9 - Database design for (web) applications

ASE230 – Server-Side Programming

Nicholas Caporusso

Objectives

- Capture key information from processes and formalize it
- Write database specifications

Agenda

1. Business rules
2. Data abstraction
3. Data models
4. From business rules to data models
5. The Entity-Relationship model

Business rules

Business rules: what

- Brief, precise, and unambiguous descriptions of a policies, procedures, or principles within a specific organization
- Apply to any organization that stores and uses data to generate information
- Description of operations that help to create and enforce actions within that organization's environment

4 types of business rules

- **Definitions**
  - specific words, phrases, terms and language used to express the rule (usually captured in a glossary)
- **Facts**
  - the starting points for applying the business rule. For example, it's a fact that a customer can apply for a loan, request a bank statement, or withdraw money. You can define facts as relationships, attributes, and structures.
- **Constraints**
  - limitations imposed upon the rule. For example, you need to have 10k in your account to apply for a loan. That's a constraint placed upon the customer.
- **Derivations**
  - how knowledge may be changed into other knowledge, possibly in different forms.

Business rules: how

- Must be rendered in writing
- Must be kept up to date
- Sometimes are external to the organization
- Must be easy to understand and widely disseminated
- Describe characteristics of the data as viewed by the company

Business rules: why

- Standardize company's view of data
- Constitute a communications tool between users and designers
- Allow designer to understand the nature, role, and scope of data
- Allow designer to understand business processes
- Allow designer to develop appropriate relationship participation rules and constraints
- Promote creation of an accurate data model

Business rules: example

- A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enroll in a particular course and take modules towards the completion that course. Each module is taught by a lecturer from the appropriate department, each lecturer tutors a group of students

Business rules: example

- A university consists of a number of **departments**. Each department offers several **courses**. A number of **modules** make up each course. **Students** enroll in a particular course and take modules towards the completion that course. Each module is taught by a **lecturer** from the appropriate department, each lecturer tutors a group of students

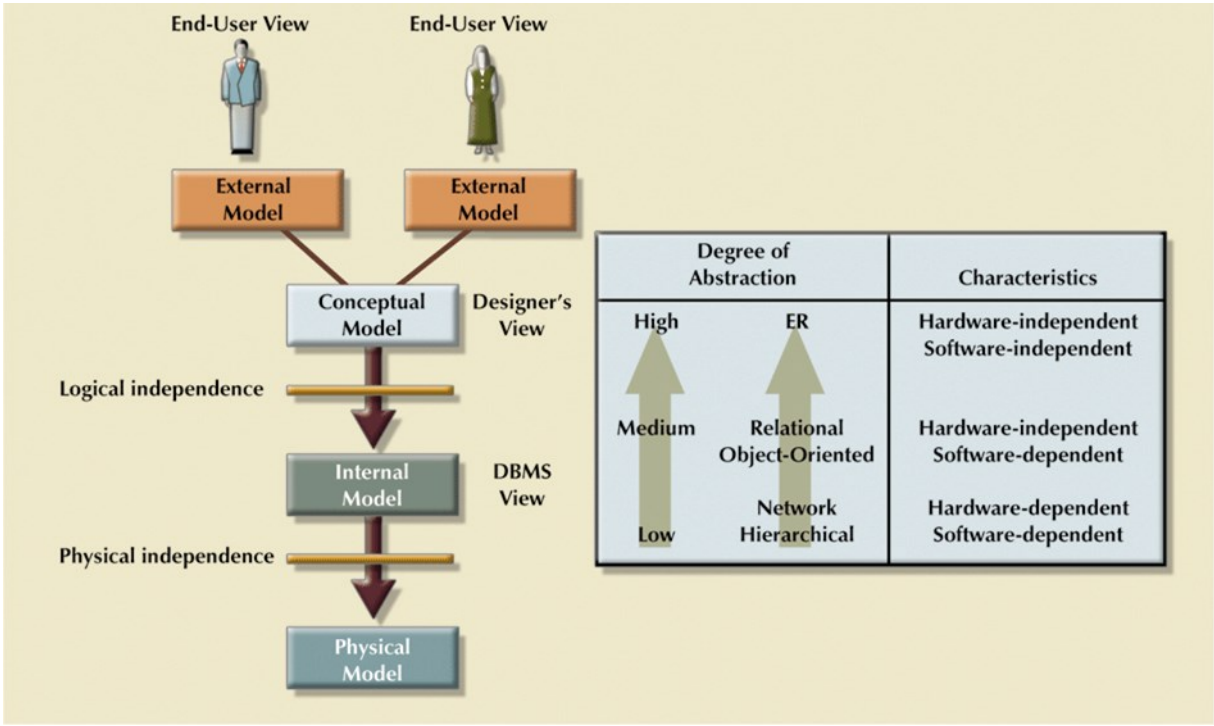
Business rules: example

- A university consists of a number of departments. Each department **offers** several courses. A number of modules **make up** each course. Students enroll in a particular course and **take** modules towards the completion that course. Each module is **taught by** a lecturer **from the** appropriate department, each lecturer **tutors** a group of students

Data abstraction

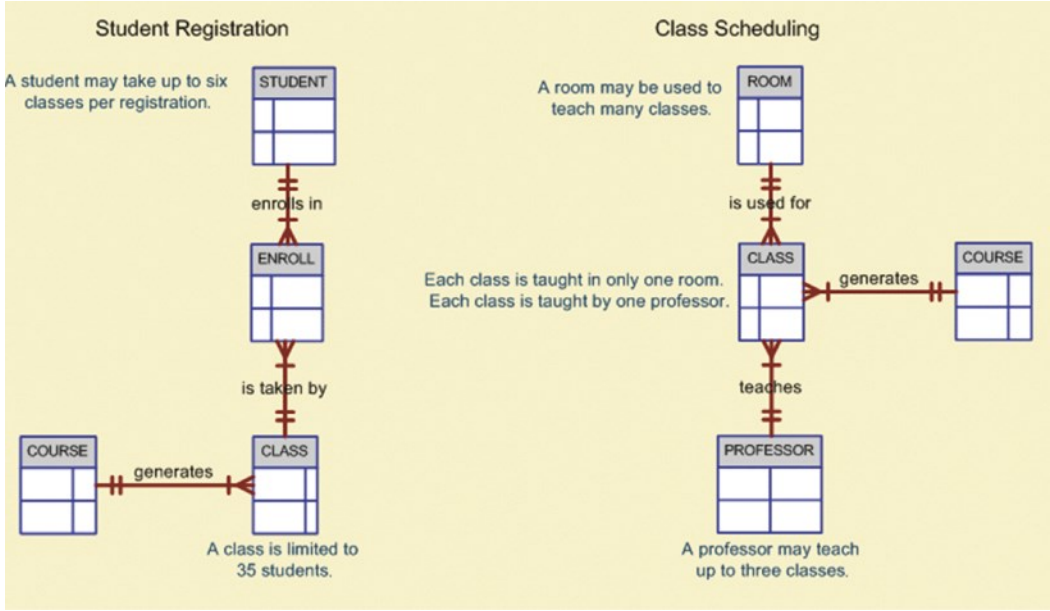
What is data abstraction?

- Way of classifying data models
- Many processes begin at high level of abstraction and proceed to an ever-increasing level of detail
- Designing a usable database follows the same basic process
- American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC)
- Defined a framework for data modeling based on degrees of data abstraction (1970s)
  - External
  - Conceptual
  - Internal



The external model

- End users' view of the data environment
- Requires that the modeler subdivide set of requirements and constraints into functional modules that can be examined within the framework of their external models

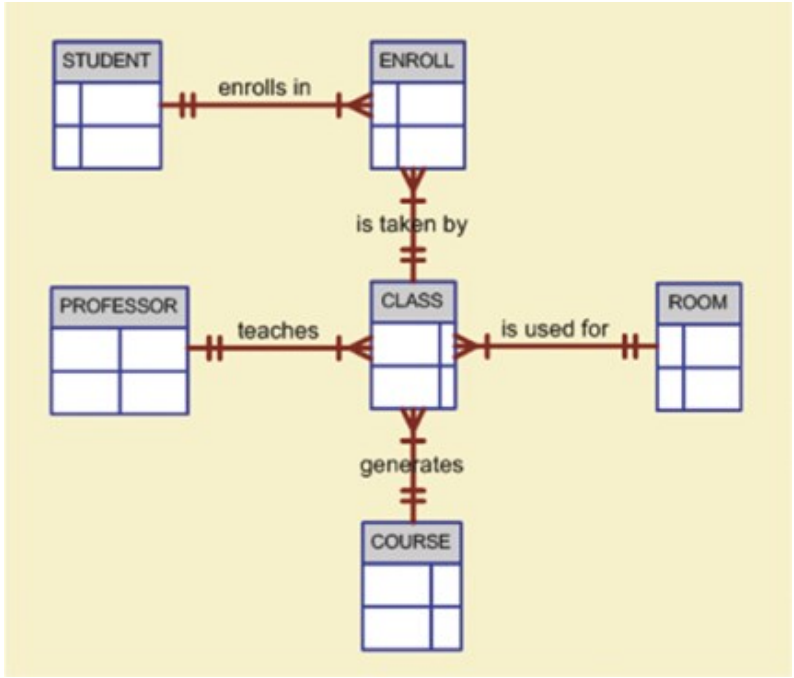


Advantages of the external model

- Easy to identify specific data required to support each business unit's operations
- Facilitates designer's job by providing feedback about the model's adequacy
- Creation of external models helps to ensure security constraints in the database design
- Simplifies application program development

The conceptual model

- Represents global view of the entire database
- Representation of data as viewed by the entire organization
- Basis for identification and high-level description of main data objects, avoiding details
- Most widely used conceptual model is the entity relationship (ER) model

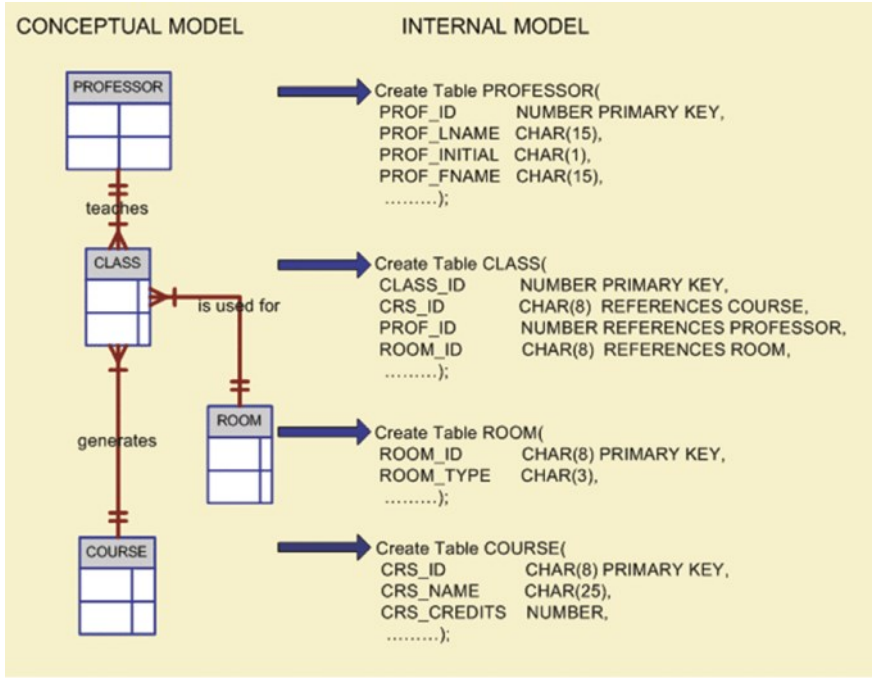


Advantages of the conceptual model

- Provides a relatively easily understood macro level view of data environment
- Independent of both software and hardware
  - does not depend on the DBMS software used to implement the model
  - does not depend on the hardware used in the implementation of the model
  - changes in either hardware or DBMS software have no effect on the database design at the conceptual level

The internal model

- Representation of the database as “seen” by the DBMS
- Maps the conceptual model to the DBMS
- Internal schema depicts a specific representation of an internal model



The physical model

- Operates at lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes
- Software and hardware dependent
- Requires that database designers have a detailed knowledge of the hardware and software used to implement database design

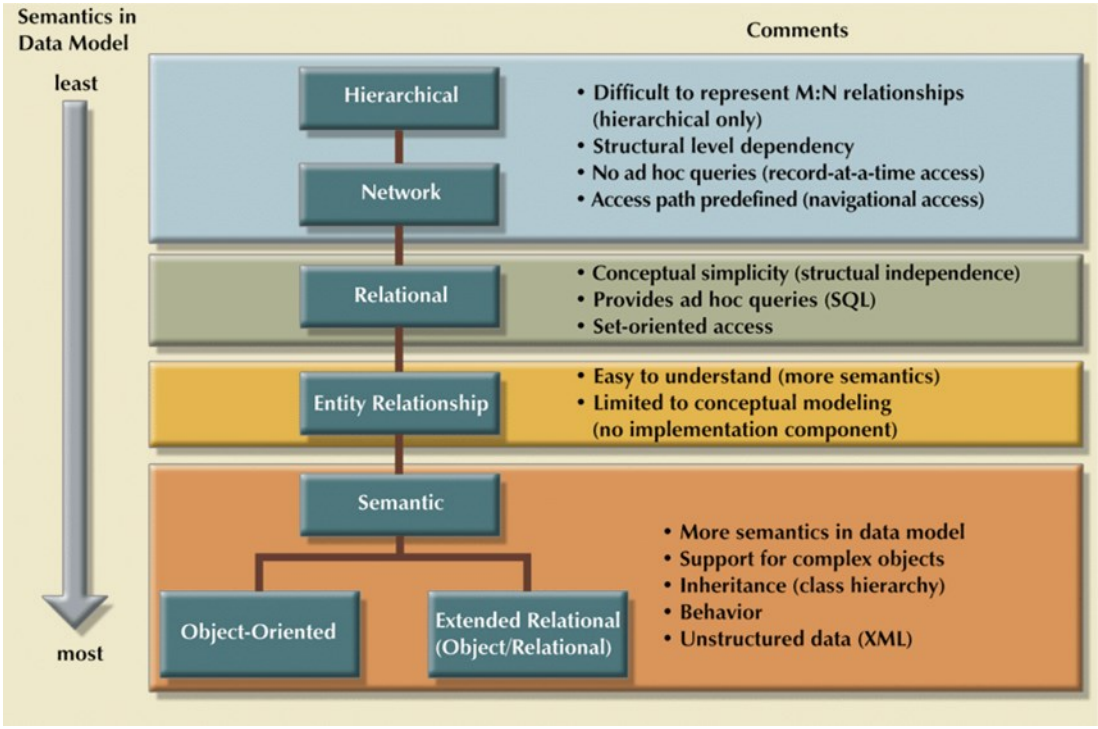
Models

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual		Global view of data (independent of database model)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Data models

The Evolution of Data Models

- Hierarchical model
  - Depicts a set of one-to-many (1:M) relationships between a parent and its children segments
- Network data model
  - Uses sets to represent 1:M relationships between record types
- Relational model
  - Current database implementation standard
  - ER model is a popular graphical tool for data modeling that complements the relational model
- Entity relationship
- Object oriented (OO)



The Relational Model

- Developed by Codd (IBM) in 1970
- Considered ingenious but impractical in 1970
- Conceptually simple
- Computers lacked power to implement the relational model
- Today, microcomputers can run sophisticated relational database software

The Relational Model

- Relational Database Management System (RDBMS)
- Performs same basic functions provided by hierarchical and network DBMS systems, in addition to a host of other functions
- Most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the user

Elements of the Relational model

- Relational diagram
  - Representation of relational database's entities, attributes within those entities, and relationships between those entities
- Relational Table
  - Stores a collection of related entities: resembles a file
- Relational table is purely logical structure
  - How data are physically stored in the database is of no concern to the user or the designer
  - This property became the source of a real database revolution

The Object Oriented Model (1/3)

- Modeled both data and their relationships in a single structure known as an object
- Object-oriented data model (OODM) is the basis for the object-oriented database management system (OODBMS)
- OODM is said to be a semantic data model

The Object Oriented Model (2/3)

- Object described by its factual content
  - Similar to an entity in the relational model
- Includes information about relationships between facts within object, and relationships with other objects
  - Unlike relational model's entity
- Subsequent OODM development allowed an object to also contain all operations
- Object becomes basic building block for autonomous structures

The Object Oriented Model (3/3)

- Object is an abstraction of a real-world entity
- Attributes describe the properties of an object
- Objects that share similar characteristics are grouped in classes
- Classes are organized in a class hierarchy
- Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of classes above it

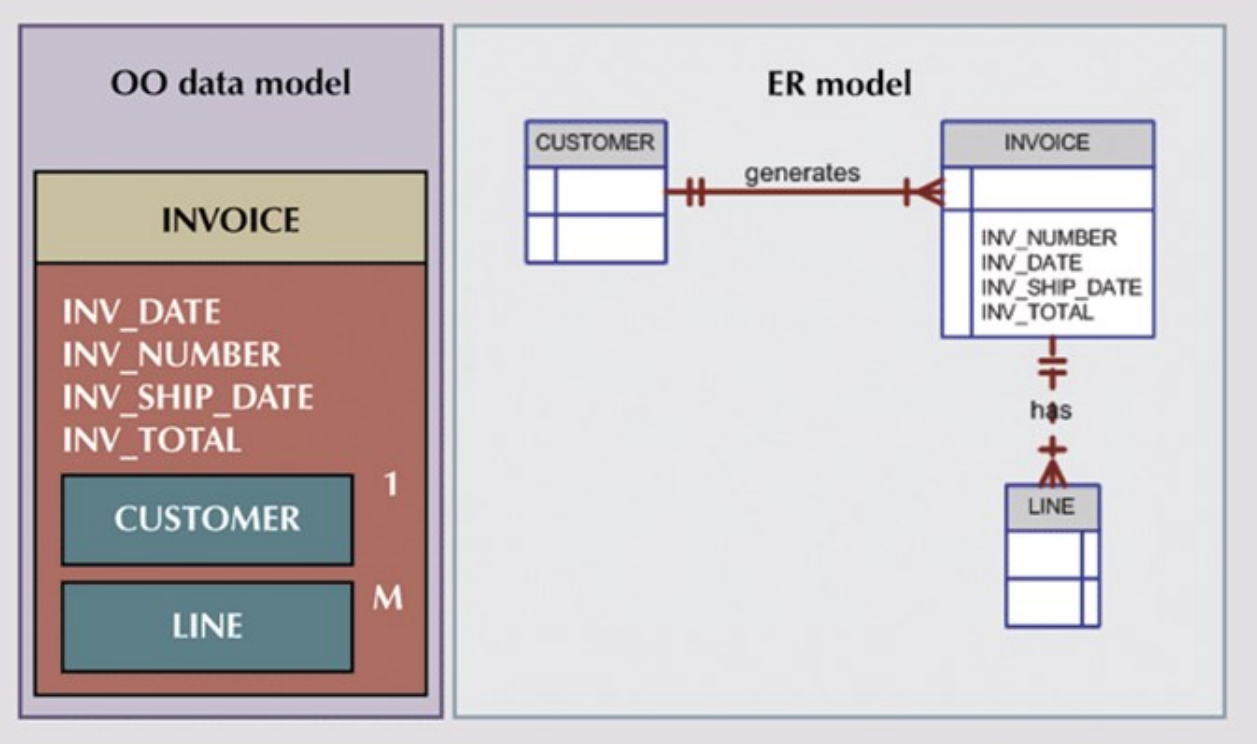
Extended Relational Data Model (ERDM)

- Semantic data model developed in response to increasing complexity of applications
- DBMS based on the ERDM often described as an object/relational database management system (O/RDBMS)
- Primarily geared to business applications

Database Models and the Internet

- Internet drastically changed role and scope of database market
- OODM and ERDM-O/RDM have taken a backseat to development of databases that interface with Internet
- Dominance of Web has resulted in growing need to manage unstructured information

Differences between ER model and OO model



From business rules to data models

Requirements collection and analysis

- Database designers interview prospective database users to understand and document data requirements
- Result: data requirements
- Functional requirements of the application

Generating entities

- Generally, nouns translate into entities
- Verbs translate into relationships among entities
- Relationships are bi-directional

Conceptual design

- Conceptual schema
- Description of data requirements
- Includes detailed descriptions of the entity types, relationships, and constraints
- Transformed from high-level data model into implementation data model

Logical and physical design

- Logical design or data model mapping
  - Result is a database schema in implementation data model of DBMS
- Physical design phase
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

The Entity Relationship Model

What is a data model

- A data model is a (relatively) simple abstraction of a complex real-world data environment
- Basic data modeling components are:
  - Entity - anything about which data are to be collected and stored
  - Attribute - a characteristic of an entity
  - Relationship - describes an association among entities
    - One-to-many (1:M) relationship
    - Many-to-many (M:N or M:M) relationship
    - One-to-one (1:1) relationship
  - Constraint - a restriction placed on the data

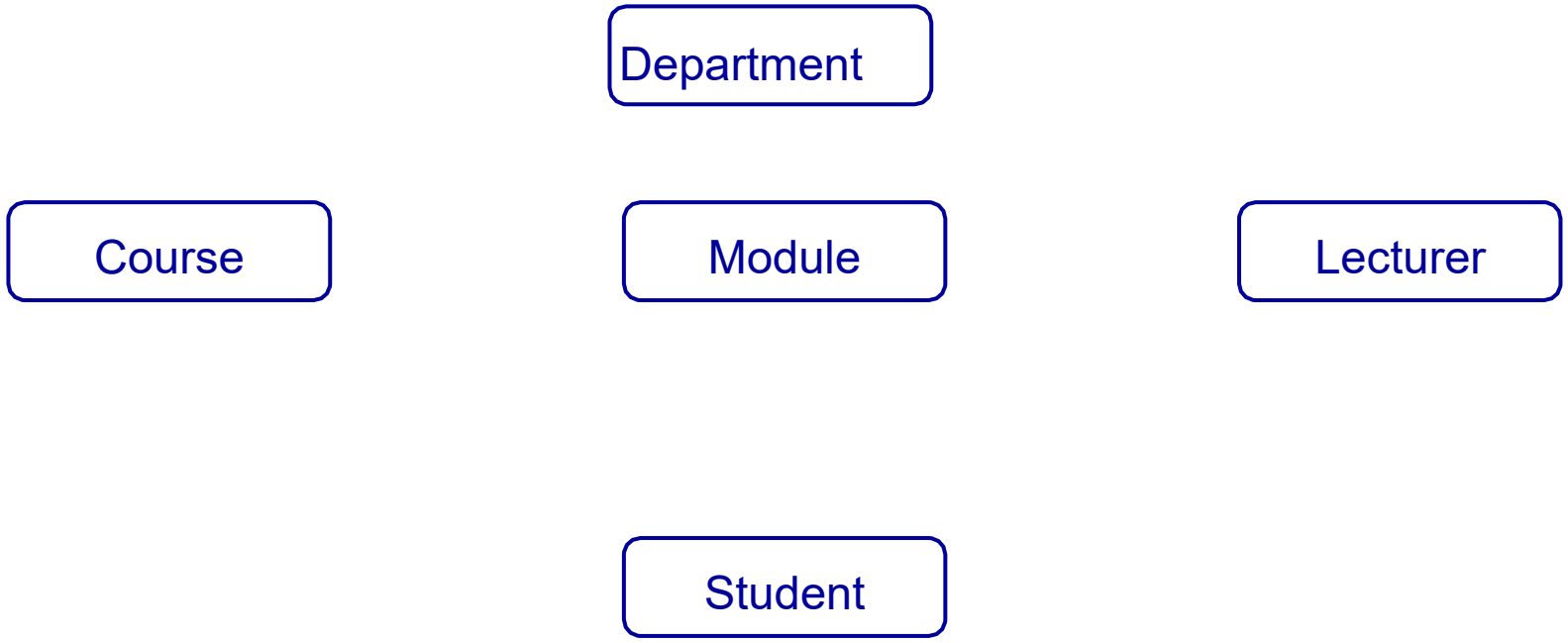
The Importance of Data Models

- Relatively simple representations, usually graphical, of complex real-world data structures
- Facilitate interaction among the designer, the applications programmer, and the end user
- End-users have different views and needs for data
- Data model organizes data for various users

Entity Relationship Diagram

- Widely accepted and adapted graphical tool for conceptual data modeling
- Introduced by Chen in 1976
- Graphical representation of entities and their relationships in a database structure
  - Entity set is collection of like entities
  - Connectivity labels types of relationships
  - Diamond connected to related entities through a relationship line
- Entity-Relationship (ER) model: popular high-level conceptual data model
- ER diagrams: diagrammatic notation associated with the ER model

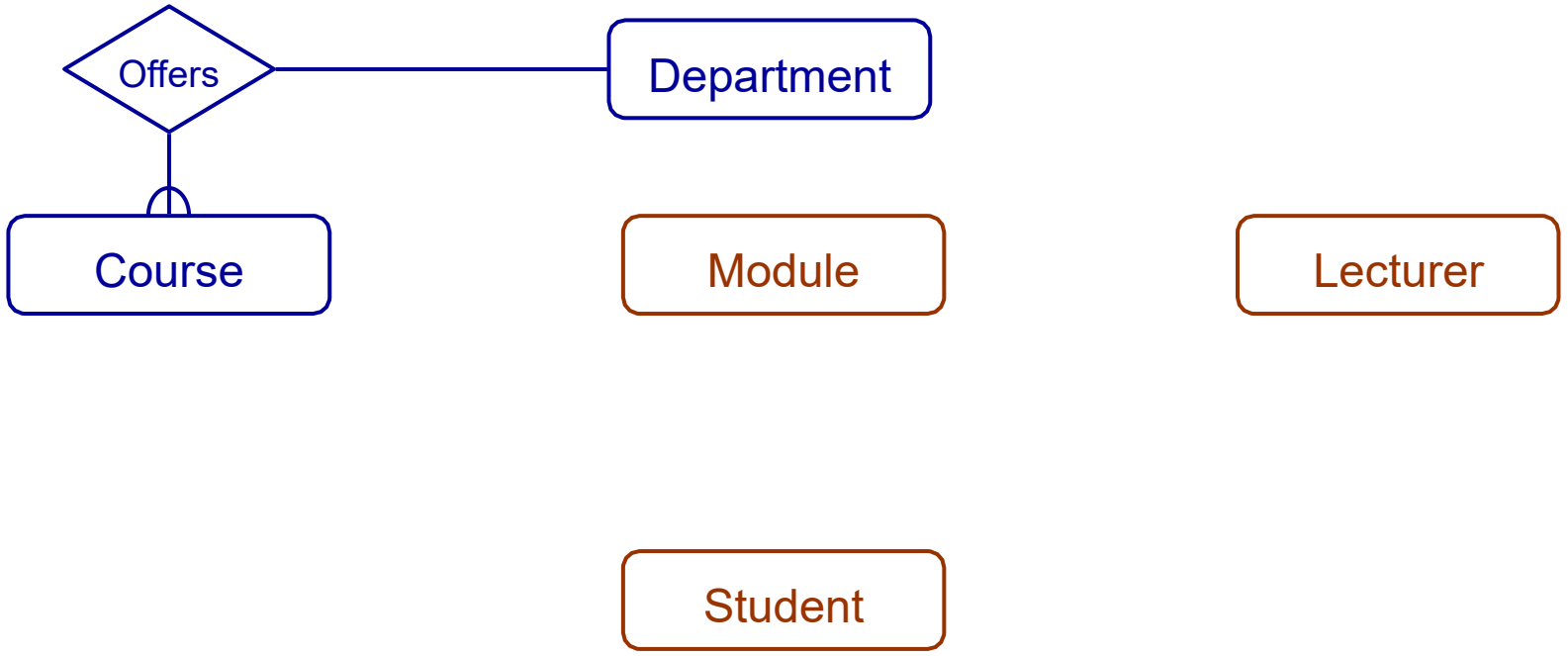
In-class exercise



Entity Relationship Modelling

In-class exercise

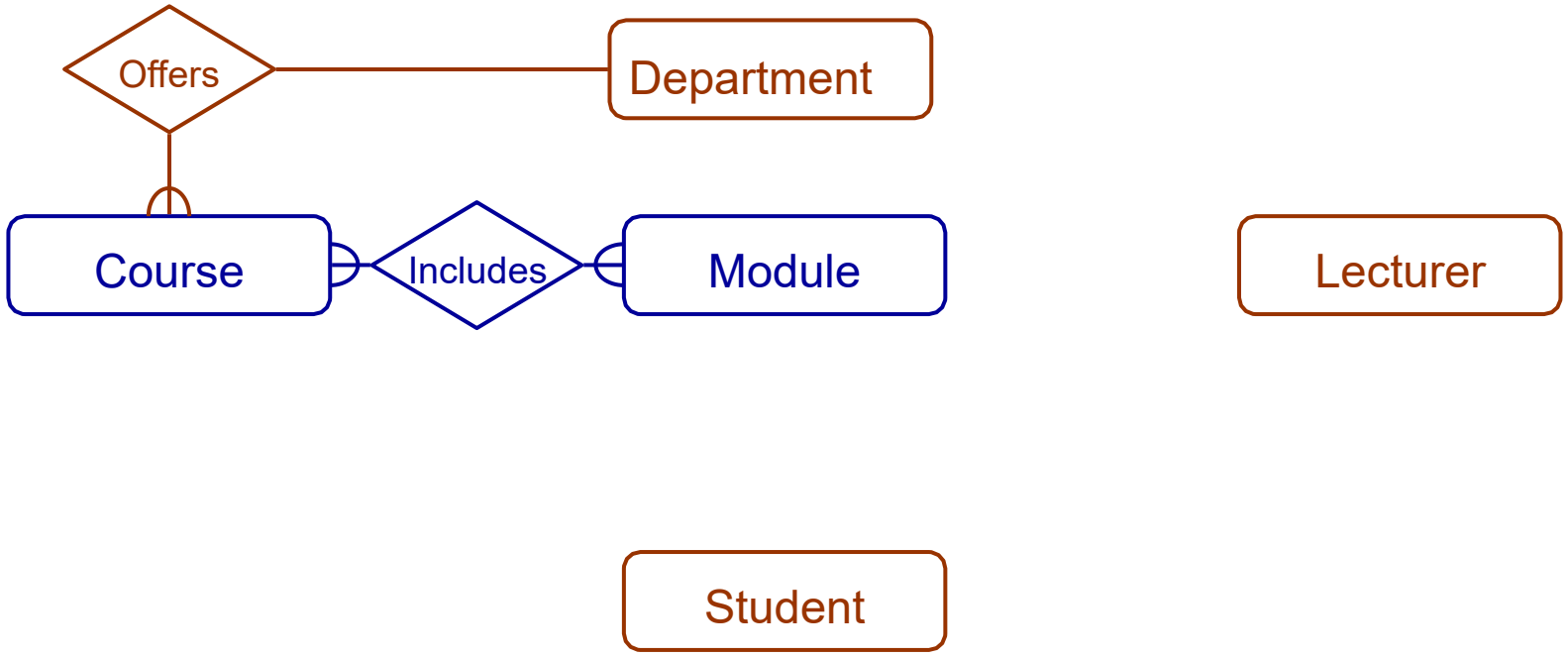
- Each department offers several courses



Entity Relationship Modelling

In-class exercise

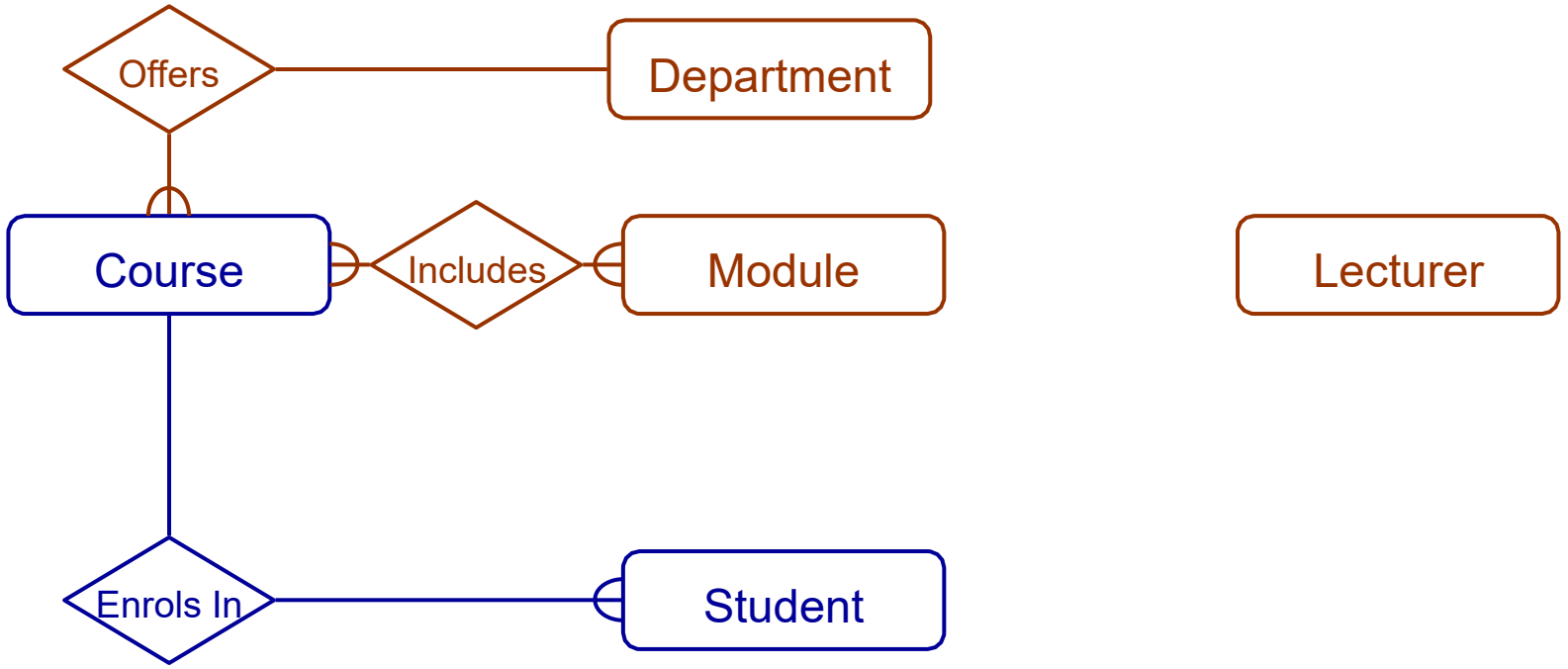
- Courses consist of a number of modules



Entity Relationship Modelling

In-class exercise

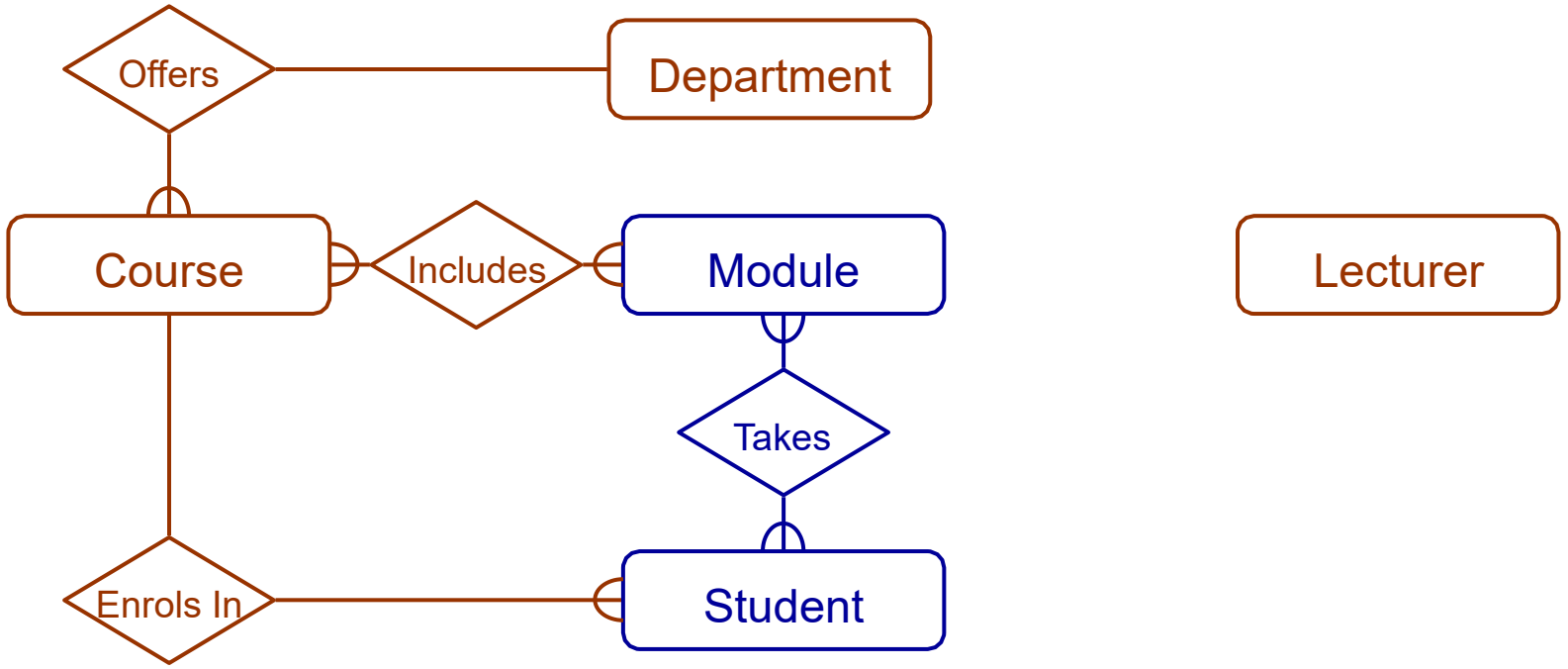
- Students enroll in a particular course



Entity Relationship Modelling

In-class exercise

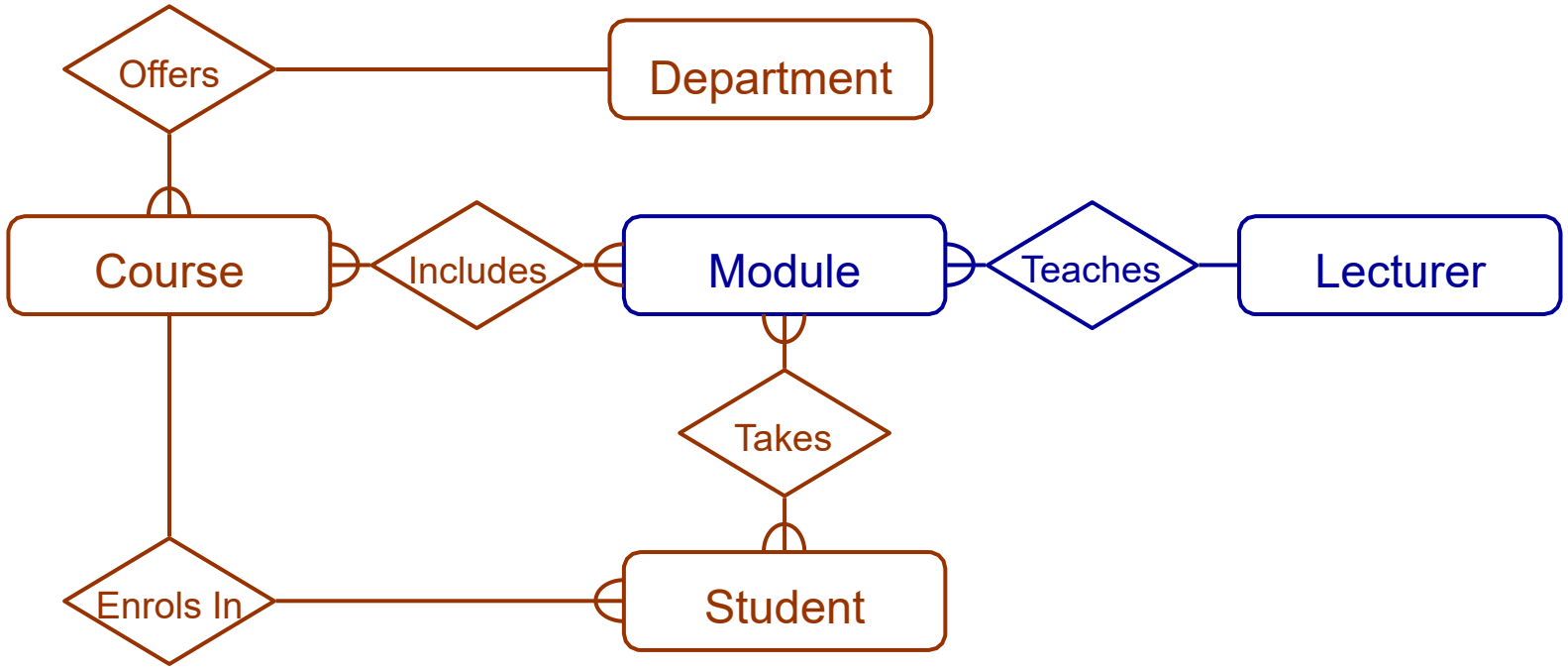
- Students take modules



Entity Relationship Modelling

In-class exercise

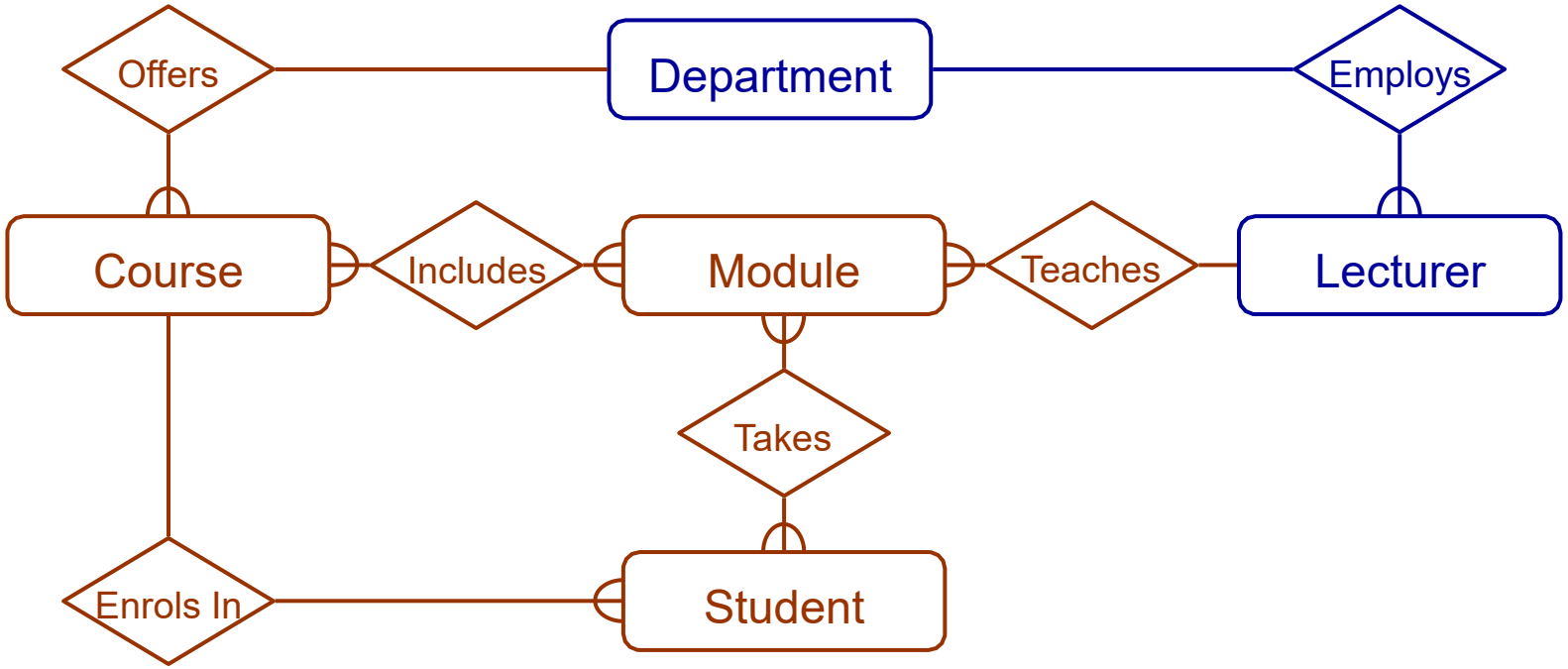
- Each module is taught by a lecturer



Entity Relationship Modelling

In-class exercise

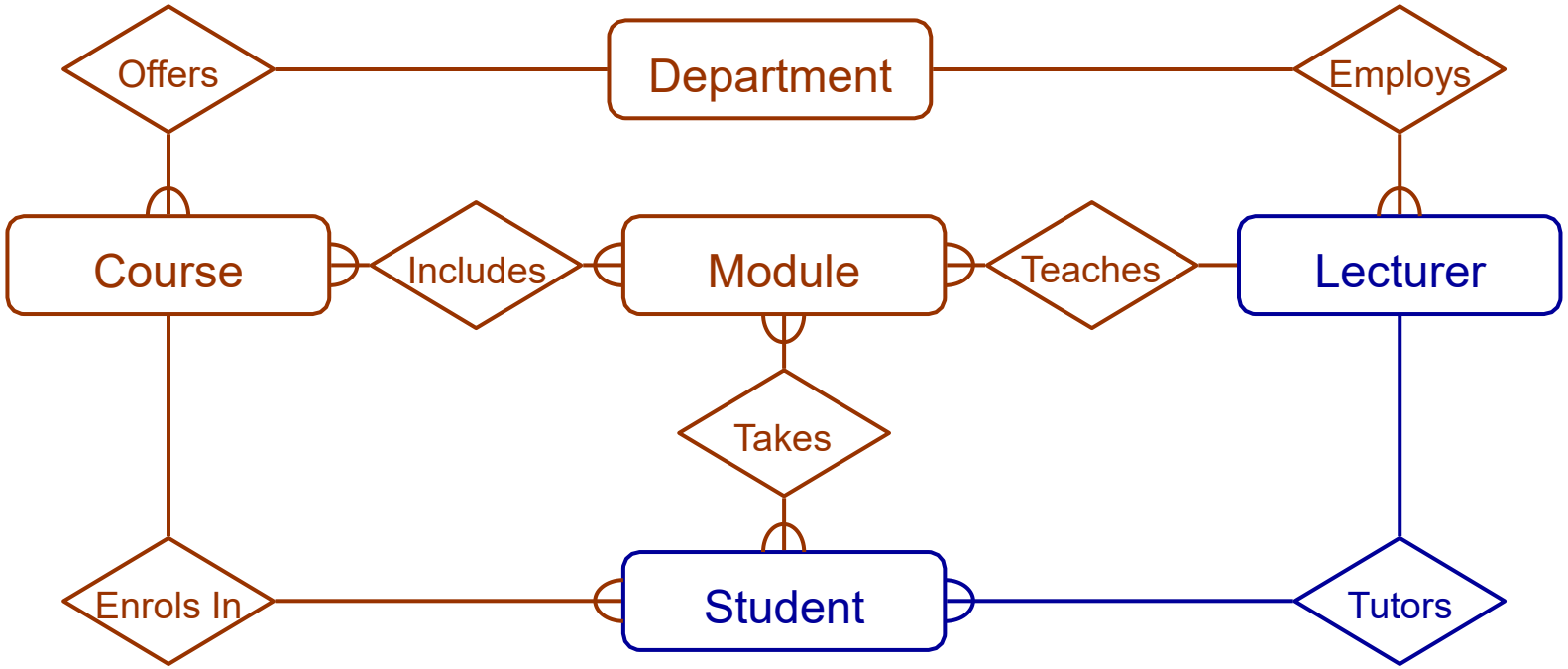
- A lecturer works in a department



Entity Relationship Modelling

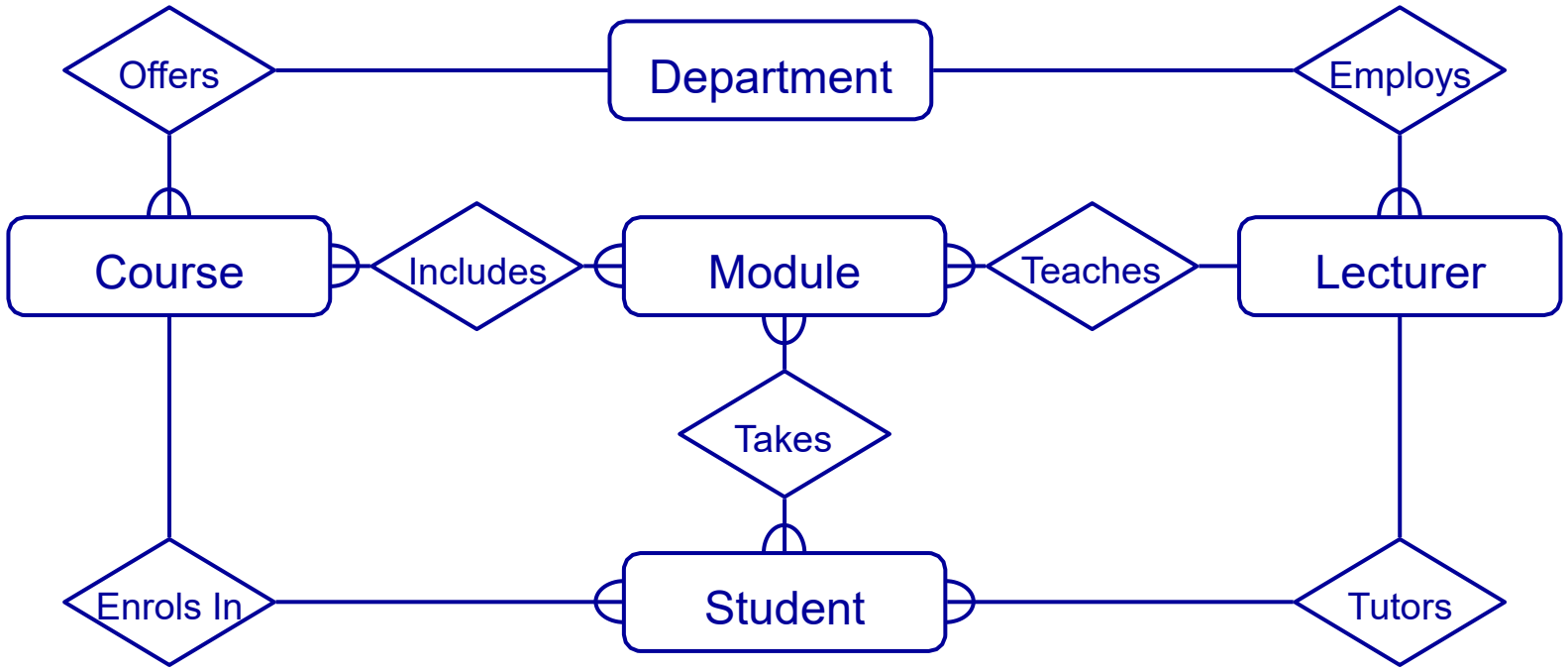
In-class exercise

- Each lecturer tutors a group of students



Entity Relationship Modelling

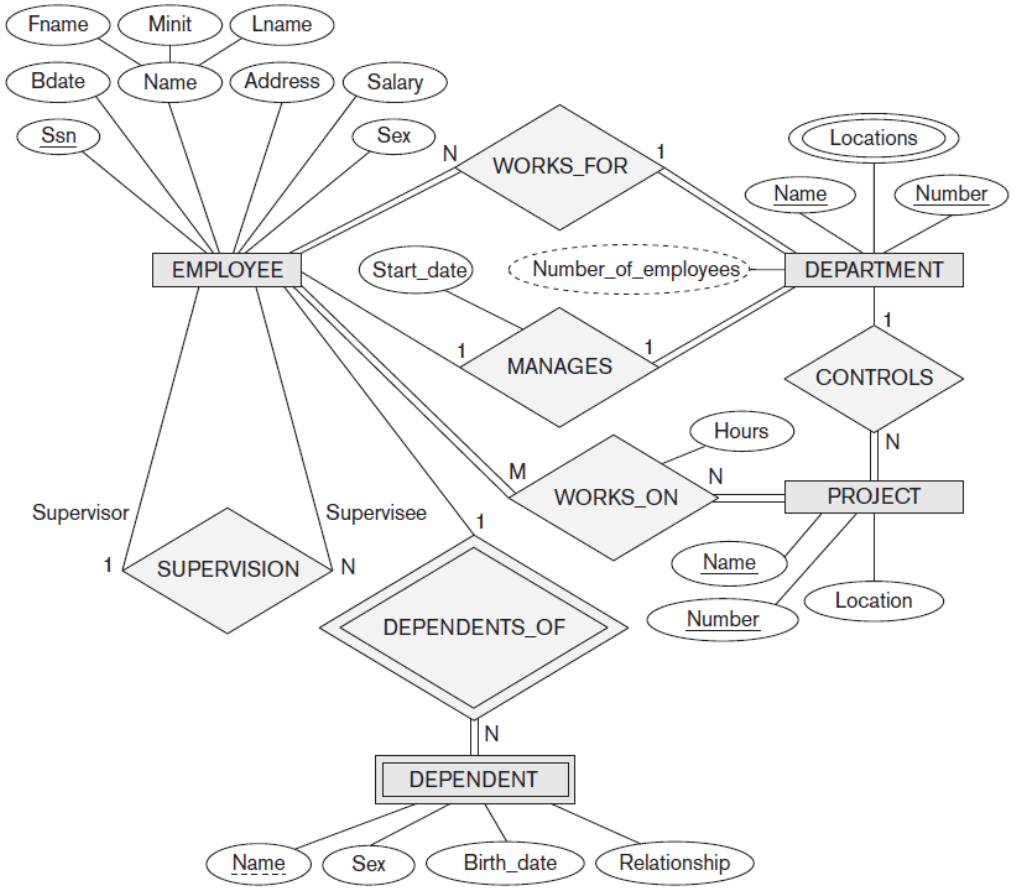
In-class exercise



Entity Relationship Modelling

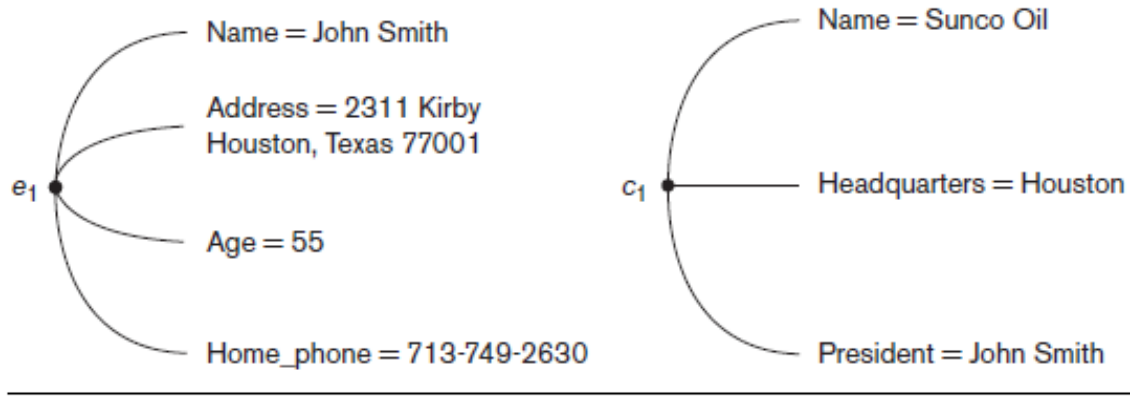
An other example

- COMPANY
  - Employees, departments, and projects
  - Company is organized into departments
  - Department controls a number of projects
  - Employee: store each employee's name, Social Security number, address, salary, sex (gender), and birth date
  - Keep track of the dependents of each employee



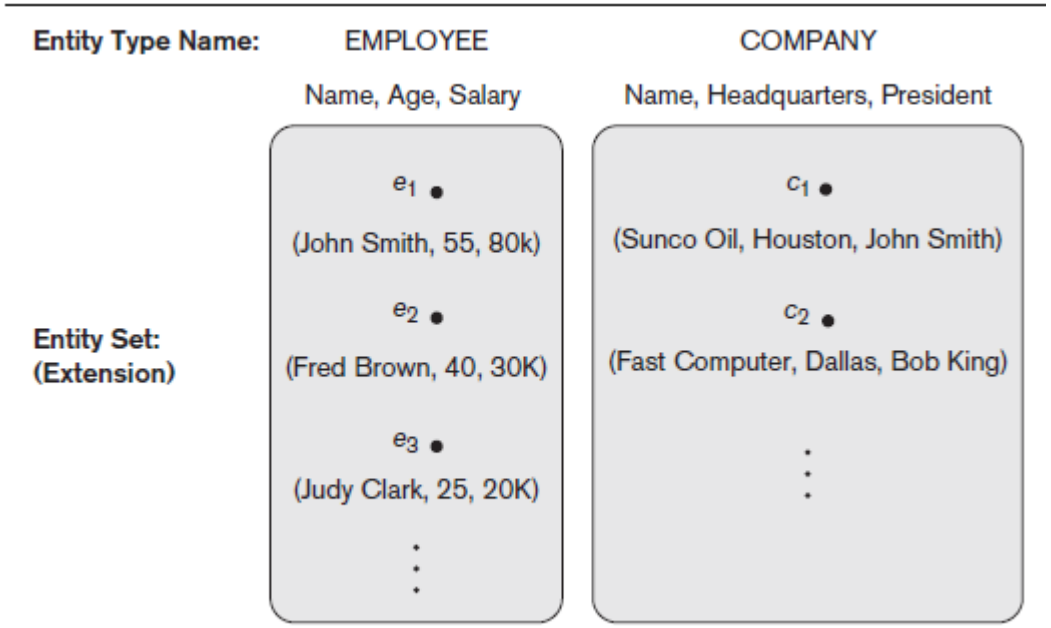
Entities and attributes

- Entity
  - Thing in real world with independent existence
- Attributes
  - Particular properties that describe entity
  - Types of attributes:
  - Composite versus simple (atomic) attributes
  - Single-valued versus multivalued attributes
  - Stored versus derived attributes
  - NULL values
  - Complex attributes

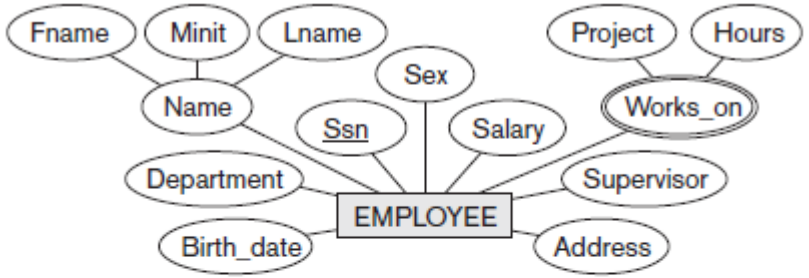


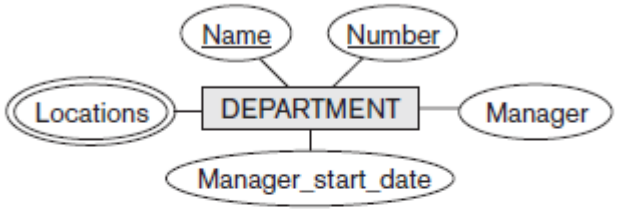
Entities, attributes, values

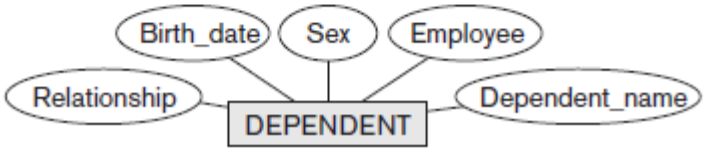
- Entities
  - Collection (or set) of entities that have the same attributes
- Key or uniqueness constraint
  - Attributes whose values are distinct for each individual entity in entity set
  - Key attribute: Uniqueness property must hold for every entity set of the entity type
- Value sets (or domain of values)
  - Specifies set of values that may be assigned to that attribute for each individual entity

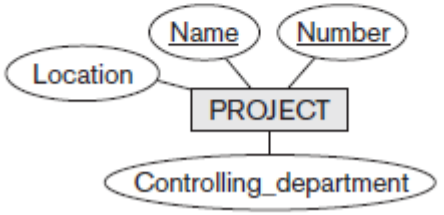


Entities and attributes (example)









Entities or attributes?

- Sometimes it is hard to tell if something should be an entity or an attribute
  - They both represent objects or facts about the world
  - They are both often represented by nouns in descriptions

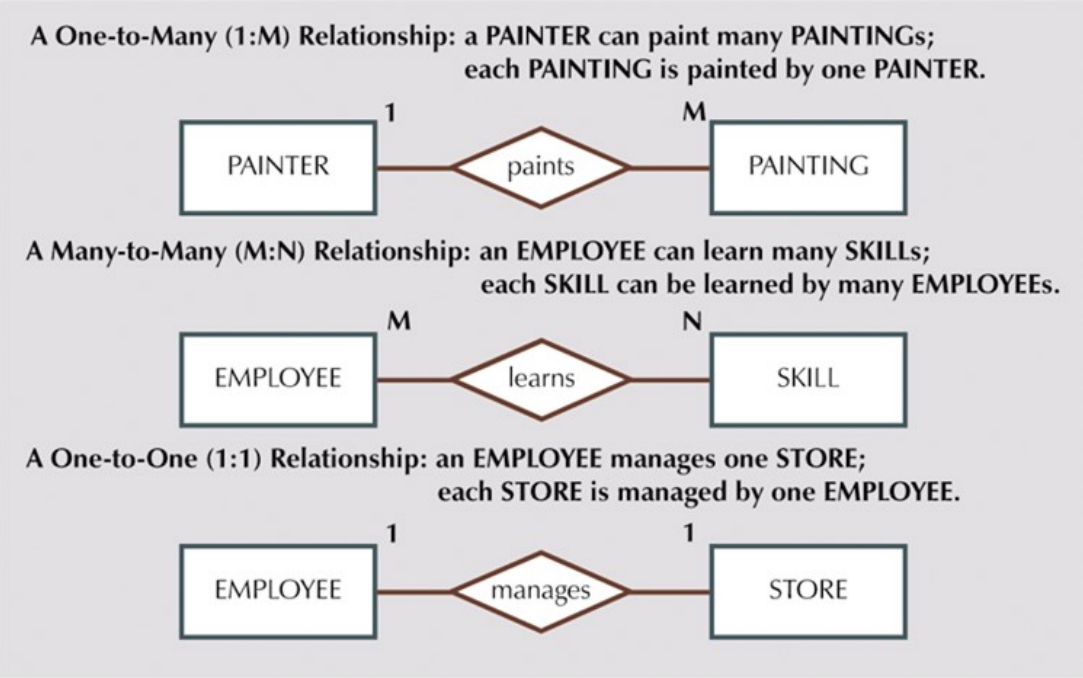
- General guidelines
  - Entities can have attributes but attributes have no smaller parts
  - Entities can have relationships between them, but an attribute belongs to a single entity

Relationships and cardinality

- Relationship
  - When an attribute of one entity type refers to another entity type
  - Represent references as relationships not attributes
- Relationship type R among n entity types E_1, E_2, ..., E_n
  - Defines a set of associations among entities from these entity types
- Relationship instances r_i
  - Each r_i associates n individual entities (e_1, e_2, ..., e_n)
  - Each entity e_j in r_i is a member of entity set E_j

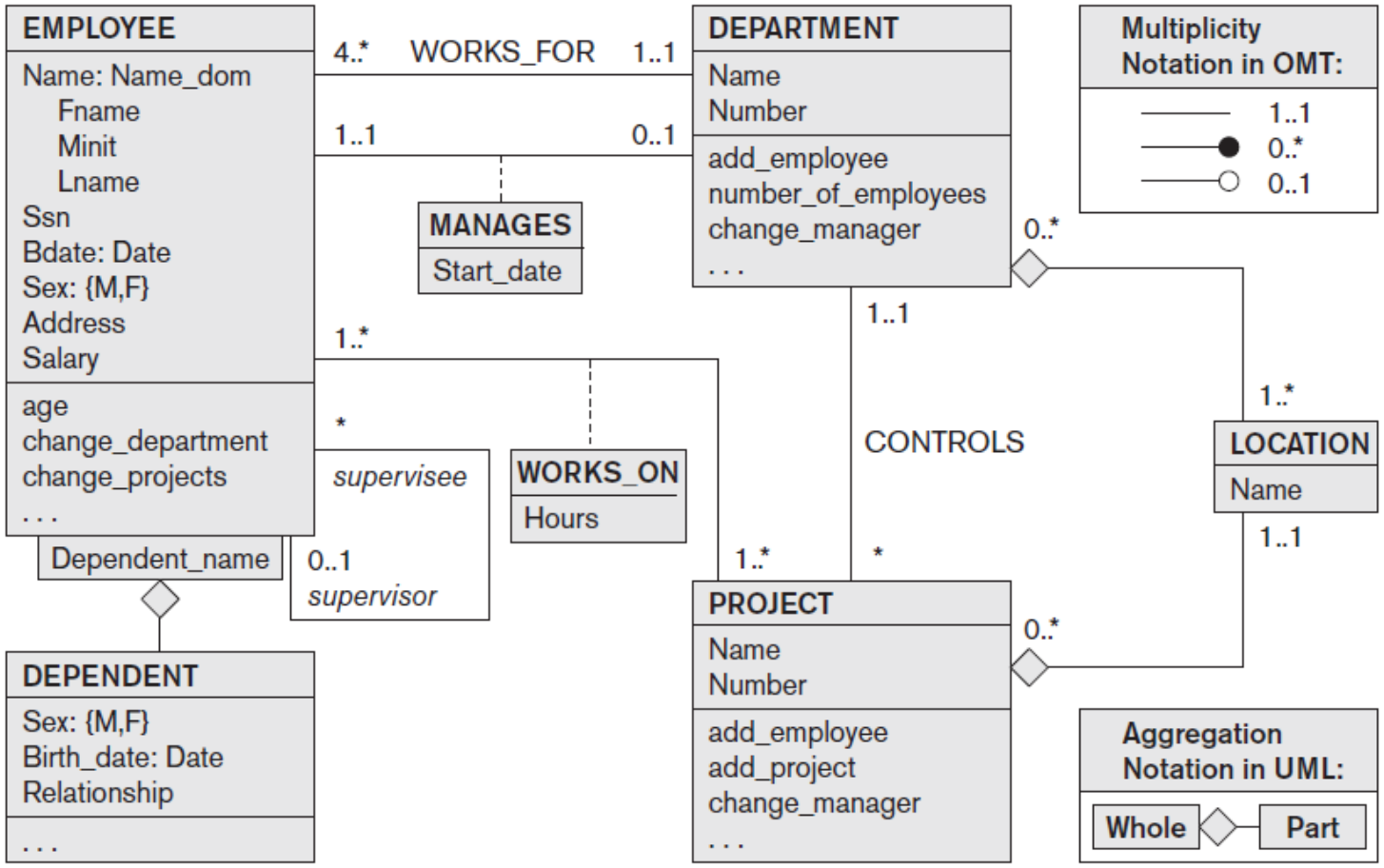
- Cardinality ratio for a binary relationship
  - Specifies maximum number of relationship instances that entity can participate in
- Participation constraint
  - Specifies whether existence of entity depends on its being related to another entity
  - Types: total and partial

Relationships



- Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
- For a 1:N relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For M:N relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes

UML modeling



Wrap-up

In-class exercise

- Design a database based on the following business rules
  - The university gym has many users who visit the gym at a given date and time and use the gym for some time
  - The gym offers different routines each involving specific types of equipment; users can train with a pre-defined routine, or they can define their own routine; both consist of exercises that use equipment
  - For each exercises, users track the time or number of repetitions, and the difficulty level (or kilograms)

Agenda

1. Database modeling in practice
2. Normalization

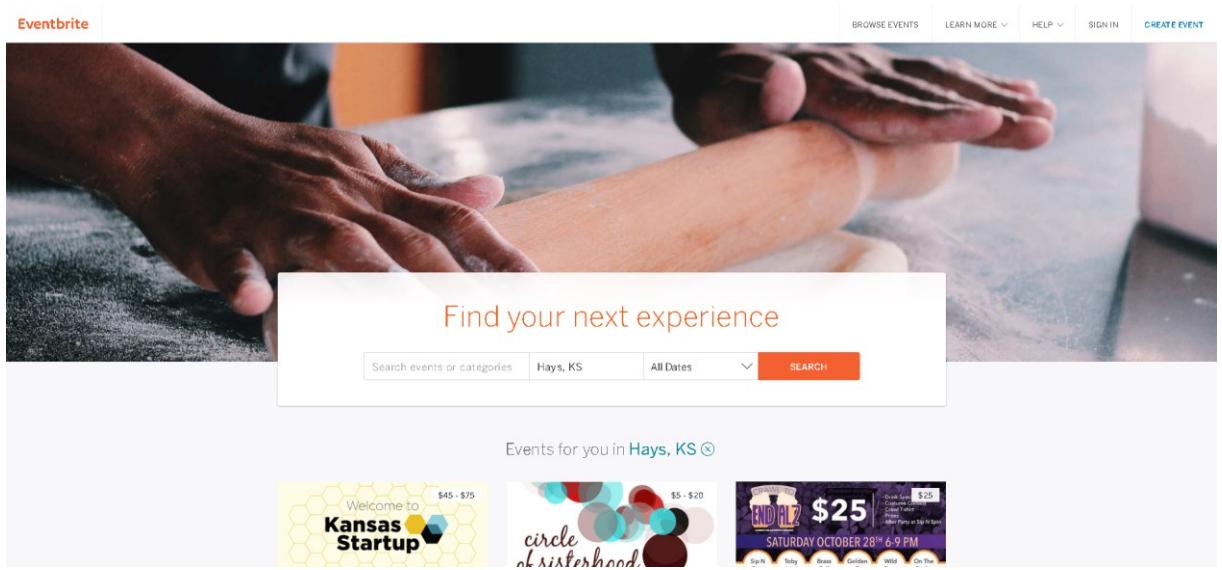
Database modeling in practice

Use case: an event website

- We are going to use an event website as a case for
  1. Defining the business rules
  2. Realizing the conceptual model
  3. Designing the Entity-Relationship model
  4. Defining the internal model of the database

Events

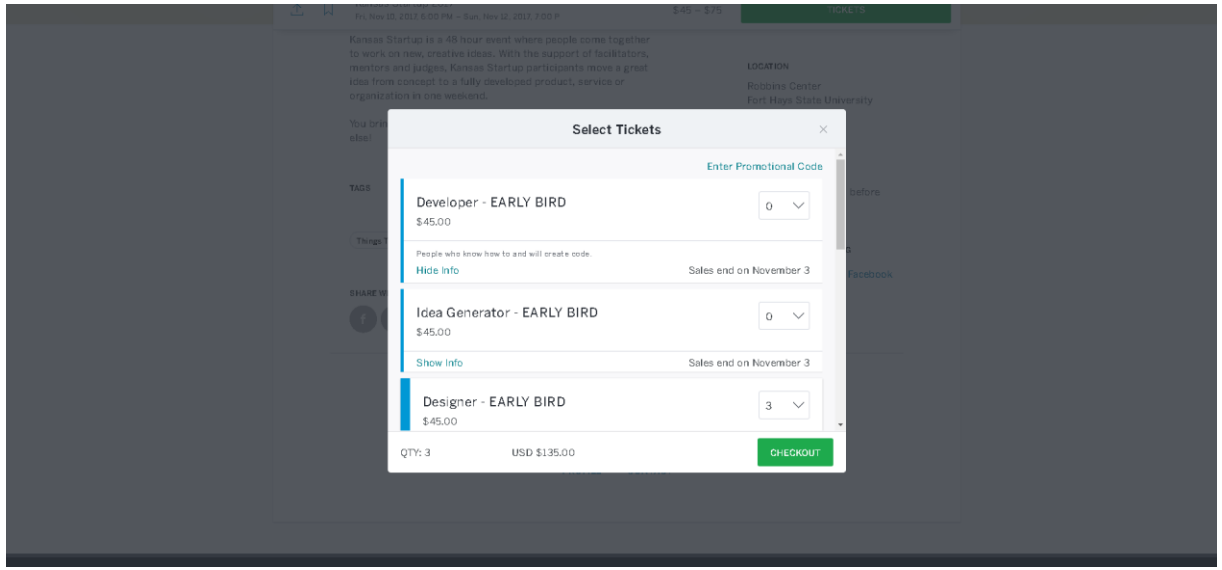
Home page



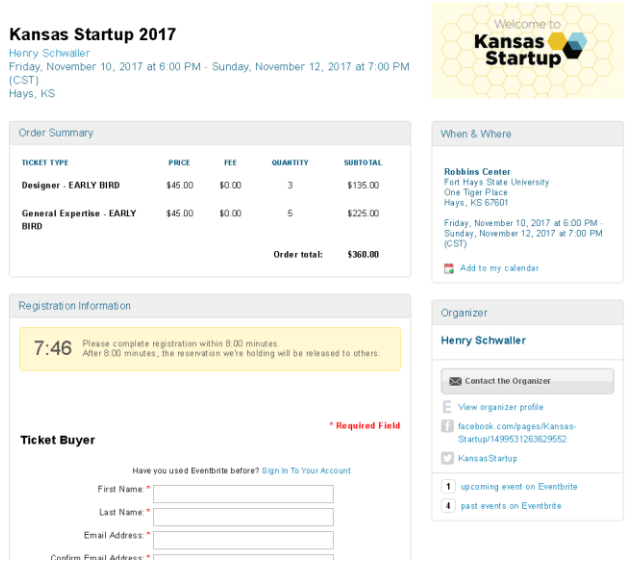
Events page

Buying tickets

Tickets page



Checkout page



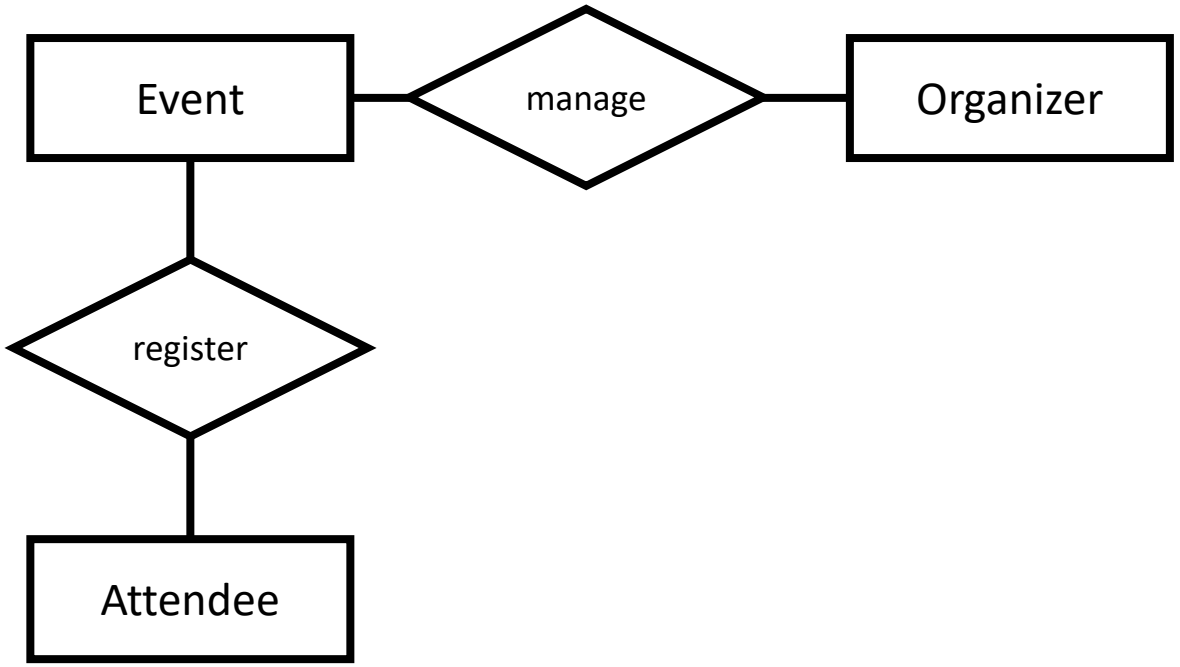
Conceptual modeling

An event website : business rules

- Attendees **participate to** events **managed by** organizers

An event website: conceptual model

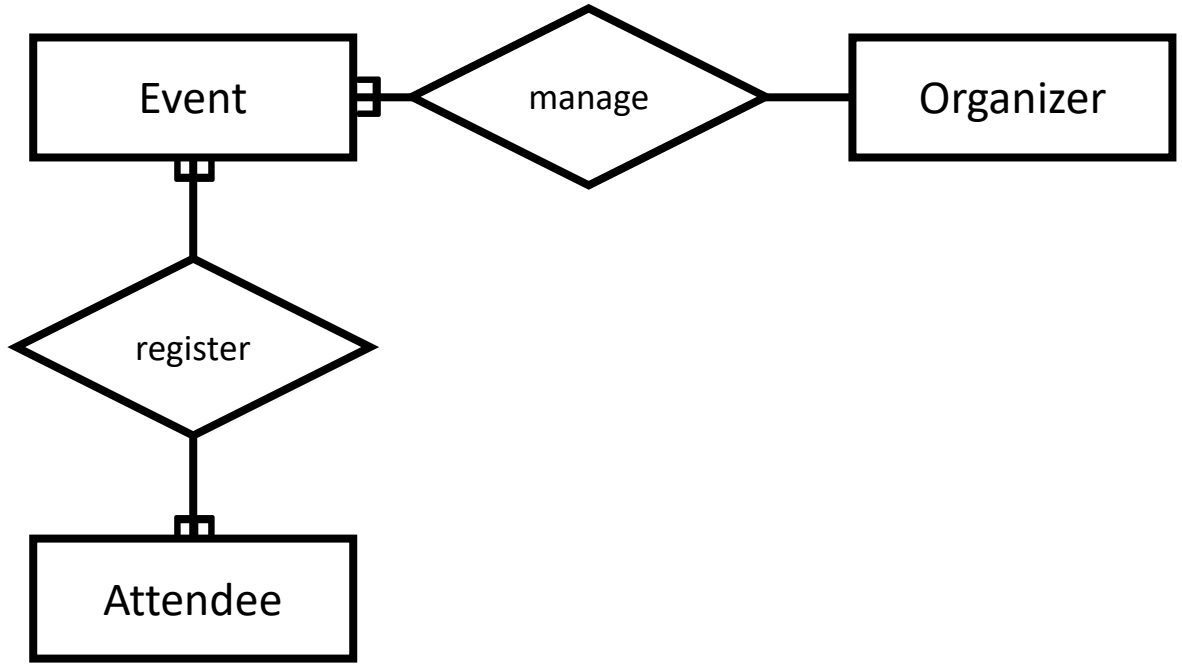
- Attendees **participate to** events **managed by** organizers



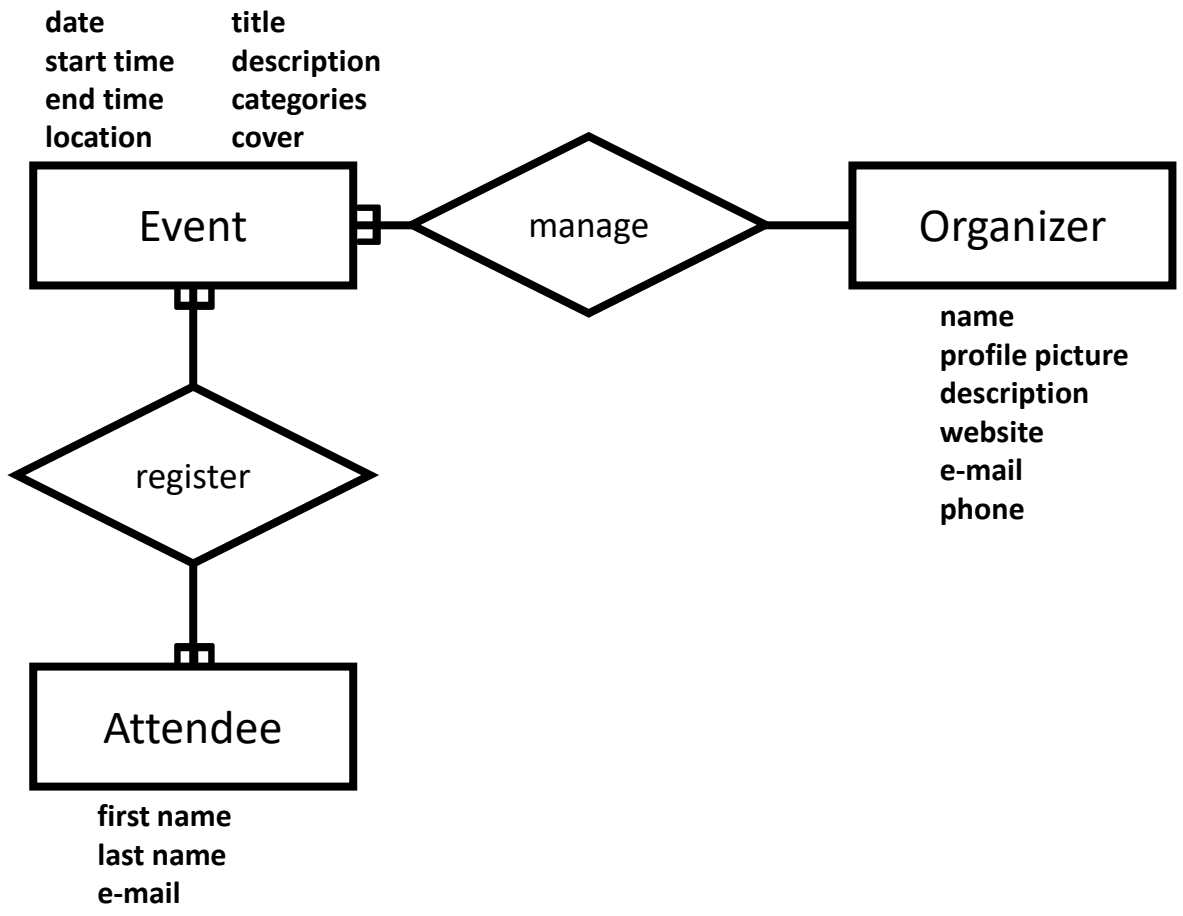
An event website: conceptual model

- One attendee can participate to many events
- One event has many attendees

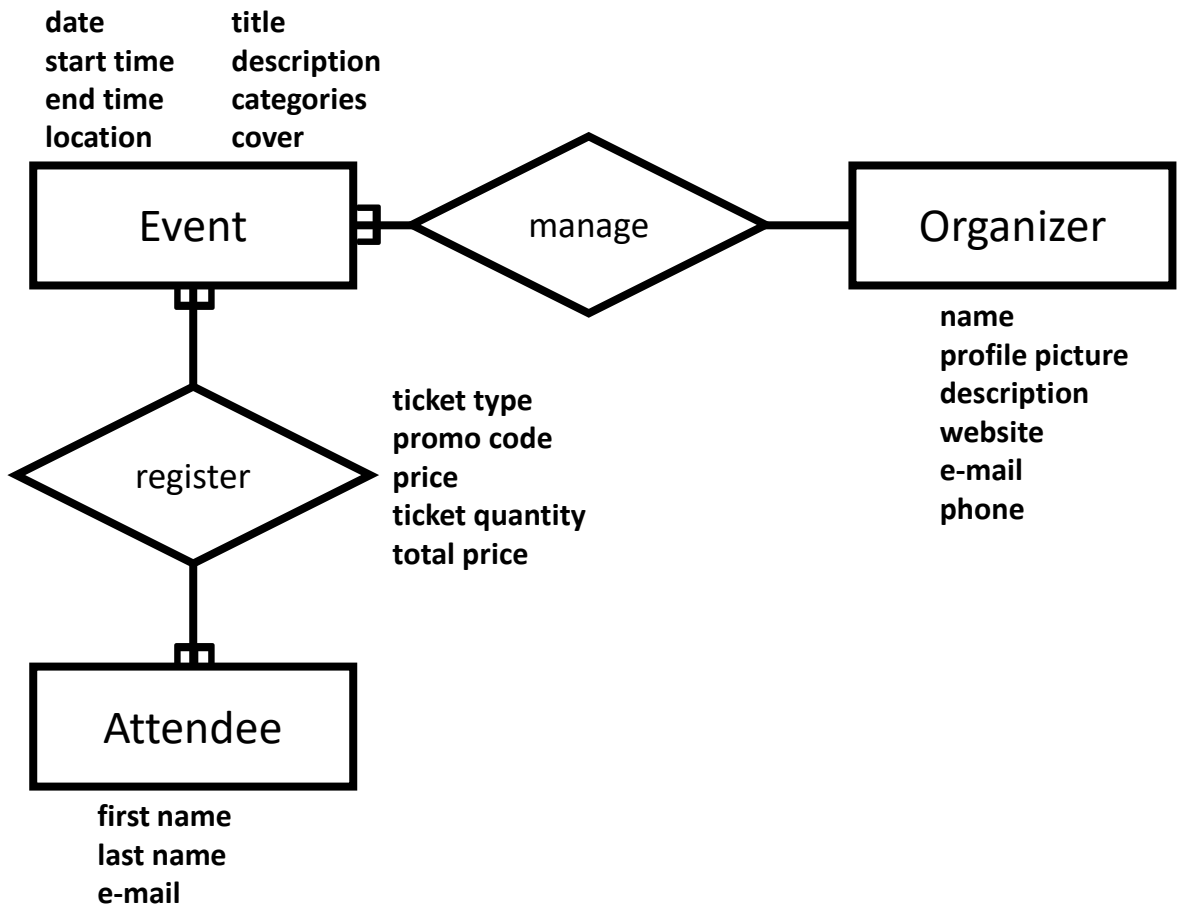
- One organizer can manage many events
- One event is managed by one organizer



Adding attributes



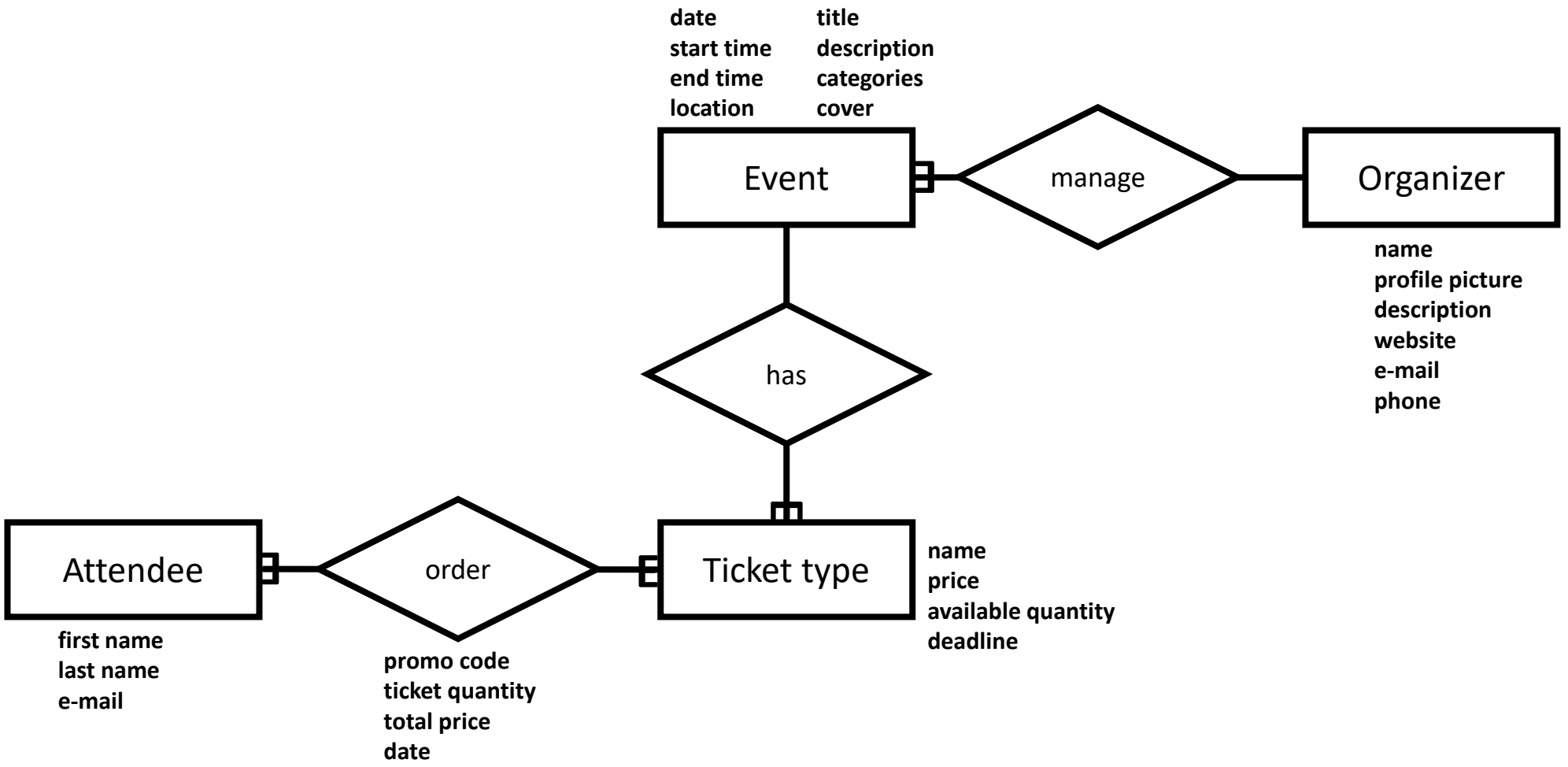
Adding attributes



An event website : business rules

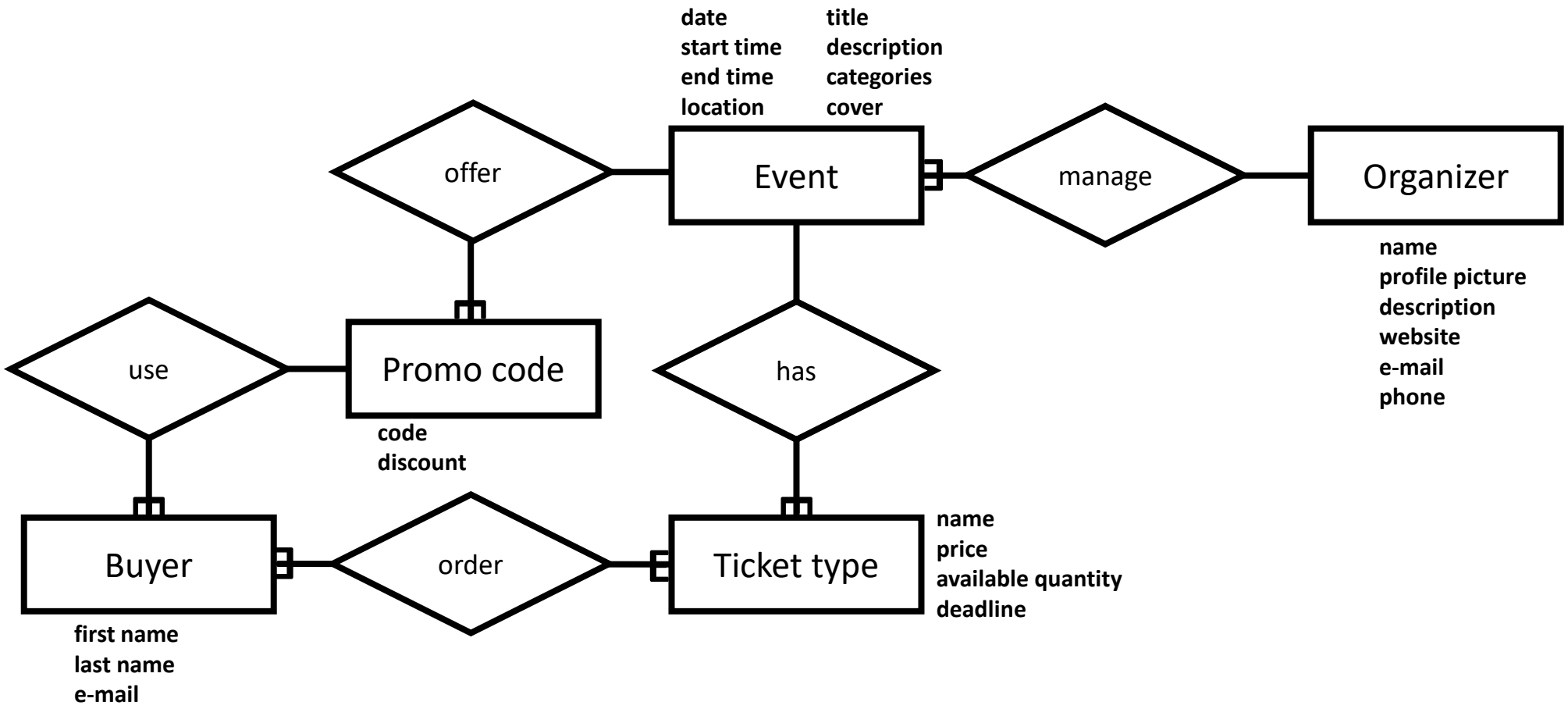
- Attendees **order** tickets **of** events **managed by** organizers

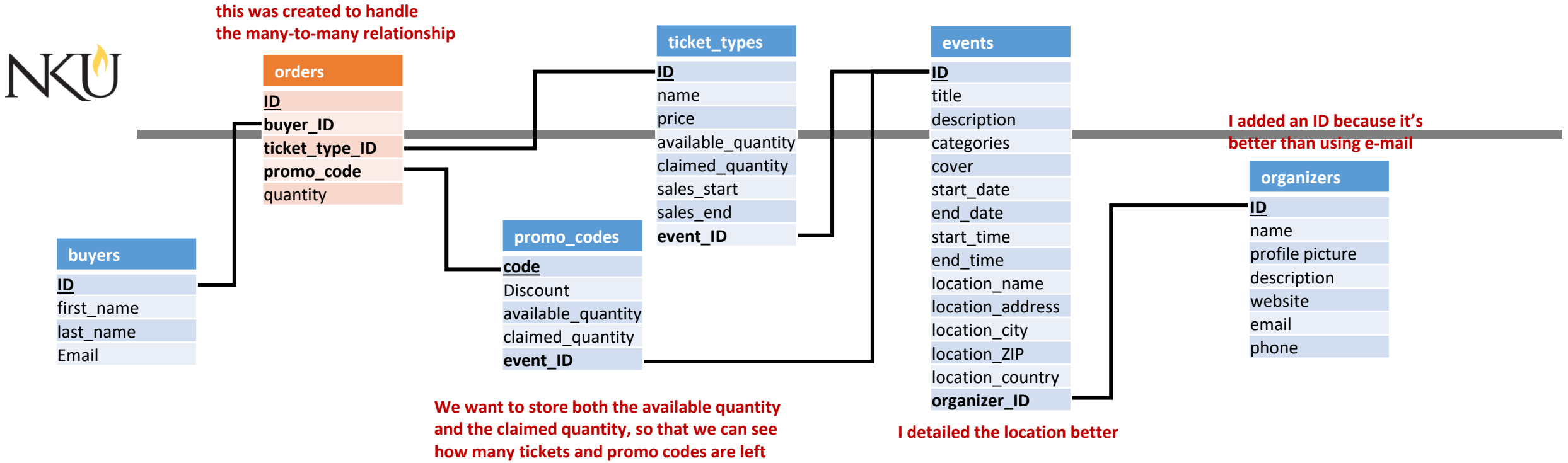
A more accurate model

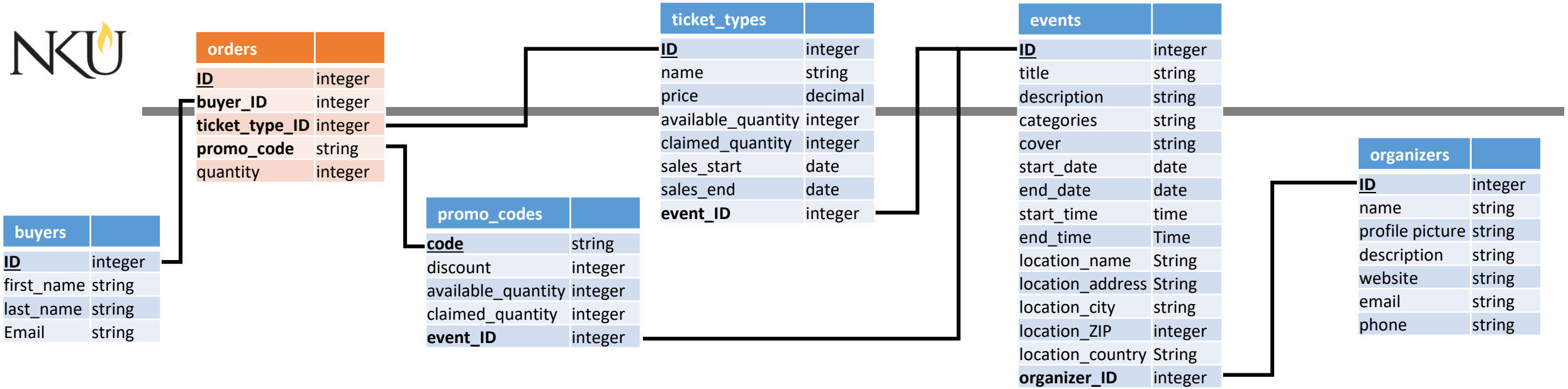


An event website: business rules

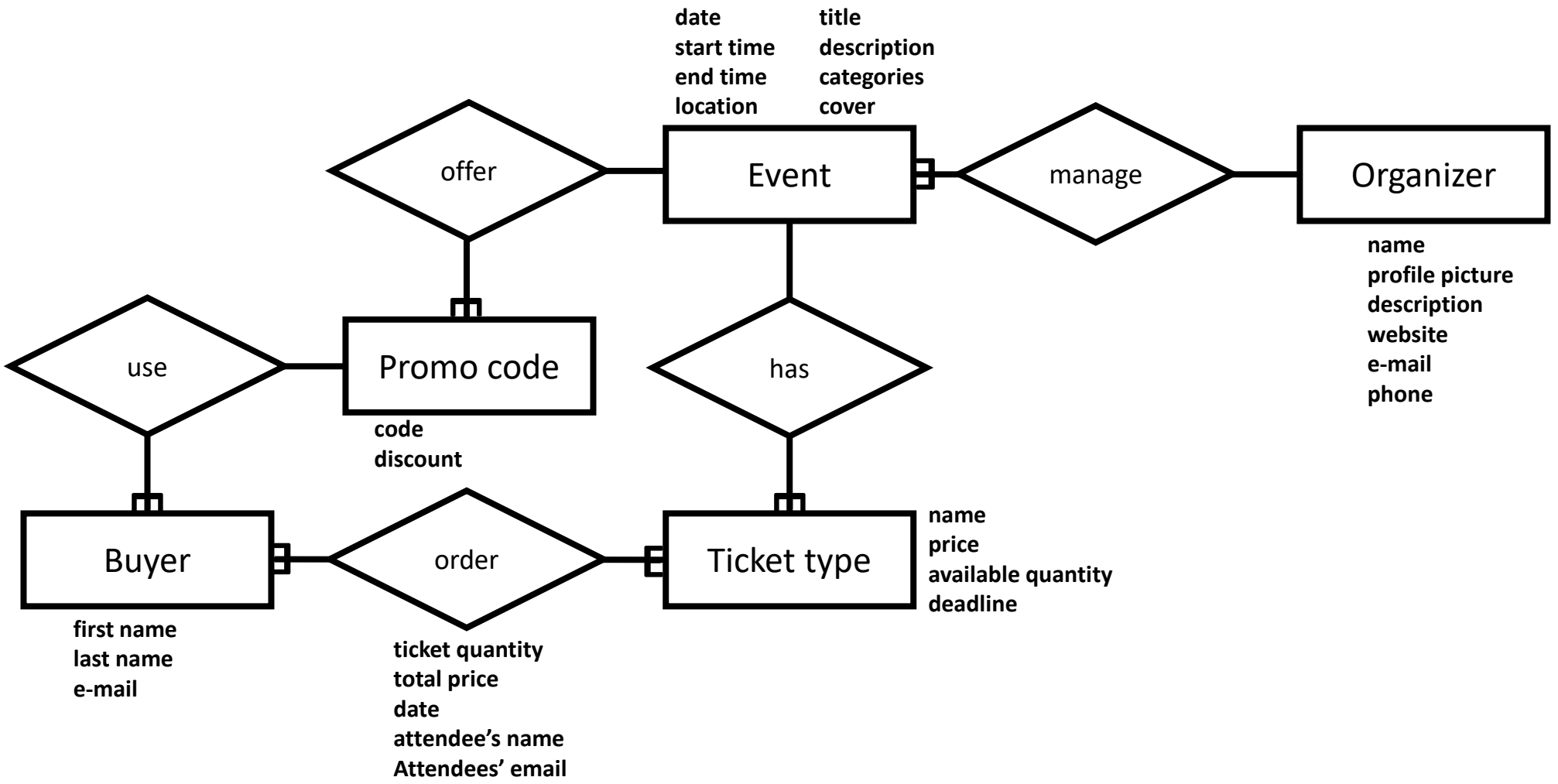
- Attendees **order** tickets **of** events **managed by** organizers
- Events **offer** promo codes **which can be used by** users


date
start time
end time
location
title
description
categories
cover
offer
Event
manage
Organizer
name
profile picture
description
website
e-mail
phone
use
Promo code
code
discount
has
Buyer
first name
last name
e-mail
order
Ticket type
name
price
available quantity
deadline


this was created to handle
the many-to-many relationship
orders
ID
buyer_ID
ticket_type_ID
promo_code
quantity
buyers
ID
first_name
last_name
Email
ticket_types
ID
name
price
available_quantity
claimed_quantity
sales_start
sales_end
event_ID
promo_codes
code
Discount
available_quantity
claimed_quantity
event_ID
We want to store both the available quantity
and the claimed quantity, so that we can see
how many tickets and promo codes are left
events
ID
title
description
categories
cover
start_date
end_date
start_time
end_time
location_name
location_address
location_city
location_ZIP
location_country
organizer_ID
I detailed the location better
I added an ID because it's
better than using e-mail
organizers
ID
name
profile picture
description
website
email
phone


orders
ID integer
buyer_ID integer
ticket_type_ID integer
promo_code string
quantity integer
ticket_types
ID integer
name string
price decimal
available_quantity integer
claimed_quantity integer
sales_start date
sales_end date
event_ID integer
events
ID integer
title string
description string
categories string
cover string
start_date date
end_date date
start_time time
end_time Time
location_name String
location_address String
location_city string
location_ZIP integer
location_country String
organizer_ID integer
organizers
ID integer
name string
profile picture string
description string
website string
email string
phone string
buyers
ID integer
first_name string
last_name string
Email string
promo_codes
code string
discount integer
available_quantity integer
claimed_quantity integer
event_ID integer

The definitive model



Internal modeling

Relational databases: terminology (1/2)

- Table: a named set of data that is organized into data types (or columns) and data values (or rows). A table has a specific number of columns, but can have any number of rows.

- Column: a named collection of data elements of the same data type. A database column is analogous to an array variable in a programming language.

- Row: a set of related data fields that share the same data type structure. A row consists of a field that corresponds to each column in a table. Rows are also known as database records. A database row can be thought of as a multi-dimensional array of values.

Relational databases: terminology (2/2)

- Database: A database is a named collection of tables that have a similar purpose.
- Field: A single database element that contains a data value. It is the intersection of a column and a row.
- Schema: Refers to the database structure of how data is organized.

How to design the internal model

- Convert names into tables (using plurals and removing spaces)
  - e.g., organizer becomes “organizers”, ticket types becomes “ticket_types”
- Convert attributes into columns (removing spaces)
  - name, profile picture, description, website, e-mail, phone
- Define the data type for each column
  - e.g., price: decimal number, name: string

An example

organizers
name
profile_picture
description
Website
email
phone

Define a primary key

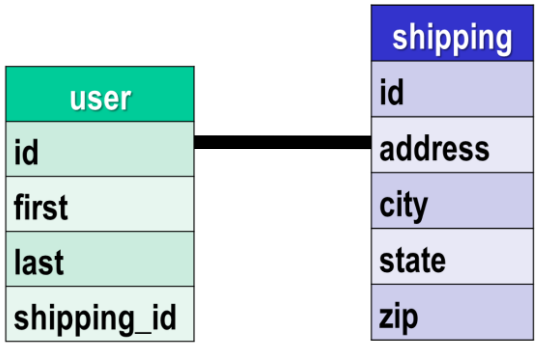
- A primary key is used as unique identifier to a database row (record)

- Characteristics:
  - Unique values (past, present, future?)
  - Value does not change
  - Only one primary key per table

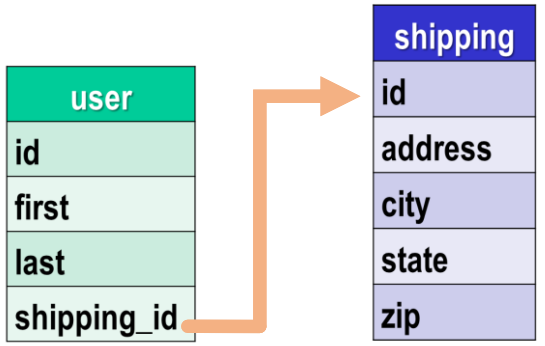
organizers
name
profile_picture
description
Website
email
phone

One to one relations

- One table is related to many (one or more) other tables
  - One to one relationships are rare
  - If there are no motivations for keeping it (e.g., performance), consider eliminating one to one relationships by combining the columns into the same table



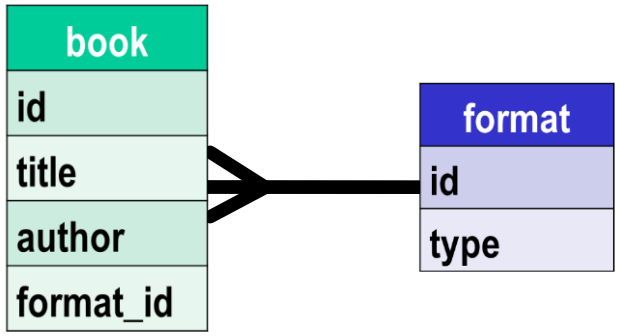
Entity Relationship Diagram



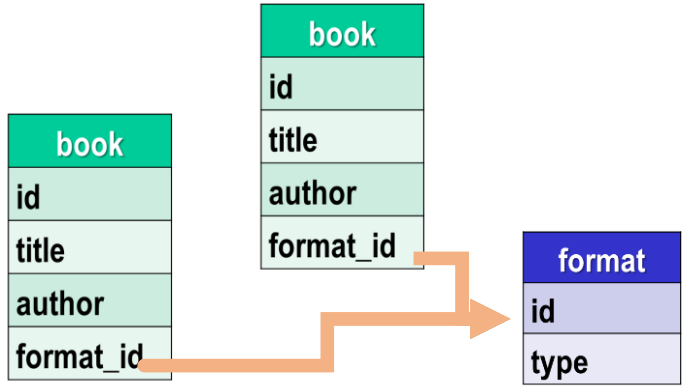
Database structure

One to many relations

- One table is related to many (one or more) other tables



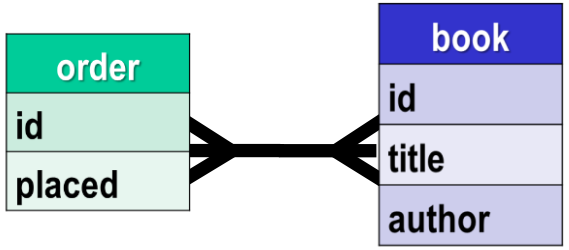
Entity Relationship Diagram



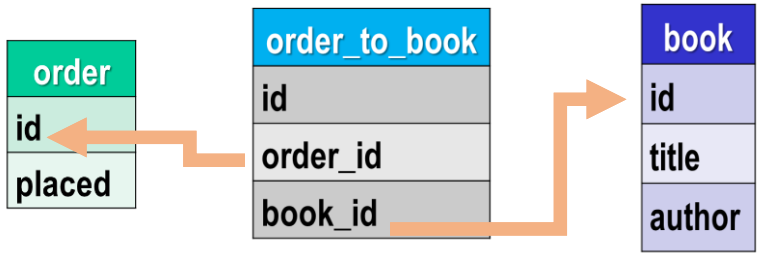
Database structure

Many to many relations

- Two (or more) tables are related to two (or more) other tables



Entity Relationship Diagram



Database structure

Foreign key

- Establishes a relationship to a key in a different table

organizers
name
profile_picture
description
Website
email
phone

events
title
date
description
start_time
end_time
organizer_name
location
…

Normalization

- Defined: The breaking apart of data into logical relationships to reduce duplication of data.

- Why Normalize?
  - Reduce duplication
  - Conserve storage space
  - Reduce maintenance

1st normal form

- Purpose: Reducing redundant data across a horizontal row

- Rules:
  - No multiple columns containing the same data
  - Column can contain only one value
  - The primary key must uniquely define the row

1st normal form: example

Before

Employee	Age	Department
Melvin	32	Marketing, Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

After

Employee	Age	Department
Melvin	32	Marketing
Melvin	32	Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

1st normal form: example

Before

Employee	Age	Department
Melvin	32	Marketing, Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

After

Employee	Age	Department
Melvin	32	Marketing
Melvin	32	Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

2nd normal form

- Purpose: Reducing redundant data in vertical columns

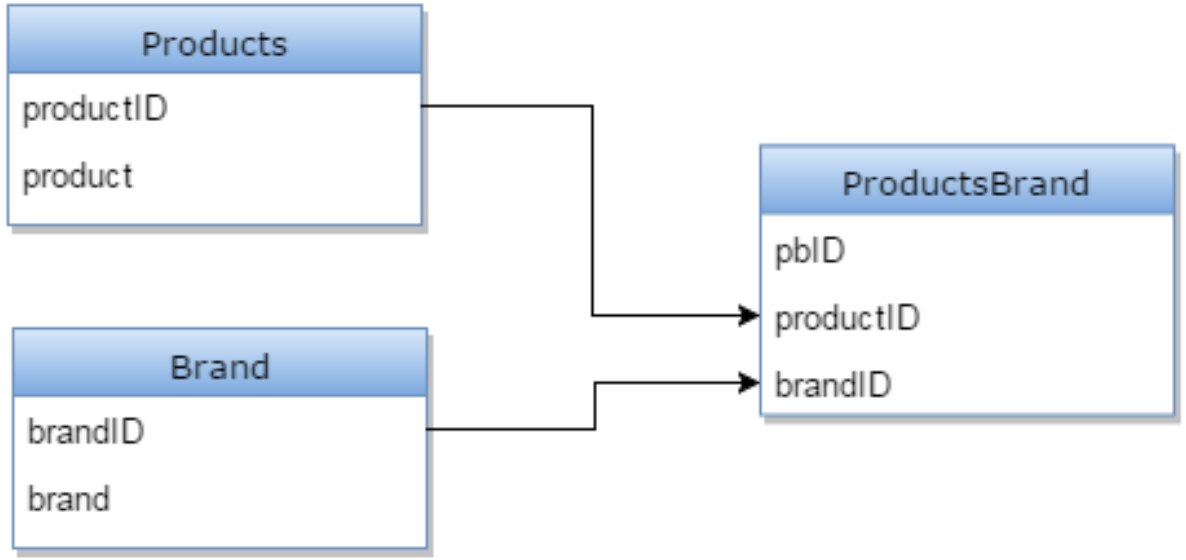
- Rules:
  - Tables must be in First Normal Form
  - Place columns that repeat values across multiple rows into a separate table
  - Place columns that aren't dependent on the primary key into a separate table
  - all attributes within the entity should depend solely on the unique identifier of the entity

2nd normal form: an example

Before

productID	product	Brand
1	Monitor	Apple
2	Monitor	Samsung
3	Scanner	HP
4	Head phone	JBL

After



2nd normal form: an example

Before

productID	product	Brand
1	Monitor	Apple
2	Monitor	Samsung
3	Scanner	HP
4	Head phone	JBL

After

Products Category table:

productID	product
1	Monitor
2	Scanner
3	Head phone

Products Brand table:

pbID	productID	brandID
1	1	1
2	1	2
3	2	3
4	3	4

Brand table:

brandID	brand
1	Apple
2	Samsung
3	HP
4	JBL

3rd normal form

- Purpose: Reduce data that is not dependant on the primary key, but is dependant on other data in the table

- Note: The Third Normal Form process may not be necessary if First and Second Normal Form process has been performed. (refinement of 1NF and 2NF)

3rd normal form: an example

Before

student_ID	student_name	city	zip

After

student_ID	student_name	zip

zip	city

More normalization forms

- Third Normal Form (3NF)
  - The entity should be considered already in 2NF and no column entry should be dependent on any other entry (value) other than the key for the table.
  - If such an entity exists, move it outside into a new table.
  - 3NF is achieved are considered as the database is normalized.
- Boyce-Codd Normal Form (BCNF)
  - 3NF and all tables in the database should be only one primary key.
- Fourth Normal Form (4NF)
  - Tables cannot have multi-valued dependencies on a Primary Key.
- Fifth Normal Form (5NF)
  - Composite key shouldn’t have any cyclic dependencies.

More normalized tables: an example

Before

student_ID	student_name	zip

zip	city

After

student_ID	student_name	zip

zip	city_ID

city_ID	name

Normalization extremes

- Avoid Extremes
  - Each data column in a separate table
    - Queries would be very large
    - Additional processing time required
  - All columns in one table
    - Data is repeated between rows
    - Requires additional storage
    - Updates on multiple rows required