Entity-Relationship Model & Converting to Tables

UNIVERSITY OF VIRGINIA
DATA SCIENCE INSTITUTE

ER Tools

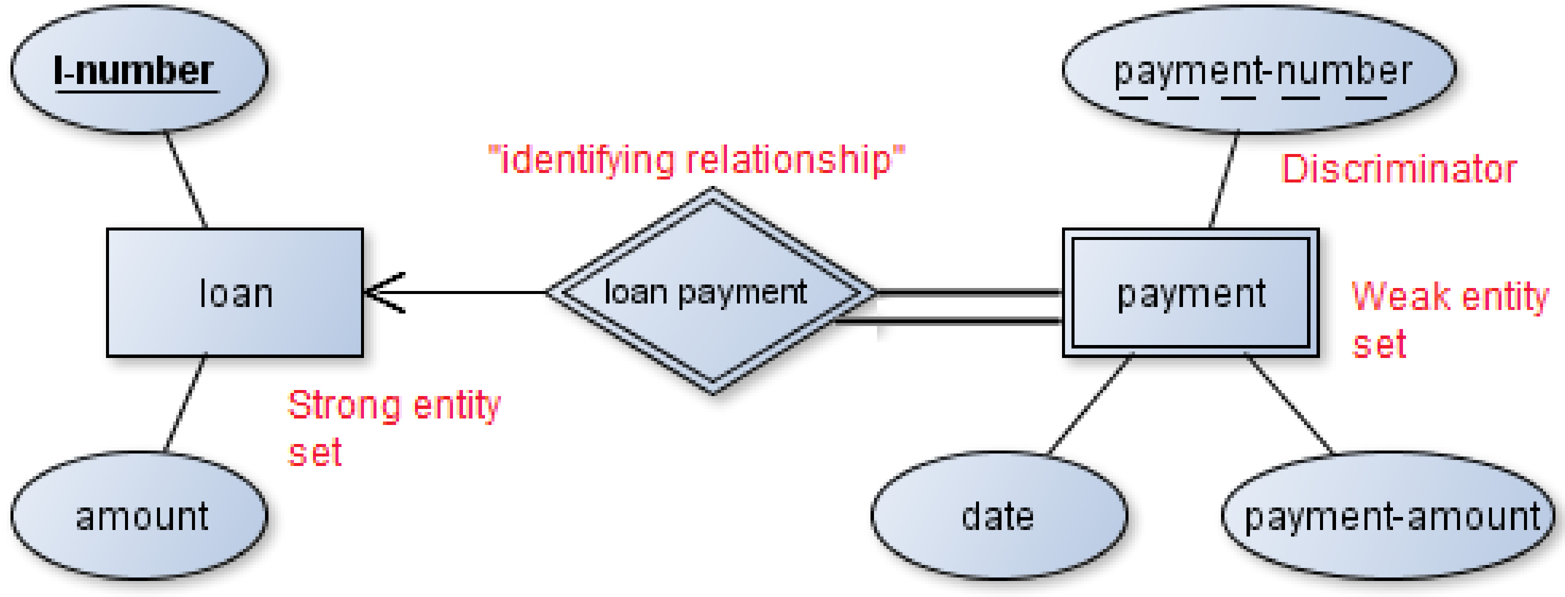
- **yEd**: http://www.yworks.com/en/products_yed_about.html
- **Dia** http://dia-installer.de/
- Others:
  - **draw.io**: https://www.draw.io/
  - **omniGraffle** (Mac) Free trial or buy it http://www.omnigroup.com/products/omnigraffle/
  - **Visio**
  - …

ER Diagram : Heros & Villains

Starting out with “**Hero**” and “**Villain**” entity sets, add attributes to each of them and come up with as many relationships that you can between heros and villains. Have fun with this and feel free to come up with some entertaining attributes and/or relationships!

Hero
Villain

Weak Entity Set



- **Weak entity set** – doesn't have sufficient attributes to form a primary key. It is dependant on the strong entity set it is associated with. The weak entity set still needs some form of identifier: the discriminator

Weak Entity Set

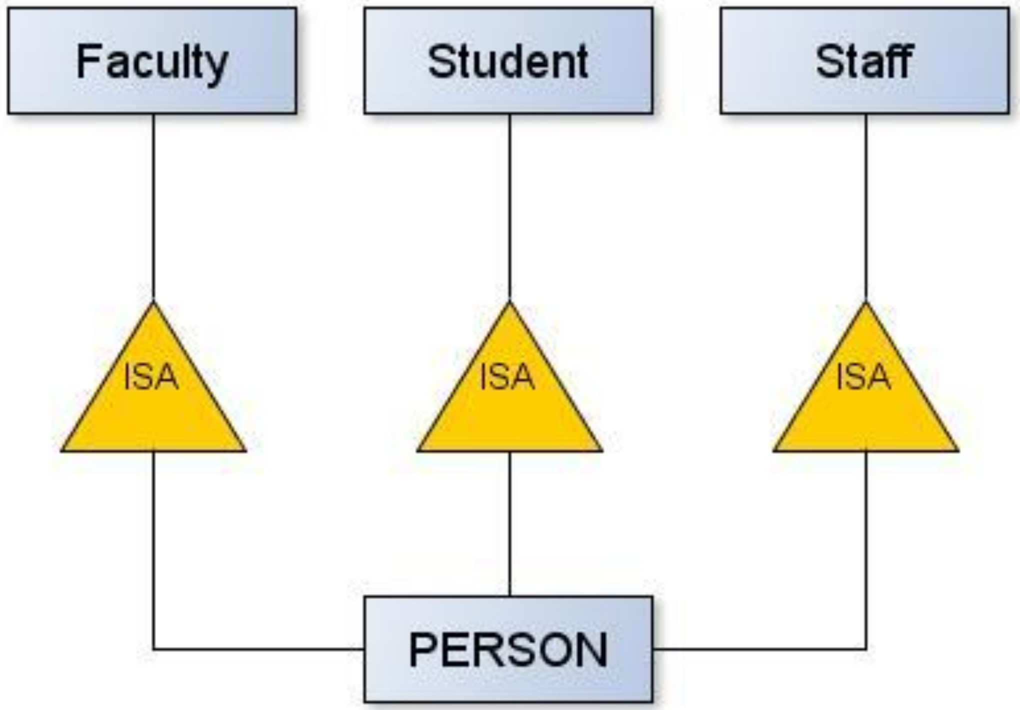
- Payment – although each payment entity is distinct, payments for different loans may share the same payment number (multiple payment #1's, 2's, etc…)
- **Discriminator** – although no primary key, we need a means of distinguishing among those entities in the entity set that depend on one particular strong entity (e.g. payment-number)
- Discriminator – represented like a primary key, except with a dashed underline
- Note: we do NOT put l-number in payment!
- By definition, there must be **total participation** between the weak entity set and its **identifying relationship**

"A student IS-A person", "A faculty member is a person"

ISA

- Entity sets = like classes in OO
- There are super-classes and sub-classes.
- They're called "IS-A" (ISA) relationships
- ** **The triangle points to the specialization** **
- Generalization – "PERSON"
- Specialization –

  "Faculty", "Student", "Staff"



➨ The specialization (subclass) sets have everything the generalization (superclass) set has but more.

Decisions to make…

- Entity set vs. Attributes
  - If has more data → entity set
  - If is the data → attribute

- Entity set vs. Relationship set
  - Verb argument:
    - Entity set: *nouns* (students, faculty, loans, …)
    - Relationship: *possession verbs* (teaches, advises, battles, owns, loan payment, …)

Decisions to make…

- Binary vs. n-ary Relationship sets
- Specialization/Generalization for modularity
- Aggregation for modularity

Converting E-R Diagrams to Tables

- Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database
- A database which conforms to an E-R diagram can be represented by a **collection of tables**
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set
- Each table has a number of columns (generally corresponding to attributes), which have unique names
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram

Strong entity set

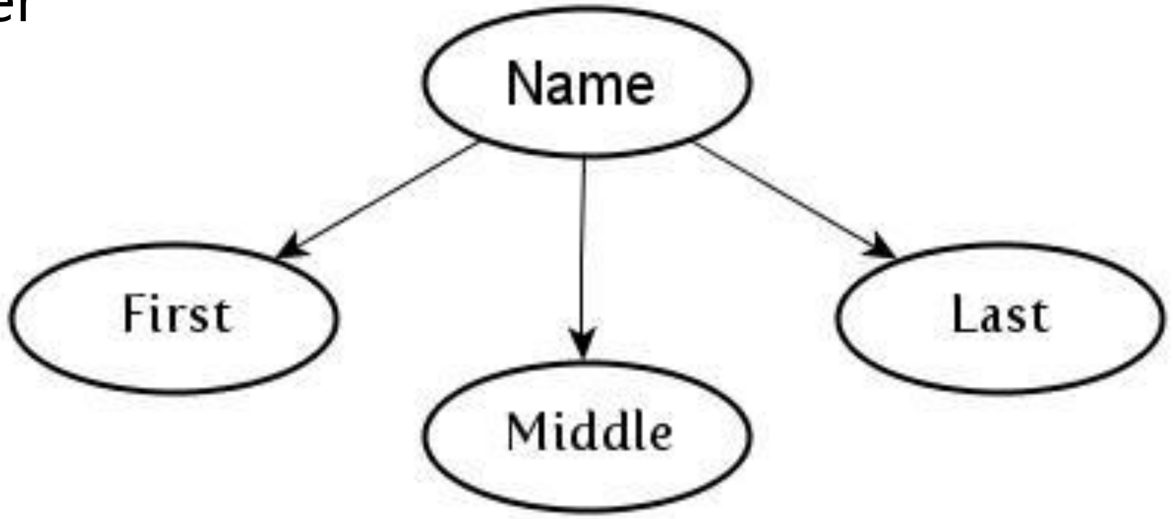
- Strong Entity Set (E) – attributes {a_1, a_2, ..., a_n} :
- Table with primary key and attributes as columns
- Primary key: same as in entity set

- e.g. **student**(<u>ID</u>, name, tot_credit) ← *Schema statement*

Strong entity set (with composite attribute)

- Create separate attributes for each component – don't include the higher level attribute "Name"
- Name becomes: first_name, middle_name, and last_name

- e.g. **instructor**(<u>ID</u>, first_name, middle_name, last_name, x, y, z)



Strong entity set (with multivalued attribute)

- Attribute M (e.g. phone) on an entity set E (e.g. Instructor) is represented by a <u>separate table</u> called EM (e.g. Instructor_phone) that is, *the concatenation of the two tables names, often separated by an underscore "_"*
- EM has attributes: primary key of E ("ID"), and attributes corresponding to multivalued attribute M
- Primary key of this table: **ALL** attributes
  (the primary key of entity set E and the attribute M)

- e.g. **Instructor_phone**(<u>ID</u>, <u>phone_number</u>)

Weak entity set

- Weak Entity Set (A)
- If A is a weak entity set, and B is the identifying strong entity set on which A depends
- Table: primary key of B and all A’s attributes
- Primary key: combination of primary key of B (strong entity set) and discriminator of A

- e.g. **payment**(<u>l-number</u>, <u>payment-number</u>, payment-amount, date)

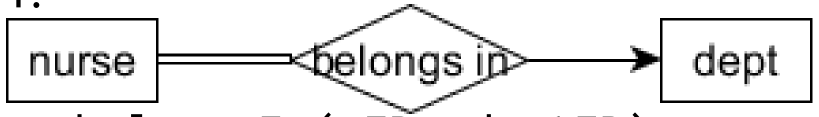
Relationship set

- **Many-to-many**
  - Table: primary keys of the two participating entity sets and any attributes on the relationship itself *(may have dups here)*
  - Primary key: the primary keys added from the participating entity sets
- **One-to-many / many-to-one**
  - If total participation on the *many* side
  - Table: adding extra attribute (the primary key of the *one* side) to the many side
  - **Think: *which side takes the primary key of the other? Side of many***
  - Primary key: Original primary key of entity set of many side

Example: *nurse and department*

Total Participation (double line)

- Scenario 1:

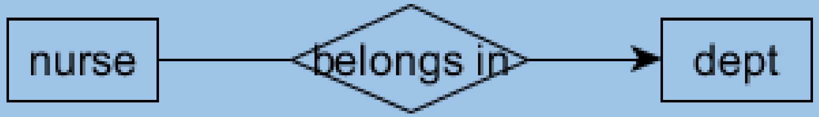


belongsIn(<u>nID</u>, deptID)

Let's start with this scenario

- **For Scenario 2:**
(many-to-one or one-to-many mapping between two strong entity sets)
- The primary key of the entity set on the "many" side (nurse) is the primary key of the relationship. So:
- belongsIn(<u>nID</u>, deptID)

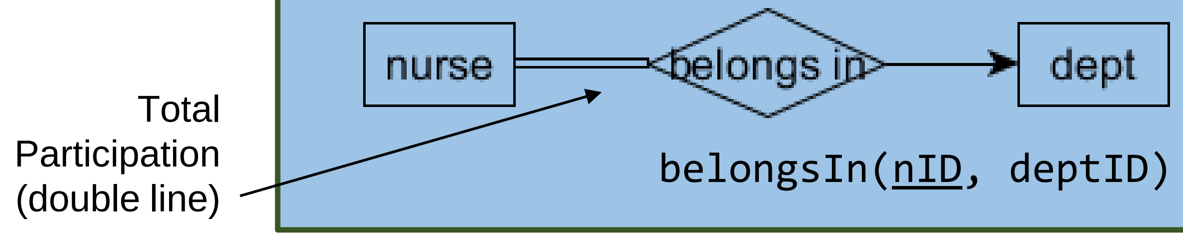
➨ Scenario 2:



belongsIn(<u>nID</u>, deptID)

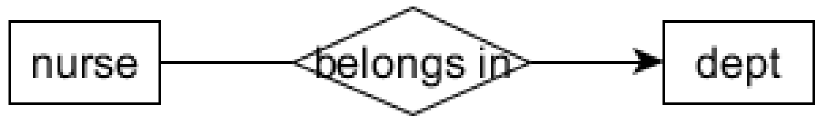
Example: *nurse and department*

Scenario 1:



belongsIn(<u>nID</u>, deptID)

Scenario 2:

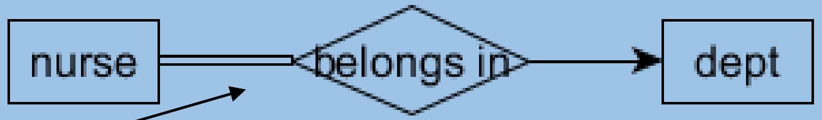


belongsIn(<u>nID</u>, deptID)

- **For Scenario 1:**
- Due to the total participation, you can ***choose*** to represent this relationship (belongsIn) in a different way:
- Can choose to add the primary key of the one side (dept) to the nurse entity set since with total participation EVERY nurse must 'belong in' a department ***((many side gets PK of other))***
- Therefore, you could choose to create the nurse table like:
- `nurse(`<u>`nID`</u>`, fname, lname, …, deptID)` (PK will still be nID)

Example: *nurse and department*

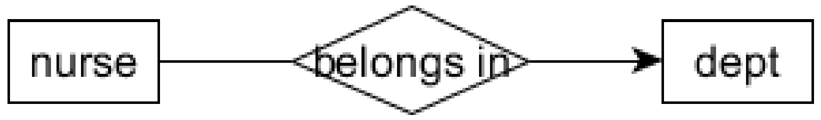
Scenario 1:



belongsIn(<u>nID</u>, deptID)

Total Participation (double line)

Scenario 2:



belongsIn(<u>nID</u>, deptID)

In this case, you will *no longer* need to create a "belongsIn" table *[This is the difference when we see **total participation** on the many side]*

ipation, you can ***choose*** to represent this relationship (belongsIn) in a different

primary key of the one side (dept) to the nurse entity set since with total
rse must 'belong in' a department ***((many side gets PK of other))***

- The...efore, you could choose to create the nurse table like:
- `nurse(nID, fname, lname, …, deptID)` (PK will still be nID)

Relationship set

- **One-to-one**
  - Table: Either side can act as the “many” side, adding the primary key of the other
  - *Which one?* Doesn’t matter – which ever seems to make the most sense
  - Primary key: Original primary key of entity set you chose to add the primary key of the other

Summary

If relation one has prikeyA and relation two has prikeyB, then…

If the relationship is

1:1	pick ONE of the two to be the primary key of the relationship
m:m	BOTH of the primary keys become the primary key of the relationship
1:m	the primary key on the MANY side is the primary key of the relationship

Look out for…

- If partial participation on the ***many*** side, watch for ***null*** values
- DO NOT do this for identifying relationship sets

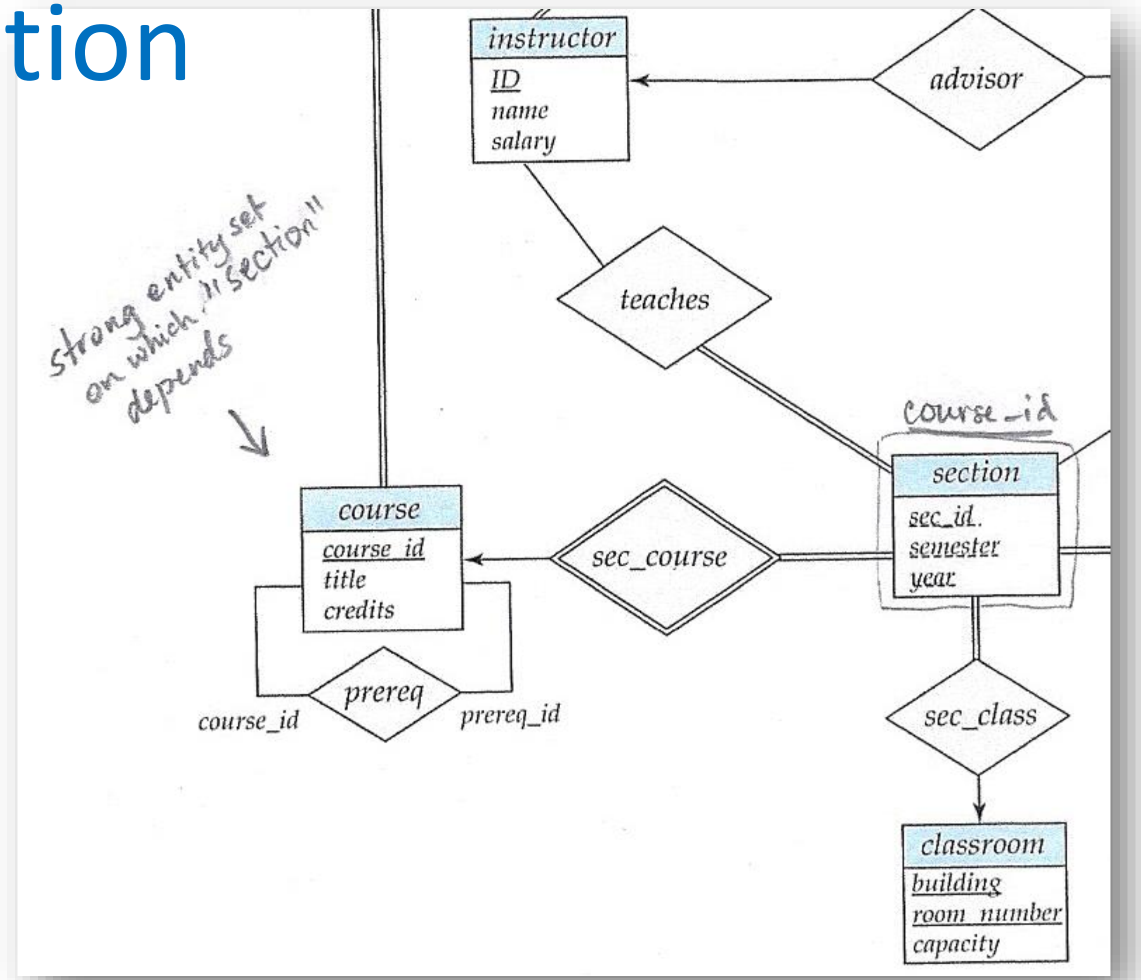
- What if you specify limits? Those limits have to be managed programmatically – **NOT in the DB**. *There isn't a good way to manage that in the database.*

Pay attention to this situation

<u>"teaches" relationship set</u>

- "**section**" is a weak entity set
- "**course**" is a strong entity set which **section** relies on
- *<u>Remember</u>*: weak entity set primary key is: primary key of strong entity set + its discriminator
- teaches(<u>ID</u>, <u>course_id,</u> <u>sec_id</u>, <u>semester</u>, <u>year</u>)

ID=instructor; **course_id**=course; rest: the discriminator of section *(which happens to be all three attributes)*



Specialization (ISA's)

Method 1: *aka…Keep Everything…*

creates table for generalization set and specialization sets

- Form a table for the higher level entity
- Form a table for each lower level entity set, including primary key of the higher level entity set and local attributes
- *Drawback*: getting info about lower levels requires accessing more tables

Specialization (ISA’s)

Method 2: *aka…Keep Specialization entity sets only*

create tables for specialization sets and no table for generalization set

- Form a table for each entity set with all local and inherited attributes.
- If specialization is total (all must be specialized), table for generalized entity is not required to store info
- *Drawback*: people that are more than one specialization will have repetition of some info (worker and customer)

Specialization (ISA’s)

- Tough decision – depends on how many attributes each one has

- Method 3? *aka…Keep generalization set only*
- Let’s assume PERSON has all the attributes and Faculty/Student/Staff have <u>one</u> attribute <u>different</u>
- You would probably make the **Person** table and just have 3 attributes at the end that would be <u>null</u> in the other cases
- It’s not pretty but its less duplication of data

Specialization (ISA’s)

- If balanced – create all four --- *method 1*
- If more attributes in specialization --- *method 2*
- If more attributes in generalization --- *method 3*

To sum up:

- Design decision – based on number of attributes
- DB administrator’s decision
- Goal: minimize duplication (there is no <u>one</u> “correct” way)

Aggregation

- [*Review this material on your own*]

- Create a table with
  - primary key of aggregated relationship,
  - primary key of the associated entity set,
  - plus any descriptive attributes