**BestCare Management System Milestone #3**

Lakkappa, Ravichandra

Epuri Nikhila Lokesh

Sam Jayanth Reddy

Harikumar Bhaveshbhai

01/26/2023

Professor Virginia Ilie

**INDEX**

Table of Contents

[ERD #1 3](#_Toc127448107)

[ERD #2 3](#_Toc127448108)

[Assumptions 5](#_Toc127448109)

[Logical Schema 6](#_Toc127448110)

[Research on SQL Server datatypes 7](#_Toc127448111)

[Database Schema 23](#_Toc127448112)

[Results 27](#_Toc127448113)

[Query Requirements 29](#_Toc127448114)

[Extra Credit 30](#_Toc127448115)

[Appendix 32](#_Toc127448116)

ERD #1

Diagram

Description automatically generated

# 

# ERD #2

A picture containing text, receipt

Description automatically generated

## Assumptions

* Patients can visit multiple Campuses for treatment and each campus can treat multiple patients.
* Campus can provide multiple services to patients and same service can be provided by multiple campuses.
* In a campus there will be various staff, but one staff person can work only in one campus.
* Staff has two sub types which are Nurse and Doctor.
* Nurse can meet any number of patients; each patient may or may not meet multiple nurses for assessment.
* Patient can visit multiple doctors for treatment and date, reason, and comments might vary based on each visit.
* Doctor can write many orders for patient, but each order should be written by a particular doctor.
* Each order has multiple medications, and particular medications can present in various orders. For every order quantity and dosage will vary based on the medication history.
* Orders can be taken by particular pharmacy based on the patient's choice and pharmacy can take multiple orders.
* Each order can have multiple Diagnosis for a patient and same Diagnosis can be done for many orders based on patient’s condition.
* Rooms can be present in many numbers of campuses and a particular room will be present in a campus.
* Room will have multiple equipment, but same equipment can’t present in same room multiple times.

## Logical Schema

Graphical user interface, text, application

Description automatically generated

# Research on SQL Server datatypes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DATA TYPE** | **DESCRIPTION** | | **WHEN TO USE** | | | |
| **Numeric** | | | | | | |
| bigint | When integer values might exceed the range that is supported by the int data type this must be used | | For Integer Values  *Range*: -2^63 to 2^63-1  *Storage*: 8 Bytes | | | |
| bit | An integer data type that can hold a value from 1 to 64. | | For values 1, 0, or null  *Storage*: 2 Bytes | | | |
| decimal | Numeric data types that have fixed precision and scale. | | For Numeric data  *Range*: - 10^38 +1 through 10^38 - 1. (Maximum precision)  *Storage:* 5-17 Bytes | | | |
| int | Integer data type | | For Integer Values  *Range*: -2^31 to 2^31-1  *Storage*: 4 Bytes | | | |
| money | Data Type for Monetary Units. | | Currency /Monetary Data  *Range*: -922,337,203,685,477.5808 to 922,337,203,685,477.5807  *Storage*: 8 Bytes | | | |
| numeric | Measurable data type. | | For Numeric data  *Range*: - 10^38 +1 through 10^38 - 1. (Maximum precision)  *Storage*: 5-17 Bytes | | | |
| smallint | Data type that stores small whole numbers. | | For Integer Values  *Range*: -2^15 to 2^15-1  *Storage*: 2 Bytes | | | |
| smallmoney | Data Type for Monetary Units. Accurate to a ten-thousandth of the monetary units that they represent. | | Currency /Monetary Data  *Range*: - 214,748.3648 to 214,748.3647  *Storage*: 4 Bytes | | | |
| tinyint | The smallest data type for integer values | | For Integer Values  *Range*: 0 to 255  *Storage*: 1 Byte | | | |
| float | Approximate-number data types for use with floating point numeric data. | For Approximate Numbers  *Range*: '1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308  *Storage*: 4-8 Bytes | | | | |
| real | Approximate-number data types for use with floating point numeric data. | For Approximate Numbers  *Range*: 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38  *Storage*: 4 Bytes | | | | | |
| **Character Strings** | | | | | | | |
| char | Character data types that are of fixed size. | | | For uniform character entries  *Range*: 1 through 8,000  *Storage*: Table specific | | | |
| varchar | Character data types that are of variable size. | | | For non-uniform character entries  *Range*: 1 through 8,000  *Storage*: 2 Bytes + Data size | | | |
| text | It is used for Variable-length non-Unicode data. | | | For binary string  *Range*: Maximum string length of 2^31-1  *Storage*: Data Size | | | |
| **Unicode Characters Strings** | | | | | | | |
| nchar | Character data types that are of fixed size | | | | For fixed-size strings  *Range*: 1 through 4,000  *Storage*: 2 \* n bytes | | |
| nvarchar | Character data types that are of variable-size | | | | For variable-size strings  *Range*: 1 through 4,000  *Storage*: 2 bytes + 2 \* n bytes | | |
| ntext | Variable-length data types for storing large Unicode data. | | | | For string data  *Range*: Maximum string length of 2^30 - 1  *Storage*: 2 \* Stringlength | | |
| **Binary Strings** | | | | | | | |
| binary | Binary data type for fixed length Binary data. | | | | For Binary values  *Range*: 1 through 8,000  *Storage*: Table Specific | | |
| image | The variable-length data type for storing large binary data. | | | | for Image files  *Range*: 0 through 2^31-1  *Storage*: 2 bytes + data size | | |
| varbinary | Binary data types for Variable-length Binary data. | | | | for images, pdf or word  *Range*: 1 through 8,000  *Storage*: 2 bytes + data size | | |
| **Date & Time** | | | | | | | |
| date | For Date Description, Format = YYYY-MM-DD | | | | for Date  *Range*: 0001-01-01 through 9999-12-31  *Storage*: 3 bytes | | |
| datetime | Defines a date combined with a time of day with fractional seconds based on a 24-hour clock. | | | | for Date & Time  *Range*: January 1, 1753, through December 31, 9999 & 00:00:00 through 23:59:59.997  *Storage*: 8 bytes | | |
| datetime2 | An extension of the existing datetime type | | | | For Date & Time 24 hr  *Range*: 0001-01-01 through 9999-12-31 & 00:00:00 through 23:59:59.9999999  *Storage*: 6-8 bytes | | |
| datetimeoffset | Defines a date that is combined with a time of a day that has time zone awareness and is based on a 24-hour clock. | | | | Date, time 24 hrs & time zone  *Range*: 0001-01-01 through 9999-12-31 & 00:00:00 through 23:59:59.9999999 & -14:00 through +14:00  *Storage*: 10 Bytes | | |
| smalldatetime | Date combined with a time of day based on a 24-hour day, with seconds always zero& without fractional seconds | | | | Date time without fraction seconds  *Range*: 1900-01-01 through 2079-06-06 & 00:00:00 through 23:59:59  *Storage*: 4 bytes | | |
| time | Defines a time of the day. The time is without time zone awareness and is based on a 24-hour clock. | | | | Time of the day  *Range*: 00:00:00.0000000 through 23:59:59.9999999  *Storage*: 5 bytes | | |
| **Other Data Types** | | | | | | | |
| cursor | A data type for variables or stored procedure OUTPUT parameters containing a cursor reference. | | | | | for Cursor reference |
| hierarchyrid | Variable length, a system data type that represents a position in a tree hierarchy. | | | | | For the value of Position in the Hierarchy  *Storage*: 5 bytes |
| sql\_variant | A data type that stores values of various SQL Server-supported data types. | | | | | For the Value of Objects of other data types  *Storage*: 8016 bytes |
| uniqueidentifier | It is a globally unique identifier. | | | | | For a column or local variable  *Storage*: 16 Bytes |
| xml | This Data type stores XML data. | | | | | For well-formed XML fragments.  *Storage*: 2 GB Max |
| Spatial Geometry types | This data type represents data in a Euclidean (flat) coordinate system. | | | | | For Geometry methods |
| table | Special data type used to store a result set for processing later | | | | | For Functions and variables |
| Spatial Geography types | Data type represents data in a round-earth coordinate system. | | | | | For Co-ordinates |
| rowversion | This data type exposes automatically generated, unique binary numbers within a database | | | | | For the value of Incrementing number  *Storage*: 8 Bytes |

Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type (Size)** | **Constraints** |
| **Patients** | | |
| **Patient ID** | Int | Primary key for PATIENT table |
| First\_Name | Varchar (50) |  |
| Last\_­Name | Varchar (50) |  |
| Date\_of\_Birth | Date |  |
| Street | Varchar (50) |  |
| City | Varchar (50) |  |
| State | Varchar (50) |  |
| ZipCode | Int |  |
| Phone\_Number | Int |  |
| Mail\_ID | Varchar (50) |  |
| Relation\_to\_Patient | Varchar (50) |  |
| ContactPerson\_FirstName | Varchar (50) |  |
| ContactPerson\_LastName | Varchar (50) |  |
| ContactPerson\_Address | Varchar (100) |  |
| Insurance\_Company\_Name | Varchar (50) |  |
| Insurance\_Number | Int |  |
| Group\_Number | Int |  |
| Insurance\_Contact\_Number | Int |  |
| **Patient\_Visit** | | |
| **Patient\_Visit\_ID** | Int | Primary key for PATIENT \_VISIT table. |
| *Patient\_ID* | Int | Foreign key for Patients table. |
| *Campus\_ID* | Int | Foreign key for Campuses table. |
| Date | Date |  |
| Reason | Varchar (50) |  |
| Comments | Varchar (50) |  |
| **Campuses** | | |
| **Campus\_ID** | Int | Primary key for CAMPUSES table. |
| Name | Varchar (50) |  |
| Street | Varchar (50) |  |
| City | Varchar (50) |  |
| State | Varchar (50) |  |
| ZipCode | Int |  |
| Contact\_Number | Int |  |
| Mail\_ID | Varchar (50) |  |
| Website | Varchar (50) |  |
| **Campus\_Services** | | |
| **Campus\_Services\_ID** | Int | Primary key for CAMPUS\_SERVICES table. |
| *Campus\_ID* | Int | Foreign key for CAMPUS table. |
| *Service\_ID* | Int | Foreign key for SERVICES table. |
| **Services** | | |
| **Service\_ID** | Int | Primary for key SERVICE table. |
| Service\_Type | Varchar (50) |  |
| Service\_Description | Varchar (250) |  |
| **Staff** | | |
| **Staff\_ID** | Int | Primary for key STAFF table. |
| First\_Name | Varchar (50) |  |
| Last\_Name | Varchar (50) |  |
| Date\_of\_Birth | Date |  |
| Email\_Id | Varchar (50) |  |
| Phone\_Number | Int |  |
| Street | Varchar (50) |  |
| City | Varchar (50) |  |
| State | Varchar (50) |  |
| Country | Varchar (50) |  |
| Staff\_Type | Varchar (15) |  |
| ZipCode | Int |  |
| *Campus\_ID* | Int | Foreign key to Campuses table. |
| **Nurses** | | |
| **Staff\_ID** | Int | Primary key for NURSE table. |
| Lincence\_no | Varchar (25) |  |
| National\_Certificate | Varchar (50) |  |
| Position | Varchar (50) |  |
| **Doctors** | | |
| **Staff\_ID** | Int | Primary key for STAFF table. |
| Specialization | Varchar (50) |  |
| Credentials | Varchar (50) |  |
| Board\_Certificate | Varchar (50) |  |
| DEA\_Number | Int |  |
| Graduation | Varchar (50) |  |
| Graduation\_Year | Int |  |
| **Nurse\_Assessment** | | |
| **Nurse\_Assessment\_ID** | Int | Primary key for NURSE\_ASSESSMENT table. |
| *Staff\_ID* | Int | Foreign key to Nurses table. |
| *Patient\_ID* | Int | Foreign key to Patients table. |
| Temperature | Varchar (50) |  |
| Weight | Varchar (50) |  |
| Blood\_Pressure | Varchar (50) |  |
| Pulse | Varchar (50) |  |
| Date | Date |  |
| **Room** | | |
| **Room\_ID** | Int | Primary key for ROOM table. |
| Room\_Type | Varchar (50) |  |
| Room\_Size | nVarchar(50) |  |
| *Campus\_ID* | Int | Foreign key to Campuses table. |
| **Equipment** | | |
| **Equipment\_ID** | Int | Primary key for EQUIPMENT table. |
| Equipment\_Type | Varchar (50) |  |
| Serial\_Number | Varchar (50) |  |
| *Room\_ID* | Int | Foreign key to Room table. |
| **Patient\_Doctor** | | |
| **Patient\_Doctor\_ID** | Int | Primary key for PATIENT\_DOCTOR table. |
| *Patient\_ID* | Int | Foreign key to PATIENT table. |
| *Staff\_ID* | Int | Foreign key to Doctors table. |
| **Orders** | | |
| **Order\_ID** | Int | Primary key for ORDERS table. |
| Order\_Time | Time |  |
| Date\_of\_Order | Date |  |
| Order\_Instructions | Varchar (250) |  |
| *Staff\_ID* | Int | Foreign key to Staff table. |
| *Pharmacy\_ID* | Int | Foreign key to Pharmacy table. |
| **Order\_Medication** | | |
| **Orders\_Medication\_ID** | Int | Primary key for ORDER\_MEDICATION table. |
| *Order\_ID* | Int | Foreign key to ORDER table. |
| *Medication\_ID* | Int | Foreign key to Medication table. |
| Quantity | Int |  |
| Dosage | Varchar (50) |  |
| **Medications** | | |
| **Medication\_ID** | Int | Primary key for MEDICATION table. |
| Name | Varchar (50) |  |
| Price | Decimal (10,2) |  |
| Medication\_Type | Varchar (50) |  |
| Manufacturer | Varchar (50) |  |
| Description | Varchar (250) |  |
| **Diagnosis\_Orders** | | |
| **Diagnosis\_Order\_ID** | Int | Primary key for DIAGNOSIS \_ORDERS table. |
| *Diagnosis\_ID* | Int | Foreign key to DIADNOSIS table. |
| *Order\_ID* | Int | Foreign key to ORDERS table. |
| **Diagnosis** | | |
| **Diagnosis\_ID** | Int | Primary key for DIAGNOSIS table. |
| Diagnosis\_Name | Varchar (50) |  |
| Diagnosis\_Code | Int |  |
| Diagnosis\_Description | Varchar (250) |  |
| **Pharmacy** | | |
| **Pharmacy\_ID** | Int | Primary key for PHARMACY table. |
| Name | Varchar (50) |  |
| Street | Varchar (50) |  |
| City | Varchar (50) |  |
| State | Varchar (50) |  |
| Country | Varchar (50) |  |
| ZipCode | Int |  |
| Phone\_Number | Int |  |
| Mail\_Id | Varchar (50) |  |
| Fax | Int |  |

Data Normalization

Numerous tables in the BestCare electronic medical record system are used to hold various sorts of data, including information on the campus, services provided, treatment rooms, equipment, staff, patients, physician-patient interactions, nurse assessments, and medication orders. In our ERD’s Entities do not have any Multivalued valued attributes in them so it meets the 1st Normalization Form. For minimizing data anomalies and ensuring data integrity, the tables are created based on their logical relationships. Each table has a unique set of columns for storing pertinent data. To guarantee that the data is in at least a third normal form, assure data consistency, and minimize data redundancy, the information is saved in different tables.

## Database Schema

A picture containing text, indoor, computer, screenshot

Description automatically generated

-- 1 QUERY

SELECT DISTINCT Patient\_ID,First\_Name,Street,City,[State],Phone\_Number

FROM Patients

ORDER BY [State];

------------------------------------------------------------

-- 2 QUERY

SELECT Orders.Order\_ID,Medications.Medication\_ID,SUM(Order\_Medication.Quantity) AS [Total\_Quantity]

FROM Orders,Order\_Medication,Medications

WHERE Orders.Order\_ID = Order\_Medication.Order\_ID

AND Order\_Medication.Medication\_ID = Medications.Medication\_ID

GROUP BY Medications.Medication\_ID,Order\_Medication.Quantity,Orders.Order\_ID

ORDER BY Order\_Medication.Quantity;

------------------------------------------------------------------------------------------------

--3 QUERY

SELECT Order\_ID,Order\_Time,Pharmacy.Pharmacy\_ID,Pharmacy.[Name]

FROM Orders,Pharmacy

WHERE Orders.Pharmacy\_ID= Pharmacy.Pharmacy\_ID

ORDER BY Order\_Time;

-------------------------------------------------------------------------------------------------

-- 4 QUERY

SELECT Patients.Patient\_ID,Patients.First\_Name, Patient\_Visit.Reason,Campuses.Campus\_ID

FROM Patients,Patient\_Visit,Campuses

WHERE Patients.Patient\_ID=Patient\_Visit.Patient\_ID

AND Patient\_Visit.Campus\_ID= Campuses.Campus\_ID;

--------------------------------------------------------------------------------------------

--5 QUERY

SELECT Doctor.Staff\_ID,Doctor.Specialization,Orders.Order\_ID,Orders.Date\_of\_Order,Medications.[Name],Order\_Medication.Quantity

FROM Doctor,Orders,Order\_Medication,Medications

WHERE Doctor.Staff\_ID= Orders.Staff\_ID

AND Orders.Order\_ID= Order\_Medication.Order\_ID

AND Order\_Medication.Medication\_ID=Medications.Medication\_ID

ORDER BY Orders.Order\_ID;

-------------------------------------------------------------------------------------------

--6 QUERY

SELECT Patients.First\_Name AS [PATIENT NAME], Patient\_Visit.Reason AS [Reason Of Visit],Campuses.Campus\_ID,Services.Service\_Type

FROM Patients,Patient\_Visit,Campuses,Campus\_Services,Services

WHERE Patients.Patient\_ID= Patient\_Visit.Patient\_ID

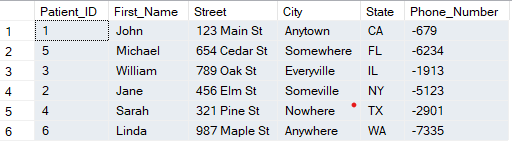
AND Patient\_Visit.Campus\_ID= Campuses.Campus\_ID

AND Campuses.Campus\_ID=Campus\_Services.Campus\_ID

AND Campus\_Services.Service\_ID=Services.Service\_ID;

## Results

Query 1



Query 2

Table

Description automatically generated with medium confidence

Query 3

Table

Description automatically generated

Query 4

Graphical user interface

Description automatically generated with low confidence

Query 5

Table

Description automatically generated with medium confidence

Query 6

Graphical user interface, application

Description automatically generated

## Query Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Query Number | Report Requirements | SQL Code | Recordset |
| Query1 | Select Patient details from patients and order by State | SELECT DISTINCT Patient\_ID,First\_Name,Street,City,  [State],Phone\_Number  FROM Patients  ORDER BY [State]; | 6 |
| Query2 | Select orders and medications from order, medication, order\_medication  Where orderID  Group by and order by for quantity. | SELECT Orders.Order\_ID,Medications.Medication\_ID,  SUM(Order\_Medication.Quantity)  AS [Total\_Quantity]  FROM Orders,Order\_Medication,Medications  WHERE Orders.Order\_ID = Order\_Medication.Order\_ID  AND Order\_Medication.Medication\_ID =  Medications.Medication\_ID  GROUP BY Medications.Medication\_ID,  Order\_Medication.Quantity,Orders.Order\_ID  ORDER BY Order\_Medication.Quantity; | 8 |
| Query3 | Select [details of pharmacy order]  From Pharmacy and order table  Where [ID]  Order by time | SELECT Order\_ID,Order\_Time,Pharmacy.Pharmacy\_ID,Pharmacy.[Name]  FROM Orders,Pharmacy  WHERE Orders.Pharmacy\_ID= Pharmacy.Pharmacy\_ID  ORDER BY Order\_Time; | 7 |
| Query4 | Select[patient campus details]  From patient, visit campus  Where [ID’s]  Joining 3 Tables | SELECT Patients.Patient\_ID,Patients.First\_Name, Patient\_Visit.Reason,Campuses.Campus\_ID  FROM Patients,Patient\_Visit,Campuses  WHERE Patients.Patient\_ID=Patient\_Visit.Patient\_ID  AND Patient\_Visit.Campus\_ID= Campuses.Campus\_ID; | 7 |
| Query5 | Select [Doctor medication details]  From Doctors orders, order\_medication  Where[ID]  Order by order ID  Joining 4 tables | SELECT Doctor.Staff\_ID,Doctor.Specialization,Orders.Order\_ID,  Orders.Date\_of\_Order,Medications.[Name],  Order\_Medication.Quantity  FROM Doctor,Orders,Order\_Medication,Medications  WHERE Doctor.Staff\_ID= Orders.Staff\_ID  AND Orders.Order\_ID= Order\_Medication.Order\_ID  AND Order\_Medication.Medication\_ID=Medications.Medication\_ID  ORDER BY Orders.Order\_ID; | 8 |
| Query6 | Joining 5 Tables  Selec[Detils]  From [Tables]  Where [ID of 5 tables] | SELECT Patients.First\_Name AS [PATIENT NAME],  Patient\_Visit.Reason AS [Reason Of Visit],  Campuses.Campus\_ID,Services.Service\_Type  FROM Patients,Patient\_Visit,Campuses,Campus\_Services,Services  WHERE Patients.Patient\_ID= Patient\_Visit.Patient\_ID  AND Patient\_Visit.Campus\_ID= Campuses.Campus\_ID  AND Campuses.Campus\_ID=Campus\_Services.Campus\_ID  AND Campus\_Services.Service\_ID=Services.Service\_ID; | 8 |

## Extra Credit

First, connect to the database management system and select the appropriate database to work. Use the CREATE TABLE command to define the structure of the table. This includes specifying the name of the table, the names and data types of each column, and any constraints or indices that should be applied to the table.

CREATE TABLE Patient (

PatientID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Address VARCHAR(255) NOT NULL,

City VARCHAR(50),

State VARCHAR(50),

ZipCode VARCHAR(10),

Email VARCHAR(255) NOT NULL UNIQUE

);

This command creates a table named "Patients" with columns for PatientsID, name, address, city, state, zip code, and email. The PRIMARY KEY constraint is added to the PatientsID column to ensure it is unique and serves as the primary key for the table. The NOT NULL constraint is added to the Name, Address, and Email columns to ensure they cannot contain null values, and the UNIQUE constraint is added to the Email column to ensure that no two customers can have the same email address.

Once the table is created, you can use the INSERT INTO command to add new rows to the table.

INSERT INTO Patients (ID, FirstName, LastName, Age, Email)

VALUES (1, 'John', 'Doe', 30, 'john.doe@example.com'),

(2, 'Jane', 'Smith', 25, 'jane.smith@example.com'),

(3, 'Bob', 'Johnson', 40, 'bob.johnson@example.com');

Retrieve data from the table, you can use the SELECT command.

SELECT ID, FirstName, LastName

FROM Patient

WHERE Age > 30;

# Appendix

Ravichandra Lakkappa ERD



Nikhil Lokesh Epuri

Diagram

Description automatically generated

Guduru Sam Jayanth Reddy

Diagram

Description automatically generated

Patel Hari

