



INFO 6210 Data Management and Database Design

Group 6: COVID-19 hospitalization management system

Pei-Ling Chiang, Shuyuan Lyu, Ssu-Yu Ning, Zhehui Yang, Bo-Zhong Zhang, Yuxin Zhang

Content

- **Introduction**
 - Background
 - Mission Objectives
- **Database Implementation**
 - ERD
 - Creating tables & inserting data
 - Stored procedure
 - Table level constraint
 - Computed column
 - Password encryption
 - Views
- **Report & Visualization**



Introduction - Background

- With the massive amount of COVID-19 patients occupying the hospital, the purpose of the design is to predict the disease prognosis, to modify the treatment strategy, and to adjust the admission standard in order to manage the COVID-19 patients more efficiently.

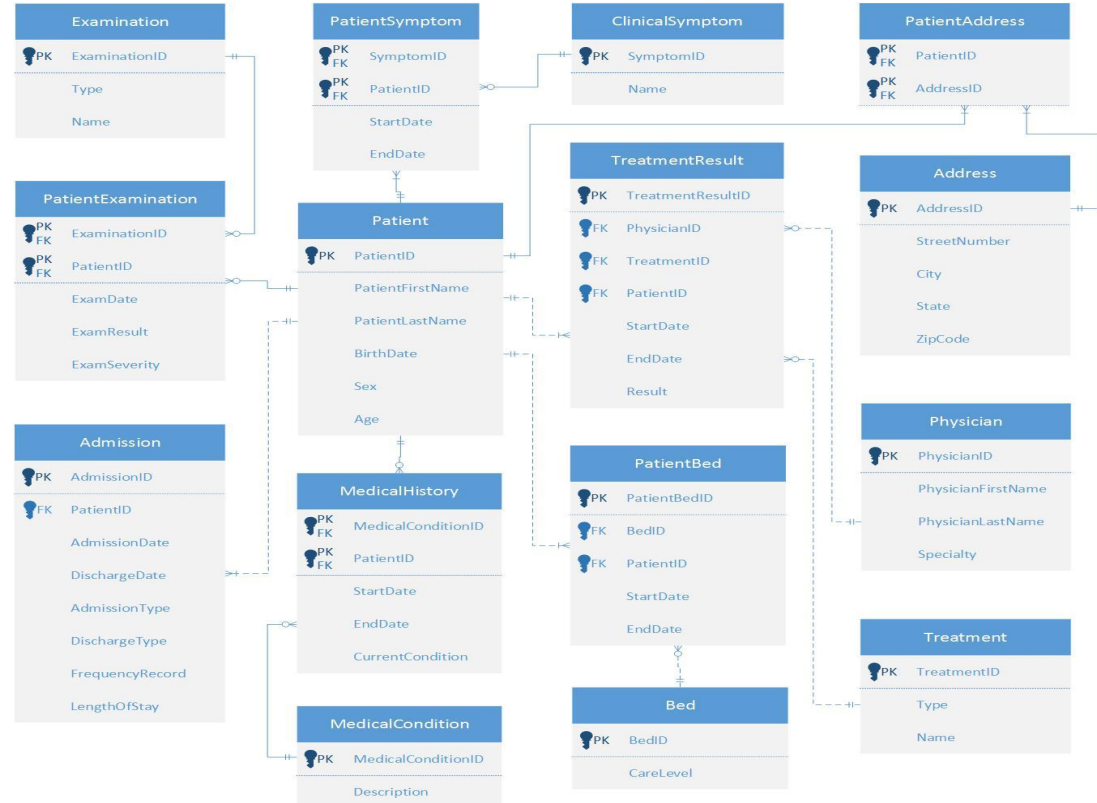


Introduction - Mission Objectives

- Track patients' addresses to send alerts and prevent community spread.
- Improve the scheduling of admission and discharge.
- Collect and analyze patients and treatment-related information that allows administrative staff to generate descriptive patients reports.
- Allow medical care providers to compare treatment results, which helps to further modify treatment strategies.
- Determine examinations and treatments that patients need based on clinical symptoms.

Database Implementation - ERD

- 9 Primary Entities:
Patient, Address,
ClinicalSymptom,
MedicalCondition,
Admission, Bed,
Physician, Treatment,
Examination
- 6 Associative Entities:
PatientAddress,
PatientSymptom,
MedicalHistory,
PatientBed,
TreatmentResult,
PatientExamination



Database Implementation - Stored Procedure

- Create a Stored Procedure - AddPhysician.

```
CREATE PROCEDURE dbo.AddPhysician
    @PhysicianFirstName VARCHAR(50),
    @PhysicianLastName VARCHAR(50),
    @Specialty VARCHAR(50)
AS
BEGIN
    INSERT INTO dbo.Physician(PhysicianFirstName, PhysicianLastName, Specialty)
    VALUES (@PhysicianFirstName, @PhysicianLastName, @Specialty)
END
```

- Physician data are inserted into database by “Stored Procedure.”

```
DECLARE @PhysicianFirstName VARCHAR(50) SET @PhysicianFirstName = 'Robert'
DECLARE @PhysicianLastName VARCHAR(50) SET @PhysicianLastName = 'Smith'
DECLARE @Specialty VARCHAR(50) SET @Specialty = 'Pulmonology'
EXEC dbo.AddPhysician @PhysicianFirstName, @PhysicianLastName, @Specialty
```

Database Implementation - Creating tables & Inserting data

- Create table and set data type, PK and FK.

Add check constraint as well.

- Most data are inserted into tables by “Inserting”.

```
CREATE TABLE dbo.Admission
(
    AdmissionID INT IDENTITY NOT NULL PRIMARY KEY,
    PatientID INT NOT NULL FOREIGN KEY REFERENCES Patient(PatientID),
    AdmissionDate DATETIME,
    DischargeDate DATETIME,
    AdmissionType VARCHAR(10),
    DischargeType VARCHAR(10),
    FrequencyRecord INT
);
ALTER TABLE dbo.Admission ADD CONSTRAINT CHK_admissiontype CHECK(AdmissionType
IN ('Regular','Emergency'));
ALTER TABLE dbo.Admission ADD CONSTRAINT CHK_dischargetype CHECK(DischargeType
IN ('Recover','Dead','null'));
ALTER TABLE Admission ADD CONSTRAINT BanKids CHECK (dbo.CheckAge(PatientID) >= 12);

INSERT INTO dbo.Admission(PatientID, AdmissionDate, DischargeDate,
    AdmissionType, DischargeType, FrequencyRecord)
VALUES (6, '2020-07-01', NULL, 'Regular', NULL, 2);
INSERT INTO dbo.Admission(PatientID, AdmissionDate, DischargeDate,
    AdmissionType, DischargeType, FrequencyRecord)
VALUES (3, '2020-07-22', '2020-07-30', 'Regular', 'Recover', 0);
INSERT INTO dbo.Admission(PatientID, AdmissionDate, DischargeDate,
    AdmissionType, DischargeType, FrequencyRecord)
VALUES (4, '2020-07-30', NULL, 'Regular', NULL, 1);
```

Database Implementation - Table Level Constraint

- We check patients' age to make sure patients < 12 years old will not be admitted.
- The insertion of patient < 12 years old into admission table will be terminated.

```
-- Drop function checkAge;
```

```
CREATE FUNCTION CheckAge (@PatientID INT)  
RETURNS SMALLINT  
AS  
BEGIN  
    DECLARE @Age SMALLINT = 0;  
    SELECT @Age = (SELECT DATEDIFF(hour, BirthDate, GETDATE())/8766  
                FROM Patient  
                WHERE PatientID = @PatientID)  
    RETURN @Age;  
END;
```



```
ALTER TABLE Admission ADD CONSTRAINT BanKids CHECK (dbo.CheckAge(PatientID) >= 12);
```


Database Implementation - Computed column

- The length of hospital stay is calculated to provide convenience for observation.

```
DROP FUNCTION fn_CountLengthOfStay

CREATE FUNCTION fn_CountLengthOfStay (@AdmissionID INT)
RETURNS INT
AS BEGIN
    DECLARE @CountDays INT =
        (SELECT DATEDIFF (DAY, AdmissionDate , COALESCE (DischargeDate, getdate()) )
         FROM dbo.Admission
         WHERE AdmissionID = @AdmissionID)
    SET @CountDays = isnull(@CountDays , 0)
    RETURN @CountDays
END

ALTER TABLE dbo.Admission
ADD LengthOfStay AS (dbo.fn_CountLengthOfStay (AdmissionID))
```

- The age is calculated to provide convenience to analyze patients' age distribution.

```
CREATE FUNCTION fn_CountAge (@PatientID INT)
RETURNS INT
AS BEGIN
    DECLARE @CountAge INT =
        (SELECT DATEDIFF (HOUR, BirthDate , GETDATE())/8766
         FROM dbo.Patient
         WHERE PatientID = @PatientID)
    RETURN @CountAge
END

ALTER TABLE dbo.Patient
ADD Age AS (dbo.fn_CountAge (PatientID))
```

Database Implementation - Views

- View 1: Observed the relationships between the patient's' medical conditions and the prognosis.
- View 2: Tracked the physicians' treatments and the effectiveness
- View 3: Tracked the geographical distribution of COVID-19 patients.
- View 4: Looked up for unoccupied beds.

```

CREATE VIEW V_UnoccupiedBeds
AS
SELECT t.BedID, CareLevel
FROM
(
  (SELECT DISTINCT BedID FROM BED
  EXCEPT
  SELECT DISTINCT BedID FROM PatientBed )
union
  (SELECT DISTINCT BedID FROM PatientBed
  EXCEPT
  SELECT DISTINCT BedID FROM PatientBed
  WHERE EndDate is null)
) t
JOIN Bed b
ON t.BedID = b.BedID

SELECT * FROM V_UnoccupiedBeds

```

V_UnoccupiedBeds	
123	BedID
123	CareLevel

	Grid	123 BedID	123 CareLevel
		123 BedID	123 CareLevel
Text	1	100	1
	2	102	1
	3	200	2
	4	201	2
	5	301	3
	6	401	4
	7	402	4
	8	403	4

Database Implementation - Column Data Encryption

- Encrypted the current condition of patients' medical history with passwords.

MedicalHistory Patient ID	MedicalHistory Encrypt CurrentCondition
1	00D16E8C8821C4409AFF39...
2	00D16E8C8821C4409AFF39...
2	00D16E8C8821C4409AFF39...
3	00D16E8C8821C4409AFF39...
4	00D16E8C8821C4409AFF39...
5	00D16E8C8821C4409AFF39...
6	00D16E8C8821C4409AFF39...
8	00D16E8C8821C4409AFF39...
10	00D16E8C8821C4409AFF39...
11	00D16E8C8821C4409AFF39...

```
CREATE MASTER KEY  
ENCRYPTION BY PASSWORD = 'Pa$$word'
```

```
CREATE CERTIFICATE CertTest  
with SUBJECT = 'Group Test Certificate',  
EXPIRY_DATE = '2026-10-31'
```

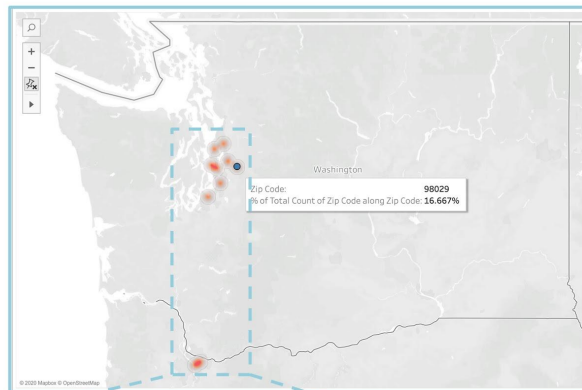
```
CREATE SYMMETRIC KEY TestSymmetric  
WITH ALGORITHM = AES_128  
ENCRYPTION BY CERTIFICATE CertTest
```

```
OPEN SYMMETRIC KEY TestSymmetric  
DECRYPTION BY CERTIFICATE CertTest
```

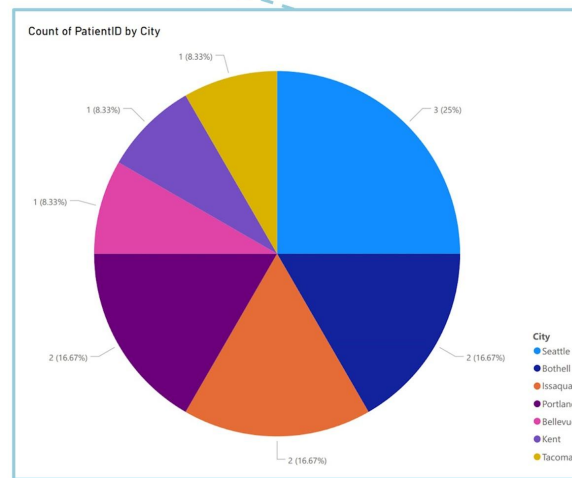
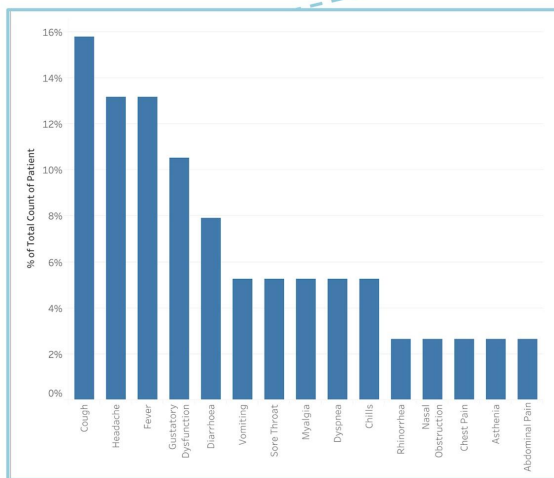
```
Insert  
INTO dbo.MedicalHistory  
(PatientID, MedicalConditionID, StartDate, EndDate, CurrentCondition)  
VALUES  
(2, 1, '2009-06-09', '2019-09-30', EncryptByKey(Key_GUID(N'TestSymmetric'), CONVERT(VARBINARY(250), 'Cured')))
```

Report & Visualization

- The top 3 most-affected cities are Seattle, Bothell and Issaquah.



- The 3 most commonly reported symptoms are cough, fever, and headache.





Thank you !
Q&A