## Lecture Title and Date

Database for Healthcare: Overview and Examples, Mon. 2/3

## Objectives of the Lecture

- Understand the relational data model and how it represents data and the relationships across it
- Explain the concept of a Relational Database Management System (RDBMS)
- Describe what constraints are in terms of a relational data model, like domain, key, and referential integrity.
- Understand the purpose of the three types of normalization in relational databases, particularly the different normal forms (1NF, 2NF, and 3NF) in its efforts to minimize data redundancy.
- Understand the purpose of and basic examples of Structured Query Language (SQL) in how it can be used to manipulate a database.
- Define what a data warehouse and data mart are, especially in how they contribute to online analytical processing.

## Key Concepts and Definitions

Relational (SQL) database:

- Database built on the relational data model in which data is stored in the form of tables that have relations to each other
- Collections of tables represent data itself and relationships between them (keys that map between tables)
- Each table has multiple columns (attributes) with unique names
- Relational database is defined as the implementation of the relational model, this is often referred to in full form as a Relational Database Management System (RDBMS)

Constraints in Relational Model:

- Domain constraint e.g., Age > 0 for column “Age”
- Key constraint: every table needs a column (or set of columns) that serve as a unique identifier, called a primary key (PK)
- Referential integrity: a column that acts as a foreign key (FK) must point to an existing column in the same or a different table

Normalization: process to organize data to minimize redundancy and remove inconsistent dependencies.

- 1NF: used to remove repeated groups
- 2NF: eliminates partial dependencies to further reduce redundancy (split the table into two or more separate tables if necessary)
- 3NF: removes transitive dependencies

Structured Query Language (SQL):

- Standard language for accessing and manipulating relational databases
- Supported operations: create new database, create new table, insert record, update record, delete record, and data retrieval
- SQL Keywords: PIVOT - turns row values into multiple columns, UNPIVOT - turns multiple columns into multiple rows

Data Warehouse: a central repository that integrates large amounts of data from various databases

- example: CDW

Data Mart: a subset of a data warehouse that focuses on a specific area

- example: MIMIC

## Main Content/Topics

Relational database constraints

Relational databases rely on three constraints: domain constraint, key constraint, and referential integrity. Each of these are fairly straightforward and almost always implied when looking at linked tables in a database, but important to uphold for larger systems. Domain constraint simply requires that each column only take values that lie inside a domain range. This is to avoid errant entries. Key constraint requires that each table have a unique identifier for each row. This column is called the **primary key**. This could be the index of the table (ex. 0-based), or the ID of the patient if every row is a different patient. Finally, referential integrity refers to how columns are mapped across tables. It requires that the **foreign key** (used to match to another table) of one table must have a matching **primary key** in another table.

Normalization

**First normal form** requires each column in a table to contain only single values. The example provided in class is when a patient has multiple comma-separated phone numbers in a single column, this needs to be corrected so each phone number has its own unique entry.

**Second normal form** reduces table size and splits into different tables to reduce partial redundancy and eliminate partial dependencies. The example may be a bit tricky, but I will outline it (shown below). In its rawest form, Patient_Name is directly derived from Patient_ID and Provider_Name is directly derived from Provider_ID. However, because all these columns are stored in the same table, there are partial dependencies that exist (ex. Patient_Name → Provider_ID → Provider_Name). In other words, reduce the tables so that all the columns are directly derived from the **primary key.** If there are other dependencies in the table, you should put those in a separate table.

| Patient_ID | Provider_ID | Patient_Name | Provider_Name |
|---|---|---|---|
| 1 | D1 | John Smith | Susan Lee |
| 2 | D2 | Mary Doe | Peter Pan |
| 3 | D1 | Kay Stone | Susan Lee |

| Patient_ID | Patient_Name |
|---|---|
| 1 | John Smith |
| 2 | Mary Doe |
| 3 | Kay Stone |

| Provider_ID | Provider_Name |
|---|---|
| D1 | Susan Lee |
| D2 | Peter Pan |

**Third normal form** is slightly different from the second normal form, but removes transitive dependencies rather than "partial" dependencies. This is when a primary key maps to a column, which matches to another column- that second matching step needs to be contained in a separate table. Using the example below, Patient_ID/Name doesn't necessarily directly map to a city, but because Zip_Code is an intermediate, the transitive dependency is present. A second table would fix this issue.

| Patient_ID | Patient_Name | Zip_Code | City |
|---|---|---|---|
| 1 | John Smith | 06511 | New Haven |
| 2 | Mary Doe | 06611 | Trumbull |

| Patient_ID | Patient_Name | Zip_Code |
|---|---|---|
| 1 | John Smith | 06511 |
| 2 | Mary Doe | 06611 |

| Zip_Code | City |
|---|---|
| 06511 | New Haven |
| 06611 | Trumbull |

<u>SQL details</u>

SQL is a language used for interacting with relational databases. SQL can be used to create new databases, insert new records, update existing records, delete records, or to retrieve data within queries. *SELECT* is a statement primarily used for developing such queries. Paired with other statements like *JOIN*, one can isolate specific columns that share a common characteristic across different tables and combine them all within a single query.

SQL is also used to investigate data within data warehouses (e.g. the Corporate Data Warehouse (CDW) at the VA). In this example, SQL can be used to identify and "wrangle" the VA sites that perform cardiac CT scans for calcium scoring and observe how such scans have changed over time, or even compare one VA to another.

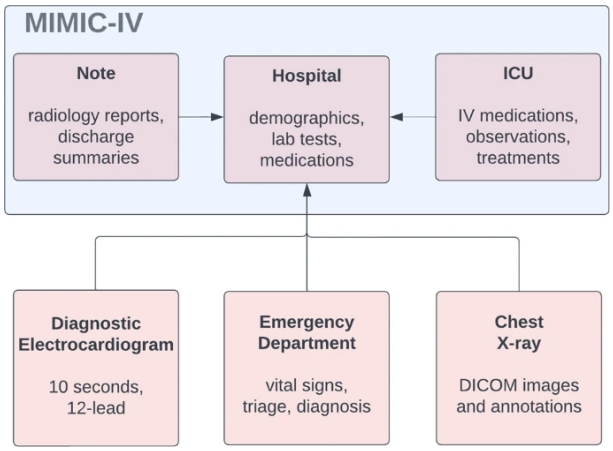
SQL is also used to query multiple tables together by joining them. In our VA example, this would be using SQL to filter for relevant CPT codes to ensure that patients and procedures are indeed real.

<u>Database examples</u>

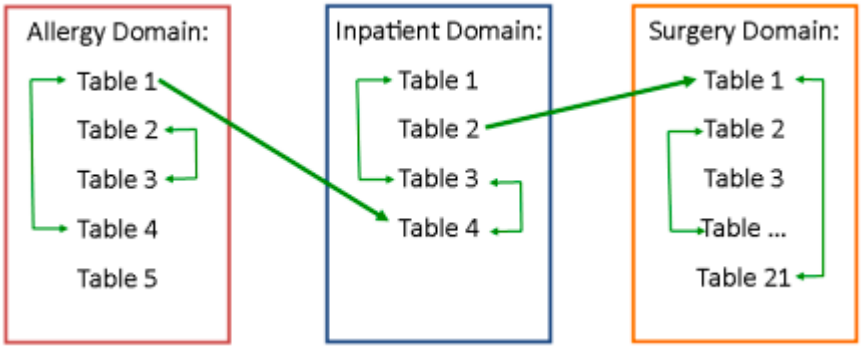
Two types of relational databases were discussed in class: data warehouse and data mart. Each has the same structure, but data marts are focused on a specific area such that it is designed for targeted users. These integrate large amounts of data from different databases.

OLAP (online analytical processing) is needed to access them, for which only read/retrieval tasks may be performed

MIMIC (datamart): The Medical Information Mart for Intensive Care (MIMIC) database consists of deidentified electronic health records from the Beth Israel Deaconess Medical Center of Boston. This is widely used in the health informatics research space, and has had a tremendous impact in facilitating data sharing and enabling big data research. Larger databases are pushed to a central server from which there is conversion of data formatting, user oversight, and de-identification before the polished data mart is made available. The figure below outlines the different tables and how they map to one another.




Corporate Data Warehouse (CDW) at the VA: CDW takes in various derived datasets from different VA offices and a VX130 system that handles large intakes of VistA files. It organizes data into different domains, where each domain contains different tables, and creates linking keys that can traverse all the tables across each domain (figure below).




Cardiac CT query example: Want to figure out which VA sites perform the cardiac CT scan and how the use of this test has changed over time.

- Tables involved: Patient table, workload/procedure table, and the CT dimensions table

1. Get CPT (not sure what the P stands for but that's the name of the CT table) data by querying CPT codes (these stand for specific procedures that we know we want to look for)

```
use cdwwork
go
drop table if exists #CPTCodes;
select CPTCode, CPTSID
into #CPTCodes
from Dim.CPT
where CPTCode in ('0144T','0147T','0149T','75571');
```

2. Now we have the codes, we can join the three tables
    a. PatientICN from patient table
    b. Sta3n, PatientSID, vProcedureDataTime, and vProcedureDateSID from the workload/procedure table
    c. CPTCode from the CPT codes table

```
drop table if exists #CoronaryCalciumProcedures;
select
 b.PatientICN
,a.Sta3n
,a.PatientSID
,a.vProcedureDateTime
,a.vProcedureDateSID
,c.CPTCode
,b.CDWPossibleTestPatientFlag
into #CoronaryCalciumProcedures
from Outpat.WorkloadVProcedure as a
left join Spatient.Spatient as b
on a.PatientSID = b.PatientSID
join #CPTCodes as c
on a.CPTSID = c.CPTSID
where a.VisitDateTime >= cast('1/1/2006' as datetime2(0))--partition key
and b.PatientICN is not null
and b.PatientICN not like '%missing%'
and b.PatientICN not like '%unknown%'
and isnull(b.CDWPossibleTestPatientFlag,'N') <> 'Y';--ALWAYs remove test patients!
```

Make sure Patients are real

## Suggest references for many of the key concepts


Johnson, A.E.W., Bulgarelli, L., Shen, L. et al. MIMIC-IV, a freely accessible electronic health record dataset. Sci Data 10, 1 (2023). https://doi.org/10.1038/s41597-022-01899-x

Plummer, N. R. (2023, January 19). Databases in clinical practice. Anaesthesia & Intensive Care Medicine. https://www.sciencedirect.com/science/article/pii/S1472029922002910

The use of databases in clinical practice by Plummer.
https://doi.org/10.1016/j.mpaic.2022.12.003