CS 500: Fundamentals of Databases

Introduction and Overview
Entity Relationship (ER) Modeling

supplementary material:
“Database Management Systems” Ch. 1, 2.1-2.5
class notes

Julia Stoyanovich (stoyanovich@drexel.edu)
Instructor: Julia Stoyanovich

- Assistant Professor of Computer Science @ Drexel
- 2009-2012: Postdoc, Visiting Scholar @ UPenn
- 2009: PhD in CS @ Columbia University
- 1998-2003: engineer at start-ups in New York City
- 1998: BS in CS & Math @ UMass Amherst

Research: data and knowledge management

- **Data, responsibly**: Fairness, accountability and transparency in data science
- **Portal**: Querying and analysis of evolving graphs: the Web, social networks
- **DB4Pref**: Managing preference data: political polls, movie rating etc
Logistics

• Website: https://www.cs.drexel.edu/~julia/cs500/

• BBLearn: discussion board, lecture videos, grade center

• This is a fast-paced course, I cannot accept late assignments

• My office hours: M 3-4pm, W noon-1pm (UC 100E)

• Accounts: tux, PostgreSQL, gitlab

• You must have at least a basic familiarity with Linux, see https://www.cs.drexel.edu/~julia/cs500/resources.html for tutorials, contact the TA with any questions
TA: Amir Pouya Aghasadeghi

- Office Hours: Wednesdays 4-6pm in CLC (UC 152) or by appointment
- Email: aa3657@drexel.edu
- Skype: amirpouya.a
- Schedule a meeting to discuss: ssh, git, PostgreSQL, Linux, project set-up
Textbook

3rd Edition (earlier editions will not do)

ISBN-10: 0072465638

Questions / Feedback

If something is bothering you - let me know early!

• Ways to get in touch with me (phone is not great)
  • Ask questions in class
  • Come to office hours
  • Send me an email
  • Post to BBLearn, for the benefit of other students
Course outline

• Modeling and design: ER and relational models, normalization

• Querying: relational algebra, SQL, Datalog

• Query execution: indexing, query optimization

• Responsible data mining
What is a DBMS?

- A **database** is a large integrated collection of data
  - Models a real-world enterprise
  - Augments raw data with metadata, to give meaning to the data
- A **Database Management System (DBMS)** is a software package designed to store and manage databases
- Our focus: **Relational Database Management Systems (RDBMS)**, centered around the **relational model**
  - Entities (e.g., students and courses)
  - Relationships (e.g., students taking courses)
The role of a DBMS

• Serves as an intermediary between the user and the database
• Enables the sharing of data
• Supports multiple alternative views of the data
A traditional database application

• Build a system to store and access information about
  • students
  • courses
  • professors
  • who takes what, who teaches what

• Functionality
  • record enrollment information
  • compute GPA for each student after each term
  • analyze student enrollment and performance in different courses, majors, departments etc
Can we do this without a DBMS?

Sure we can! We can store data in files:

```
students.txt  courses.txt  professors.txt
```

... then write a C or Java program to implement specific tasks.

Different users can execute these tasks (run the programs)
Without a DBMS

Task: Enroll “Mary” in “CS500”

write a Java program that does the following:

read students.txt
read courses.txt
find & update record “Mary”
find & update record “CS500”
write students.txt
write courses.txt
Without a DBMS

- System crashes
- Large datasets (say, 500GB)
- Simultaneous access by many users

```
read students.txt
read courses.txt
find & update record “Mary”
find & update record “CS500”
write students.txt
write courses.txt
```
Without a DBMS

- Format of `students.txt` changes

  - read `students.txt`
  - read `courses.txt`
  - find & update record “Mary”
  - find & update record “CS500”
  - write `students.txt`
  - write `courses.txt`

- Certain users should only be allowed to see parts of the file `courses.txt`
Enter a DBMS

- When in doubt - introduce a level (or levels) of abstraction!
- Many views, one conceptual (logical) schema, one physical schema
  - Views describe how users see the data.
  - A logical schema defines the logical structure of the data.
  - A physical schema describes the files and indexes used.

![Diagram: Conceptual Schema (root) with branches to View 1, View 2, View 3, descending to Physical Schema.]
Example

- View 1: Course_Info(cid: string, enrollment: int)
- View 2: Student_Info (sid:string, name: string, gpa: real)

- Conceptual schema
  - Students (sid:string, name: string, gpa: real)
  - Courses (cid: string, name: string, credits: int)
  - Enrolled (sid: string, cid: string, grade: real)

- Physical schema
  - Courses, Enrolled stored as unordered files
  - Students stored sorted by sid
A key concept: data independence

- Applications (and users) are insulated from how data is structured and stored
- **Logical data independence**: protection from changes in logical structure of the data
- **Physical data independence**: protection from changes in physical structure of the data
So, why use a DBMS?

• Technical reasons
  • Data independence and efficient access
  • Reduced application development time
  • Data integrity and security
  • Uniform data administration
  • Concurrent access, recovery from crashes

• Business reasons
  • Reusing existing approaches makes data management more cost-effective!
A word of caution

• DBMS give us the tools to make data management more convenient, efficient, cost-effective

• But, like any technology, databases will only make our lives easier when used appropriately!
Why I love databases

- Beautiful
- Practical
- Never boring
Sets

A good overview at

http://en.wikipedia.org/wiki/
Set_theory#Basic_concepts_and_notation

• A set is an *unordered collection* of objects

• An object belonging to a set is an *element* (or a *member*) of that set, written $a \in A$

• We denote sets with capital letters, elements with lowercase letters

If you are rusty on these concepts, it is your responsibility to get up to speed!
Sets

• An empty set is a set that contains no elements. \( A = \emptyset \)

• We say that \( A \) is a subset of \( B \) if all elements of \( A \) are also members of \( B \). Then \( B \) is a superset of \( A \). \( A \subseteq B \quad B \supseteq A \)

• \( A \) is a proper subset of \( B \) if \( A \) is a subset of \( B \), and there exists at least one element \( b \in B \) such that \( b \notin A \) \( A \subset B \quad B \supset A \)

• Any set \( A \) a subset of itself. A proper subset?

• \( A = \emptyset \) is a subset of any set. A proper subset?
Sets

- The *size* (aka *cardinality*) of a set is the number of elements in the set $|A|$
- The size of an empty set is 0
- Some sets are of infinite size, e.g., the set of all integers, all prime numbers, etc
- In databases we usually work with large finite sets
Set operations

Intersection

\[ A \cap B \]

Union

\[ A \cup B \]

Difference

\[ B \setminus A \]

Venn diagrams from http://en.wikipedia.org/wiki/Venn_diagram
Set operations

How can you express *symmetric difference* using set operations on the previous slide?
Cartesian product

Cartesian product of \( A \) and \( B \), denoted \( A \times B \), is a set of ordered pairs \( (a, b) \), where \( a \in A, b \in B \).

Example: \( A = \{1, 2, 3\} \quad \text{and} \quad B = \{3, 4\} \)

\[
A \times B = \{(1,3), (1,4), (2,3), (2,4), (3,3), (3,4)\}
\]

Cross-product is another name for this operations
ER Modeling

- The Entity Relationship (ER) model is used for logical design of a database
- Introduced by Chen in 1976
- What are the kinds of entities and relationships in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the business rules that hold?
- What operations are required / frequent?

Key insight: some designs are clearly wrong, but there are often multiple correct ones!
ER model basics

- **Entity** - a real-world object distinguishable from other objects, e.g., *Mary Jones, CS500*

- **Attribute** - a characteristic of an entity, e.g., name, age, social security number (ssn)

- **Entity set** - a collection of similar entities, where similar means (for now) that they share the same attributes
Keys-keys-keys

Each entity set must have a key attribute (a.k.a. an identifier)

- a key uniquely identifies an entity within the entity set
- a key is either simple (1 attribute) or composite (multiple attributes)

Keys are important! They allow us to:

- unambiguously refer to an entity within a set
- link together entities across entity sets
Attributes

- Each attribute has a domain - a set of possible values (e.g., GPA, gender, name)
- Required vs. optional (e.g., last name vs. middle name)
- Simple vs. composite (e.g., age vs. address)
- Single-valued vs. multi-valued (e.g., car make vs. color)
-Stored vs. derived (e.g., dob vs. age) - advantages / disadvantages?
Relationships and relationship sets

- A relationship is an association between two or more entities, e.g., Jane works in IT.

- A relationship set is a collection of similar relationships, e.g., Employees work in Departments.
Descriptive attributes

A relationship set can have descriptive attributes
N-ary relationship sets

An \textit{n-ary relationship set} $R$ associates $n$ entity sets $E_1, E_2, \ldots, E_n$. Each relationship $r$ relates entities $e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n$. This is a ternary relationship set.
Recursive relationship sets

The same entity set can take on different roles in the same relationship set. This is called a recursive relationship set.
An *instance* of a relationship set

A relationship belongs to (is an element of) a relationship set, and so must be uniquely identifiable by participating entities.

Meaning: each (employee, department) pair appears in the relationship set only once. Because it’s a set!
Examples

Draw ER diagrams representing entity sets and relationship sets described below.

*Men*(ssn, name, dob)  *Women*(ssn, name, dob)

*Currently_Married* (marriage_date)

*Cars*(vin, plate, state)  *Parking*(lot, address)

*Assigned_To()*
Examples (solution)

Draw ER diagrams representing entity sets and relationship sets described below.

*Men*(ssn, name, dob)  *Women*(ssn, name, dob)

*Currently_Married* (marriage_date)
Examples

Draw ER diagrams representing entity sets and relationship sets described below.

\textit{Cars}(\textit{vin, plate, state}) \quad \textit{Parking}(\textit{lot, address})

\texttt{Assigned\_To()}
Draw ER diagrams representing entity sets and relationship sets described below.

*Cars*(vin, plate, state)  *Parking*(lot, address)

Assigned_To()
Entity vs. Attribute

- It is not always easy to tell whether an attribute warrants creating an entity set of its own

- Difficult cases
  - multi-valued attributes, e.g., colors of a car (exterior / interior)
  - composite attributes, e.g., address

- Rule of thumb: composite attributes should be modeled as entity sets if we care about their structure

- Rule of thumb: multi-valued attributes should be modeled as entity sets
Another tricky case: descriptive attributes of relationship sets

Business rule: An employee many manage the same department during different periods of time
What is the difference?
Examples

Draw ER diagrams representing entity sets and relationship sets described below.

Courses are offered by departments. A course may be offered in multiple terms, and each offering must be recorded.

Courses are offered by departments. A course may be offered in multiple terms, and only the most recent offering must be recorded.

Courses are offered by departments. A course may be offered in multiple terms, and only the first offering must be recorded.
Examples (solution)

Draw ER diagrams representing entity sets and relationship sets described below.

Courses are offered by departments. A course may be offered in multiple terms, and each offering must be recorded.
Examples (solution)

Draw ER diagrams representing entity sets and relationship sets described below.

Courses are offered by departments. A course may be offered in multiple terms, and only the most recent offering must be recorded.

Courses are offered by departments. A course may be offered in multiple terms, and only the first offering must be recorded.

same answer for both!
ER model: beyond the basics

- So far we saw the basic ER model
- From the next slide on we will look at additional features: annotating / highlighting edges to express more detailed business rules
- Our goal is to express the following concepts for relationships:
  1. key constraints
  2. participation constraints
Multiplicity (aka connectivity)

- many to 1
- 1 to 1
- many to many
Key constraints

- Business rule: A professor may teach 0, 1 or multiple classes. Each class is taught by at most one professor.

- The restriction that each class is taught by at most one professor is an example of a key constraint.

- Intuitively: given a Classes entity, we can uniquely determine which Professors entity it maps to.

- This gives a way to interpret the one side of one-to-one and one-to-many relationship sets.
Key constraints

Business rule: A professor may teach 0, 1 or multiple classes. Each class is taught by *at most one* professor.
Key constraints

Business rule: A professor teaches *at most one* class. Each class is taught by *at most one* professor.
Participation constraints

• Business rule: A professor may teach 0, 1 or multiple classes. Each class is taught by some professor.

• The restriction that each class is taught by some (read “at least one”) professor is an example of a participation constraint.

• Intuitively: each Classes entity participates in the relationship set teach

• Participation and key constraints are orthogonal: you can have one without the other
Participation constraints

Business rule: A professor may teach 0, 1 or multiple classes. Each class is taught by *some* professor.
Participation constraints

Business rule: Each professor teaches *some* class. Each class is taught by *some* professor.
Participation constraints

Business rule: Each professor teaches some class. Each class is taught by exactly one professor.
Explain the business rules
Examples

Draw ER diagrams representing entity sets and relationship sets described below. Clearly mark all key and participation constraints.

Employees have managers. An employee has at most one manager. A manager may manage any number of employees.

Professors advise students. Each professor advises at most one student. Each student has exactly one advisor.
Examples (solution)

Draw ER diagrams representing entity sets and relationship sets described below. Clearly mark all key and participation constraints.

Employees have managers. An employee has at most one manager. A manager may manage any number of employees.
Examples (solution)

Draw ER diagrams representing entity sets and relationship sets described below. Clearly mark all key and participation constraints.

Professors advise students. Each professor advises at most one student. Each student has exactly one advisor.
Example

• An author is described by a name and a date of birth (dob). No two authors have the same combination of name and dob.
• A book is described by an ISBN, a title and a year when it was written (year). No two books have the same ISBN number.
• A book is written by exactly one author.
• An author is only included in the database if she has authored at least one book.
Example (solution)

- An author is described by a name and a date of birth (dob). No two authors have the same combination of name and dob.
- A book is described by an ISBN, a title and a year when it was written (year). No two books have the same ISBN number.
- A book is written by exactly one author.
- An author is only included in the database if she has authored at least one book.
Example

• A scientist is described by a name, a field of study, and a date of birth (dob). No two scientists have the same combination of name and dob.
• A discovery has a name. No two discoveries have the same name.
• An award has a name, which uniquely identifies it.
• Scientists make discoveries. A discovery is made by one or several scientists. A scientist who made no discoveries is not tracked in our database.
• Scientists receive awards. A scientist may receive the same award more than once in different years. All years in which a scientist received a particular award must be recorded.
Example (solution)

- A scientist is described by a name, a field of study, and a date of birth (dob). No two scientists have the same combination of name and dob.
- A discovery has a name. No two discoveries have the same name.
- An award has a name, which uniquely identifies it.
- Scientists make discoveries. A discovery is made by one or several scientists. A scientist who made no discoveries is not tracked in our database.
- Scientists receive awards. A scientist may receive the same award more than once in different years. All years in which a scientist received a particular award must be recorded.
Draw an ER diagram that encodes the following business rules. Clearly mark all key and participation constraints.

Chefs work at restaurants. A chef is uniquely identified by an SSN, and is also described by a name and a cuisine in which she specialized. A restaurant is uniquely identified by a combination of name and city. Each chef works in at least one restaurant, and each restaurant must have at least one chef working at it. Some chefs own restaurants, and if a chef owns a restaurant - she is its sole owner.
ER modeling

Draw an ER diagram that encodes the following business rules. Clearly mark all key and participation constraints.

Chefs work at restaurants. A chef is uniquely identified by an SSN, and is also described by a name and a cuisine in which she specialized. A restaurant is uniquely identified by a combination of name and city. Each chef works in at least one restaurant, and each restaurant must have at least one chef working at it. Some chefs own restaurants, and if a chef owns a restaurant - she is its sole owner.
Key and participation constraints: recap

- **Key constraint**
  - denoted by a line with an arrowhead towards the relationship set
  - occurs only in 1-to-many or 1-to-1 relationship sets
  - example: a course is taught by one / at most one / exactly one professor

- **Participation constraint**
  - denoted by a bold line towards the relationship set
  - can occur on the 1 or on the many side of a relationship set
  - example: a course must be taught by a professor / a course is taught by at least one professor
Weak entities

• An entity is *weak* (or existence-dependent) if it can exist only when associated with an occurrence of the *identifying owner*

• Otherwise an entity is strong, of just “an entity”

• **Important:** weak entities cannot be identified without the identifying owner, and so they do not specify a complete key.
Weak entities

- The identifying owner entity and the weak entity must participate in a one-to-many relationship set. This relationship set is called the identifying relationship set of the weak entity.

- The weak entity must have total participation in the identifying relationship set.
Summary of conceptual design

- Entity sets / entities / attributes
- Relationship sets / relationships
- Multiplicity: 1-to-1, 1-to-many, many-to-many
- Key constraints: given an entity, can we identify other entities participating in the relationship
- Participation: partial (not all entities in the set participate) vs. total (all do)
- Weak entities: cannot be identified without an identifying owner, participation must be total
The Extended ER model: Entity Clustering

• ER diagrams can get large (tens or hundreds of entity sets and relationship sets)

• For readability we sometimes group together related entity sets and relationship sets into a single entity cluster

• We will see an example next
A night at the opera

“I love the opera. Let me develop a useful application for this domain.”

What are the entities?
What are the relationships?
What are the business rules?
Opera: entity sets

- **Opera** *(title, year)* - may have more than one opera by the same title, but not in the same year (e.g., Otello - Verdi 1887 / Rossini 1816)

- **Composer** *(name, birth, death)* - may have more than one composer with the same name, but not also with the same birth and death years

- **Librettist** *(name, birth, death)*

- **Company** *(name)*

- **Venue** *(name, city)*

- **Production** - an entity set or a relationship set, linking Opera and Company?

- **Performance** - an entity set or a relationship set, linking Production and Venue
Opera: relationship sets

- Opera is written by Composer / Librettist
- each Composer composed at least one Opera
- each Librettist wrote the libretto for at least one Opera
- an Opera by the same name may be written by a different (Composer, Librettist) pair
- A piece is produced by a Company
- A production is performed at a Venue
Opera: the model

- **COMPOSER**
  - name
  - birth
  - death

- **LIBRETTIST**
  - name
  - birth
  - death

- **OPERA**
  - name
  - year

- **COMPANY**
  - name
  - season

- **VENUE**
  - name
  - city

- **PRODUCTION**
  - performed_at
  - date

- **PIECE**
  - written_by
  - produced_by