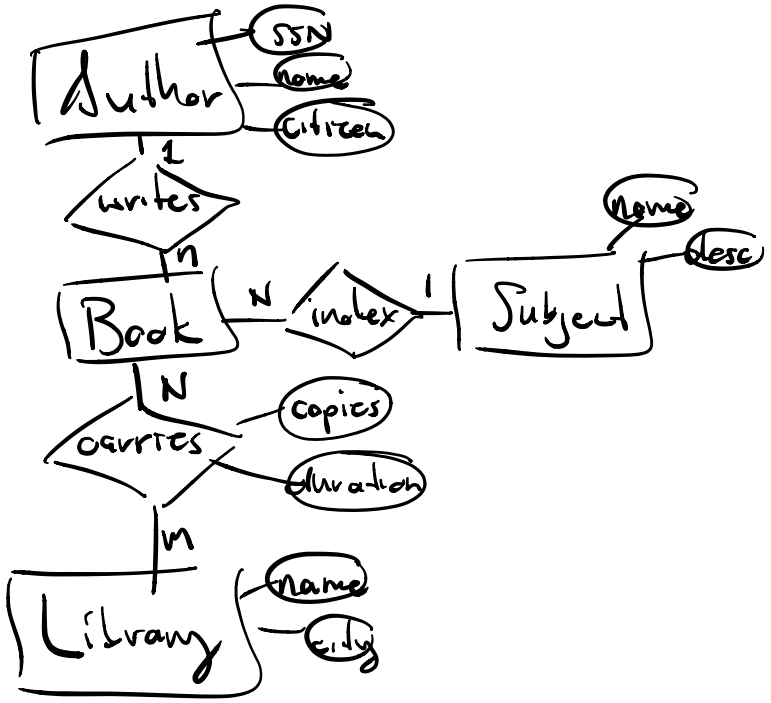
Logical Data Model

- so far, we've mapped the data agnostic of any particular implementation
- now, we want to think about what kind of DB to use, if not the specific DBMS
  - in our case the relational model is the de facto choice



In the relational model, each relation is represented by a file, drawn as a table.

So, the above ER diagram would look like:

Authors

name	SSN	citizenship
Isaac Asimov	123	UK
Stephen King	321	US
...		

Books

title	ISBN	SSN_author
Foundation	1115	123
Intro DB	2215	321
...		

Writes

SSN	ISBN
123	1115
321	2215
...	

← each row in a relationship set relation, represents one instance of that relationship

Index

ISBN	name
1115	Sci-Fi
2215	Comp Sci

Relational Model

- both Entities and Relationships are represented by relations (aka tables, aka sets of tuples)
- each entity is a single tuple (aka row) in the relation
- each attribute of an entity is a column (aka field) in the tuple (aka row)
  - remember that all entities in an Entity set have the same attributes.
- each relationship is a single tuple (aka row) in the relation
- the fields for a relationship relation are the primary keys of the participating entities, as well as any attributes that the relationship itself has

ex. Carries

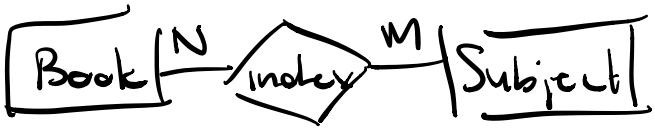
ISBN	libraryname	libraryloc	copies	duration
1115	DPL	1115 Street	4	1 year
2215	DPL	1115 Street	1	5 years

1 — primary key of Book

1 — primary key of Library

★ OPTIMIZATION: if the relationship is 1:N or N:1 and has no attributes of its own, you don't need a separate relation for that relationship. You can just incorporate the primary key of the 1-side into entities on the N-side

why can't we optimize for N:M relationships? say for example:



then we could try:

Book

ISBN	Title	Subjects
1019	Cats	{philosophy, cats}
.....		

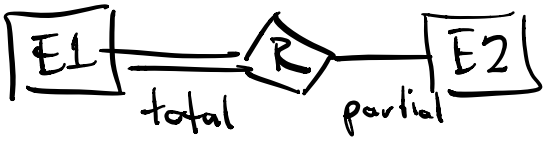
all DBs (?) use fixed-width fields, so we can't have an attribute with no limit on its length, as would be necessary for an N:M relationship, where N and M are unbounded

General Comments about Relations:

- a relation is a set of tuples
- the # of tuples in a relation is theoretically unbounded
- the order of tuples in a relation is not important
- the order of attributes in a tuple <u>is</u> important
- a value may appear multiple times in a "column"
  - * EXCEPT if that column/attribute (set) is a primary key

Additional Concepts for Logical Data Model

Participation constraints:



Referential Integrity

- for entities that have a participation constraint, referential integrity is the term to denote whether each entity fulfills that participation constraint
  - eg. if all Books <u>must</u> have a subject, then a book without a Subject would be breaking referential integrity
- this also deals with having references that

are valid

eg. if a book has the subject "cats" but there is no "cats" in our Subject entity set, that also breaks referential integrity.

- with good upfront design, you shouldn't have to worry about this

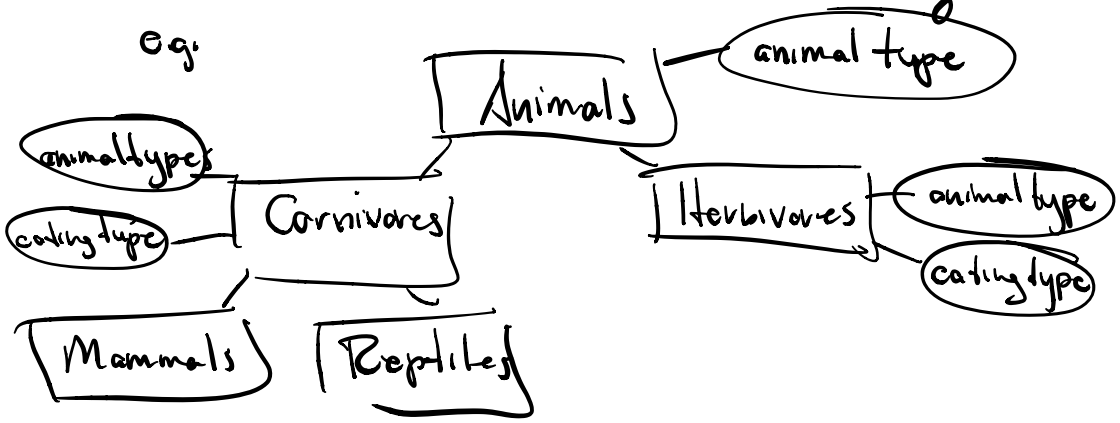
Generalization Hierarchy -

- referring to hierarchical data models

e.g. my Honda is a Car
Cars are Vehicles
Vehicles are Physical Objects

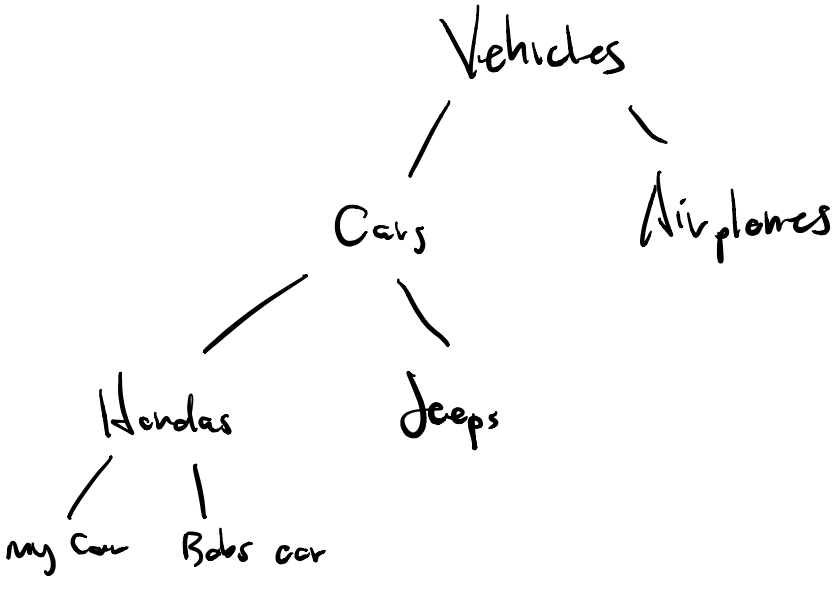
- similar to inheritence or class hierarchy in programming
- you might want redundant data with different relations for levels of the hierarchy

eg.



Creatures

name	animal	eatingtype	taxonomy
Lizard	True	carnivore	reptile
....			
....			

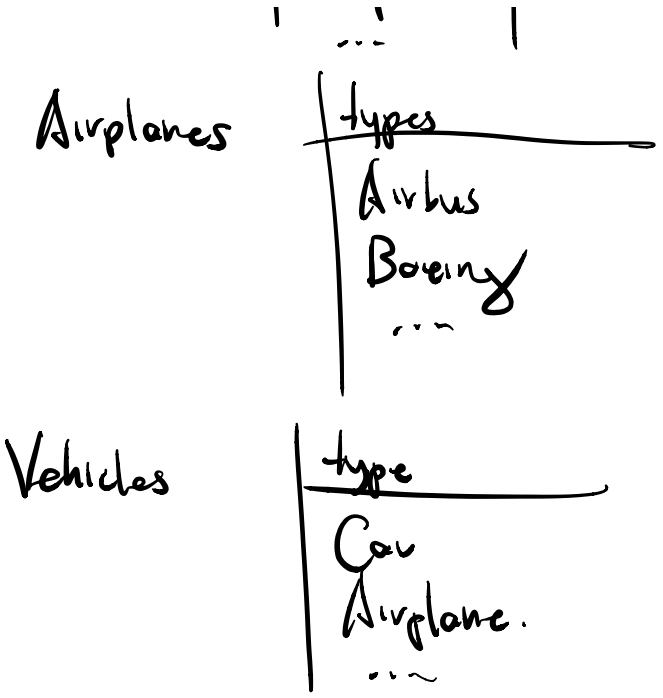


Hondas

owner	VIN
Bob	123
Me	321
.....	

Cars

type
Honda
Jeep



- these kinds of relationships might be examples of the greater class of "is a" relationships

Ordered Relations

- a relation is an unordered set of tuples
- however, sometimes the values in a particular field of a relation will provide an implicit order to the tuples
- most often, this will happen with a primary key
  eg. a field full of unique values that are numbers of strings can be sorted numerically or alphabetically, or otherwise

* this will almost always be the case for an 'id' field.

ID Field

- in the real world, essentially every table in a database will have an additional field whose sole purpose is to uniquely identify rows in that table
- this is usually the 'id' field
- 'id' is, if nothing else, an auto-incrementing integer
- the 'id' field is very key often the primary key of a table
- this is a powerful concept but also dangerous
  - manually selecting appropriate primary keys help avoid duplicates, etc...
  - you could just add an id field as a component of the primary key (as a compromise)

Author

id	name	SSN	citizenship
1	Isaac A.	134	UK
2	Stephen K.	123	US
3	...	...	...