WeCloudData

Databases and Data Warehouse

Data Engineering Diploma

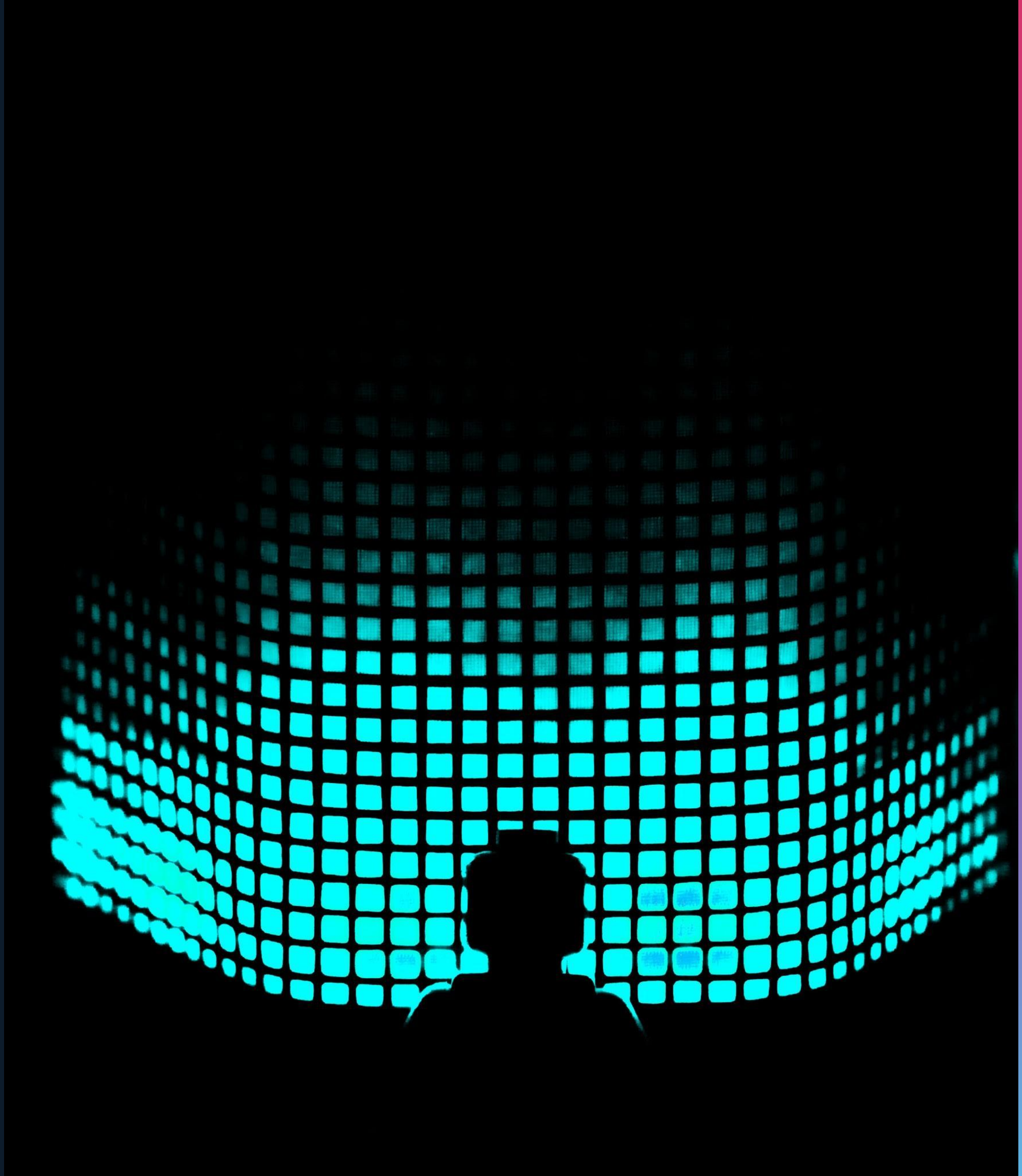
Module 1 :

Database Introduction

Learning Objectives

In this module, we will introduce:

- What is a database and database management systems?
- How the databases evolved over time?
- A 3-step database design process



Database and Database Management System

Database Introduction

- Database

  A database is an organized collection of structured information or data, typically stored and accessed electronically from a computer system

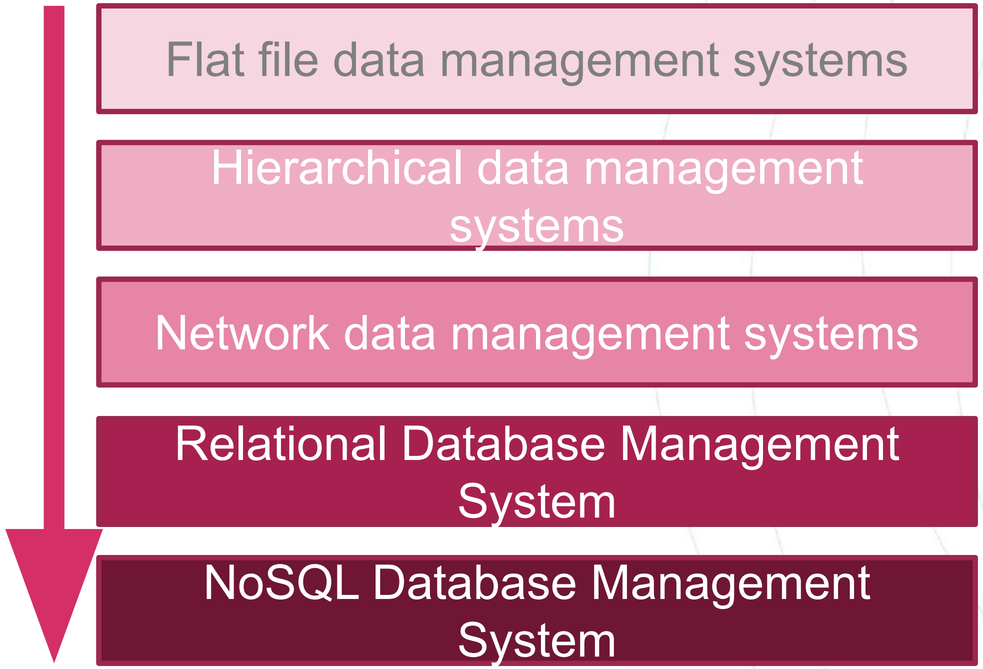
- Database Management System

  DBMS is an interface for end users to work with the database. DBMS allows users to access, update, manage the information in the database. It can also help with administration actions such as performance tuning, monitoring, backups, snapshots, and recovery.

Early Database Management Systems

Database Introduction

- Database systems prior to relational databases:
  - Flat file data management systems
  - Hierarchical data management systems
  - Network data management systems
- The disadvantages of early database management systems include duplicate data, difficulty implementing security, inefficient searching, and difficulty maintaining program code to access databases.
- Also, there was no independence between logical organization of database.
  - As a result, the programs using these had to change each time the structure of the database changes.


Flat file data management systems
Hierarchical data management systems
Network data management systems
Relational Database Management System
NoSQL Database Management System

Early Database Management Systems

Database Introduction

- Relational databases provided a lot of improvements over the previous DBMS.
  - Relational databases separated the logical organization of data structures from the physical storage of those structures.
- Relational databases were based on a formal mathematical model that used relational algebra to describe data and their relations.
  - Codd and others developed rules for designing relational databases that eliminated the potential for some types of data anomalies, such as inconsistent data.
- The exponential growth of e-commerce and social media led to the need for data management systems that were scalable, low cost, flexible, and highly available.
  - Achieving some of these objectives with relational databases is possible in some cases, but often with difficulty and potentially high costs.
- NoSQL databases were created to address the limitations of relational database management systems.

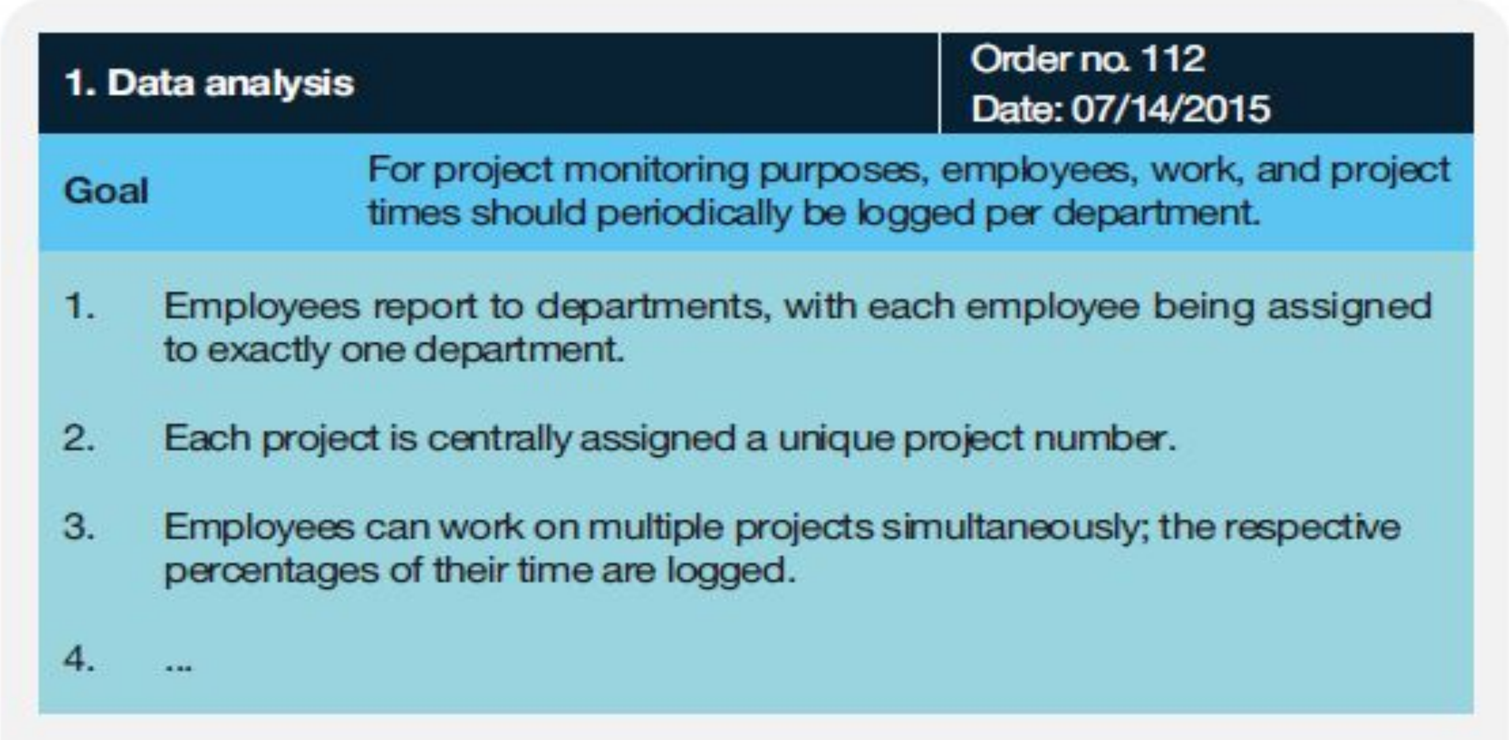
Steps in designing database

Database Introduction

- A simple 3 steps process.
  - Understand the business requirements (iterative process)
  - Designing a conceptual data model (entity-relationship model)
  - Converting it to relational or non-relational database

Steps in designing database: Step 1

Database Introduction



1. Data analysis	Order no. 112 Date: 07/14/2015
Goal	For project monitoring purposes, employees, work, and project times should periodically be logged per department.

1. Employees report to departments, with each employee being assigned to exactly one department.

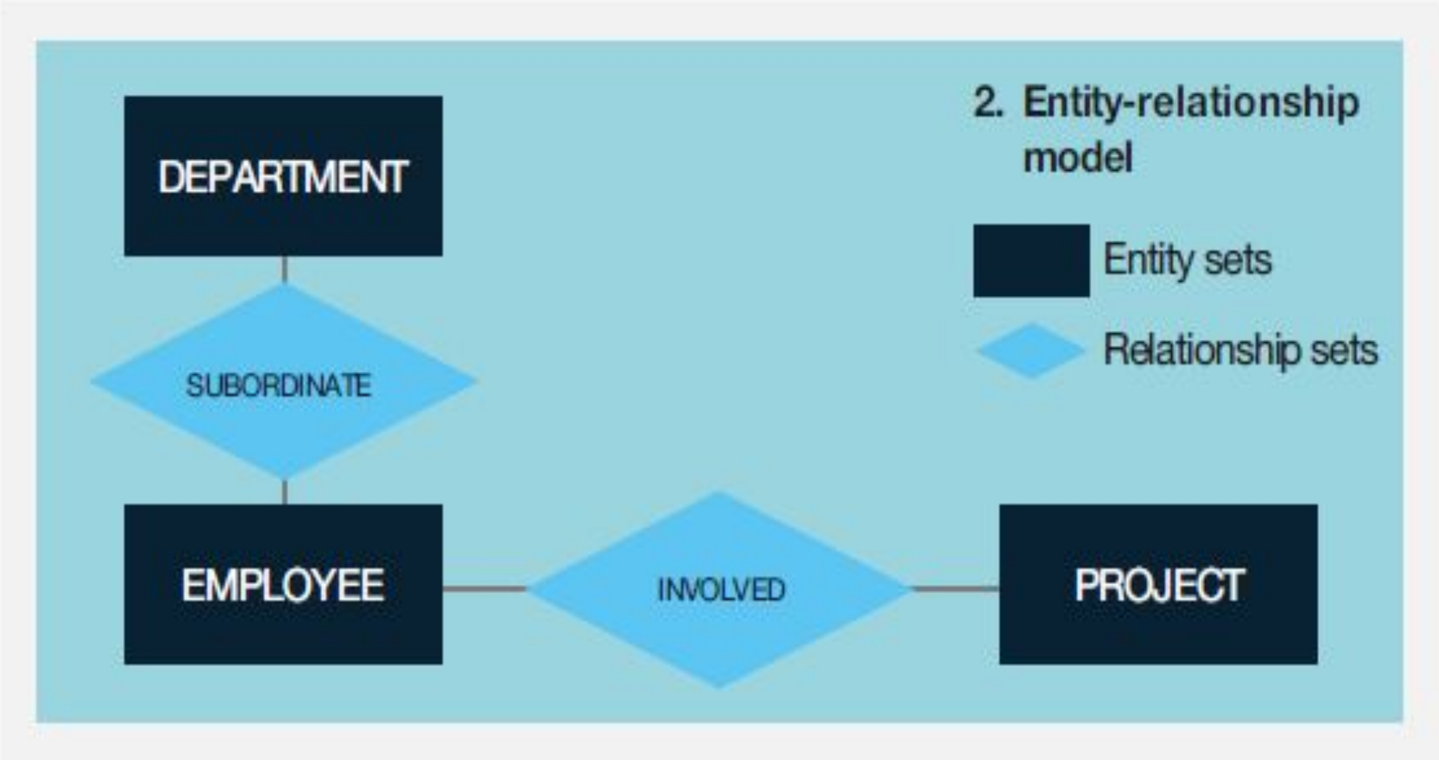
2. Each project is centrally assigned a unique project number.

3. Employees can work on multiple projects simultaneously; the respective percentages of their time are logged.

4. ...

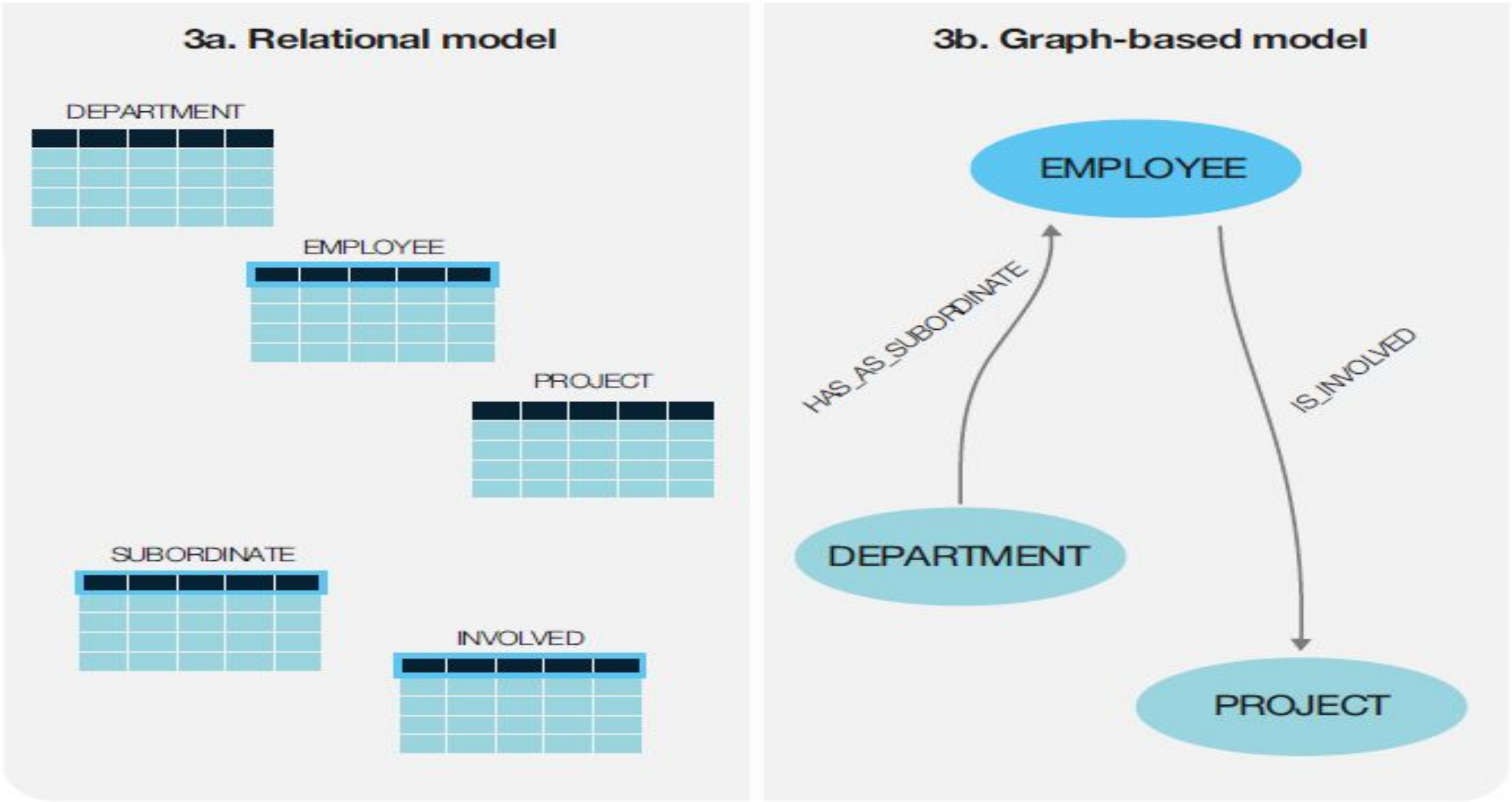
Steps in designing database: Step 2

Database Introduction



Steps in designing database: Step 3

Database Introduction



Data Models

Database Introduction

Let’s think about data models

- Data models are critical to how you write software. These models can shape the way you’re thinking about solving the problem at hand.

- Different ways to express the data models when working as a developer, mapping your API/data structures to real world use case.
  - JSONs
  - XMLs
  - Tables
  - Flat files

- Relational vs Document model
  - RDBMS - regular structures, tables, relationships
  - Document - example of NoSQL (Not only SQL)

Data Models

Database Introduction

- Does data modeling happen before you create a database, or is it an iterative process?
  - It's an iterative process.
  - Data engineers continually reorganize, restructure, and optimize data models to fit the needs of the organization.

- Who does data modeling?
  - Everyone who deals with data.
  - If you are working with data, you need to know the fundamentals of data modeling

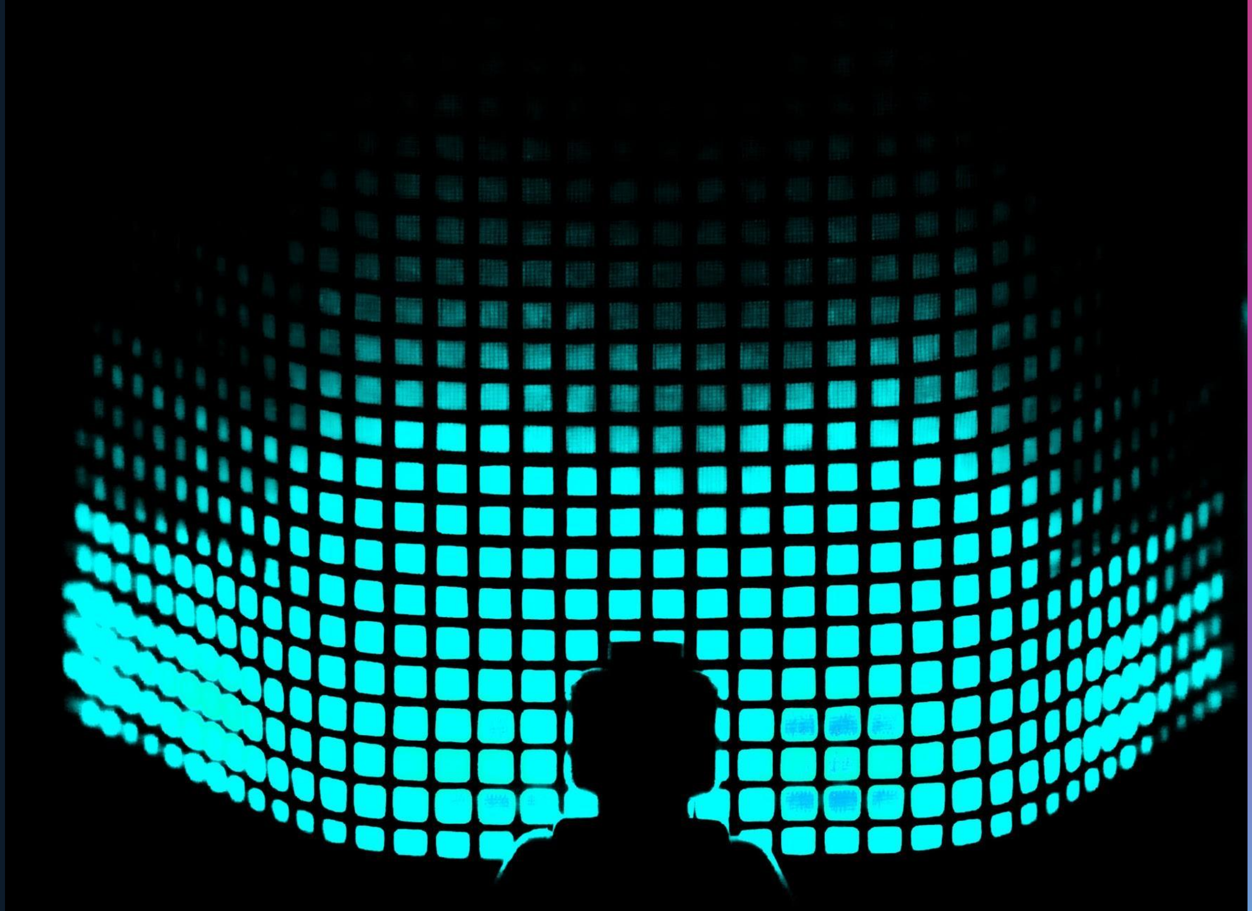
Module 2 :

Relational Databases

Learning Objectives

In this module, we will introduce:

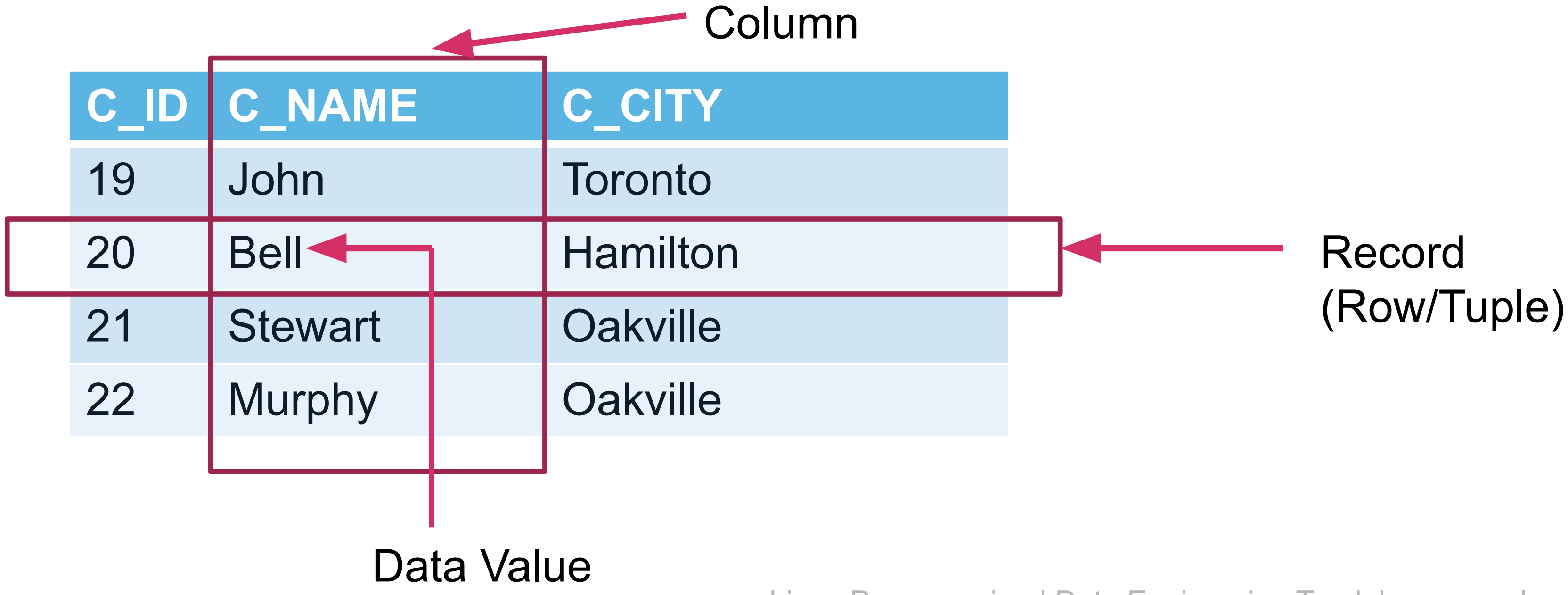
- Relational Databases and Relational Database Management Systems.
- Important Database terminologies and concepts
- Normalization
- Benefits and Limitations of relational databases



Relational Database

Relational Databases

- A relational database is a collection of tables of data created and organized as per the relational model.
- Data is stored as collection of tables where each table has a unique name and **PRIMARY KEY** to uniquely identify each row.



Column

C_ID	C_NAME	C_CITY
19	John	Toronto
20	Bell	Hamilton
21	Stewart	Oakville
22	Murphy	Oakville

Record
(Row/Tuple)

Data Value

Relational Model: Table

Relational Databases

- Thus, in relational model the term **relation** is used to refer to a table.
- A table or relation is a set of tuples presented in tabular form and meeting the following requirements:
  - Table name: A table has a unique table name.
  - Attribute name: All attribute names are unique within a table and label one specific column with the required property.
  - No column order: The number of attributes is not set, and the order of the columns within the table does not matter.
  - No row order: The number of tuples is not set, and the order of the rows within the table does not matter.
  - Primary key: One attribute or a combination of attributes uniquely identifies the tuples within the table and is declared the identification key.
- **The relational model represents both data and relationships between data as tables.**

Transaction

Relational Databases

A transaction is a group of commands, to be executed on database, treated as a single unit.

- A successful transaction must pass “ACID test”, i.e. it must be:
  - Atomic
  - Consistent
  - Isolated
  - Durable

ACID Transactions

Relational Databases

- Atomicity:
  - The transaction is executed in its entirety or not executed at all
  - Users should not have to worry about the incomplete transactions, e.g, when system crashes

```
USE [SQL TEST]
 GO

BEGIN TRANSACTION
        UPDATE [DimProduct]
                SET [StockLevel] = 4700
                WHERE [ProductKey] = 213
        INSERT INTO [Sales] ([ProductKey],[OrderQuantity],[UnitPrice],[SalesAmount])
        VALUES (213, 300, 48.0673, ‘This is wrong’)
COMMIT TRANSACTION
```

ACID Transactions

Relational Databases

- Consistency:
  - Only transactions that abide by constraints and rules are written into the database, otherwise the database keeps the previous state.
- Isolation:
  - Every transaction is processed in isolation and are independent of other transactions.
  - If two transactions are executing concurrently, each one will see the world as if they were executing sequentially, and if one needs to read data that is written by another, it will have to wait until the other is finished.
  - Most databases use locking mechanisms to implement isolation
- Durability
  - Once the transaction has been successfully completed, its effects should persist even if the system crashes before all its changes are reflected on the disk

ACID Transactions

Relational Databases

- Isolation: Example

Session 1

```
USE [SQL TEST]
 GO

BEGIN TRANSACTION
        UPDATE [DimProduct]
                SET [StockLevel] = 5700
                WHERE [ProductKey] = 213
```

Session 2

```
USE [SQL TEST]
 GO

SELECT
[ProductKey],[ProductName],[Color],[StandardCost],
[ListPrice], [DealerPrice], [StockLevel]

FROM [dimProduct]
```

This will not show results as first transaction is not completed

Important Database Terminologies

Relational Databases

- Database Schema:
  - A representation of the structure of a database. (tables, relationships, constraints)
  - It may be graphical or textual.
  - Graphical representations typically involve the use of boxes that represent database tables and arrows that represent inter-table relationships.
  - Textual schema representations utilize Database Definition Language (DDL) statements to describe a database design
- Database instance: An instance is a combination of the software and memory used to access that data
- Relations: In relational databases, table is often referred to as “relation”.

Important Database Terminologies

Relational Databases

- Tuple: A row in a table is known as a record, or a tuple.
- Attribute: An individual piece of data in a record is known as a field, or attribute.
- Domain: All the possible allowable values for an attribute
- Cardinality: How unique an attribute is in terms of its data values.
  - High Cardinality: Large percentage of unique values
  - Low Cardinality: Lot of repeated values across tuples
- Join:
  - An operation in which the rows of one table are related to the rows of another through common column values.
- Commit:
  - The action that causes the all of the changes made by a particular transaction to be reliably written to the database files and made visible to other users.

Important Database Terminologies

Relational Databases

- Metadata: "Data about data."
  - In a DBMS context, data stored in columns of a table have certain attributes, such as the type, length, description or other characteristics that allow the DBMS to process the data meaningfully or allow the users to understand it better.
- Keys:
  - Attributes/set of attributes used to establish relationship between tables.
  - It helps to identify a row(tuple) in a relation(table).
  - Help you to enforce identity and integrity in the relationship.

Primary Key

C_ID	C_NAME	C_CITY
19	John	Toronto
20	Bell	Hamilton
21	Stewart	Oakville
22	Murphy	Oakville

Different types of Keys

Relational Databases

- Primary Key - is a column or group of columns in a table that uniquely identify every row in that table.
- Candidate Key - is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- Alternate Key - is a column or group of columns in a table that uniquely identify every row in that table.
- Foreign Key - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.

Different types of Keys

Relational Databases

- Compound Key - A key that consists of 2 or more attributes that uniquely identify an entity occurrence. Each attribute that makes up the compound key is a simple key in its own right.
- Composite Key - Has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- Surrogate Key - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

Primary Key :

1. There can only be one primary key for a table.
2. The primary key consists of one or more columns.
3. The primary key enforces the entity integrity of the table.
4. All columns defined must be defined as NOT NULL.
5. The primary key uniquely identifies a row.

Keys: Example

Relational Databases

C_ID	C_CITY_ID	C_FIRST_NAME	C_LAST_NAME	C_MOBILE	C_EMAIL
19	3	Susan	Williams	613-555-0170	Susan_williams@yahoo.com
20	4	Bryan	Johnson	613-555-0180	bryanw@live.com
21	5	Stephen	Martin	613-555-0131	yxing@outlook.com
22	3	Barbara	Cooper	613-555-0152	bcooper@verizon.net
25	7	Frank	Terry	416-555-0108	frank.terry@gmail.com

Primary Key

Foreign Key

Alternate Keys

C_CITY_ID	C_CITY
3	Toronto
4	Hamilton
5	Oakville
6	Mississauga
7	Markham

Primary Key

Keys: Example

Relational Databases

- C_ID and C_MOBILE : can be considered as Superkeys
  - Super key can have extra attributes that are redundant for distinct identification.
- C_ID, C_MOBILE and C_EMAIL : candidate keys
  - A candidate key can have more than one attributes but the values should be distinct
  - A candidate key must uniquely identify each row in table
- In case, we do not have a single column that can uniquely identify rows in table, we can have compound key.

Surrogate Key

C_ID	PRODUCT_ID	ORDER_ID	PRODUCT
20	P11	17823	Adidas Shoes
20	P34	17823	Acer Laptop
27	P12	789	Floor Lamp
25	P3	543	Flower bouquet

- Use Surroga rimary Key
  - They do not provide any relation to the table data and are usually serially ordered integers.

Entity Relationship Diagram (ERD)

Relational Databases

- ERD is a data modeling technique to produce conceptual data model of an information system.
- ERD illustrates logical structure of database
- Major activities while developing ERD are:
  - Identifying entities, attributes and their relationships
  - Entity -> Table
  - Attribute -> Column
  - Relationship -> Connector Line
- Entity:
  - "...anything (people, places, objects, events, etc.) about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.)."
- Attribute: Property of an entity
- Relationship: Associations between entities. (verb connecting two or more entities)
  - Example: Employees are assigned to projects

Entity Relationship Diagram (ERD)

Relational Databases

- Relationships should be classified in terms of cardinality.
  - One-to-one, one-to-many, etc.
- Business Rules are used to define entities, attributes, relationships and constraints.
  - The data can be considered significant only after business rules are defined.
- https://www.lucidchart.com/pages/er-diagrams/#section_3

Relational Database Management Systems (RDBMS)

Relational Databases

- Relational database management systems (RDBMS) are integrated systems for the consistent management of tables/relational databases.
  - They provide service functionalities and the descriptive language SQL for data description, selection, and manipulation.
- RDBMS has 2 major components:
  - Storage Component :
    - Stores both data and the relationships between pieces of information in tables.
    - Also contains the predefined system tables necessary for database operation. These contain descriptive information and can be queried but not manipulated by users.
  - Management Component:
    - Provides relational data definition, selection, and manipulation language SQL.
    - Also contains service functions for data restoration after errors, for data protection, and for backup.

Common types of Relational Databases

Relational Databases

- Oracle
- Teradata
- MySql
- PostgreSQL
- Sqlite

Database design: Structuring your Database

Relational Databases

- Two important concepts in data modeling for relational databases are:
  - Normalization
  - Denormalization
- Normalization: Process of structuring/organizing the data in the database to reduce the data redundancy and increase the data integrity by removing anomalies.
  - Data redundancy: multiple copies of same data
  - Data integrity: the accuracy, completeness, and reliability of data according to domain business rules and references throughout its lifecycle. It is an assurance that you will get the correct answer when you query the database.
- Denormalization: Process of adding redundant data to one or more tables. This can help avoid joins and increase the read performance (may come at the expense of losing write performance)

Normalization

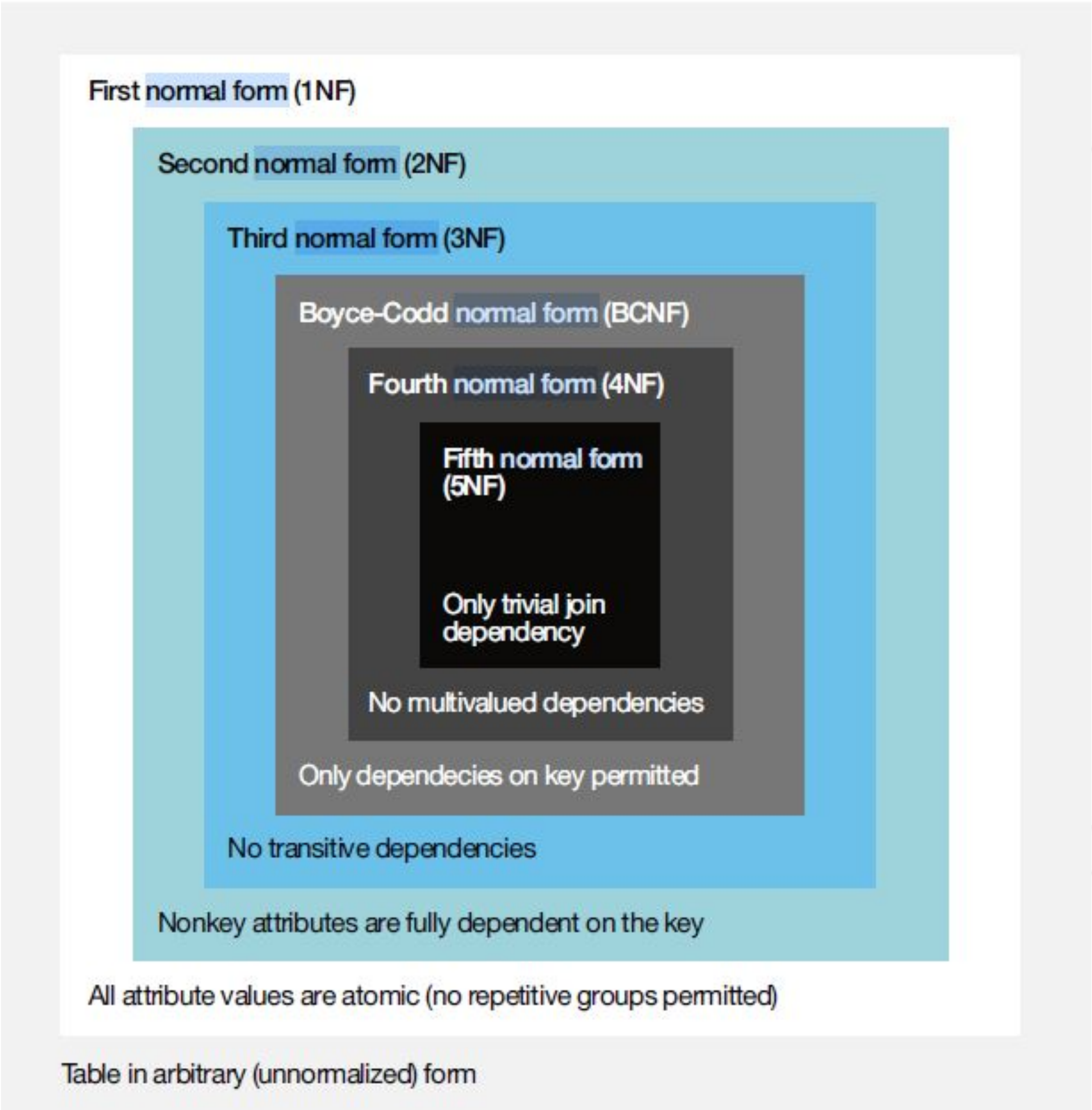
Relational Databases

- When database is not normalized, 3 types of anomalies can occur:
  - Insertion Anomaly : occurs when you are inserting inconsistent information into a table
  - Deletion Anomaly : occurs when you delete a record that may contain attributes that shouldn't be deleted
  - Update Anomaly : changing existing information incorrectly
- Normalization uses a series of normal forms to split the relations (tables) into more structured relations that allows users to perform insert, update and delete operations without introducing any anomalies.
- Most used Normal Forms:
  - First normal form(1NF)
  - Second normal form(2NF)
  - Third normal form(3NF)
  - Boyce & Codd normal form (BCNF)

Normalization

Relational Databases

- The normal forms progressively limit acceptable tables.
- Usually only the first three normal forms are used.
  - since multivalued and join dependencies rarely occur in practice and corresponding use cases are hard to come by

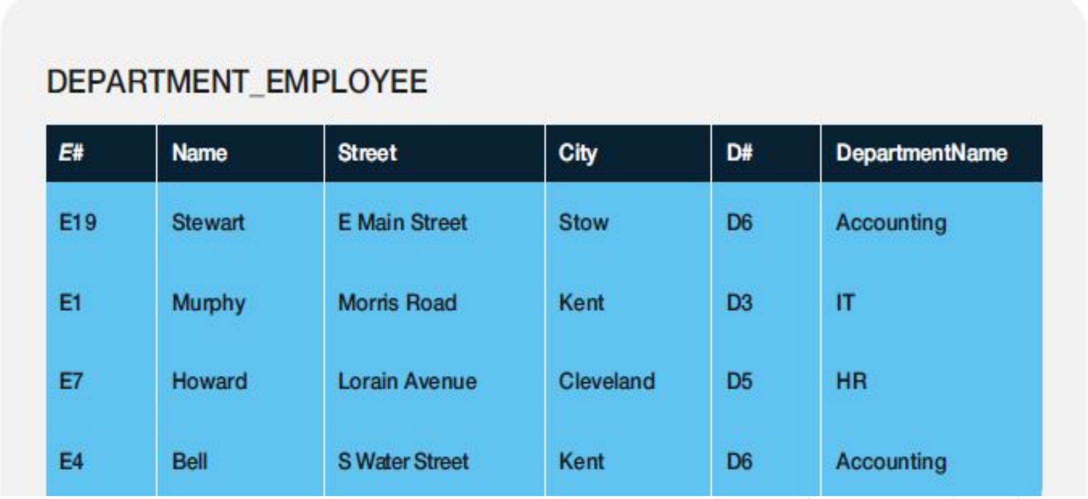


First normal form (1NF)
Second normal form (2NF)
Third normal form (3NF)
Boyce-Codd normal form (BCNF)
Fourth normal form (4NF)
Fifth normal form (5NF)
Only trivial join dependency
No multivalued dependencies
Only dependecies on key permitted
No transitive dependencies
Nonkey attributes are fully dependent on the key
All attribute values are atomic (no repetitive groups permitted)
Table in arbitrary (unnormalized) form

Dependencies and Normal Forms

Relational Databases

- Normal Forms: used to discover and eliminate dependencies within tables in order to eliminate/avoid redundant information and the resulting anomalies.
- An attribute in a table is redundant if individual values of this attribute can be omitted without a loss of information.
  - Example: Here, in **DEPARTMENT_EMPLOYEE** table, attribute **DepartmentName** is redundant since the same value is listed in the table multiple times.



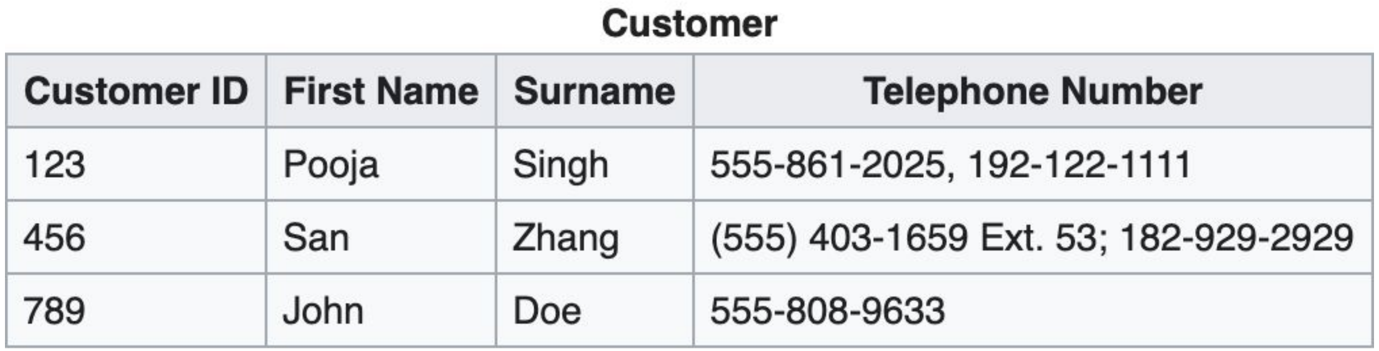
DEPARTMENT_EMPLOYEE

E#	Name	Street	City	D#	DepartmentName
E19	Stewart	E Main Street	Stow	D6	Accounting
E1	Murphy	Morris Road	Kent	D3	IT
E7	Howard	Lorain Avenue	Cleveland	D5	HR
E4	Bell	S Water Street	Kent	D6	Accounting

First Normal Form (1NF)

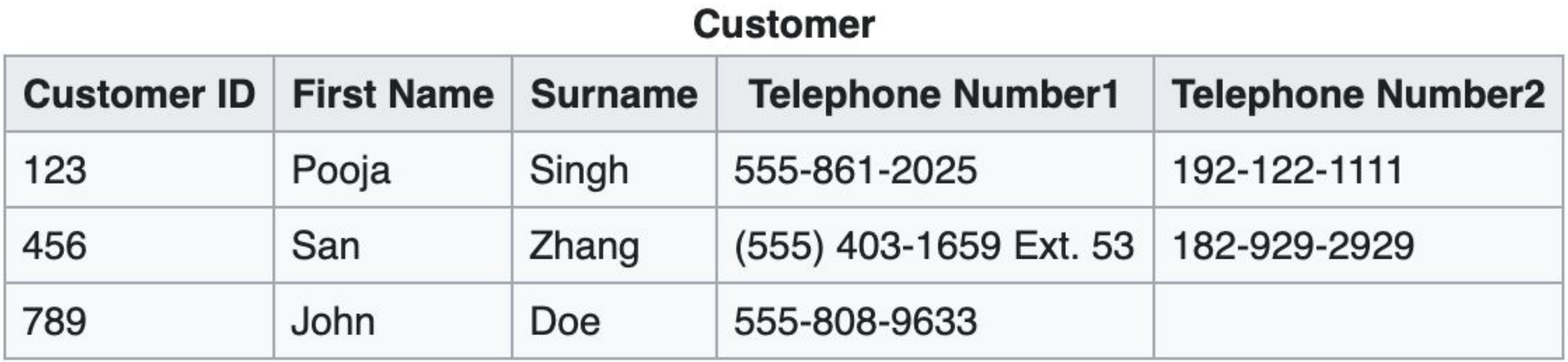
Relational Databases

- A table is in the first normal form when the domains of the attributes are atomic.
- The first normal form requires that each attribute get its values from an unstructured domain, and there must be no sets, lists, or repetitive groups within the individual attributes.



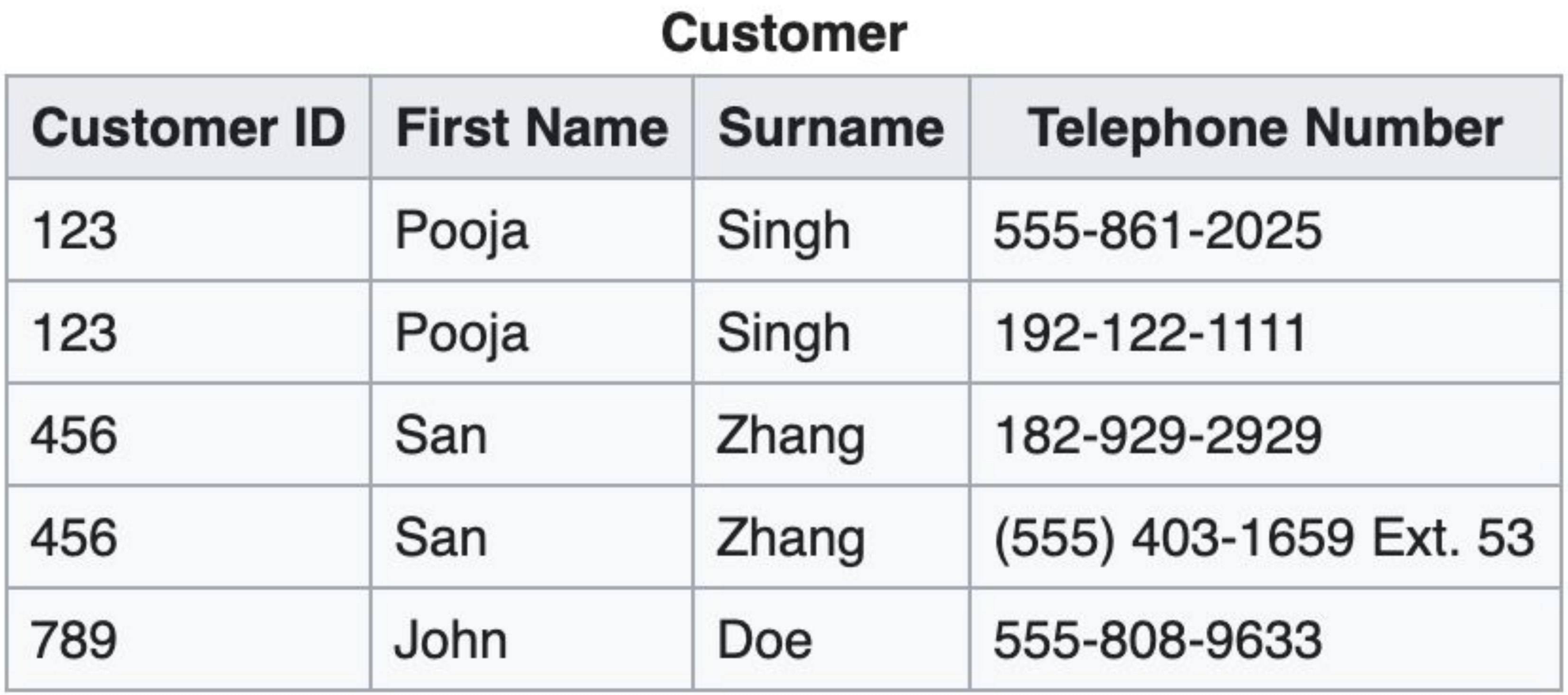
Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633



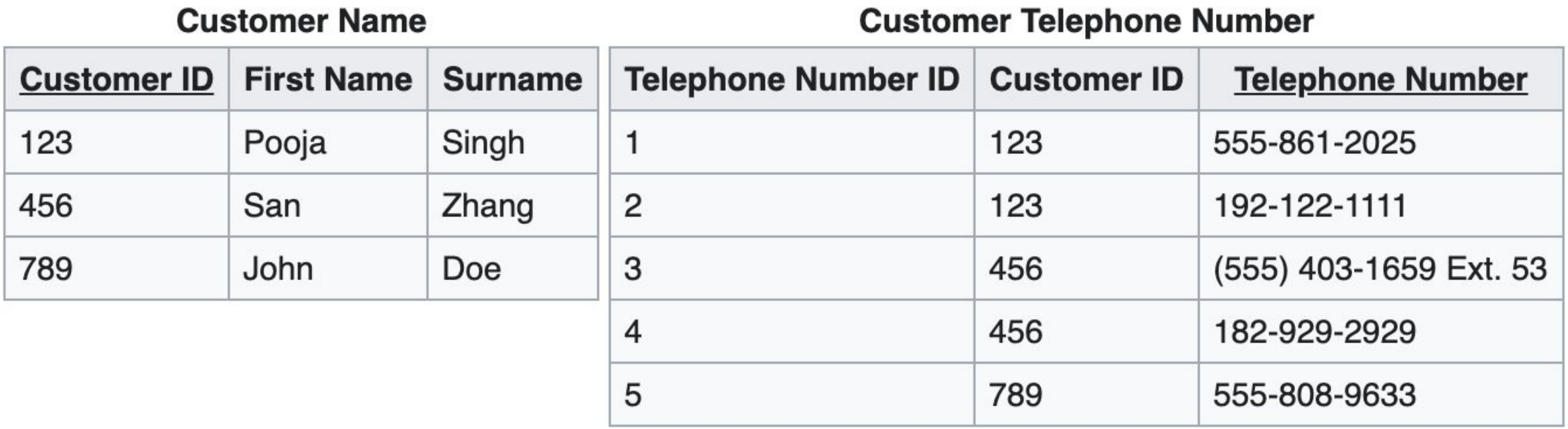
Customer

Customer ID	First Name	Surname	Telephone Number1	Telephone Number2
123	Pooja	Singh	555-861-2025	192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53	182-929-2929
789	John	Doe	555-808-9633	



Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025
123	Pooja	Singh	192-122-1111
456	San	Zhang	182-929-2929
456	San	Zhang	(555) 403-1659 Ext. 53
789	John	Doe	555-808-9633



Customer Name

Customer ID	First Name	Surname
123	Pooja	Singh
456	San	Zhang
789	John	Doe

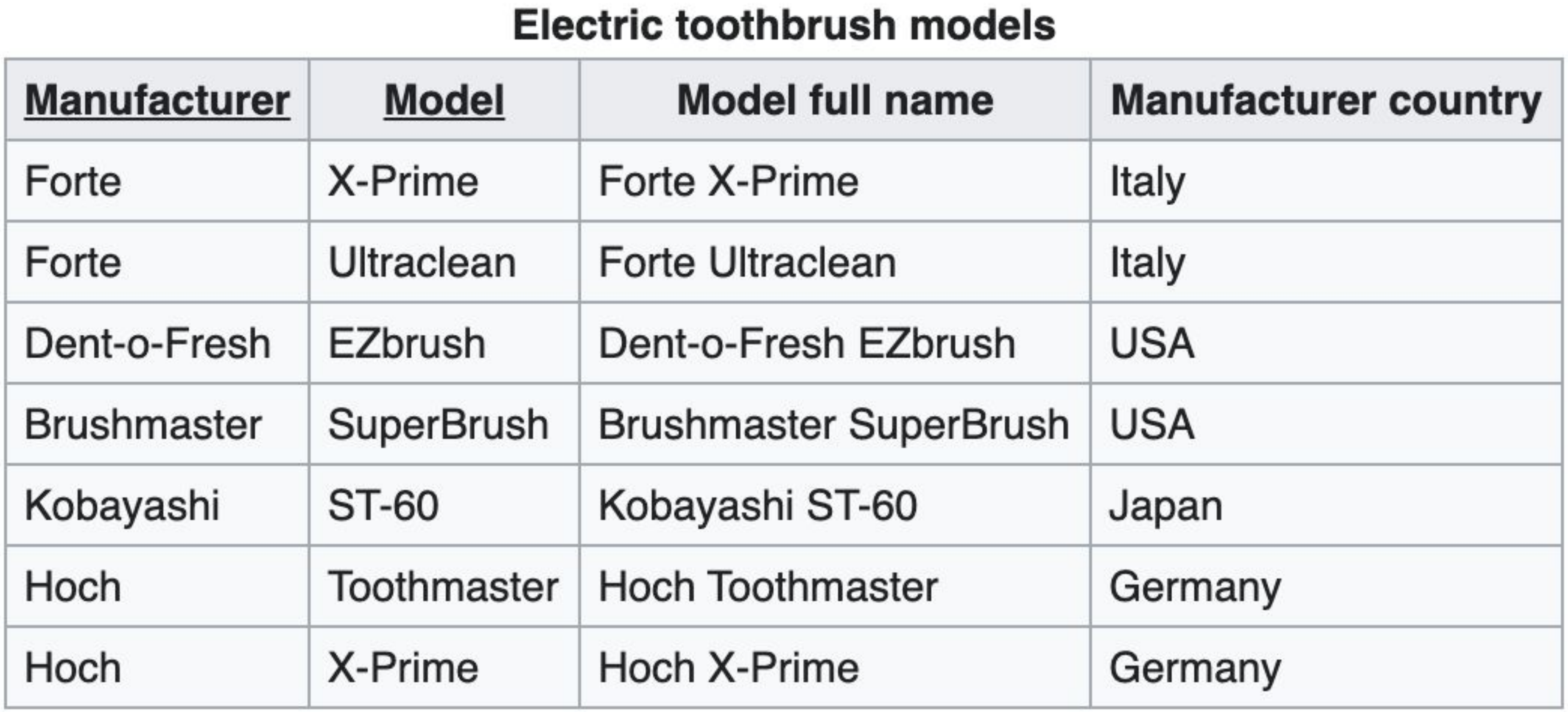
Customer Telephone Number

Telephone Number ID	Customer ID	Telephone Number
1	123	555-861-2025
2	123	192-122-1111
3	456	(555) 403-1659 Ext. 53
4	456	182-929-2929
5	789	555-808-9633

Second Normal Form (2NF)

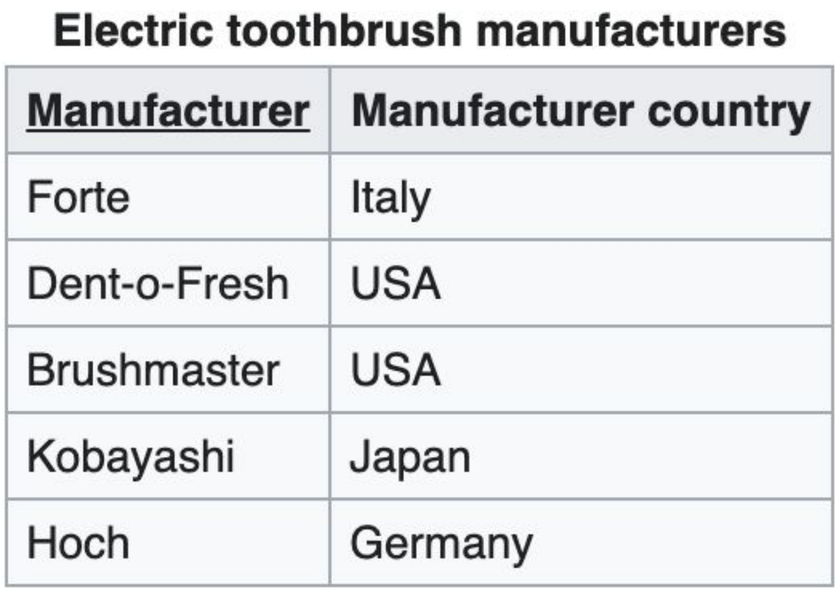
Relational Databases

- A table is in the second normal form when, in addition to the requirements of the first normal form, each non-key attribute is fully functionally dependent on each key.
- An attribute B is functionally dependent on an attribute A if for each value of A, there is exactly one value of B (written as A -> B). A functional dependency of B on A, therefore, requires that each value of A uniquely identifies one value of B



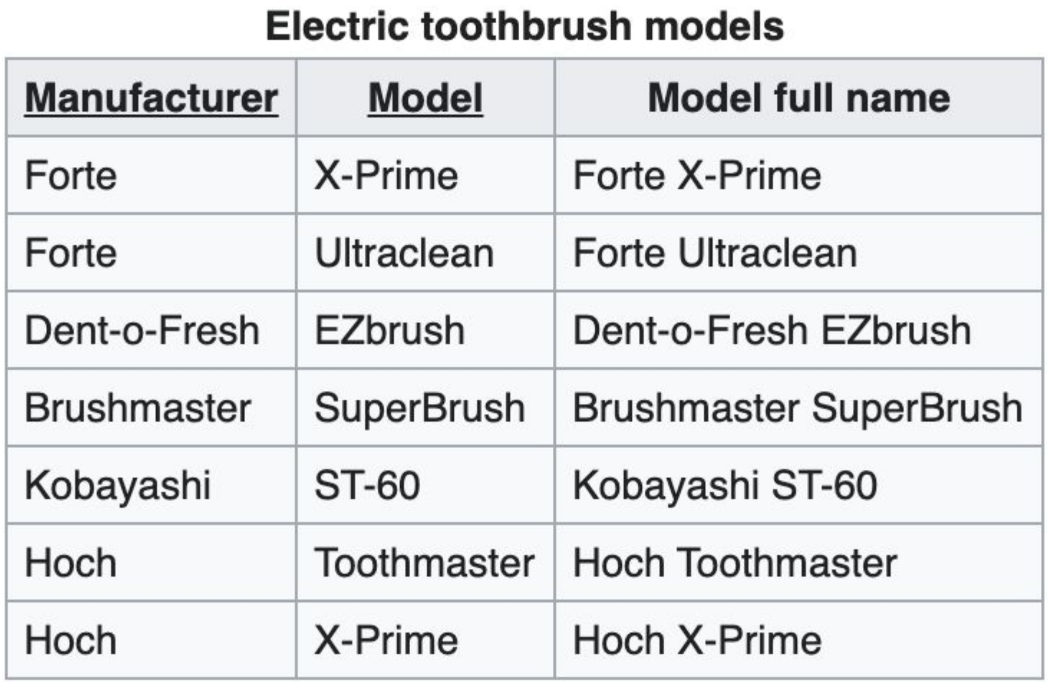
Electric toothbrush models

Manufacturer	Model	Model full name	Manufacturer country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Brushmaster	SuperBrush	Brushmaster SuperBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany



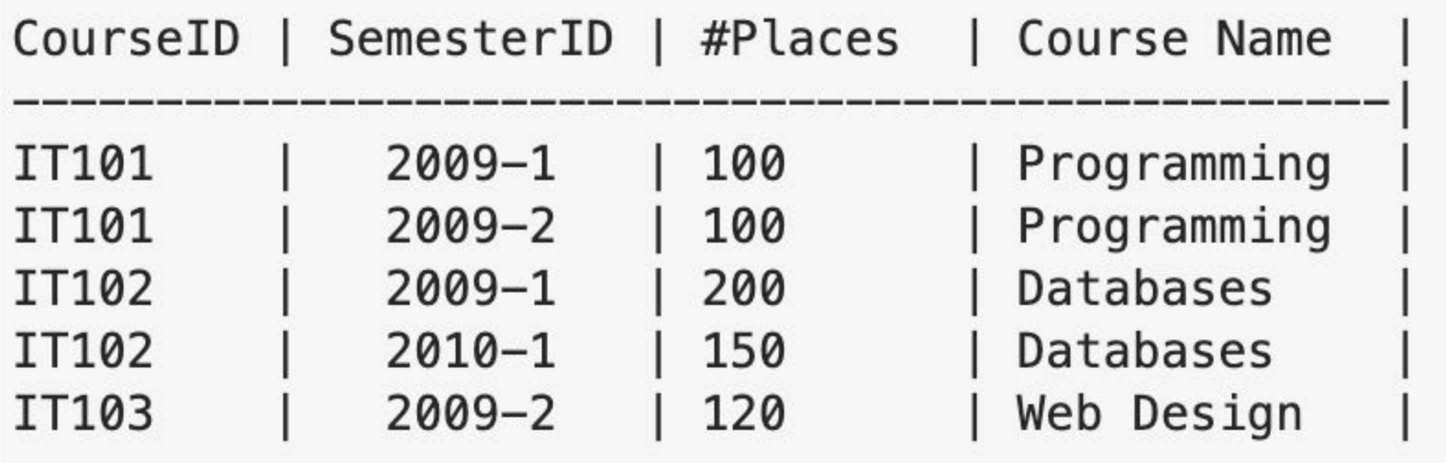
Electric toothbrush manufacturers

Manufacturer	Manufacturer country
Forte	Italy
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japan
Hoch	Germany

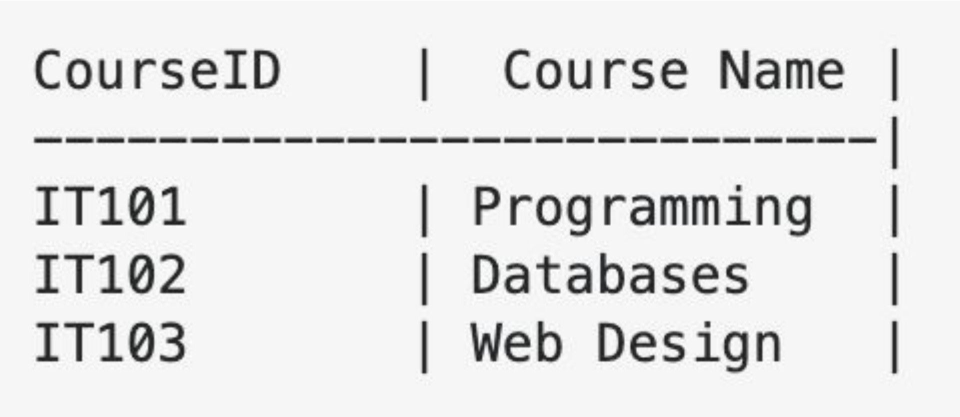


Electric toothbrush models

Manufacturer	Model	Model full name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Brushmaster	SuperBrush	Brushmaster SuperBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime



```
CourseID | SemesterID | #Places  | Course Name  |
------------------------------------------------|
IT101    |   2009-1   | 100      | Programming  |
IT101    |   2009-2   | 100      | Programming  |
IT102    |   2009-1   | 200      | Databases    |
IT102    |   2010-1   | 150      | Databases    |
IT103    |   2009-2   | 120      | Web Design   |
```

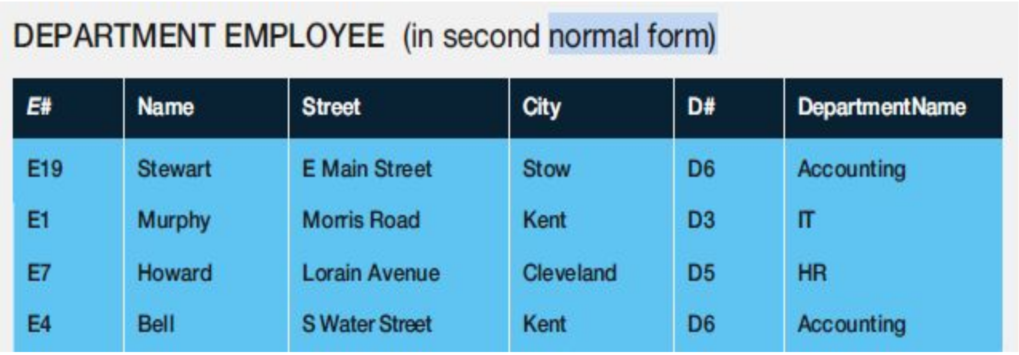


```
CourseID    |  Course Name |
---------------------------|
IT101       | Programming  |
IT102       | Databases    |
IT103       | Web Design   |
```

Third Normal Form (3NF)

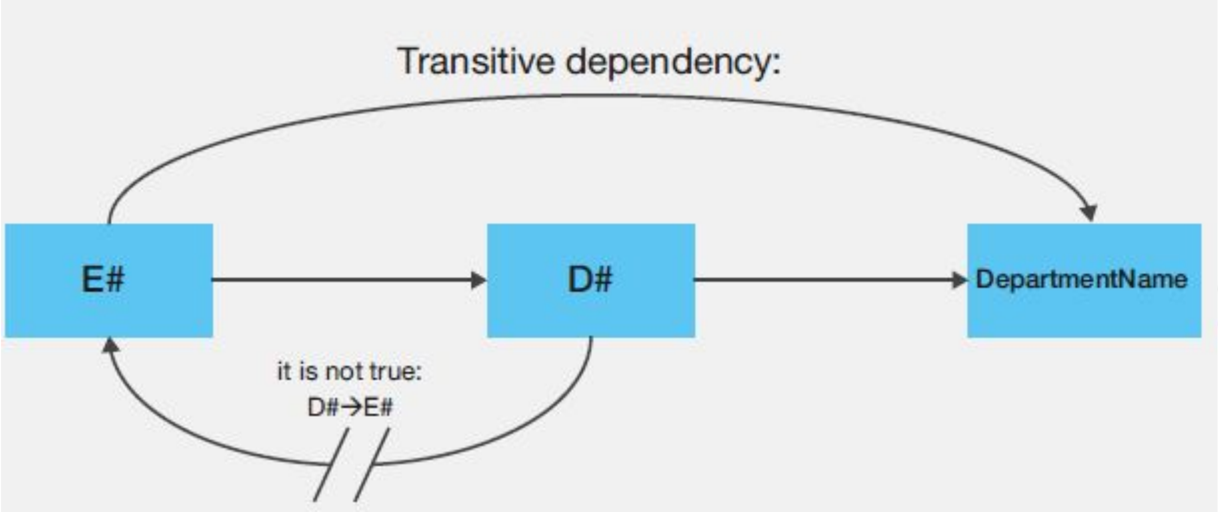
Relational Databases

- A table is in the third normal form when, in addition to the requirements of the second form, no non-key attribute is transitively dependent on any key attribute.
- Transitive dependency:
  - An attribute C is transitively dependent on A if B is functionally dependent on A, C is functionally dependent on B, and A is not functionally dependent on B.



DEPARTMENT EMPLOYEE (in second normal form)

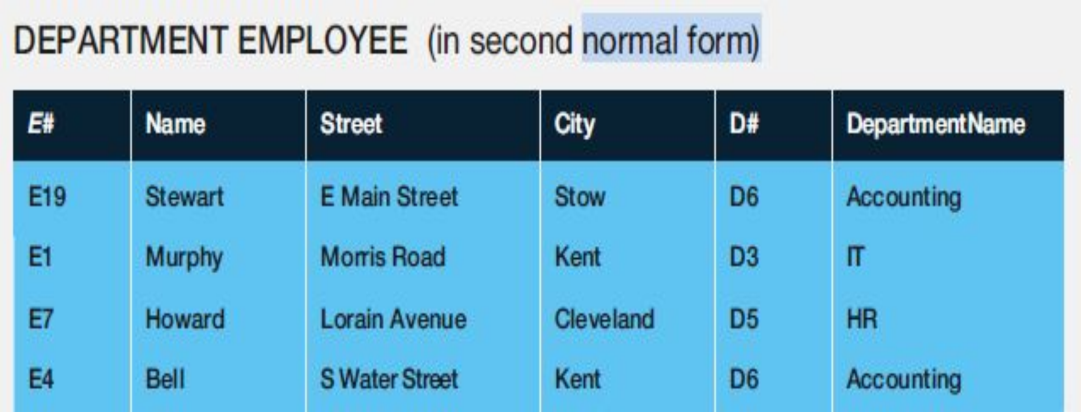
E#	Name	Street	City	D#	DepartmentName
E19	Stewart	E Main Street	Stow	D6	Accounting
E1	Murphy	Morris Road	Kent	D3	IT
E7	Howard	Lorain Avenue	Cleveland	D5	HR
E4	Bell	S Water Street	Kent	D6	Accounting



Transitive dependency:

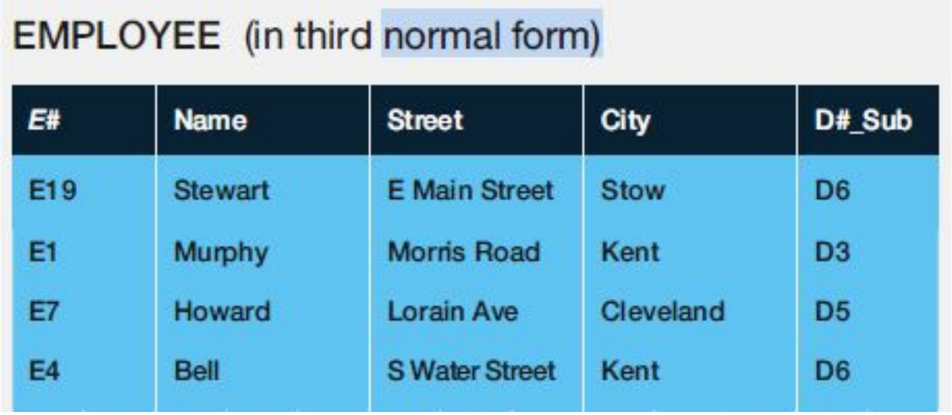
Third Normal Form (3NF)

Relational Databases



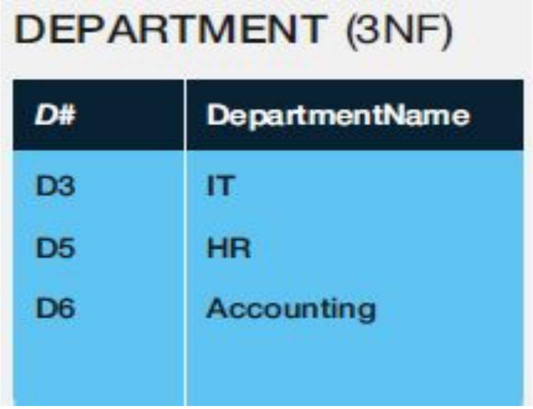
DEPARTMENT EMPLOYEE (in second normal form)

E#	Name	Street	City	D#	DepartmentName
E19	Stewart	E Main Street	Stow	D6	Accounting
E1	Murphy	Morris Road	Kent	D3	IT
E7	Howard	Lorain Avenue	Cleveland	D5	HR
E4	Bell	S Water Street	Kent	D6	Accounting



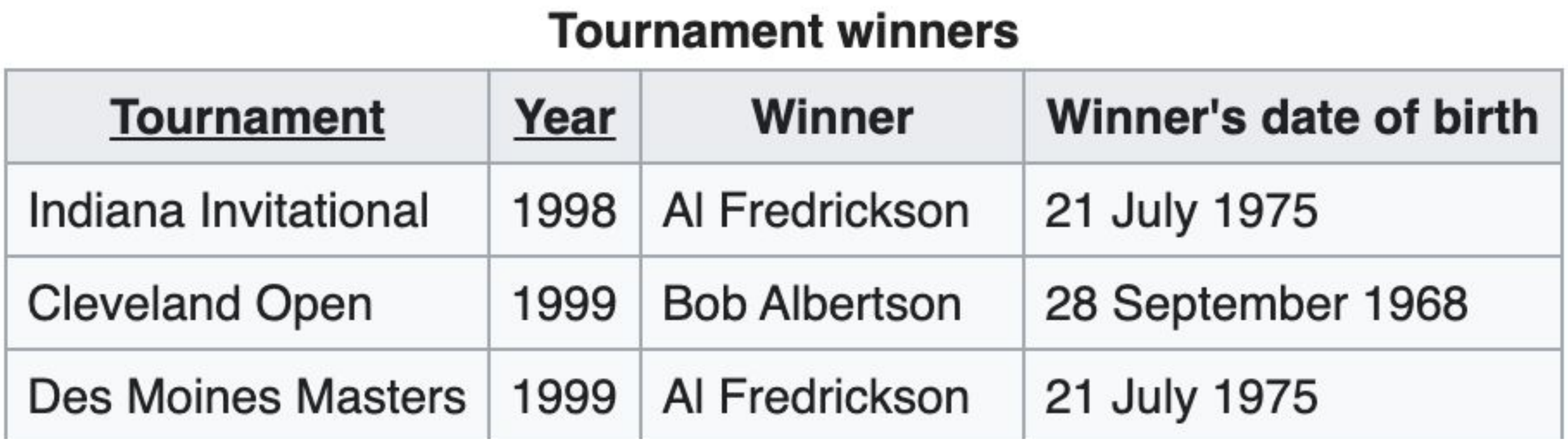
EMPLOYEE (in third normal form)

E#	Name	Street	City	D#_Sub
E19	Stewart	E Main Street	Stow	D6
E1	Murphy	Morris Road	Kent	D3
E7	Howard	Lorain Ave	Cleveland	D5
E4	Bell	S Water Street	Kent	D6



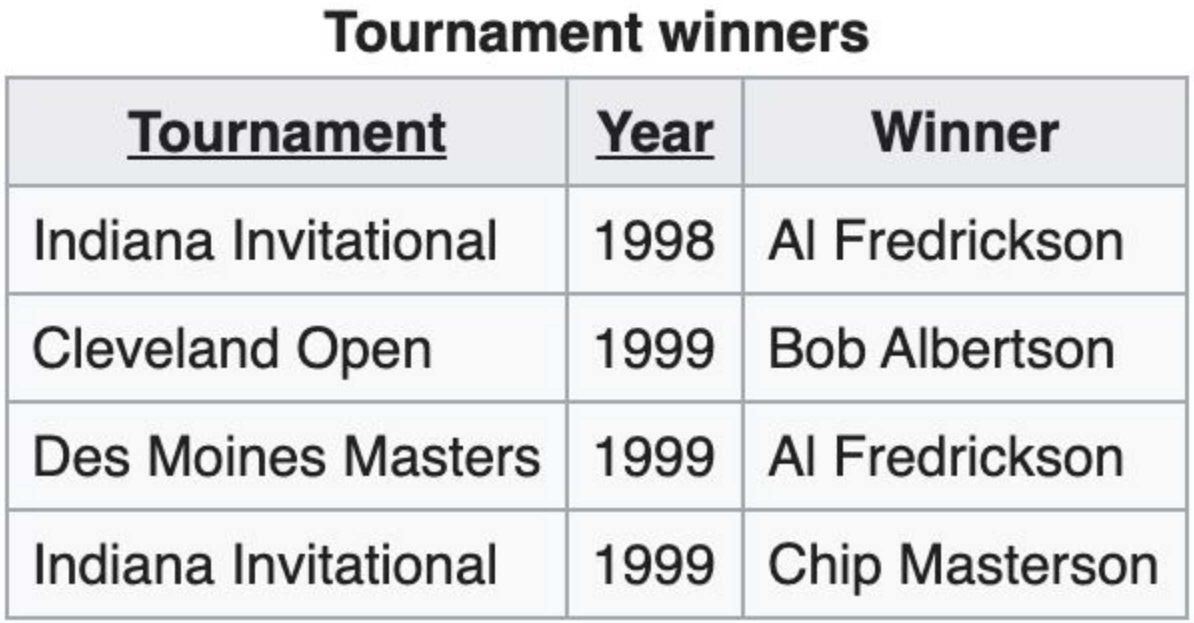
DEPARTMENT (3NF)

D#	DepartmentName
D3	IT
D5	HR
D6	Accounting



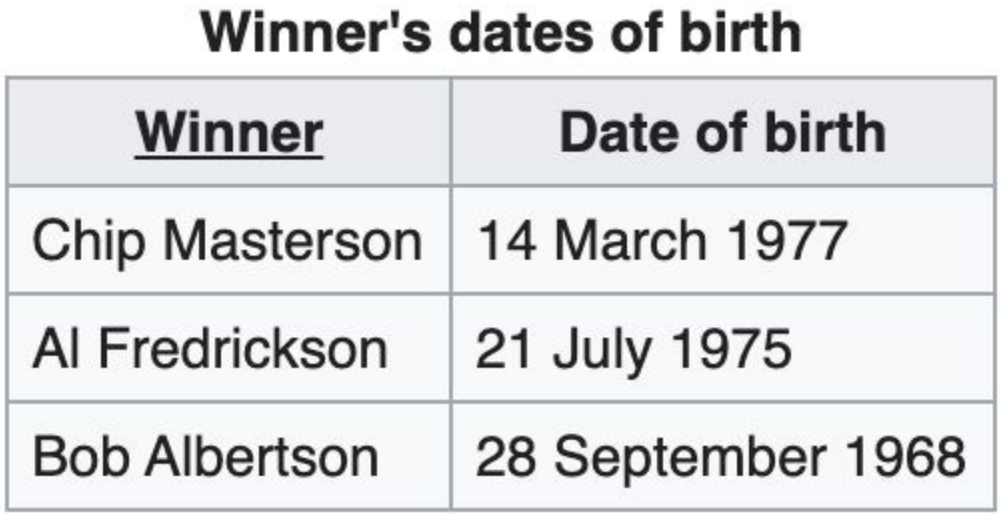
Tournament winners

Tournament	Year	Winner	Winner's date of birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975



Tournament winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson



Winner's dates of birth

Winner	Date of birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Referential Integrity

ActivityID	ActivityDate	Winner
A1	14th March	S2
A2	17th March	S1
A8	25th March	S2

StudentID	StudentName	StudentMajor
S2	Priya	Computers
S1	Mike	History

Denormalization

Relational Databases

- Normalization is only one of many database design goals.
- Normalized (decomposed) tables require additional processing, reducing system speed.
- Normalization purity is often difficult to sustain in the modern database environment.
  - The conflict between design efficiency, information requirements, and processing speed are often resolved through compromises that include denormalization.
- Denormalization is trying to increase performance by reducing the number of joins between tables (as joins can be slow).
  - Data integrity will take a bit of a potential hit, as there will be more copies of the data (to reduce JOINS).

Relational Databases: Benefits

Relational Databases

- Standardized data model
  - Takes time and efforts to convert the raw data to rows and columns.
  - But once done, SQL can be used to query the tables.
- Strong mapping to business requirement
- ACID Transactions
- Data Integrity is ensured
  - Data that does not adhere to the checks are not inserted into the tables (SQL constraints)
- Standard Query Language (SQL), universally used language
- Simplicity

Relational Databases: Issues

Relational Databases

- Scalability:
  - Issues with scaling up
  - Not designed to be distributed
- Impedance mismatch:
  - Object Relational Mapping doesn't work quite well (Think Hibernate - too much boilerplate code)
- Rigid schema design
- Replication
- Joins across multiple nodes/servers? Hard

Use Case: Bookstore

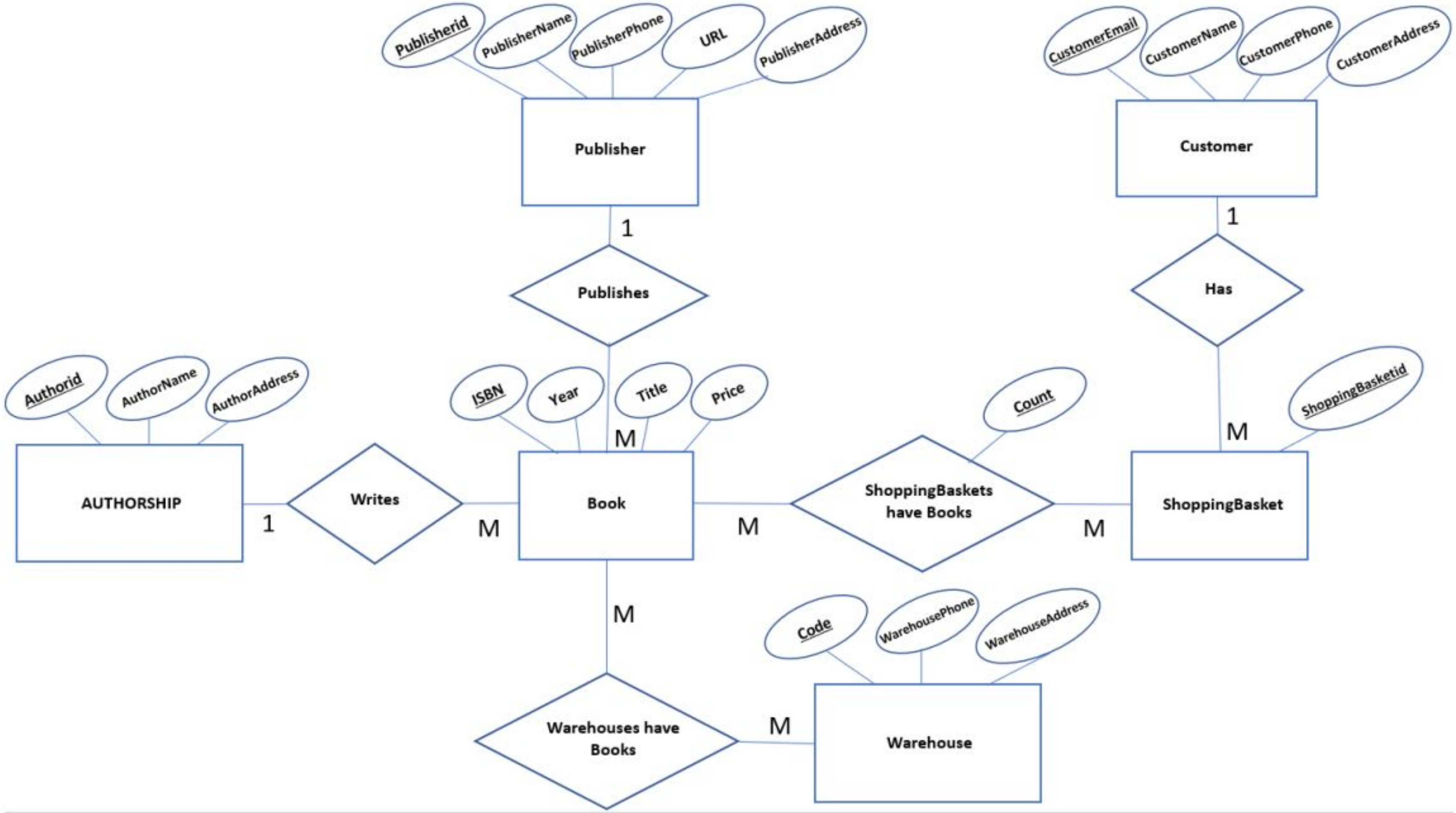
Relational Databases

Information management system for an Online Bookstore (e.g., indigo.ca).

- The Book store has registered customers in order to sell books.
- It also contains publisher's information, and a customer can place the book he desires to buy on a shopping basket.
- A customer has an email, name, phone and address.
- A book has and ISBN, year, title and price.
- Publisher has a name, address, phone and url and publishes several books, but one book can be published by one publisher.
- An author has a name and address and can write several books.
- Books can be written by only one author and they are stored on many warehouses and one warehouse has many books.
- A customer can have several shopping baskets
- Each shopping basket belongs to one customer, where each shopping basket can contain several books.

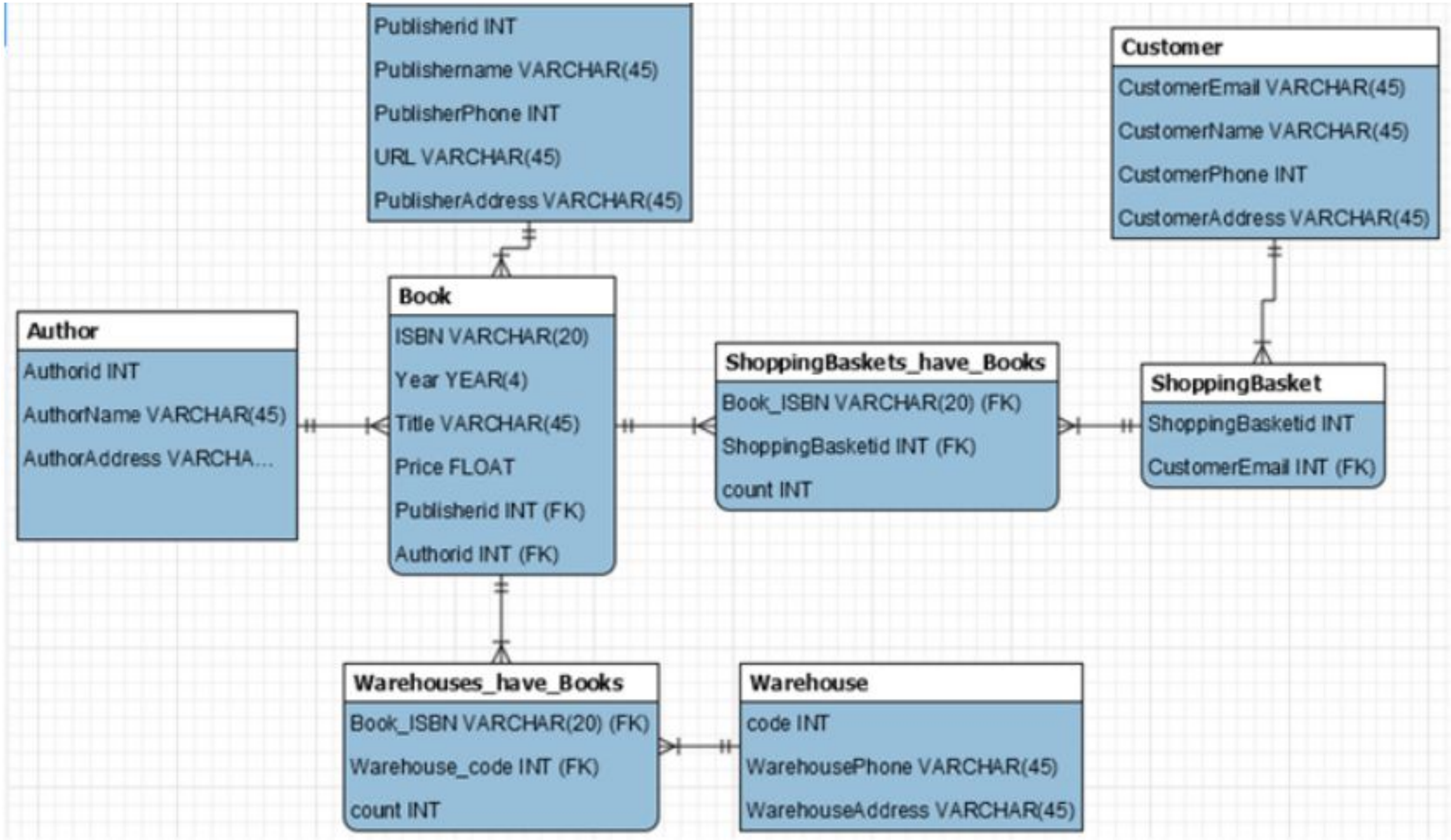
Use Case: Bookstore – Conceptual Design

Relational Databases



Use Case: Bookstore – Logical Design

Relational Databases



Publisherid INT
PublisherName VARCHAR(45)
PublisherPhone INT
URL VARCHAR(45)
PublisherAddress VARCHAR(45)

Customer
CustomerEmail VARCHAR(45)
CustomerName VARCHAR(45)
CustomerPhone INT
CustomerAddress VARCHAR(45)

Author
Authorid INT
AuthorName VARCHAR(45)
AuthorAddress VARCHA...

Book
ISBN VARCHAR(20)
Year YEAR(4)
Title VARCHAR(45)
Price FLOAT
Publisherid INT (FK)
Authorid INT (FK)

ShoppingBaskets_have_Books
Book_ISBN VARCHAR(20) (FK)
ShoppingBasketid INT (FK)
count INT

ShoppingBasket
ShoppingBasketid INT
CustomerEmail INT (FK)

Warehouses_have_Books
Book_ISBN VARCHAR(20) (FK)
Warehouse_code INT (FK)
count INT

Warehouse
code INT
WarehousePhone VARCHAR(45)
WarehouseAddress VARCHAR(45)

Use Case: Bookstore – Physical Design

Relational Databases

```
CREATE TABLE IF NOT EXISTS `BookShop`.`Publisher` (
 `Publisherid` INT NOT NULL,
 `Publishername` VARCHAR(45) NOT NULL,
 `PublisherPhone` INT,
 `URL` VARCHAR(45),
 `PublisherAddress` VARCHAR(45) ,
 PRIMARY KEY (`Publisherid`))
```

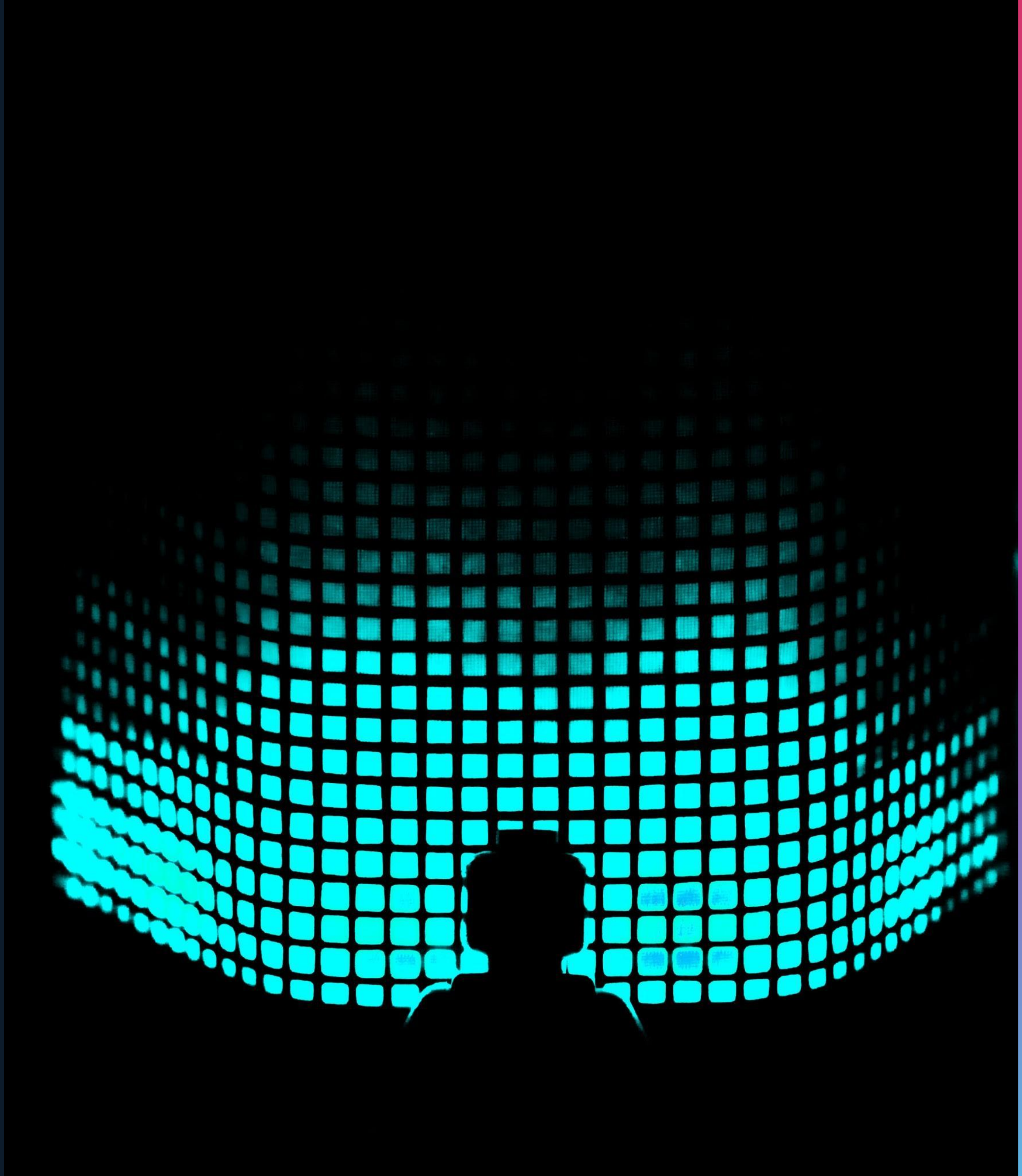
Similarly, SQL DDL (Data Definition Language) statements can be used to create other tables in the database.

Module 3 :
NoSQL Databases

Learning Objectives

In this module, we will introduce:

- What does NoSQL mean?
- The what, why, when and how of NoSQL Databases
- Benefits and Issues in using NoSQL Databases
- How to decide between Relational and NoSQL Database.



In RDBMS

Restaurants (ID, name, City, cuisine, score)

Reviews (ID, rest_fk, stars, author_fk, text)

Users(ID, name, image, last login, city, rank)

Restaurant Review Screen (what should we show here?)

NoSQL way

1. I need to add noise levels
2. bars, game nights, music festivals, etc.

what should I do?

- Put data in places where you can grab it altogether
- No joins (mongo has $lookup though)
- probably you'll have duplicate data
  - no joins,
  - reads end up being really fast
  - writes can suffer (will have to change data in multiple locations)
  - In most apps reads outnumber the writes
- data distributed across multiple machines

Need for NoSQL Databases

NoSQL Databases

Several driving forces behind the adoption of NoSQL databases, including:

- greater scalability (horizontal sharding)
- high read throughput
- very large datasets
- expressive data models (less/non structured/schema less)

What is NoSQL Database?

NoSQL Databases

- NoSQL Databases can be referred to as new generation databases that follow non-relational model, are distributed, open-source, schema-less and horizontally scalable.
- Do not require fixed table schema nor do they use the concept of **joins**.
- Facebook developed Cassandra, one of the most popular NoSQL databases

What is NoSQL Database?

NoSQL Databases

- NoSQL databases have the following properties:
  - The database model is not relational.
  - The focus is on distributed and horizontal scalability.
  - There are weak or no schema restrictions.
  - Data replication is easy.
  - Easy access is provided via an API.
  - The consistency model is not ACID. (CAP Theorem)

NoSQL Databases: BASE Properties

- In relational database systems, transactions at the highest isolation level are always atomic, consistent, isolated, and durable (ACID Properties)
- Web-based applications, on the other hand, are geared towards high availability and the ability to continue working if a computer node or a network connection fails.
- Such partition tolerant systems use replicated computer nodes and a softer consistency requirement called BASE (basically available, soft state, eventually consistent).
  - This allows replicated computer nodes to temporarily hold diverging data versions and only be updated with a delay.
- Instead of ACID approach, NoSQL database follows BASE properties.

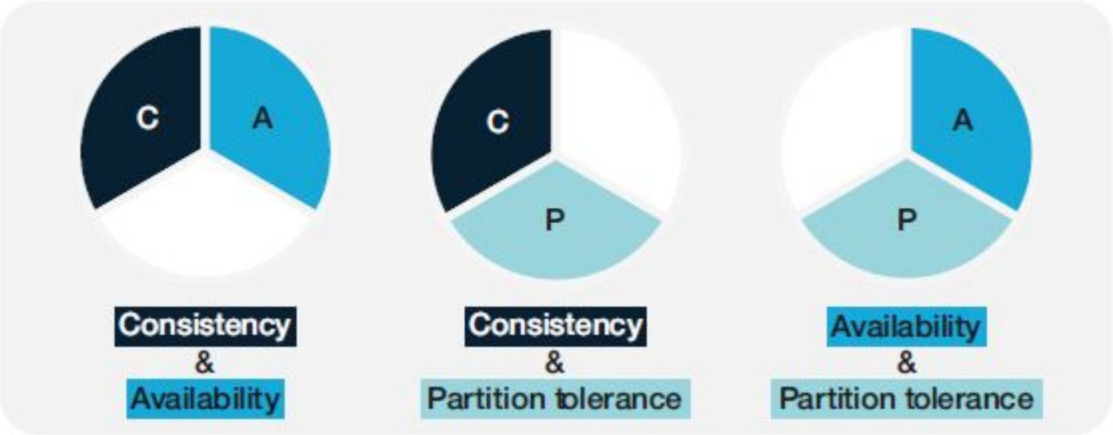
CAP Theorem

NoSQL Databases

- Consistency (C):
  - When a transaction changes data in a distributed database with replicated nodes, all reading transactions receive the current state, no matter from which node they access the database.
- Availability (A):
  - Running applications operate continuously and have acceptable response times.
- Partition tolerance (P):
  - Failures of individual nodes or connections between nodes in a replicated computer network do not impact the system, and nodes can be added or removed at any time without having to stop operation.
- The CAP theorem states that in any massive distributed data-management system, only two of the three properties consistency, availability, and partition tolerance can be ensured.

CAP Theorem

NoSQL Databases



- The 3 possible combinations under CAP theorem.
- Google Spanner is a database that complies with CAP theorem.

CAP Theorem

NoSQL Databases

- Today, NoSQL databases are classified based on the two CAP characteristics they support:
- CP database:
  - A CP database delivers consistency and partition tolerance at the expense of availability.
  - When a partition occurs between any two nodes, the system must shut down the non-consistent node (i.e., make it unavailable) until the partition is resolved.
- AP database:
  - An AP database delivers availability and partition tolerance at the expense of consistency.
  - When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others.
  - When the partition is resolved, the AP databases typically resync the nodes to repair all inconsistencies in the system.
- CA database:
  - A CA database delivers consistency and availability across all nodes.
  - It can’t do this if there is a partition between any two nodes in the system, however, and therefore can’t deliver fault tolerance.

NoSQL Databases: BASE Properties

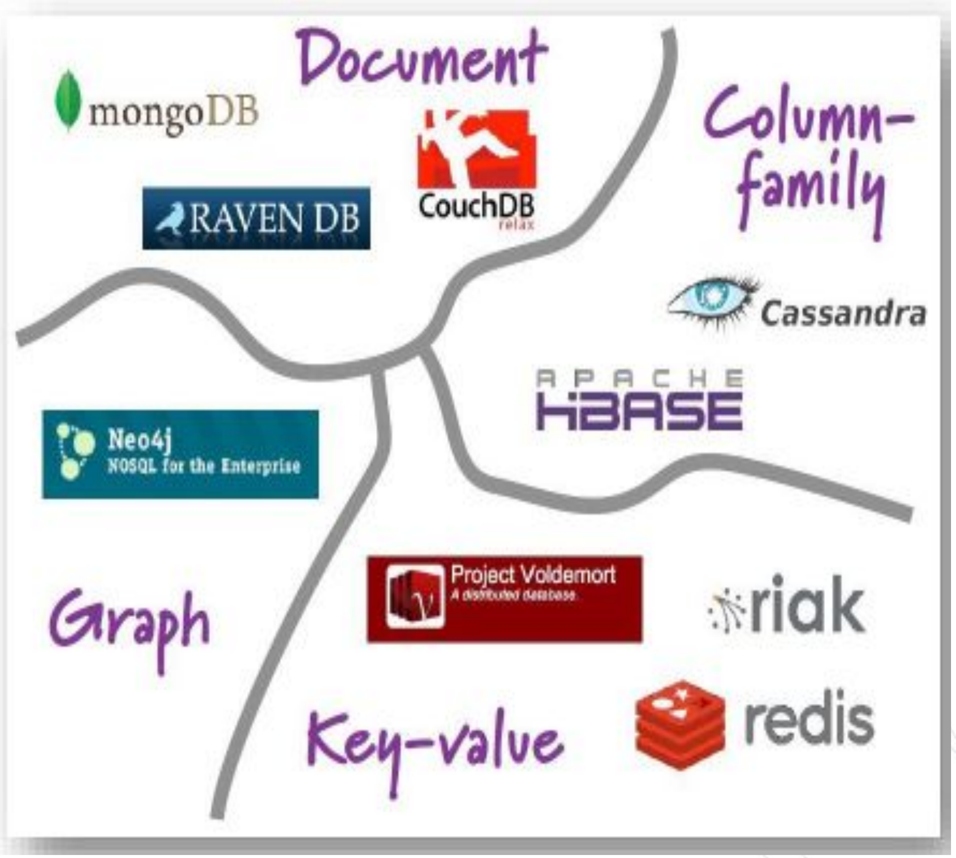
NoSQL Databases

- Basically Available:
  - There can be a partial failure in some parts of the distributed system and the rest of the system continues to function.
  - Often multiple copies of data are maintained on different servers allowing the database to respond to queries even if one of the servers has failed.
  - Example: if a NoSQL database is running on 10 servers without replicating data and one of the servers fails, then 10% of the users' queries would fail, but 90% would succeed.
- Soft State:
  - Data may eventually be overwritten with more recent data.
- Eventually Consistent:
  - An important aspect of NoSQL databases.
  - There may be times when the database is in an inconsistent state. Eventually, the replication mechanism in the NoSQL database will update all copies.

NoSQL Databases

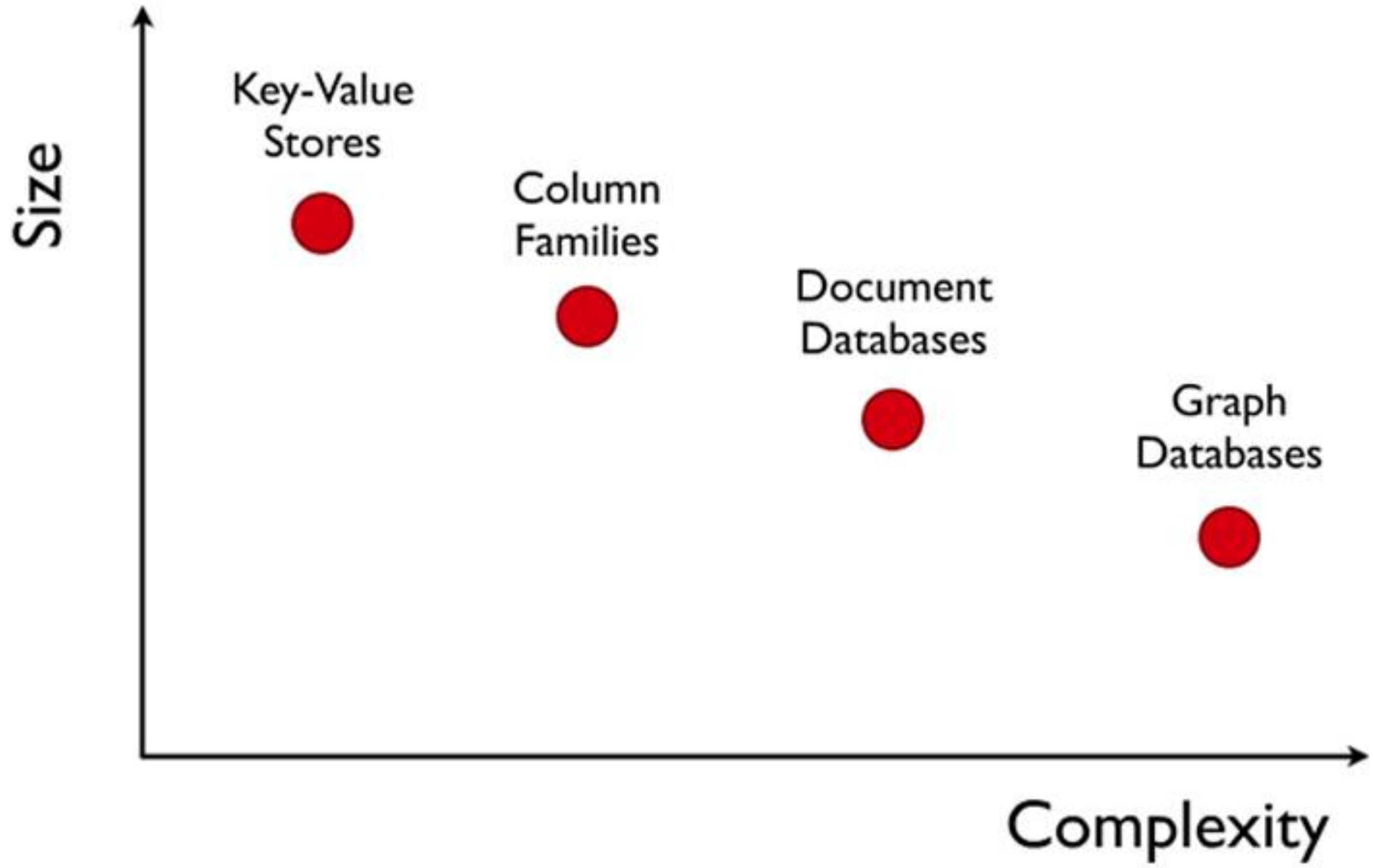
NoSQL Databases

- Core NoSQL technologies are:
  - Key-value stores
  - Column family databases
  - Document stores
  - Graph databases
- These four database models, also called core NoSQL models.
- Each Database has its own query language.
- Other types of NoSQL databases fall in the category of Soft NoSQL, e.g., object databases, grid databases, and the family of XML databases



NoSQL Databases: Complexity

NoSQL Databases



NoSQL Databases: Benefits

NoSQL Databases

- High Performance
- Horizontal Scaling
- Simple Storage mechanism
- Flexible Storage
- No ACID properties
- Avoids complex SQL (joins can be expensive)

NoSQL Databases: Issues

NoSQL Databases

- Non-standardized Query Language
  - Each database has its own query language.
  - Thus, need for learning each query language for a given DBMS
- Data Consistency:
  - Because NoSQL databases do not perform ACID transactions.
  - Instead, it relies on the principle of “eventual consistency”.
- Ensuring data integrity is the responsibility of the developer
  - Checking for data integrity is not a function of DBMS

NoSQL and Relational Databases: Areas of Difference

NoSQL Databases

- Data Models:
  - Build an application without the need to define schema in NoSql Databases.
  - Relational databases require the schema definition prior any data can be captured.
- Data Structure:
  - NoSQL Databases are designed to handle unstructured data.
  - Example: texts, video, tweets and other social media posts etc.
- Development Model:
  - NoSQL Databases are open-source thus no heavy investment required.
  - Relational databases software require licensing fees.

Relational or NoSQL ?

NoSQL Databases

- Choose Relational
  - If you'll have multiple separate apps/programs all talking to the same DB, or you have a definite rigid schema you need to adhere to, etc. a SQL solution starts to shine.
  - If you want to slice or dice your data in different ways
  - Fully ACID systems
  - eg. - billing, payments, gaming

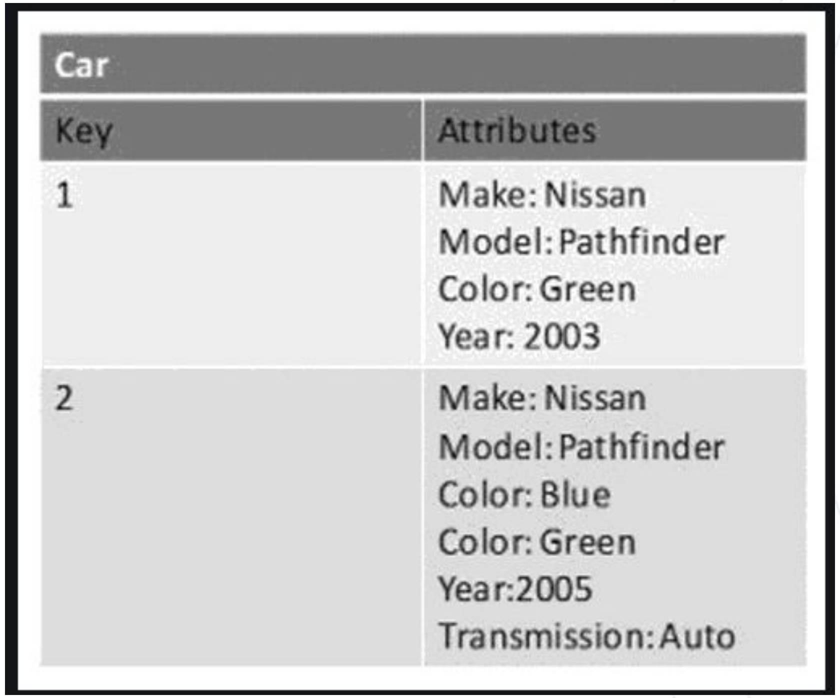
- Choose NoSQL
  - ○ Ideal for apps with no specific schema definitions where your schema is part of your program and evolves with your program
  - ○ if you want to use the same data back and forth into persistence
  - ○ Do not want full ACID properties (eventual consistency significantly reduces DML statements)
  - ○ eg. - who's online at a given time, how many users liked your post are use cases where strong consistency is not required.

**To have strong consistency, developers must compromise on the scalability and performance of their application. Simply put, data has to be locked during the period of update or replication process to ensure that no other processes are updating the same data.*

Key-value Stores

NoSQL Databases

- Simplest NOSQL databases
- The main idea is the use of a hash table
- Access data (values) by strings called keys
- Data has no required format
- Data model: (key, value) pairs
- Basic Operations: Insert(key,value), Fetch(key), Update(key), Delete(key)

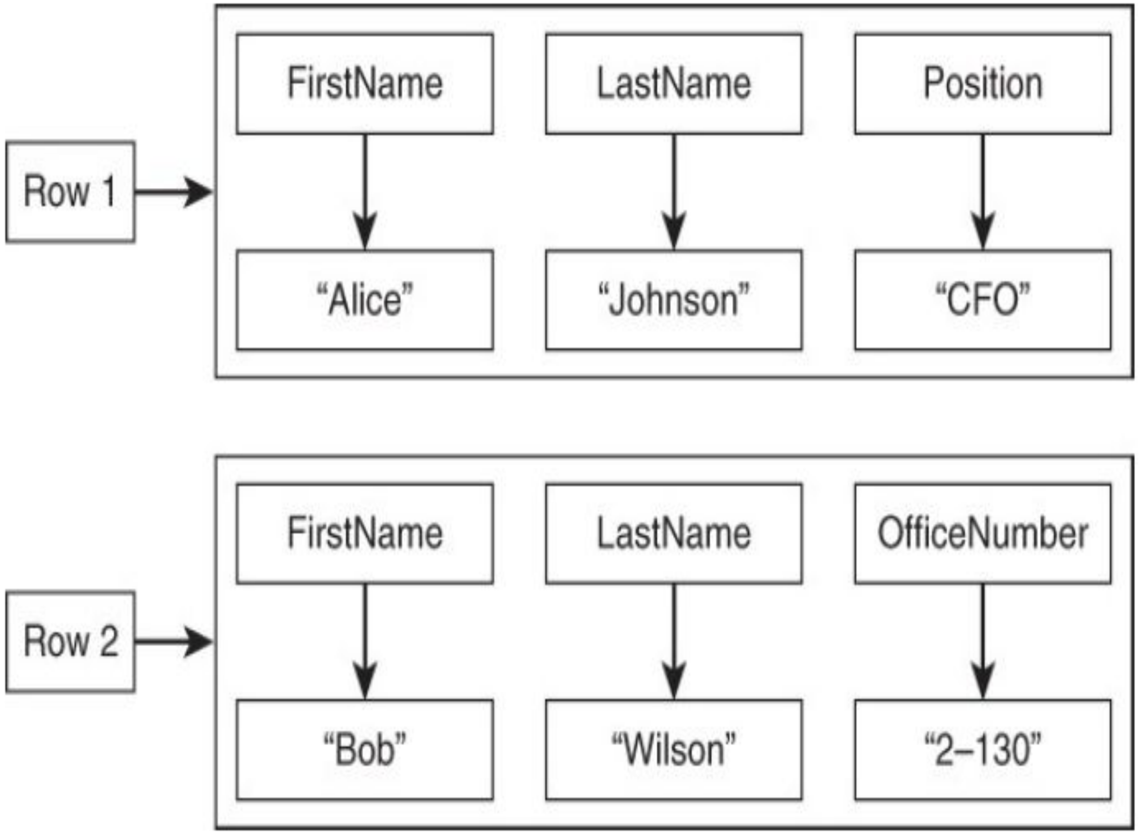


Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Column Family Databases

NoSQL Databases

- The most complex of NoSQL database types.
- The column is lowest/smallest building block of database.
- A column is a name and a value.
    - Some column family databases keep a time stamp along with a name and value.
- A set of columns makes up a row.
- Rows can have the same columns, or they can have different columns.



Row 1

FirstName → "Alice"

LastName → "Johnson"

Position → "CFO"

Row 2

FirstName → "Bob"

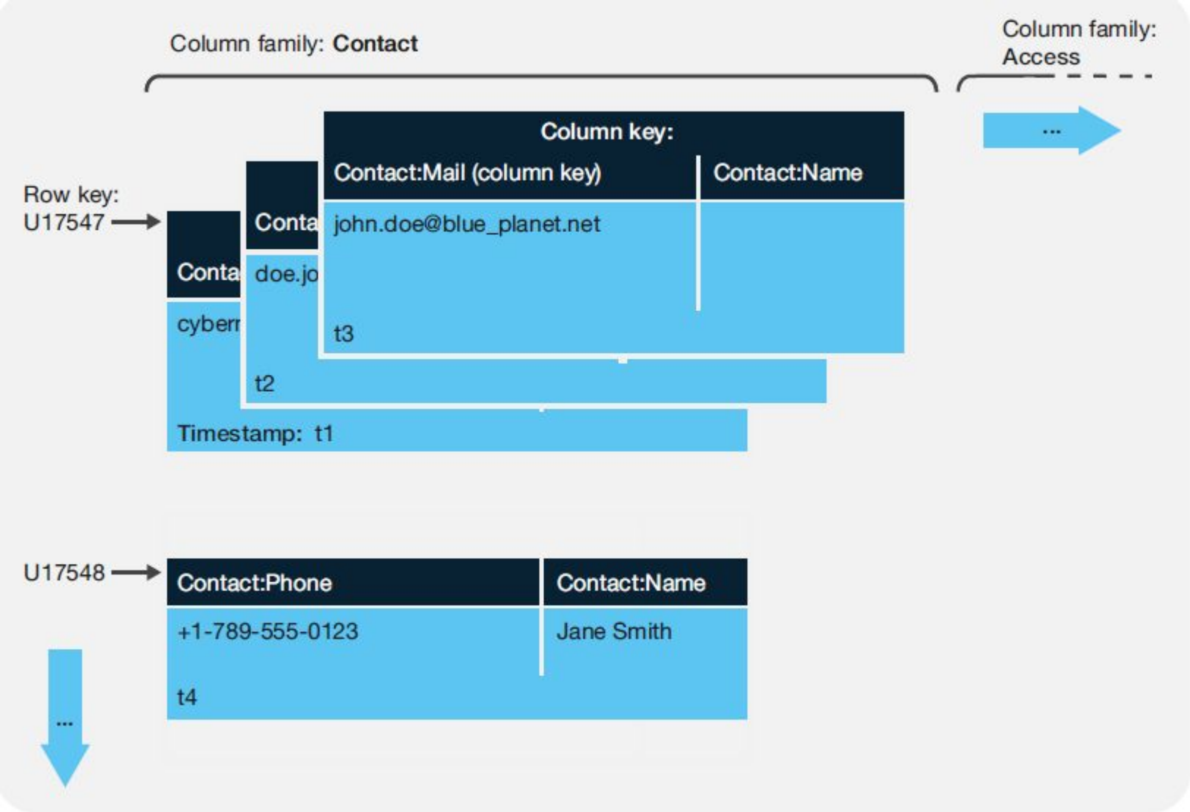
LastName → "Wilson"

OfficeNumber → "2–130"

Column Family Databases: Google BigTable

NoSQL Databases

- In the Bigtable model, a table is a sparse, distributed, multidimensional, sorted map. https://cloud.google.com/bigtable/docs/how-to



Document Databases

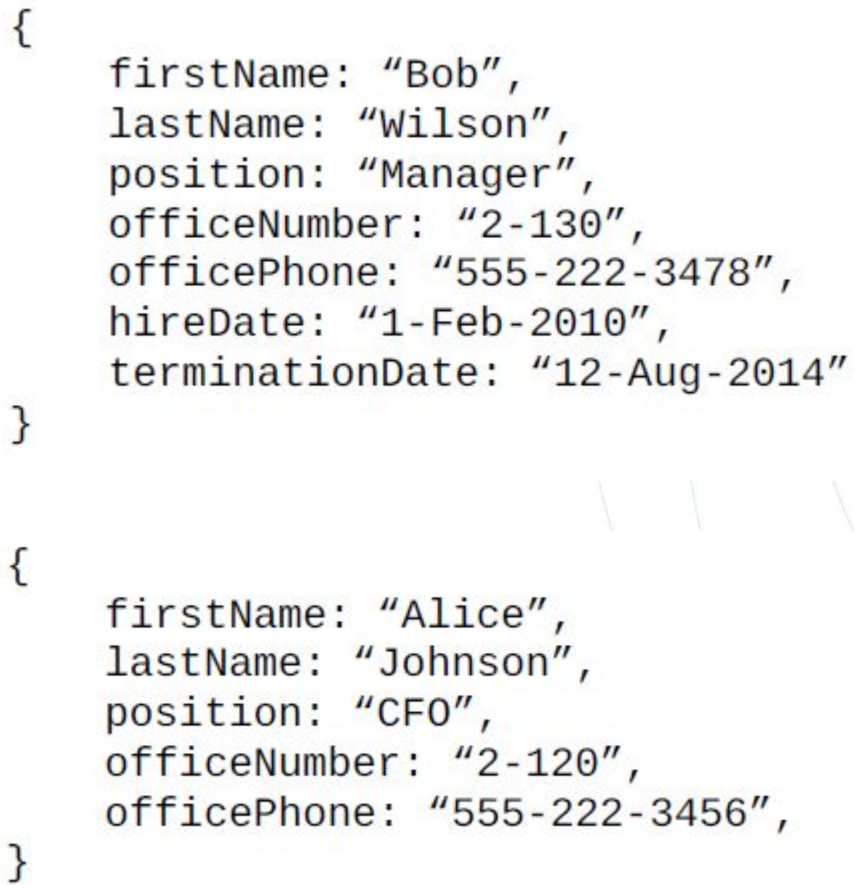
NoSQL Databases

- Also called document-oriented databases.
- Use a key-value approach to storing data but with important differences from key-value databases.
- A document database stores values as documents.
- Documents are semi-structured entities:
  - Such as JavaScript Object Notation (JSON) or Extensible Markup Language (XML).
- Important characteristic: Do not have to define a fixed schema before you add data to the database.
  - Simply adding a document to the database creates the underlying data structures needed to support the document.

Document Databases

NoSQL Databases

- The attributes hireDate and terminationDate are in Bob’s document but not Alice’s.
- This is not a problem from the database perspective.
- Developers can add attributes as needed, but their programs are responsible for managing them.
- Document databases provide application programming interfaces (APIs) or query languages that enable you to retrieve documents based on attribute values.

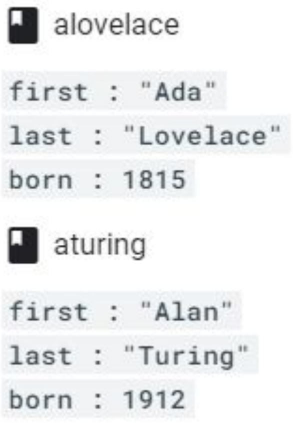


```
{
        firstName: “Bob”,
        lastName: “Wilson”,
        position: “Manager”,
        officeNumber: “2-130”,
        officePhone: “555-222-3478”,
        hireDate: “1-Feb-2010”,
        terminationDate: “12-Aug-2014”
}

{
        firstName: “Alice”,
        lastName: “Johnson”,
        position: “CFO”,
        officeNumber: “2-120”,
        officePhone: “555-222-3456”,
}
```

Document Database: Google Firestore

NoSQL Databases

- In Firestore, the unit of storage is the document.
  - there are no tables or rows.
- All documents must be stored in collections.
  - Documents can contain subcollections and nested objects, bo which can include primitive fields like strings or complex obje like lists.
- Every document in Firestore is uniquely identified by its loca within the database.
- https://cloud.google.com/firestore/docs/data-model

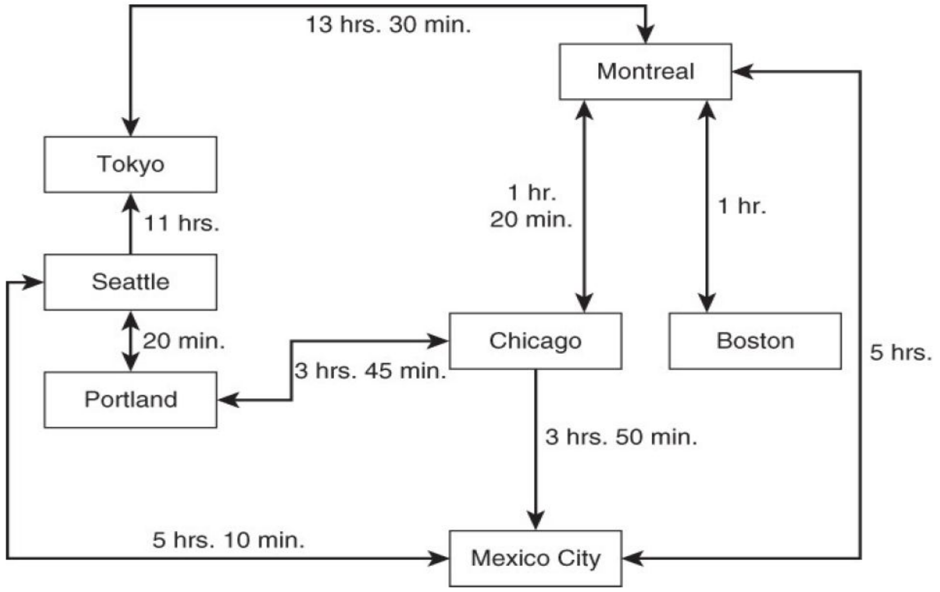


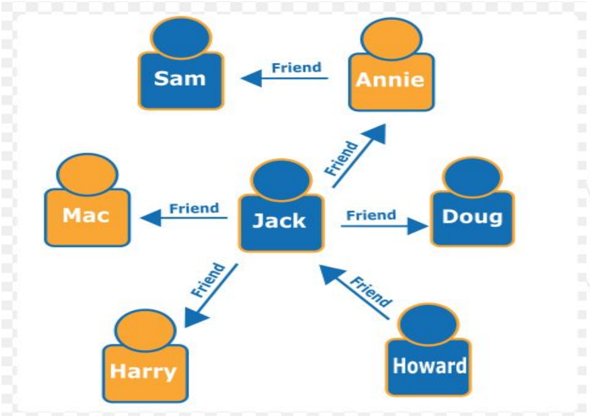
```
alovelace

first : "Ada"
last : "Lovelace"
born : 1815

aturing

first : "Alan"
last : "Turing"
born : 1912
```

Graph Databases

NoSQL Databases

- A graph database uses structures called nodes and relationships.
- A node is an object that has an identifier and a set of attributes.
- A relationship is a link between two nodes that contain attributes about that relation.
- Nodes can represent people, and relationships can represent their friendships in social networks.
- A node could be a city, and a relationship between cities could be used to store information about the distance and travel time between cities.





NoSQL Databases: Real Life Use-Cases

NoSQL Databases

- **Ad targeting platforms: Paypal**
  - Where to display the ad or an offer on the Web page is important for direct revenue.
  - To decide where to put the ad/offer and which user groups to target, ad platforms collect user behavioral data, demographic data and psychographic characteristics.
  - NoSQL database enables to track these attributes.

- **Content and Metadata Store: McGraw-Hill**
  - Need to store the contents, documents and articles in order to integrate different learning tools into a single platform.
  - Metadata is most heavily accessed data, in case of content-driven applications, and needs low response time.
  - Document databases can be used to build custom content-driven applications for increased flexibility and fast access.

NoSQL Databases: Real Life Use-Cases

NoSQL Databases

- **Data Driven User Profiles**
  - NoSQL database can be used to store User IDs, their preferences, multiple ID mappings and additional user information so that the app can quickly look up a user and authenticate the process.
  - https://developer.mongodb.com/how-to/creating-user-profile-store-game-nodejs-mongodb/

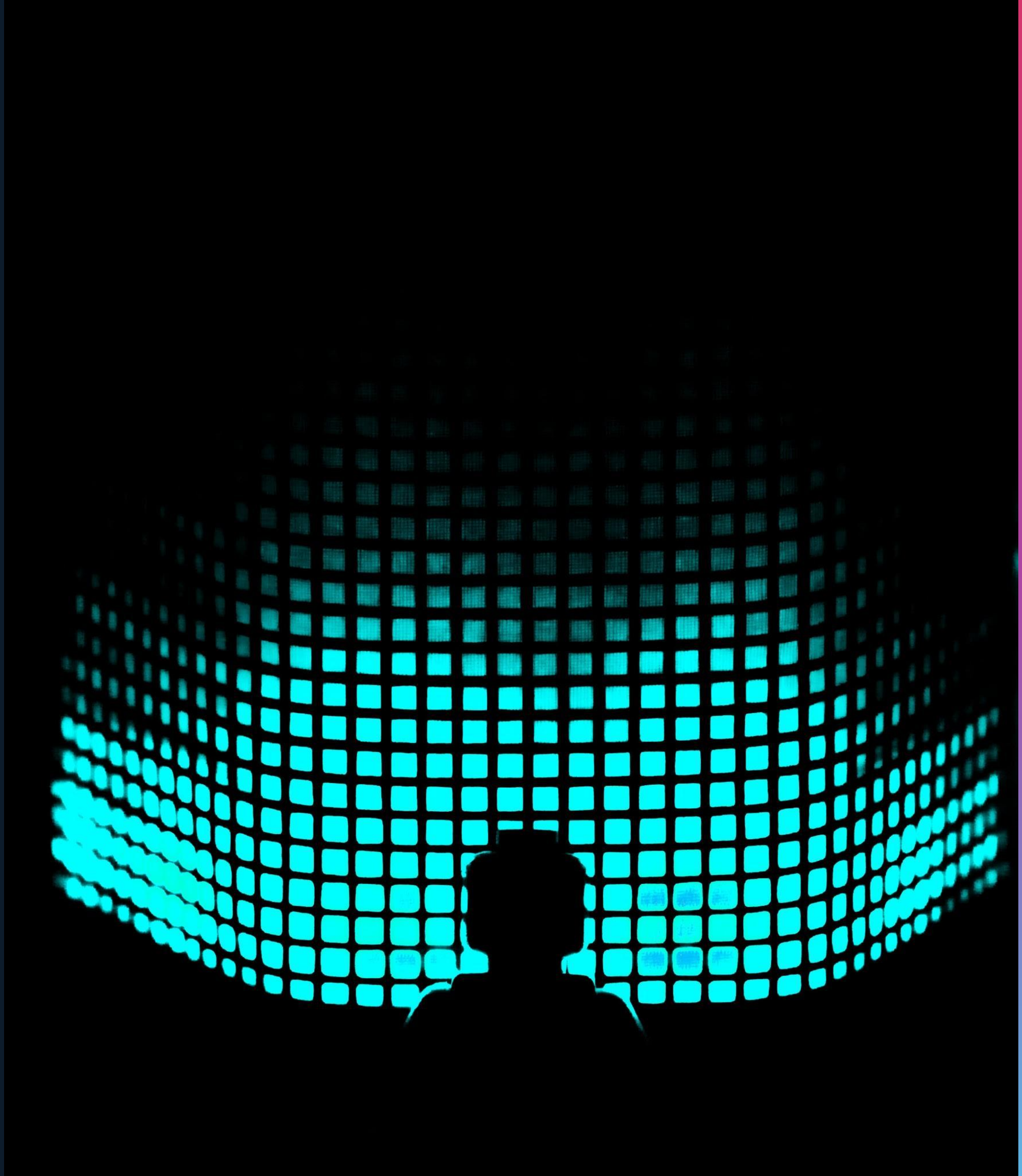
Module 2 :

Data Warehouse Introduction

Learning Objectives

In this module, we will introduce:

- What is Data Warehouse



What is Data Warehouse?

Data Warehouse introduction

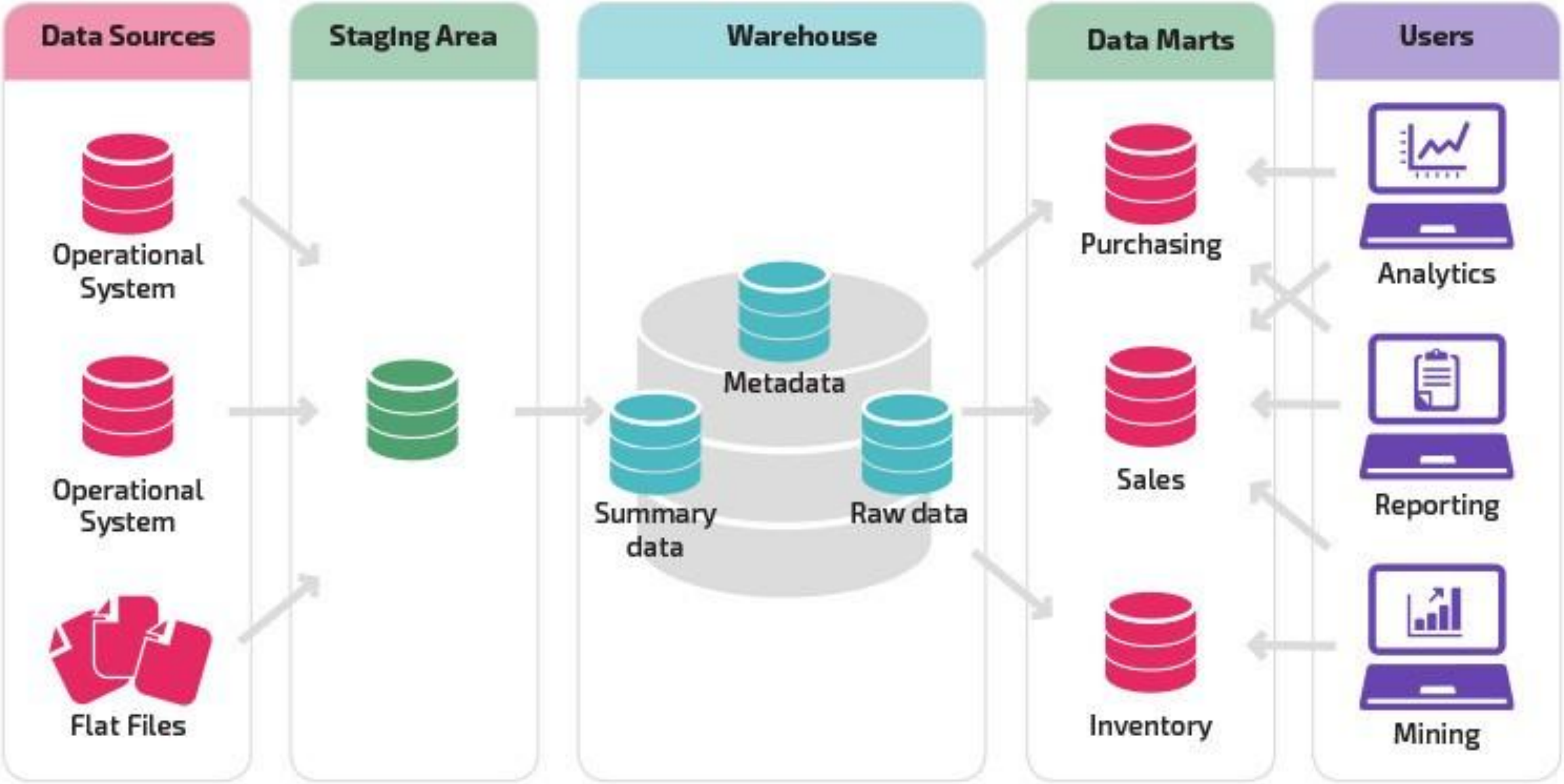
“ *A data warehouse is a system that extracts, cleans, conforms and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making…*”

…”*It’s the place where users go to get their data*”

- Ralph kimbal

What is Data Warehouse?

Data Warehouse introduction



Data Warehousing Goals

Data Warehouse introduction

- Make information accessible
- Present information consistently
- The DW/BI system must adapt to change
- Present the information in a timely way
- The information must be secured
- Serve as trustworthy and authoritative foundation for improved decision making
- The business community must accept the DW/BI system to deem it successful

Operational vs Analytical Systems

Dimensional Modeling

	Operational System	Analytical System
Purpose	Execution of a business process	Measurement of a business process
Primary Interaction Style	Insert, Update, Query, Delete	Query
Scope of Interaction	Individual transaction	Aggregated transactions
Query Patterns	Predictable and stable	Unpredictable and changing
Temporal Focus	Current	Current and historic
Design Optimization	Update concurrency	High-performance query
Design Principle	Entity-relationship (ER) design in third normal form (3NF)	Dimensional design (Star Schema or Cube)
Also Known As	Transaction System On Line Transaction Processing (OLTP) System Source System	Data Warehouse System Data Mart

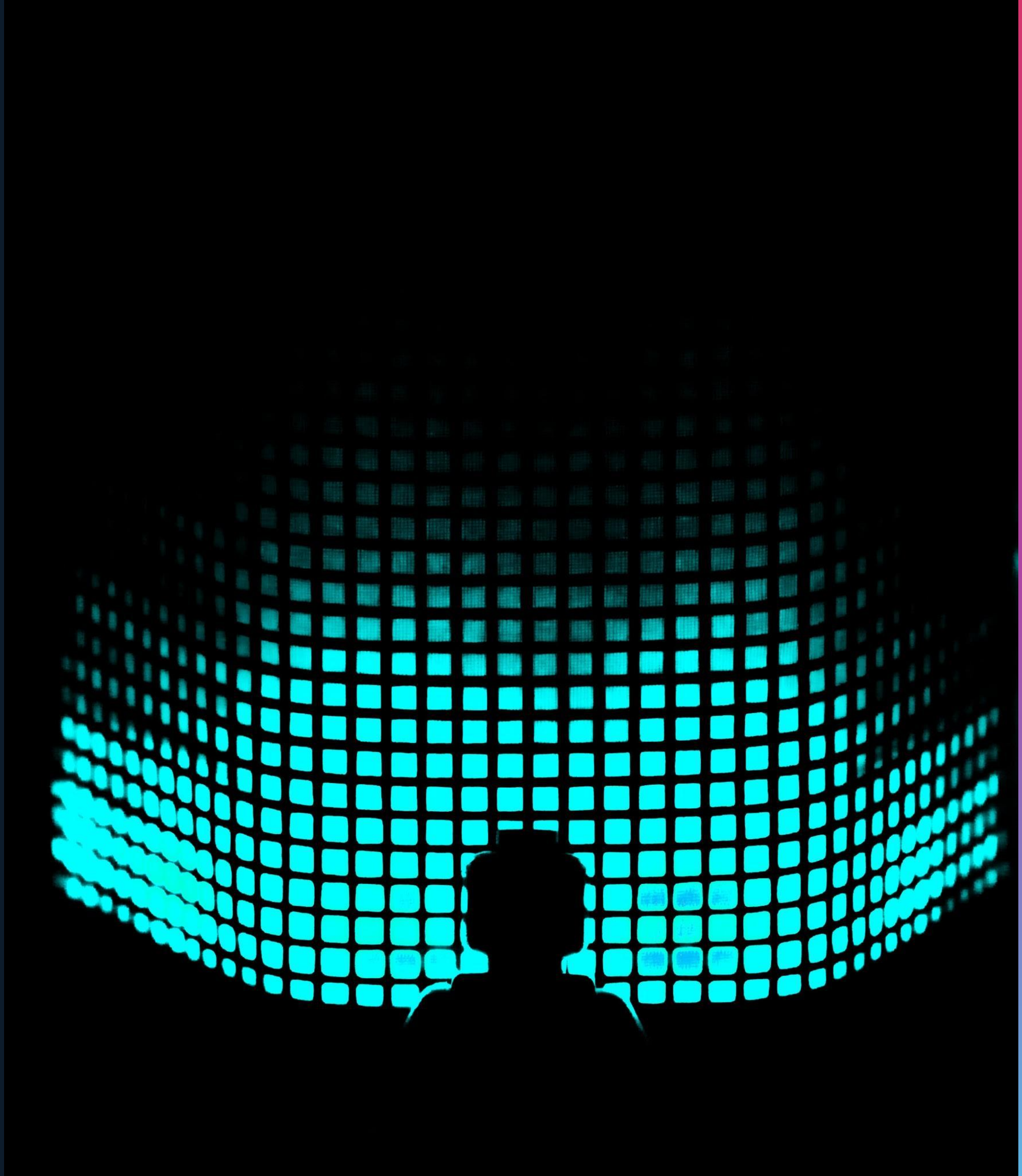
Module 3:

Dimension Modeling

Learning Objectives

In this module, we will introduce:

- Dimensional Modeling
- Facts and Dimension tables
- Kimball Data Warehouse design technique



What is Dimensional Modeling?

Dimensional Modeling

- Developed by Ralph Kimball, Dimensional data modeling is one of the data modeling techniques used in data warehouse design.
- Ralph Kimball developed dimensional model in response to a demand from end-users for an easy way to specify Reports.
- **Goal of dimensional modeling:**
  - To improve the data retrieval, it is optimized for the SELECT operation.
- Dimensional data modelling is best suited for the data warehouse star and snowflake schema.
- How is it different from ER Modeling?
  - Goal of ER modeling: Normalize the data by reducing the redundancy.
  - Whereas Dimensional modeling addresses following 2 requirements:
    1. Deliver data that’s understandable to the business users
    2. Deliver fast query performance.

What is Dimensional Modeling?

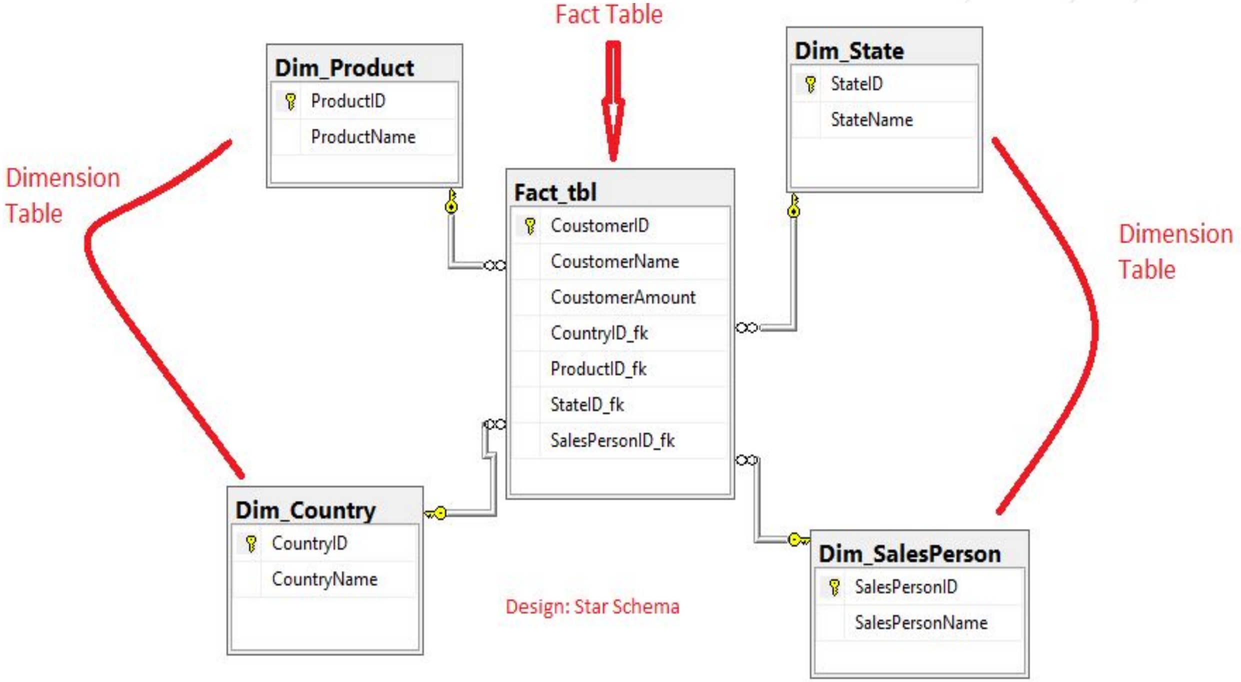
Dimensional Modeling

Dimensional model is assembled by

- Fact Table — Store performance measurements resulting from business event.
- Dimension Table — Integral companions to a fact table. (contains textual context associate with a business process measurement event)

Implemented in a relational database, the dimensional model is called a star schema.

Implemented in a multidimensional database, it is known as a cube.



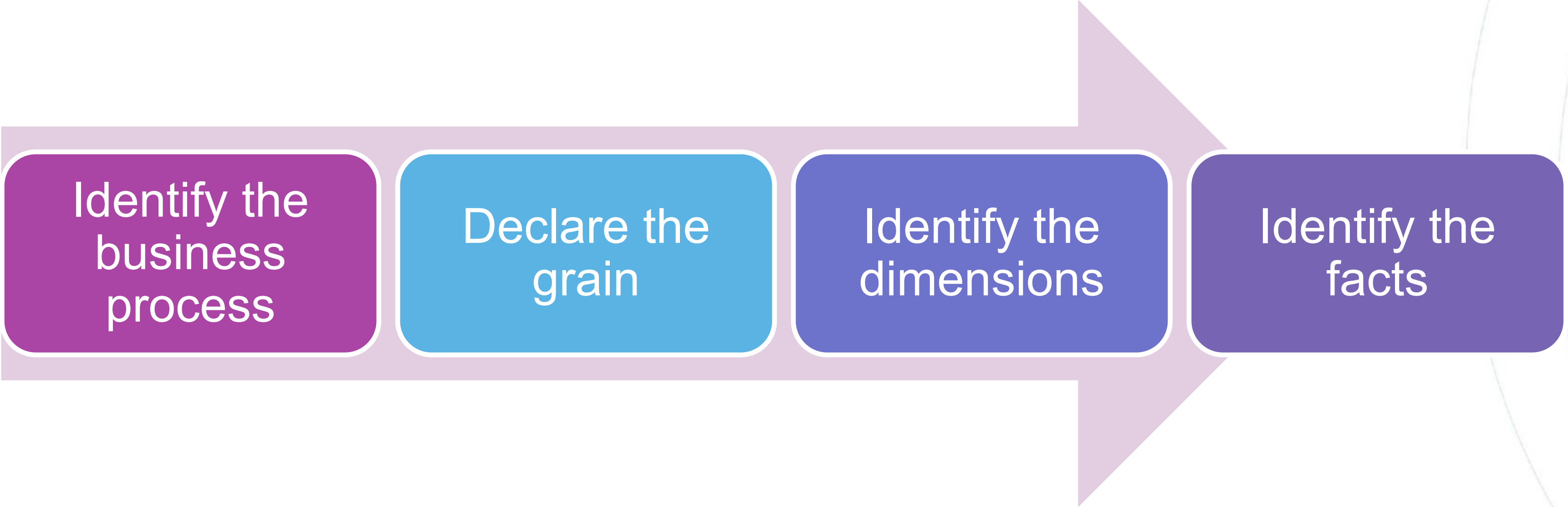
Measurements → Facts

Context Descriptors → Dimensions

Kimball Dimensional Modeling Technique

Dimensional Modeling

- The four key decisions made during the design of a dimensional model include:



Identify the business process

Declare the grain

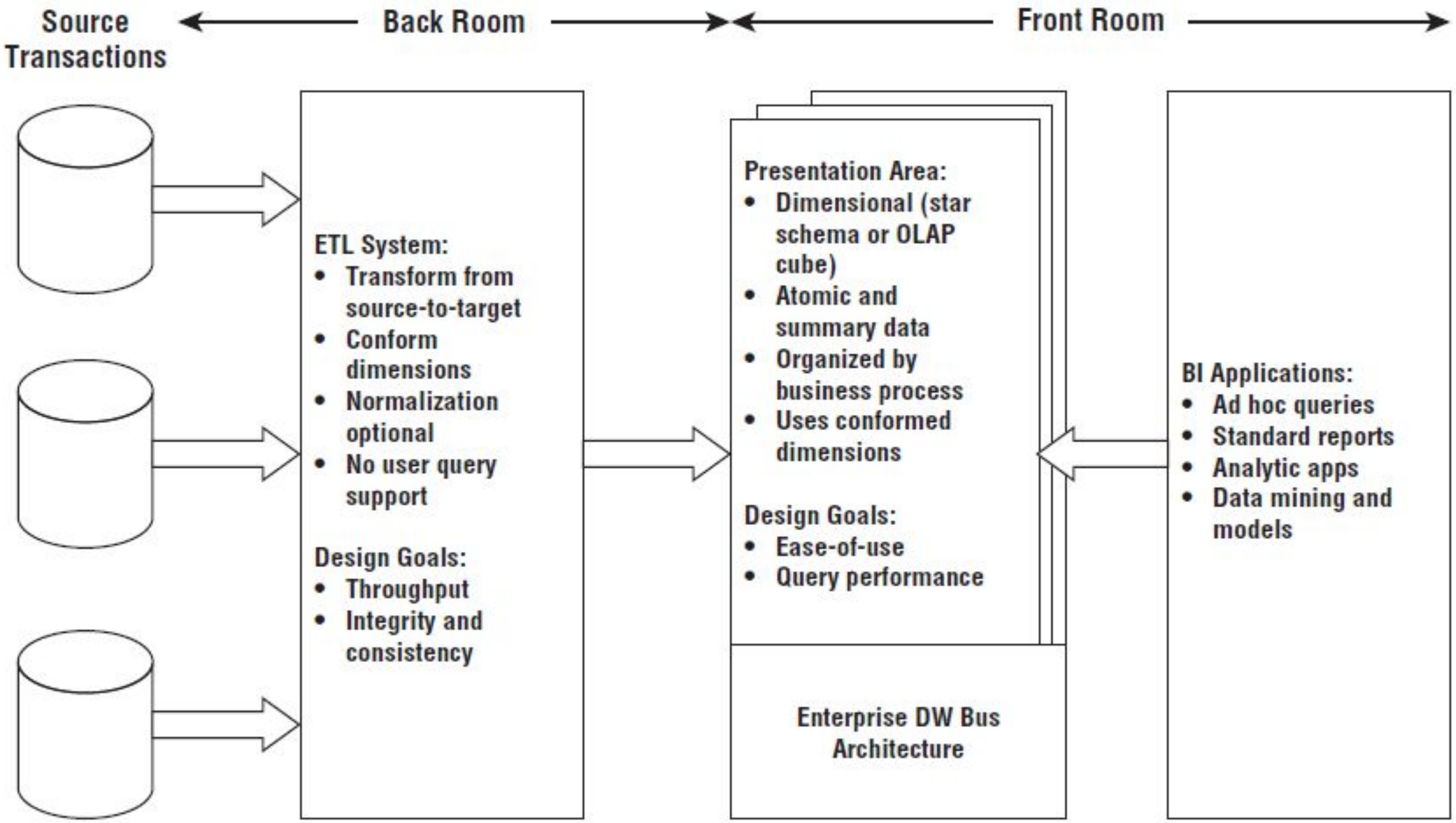
Identify the dimensions

Identify the facts

- Following the business process, grain, dimension, and fact declarations, the design team determines the table and column names, sample domain values, and business rules.

Kimball DW/BI Architecture

Dimensional Modeling



ETL System:

- Transform from source-to-target
- Conform dimensions
- Normalization optional
- No user query support

Design Goals:

- Throughput
- Integrity and consistency

Presentation Area:

- Dimensional (star schema or OLAP cube)
- Atomic and summary data
- Organized by business process
- Uses conformed dimensions

Design Goals:

- Ease-of-use
- Query performance

Enterprise DW Bus Architecture

BI Applications:

- Ad hoc queries
- Standard reports
- Analytic apps
- Data mining and models

1. Identify the Business Process

Dimensional Modeling

- Operational activities performed by your organization
  - *taking an order, processing an insurance claim, registering students for a class, or snapshotting every account each month*.

- Performance metrics $\xrightarrow{\text{Translate into}}$ facts in a fact table
- Choosing the process is important because it defines a specific design target and allows the grain, dimensions, and facts to be declared.
- Determine discrete business processes:
  - Sales, inventory, customer registration etc.
- Each business process corresponds to a row in the enterprise data warehouse bus matrix.

2. Declare the grain

Dimensional Modeling

- Grain: level of detail stored in the data warehouse.
  - Do we store all products, or just store the product categories?
- Each fact table can have a different grain. Try to implement the lowest possible dimension grain:
  - Not because the users need individual records
  - But because the users might want to aggregate the data at different level of details
- Each proposed fact table grain results in a separate physical table;
  - different grains must not be mixed in the same fact table.
- Example grain declarations include:
  - One row per scan of an individual product on a customer's sales transaction
  - One row per line item on a bill from a doctor
  - One row per individual boarding pass scanned at an airport gate
  - One row per daily snapshot of the inventory levels for each item in a warehouse
  - One row per bank account each month

3. Identify Dimensions

Dimensional Modeling

- “who, what, where, when, why, and how” context surrounding a business process event.
- Dimension tables contain the descriptive attributes used by BI applications for filtering and grouping the facts.
- Selection Criteria (Gender)
- How do you want to slice the data?
- Time Dimension is always there.

4. Identify Facts

Dimensional Modeling

- Facts: measurements that result from a business process event
  - almost always numeric

- Within a fact table, only facts consistent with the declared grain are allowed.
  - For example, in a retail sales transaction, the quantity of a product sold, and its extended price are good facts, whereas the store manager's salary is disallowed.

Fact Table Types

Dimensional Modeling

- Transactional — holds data of most detailed level. One row per fact occurs at a certain point.
- Periodic snapshots— depends on Transactional fact table, deliver the chosen performance. Holds a snapshot of all data for a specific points in time. It can be a day, a week, or a month.
- Temporal snapshots — introduces the concept of time Intervals into a fact table, allowing to save a lot of space, optimizing performances while allowing the end user to have the logical equivalent of the "picture of the moment" they are interested in.
- Accumulating snapshots — show the activity of process that has well-defined beginning and end. Often has multiple date columns, each represent a milestone in the process. With so many date columns, it can be difficult to manage.

Designing Fact Table: Steps

Dimensional Modeling

- Identify a business process for analysis (like sales).
- Identify measures of facts (sales dollar), by asking questions like ‘what number of X are relevant for the business process?’, replacing the X with various options that make sense within the context of the business.
- Identify dimensions for facts (product dimension, location dimension, time dimension, organization dimension), by asking questions that make sense within the context of the business, like ‘analyse by X’, where X is replaced with the subject to test.
- List the columns that describe each dimension (region name, branch name, business unit name).
- Determine the lowest level (granularity) of summary in a fact table (e.g. sales dollars).

10 Essential Rules of Dimensional Modeling

Dimensional Modeling

Rule #1: Load detailed atomic data into dimensional structures.

Rule #2: Structure dimensional models around business processes.

Rule #3: *Ensure that every fact table has an associated date dimension table.*

Rule #4: *Ensure that all facts in a single fact table are at the same grain or level of detail.*

Rule #5: *Resolve many-to-many relationships in fact tables.*

Rule #6: *Resolve many-to-one relationships in dimension tables.*

Rule #7: *Store report labels and filter domain values in dimension tables.*

Rule #8: *Make certain that dimension tables use a surrogate key.*

Rule #9: *Create conformed dimensions to integrate data across the enterprise.*

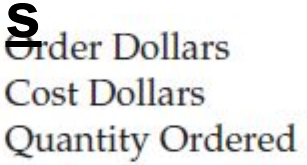
Rule #10: *Continuously balance requirements and realities to deliver a DW/BI solution that's accepted by business users and that supports their decision-making.*

https://www.kimballgroup.com/2009/05/the-10-essential-rules-of-dimensional-modeling/

Facts and Dimensions Example: Order Process

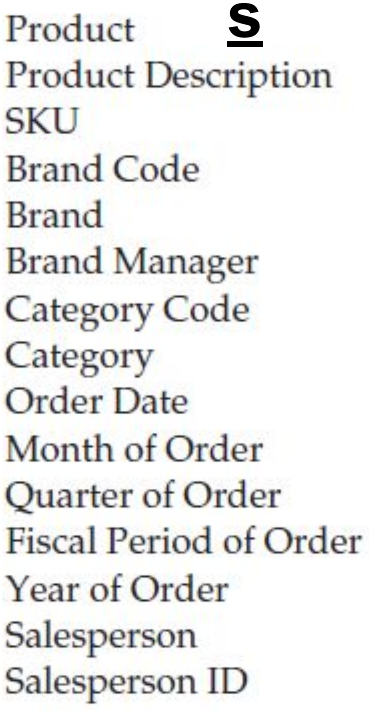
Dimensional Modeling

Facts

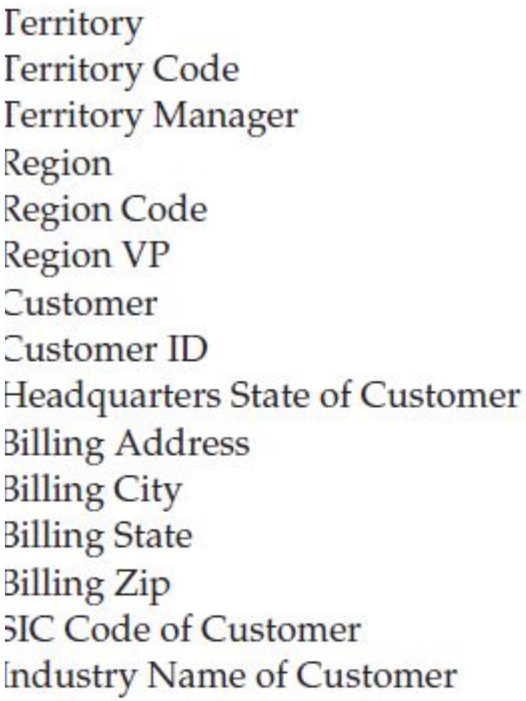


Order Dollars
Cost Dollars
Quantity Ordered

Dimensions



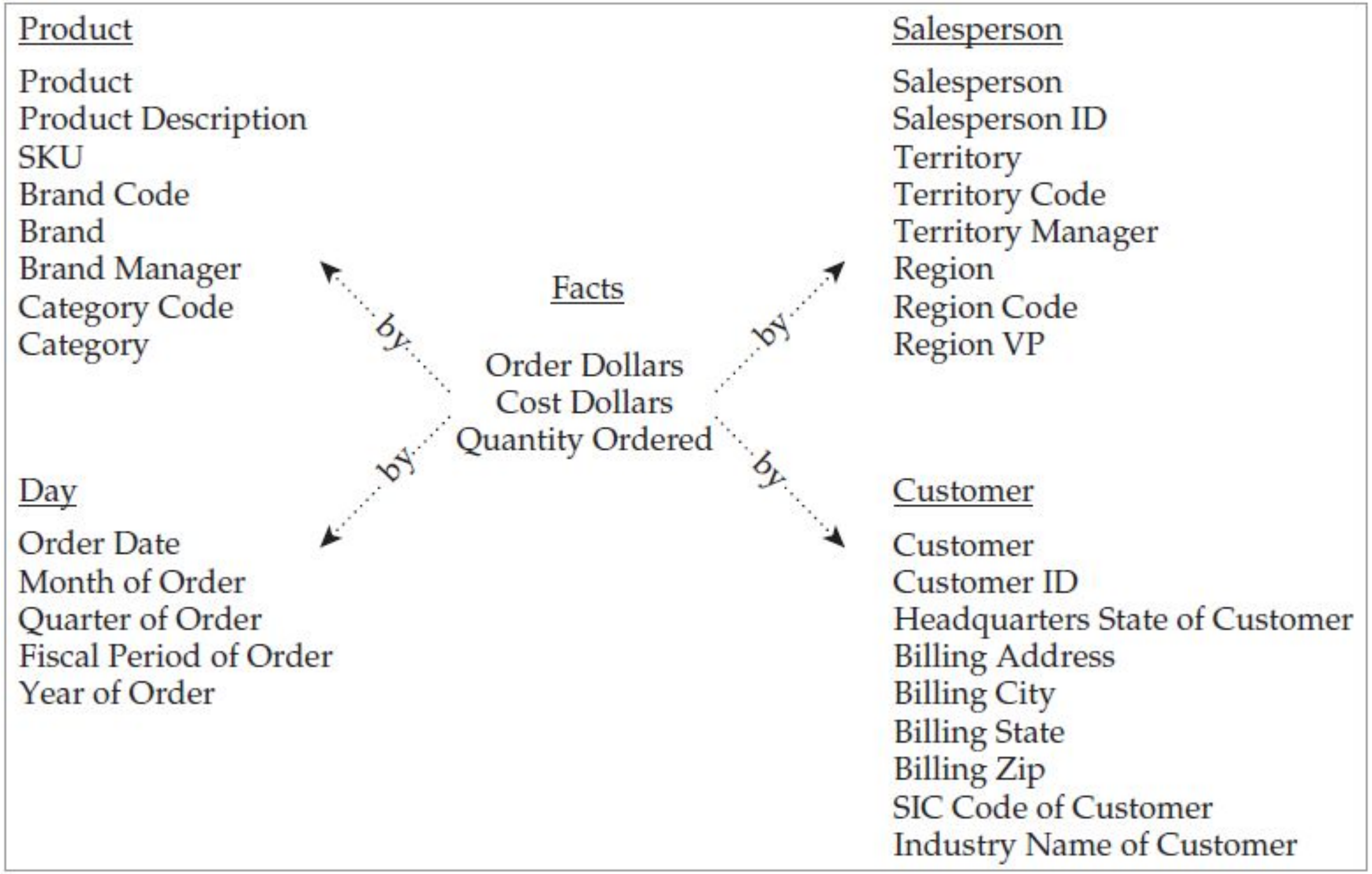
Product
Product Description
SKU
Brand Code
Brand
Brand Manager
Category Code
Category
Order Date
Month of Order
Quarter of Order
Fiscal Period of Order
Year of Order
Salesperson
Salesperson ID



Territory
Territory Code
Territory Manager
Region
Region Code
Region VP
Customer
Customer ID
Headquarters State of Customer
Billing Address
Billing City
Billing State
Billing Zip
SIC Code of Customer
Industry Name of Customer

Facts and Dimension Example: Order Process

Dimensional Modeling



Product

Product
Product Description
SKU
Brand Code
Brand
Brand Manager
Category Code
Category

Salesperson

Salesperson
Salesperson ID
Territory
Territory Code
Territory Manager
Region
Region Code
Region VP

Facts

Order Dollars
Cost Dollars
Quantity Ordered

Day

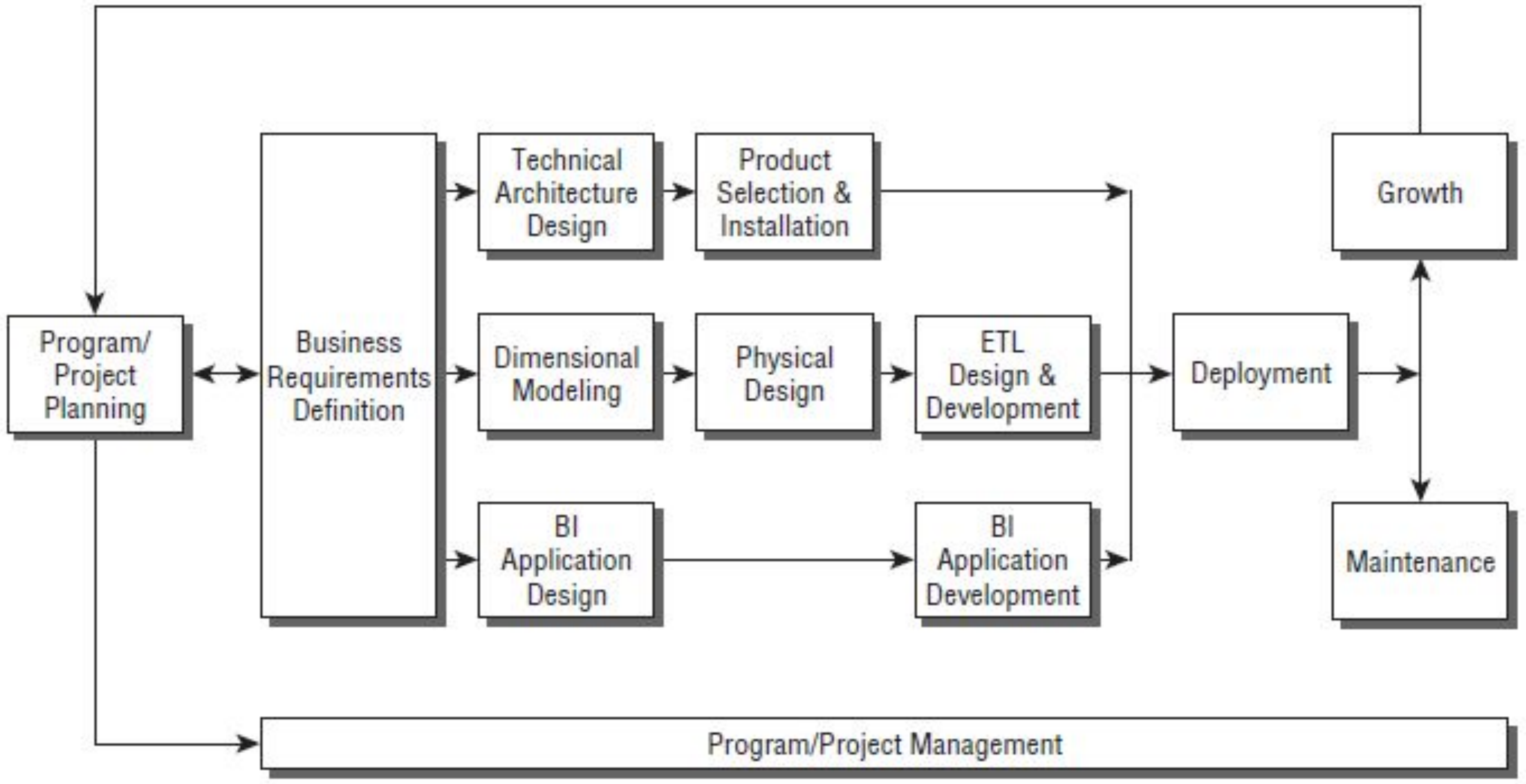
Order Date
Month of Order
Quarter of Order
Fiscal Period of Order
Year of Order

Customer

Customer
Customer ID
Headquarters State of Customer
Billing Address
Billing City
Billing State
Billing Zip
SIC Code of Customer
Industry Name of Customer

Kimball Lifecycle

Dimensional Modeling



Program/ Project Planning
Business Requirements Definition
Technical Architecture Design
Product Selection & Installation
Dimensional Modeling
Physical Design
ETL Design & Development
BI Application Design
BI Application Development
Deployment
Growth
Maintenance
Program/Project Management

WeCloudData