# Introducing the Entity-Relationship (ER) Model

* Proposed by Peter P. Chen in 1976
* A semantic data model
* ER model views that the real world consists of:
  + A collection of **entities** and
  + The **relationships** among those entities.

## What is an Entity?

* Conceptual definition: A distinguishable object that exists.
* Operational definition: An important business object with (multiple) valuable properties to store
* Entity set: A set of entities of the same type. (When we refer informally to an ‘entity’, we typically actually mean an entity set
  + Example: Student, University, Course, Customer, Account are entity sets.
  + Bob Q. Peterson is an entity in the Student entity set.
* How to distinguish one entity set to another? By the value of their *attributes* (properties)
  + Example: Each customer has name, street, city.
    - Customer (Name, Street, City)

## What is a Relationship?

* An association between entities
* A semantic connection that needs to be remembered by the system.
* Relationship set: A set of relationships of the same type.
* Examples
  + Students TAKE Courses. Take is the relationship between the student and course entities
  + Customers HAVE Accounts.

## Why use the ER Model?

* Easy to use to model the real world for database design
* Can easily convert to a record-based model.
* An ER diagram is a succinct representation of a database schema.
* It is a communication tool among users, designers, programmers, and business persons.
* Most CASE (Computer-Aided Software Engineering) tools support ER modeling

## Relational Model (RM)

* Proposed by E. F. Codd in 1970
* Based on mathematical set theory.
* RM represents data and their relationships by a set of *relations* (a term from set theory that we can think of as equivalent to *tables*)
* Two properties of a set
  + No redundancy:  
    {Bob, Jack, Cindy} is a set.  
    {Bob, Jack, Bob, Cindy} is not a set.
  + No ordering  
    {Bob, Jack, Cindy}= {Jack, Bob, Cindy}= {Jack, Cindy, Bob}

## Example Relational Model:

CUSTOMER (CUSTOMER\_ID, NAME, STREET, CITY, NUMBER)

ACCOUNT (ACCT\_NUMBER, BALANCE, CUSTOMER\_ID))  
*(primary keys are underlined in the above)*

Note that:

* The above notation is usually used to represent the structure of a table. That is, relation name, open parenthesis, a list of attribute names, and followed by a closing parenthesis.
* Primary key is underlined. The primary key is an attribute that can uniquely identify each instance of the relation.
* Foreign key is indicated with an asterisk (\*); (we haven’t covered foreign keys yet, so don’t worry if this term is unfamiliar).
* The structure of each relation is called a **scheme**. A scheme is a relation name with a set of attributes.
  + The first line is the CUSTOMER scheme
  + The second line is the ACCOUNT scheme.
* They together form a **schema**. (The overall structure of a logical database. In this case the CUSTOMER and ACCOUNT ‘schemes’ together form the schema.)
* Attributes (also called Fields, Column names, Columns)
  + The ordering of attributes is not meaningful
  + But we typically out primary key first (the attribute responsible for assuring each record is uniquely identifiable from every other record -- more on this in a bit!)
  + A relation cannot have redundant attributes
* Data *instances* (also called *tuples*, *rows*, or *records*)
  + The actual data you are storing
  + An example CUSTOMER instance/record/row/tuple:
  + (‘Pete Peirce’, ‘1420 Pine St.’, ‘Philadelphia’, ‘215-670-5000’)
  + The ordering of tuples is not meaningful and the ordering of one record before or after another does not matter
* A relationship between tables is represented by the same attribute name (or different attribute name with the same meaning) in two or more tables.
  + This is the ‘relation’ in the relational model: the attribute that relates data in one table to the data in another

## The Entity-Relationship (ER) Model in Further Detail

### Identifying Candidate Entities, Relationships and Attributes

#### Entities

* Entities are PRIMARY THINGS of a business about which users need to record data.
  + They almost always become tables in the underlying database.
  + *Nouns* are candidates for entity types.
  + Like nouns, Entity Types may be classified into six classes:
    - People - e.g. Employees, Students, Customers
    - Places – e.g. Cities, Offices, Routes
    - Things (tangible physical objects) – e.g. Equipment, Products, Buildings
    - Organizations – e.g. Teams, Suppliers, Departments
    - Events (things that happen to some other entity at a given date and time or as steps in an ordered sequence) e. g. Employee promotions, Orders, Payments
      * The occurrence of an event is identified by at least two factors. One is time or date and the other is the identifier of the entity
    - Concepts (intangible ideas used to keep track of business activities) – e.g. Projects, Accounts, Complaints
* Questions To Ask in Identifying Entities
  + What things do we need to keep data about?
  + What things are essential to the organization?
  + What things do we talk about in the organization?
  + What queries and reports do we need?
* Entities Have Two Properties
  + 1.) Identifying attributes (entity identifier or primary key)
    - Each entity instance must be uniquely identifiable from all other instances; the identifying attribute (primary key) assures that this is the case
  + 2.) Descriptive attributes
* Naming Entities
  + Use a singular noun for an entity name
  + Avoid vague / evasive names such as Task, Form, Operation, Schedule, etc.
  + Have a company-wide naming convention
    - E.g. is it Client, client, Customer, customer, cust or Cust?

#### Attributes

* Attributes are the properties of an entity
* Each attribute has a domain (field type).
  + Domain: all of the values that an attribute is allowed to contain
  + Examples:
    - The domain for salary for a given organization might be decimal values between $35,000 and $500,000
    - The domain for marital status: ‘S’ (single), ‘M’ (married), ‘D’ (divorced), ‘S’ (separated), ‘W’ (widowed)
    - For name, the domain might be any string value
  + Deciding whether an attribute should be a numeric value or string value
    - Ex: Order#, Emp#, Salary, Bonus, SSN
      * If you need to be able to manipulate an attribute mathematically, make it a numeric data type; otherwise, make it a string data type
      * Salary needs to be numeric – need to give a 10% raise
      * SSN, while numeric, is never going to be added, divided, etc. à string is fine.
* Attributes by role
  + Descriptive attributes – the properties you want to track
  + Identifying attribute (entity identifier or Primary Key)
    - Allows each record to be uniquely identified and distinguished from every other record
    - If one or a combination of the descriptive attributes works, use it (although make sure it guarantees uniqueness)
    - Otherwise, create a synthetic (i.e. made-up) ID number attribute (e.g. order number, SSN, etc.)
* Attributes by type
  + Atomic – a single part with only one value at a time
    - Example: last name
  + Composite – composed of many parts
    - Example: address (can be decomposed into house number, street, city, state, etc.)
  + Multi-valued – has more than one value at once – e.g. languages spoken, certifications, etc.
  + Derived – calculated values – e.g. age
    - Why don’t we store age directly in the Database? We can use functions built into the DBMS to calculate these from underlying values that are stored in the DB.
    - Can calculate age based on birthdate (stored in the system) and the current date
  + We want each attribute in RDBMS ultimately to have only one value at a time (atomic value).
  + English grammar components for attributes – they can be nouns, adjectives, or adverbs

##### Attribute Syntax rules for ERDs

* + Every entity type should have an identifying attribute (Entity ID (EID) or Primary Key)
  + Underline the EID on the ERD
  + An EID cannot be NULL (called the “Entity Constraint”)
    - Why Not? NULL (i.e. no value) does not assure uniqueness
* Attribute conventions:
  + Use an attribute name that is unique across the *entire schema*
  + Have an organization-wide naming convention regarding:
    - Capitalization
    - Separation - underscores, dashes, CamelCase, etc.
  + Be as descriptive as possible
  + Model the data at the level they are used
    - If you need to analyze cities, states and/or zip, model them as separate attributes
    - Full text searching within a given attribute can help if you don’t, but best to avoid the need. Break attributes down into their component parts
  + Example convention:
    - Annual\_Interest\_Rate, Due\_Date (Each word capitalized, separated by an underscore, no abbreviations)
  + The ideal: When two parts of the organization are talking about the same thing, you KNOW it, because those things are NAMED exactly the same too

### Rules for Entity Analysis in ER modeling

* Every noun is a candidate for an entity.
* Every entity type should be important in its own right within the problem domain. Entities should be things about which you need to capture *multiple* pieces of information
  + Address is not a good entity.
  + Customer, Vendor, and Product are good entities. They are important in their own right and you’d want to capture lots of info about each.
* If an object type (noun) has only one property to store then it is an attribute of another entity, otherwise it is an entity.
* Do not include *singletons* in your model:
  + If an object type has only one data instance (i.e. there are only one of them), then do not model as an entity type.
    - For example, if you are developing a DB for an independent, mom-and-pop grocery store, you don’t need a “store” entity, because there is only one store in the problem domain.
    - On the other hand, if the DB is for a chain like Giant or Wegmans, then store would be an important entity because there are many of them, they are important, and there are multiple items of information you’d want to capture about them.
* If an object needs to have a unique identifier, THEN model as an entity.
* Every entity should represent only **one** concept
  + An entity that hold info about employees AND their projects is a bad idea and can cause data integrity problems.
  + A description of an entity should never need to contain the word ‘or’
* All the instances of a given entity should have the same characteristics (attributes)
  + An exception to this rule is the specialization/generalization of the Extended Entity-Relationship Diagram (EER) covered in the next section of this objective

#### Relationships

* Actions between associated entity types.
* Each relationship has characteristics including:
* Degree (recursive, binary, ternary or n-ary)
* Cardinality (1:1; 1:N; M:N)

##### 

##### 

##### Identifying Relationships

* What sentences related to the problem domain can be constructed of the form ‘Entity Verbs Entity’?
  + Employee has children (Existence relationships)
  + Professor teaches courses (Functional relationships)
  + Customer places order (Event relationships)
* Ignore verbs that are not tracked by your enterprise (i.e. do not relate entities that you don’t track).
* Verbs must represent interactions that need to be remembered by the system.
* Relationships do NOT represent a flow of data or activity in the real world.
  + E.g., NOT something like the following: submit books & ID to librarian and demagnetize books)
    - This type of phenomenon is captured in other systems analysis models (particularly the Data Flow Diagram and Interaction Diagram)

##### Other Relationship Rules

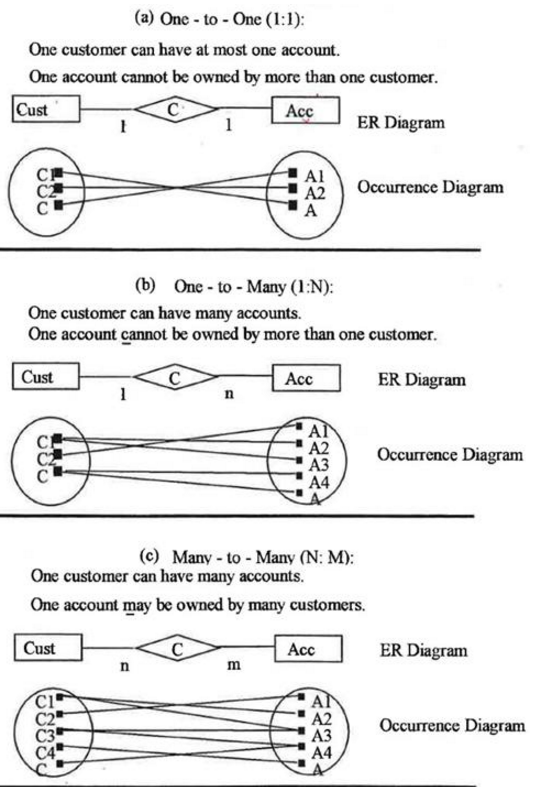
* Relationships and cardinality/participation constraints represent business rules.
* Use a meaningful verb in the present tense for a relationship name
* All relationships are bidirectional.
  + For you grammar buffs, one direction will represent the active voice and the other the passive:
    - A Customer Orders a Product (active)
    - A Product is Ordered By a Customer



* *Binary modeling* tools (unlike the one we are learning here) do not allow relationships to have attributes and indeed do not model relationships directly at all.
  + (Binary modeling tools include products like IDEF1X, ERWin, IEF, ADW, Bachman, Oracle CASE\*Method)
  + They connect entities together directly without the use of the relationship diamond.
* See Figure 2 of the “A Practical Guide to Entity-Relationship Modeling” article for other notations.
* Our model is called *N-ary model* in that it
  + Allows a ternary relationship
  + Uses a diamond for a relationship
  + Allows relationships to have attributes
    - Relationship attributes often have to do with *time*

## Cardinality Constraints and Mapping Diagrams

Cardinality expresses the *maximum* number of entities that can be associated to another entity via a relationship.



## 

## Participation Constraints (PC)

* Constraint for the *minimum* degree of a relationship (as opposed to cardinality, which was maximum degree)
  + There are two possible values:
    - Total (or mandatory or full) participation: 1
    - Partial (optional) participation: 0
* To identify participation constraint, ask the following questions for each entity type:
  + Can an Employee exist without working for a Dept?
    - If yes, then Emp has partial (optional) participation to WORK relationship
    - Otherwise it has total (mandatory) participation to WORK relationship
  + Can a Dept exist without having any employees?
    - If yes then Dept. has partial (optional)
    - Otherwise Dept. has total (mandatory) participation



* Both employee and department have partial participation.
* Employees do NOT have to work for a department.
* Departments do NOT have to have an employee working in them.



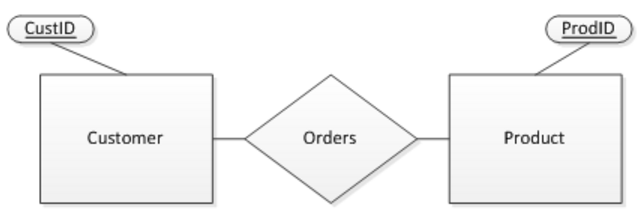
* Employee has mandatory participation, Department partial.
* Employees MUST work for at least 1 department
* Departments do NOT have to have an employee working in them.



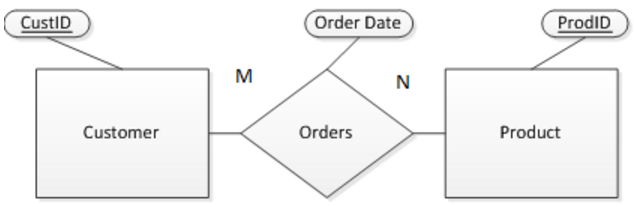
* Department has mandatory participation, Employee partial.
* Employees do NOT have to work for any department.
* Departments MUST have at least 1 employee working in them.

## Relationship Attributes

* There are two types of relationship attributes – Identifying and Non-identifying
* 1.) Relationship Identifier (RID)
  + Mandatory, Identifying
  + **Not** depicted on the ERD
  + Concatenation of identifiers of participating entity types regardless of the cardinality:



* The RID for “Orders” above would be CustID, ProdID
* Because relationship identifiers are always simply the concatenation of the IDs of all participating entities, we do not need to depict them on the ERD
* 2.) Non-identifying Relationship Attributes
  + Optional
  + Depicted on the ERD
  + Properties of a relationship, not of an entity
  + Must be shown on the ERD
  + Also called non-key attributes of a relationship
  + Often non-key relationship attributes have to do with duration, dates or the passage of time
  + Example:



* Order Date can’t be associated with either “customer” or with “product” because customers may place many orders and products are ordered by many customers.
* Order\_date thus needs to be an attribute of the relationship “orders”
* Non-identifying relationship attributes often have to do with time and dates (much like “Order Date” here)
* Relationship attributes are typically necessary only when relationships are M:N cardinality

## Well-formed ERD Summary

Basic Rules

* All entities and relationships must be connected.
* Each entity must have at least one relationship.
* All entity names must be unique.
* Use a singular noun for an entity name and an attribute name.
* A relationship cannot be directly connected to another relationship.
* Use a meaningful verb for a relationship name whenever possible.
* Every entity must have at least one unique key (atomic or concatenated).

Other Tips

* Do not show the EID of one entity in either another entity or a relationship.
* Minimize line-crossing in ERDs.
* List all attributes of an entity on one side for visual simplicity.
* Arrange attributes left-to-right and top-down.
* Underline entity identifiers.
* Use the DB-wide unique name for each attribute
* The data you want to store in the DB must appear somewhere in the ERD.
* Create relationships with the most central entity first.
* Minimize redundant relationships.

## Introduction to Primary Keys (PKS) in the ER Model

* PK Requirements:
  + Uniqueness: No two instances have the same value in the table.
  + Minimality: There should be no redundant attribute in a key.
  + Must be *known* at all times
    - Thus, must not have a null value (entity constraint)
* Widely used in applications
* Should never be changed
* Shorter is better
* Controlled by the database administrator

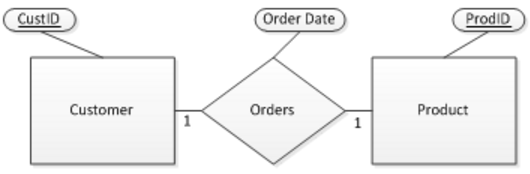
### Guidelines for keys

* Use meaningful keys whenever possible
  + Note, however, meaningful keys tend to be longer than arbitrary ID numbers)
* Composite key = the combination of several non-key attributes to assure uniqueness (first name plus last name plus birthday [MAYBE – won’t work in every situation])
* Generate a surrogate key (i.e. ID) for a very large composite key
* Avoid letters 0, I, Z, S, and V because they can be confused with 0, 1,2, 5, and U when hand-written
* Avoid letters that sound similar like M and Nor B,C,D,G,P,T, and V
* Divide large digits into chunks
* 416-555-1212 instead of 4165551212
* Record time on a 24-hour clock
* Use intuitive, information-bearing keys (C617 for Rm17 at 6th floor at Wing C)
* Use of a check digit can help prevent errors when referring to IDs (See <http://en.wikipedia.org/wiki/Check_digit> for more info)
  + An extra digit calculated from other digits, to identify common errors such as:
    - transcription errors: incorrect digit error (7 instead of 1)
    - transposition error: two numbers are reversed (4576 instead of 4567)

### Determining Primary Keys in the ER Model

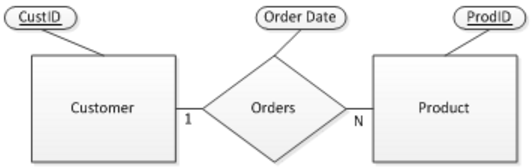
* Primary Key of an Entity = its entity identifier
* Primary Key of a relationship is dependent on the cardinalities:

**1:1 Relationship:**

****

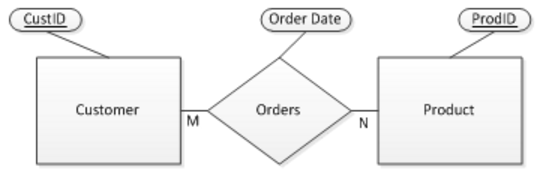
* The PK of a 1:1 relationship can be the PK of **EITHER** participating entity (Why?)
* Order’s PK = EITHER CustID or ProdID

**1:N Relationship:**

****

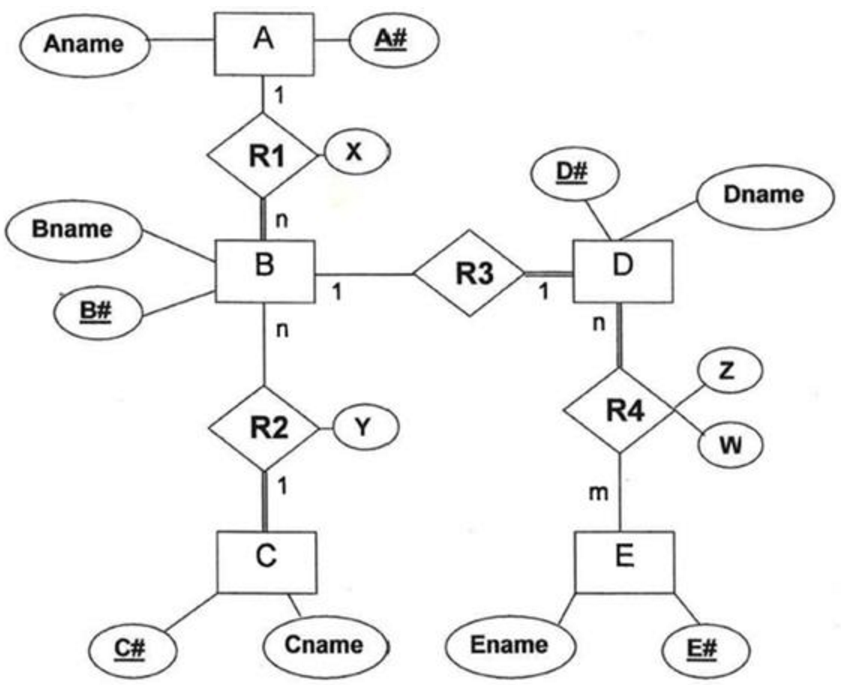
* The primary key of a 1:N relationship is the same as the primary key of the **N-Side** entity. (Why?)
* Order’s PK = **ProdID**

**M:N Relationship:**

****

* The primary key of a M:N relationship is the concatenation of the primary keys of BOTH entities (Why?)
* Order’s PK = CustID, ProdID

### Review of Primary Key Candidates and Relationship Identifiers:



|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Relationship ID** | **Possible Primary Key(s)** |
| A | Entity | N/A | A# |
| B | Entity | N/A | B# |
| C | Entity | N/A | C# |
| D | Entity | N/A | D# |
| E | Entity | N/A | E# |
| R1 | Relationship | A# B# | B# |
| R2 | Relationship | B# C# | B# |
| R3 | Relationship | B# D# | Either B# OR D# works |
| R4 | Relationship | D# E# | D#E# (That is, the concatenation [combination] of D# & E# together) |

## Sneak Preview – next objective in this competency and next step in the DB development lifecycle process:

“Translating” our ERD into a set of “relational schema” that provide a roadmap for what tables we will create in our DB and what attributes those tables will contain.

We need to be able to turn (“translate”) our ERD into a plan for which tables and attributes and how those tables relate to one another.

Some notes about how this will work:

* Every entity will become a table in our DB
* *Some* of the relationships will become tables, depending on:
  + The cardinalities of the entities participating in the relationship
  + The participation of the entities participating in the relationship
  + Whether or not the relationship has any non-key attributes
  + What the operational priorities for the resultant database are
* Knowing how we determine the potential primary keys for a given relationship will be essential in understanding the rules determining how we go about deciding which relationships in our ERD will become tables in our DB and how the resultant tables will relate together
  + So before you press on to Week 4 and Translation, be sure to review the rules for determining a relationship’s possible PK(s)