Lesson 2: Data Modelling

Models != Reality

- Models are ALWAYS abstractions of reality.
- The goal is to capture as much of the complexity as possible and importantly as much needed by the end users.
- There isn't usually "one model to rule them all", that is, often there is not one right answer.

Data Modeling/Models

Data Modeling - The iterative process of creating a model of data for a real-world problem.

Data Model - A representation, usually graphic, of a complex "real-world" data structure. Data models are used in the database design phase of the Database Life Cycle.

Using **input** from many perspectives and business rules, Data Modeling is the **process**, while a Data Model is the **output**.

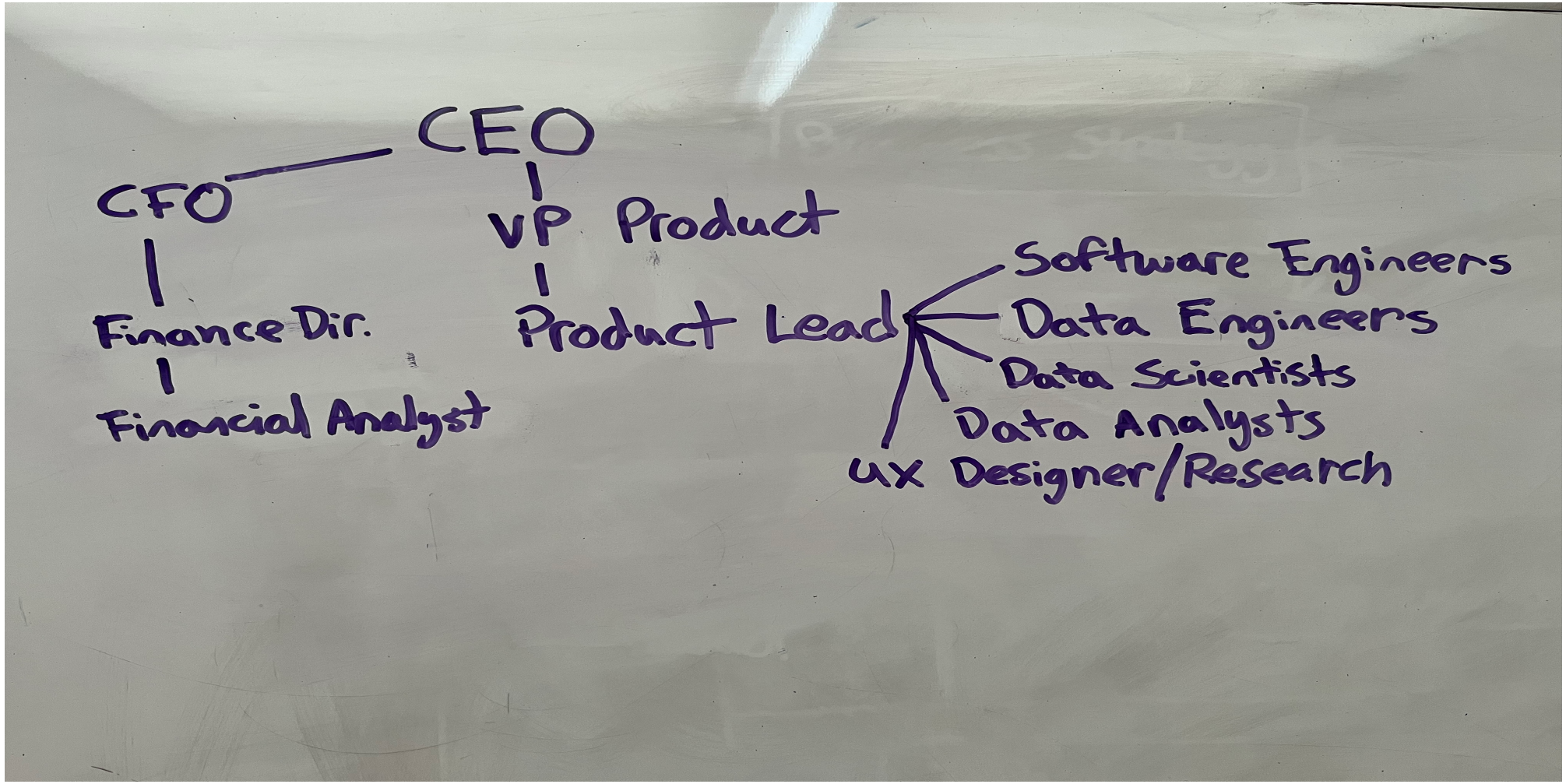
Business Rules

- A brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization. (Page 39)
- Definitions
  - Example: To my business a "year" means a "calendar year" running from January 1 through December 31.
- Or processes
  - Example: A warehouse worker scans an order, retrieves the items, packages them, attaches the order sticker(s) and sends the package to the shipping department.

Data Modeling the Process

- Data Modeling is a collaborative, cross-disciplanary and iterative exercise.
  - It should never be done in a bubble, especially just by technology teams.
  - Everyone needs to understand this.
- This is requirements gathering exercise.
  - What do stakeholders X, Y and Z need and expect?
- Part of this is a negotiation.
  - What things are "need to have" vs. "nice to have".
  - We should consider doing thing X this way

Perspectives Matters

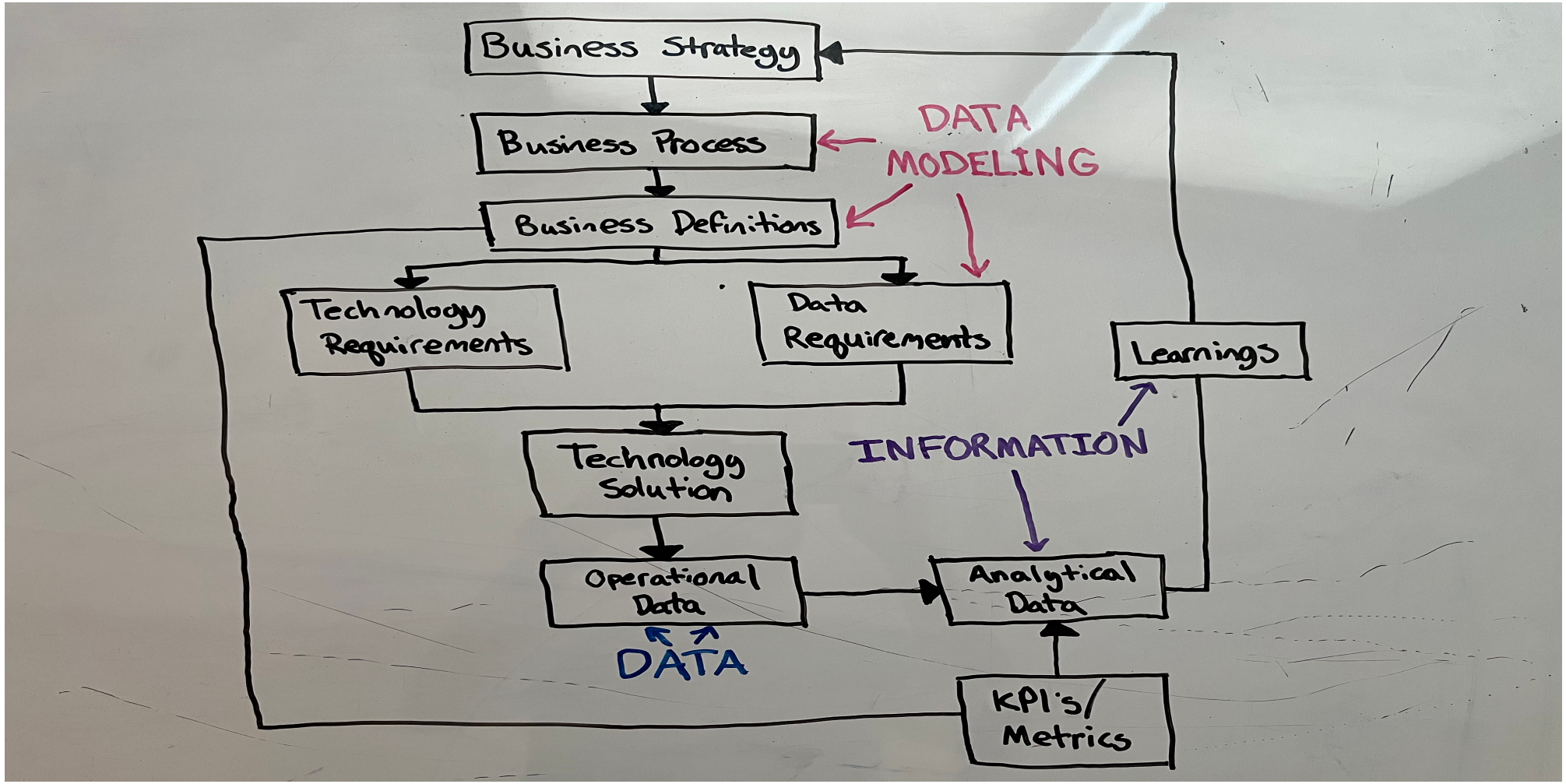


Don't Overlook Data Modeling

Data Model Failures

- It almost always becomes the problem of the technology team(s).
- Make everyones' work more difficult.
- Can cause financial loss (direct business) or legal risk.
- Causes wasteful spending on technology doesn't work.
    - Technology is not cheap!
    - Constant turnover of technology is challenging for everyone.

Putting a Few Things Together



Entity

- A person, place, thing, concept, or event for which data can be stored.
- In general an entity is a noun.
  - A **student** at SUNY Orange.
  - A **classroom** in this building.
  - The **products** a business is selling.

Entities Exercise

- On your own, come up with a few entities that you would use to create a data model of SUNY Orange. (~5 minutes)
- Let's discuss what entities you came up.
    - Is anyone of you more right than the other?

Attribute

- A characteristic of an entity or object. An attribute has a name and a data type (remember we talked about those).
- Very broadly you can think of these as adjectives that are descriptors of the entity.
    - The **color** of product A.
    - The **dimensions** of product A.
    - The **age** of employee A.

Attributes Exercise

- On your own, come up with a few entities that you would use to create a data model of SUNY Orange. (~5 minutes)
- Let's discuss what entities you came up.
    - Is anyone of you more right than the other?

Relationship

- An association between two or more entities.
- In general a relationship is a verb.
  - A degree program *offers* a class
  - A student *takes* a class
  - An instructor *teaches* a class

Types of Relationships

- **One-to-One (1:1)** - A relationship between two or more entities where one entity is associated with one and only one other entity.
- **One-to-Many (1:M)** - A relationship between two or more entities where one entity is associated with one or more other entities. You can think of many as one or more.
- **Many-to-Many (M:N or M:M)** - A relatinoship between two or more entities in which one occurrence of an entity is associated with many occurrences of a related entity and one occurrence of the related entity is associated with many occurrences of the first entity.

Relationships Exercise

- On your own, come up with a few entities that you would use to create a data model of SUNY Orange. (~5 minutes)
- What entities are related? What is the nature of this relationship?

Constraint

- Restrictions or specific logic applied, typically based on business rules.
- Constraints can be applied to entities, attributes and relationships.

Key Constraints

- Key Constraints/Refererential Integrity
  - This constraint ensures that the relationships from the data model are being upheld.
  - We will be getting further into this in the weeks to come.

Other Constraints

- Unique constraints ensure that the values within a particular column are distinct from one another and can occur once and only once.
    - Example: In a table of bank accounts, the identifier of a bank account can occur once and only once.
- A NOT NULL constraint prevents a value from having a NULL value.
    - Example: If a customer at a bank just created a new bank account, the identifier of that account cannot be NULL.

Other Constraints (Continued)

- Check constraints (applicable to attributes) check whether a certain condition has been met.
    - Example: A column holding the quantity of items sold must have a value greater than 0.
- Default constraints enable a default value to be used if one is not provided by a user.
    - Example: In a sales table, the default date of the sale is the current day (2025-01-27)

Constraints Exercise

- Continuing to build off of our SUNY Orange example, what constraints might be applicable within your data model?

Naming Conventions

- Consistent and descriptive naming is important.
  - For instance, if you named the entities in your data model "Entity_0", "Entity_1" and "Entity_2" you would eventually forget what they represented.
  - If "Entity_0" represented people then call it "people" or "person".
- Make sure to also appropriately name attributes and relationships too.
  - Let's say a product_id is used to relate to entities then name it product_id in both entities, not "product_id" in one and "id" in another.
  - In the long run, you'll be glad that you did this!

Other Important Considerations for data modeling

- While a data model is static for a period of time, they should be considered "living" because of how businesses and needs grow and change over time
- Try and use pre-existing standards if/when they're available and they meet the business needs.
    - Example: ISO 3166 defines country codes or ISO 8601 defines a standard for datetime/timestamp data.
    - These standards may function as reference data (an entity) or apply a specific format for an attribute.

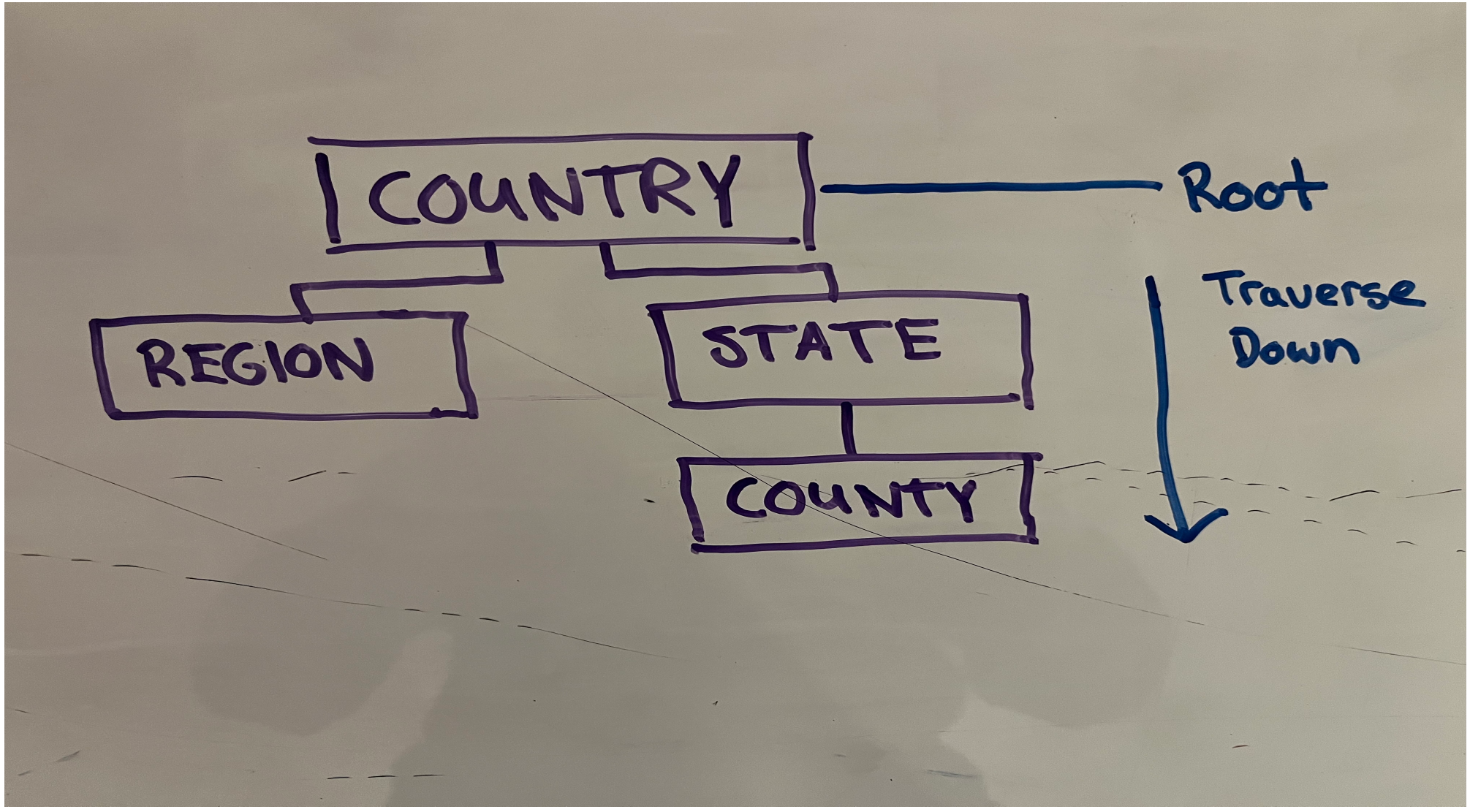
Big Data vs. Right Sized Data

- While we are in the age of big data, it doesn't mean we should default to this frame of mind.
    - The utility of data is not based on how much data you're capturing, but rather how your turning this data into information.
- Data modeling and your data systems shoud always emphasize the need to have (mission critical) data vs. nice to have data
    - Mozilla's Lean Data Practices describe this in more detail
    - Need to have data is usually easy to justify without much thought, while nice to have data is much more difficult "I might use this some day"
- Big Data systems are much costlier and require significantly more time and effort to build and maintain...so make sure it's absolutely necessary.

Hierarchical Models

- Hierarchical models use parent/child relationships.
  - A child can only belong to one parent.
  - A parent can have many children.
  - Linked by the physical arrangement of records.
  - Prone to data redundancy because of the strict parent/child relationships

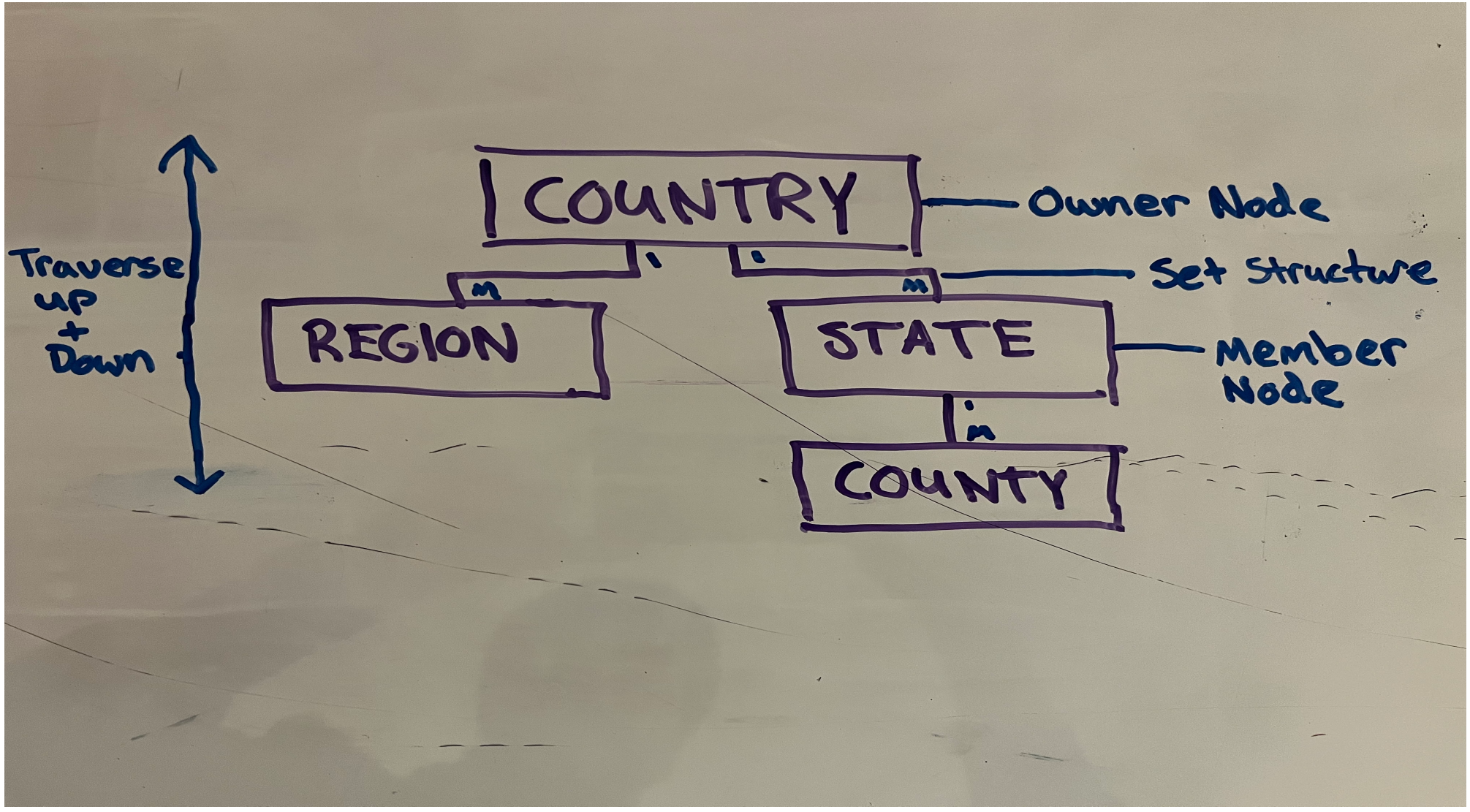
Hierarchical Model Example



Network Models

- Network models are represented by nodes and set structures
- Two key concepts that came out of this were:
  - A **schema** is the conceptual organization of the entire database.
  - The **subschema** defines the portion of the database “seen” by the application programs that actually produce the desired information from the data within the database.

Network Model Example



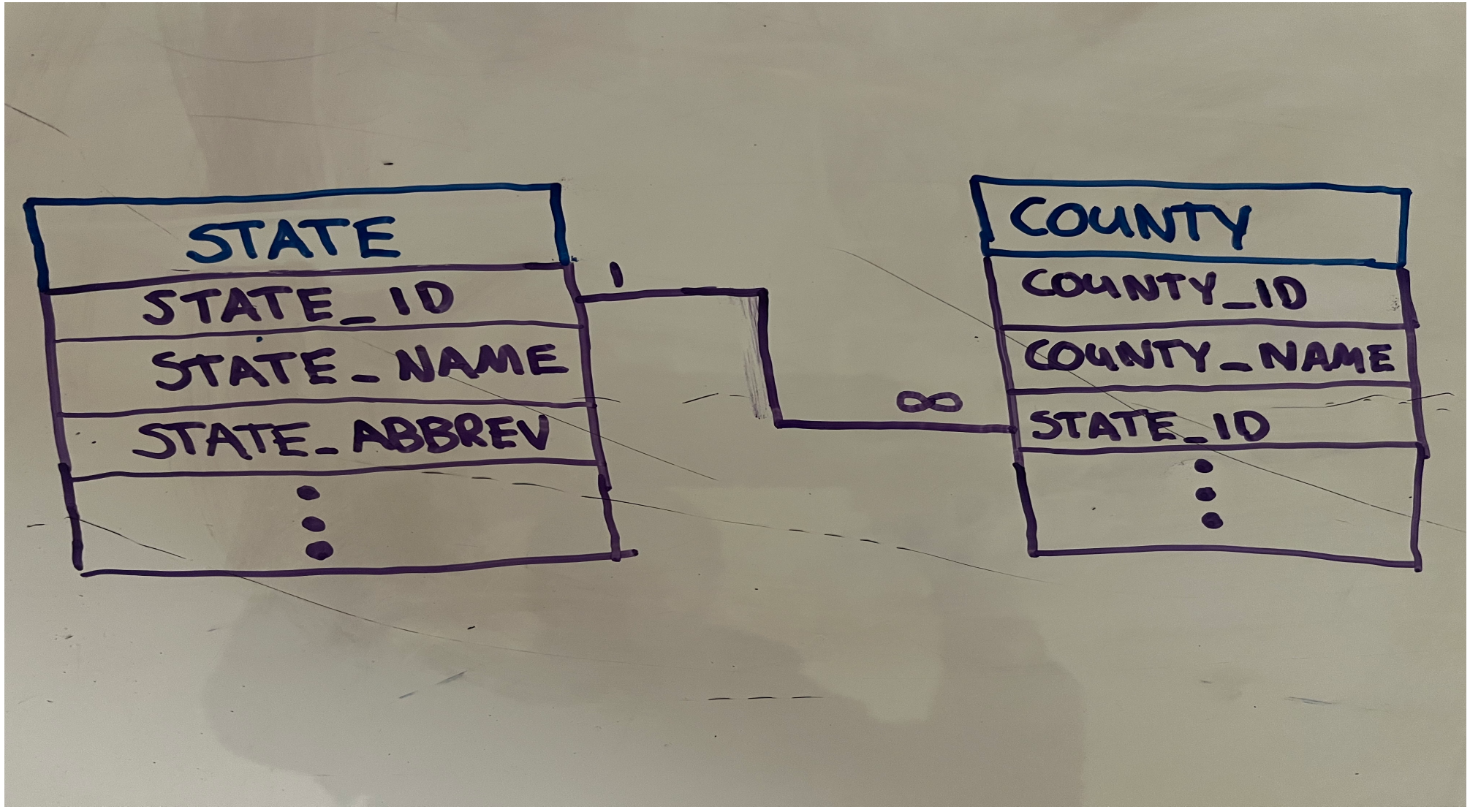
Relational Model

- Introduced in 1970 by E.F Codd
  - Orignally impractical because of the limitations of computers, but eventually led to the "database revolution"
- Yields structural independence (physical storage doesn't matter to end user)
- Implemented through a RDBMS
  - The complexities (especially the physical details) are hidden from the user.

Relational Model (Continued)

- Entity == Table
- Tuple == Row
- Attribute == Column

Relational Diagrams



The Three Parts of an RDBMS

1. User Interface
    - Varies a bit based on the RDBMS, but always provides the ability to write and execute SQL.
2. A collection of tables that are stored within a database.
    - Tables present data in rows and columns, so it's easy to understand.
3. A SQL Engine
    - This translates what we write in SQL to acesss data or manipulate tables into detailed instructions for the RDMS.
    - The internal workings of a RDBMS allow it to make decisions about the most efficient way to execute on these instructions

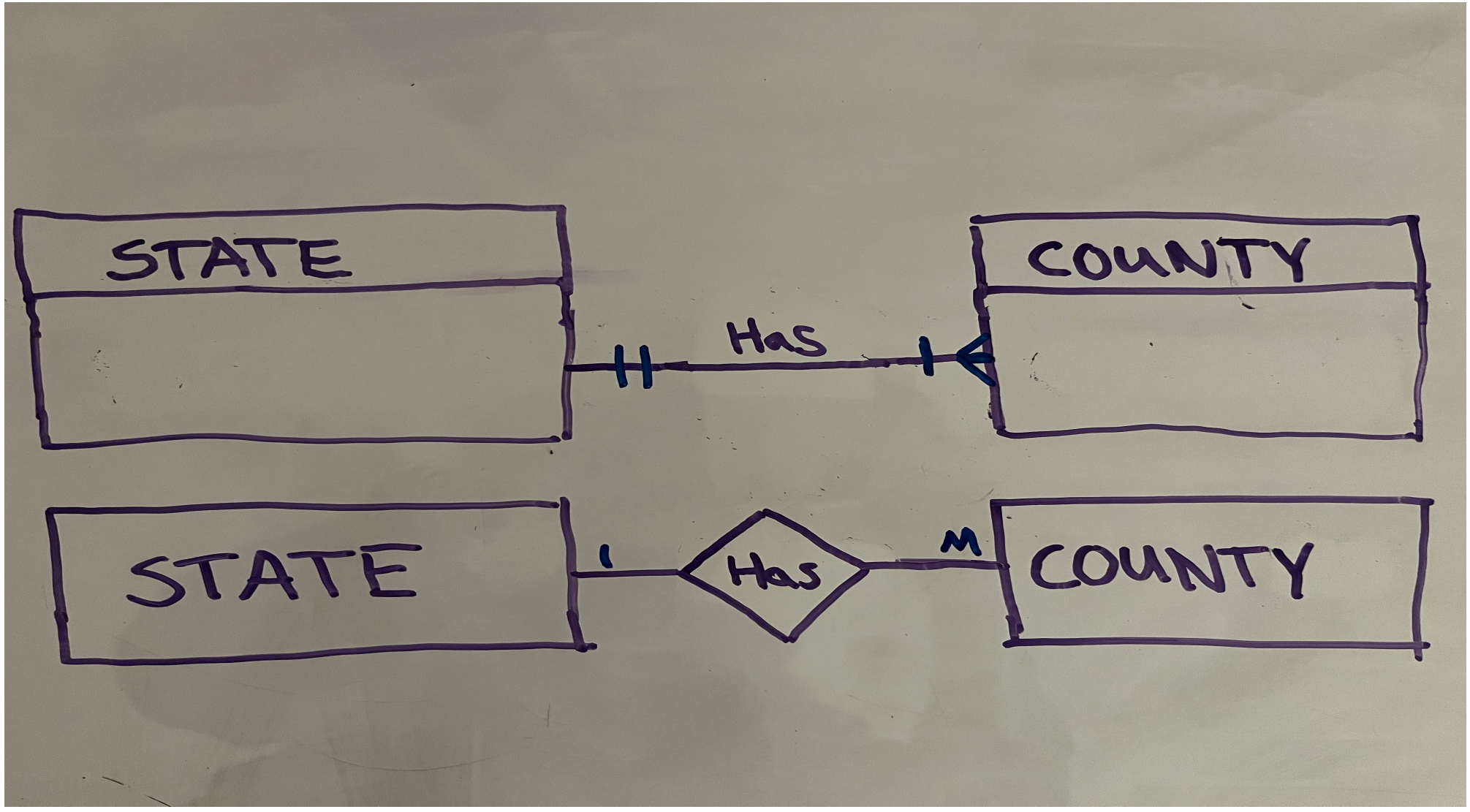
The Entity Relationship Model

- Instances and Occurences
  - These represent the an individual row of data.
- Entity Set
  - Often just referred as "entity" represent the collection of rows aka a table.
- Connectivity
  - Also known as the type of relationship between entities.
- Commonly represented Entity Relationship Diagram (ERD)

The Entity Relationship Model (ERM)

- Entity Set == Table
- Entity Instance/Occurence == Row

Entity Relationship Diagram Notation



Semantic Data Models and Beyond

- Semantic Data Models we aim to understand each piece of data within the context of its meaning. Each row represents something and can be mapped to additional attributes.
- NoSQL focuses on Big Data and the storage and processing of structured and unstructured data.

The 3 V's

- Velocity - how fast data is being consumed and is needed to be turned into information.
- Variety - the fact that data comes from multiple different sources, not all of which can follow the same model
- Volume - how much data is being captured and stored.

Degrees of Data Abstraction

External Model

- An end users view of the data.
  - Represented by the external schema(s).
  - This view is only what matters to them and their part of the business.
  - The view of different end-users will be different, so there isn't one external model.

Conceptual Model

- The macro-level view of a database for use by the entire organization.
  - It provides a high-level description of the main database objects without getting too in depth.
  - It accounts for all of the external views.
  - Graphically represented by the ERD.
- A conceptual model is both hardware and software agnostic.
  - The conceptual model could be applied using any DBMS.

Internal Model

- The representation of the database as seen by the DBMS itself.
  - Matching the conceptual model to the chosen implementation.
  - We'll be solely focused on the relational model for this class.

```
CREATE TABLE IF NOT EXISTS citclass.students (first_name VARCHAR(255)
                                              ,last_name VARCHAR(255)
                                              ,id INT);
```

Physical Model

- How the data will be physically stored and accessed.
  - It is not hardware or software independent.
  - THe relational model minimizes the need to worry about physical storage and focus more on the fine-tuning.
  - For the purposes of this class it's not something we will dive into further.