INTRODUCCIÓN AL MODELO DE DATOS OMOP-CDM

Alberto Labarga

2023-09-14

Telecommunications Engineer
Head of Biomedical Data Hub @BSC
More than 20 years teaching

open data - open source – open science

I am Alberto

alabarga
alabarga
/in/albertolabarga

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- European Bioinformatics Institute (Senior Data Scientist)
  - Led the Service Oriented Architecture
  - Genomics databases, web services, semantic web

- Navarrabiomed (Head of bioinformatics)
  - Led the genome sequencing bioinformatics infrastructure (NAGEN, PharmaNAGEN)
  - Multiomics data integration

- IOMED (Head of data)
  - Led data normalization efforts in more than 20 hospitals (OMOP-CDM, NLP)
  - 10M patients, 200million health records

- Barcelona Supercomputing Center (Head of Biomedical Data Hub)
  - Lead the Biomedical Data Hub
  - Open science, FAIR data

Descripción del curso

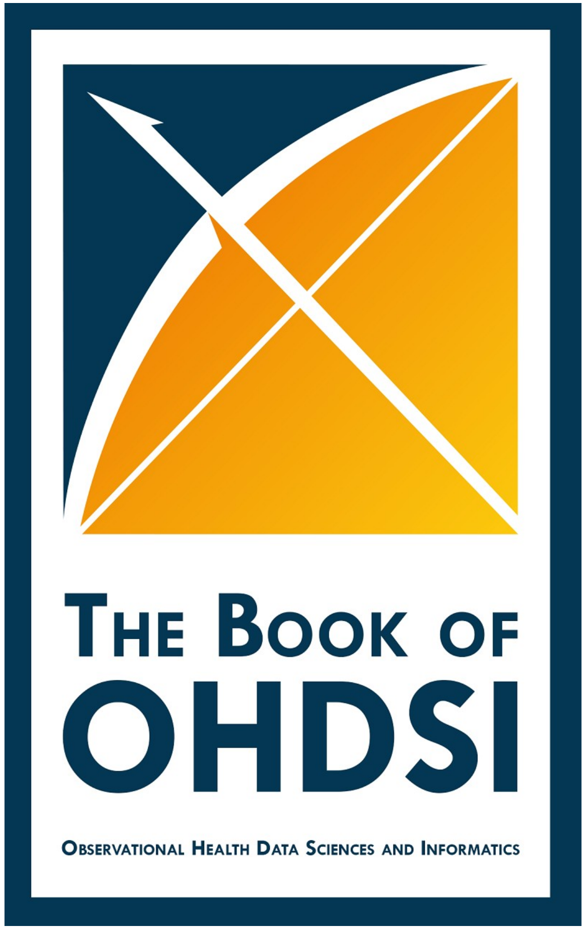
El curso tiene como objetivo proporcionar una comprensión profunda del modelo de datos **OMOP-CDM de la OHDSI**, sus vocabularios y cómo usar la **aplicación web Atlas** para crear cohortes y extraer datos para **estudios observacionales**. El curso cubrirá los conceptos y principios básicos del modelo de datos OMOP-CDM, incluida su estructura, tablas clave y vocabularios. Además, el curso cubrirá los aspectos prácticos de trabajar con Atlas, incluido cómo crear y ejecutar consultas de **cohortes**.

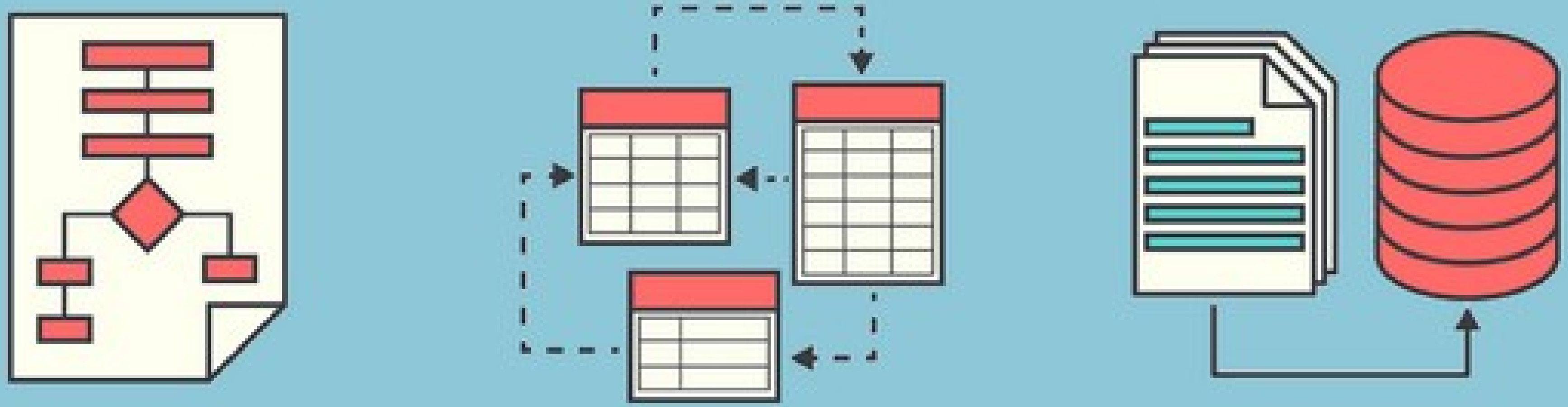
Esquema

- Introducción al modelo de datos OMOP-CDM
  - Descripción general del modelo de datos OMOP-CDM
  - Conceptos y principios clave del modelo de datos OMOP-CDM
  - Comprensión de la estructura y tablas clave del modelo de datos OMOP-CDM
- Vocabularios OMOP-CDM

Recursos

- Sitio web: https://github.com/alabarga/omop-cdm-course
- Diapositivas
- El Libro de OHDSI
- Entorno local (docker)
  - Base de datos OMOP-CDM
  - Consultas de ejemplo
  - Atlas





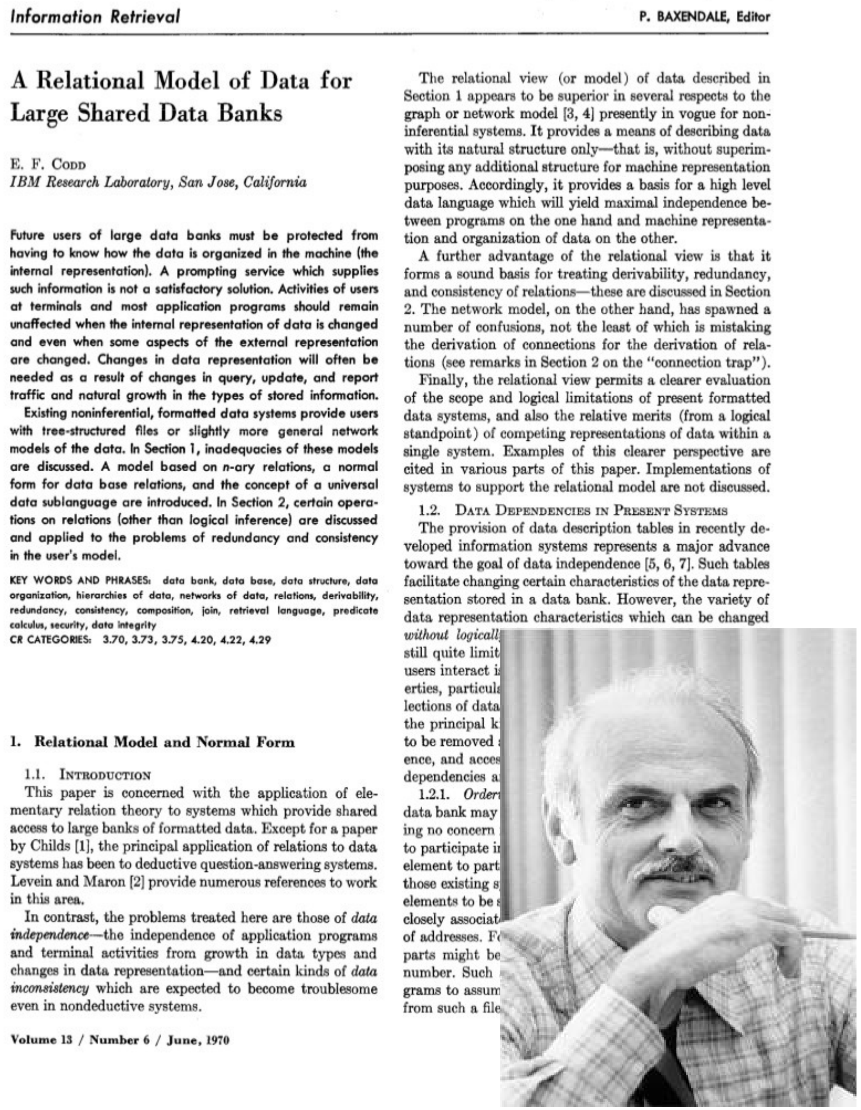
Databases

Bases de datos

- Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
- Un sistema **gestor de bases de datos** (SGBD) es un **software** que gestiona una o más bases de datos y nos permite explotar los datos almacenados en ellas de forma relativamente simple mediante **SQL**

Modelo relacional

Edgar Frank Codd, en su artículo"A Relational Model of Data for Large Shared Data Banks" en 1970, definió el modelo relacional y publicó una serie de reglas para la evaluación de administradores de sistemas de datos relacionales y así nacieron las bases de datos relacionales.



Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

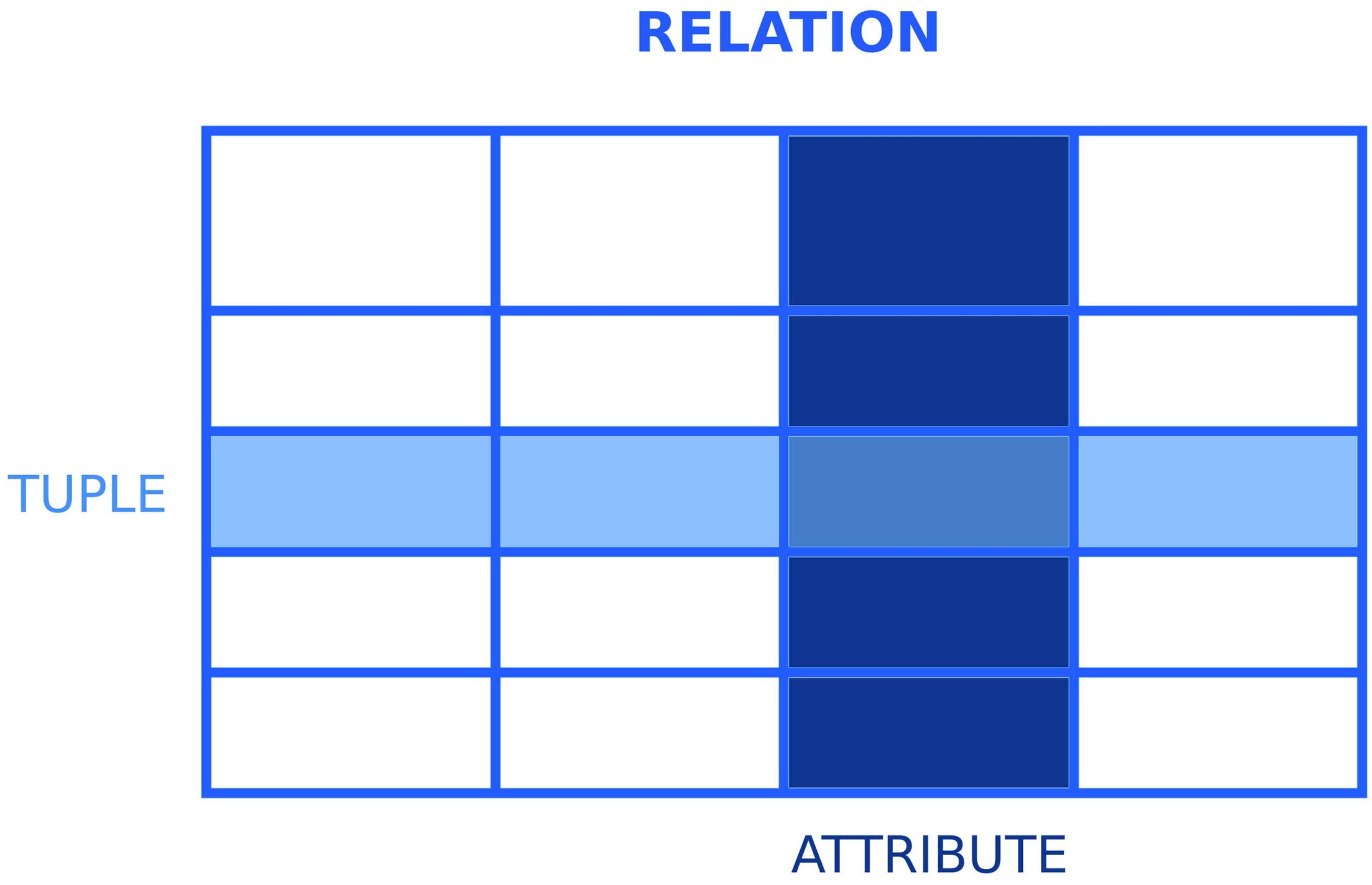
Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

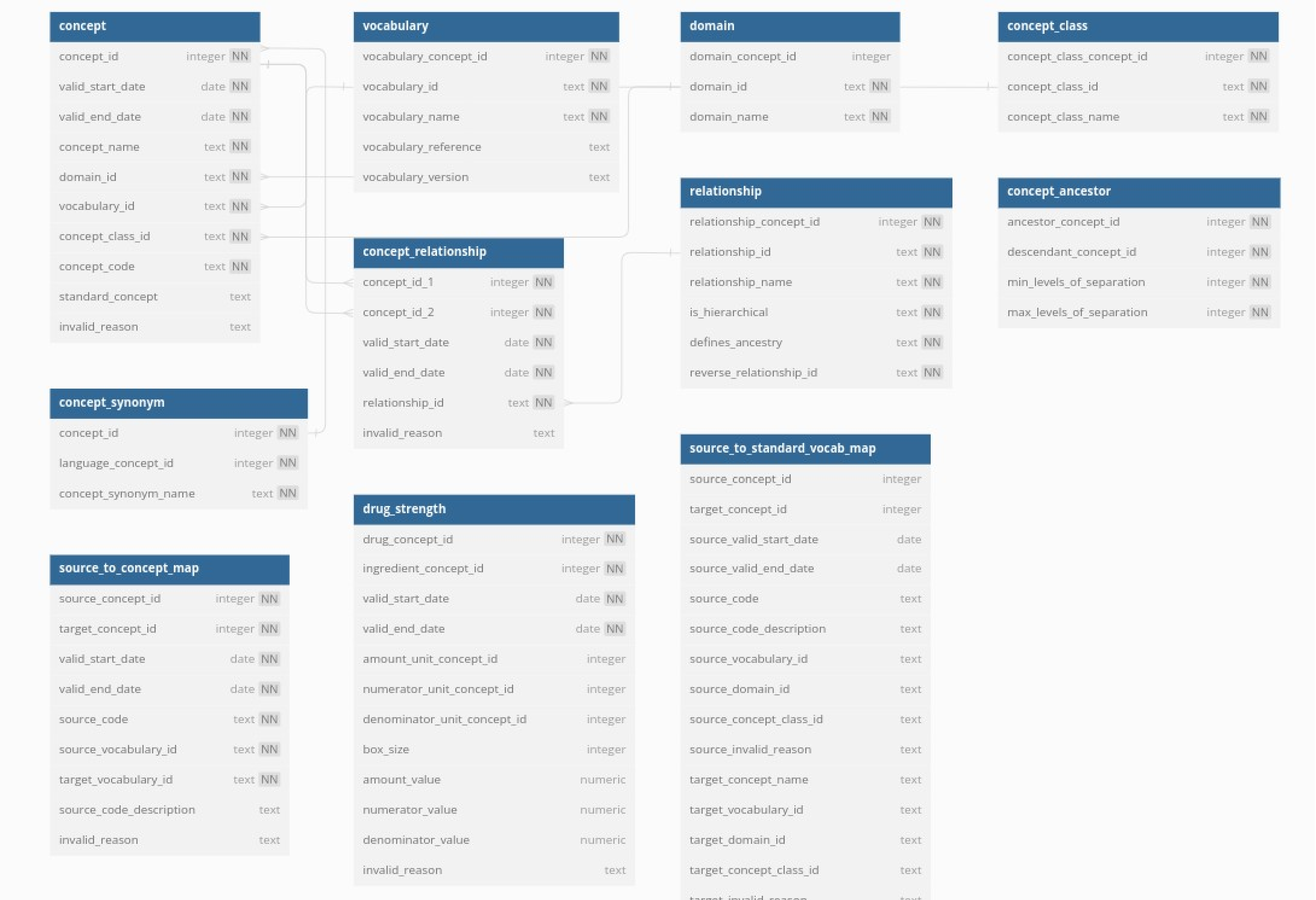
1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed
without logicall
still quite limit
users interact i
erties, particul
lections of data
the principal k
to be removed
ence, and acces
dependencies a
1.2.1. *Order*
data bank may
ing no concern
to participate i
element to part
those existing s
elements to be
closely associat
of addresses. F
parts might be
number. Such
grams to assum
from such a file

Volume 13 / Number 6 / June, 1970

Relation = Table





concept		
concept_id	integer	NN
valid_start_date	date	NN
valid_end_date	date	NN
concept_name	text	NN
domain_id	text	NN
vocabulary_id	text	NN
concept_class_id	text	NN
concept_code	text	NN
standard_concept	text	
invalid_reason	text	

vocabulary		
vocabulary_concept_id	integer	NN
vocabulary_id	text	NN
vocabulary_name	text	NN
vocabulary_reference	text	
vocabulary_version	text	

domain		
domain_concept_id	integer	
domain_id	text	NN
domain_name	text	NN

concept_class		
concept_class_concept_id	integer	NN
concept_class_id	text	NN
concept_class_name	text	NN

concept_relationship		
concept_id_1	integer	NN
concept_id_2	integer	NN
valid_start_date	date	NN
valid_end_date	date	NN
relationship_id	text	NN
invalid_reason	text	

relationship		
relationship_concept_id	integer	NN
relationship_id	text	NN
relationship_name	text	NN
is_hierarchical	text	NN
defines_ancestry	text	NN
reverse_relationship_id	text	NN

concept_ancestor		
ancestor_concept_id	integer	NN
descendant_concept_id	integer	NN
min_levels_of_separation	integer	NN
max_levels_of_separation	integer	NN

concept_synonym		
concept_id	integer	NN
language_concept_id	integer	NN
concept_synonym_name	text	NN

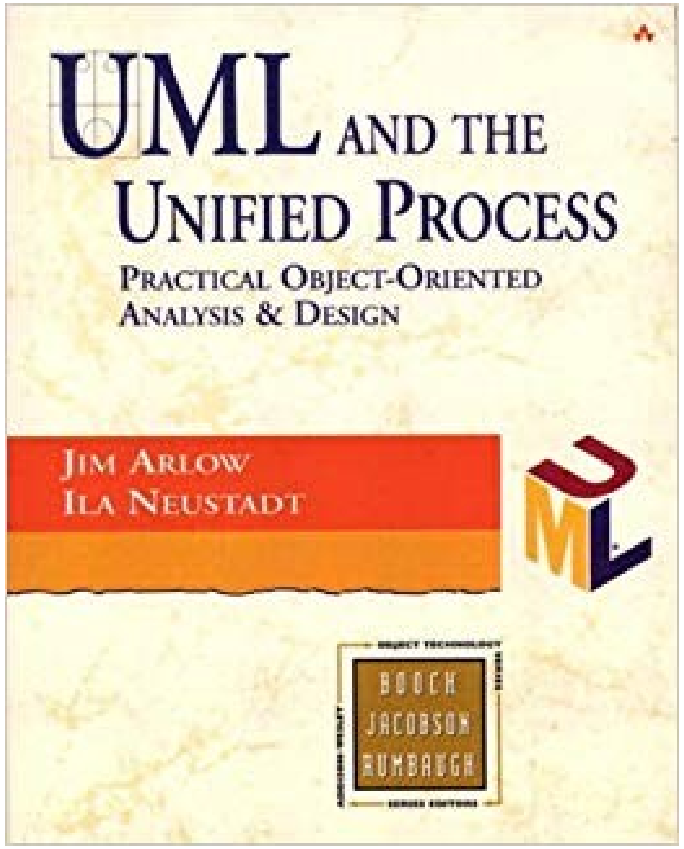
source_to_concept_map		
source_concept_id	integer	NN
target_concept_id	integer	NN
valid_start_date	date	NN
valid_end_date	date	NN
source_code	text	NN
source_vocabulary_id	text	NN
target_vocabulary_id	text	NN
source_code_description	text	
invalid_reason	text	

drug_strength		
drug_concept_id	integer	NN
ingredient_concept_id	integer	NN
valid_start_date	date	NN
valid_end_date	date	NN
amount_unit_concept_id	integer	
numerator_unit_concept_id	integer	
denominator_unit_concept_id	integer	
box_size	integer	
amount_value	numeric	
numerator_value	numeric	
denominator_value	numeric	
invalid_reason	text	

source_to_standard_vocab_map	
source_concept_id	integer
target_concept_id	integer
source_valid_start_date	date
source_valid_end_date	date
source_code	text
source_code_description	text
source_vocabulary_id	text
source_domain_id	text
source_concept_class_id	text
source_invalid_reason	text
target_concept_name	text
target_vocabulary_id	text
target_domain_id	text
target_concept_class_id	text
target_invalid_reason	text

Unified Modeling Language

- UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.
- El estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación.



SQL

SQL

- El lenguaje principal de una base de datos relacional es SQL (Structured Query Language).
- Ofrece funcionalidades para:
  - Definir (DDL - alteramos estructuras de las tablas)
  - Manipular (DML - trabajamos con los datos)

SQL components

- **Data Definition Language (DDL)**
  - **Deals with structural aspect of the database : creation, modification, deletion of tables**
- Data Manipulation Language (DML)
  - This allows modification of the data contained in the tables: insertion, deletion, selection,changing (even aggregation i.e count,sum,average)
- Data Control Language (DCL)
  - This deals with maintaining the security of the database using permissions and access control
- Transaction Control Language (TCL)
  - This deals with maintaining the integrity of the database using permissions, transactions

CREATE

- CREATE DATABASE my_database
- CREATE SCHEMA my_schema
- CREATE TABLE my_table

OMOP-CDM DB schemas

- raw: datos originales
- cdm: tablas clínicas
- vocabularies: vocabularios
- nlp: resultados de nlp
- results: cohortes, calidad, etc
- webapi: Atlas
- tmp: tablas temporales (Oracle)

CREATE TABLE

```
CREATE TABLE [IF NOT EXISTS] [schema_name].table_name (
column_1 data_type PRIMARY KEY,
column_2 data_type NOT NULL UNIQUE,
column_3 data_type DEFAULT 0,
table_constraints
)
```

Data types

- NULL. The value is a NULL value.
- INTEGER. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- REAL. The value is a floating point value, stored as an 8-byte IEEE floating point number.
- TEXT. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- BLOB. The value is a blob of data, stored exactly as it was input.

FOREIGN KEY

-
  ```
  FOREIGN KEY (foreign_key_columns)
     REFERENCES parent_table(parent_key_columns)
        ON UPDATE action
        ON DELETE action;
  ```

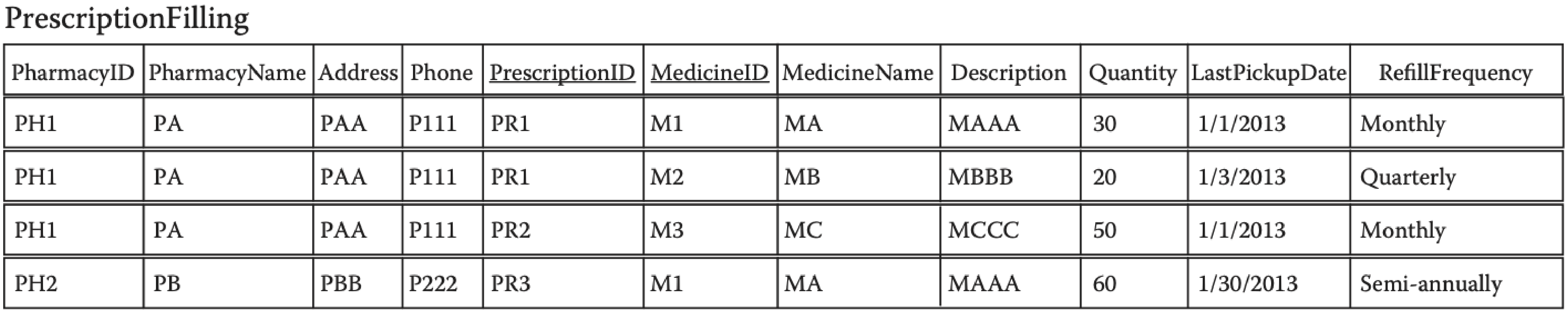
-
  ```
  ALTER TABLE child ADD CONSTRAINT fk_child_parent
                  FOREIGN KEY (parent_id)
                  REFERENCES parent(id);
  ```

DROP

- DROP TABLE table_name

- ALTER TABLE table_name
  DROP COLUMN column_name;

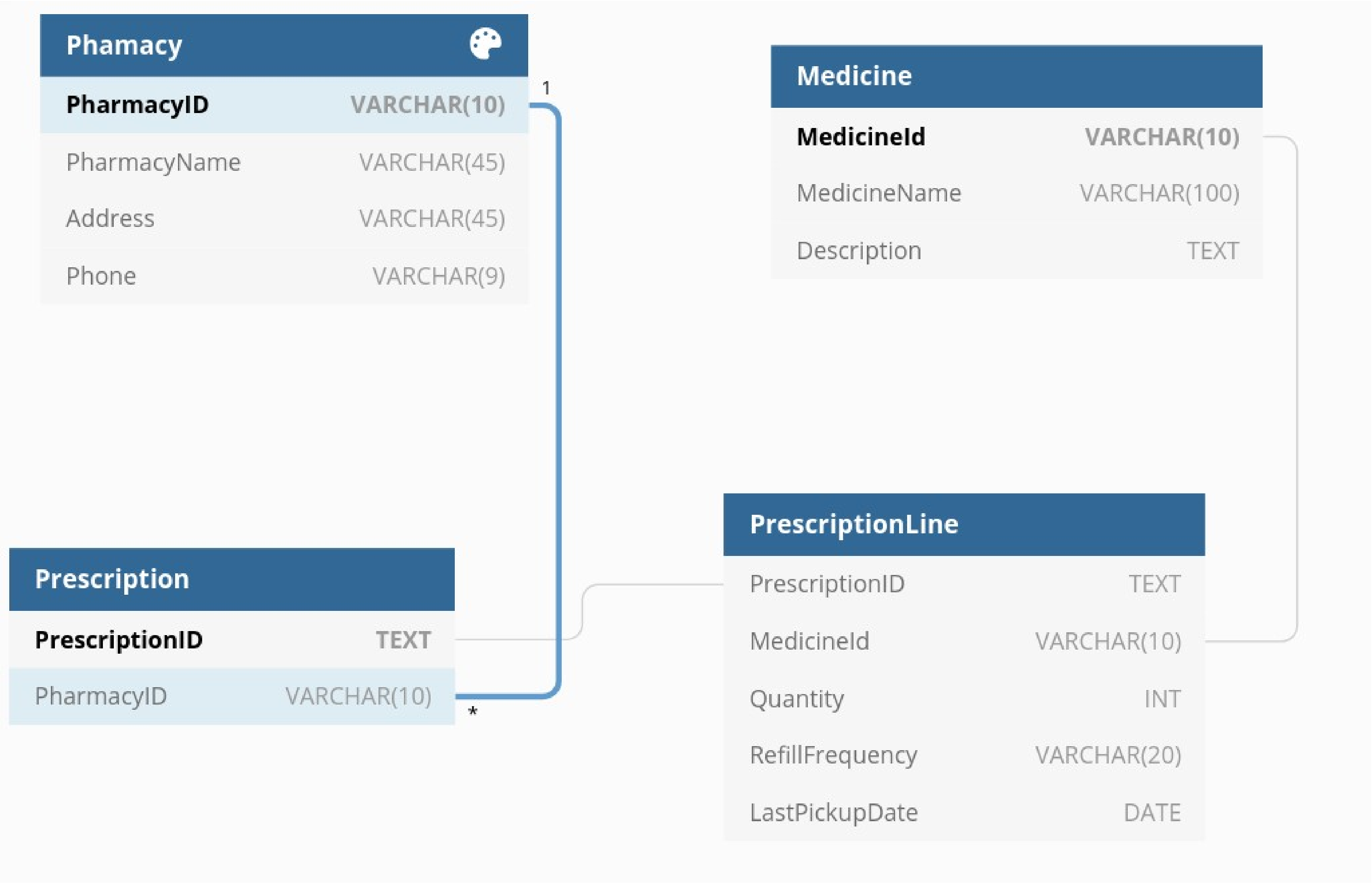
Normalization



PrescriptionFilling

PharmacyID	PharmacyName	Address	Phone	PrescriptionID	MedicineID	MedicineName	Description	Quantity	LastPickupDate	RefillFrequency
PH1	PA	PAA	P111	PR1	M1	MA	MAAA	30	1/1/2013	Monthly
PH1	PA	PAA	P111	PR1	M2	MB	MBBB	20	1/3/2013	Quarterly
PH1	PA	PAA	P111	PR2	M3	MC	MCCC	50	1/1/2013	Monthly
PH2	PB	PBB	P222	PR3	M1	MA	MAAA	60	1/30/2013	Semi-annually

Normalization



Phamacy	
PharmacyID	**VARCHAR(10)**
PharmacyName	VARCHAR(45)
Address	VARCHAR(45)
Phone	VARCHAR(9)

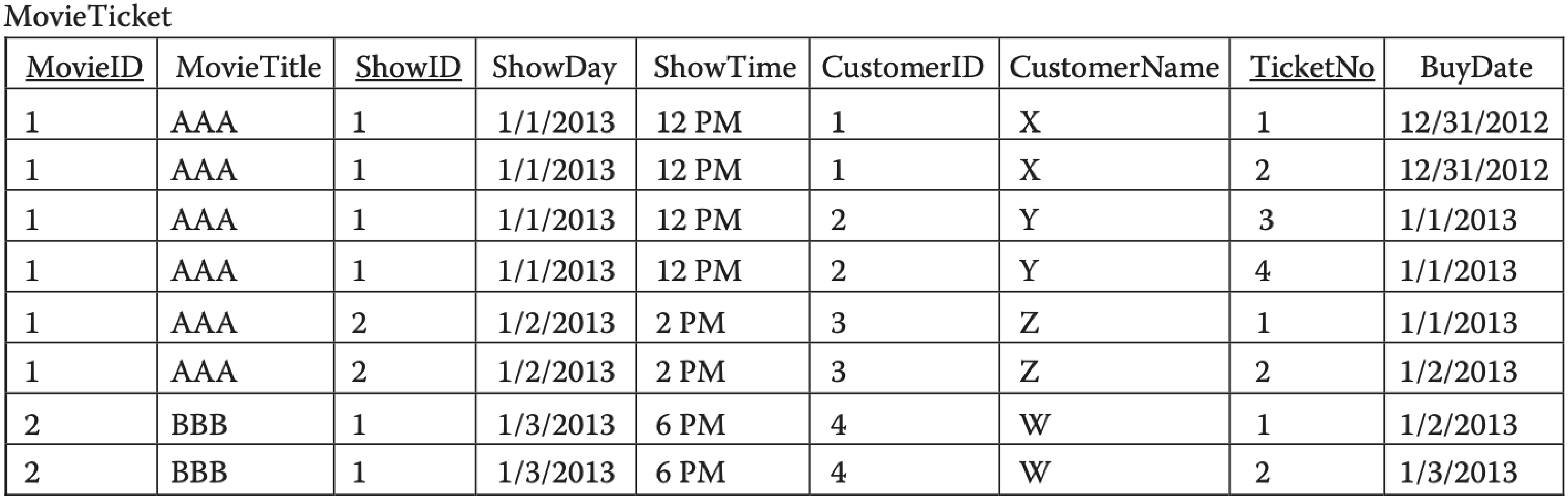
Medicine	
MedicineId	**VARCHAR(10)**
MedicineName	VARCHAR(100)
Description	TEXT

Prescription	
PrescriptionID	**TEXT**
PharmacyID	VARCHAR(10)

PrescriptionLine	
PrescriptionID	TEXT
MedicineId	VARCHAR(10)
Quantity	INT
RefillFrequency	VARCHAR(20)
LastPickupDate	DATE

https://dbdiagram.io/d/5f33ba32e1246d54aa2d0f95

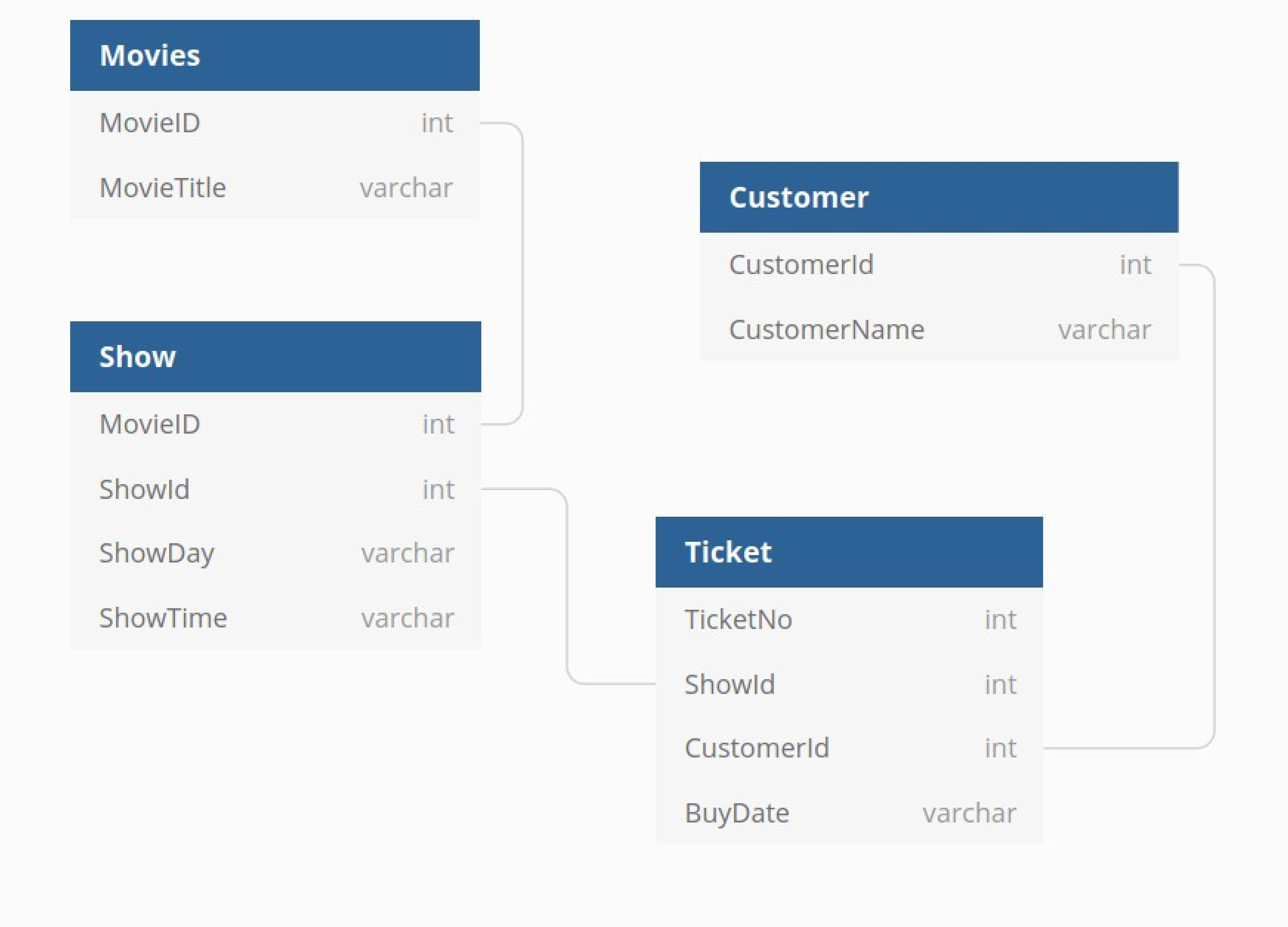
Normalization



MovieTicket

MovieID	MovieTitle	ShowID	ShowDay	ShowTime	CustomerID	CustomerName	TicketNo	BuyDate
1	AAA	1	1/1/2013	12 PM	1	X	1	12/31/2012
1	AAA	1	1/1/2013	12 PM	1	X	2	12/31/2012
1	AAA	1	1/1/2013	12 PM	2	Y	3	1/1/2013
1	AAA	1	1/1/2013	12 PM	2	Y	4	1/1/2013
1	AAA	2	1/2/2013	2 PM	3	Z	1	1/1/2013
1	AAA	2	1/2/2013	2 PM	3	Z	2	1/2/2013
2	BBB	1	1/3/2013	6 PM	4	W	1	1/2/2013
2	BBB	1	1/3/2013	6 PM	4	W	2	1/3/2013

Normalization



Movies	
MovieID	int
MovieTitle	varchar

Show	
MovieID	int
ShowId	int
ShowDay	varchar
ShowTime	varchar

Customer	
CustomerId	int
CustomerName	varchar

Ticket	
TicketNo	int
ShowId	int
CustomerId	int
BuyDate	varchar

SQL components

- Data Definition Language (DDL)
  - Deals with structural aspect of the database : creation, modification, deletion of tables
- **Data Manipulation Language (DML)**
  - **This allows modification of the data contained in the tables: insertion, deletion, selection,changing (even aggregation i.e count,sum,average)**
- Data Control Language (DCL)
  - This deals with maintaining the security of the database using permissions and access control
- Transaction Control Language (TCL)
  - This deals with maintaining the integrity of the database using permissions, transactions

SELECT

- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables

SELECT

- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables
- WHERE clause
  - Provides constraints on the records to be returned

SELECT

- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables
- ORDER BY clause
  - Sorts the records that are returned

SELECT

- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables
- GROUP BY clause
  - Eliminates duplication and provides for using aggregate functions

SELECT

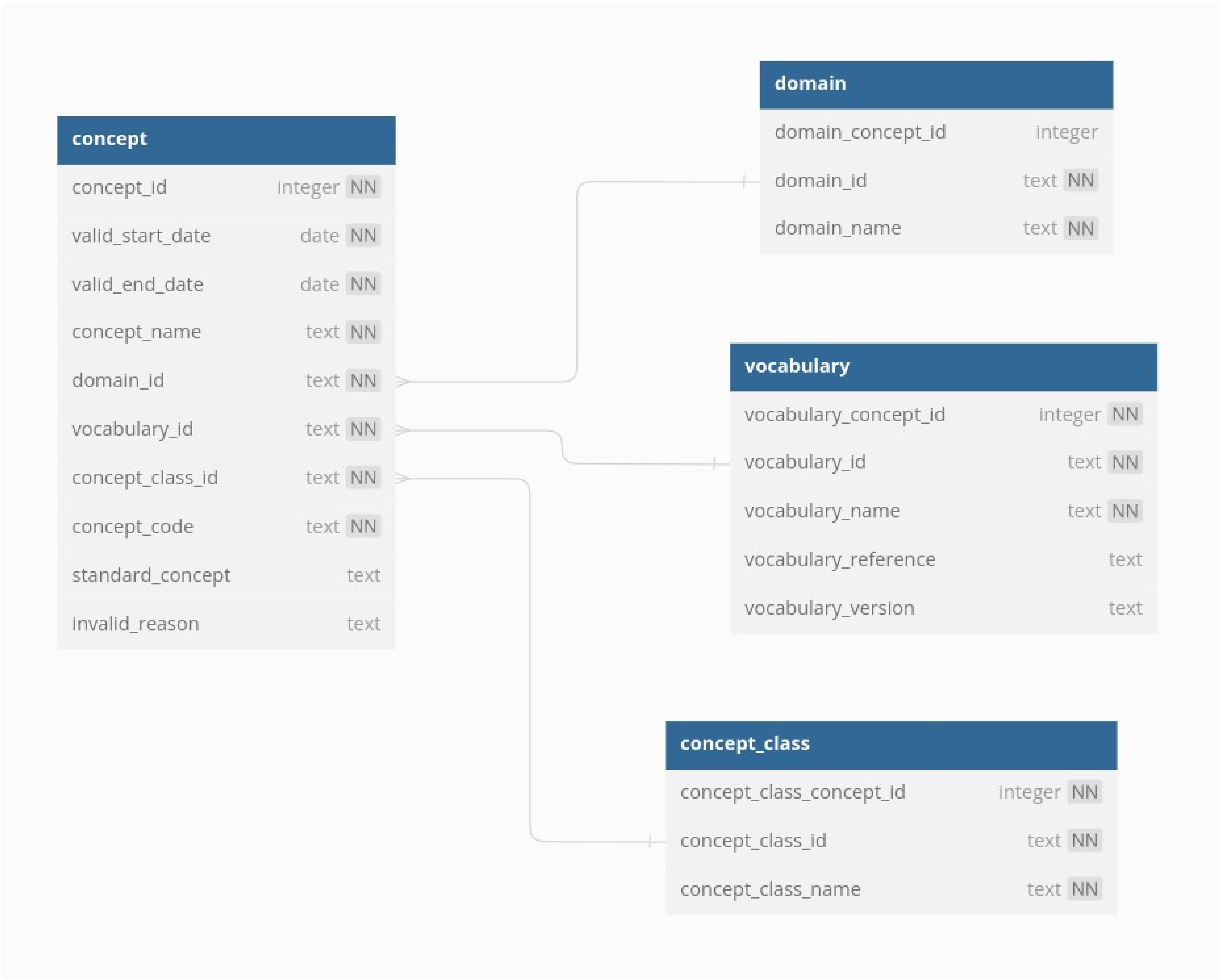
- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables
- GROUP BY clause
  - Eliminates duplication and provides for using aggregate functions
- HAVING clause
  - Provides constraints on items in a GROUP BY clause, including aggregate functions

SELECT

- SELECT statement
  - Declares the fields to be returned by the query
  - ‘*’ returns all fields
- FROM clause
  - Defines the table being queried
  - Can contain JOIN clauses to query multiple tables
- WHERE clause
  - Provides constraints on the records to be returned
- GROUP BY clause
  - Eliminates duplication and provides for using aggregate functions
- HAVING clause
  - Provides constraints on items in a GROUP BY clause, including aggregate functions
- ORDER BY clause
  - Sorts the records that are returned

JOIN

- SELECT
- FROM table1 t1
- INNER JOIN table2 t2
- ON t1.pk = t2.fk



concept		
concept_id	integer	NN
valid_start_date	date	NN
valid_end_date	date	NN
concept_name	text	NN
domain_id	text	NN
vocabulary_id	text	NN
concept_class_id	text	NN
concept_code	text	NN
standard_concept	text	
invalid_reason	text	

domain		
domain_concept_id	integer	
domain_id	text	NN
domain_name	text	NN

vocabulary		
vocabulary_concept_id	integer	NN
vocabulary_id	text	NN
vocabulary_name	text	NN
vocabulary_reference	text	
vocabulary_version	text	

concept_class		
concept_class_concept_id	integer	NN
concept_class_id	text	NN
concept_class_name	text	NN

INSERT INTO

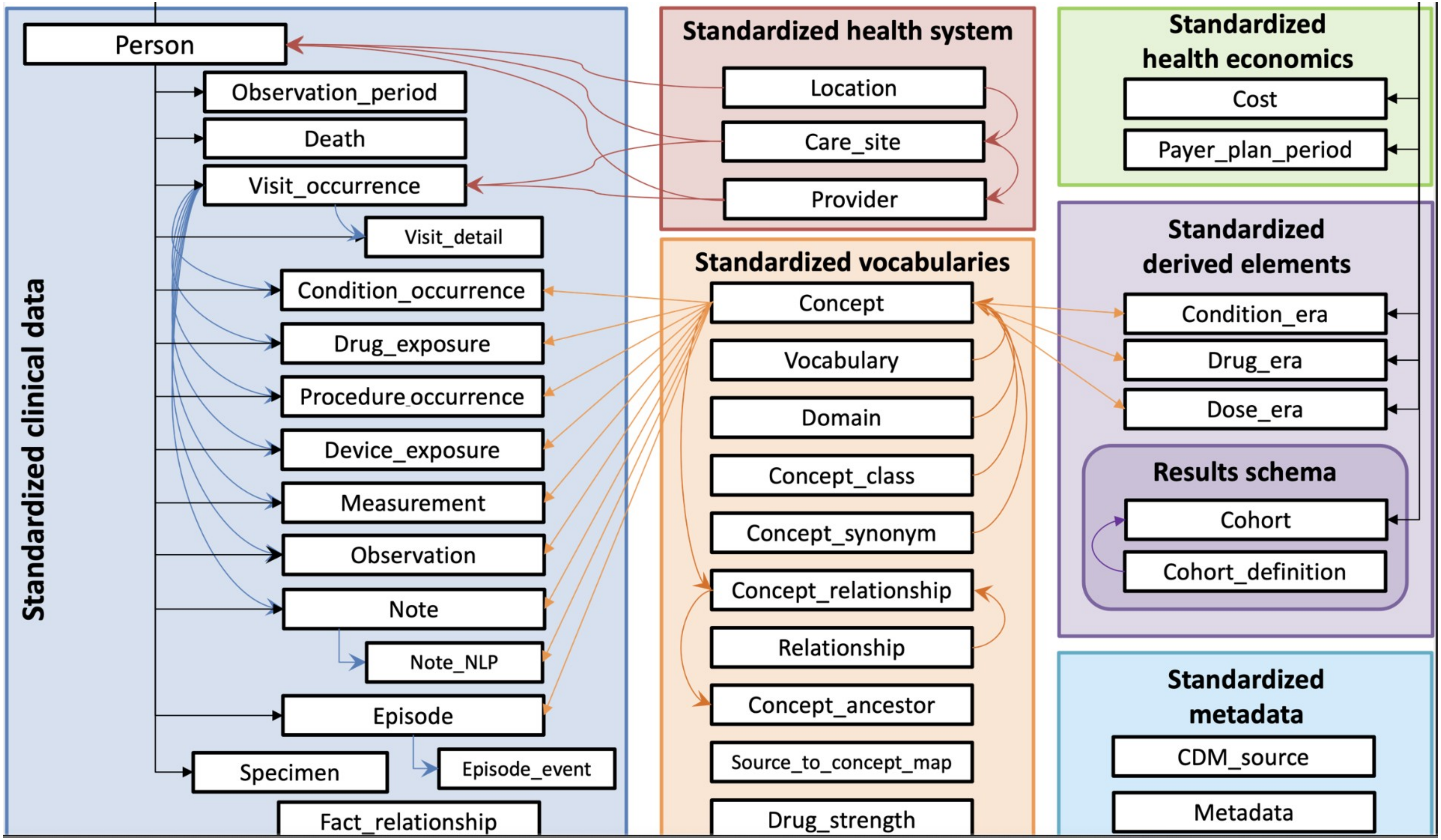
```
INSERT INTO table_2
SELECT field_1, field_2
FROM table$_1$
  [WHERE conditions];
```

UPDATE

```
UPDATE table
SET column_1 = new_value_1,
    column_2 = new_value_2
WHERE
    search_condition
```

DELETE

```
DELETE FROM table_name
WHERE condition
```


Standardized clinical data
Person
Observation_period
Death
Visit_occurrence
Visit_detail
Condition_occurrence
Drug_exposure
Procedure_occurrence
Device_exposure
Measurement
Observation
Note
Note_NLP
Episode
Specimen
Episode_event
Fact_relationship
Standardized health system
Location
Care_site
Provider
Standardized vocabularies
Concept
Vocabulary
Domain
Concept_class
Concept_synonym
Concept_relationship
Relationship
Concept_ancestor
Source_to_concept_map
Drug_strength
Standardized health economics
Cost
Payer_plan_period
Standardized derived elements
Condition_era
Drug_era
Dose_era
Results schema
Cohort
Cohort_definition
Standardized metadata
CDM_source
Metadata

Recursos

- Sitio web: https://github.com/alabarga/omop-cdm-course
- Diapositivas
- El Libro de OHDSI
- EHDEN academy
- OHDSI workshops
- Atlas tutorials
- Entorno local (docker)