Database Systems

Luiz Jonatã Pires de Araújo
l.araujo@innopolis.university

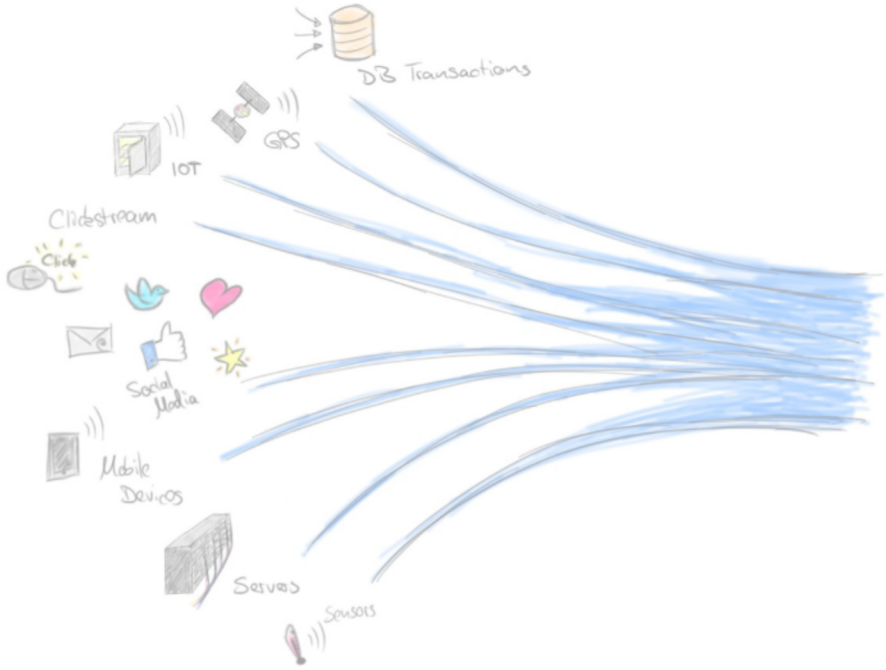
1. **Recap: Introduction; Relational Model; Normalization**

Giving credit where it's due:

- This material is based on the 7th edition of 'Fundamentals of Database Systems" by Elmasri and Navathe
- https://www.pearson.com/us/higher-education/program/Elmasri-Fundamentals-of-Database-Systems-7th-Edition/PGM189052.html

Outline

- Introduction to databases
- Relational Model
- Normalization



Introduction to databases

Concepts of databases: nature of data

- **Information systems**: An information system is a computer-based system which is integrated with the set of different components for the collection, process, storage and transmission of the data.
  It is used in decision making process.

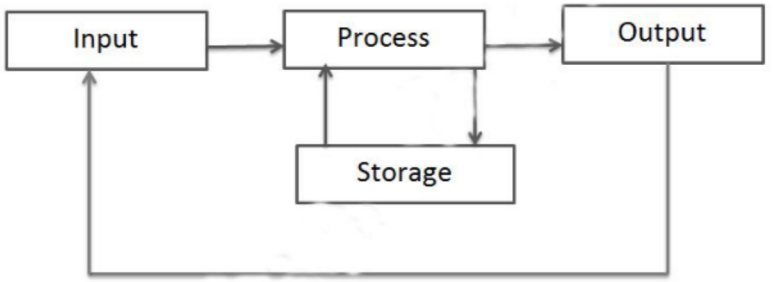


Fig: A logical model of Information System

- **Database**: a collection of related data, where data means recorded facts.
- A typical database represents some aspect of the real world.
- A **database management system** (DBMS) is a computerized system for implementing and maintaining a computerized database.

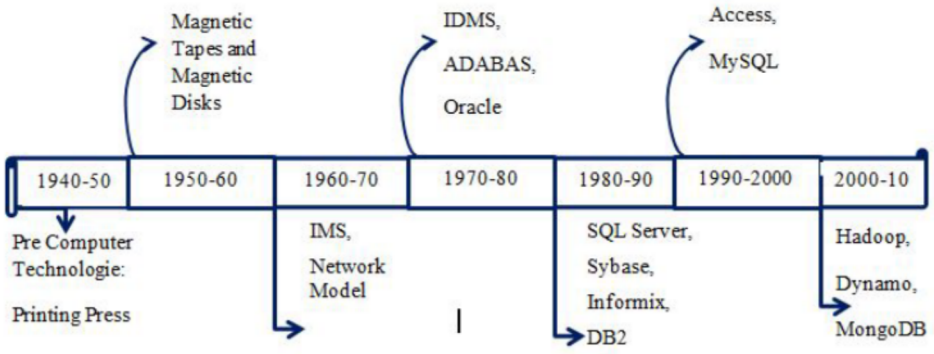
Concepts of databases: actors

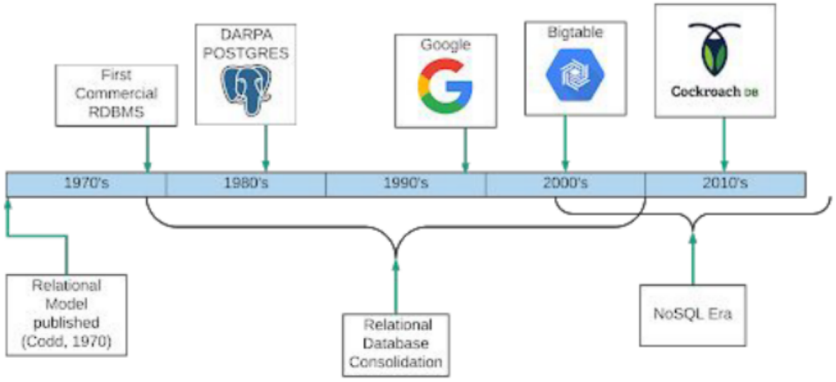
- **Database administrator** (DBA): Authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as security breaches and poor system response time.
- **Database Designers**: Identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.
- **System Analysts** and **Application Programmers** (**Software Engineers**): Determining the requirements of end users and develop specifications for standard canned transactions that meet these requirements. Application programmers implement these specifications as programs; then they test, debug, document, and maintain these canned transactions.

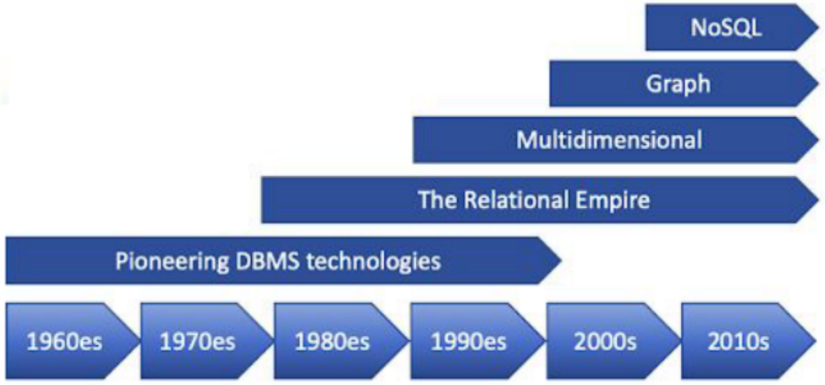
Concepts of databases: utilities

- Four basic functions of **CRUD**: create, read (aka retrieve), update, and delete data.
- **Loading**. A loading utility is used to load existing data files—such as text files or sequential files—into the database.
- **Backup**. A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium.
- **Database storage reorganization**. This utility can be used to reorganize a set of database files into different file organizations and create new access paths to improve performance.
- **Performance monitoring**. Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether to reorganize files or whether to add or drop indexes to improve performance.

Concepts of databases: timeline



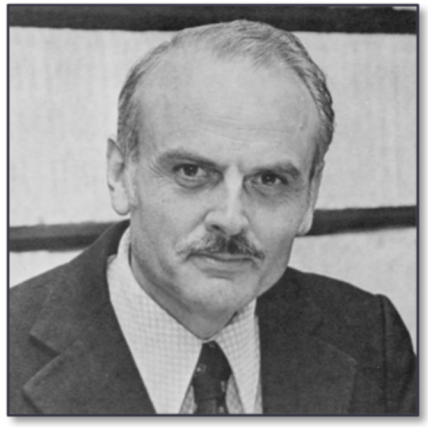




Relational Model

Relational Model Concepts

- The relational data model was first introduced by Ted Codd of IBM Research in 1970.
- The first commercial implementations of the relational model became available in the early 1980s.
- Current popular commercial relational DBMSs (RDBMSs) include DB2 (from IBM), Oracle (from Oracle), Sybase DBMS (now from SAP), and SQL Server and Microsoft Access (from Microsoft).
- In addition, several open source systems, such as MySQL and PostgreSQL, are available.



Source: IT Hall of fame

Definitions

- The relational model represents the database as a collection of tables of values or, to some extent, a flat file of records.
- Each row or record has a simple linear or flat structure.
- It represents a collection of related data values.
- A row represents a fact that typically corresponds to a real-world entity or relationship.
- The table name and column names are used to help to interpret the meaning of the values in each row.

STUDENT

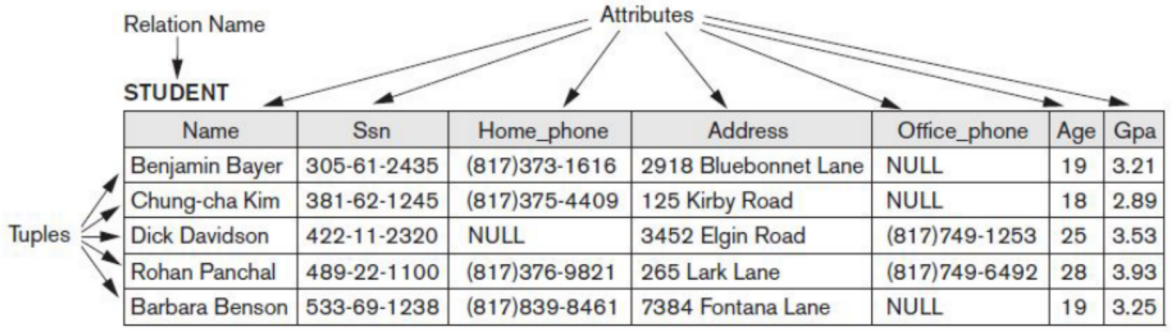
Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

Domains, Attributes, Tuples, and Relations

- All values in a column are of the same data type.
- In the formal relational model terminology, a row is called a tuple, a column header is called an attribute, and the table is called a relation.
- The data type describing the types of values that can appear in each column is represented by a domain of possible values.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Notation for relation: STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

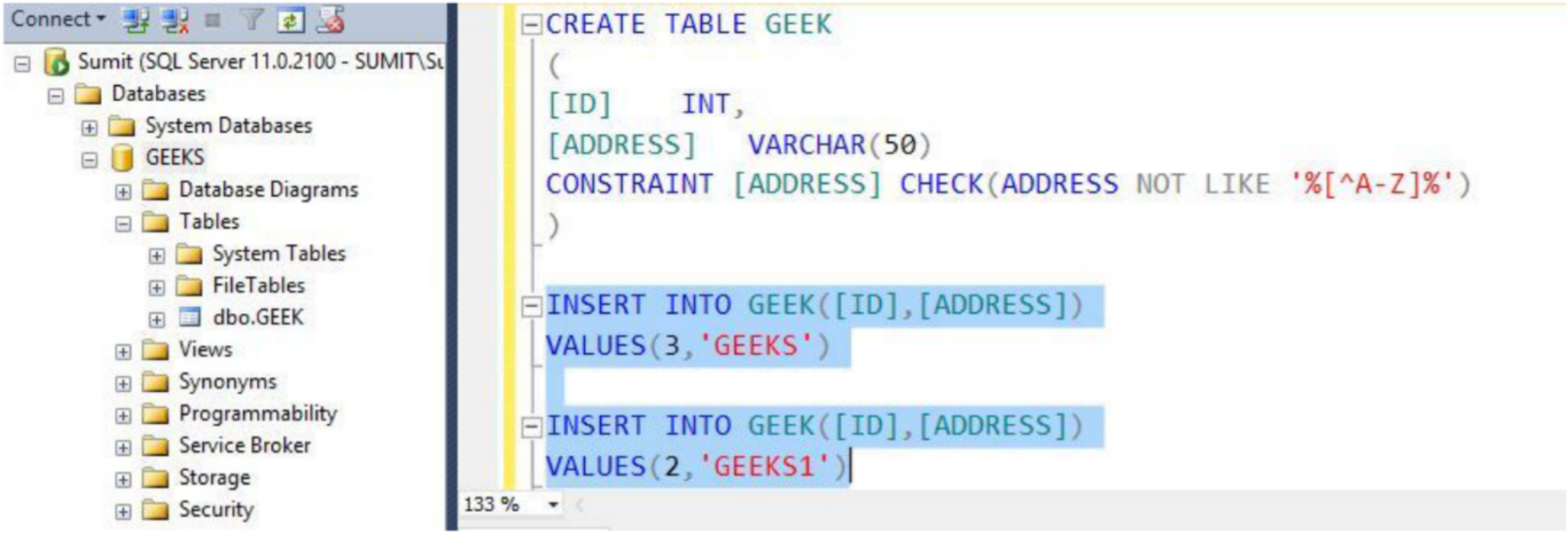
Constraints

- Domain
- Key
- NULL
- Entity integrity
- Referential integrity
- Domain related

Relational Model Constraints: Domain

Domain Constraints (within a single relation and its attributes)

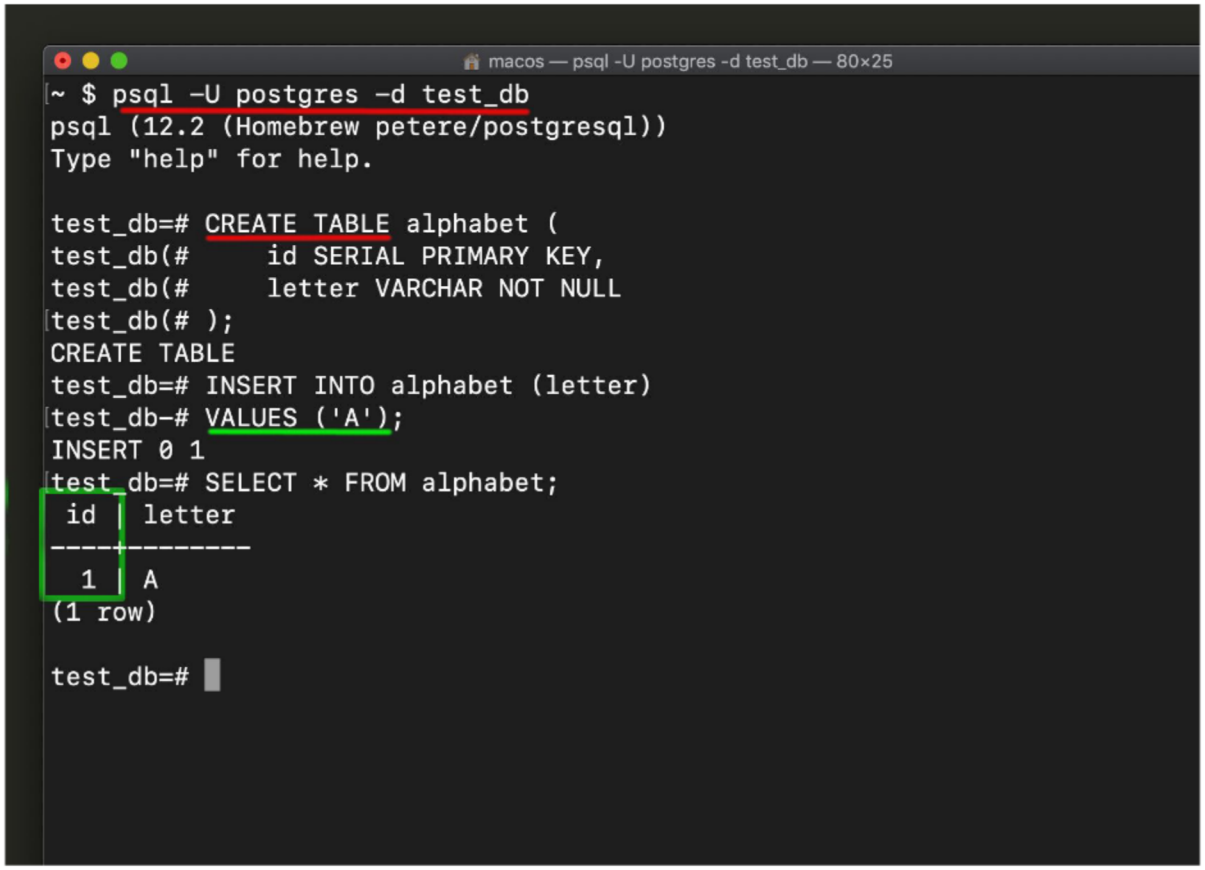
- Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A). Example: integers (such as short integer, integer, and long integer)
- Domains can also be described by a subrange of values from a data type or as an enumerated data type in which all possible values are explicitly listed.


Connect
Sumit (SQL Server 11.0.2100 - SUMIT\Su
Databases
System Databases
GEEKS
Database Diagrams
Tables
System Tables
FileTables
dbo.GEEK
Views
Synonyms
Programmability
Service Broker
Storage
Security
CREATE TABLE GEEK
(
[ID] INT,
[ADDRESS] VARCHAR(50)
CONSTRAINT [ADDRESS] CHECK(ADDRESS NOT LIKE '%[^A-Z]%')
)
INSERT INTO GEEK([ID],[ADDRESS])
VALUES(3,'GEEKS')
INSERT INTO GEEK([ID],[ADDRESS])
VALUES(2,'GEEKS1')
133 %

Relational Model Constraints: Key

Key Constraints (within a single relation and its attributes)

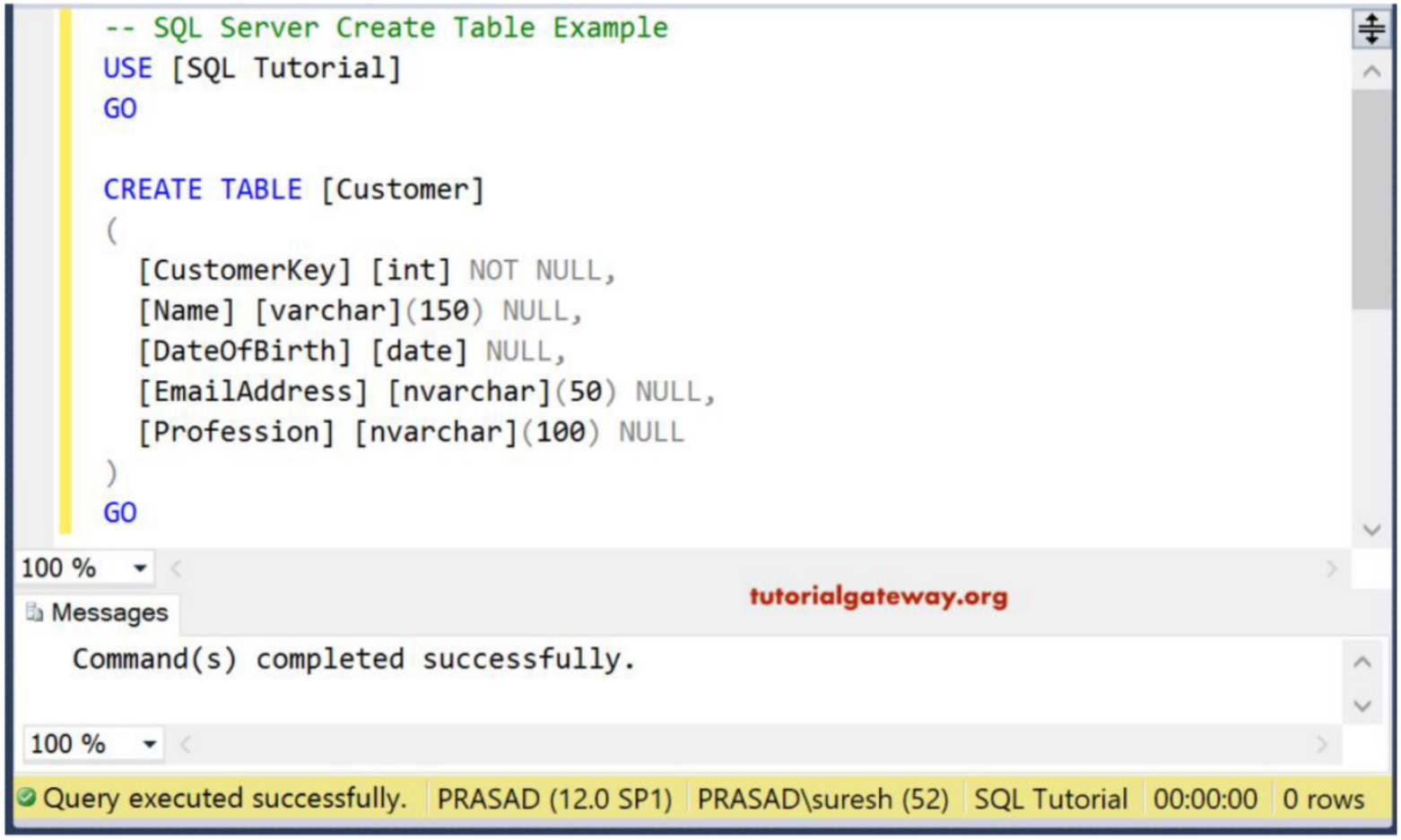
- t1[SK] ≠ t2[SK], then for any two distinct tuples t1 and t2 in a relation state r of R
- In general, a relation schema may have more than one key. In this case, each of the keys is called a candidate key.
- It is common to designate one of the candidate keys as the primary key of the relation. This is the candidate key whose values are used to identify tuples in the relation.


macos — psql -U postgres -d test_db — 80×25
~ $ psql -U postgres -d test_db
psql (12.2 (Homebrew petere/postgresql))
Type "help" for help.
test_db=# CREATE TABLE alphabet (
test_db(#   id SERIAL PRIMARY KEY,
test_db(#   letter VARCHAR NOT NULL
test_db(#);
CREATE TABLE
test_db=# INSERT INTO alphabet (letter)
test_db-# VALUES ('A');
INSERT 0 1
test_db=# SELECT * FROM alphabet;
 id | letter
----+--------
  1 | A
(1 row)
test_db=#

Relational Model Constraints: NULL

NULL Values Constraints (within a single relation and its attributes)

- Whether NULL (empty) values are or are not permitted.
- For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.


-- SQL Server Create Table Example
USE [SQL Tutorial]
GO
CREATE TABLE [Customer]
(
[CustomerKey] [int] NOT NULL,
[Name] [varchar](150) NULL,
[DateOfBirth] [date] NULL,
[EmailAddress] [nvarchar](50) NULL,
[Profession] [nvarchar](100) NULL
)
GO
100 %
tutorialgateway.org
Messages
Command(s) completed successfully.
100 %
Query executed successfully.
PRASAD (12.0 SP1)
PRASAD\suresh (52)
SQL Tutorial
00:00:00
0 rows

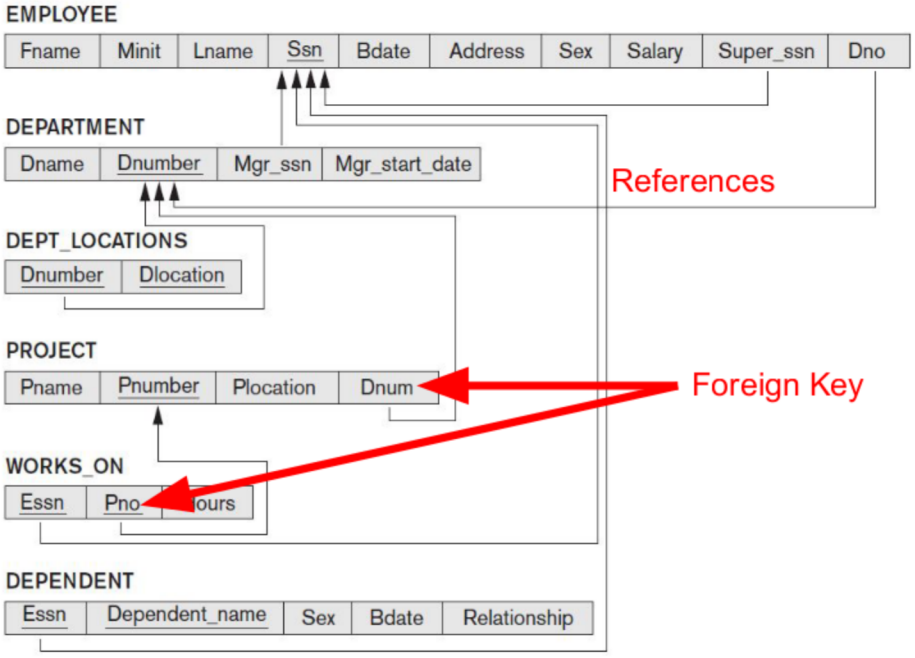
Relational Model Constraints: Entity Integrity

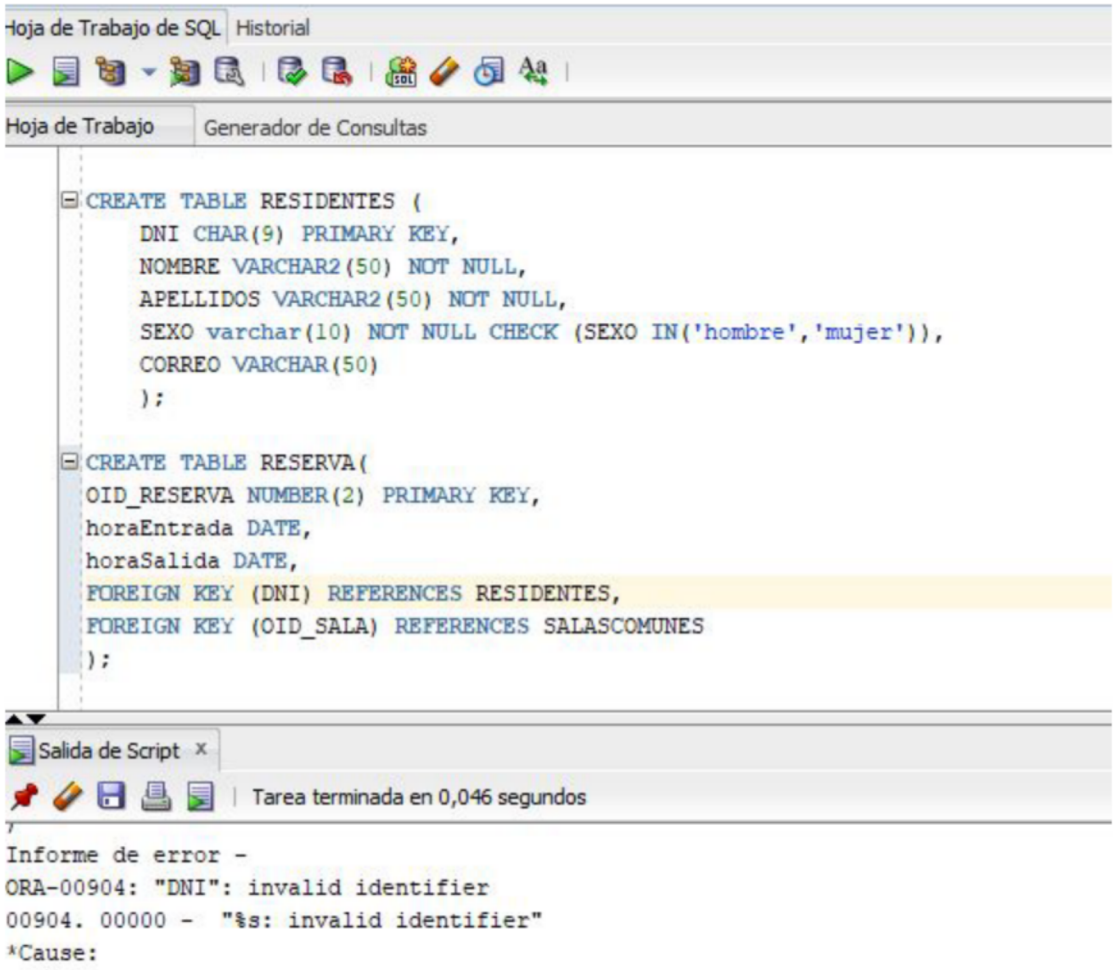
- No primary key value can be NULL
- This is because the primary key value is used to identify individual tuples in a relation
- Having NULL values for the primary key implies that we cannot identify some tuples

Relational Model Constraints: Referential Integrity Constraint

- It is specified between two relations to maintain the consistency among tuples in the two relations.
- A tuple in one relation A that refers to another relation B must refer to an existing tuple in B.

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.




Hoja de Trabajo de SQL
Historial
Hoja de Trabajo
Generador de Consultas
CREATE TABLE RESIDENTES (
DNI CHAR(9) PRIMARY KEY,
NOMBRE VARCHAR2(50) NOT NULL,
APELLIDOS VARCHAR2(50) NOT NULL,
SEXO varchar(10) NOT NULL CHECK (SEXO IN('hombre','mujer')),
CORREO VARCHAR(50)
);
CREATE TABLE RESERVA(
OID_RESERVA NUMBER(2) PRIMARY KEY,
horaEntrada DATE,
horaSalida DATE,
FOREIGN KEY (DNI) REFERENCES RESIDENTES,
FOREIGN KEY (OID_SALA) REFERENCES SALASCOMUNES
);
Salida de Script
Tarea terminada en 0,046 segundos
Informe de error -
ORA-00904: "DNI": invalid identifier
00904. 00000 - "%s: invalid identifier"
*Cause:

Relational Model Constraints: others

- *Example: the salary of an employee should not exceed the salary of the employee's supervisor and the maximum number of hours an employee can work on all projects per week is 56*
- Such constraints can be specified and enforced within the application programs that update the database, or by using a general-purpose constraint specification language
- Mechanisms called triggers and assertions can be used in SQL, through the CREATE ASSERTION and CREATE TRIGGER statements

```sql
-- Example for INSTEAD OF INSERT Triggers in SQL Server
USE [SQL Tutorial]
GO
CREATE TRIGGER InsteadOfINSERTTriggerExample on [EmployeeTable]
INSTEAD OF INSERT
AS
BEGIN
    INSERT INTO [EmployeeTableAudit](
                [ID],[Name],[Education],[Occupation]
               ,[YearlyIncome],[Sales],[ServerName],[ServerInstanceName],[Insert Time])
    SELECT      ID, Name, Education, Occupation,
                YearlyIncome, Sales,
                CAST( SERVERPROPERTY('MachineName') AS VARCHAR(50)),
                CAST( SERVERPROPERTY('ServerName') AS VARCHAR(50)),
                GETDATE()
    FROM INSERTED
    WHERE YearlyIncome <= 70000;
    PRINT 'We Successfully Fired Our INSTEAD OF INSERT Triggers in SQL Server.';

    INSERT INTO [EmployeeTable](
            [Name], [Education], [Occupation], [YearlyIncome], [Sales])
    SELECT  Name, Education, Occupation, YearlyIncome, Sales
    FROM INSERTED
    WHERE YearlyIncome > 70000;
END
GO
```

100 %

Messages

```
Command(s) completed successfully.
```

Functional Dependencies

Functional dependencies

- Formal tool for analysis of relational schemas
- The single most important concept in relational schema design theory
- Constraint between two sets of attributes from the database

Let us think of the whole database as being described by a single universal relation schema R = {A1, A2, ... , An}

Definition. A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

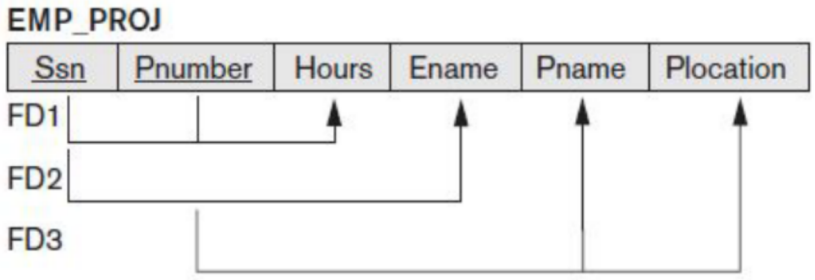
Functional dependencies

Definition. A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

- The values of the X component of a tuple uniquely (or functionally) determine the values of the Y component
- We also say that there is a functional dependency from X to Y, or that Y is functionally dependent (FD) on X
- X□Y in R

Functional dependencies: example

- Consider the relation schema EMP_PROJ
- From the semantics of the attributes and the relation, we know that the following functional dependencies should hold:
- Ssn ➔ Ename
- Pnumber ➔ {Pname, Plocation}
- {Ssn, Pnumber} ➔ Hours



Testing and improving relation schemas using normalization

Normalization of Relations

- The normalization process, as first proposed by Codd (1972), takes a relation schema through a series of tests to certify whether it satisfies a certain normal form (1NF, 2NF, 3NF, BCNF, 4NF, 5NF)
- Normalization of data can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) **minimizing redundancy** and (2) **minimizing the insertion, deletion, and update anomalies**

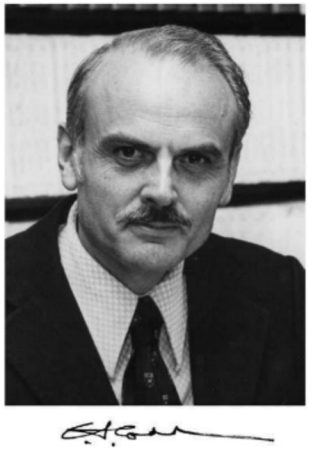
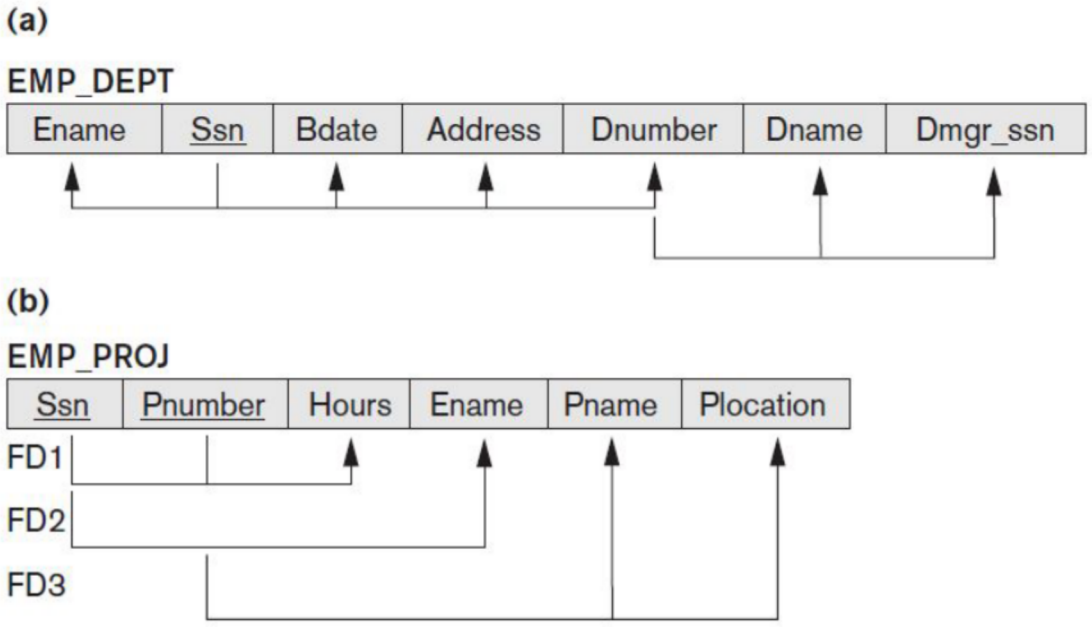


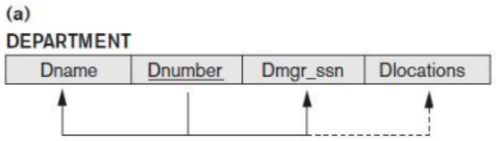
Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



First normal form (1NF)

First normal form (1NF)

- It disallows multivalued and/or composite attributes
- The domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute
- The only attribute values permitted by 1NF are single atomic (or indivisible) values



(b)
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Achieving first normal form (1NF)

Option 1

- Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT.
- This decomposes the non-1NF relation into two 1NF relations.

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Achieving first normal form (1NF)

Option 2

- Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT
- The primary key becomes the combination {Dnumber, Dlocation}
- This solution has the disadvantage of introducing redundancy in the relation and hence is rarely adopted

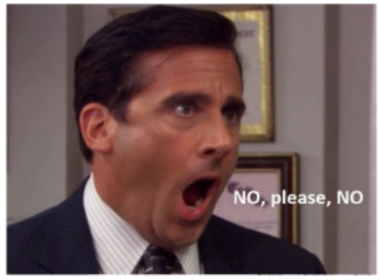
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Achieving first normal form (1NF)

Option 3

- If a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes: Dlocation1, Dlocation2, and Dlocation3



Second normal form (2NF)

Second normal form (2NF)

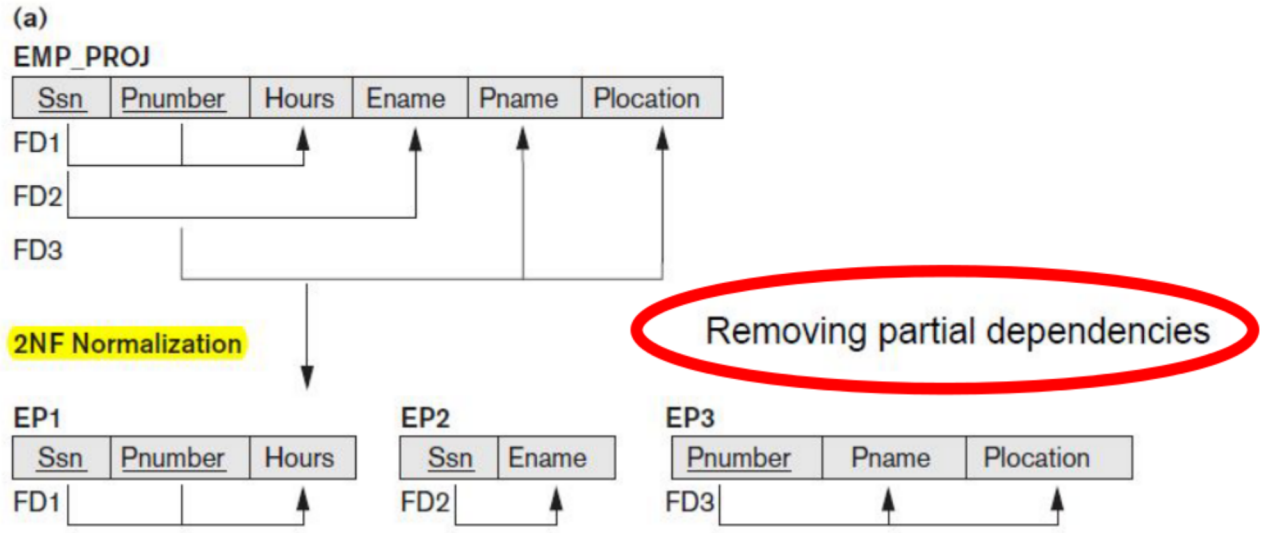
- It is based on the concept of full functional dependency
- A functional dependency X → Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold anymore
- In other words, (X − {A}) does not functionally determine Y
- A functional dependency X → Y is a partial dependency if some attribute A ε X can be removed from X and the dependency still holds
- In other words, (X − {A}) → Y

Second normal form (2NF)

- The test for 2NF involves testing the left-hand side of functional dependencies
- If the primary key contains a single attribute, the test need not be applied at all

An attribute is called *nonprime* if it is **not** a member of any candidate key.

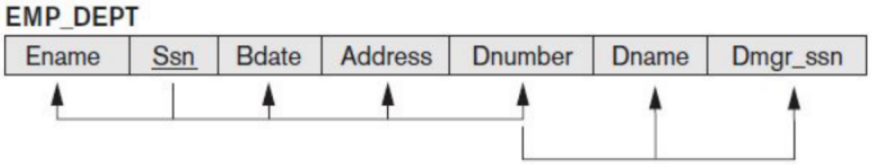
Definition. A relation schema R is in 2NF if every nonprime attribute A in R is *fully functionally dependent* on the primary key of R.


(a)
EMP_PROJ
Ssn
Pnumber
Hours
Ename
Pname
Plocation
FD1
FD2
FD3
2NF Normalization
Removing partial dependencies
EP1
Ssn
Pnumber
Hours
FD1
EP2
Ssn
Ename
FD2
EP3
Pnumber
Pname
Plocation
FD3

Third normal form (3NF)

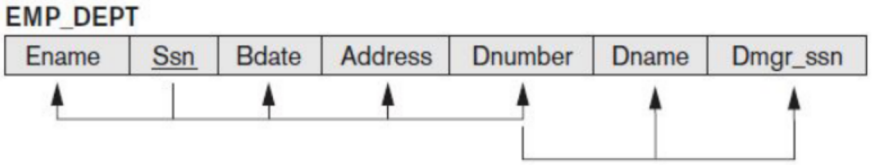
Second normal form (3NF)

- Third normal form (3NF) is based on the concept of transitive dependency
- A functional dependency X → Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold



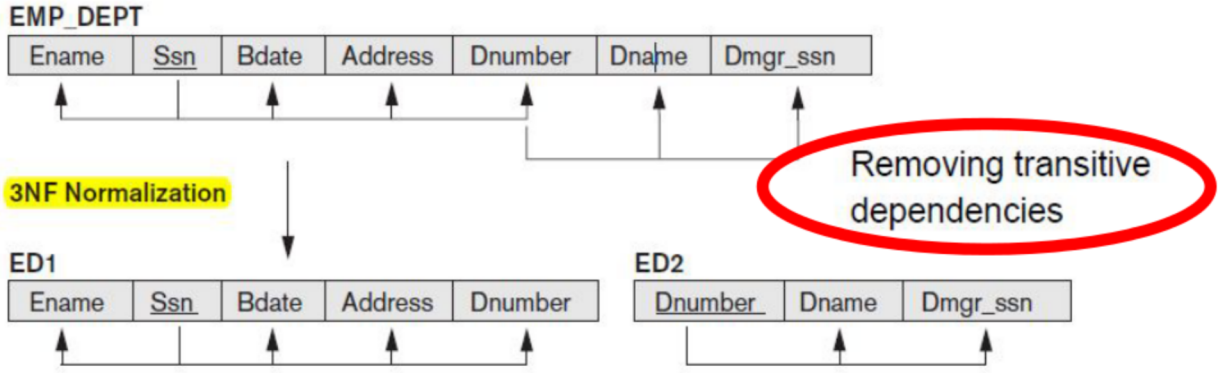
Second normal form (3NF)

- Third normal form (3NF) is based on the concept of transitive dependency
- A functional dependency X → Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold



Second normal form (3NF)

Definition. According to Codd's original definition, a relation schema R is in **3NF** if it satisfies 2NF *and* no nonprime attribute of R is transitively dependent on the primary key.



Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Comments on normalization

- In practice, many databases are de-normalized to greater or lesser degree
- The reason most often stated has to do with performance - a de-normalized database may require fewer joins and can, therefore, be faster for retrievals
- Finally, it should be noted that dealing with many-to-many relationships raises some issues that cannot be fully resolved through normalization
- https://www.ibm.com/developerworks/library/wa-dbdsgn2/index.html#resourcelist

Thank you