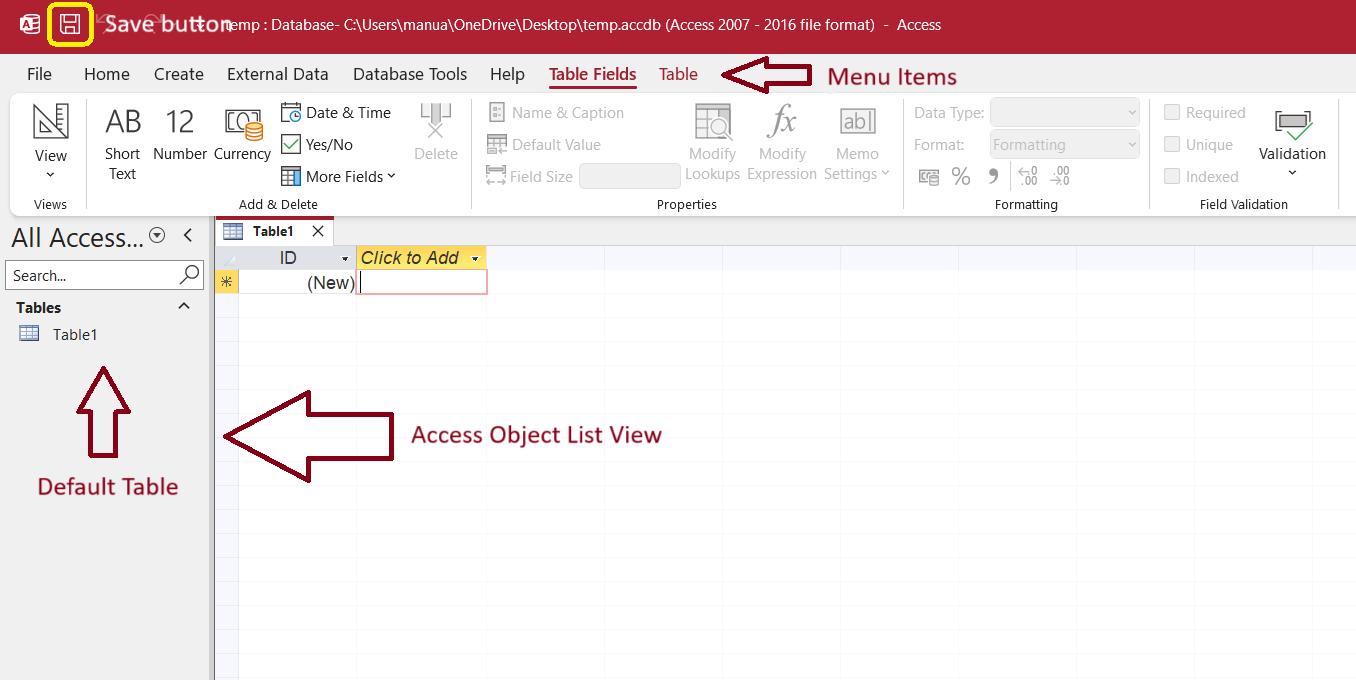
In this window, we can see there are option to select Blank Database, or we can use the pre-made templates to quickly get started with. The 2nd half of the window shows recent files that are previously opened. For fresh installations, you won’t see anything, but in my pc I already opened databases previously so I get a list of databases to open.



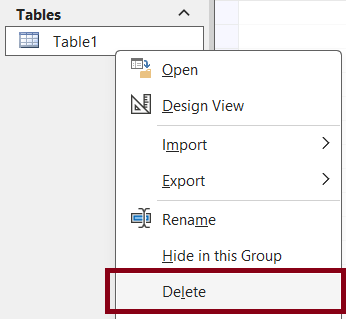
In the menu bar on left, click the new menu item to open a new window that shows a list of templates to choose from or we start our own from scratch using Blank database option. So, we are using Blank database.



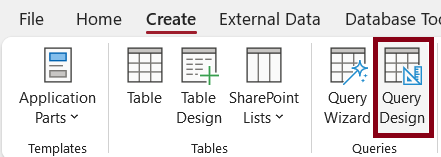
A popup window opens that asks few information, Name of the database file, where to store the file in disk. Once everything is done, we can click Create button to create blank database.



After creating the database, the file will be opened automatically. And a new window will be presented with a lot of menus and options. On the top of the window, we have Save button to save any changes made to the database. Below that, we have menu tab items. And on left side of the window, we have Access Object List Viewer which lists all the objects related to Access such as Tables, Forms, Reports, Queries etc. The main window that is focused on is Data sheet viewer of the table. Once a new database is created, automatically a new Table called ***Table1*** is created. We need to delete by right clicking on the Tables view on left side.



We need to create a table, Goto Create menu and click Query Design to open query editor window.

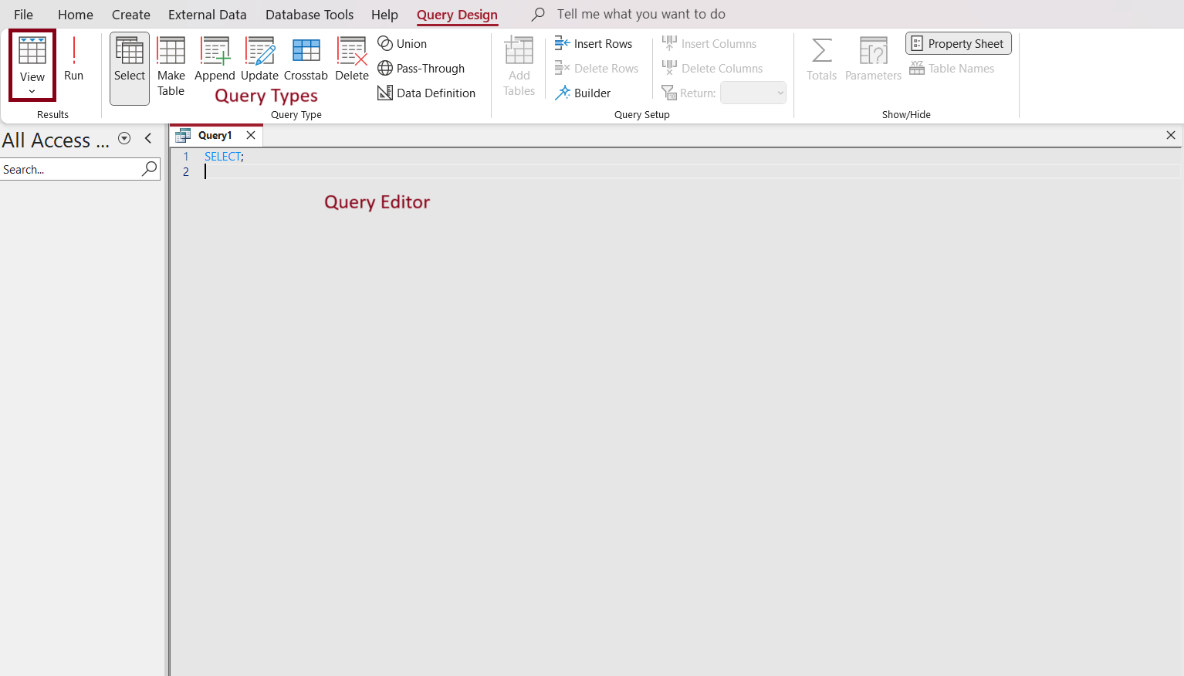


After opening query window, we need to select the SQL view using the dropdown menu on top left corner.

A screenshot of a computer

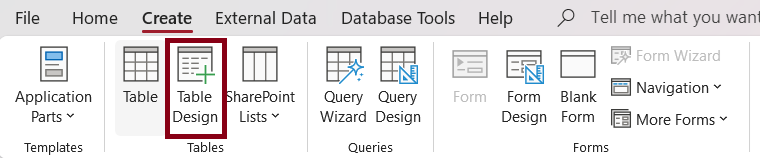
AI-generated content may be incorrect.

After selecting, the window should look something like this. Here we can write actual SQL queries in the Query Editor and run them against the database.

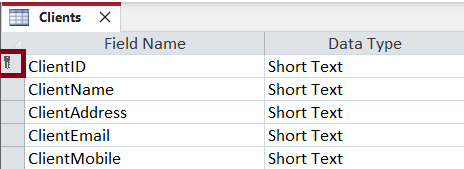


## **Tables Creation for Entities**

For creating tables for our entities, we use Table Design feature as it is easy and convenient. All the tables in the database are **3NF compliant**.



Using Table Design feature, we need to list out all the attributes for that entity and its data type.



After entering all the details, to create the table we need to save it, after that it will display a popup message to enter the table name. In ClientID, there is a key icon which means primary key.

A screenshot of a computer

AI-generated content may be incorrect.

### **Database Normal Forms (Normalization)**

Database normalization is a database design principle for organizing data in an organized and consistent way. It helps you avoid redundancy and maintain the integrity of the database. It also helps you eliminate undesirable characteristics associated with insertion, deletion, and updating.

**What is the purpose of Normalization?**

The main purpose of database normalization is to avoid complexities, eliminate duplicates, and organize data in a consistent way. In normalization, the data is divided into several tables linked together with relationships.

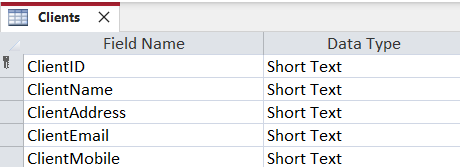
In practice, there are **three** Normal Forms currently in use, but total there are **six** Normal Forms.

1. **1st Normal Form:** For a table to be in the first normal form, it must meet the following conditions.
   1. A single cell must not hold more than one value.
   2. There must be a primary key for identification.
   3. No duplicated rows or columns.
2. **2nd Normal Form:** A table is said to be in 2NF if it meets the following conditions.
   1. It should be 1st Normal Form compliant.
   2. It has no partial dependency. That is, all non-key attributes are fully dependent on a primary key.
3. **3rd Normal Form:** So, for a table to be in 3NF, it must:
   1. Be in 2nd Normal Form.
   2. It should not have transitive partial dependency.

### **Clients Table**

The Clients table stores information about the company’s clients.

Clients table has 5 attributes including ClientID, ClientName, ClientAddress, ClientEmail, ClientMobile and ClientID is the primary key. The below screenshot shows the table structure with data type.



Every attribute has a validation rule and must be provided some valid data. Input mask has been set for ClientID as **"CL-"099999** and ClientMobile as **"0"000\ 000\ 0000** is set, so user can easily enter data without confusions.

### **Employees Table**

The Employees table stores all the data related to employee who works for the company.

Employees table has 4 attributes including EmployeeID, EmployeePayrollNumber, EmployeeName, EmployeeMobile and EmployeeID is the primary key. The below screenshot shows the table structure with data type.

A screenshot of a computer

AI-generated content may be incorrect.

Every attribute has a validation rule and must be provided some valid data. Input mask has been set for EmployeeID as **"EM-"099999** and EmployeeMobile as **"0"000\ 000\ 0000** is set, so user can easily enter data without confusions.

### **Service Catalogue Table**

The Service Catalogue table stores all the services provided by the company.

Service Catalogue table has 4 attributes including ServiceID, ServiceName, ServiceDescription, ServiceHourlyRate and ServiceID is the primary key. The below screenshot shows the table structure with data type.

A screenshot of a computer

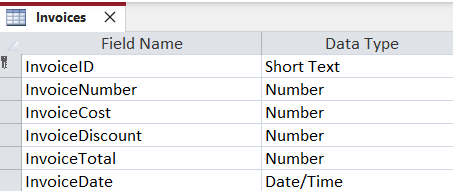
AI-generated content may be incorrect.

Every attribute has a validation rule and must be provided some valid data. Input mask has been set for ServiceID as **"SC-"099999**.

### **Invoices Table**

The Invoices table stores all the billing related information for every client appointment.

Invoices table has 6 attributes including InvoiceID, InvoiceNumber, InvoiceCost, InvoiceDiscount, InvoiceTotal, InvoiceDate and InvoiceID is the primary key. The below screenshot shows the table structure with data type.



Every attribute has a validation rule and must be provided some valid data. Input mask has been set for InvoiceID as **"IN-"099999**.

### **ClientAppointments Table**

The Client Appointments table stores all the information about appointment with client and employee for a particular service.

Client Appointments table has 5 attributes including ClientAppointmentID, ServiceID, ClientID, EmployeeID, AppointmentDate and ClientAppointmentID is the primary key. And Foreign keys are, ServiceID which is linked to ServiceCatalogue table, EmployeeID is linked to Employees table, and finally ClientID linked to Clients table. The below screenshot shows the table structure with data type.

A screenshot of a computer

AI-generated content may be incorrect.

Every attribute has a validation rule and must be provided some valid data. Input mask has been set for ClientAppointmentID as **"CA-"099999**.

### **ServicesProvided Table**

The Services Provided table stores information about which employee provided what service to which client.

Services Provided table has 4 attributes including ServiceID, EmployeeID, ClientAppointmentID, Expenses. The below screenshot shows the table structure with data type.

A screenshot of a computer

AI-generated content may be incorrect.

### **InvoiceRows Table**

The Invoice Rows links the Invoice table to the ClientAppointments table.

Invoice Rows table has 3 attributes including InvoiceRowID, InvoiceID, ClientAppointmentID, and InvoiceID is the primary key. The below screenshot shows the table structure with data type. The Foreign keys are InvoiceID which is linked to Invoices table and ClientAppointmentID is linked to ClientAppointments table.

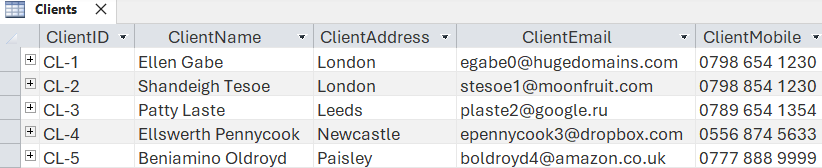
A screenshot of a computer

AI-generated content may be incorrect.

Every attribute has a validation rule and must be provided some valid data. Input mask has been set for InvoiceRowID as **"IR-"099999**.

# **Populating Tables with Data**

1. Clients



1. Employees

A screenshot of a computer

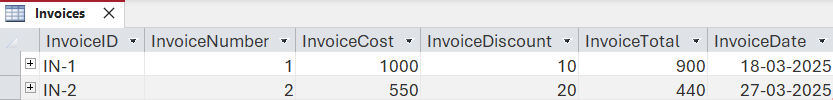
AI-generated content may be incorrect.

1. ServiceCatalogue

A screenshot of a computer

AI-generated content may be incorrect.

1. Invoices



1. ClientAppointments

A screenshot of a computer

AI-generated content may be incorrect.

1. ServiceProvided

A screenshot of a computer

AI-generated content may be incorrect.

1. InvoiceRows

A screenshot of a computer

AI-generated content may be incorrect.

# **Structured Query Language**

Structured Query Language (SQL) is a common language for managing and manipulating relational database management systems. It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

In SQL, there are a total of 4 different types of SQL Commands.

1. **DDL (Data Definition Language):** It consists of SQL commands that can be used to defining, altering, and deleting database structures such as tables and indexes.

Commands used:

|  |  |
| --- | --- |
| **Command** | **Description** |
| CREATE | Used for creating database, table, indexes and views. |
| DROP | Used for deleting database objects i.e. tables. |
| ALTER | Used for modifying the structure of the table. |
| TRUNCATE | Deletes all the records from a specified table. |

1. **DML (Data Manipulation Language):** The SQL commands that deals with manipulation of the data in a table.

Commands used:

|  |  |
| --- | --- |
| **Command** | **Description** |
| INSERT | Insert data into database. |
| UPDATE | Updates an existing row(s) in database, optionally we can pass condition to update row otherwise it will update everything. |
| DELETE | Delete record from a table with a condition. |

1. **DCL (Data Control Language):** DCL (Data Control Language) includes commands such as **GRANT** and **REVOKE** which mainly deal with the rights, permissions, and other controls of the database system.

|  |  |
| --- | --- |
| **Command** | **Description** |
| GRANT | Assigns new privileges to a user account, allowing access to specific database objects, actions, or functions. |
| REVOKE | Removes previously granted privileges from a user account, taking away their access to certain database objects or actions. |

1. **TCL (Transaction Control Language):** sdsd

## **Queries**

1. **Appointment List:** Lists out all the appointments with proper names instead of IDs. This uses inner join to link other tables such as Clients, Client Appointments and Service Catalogue.

SELECT clientappointments.clientappointmentid,

       servicecatalogue.servicename,

       clients.clientname,

       employees.employeename

FROM employees

    INNER JOIN(clients

    INNER JOIN(servicecatalogue

    INNER JOIN clientappointments

        ON servicecatalogue.serviceid = clientappointments.[serviceid])

        ON clients.clientid = clientappointments.clientid)

        ON employees.employeeid = clientappointments.employeeid;

Result:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Clients List:** Lists out all the client data from the Clients table.

SELECT Clients.ClientID,

       Clients.ClientName,

       Clients.ClientAddress,

       Clients.ClientEmail,

       Clients.ClientMobile

FROM Clients;

Result:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Employee List:** Lists out all employee data.

SELECT Employees.EmployeeID,

       Employees.EmployeePayrollNumber,

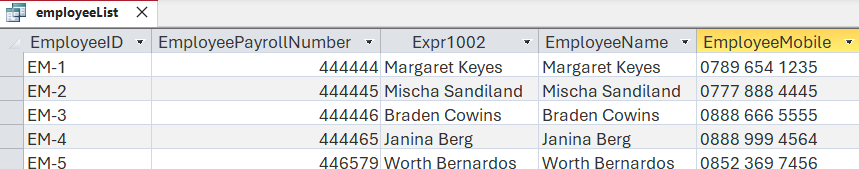
       Employees.EmployeeName,

       Employees.EmployeeName,

       Employees.EmployeeMobile

FROM Employees;

Result:



1. **Get Client By ID:** Gets a particular client by their ID. It uses parameters to get Client ID from user. A modal popup will be shown to user to enter the ID.

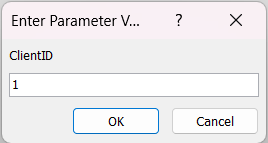
PARAMETERS ClientID Text ( 255 );

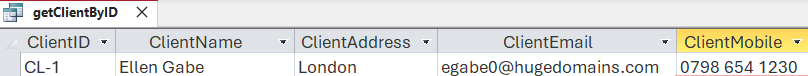
SELECT clientList.\*

FROM clientList

WHERE (((clientList.ClientID) = [ClientID]));

Result:





1. **Insert a new Client:** Creates a new record in the Clients table with the given data using insert query.

INSERT INTO Clients

(ClientID, ClientName, ClientAddress, ClientEmail, ClientMobile)

VALUES

("6", "Prajwal Aradhya", "London", "prajwal@email.com", "7757184256");

Result:



1. **Insert a new Employee:** Creates a new Employee record in Employees table with given data using insert query.

INSERT INTO Employees

(EmployeeID, EmployeeName, EmployeePayrollNumber, EmployeeMobile)

VALUES

("6", "Sayyar", 789456, "7757185698");

Result:



1. **Updating Service Catalogue:** TODO:

UPDATE ServiceCatalogue

SET ServiceHourlyRate = 55

WHERE ServiceID = "1";

Result before running query:



Result after running query:



1. **Deleting Service Catalogue:** TODO:

DELETE FROM ServiceCatalogue

WHERE ServiceID = "6";

Result before running query:

A yellow line on a white background

AI-generated content may be incorrect.

Result after running query:

A screenshot of a computer

AI-generated content may be incorrect.

1. **Update Client Appointment:** TODO:

UPDATE ClientAppointments

SET AppointmentDate = "20-3-2025"

WHERE ClientAppointmentID = "2";

Result before running query:



Result after running query:



1. **Update Services Provided:** TODO:

UPDATE ServicesProvided

SET Expenses = 700

WHERE ServiceProvidedID = 5;

Result before running query:



Result after running query:



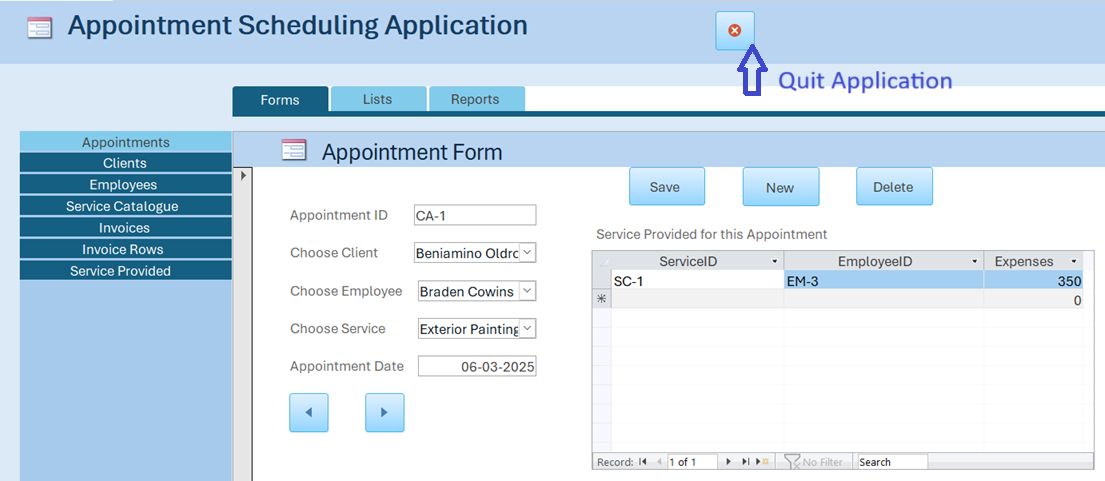
# **Database Application**

Appointment Scheduling Application is developed using horizontal + vertical navigation form, horizontally we have Forms, Lists and Reports tabs and all of them have their own sub-forms to visualize the data.

For every form, we have Save, New and Delete buttons for ease of use, so user doesn’t have to navigate using built-in controllers which are hard. And to navigate between records, we have next and previous buttons, so user can go to next or previous record easily.

In Forms tab, we have forms to all tables that user can fill.

1. **Appointments Form**: Here for choosing client, we added combo box to select the client using a drop down, instead of entering the Client ID manually. Same process is used for Choose Employee and Choose Service field.



1. **Client Form**

A screenshot of a computer application

AI-generated content may be incorrect.

1. **Employee Form**

A computer screen shot of a application

AI-generated content may be incorrect.

1. **Service Catalogue Form**

A screenshot of a application

AI-generated content may be incorrect.

1. **Invoices Form:** In Invoices Form, we used related list to get all the Invoice rows which are related to current invoice. This helps better auditing.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Invoice Rows Form**: In this form, we used combo box to select Invoice and Client Appointment IDs.

A screenshot of a computer application

AI-generated content may be incorrect.

1. **Service Provided Form**: In this form, combo box is used to choose Employee, Service and which Client Appointment is associated with, the Expenses field is formatted to show the pound symbol, so the user can recognize it is a currency field.

A screenshot of a computer

AI-generated content may be incorrect.

In Lists tab, we have data sheets that are generated from the queries that are made in above sections.

1. **Appointments List**: In this list, it shows Service Name, Client Name, Employee Name and Client Appointment ID. So, this gives more useful information than just showing regular IDs that are stored in the table.

A screenshot of a application

AI-generated content may be incorrect.

1. **Clients List**

A screenshot of a computer application

AI-generated content may be incorrect.

1. **Employees List**

**A screenshot of a application

AI-generated content may be incorrect.**

1. **Invoices:** In this list, the InvoiceNumber is shown as it is user-generated instead of InvoiceID.

**A screenshot of a schedule

AI-generated content may be incorrect.**

1. **Services Catalogue**

**A screenshot of a computer application

AI-generated content may be incorrect.**

1. **Services Provided:** In this list, actual data is shown instead of IDs.

**A screenshot of a schedule

AI-generated content may be incorrect.**

1. **Invoice Row:** In this list, Service Name and Client Name is taken instead of their IDs for better readability.

**A screenshot of a application

AI-generated content may be incorrect.**

In Reports Tab, we have 2 reports

1. **Invoices**: This gives all the invoice data in one single page with total Invoice value on top right.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Invoices By Month**: This Report is special and gives dynamic data based on user input. When opened, system is prompted to user to give start and end date, so the system can generate an invoice from the start and end date given by the user. Once given, Invoice Report is generated and on top-right, Toval Invoice Value, From Date and To Date is mentioned.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

# **Consideration on Privacy and Security**

In the Appointment Scheduling Database for Service-Based SME, the proposed database design does not contain the suggested methods for privacy and security. This presents additional risks

# **Critical Evaluation**

**Strengths and Weaknesses of the Database Design**

The database system developed for the Appointment Scheduling for a Service-Based SME project demonstrates a well-structured and normalized relational database. The conceptual design, implementation, and integration of Microsoft Access functionalities effectively capture the business requirements outlined in the coursework. The Entity-Relationship (ER) diagram accurately models relationships between entities, ensuring data integrity through well-defined primary and foreign keys.

One of the major strengths of the design is the third normal form (3NF) compliance, ensuring minimal data redundancy and efficient data retrieval. The use of referential integrity constraints prevents orphan records and enhances data consistency. The separation of entities such as Clients, Employees, Service Catalogue, and Invoices into distinct tables ensures data integrity and minimal redundancy. The use of junction tables (e.g., ServicesProvided, InvoiceRows) to handle many-to-many relationships is a notable strength, aligning with best practices for relational databases. Additionally, structured query language (SQL) queries were successfully implemented to retrieve, insert, update, and delete records efficiently, supporting the required CRUD operations and business reports.

However, some areas for improvement were identified. Firstly, the graphical user interface (GUI) in Microsoft Access could be enhanced to provide a more intuitive user experience. While the existing forms, reports, and navigation menus function well, a more streamlined and visually appealing interface could improve usability. Secondly, the security measures in the database could be further refined. Although access controls and validation rules were implemented, additional security measures such as role-based access control (RBAC) and encryption for sensitive data (e.g., client contact details) could strengthen data protection. Additionally, the partial ER diagrams only include primary key attributes, omitting composite or multi-valued attributes that could enhance clarity. Furthermore, the reliance on Microsoft Access’s auto-generated surrogate keys (e.g., ClientID as "CL-099999") may limit flexibility if the system scales beyond its current scope. If the business grows significantly, a migration plan to a more robust database management system (e.g., MySQL or PostgreSQL) should be considered.

**Effectiveness of Implementation**

The implementation phase demonstrated a solid grasp of relational database principles. The tables and relationships were correctly established, enforcing entity integrity and referential integrity. The queries successfully facilitated business operations such as retrieving client appointments, generating invoices, and tracking service history. Key constraints, such as foreign keys and validation rules (e.g., input masks for phone numbers), enforced data integrity effectively. The SQL queries (e.g., Appointment List, Get Client By ID) demonstrated practical utility, though some queries lacked parameterization, which could improve user interaction.

Additionally, Microsoft Access was chosen as the database management system (DBMS) due to its ease of use and built-in functionalities. The decision to use Access was justified by its integration with Microsoft Office Suite and support for small business applications. However, while Access is user-friendly, it has limitations in terms of concurrent users and scalability. The application’s GUI is functional, with forms and combo boxes simplifying data entry. However, the absence of error-handling mechanisms in forms (e.g., preventing invalid date inputs) is a missed opportunity to enhance robustness. Future iterations of this database could consider migration to SQL Server or cloud-based solutions for better performance.

# **Contributions:**

Each member of the group played a crucial role in ensuring the successful completion of this database project. Below is a breakdown of individual contributions:

1. **@Imran**  – Critical Evaluation of Database Structure
   * Focused on ensuring the database adhered to Third Normal Form (3NF) by analyzing table relationships and eliminating redundancy.
   * Verified primary and foreign key constraints to maintain referential integrity.
   * Reviewed the ER diagrams to ensure they accurately represented the logical design before implementation.
   * Identified potential weaknesses, such as missing composite attributes in the initial ER diagrams, and suggested improvements.
2. **@Adil** – Privacy and Security Considerations
   * Conducted a risk assessment of the database design to identify sensitive data (e.g., client emails, mobile numbers).
   * Proposed GDPR-compliant measures, including data minimization, access controls, and secure storage practices.
   * Drafted recommendations for future enhancements, such as field-level encryption for personally identifiable information (PII).
   * Ensured the "Consideration on Privacy and Security" section (Task 6) addressed legal and ethical concerns.
3. **@Sayyar** – Revising the Draft for Clarity and Coherence
   * Restructured the report to improve readability and logical flow, ensuring each section (e.g., ER diagrams, normalization, queries) was well-explained.
   * Standardized terminology across the document to maintain consistency.
   * Refined SQL query explanations and screenshot annotations to make them more user-friendly.
   * Ensured the final submission adhered to academic writing standards and coursework guidelines.
4. **@Sapana** – Supporting Team Members and Final Revisions
   * Assisted in testing SQL queries to verify accuracy before inclusion in the report.
   * Helped populate sample data into tables to demonstrate relationships and constraints.
   * Proofread the final draft, correcting grammatical errors and formatting inconsistencies.
   * Provided feedback on the user interface (UI) design in Microsoft Access to ensure ease of use.

**Personal Reflection on Learning Experience**

This coursework provided invaluable experience in database design and implementation, reinforcing core concepts such as data normalization, relational schema development, and SQL querying. Designing the ER diagram, defining table structures, and enforcing data integrity constraints deepened our understanding of relational database management systems (RDBMS). The project also highlighted the importance of query optimization and database security considerations, crucial for real-world database applications.

One key takeaway was the significance of structuring relationships appropriately to avoid redundancy and improve efficiency. Working with Microsoft Access also demonstrated the practicality of low-code database solutions for small businesses while exposing limitations that arise when dealing with large datasets or complex business rules. Constructing queries and forms reinforced the value of SQL and GUI integration in real-world applications.

**Areas for Improvement and Recommendations**

While the project met its core objectives, a few areas could be improved:

1. **Enhanced Documentation:** The ER diagrams should include all attributes to fully capture the design.
2. **Query Optimization:** Some queries could benefit from dynamic parameters or joins to reduce redundancy.
3. **Security Measures:** Implement user authentication mechanisms, role-based access controls, and encryption for sensitive fields (e.g., ClientEmail) to strengthen data protection.
4. **Improved User Interface:** A more user-friendly and aesthetically appealing GUI in Microsoft Access would enhance usability.
5. **Scalability Considerations:** Future iterations should explore migrating to a **more scalable DBMS** to accommodate business growth.
6. **Automation and Reporting Enhancements:** More automated reports and dynamic query-based forms could improve data retrieval efficiency.
7. **Performance Optimization:** Indexing frequently queried fields could enhance query performance in large datasets.

# **Conclusion:**

This coursework provided a hands-on understanding of relational database design, from conceptual modeling in ER diagrams to practical implementation in Microsoft Access. The project successfully met core requirements, including:

* Normalization (3NF) to eliminate redundancy
* Effective SQL queries for data retrieval and manipulation
* User-friendly forms and reports for real-world usability

However, some areas could be improved in future iterations:

1. **Enhanced ER Diagrams** – Including all attributes (not just primary keys) would provide a more complete design overview.
2. **Dynamic Query Parameters** – Some queries could be improved with user-input prompts for flexibility.
3. **Advanced Security** – Implementing encryption for sensitive fields would strengthen data protection.

Overall, this project reinforced the importance of balancing theory with practical constraints (e.g., Access limitations) while highlighting the collaborative effort needed in database development. The experience has been invaluable in bridging academic concepts with real-world application, preparing us for more complex database challenges in the future.