# Data Science & Analytics

Hiren Deliwala & Dr. Jian ZHANG

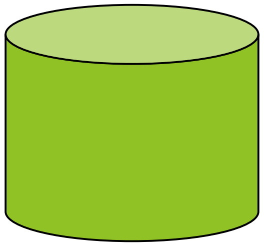
## Agenda

- Analytics Theory
  - Information Management
  - Statistics
- Data Modeling
  - Introduction
  - Key Concepts
  - How to build a data model?

## Business Intelligence

Business Intelligence enables the business to make intelligent, fact-based decisions

Aggregate Data → Present Data → Enrich Data → Inform a Decision



Database, Data Mart, Data Warehouse, ETL Tools, Integration Tools

Reporting Tools, Dashboards, Static Reports, Mobile Reporting, OLAP Cubes

Add Context to Create Information, Descriptive Statistics, Benchmarks, Variance to Plan or LY

Decisions are Fact-based and Data-driven

## Analytics - Four Key Disciplines

1. Information Management
2. Statistics
3. Information Delivery
4. High Performance Computing

## Information Management

- Aggregating, standardizing, restructure, and preparing the rising quantities of data for analysis
- What is it?
- Where is it?
- How good is it?
- Is there enough of it?
- Is it ready for analysis?

## Information Management

1. Data Governance
2. Data Provisioning
3. Data Aggregation
4. Data Enrichment
5. Data Structuring
6. Data Quality
7. Data Integration

## Statistics

- Descriptive Statistics – describe how things are
  - E.g. Revenue last quarter, patients admitted this year, Retail spending this year, Readmission rates
  - Answers “What” questions
  - Very pervasive
- Inferential Statistics – Explore relationships between concepts and draw inferences from what we observe
  - Answers “Why” and “How” questions
  - E.g. Predictive Analytics, Scoring, Forecasting

# Statistics

|  |  |  |
|---|---|---|
| Models | Predictive Analytics | Scoring |
| Forecasting | Descriptive Modeling | Data Mining |
| Optimization | Simulation | Data Visualization |

## Data Modeling

## What is a Data Model?

- Definition: precise description of the data content in a system

- Types of data models:
  1. Conceptual: describes WHAT the system contains
  2. Logical: describes HOW the system will be implemented, regardless of the DBMS
  3. Physical: describes HOW the system will be implemented using a specific DBMS

## Why do we need to create data models?

- To aid in the development of a sound database design that does not allow anomalies or inconsistencies

- Goal:  to create database tables that do not contain duplicate data values that can become inconsistent

## Types of Data Models

- Entity-Relationship (E-R) Models
  - Only addresses data and relationships
  - Classic, simplest
    - Best for deriving a sound table design
  - Many extensions/variations exist
  - Basis for most other modeling approaches
  - Is DBMS and hardware independent
  - Supports the user's perception of the data
- UML (unified modeling language)
  - Class models
  - Goes beyond data, also models behaviors

# ER Modeling

## Data Model Building Blocks

| | | |
|---|---|---|
| Entities | Attributes | Keys |
| Relationships | Cardinality | Optionality |

## Data Entities

- Entity
  - A "thing" about which you want to store data in an application
  - Multiple examples (instances) of the entity must exist
  - Goal:
    - Store data about each entity in a separate table
    - Do not store duplicate data in multiple tables or records
  - Examples:

Customer

Product

Employee

# ER Model Attributes

- Attribute
  - A characteristic (data field) of an entity that you want to store in the database
  - Synonyms include *element*, *property*, and *field*.
    - Examples: CUST_ID, PROD_DESC
- Attribute value
  - The value of a particular attribute for a particular entity instance
  - Examples: 42, "Nuts Not Nachos"

Student
- Last Name
- First Name
- Middle Initial
- Street
- City
- State
- Country
- Postal Code
- Home Phone Number
- Mobile Phone Number
- Date of Birth
- Gender
- Race
- Major
- Grade Point Average

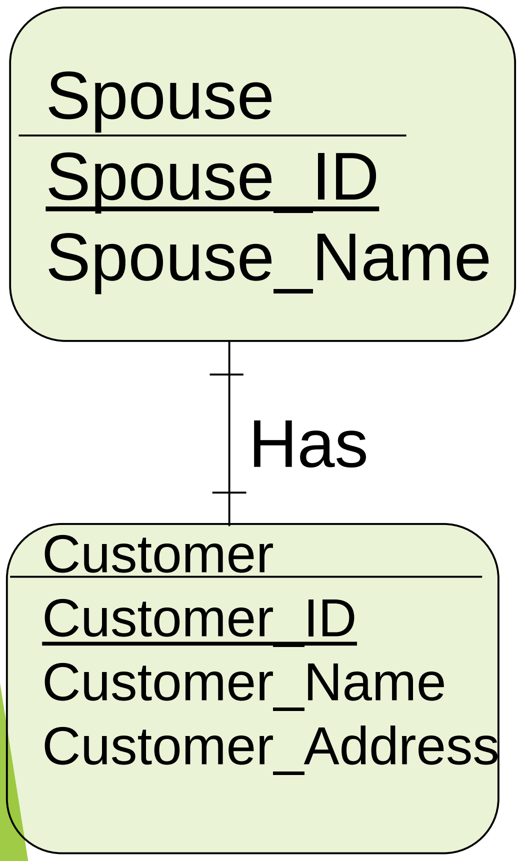
## Keys

- A key is a data item that allows us to uniquely identify individual occurrences or an entity type.
- A candidate key is an attribute or set of attributes that uniquely identifies individual occurrences or an entity type.
- An entity type may have one or more possible candidate keys, the one which is selected is known as the primary key.
- A composite key is a candidate key that consists of two or more attributes
- The name of each primary key attribute is underlined.

# Relationships

- Conceptually, entities and attributes do not exist in isolation.
- Relationships specifies the number of instances of one entity that can be associated with instances of a related entity
- A relationship is a natural business association that exists between one or more entities. The relationship may represent an event that links the entities, or merely a logical affinity that exists between the entities.
- Types:
  - 1:M
  - 1:1
  - M:M
  - “M” denotes some value greater than 1 whose upper bound is undetermined
- On an ER diagram, if the end of a relationship is straight, it represents 1, while a "crow’s foot" end represents many.

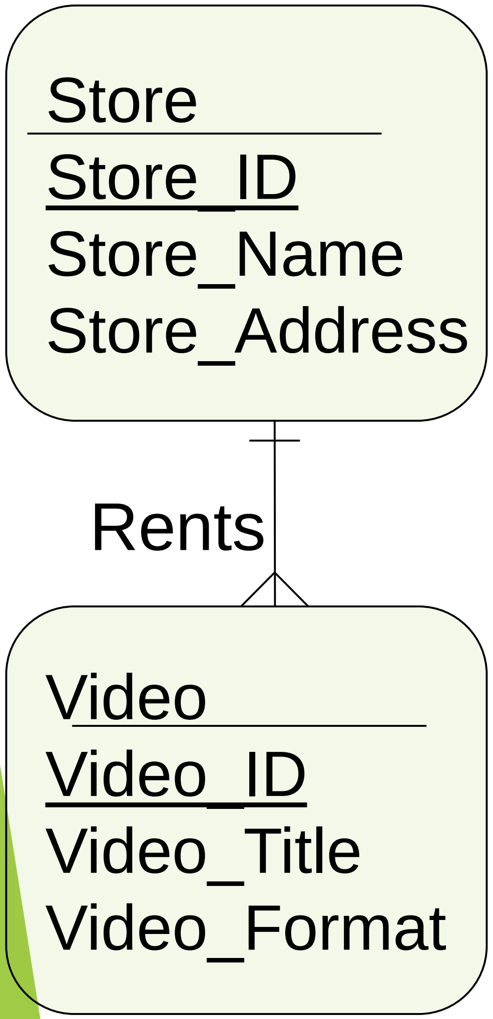
# Example 1:1 Relationship



| Spouse_ID | Spouse_Name |
|---|---|
| 52 | Ryan, Judy |
| 53 | Redmann, Rudy |

| Customer_ ID | Customer_ Name | Customer_Address |
|---|---|---|
| 1 | Ryan, Paul | 5454 Hyde Court |
| 2 | Myers, Mary | 112 Birch Place |

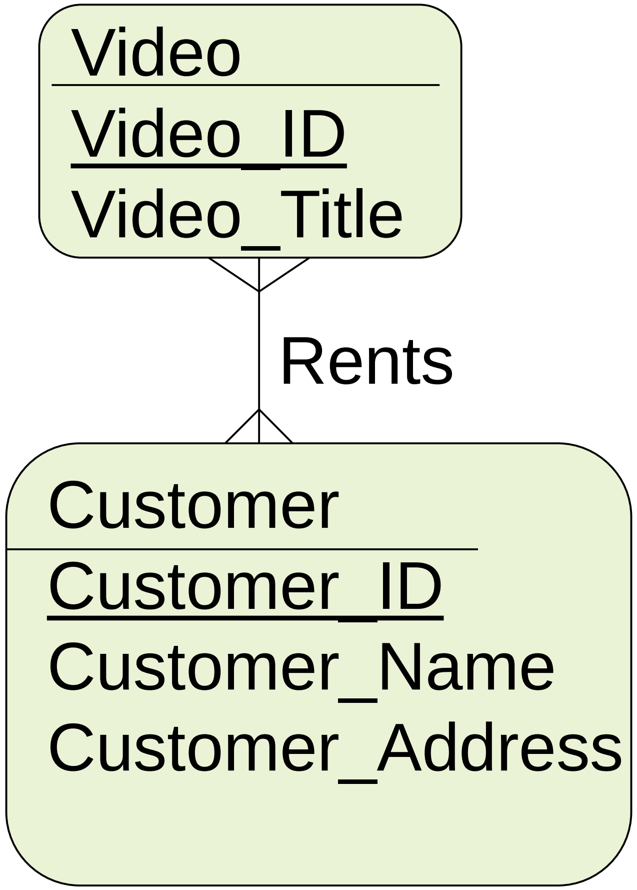
# Example 1:M Relationship



| Store_ID | Store_Name | Store_Address |
|---|---|---|
| 1 | Northside | 3233 Wisconsin St. |
| 2 | Southside | 4211 Golf Road |

| Video_ID | Video_Title | Video_Format |
|---|---|---|
| 1000 | The Princess Bride | DVD |
| 1001 | Sideways | Bluray |
| 1002 | Just Visiting | DVD |
| 1003 | Crash | Bluray |

## Example M:M Relationship



# Cardinality Notations

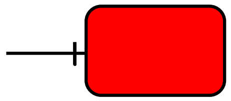
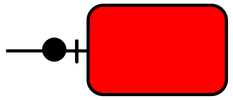
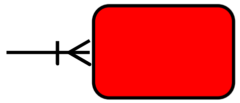
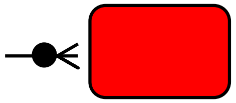
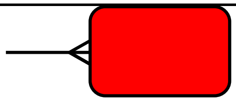
| Cardinality Interpretation | Minimum Instances | Maximum Instances | Graphic Notation |
|---|---|---|---|
| Exactly one | 1 | 1 |  |
| Zero or one | 0 | 1 |  |
| One or more | 1 | many ( > 1 ) |  |
| Zero, one, or more | 0 | many ( > 1 ) |  |
| More than one | > 1 | > 1 |  |

**Figure 5.3**

## Optionality

A relationship can be optional or mandatory.

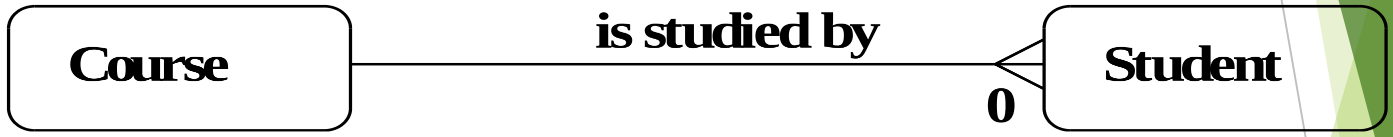
- If the relationship is mandatory
  - an entity at one end of the relationship must be related to an entity at the other end.
- If the relationship is optional
  - Entities at each end of the relationship may or may not have to relate to each other.

## Optionality cont…

- The optionality can be different at each end of the relationship
  - For example, a student must be on a course. This is mandatory. To the relationship ‘student studies course’ is mandatory.
  - But a course can exist before any students have enrolled. Thus the relationship ‘course is_studied_by student’ is optional.
- To show optionality, put a circle or ‘0’ at the ‘optional end’ of the relationship.

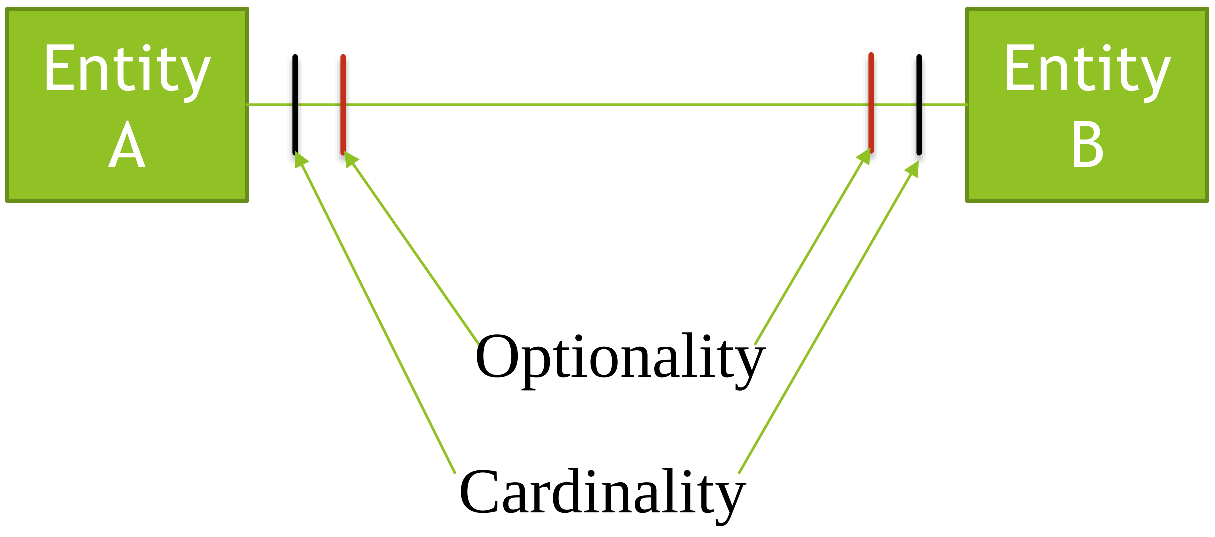
## Optionality cont…

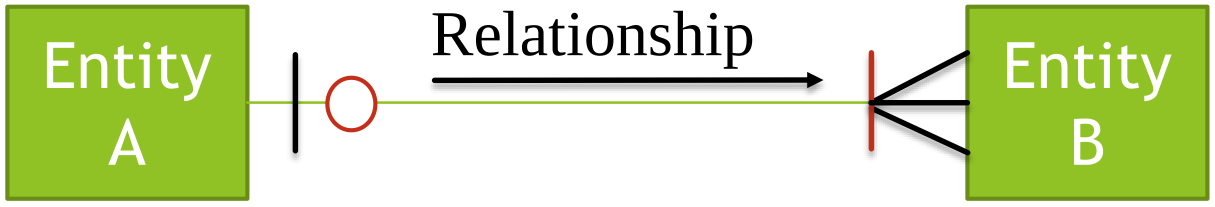
- As the optional relationship is ‘course is_studied_by student’, and the optional part of this is the student, then the ‘O’ goes at the student end of the relationship connection.



- It is important to know the optionality because you must ensure that whenever you create a new entity it has the required mandatory links.

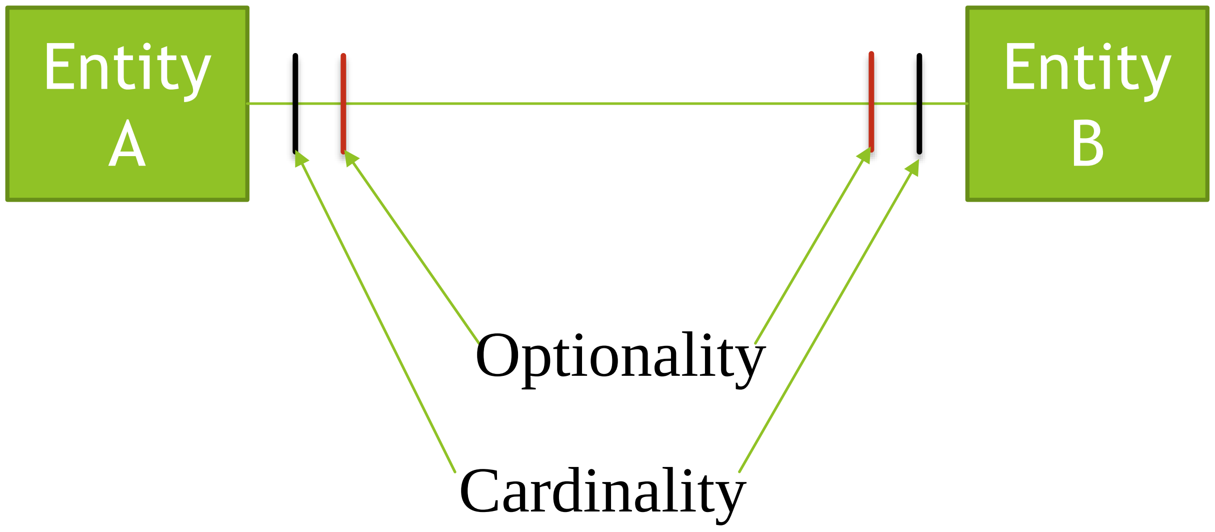
# Optionality & Cardinality 1:N

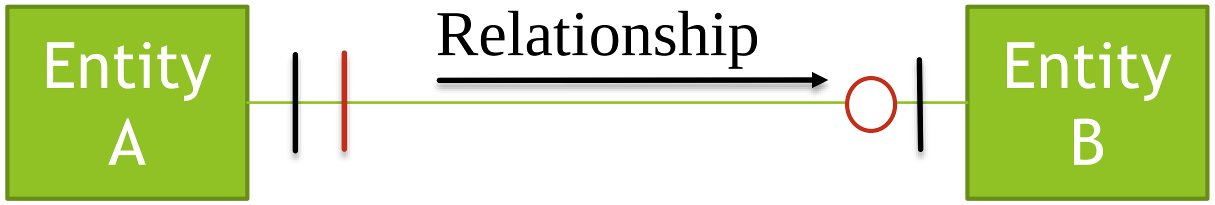




Relationship is 1:N

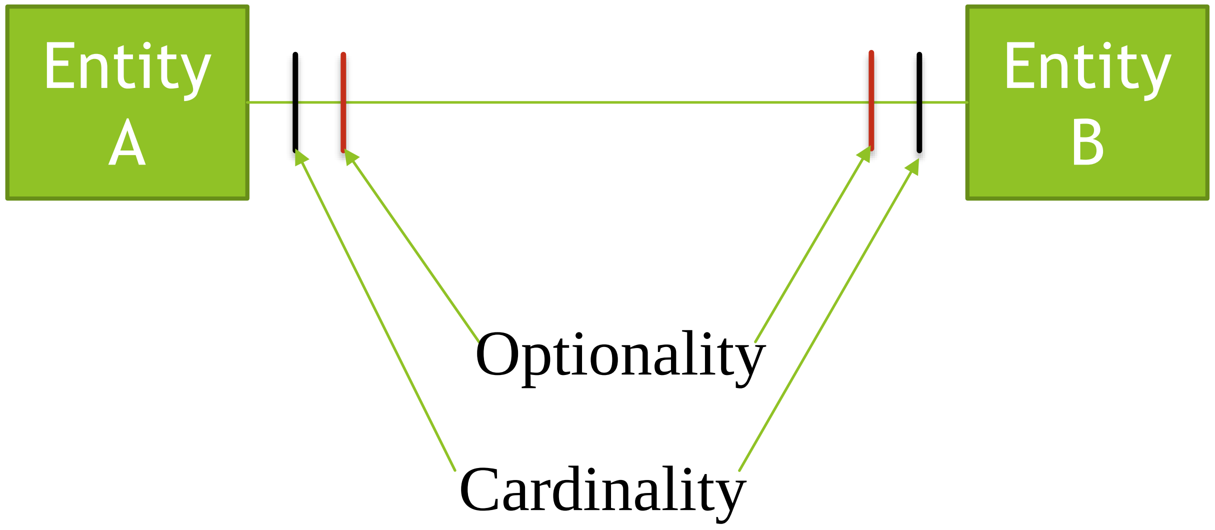
# Optionality & Cardinality 1:1

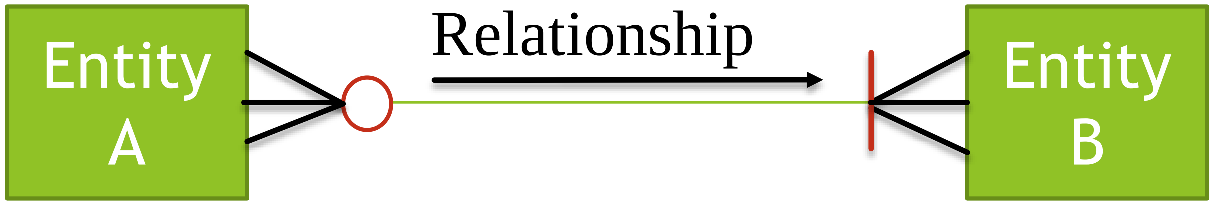




Relationship is 1:1

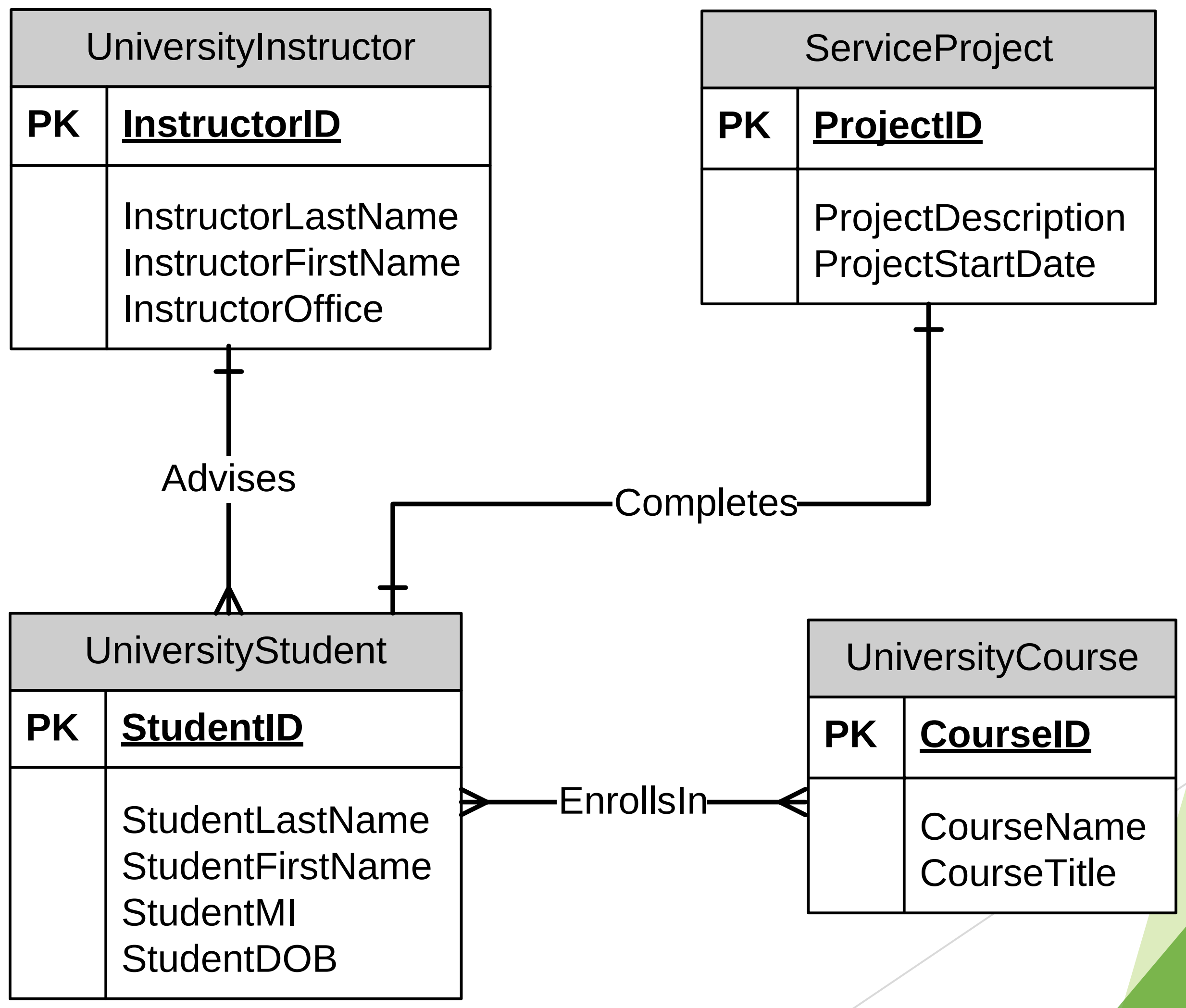
# Optionality & Cardinality N:N





Relationship is N:N

# Example ER Model



## Recursive Relationships

- It is possible to have a n-ary relationship (e.g. quaternary or unary).
- Unary relationships are also known as a *recursive* relationship.
- It is a relationship where the same entity participates more than once in different roles.
- In the example above we are saying that employees are managed by employees.
- If we wanted more information about who manages whom, we could introduce a second entity type called manager.

## Deriving the relationship parameters

To check we have the correct parameters (sometimes also known as the degree) of a relationship, ask two questions:

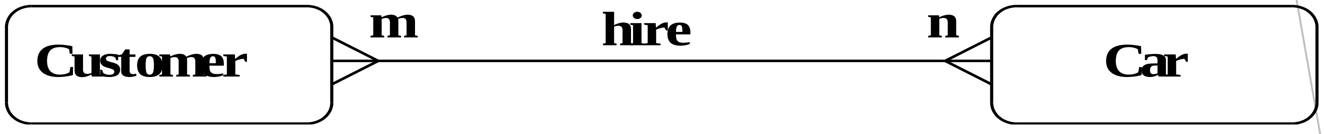
- One course is studied by how many students? Answer => ‘zero or more’.
  - This gives us the degree at the ‘student’ end.
  - The answer ‘zero or more’ needs to be split.
    - The ‘more’ means that the cardinality is ‘many’.
    - The ‘zero’ means that the relationship is ‘optional’.
  - If the answer was ‘one or more’, then the relationship would be ‘mandatory’.

# Relationship parameters cont...

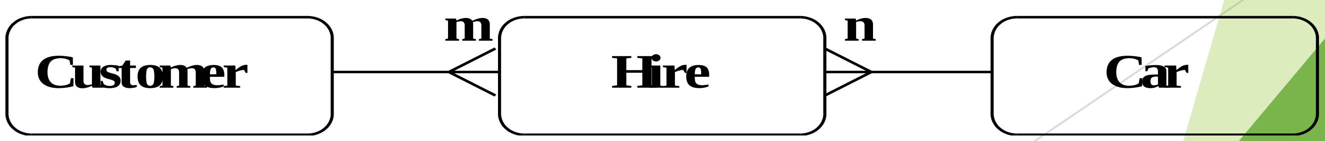
- One student studies how many courses? Answer => ‘One’
  - This gives us the degree at the ‘course’ end of the relationship.
    - The answer ‘one’ means that the cardinality of this relationship is 1, and is ‘mandatory’
  - If the answer had been ‘zero or one’, then the cardinality of the relationship would have been 1, and be ‘optional’.

## Splitting n:m Relationships - example

Consider the case of a car hire company. Customers hire cars, one customer hires many cars and a car is hired by many customers.



The many to many relationship can be broken down to reveal a ‘hire’ entity, which contains an attribute ‘date of hire’.



## How to Build Data Models

## Building Data Model

- Entity Discovery
- Relationships
- Cardinality & Optionality
- Context Data Model
- Key Based Data Model
- Fully Attributed Entities

## How to Construct Data Models

### Entity Discovery

- Discover those fundamental entities in the system that are or might be described by data.
- There are several techniques that may be used to identify entities.
  - During interviews or Joint Design sessions with system owners and users, pay attention to key words in their discussion.
  - During interviews or JAD sessions, specifically ask the system owners and users to identify things about which they would like to capture, store, and produce information.

## How to Construct Data Models

### Entity Discovery

- A true entity has multiple instances—dozens, hundreds, thousands, or more!
- Entities should be named with *nouns* that describe the person, event, place, or tangible thing about which we want to store data.
  - Try not to abbreviate or use acronyms.
  - Names should be singular so as to distinguish the logical concept of the entity from the actual instances of the entity.
- Define each entity in business terms.
  - Don’t define the entity in technical terms, and don’t define it as ‘data about …’.
  - Your entity names and definitions should establish an initial glossary of business terminology that will serve both you and future analysts and users for years to come.

## How to Construct Data Models

**Define the relationships**

- Examine each entity type to see its relationship to the others.

**Describe the cardinality and optionality of the relationships**

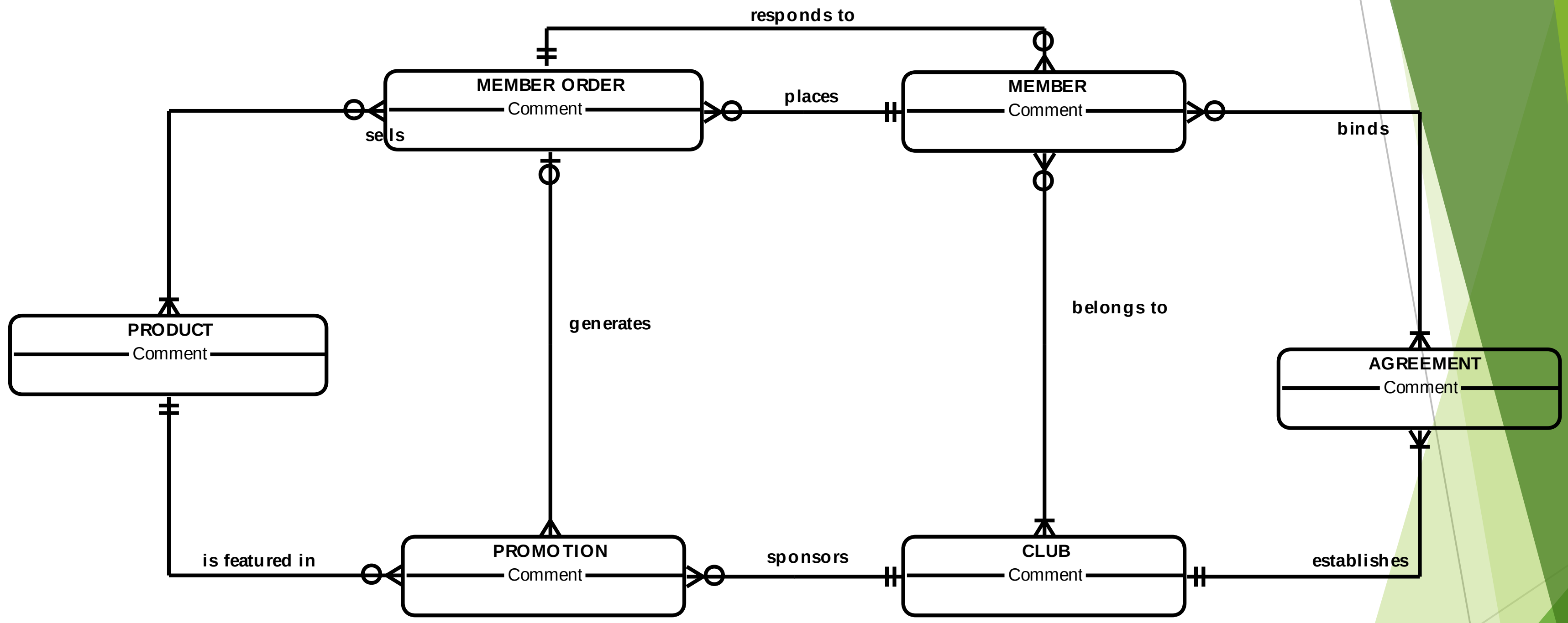
- Examine the constraints between participating entities.

# How to Construct Data Models

## The Context Data Model

- The second task in data modeling is to construct the context data model.
  - The context data model includes the fundamental or *independent* entities that were previously discovered.
    - An **independent entity** is one which exists regardless of the existence of any other entity. Its primary key contain no attributes that would make it dependent on the existence of another entity.
    - Independent entities are almost always the first entities discovered in your conversations with the users.
  - Relationships should be named with verb phrases that, when combined with the entity names, form simple business sentences or assertions.
    - Always name the relationship from parent-to-child.

# A Context Data Model



## How to Construct Data Models

**The Key-Based Data Model**

- The third task is to identify the keys of each entity.
- The following guidelines are suggested for keys:
    - The value of a key should not change over the lifetime of each entity instance.
    - The value of a key cannot be null.
    - Controls must be installed to ensure that the value of a key is valid.

## How to Construct Data Models

### The Key-Based Data Model

- Some experts suggest that you *avoid* **intelligent keys** because the key may change over the lifetime of the entity instance.
  - An intelligent key is a business code whose structure communicates data about an entity instance (such as its classification, size, or other properties).
  - A code is a group of characters and/or digits that identifies and describes something in the business system.
- Other experts suggest that you *use* **intelligent keys** because business codes can return value to the organization because they can be quickly processed by humans without the assistance of a computer.

## How to Construct Data Models

### The Key-Based Data Model

- Consider inventing a surrogate key instead to substitute for large concatenated keys of independent entities.
  - This suggestion is not practical for associative entities since because each part of the concatenated key is a foreign key that must precisely match its parent entity’s primary key.
- If you cannot define keys for an entity, it may be that the entity doesn’t really exist—that is, multiple occurrences of the so-called entity do not exist.

## How to Construct Data Models

### The Key-Based Data Model

- Business Codes
  - There are several types of codes and they can be combined to form effective means for entity instance identification.
    - **Serial codes** assign sequentially generated numbers to entity instances.
      - Many database management systems can generate and constrain serial codes to a business' requirements.
    - **Block codes** are similar to serial codes except that serial numbers are divided into groups that have some business meaning.
    - **Alphabetic codes** use finite combinations of letters (and possibly numbers) to describe entity instances.
      - Alphabetic codes must usually be combined with serial or block codes in order to uniquely identify instances of most entities.

## How to Construct Data Models

### The Key-Based Data Model

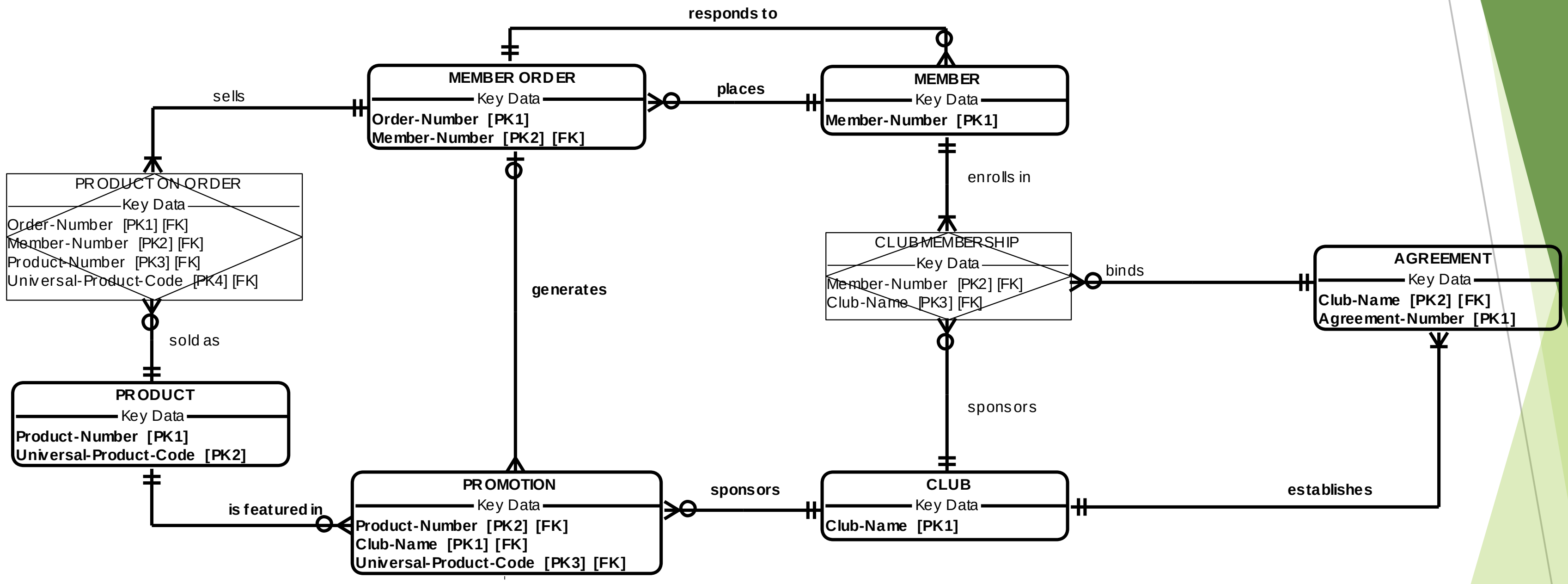
- Business Codes
  - There are several types of codes and they can be combined to form effective means for entity instance identification. (continued)
    - In **significant position codes**, each digit or group of digits describes a measurable or identifiable characteristic of the entity instance.
      - Significant digit codes are frequently used to code inventory items.
    - **Hierarchical codes** provide a top-down interpretation for an entity instance.
      - Every item coded is factored into groups, subgroups, and so forth.

## How to Construct Data Models

### The Key-Based Data Model

- Business Codes
  - The following guidelines are suggested when creating a business coding scheme:
    - Codes should be expandable to accommodate growth.
    - The full code must result in a unique value for each entity instance.
    - Codes should be large enough to describe the distinguishing characteristics, but small enough to be interpreted by people *without a computer*.
    - Codes should be convenient. A new instance should be easy to create.

# A Key Based Data Model



## How to Construct Data Models

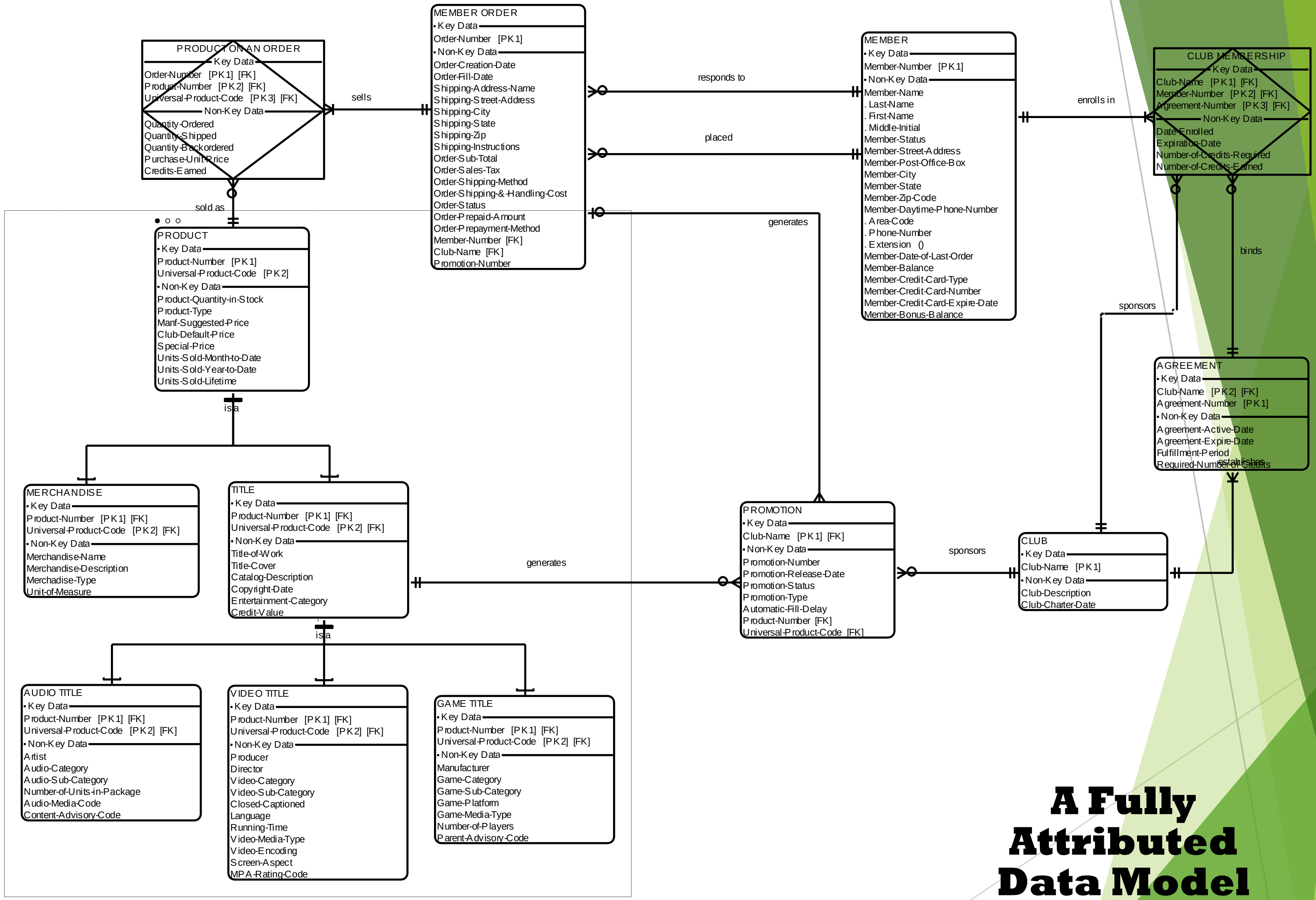
### The Fully Attributed Data Model

- The fifth task is to identify the remaining data attributes.
  - The following guidelines are offered for attribution.
    - Many organizations have naming standards and approved abbreviations.
      - The data or repository administrator usually maintains such standards.
    - Many attributes share common base names such as NAME, ADDRESS, DATE.
      - Unless the attributes can be generalized into a supertype, it is best to give each variation a unique name such as:
        CUSTOMER NAME vs SUPPLIER NAME
      - Names must be distinguishable across projects.
    - Logical attribute names should not be abbreviated.

# How to Construct Data Models

## The Fully Attributed Data Model

- For attributes that have only YES or NO values, name as questions.
  - For example, CANDIDATE FOR A DEGREE?
- Each attribute should be mapped to only one entity.
  - Foreign keys are the exception – they identify associated instances of related entities.
- An attribute’s domain should not be based on logic.



# A Fully Attributed Data Model

## How to Construct Data Models

### The Fully Described Model

- The last task is to fully describe the data model.
  - This task is the most time consuming.
  - This task can be started in parallel with the key-based model or fully attributed model, but it is usually the last data modeling task completed.
  - At this time the descriptions for the attributes are still incomplete – they require domains.
    - Most CASE tools provide extensive facilities for describing the data types, domains, and defaults for all attributes to the repository.

## How to Construct Data Models

### The Fully Described Model

- Additional descriptive properties may be recorded for attributes such as:
  - Who should be able to create, delete, update, and access each attribute?
  - How long should each attribute (or entity) be kept before the data is deleted or archived?

## ER Modelling Process

- ER modelling is iterative, so expect to draw several versions. Note that there is no one right answer to the problem, but some solutions are better than others!

## Drone Bus Company

A Drone Bus Company owns a number of busses. Each bus is allocated to a particular route, although some routes may have several busses. Each route passes through a number of towns. One or more drivers are allocated to each stage of a route, which corresponds to a journey through some or all of the towns on a route. Some of the towns have a garage where busses are kept and each of the busses are identified by the registration number and can carry different numbers of passengers, since the vehicles vary in size and can be single or double-decked. Each route is identified by a route number and information is available on the average number of passengers carried per day for each route. Drivers have an employee number, name, address, and sometimes a telephone number.

## Drone Bus Company

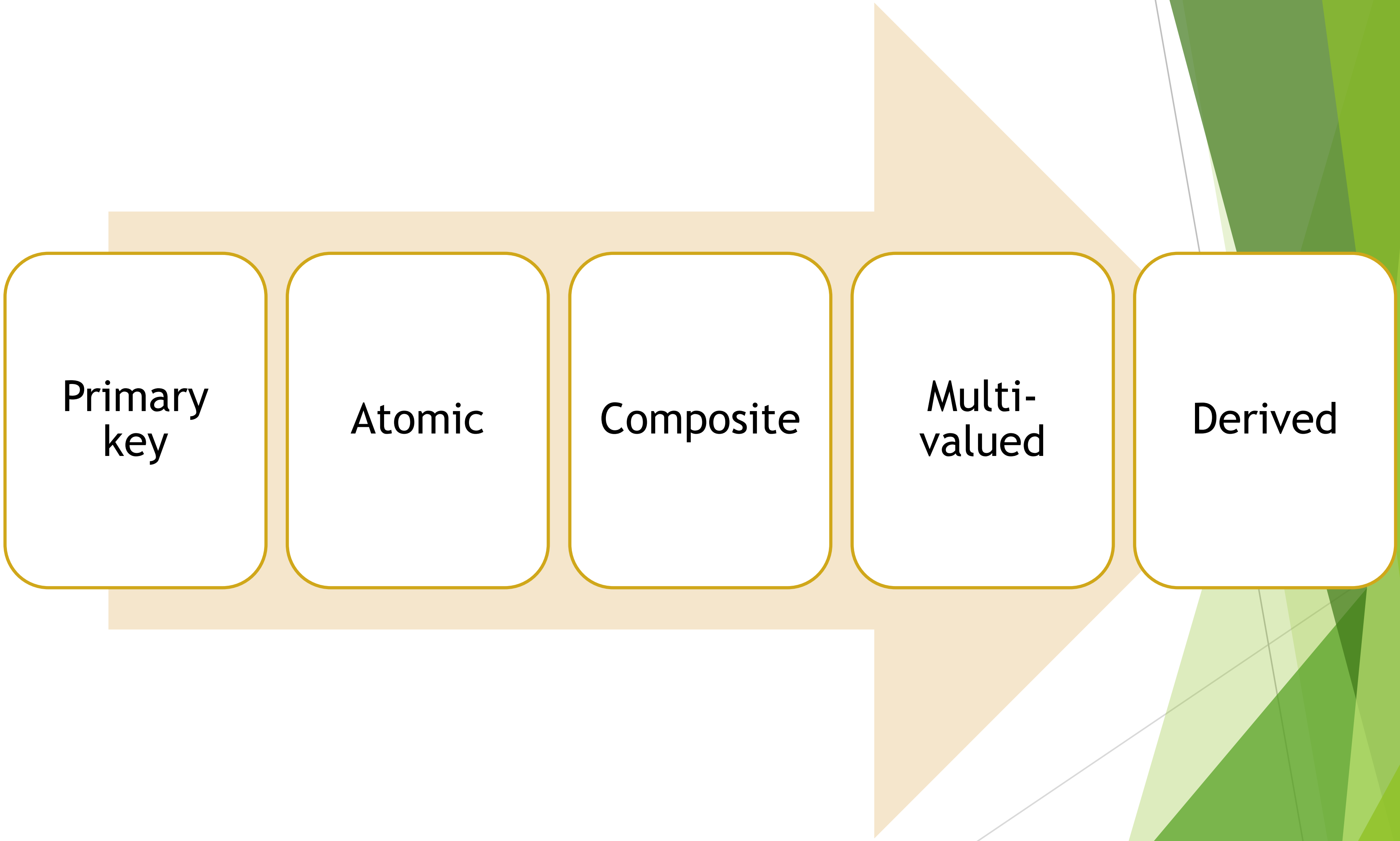
- Entity Discovery
- Relationships
- Cardinality & Optionality
- Context Data Model
- Key Based Data Model
- Fully Attributed Entities

## Veggie Burger World Distribution Center

Veggie-Burger World Distribution Center serves as a supplier to 45 Veggie-Burger World franchises. You are involved with a project to build a database system for distribution. Each franchise submits a day-by-day projection of sales for each of Burger World's menu products - the products listed on the menu at each restaurant - for the coming month. All menu product require ingredients and/or packaging items. Based on projected sales for the store, the system must generate a day-by-day and ingredients need and then collapse those needs into one-per-week purchase requisitions and shipments.

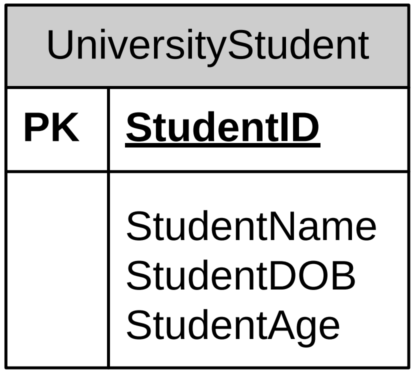
## Reference - More about Attributes

## Attributes Selection Issues



## Primary Key Attributes

- Attribute whose value is unique for every entity instance
  - Every entity MUST have a PK
  - **Designate by:**
    - Placing as first attribute in the entity
    - Underline
    - Label using "PK"



| UniversityStudent | |
|---|---|
| **PK** | <u>**StudentID**</u> |
| | StudentName<br>StudentDOB<br>StudentAge |

# Selecting Primary Keys

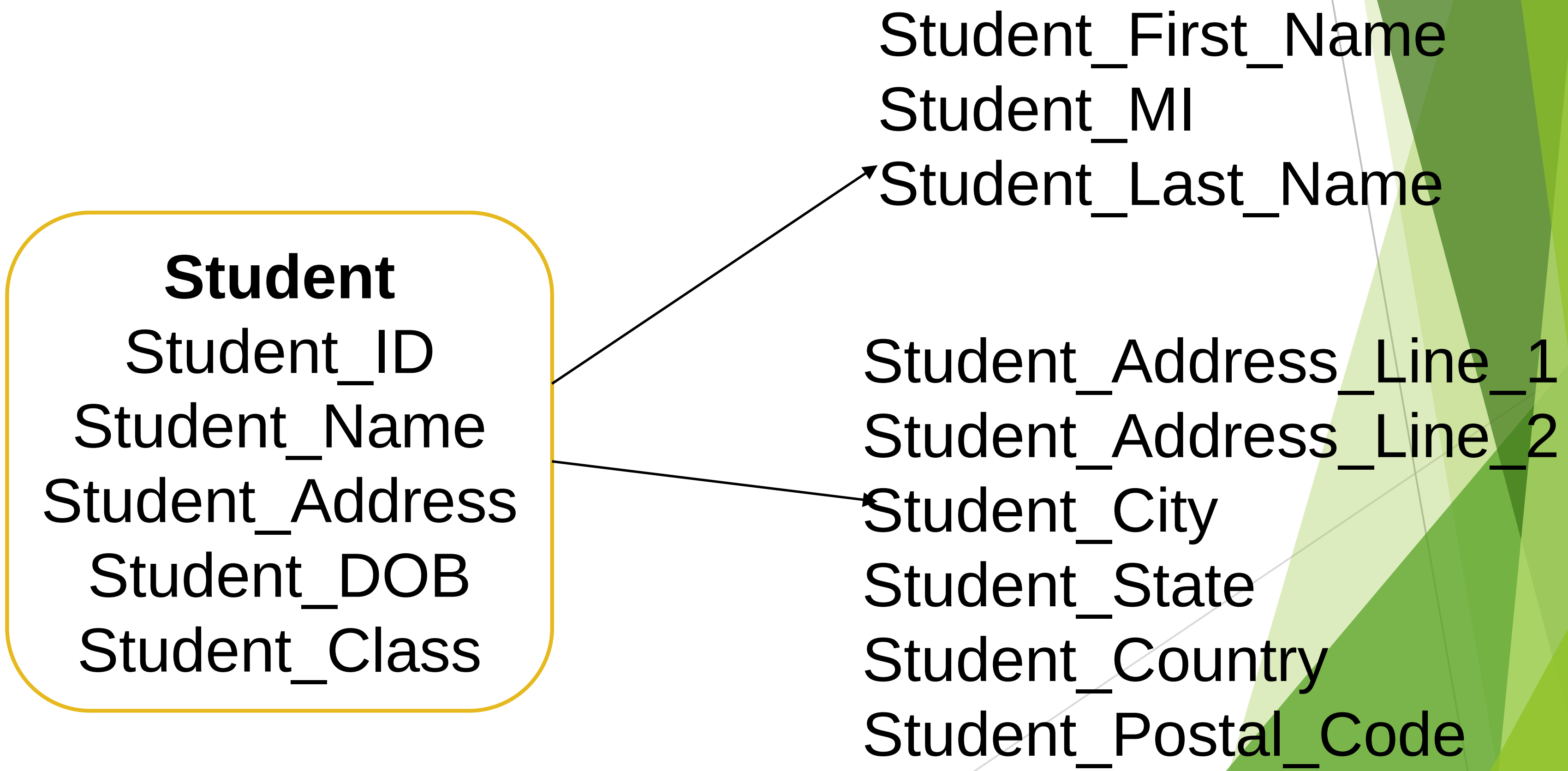
- Must be values that are:
  - Unique for every possible record
  - **Do not change**
  - Best practice:  numeric with no blank spaces or formatting characters
- Often you need to create a <u>surrogate key</u>
  - ID value that serves only to identify the object in the database
  - Exception:  objects with "natural" primary keys
    - SKU
    - ISBN
    - VIN

## Atomic and Composite Attributes

- Atomic attribute:  represents a single data value
  - 15, “Daniel", 12/25/2009
- Composite attribute:  can be decomposed into atomic attributes
  - "James B. Brown"
  - "5580 Pinewood Road, Eau Claire, WI 54701"
  - Should you ever allow a composite attribute in a database?

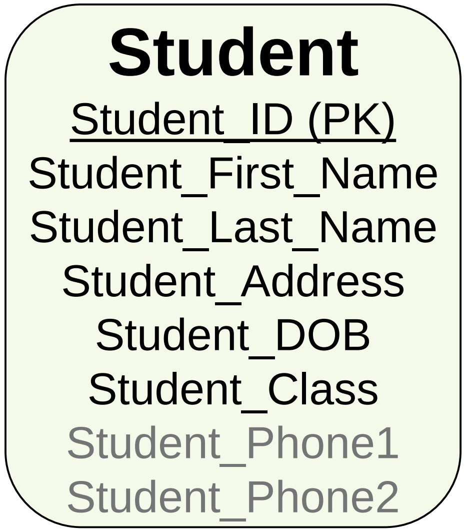
# Composite Attributes

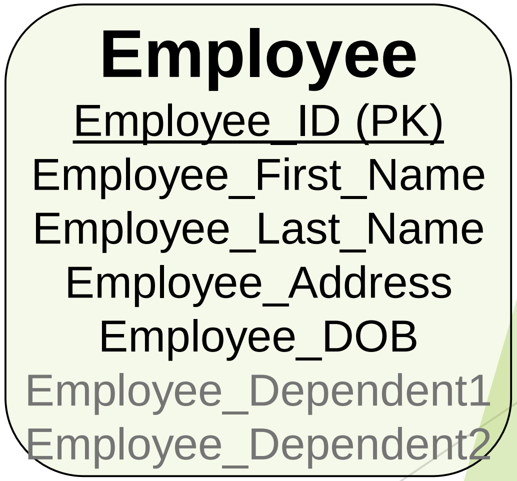
- Decompose into atomic components for:
  - Sorting
  - Searching
  - Formatting



# Multi-Valued Attributes

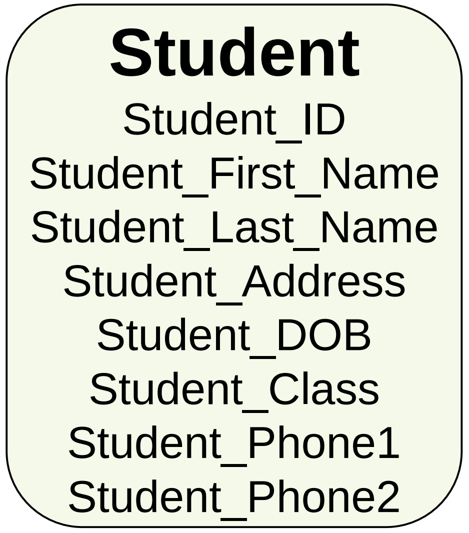
- Can have multiple values for the same entity

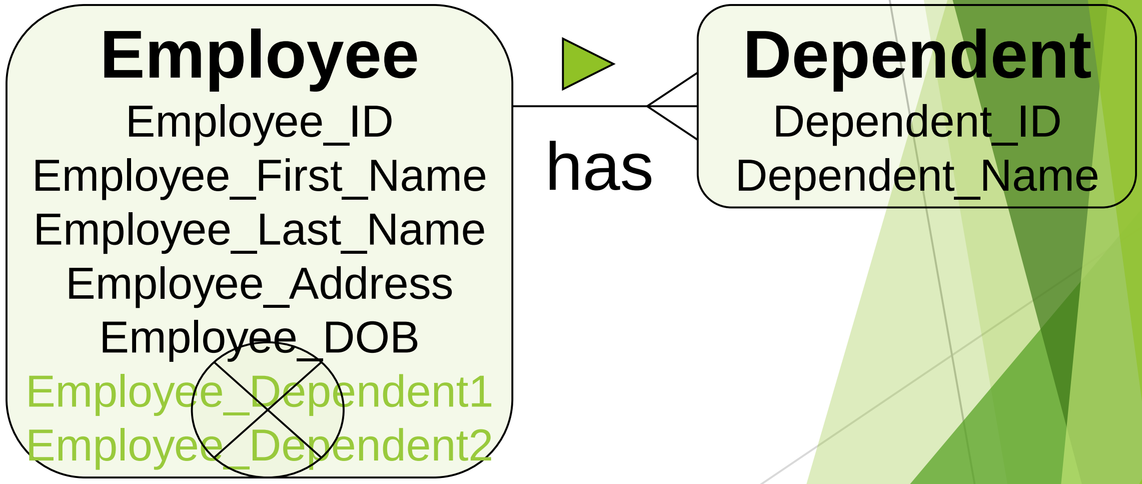




## Handling Multi-valued Attributes

- If it has a definite maximum number, leave as a repeating attribute
- If the upper limit is variable, make a new entity





## Derived Attributes

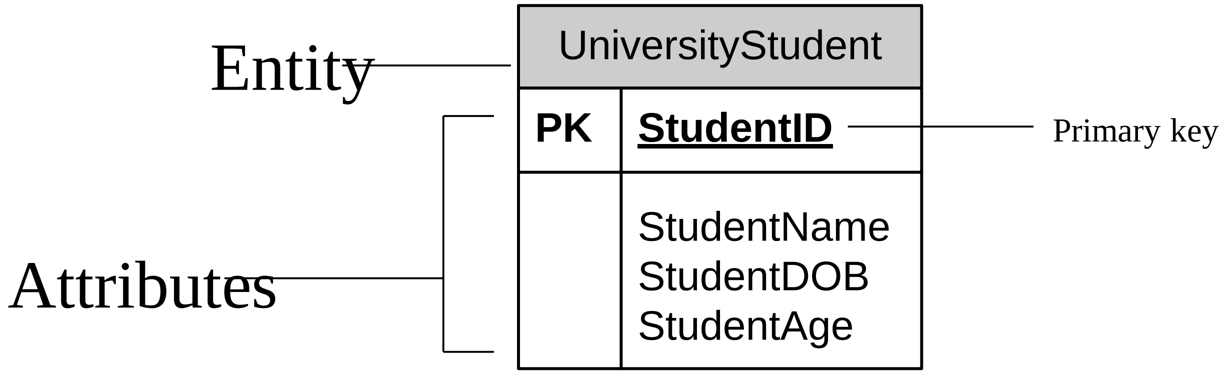
- Value that can be derived from other attributes
  - Student_Age = 22 (DOB = 11/20/1986, current date is 11/13/2009)
  - Order_Total = $500 (Item 1 cost = $200, Item 2 cost = $300)

## Handling Derived Attributes

- Store the underlying data values from which you can derive the attribute value …
  - Examples:
    - DOB => Age
    - CurrentPrice and UnitsSold of an item (for a sales order)
- … unless the underlying values can change!
  - PRODUCT_PRICE, COURSE_CREDITS

# ER Model Notation

- Represent entities as rectangles
- List attributes within the rectangle


Entity
Attributes
UniversityStudent
PK
StudentID
StudentName
StudentDOB
StudentAge
Primary key