

Data(base) Models



The Grand Idea

People have different views of data so we must model it in some way that abstracts the details into a more universal understanding. Without this "language" to speak in these people cannot clearly communicate their viewpoints and needs.

Design

- Database design concerns itself with the whole picture—structure, storage, and management of data
- Data modeling is a component of design
 - Def. → simplified abstraction of complex real-world structures, objects, or events
 - It is typically graphical simplification of data's meaning and relationships
- What the heck does that mean?
 - Something that is easy-to-read and shows the complexity at a glance



Database modeling

- Book → a data model represents data structures and their characteristics, relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain
- A data model explicitly determines the meaning of structured data
- Structured → data that does not need to have its meaning elaborated; no context required
- Unstructured → the opposite, examples include images, sounds, binary files, and language

Alternative (for programmers)

- Data models are akin to data structures
- When we want to model something we determine the structure or relationships
 - We take data as programmers and put them into trees, lists, etc. which take advantage of relationships in the data
 - We use similar thinking when creating data models, we simply implement and represent the ideas in slightly different ways

Three Components of a Model

1. A description of the data structure that will store the data
2. A set of enforceable rules to guarantee integrity
3. A data manipulation methodology to support data transformations

Note about terms

- Data model and database model are usually considered interchangeable terms
- The book uses data model to mean what it sounds like and database model to mean an implementation (re: database) of a data model in a database system

Patience

- Coming up with a data model is not a one-off task
- Models require *iterative refinement* as understanding of the domain increases
- When complete, a data model should be a complete description of the domain, a blueprint of how to program the database

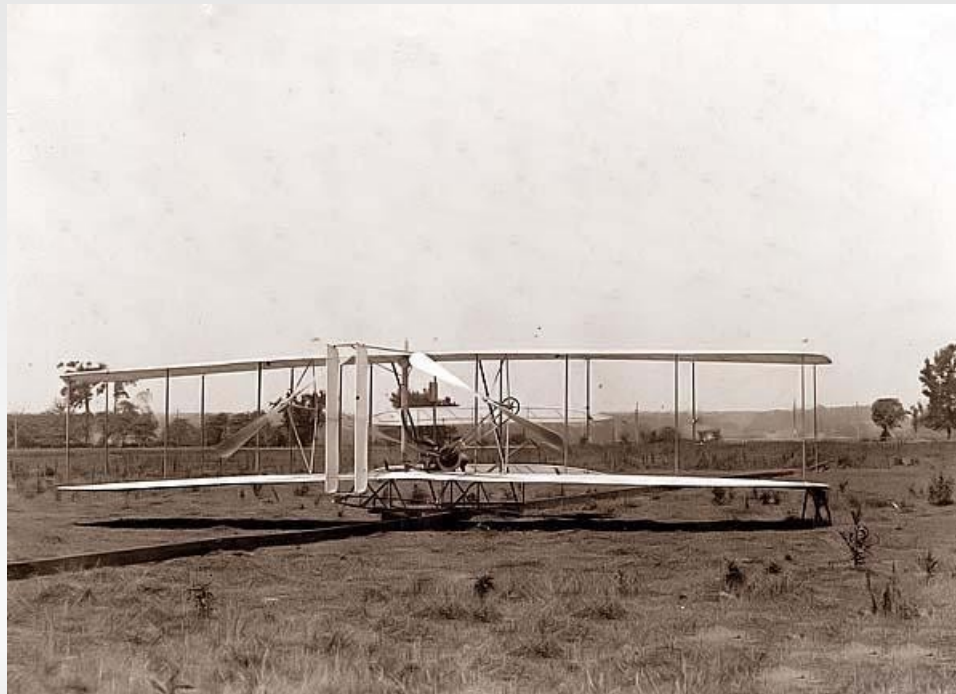
Case Study

Modeling Cars



Another

Modeling Airlines

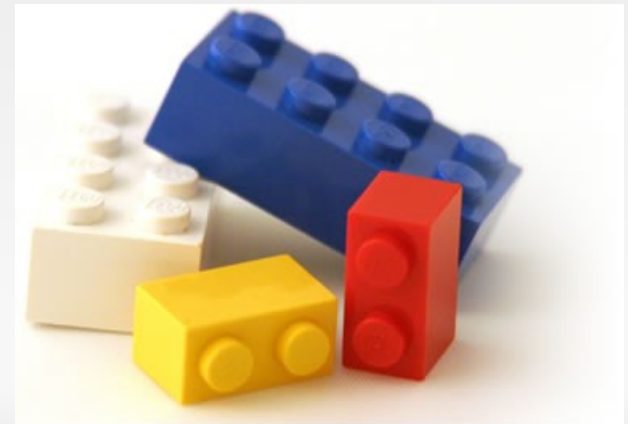


Art and Science

- Being clever and using good judgement are the former standards of database design
- As we (hopefully) just saw, everyone has a different opinion on good design
 - Especially when you consider designers, programmers, managers, other types of workers, and users all must be involved!
- Tools and systems have been developed to help lessen this problem

Building Blocks

- Data builds to information builds to knowledge
- Raw materials create building materials create buildings
- Subatomic particles compose atoms which in turn compose larger structures
- What am I getting at here?
 - **Entities, attributes, relationships, constraints**



Entities

- Def. → anything about which data is/are to be collected and stored
- These can be any real-world object
 - Customers, parts
 - Flight routes, toxic assests
- They are also **distinguishable**, meaning each is both **unique** and **distinct**



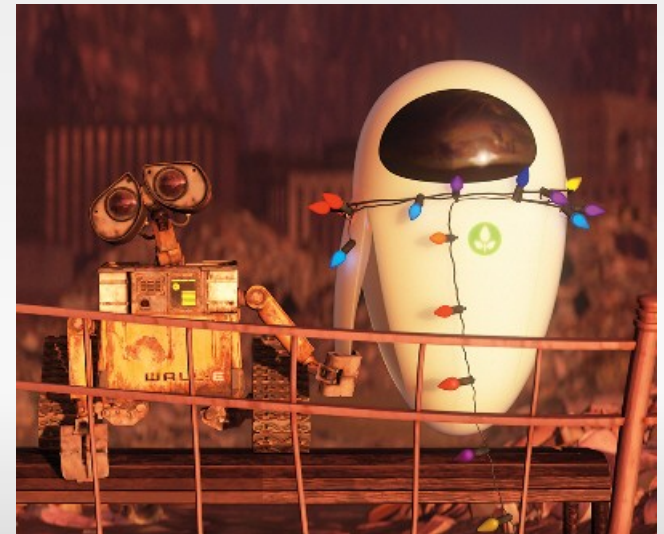
Attributes

- A characteristic *about* an entity



Relationships

- One of the most important and hardest things to get right in databases (as well as life)
- Relationships describe **associations** *among or between* entities
- There is a relationship between product and manufacturer, for example



Relationship Types

- One-to-many, 1:M, 1..*
 - One director can make many films, but each film only has one director
- Many-to-many, M:N, *..*
 - A director can make films in many genres, and genres have many directors who make films in them
- One-to-one, 1:1, 1..1
 - A script is written for only one movie and a movie is filmed with only one script

Relationship Types

- Make sure you understand those differences
- Important to recognize each is bidirectional
 - The relationship each way must be defined!

Constraints

- **Restrictions** placed on data
- Main enforcement mechanism to maintain data integrity
- More simply, these are rules that apply to data
 - A GPA must be between 0 and 4
 - Area codes are 3 digits

How to figure them out

- Examining the data to figure out the entirety of these categories is a daunting task
- A place to start is to think about what sort of data the organization has and how it is used
- **Business rules** → brief, precise, and unambiguous description of a policy, procedure or principle within a specific organization
 - These can be extremely helpful in your design process

Where to find business rules

- Best place (despite what the book says) is written rules
- Without these you run into the perceptions problem, same as how people view data differently
- Not all rules can be modeled, even if we can enforce them programmatically

Business to Database

- Nouns will typically be objects and verbs a relationship
- "A **customer** may *generate* many invoices"
 - Customer, invoices are objects or entities
 - Generate is a relationship
 - What sort of relationship is it?

Model Timeline

TABLE 2.1 Evolution of Major Data Models

GENERATION	TIME	MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File System	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and Network Data Model	IMS ADABAS IDS-II	Early database systems Navigational access
Third	Mid-1970s to present	Relational Data Model	DB2 Oracle MS SQL-Server MySQL	Conceptual simplicity Entity Relationship (ER) modeling and support for relational data modeling
Fourth	Mid-1980s to present	Object-Oriented Extended Relational	Versant FastObjects.Net Objectivity/DB DB/2 UDB Oracle 10g	Support complex data Extended relational products support objects and data warehousing Web databases become common
Next Generation	Present to future	XML	dbXML Tamino DB2 UDB Oracle 10g MS SQL Server	Organization and management of unstructured data Relational and object models add support for XML documents

Hierarchichal Model

- Resembles a tree data structure
- The book uses the term segment to mean both levels and nodes, we will use the individual terms
- Each node can have any number of children
 - What is this relationship type?
- Each node can be considered the root of its own subtree

Hierarchical Model Problems

- Easy to understand but can be tricky to manage and implement
- Lacks structural independence as well as no standard implementation
- All relationships are 1:M and many types of data relationships cannot be modeled this way

Network Model

- Created make complex data representation easier, improve performance, and create a standard
- Defined these components:
 - **Schema** – like a blueprint for a database, explains the names, record types, and components of records
 - **Subschema** – a subset of the database that applications can use to access only what they need
 - **Database Management Language** – defines the environment in which data can be managed

The DML

- The database management language in turn defined these components
 - **schema Data Definition Language (DDL)** – define the schema components
 - Similar ones for the subschema
 - **Data manipulation language** – how to work with data

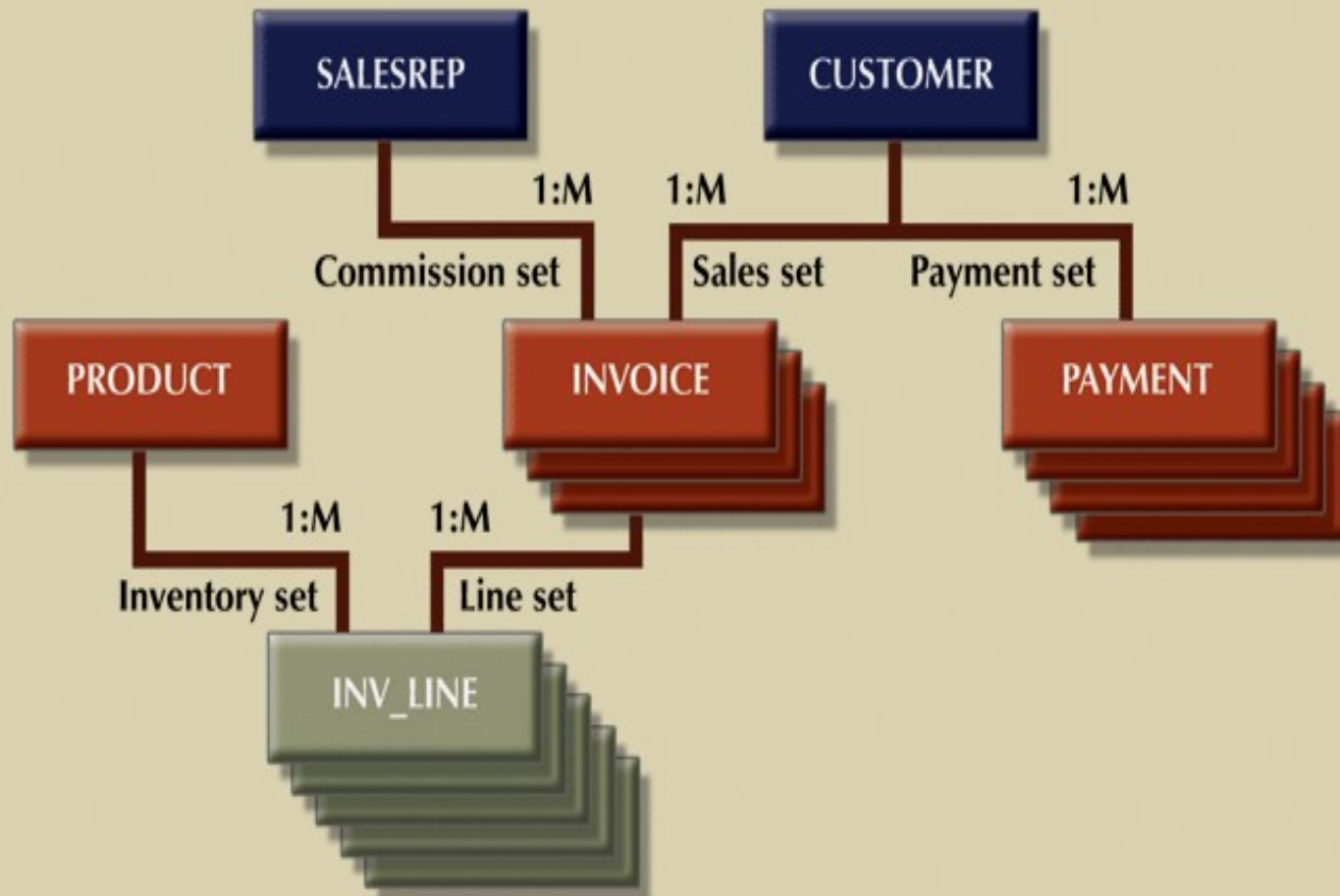
The Important Stuff: Relationships

- Entities/objects are kept in collections
- Each collection is called a set and each set has an owner
 - This relationship is a 1:M for owner:set
- Like hierarchical so far but...
 - A set can have more than one owner
 - While at first glance this may seem like M:N it isn't

Network Model

FIGURE 2.2

A network data model



Network Model Problems

- Again lacks structural independence
- No ad hoc querying ability nor query language
- The weird way relationships are handled eventually make network models too cumbersome to deal with

Relational Model (wooahoo!)

- Very simple conceptually
- Made DB use much simpler for programmers and users
- Abstracted the physical and logical representations
 - Structural independence for the first time!
- Originally thought to be too computationally expensive

Relations

- **Relations** form the base of the relational model
 - Don't confuse a relation with a relationship
- The easy way to think of a relation is what we'd call a table
 - A matrix of rows and columns where a row is an entity and a column is an attribute
 - The intersection is a particular entry's attribute value
 - Rows are called **tuples**

The RDBMS

- An RDBMS is similar to the basic DMBS definition from last chapter as well as the ones that serve the prior two models
- However, it adds more functionality
- It abstracts the physical representation of the data, users only worry about knowing tables and what is in them and the DBMS worries about how to pull them out of storage

The RDBMS

- Tables are related through common attributes
 - Ex. an Orders table will have orders and the part numbers in them, with the Parts table will also have the part numbers listed
- The tables are independent so this could be seen as data redundancy (bad!)
 - We'll learn later how to keep the relationship but eliminate the redundancy
- Relationships can be marked as any of the types we've discussed (1:1, 1:M, M:N)

A relationship example

FIGURE 2.3 Linking relational tables

Table name: AGENT (first six attributes)

Database name: Ch02_InsureCo

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

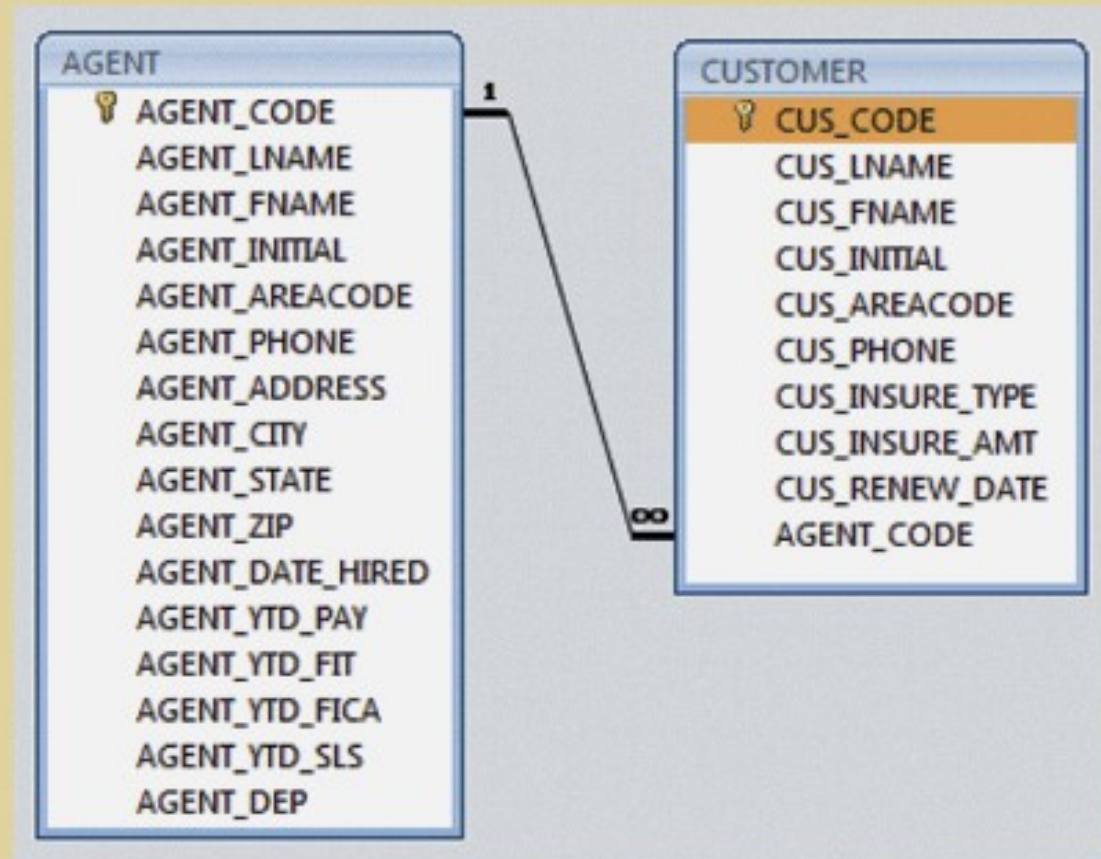
Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_INSURE_TYPE	CUS_INSURE_AMT	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	T1	100.00	05-Apr-2008	502
10011	Dunne	Leona	K	713	894-1238	T1	250.00	16-Jun-2008	501
10012	Smith	Kathy	vV	615	894-2285	S2	150.00	29-Jan-2009	502
10013	Olowski	Paul	F	615	894-2180	S1	300.00	14-Oct-2008	502
10014	Orlando	Myron		615	222-1672	T1	100.00	28-Dec-2008	501
10015	O'Brian	Amy	B	713	442-3381	T2	850.00	22-Sep-2008	503
10016	Brown	James	G	615	297-1228	S1	120.00	25-Mar-2009	502
10017	vWilliams	George		615	290-2556	S1	250.00	17-Jul-2008	503
10018	Farriss	Anne	G	713	382-7185	T2	100.00	03-Dec-2008	501
10019	Smith	Olette	K	615	297-3809	S2	500.00	14-Mar-2009	503

Relational Diagram

- If we aren't interested in the data and just the design we use **relational diagrams**
- Note that typically diagrams would include the data types

FIGURE 2.4 A relational diagram



Tables

- Again..
 - Rows are entities
 - Columns are attributes
- A table should contain a set of related entities
 - The Parts table should not have a customer entity in it
- All we concern ourselves with is tables and their relationships, this is why Relational DBs have structural independence

SQL

- RDBs also introduced **Structured Query Language (SQL)**, which is really three parts
 - End-user interface – a GUI way of generating SQL queries, code is generated rather than programmed
 - The collection of tables – all data is perceived to be in tables, regardless of their physical locations, and are simply an easy way for users to understand the data; tables are independent and tables are related by common attributes
 - SQL engine – processes the user requests (both as SQL code or auto-generated from the interface), the implementation and execution don't matter, thus it is a **declarative language**

The Entity Relationship Model

- Relational DBs are good, but aren't great for modeling the design
- Thus Entity Relationship Model (ERM) is born
- Designed to be a graphical tool for showing entities and relationships
- More or less is used to complement a Relational DB

Model Components

- Entities – anything about which data are collected and stored
 - In ERM represented by a rectangle with name in caps inside in its singular form
 - Tables from relational models are entities in ERM
 - Rows of a relational table are called **entity instances**
 - **Entity Set** versus **Entity Instance/Occurrence**
 - Entities, much like before, have **attributes**

Model Components

- Relationships – associations between data
 - ERM uses the term **connectivity** when there is an association between *two* entities
 - Normal types: 1:1, 1:M, M:N
 - These are typically labeled based on the verb used to describe the connection

Entity Relationship Diagram

- Entities are rectangles, connections are lines
- There are two types of doing it though
 - Chen Notation
 - Crow's Foot

ERD Examples

FIGURE 2.5 The Chen and Crow's Foot notations

Chen Notation

A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.



Crow's Foot Notation



A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.



A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.



Object-Oriented Model

- Instead of entities and relationships, have an ***object*** model both within itself
 - Similar to how objects in OO programming have both data and methods
- Considered a ***semantic data model*** because of the meaning created by this combination
- Later the ability to store ***operations*** inside the object was also added
 - Operations could be things like modify its data, access its data, or printing itself
- Thus, objects are said to be ***self-contained***

OO Components

- An object is roughly equivalent to an instance of an ER entity or RM table
- Attributes described the properties of the object
- Objects that share similar characteristics are grouped into **classes**
 - Similar to the ER entity set
 - Different in that classes have methods
- Classes have **inheritance** and can be thought of as being in a tree structure

Unified Modeling Language (UML)

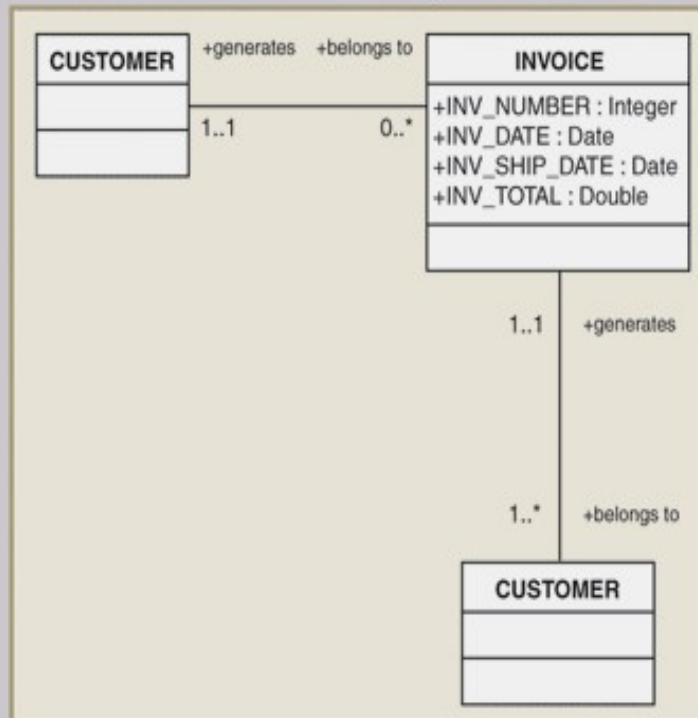
- Yet another diagramming system

FIGURE 2.6 A comparison of OO, UML, and ER models

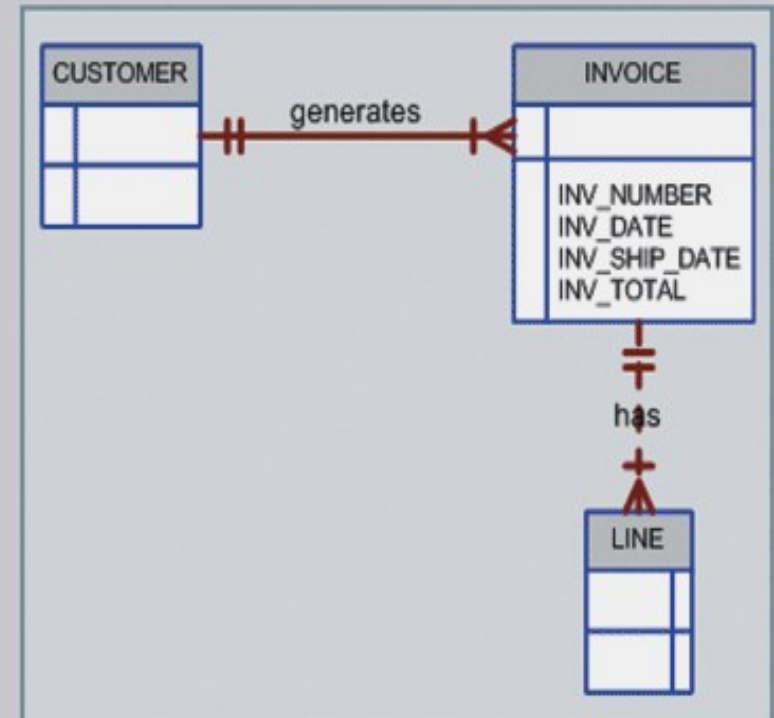
Object representation



UML class diagram



ER model



Getting practical

- RM is how we mostly think of databases
 - Extended RM (ERM) was created to address OO's challenge and is the typical implementation
- OO and RM are adopting ideas from each other
- Pure OO is not really used other than in academic or research settings
- The Internet is shaping the future of databases, especially XML interfaces

Good Summary of the Models

The figures on pages 47 & 48 in your book

Proper Abstraction

- Remember pulling out the state and zip from the addresses last time?
 - This was an appropriate level of abstraction (normalization really), if we tried any more it would be cumbersome
- There are many different data models because there are many ways to abstract data and by how much
- ANSI defined a framework for data abstraction that has three levels

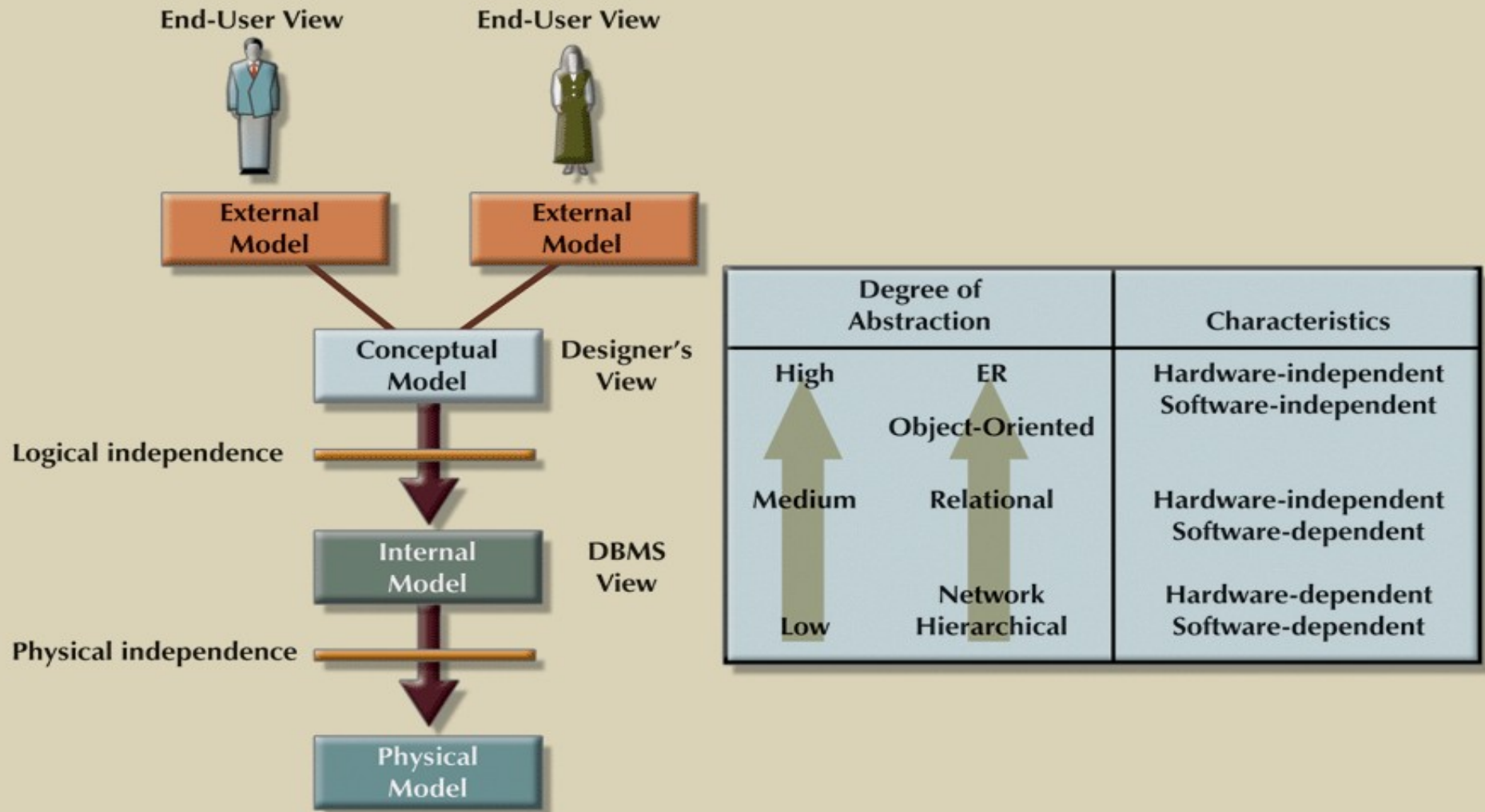
The Car Example

- Car Design can be thought of in three stages
 - Concept
 - Engineering
 - Production
- The level of abstraction changes in each
 - Do the same thing when designing a database

Data Abstraction Levels

FIGURE 2.8

Data abstraction levels

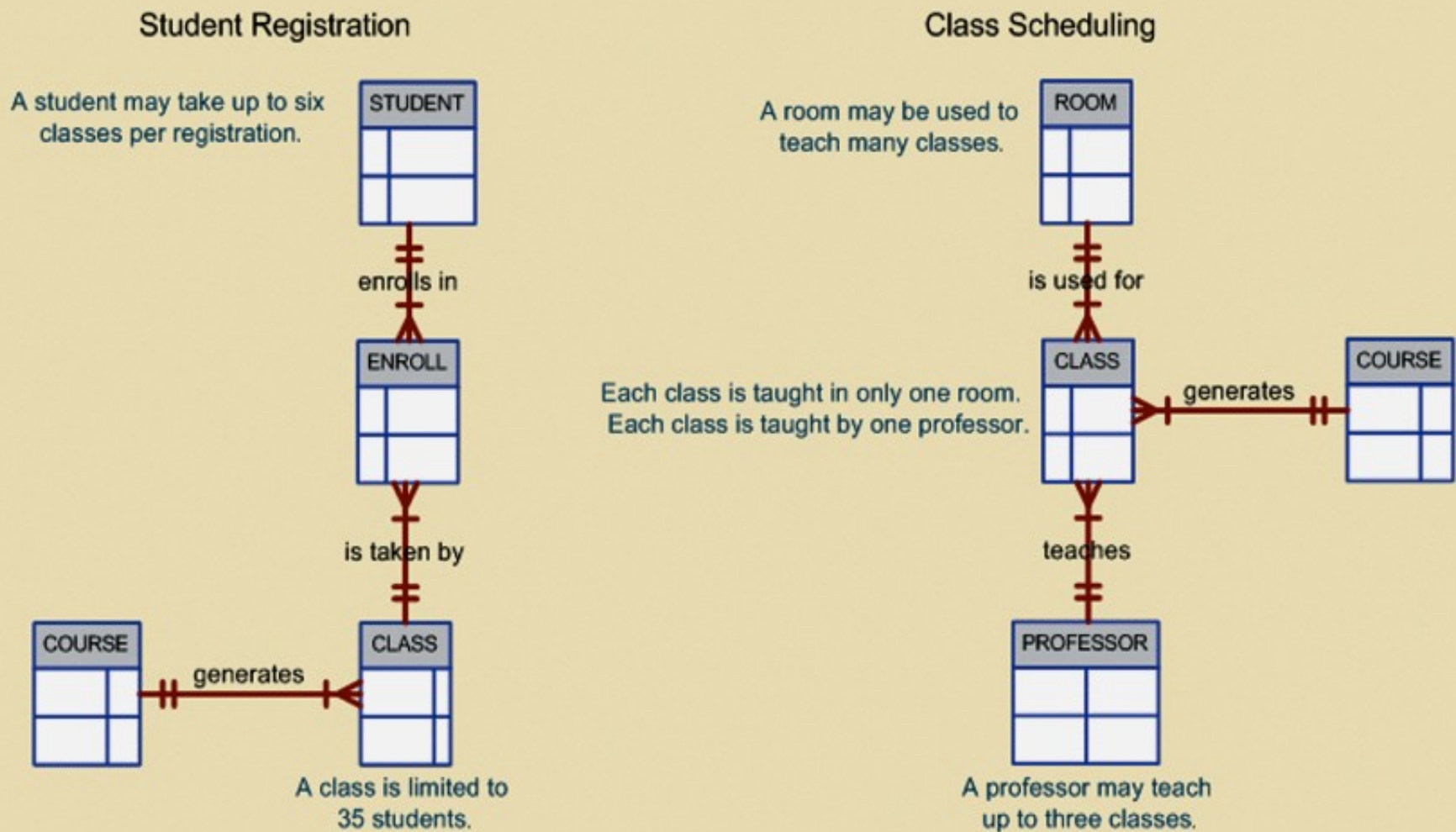


ANSI Level 1 – External Model

- This is how "external people" (re: users) think of the data
- Often these users do particular jobs and so only need a part of the data (*a subset*)
- These views are called ***schemas***
- Each schema contains all of the entities, relationships, and constraints needed to describe its environment

Example

FIGURE 2.9 External models for Tiny College



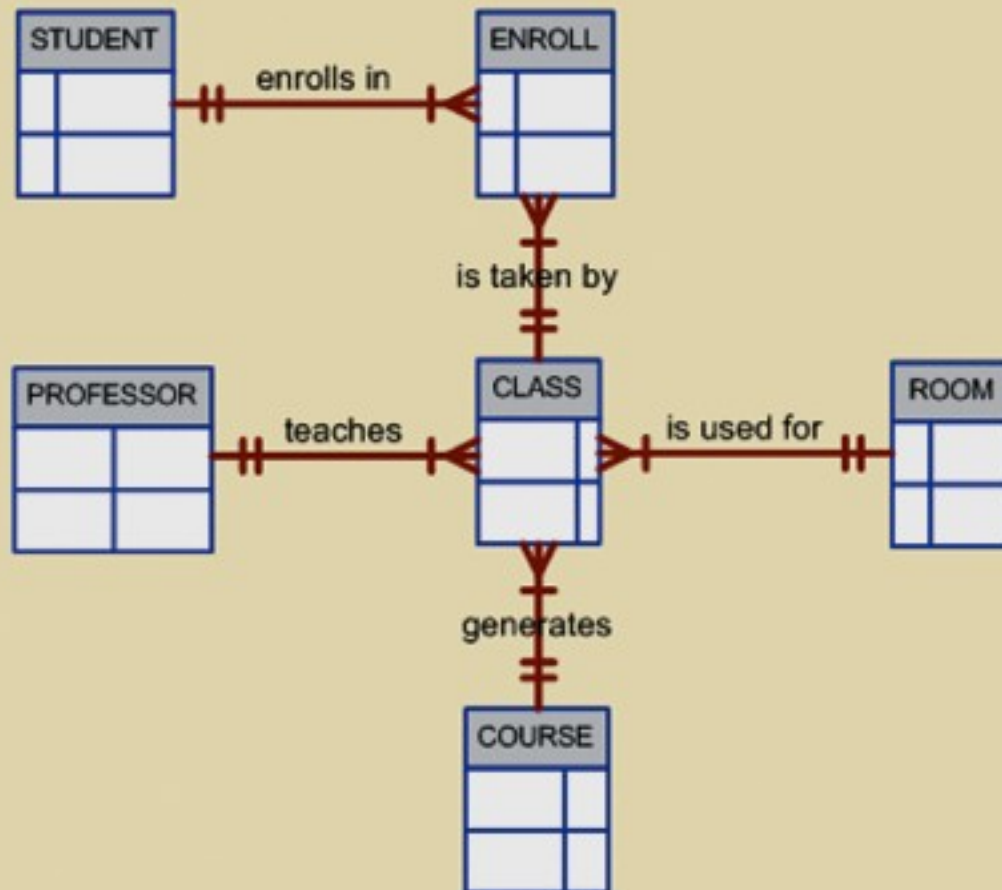
ANSI Level 2 - Conceptual

- The external subsets are independent yet share entities
- The conceptual level merges all these subsets into one global view of the database
- It is a logical design because it has neither software nor hardware independence

Example

FIGURE
2.10

Conceptual model for Tiny
College



ANSI Level 3 – Internal Model

- Takes the logical design of the conceptual model and turns it into an actual implementation as used by a DBMS
- Any model can be chosen as the implementation, most commonly this would be the RM/ERM with SQL specifications as a result
- At this point you have chosen the software implementation of the DBMS to go with, so it is ***software dependent***

Another Level: Physical Model

- **Not** part of the ANSI standard
- Describes the physical storage and access
- Both software and hardware dependent
- The modern data models (RM, ERM, OO) don't require you to worry about this, but the older ones did