

Relational Database Design
Part I

CompSci 316
Introduction to Database Systems

2

Announcements (Tue. Sep. 4)

- ❖ Homework #1 is out
 - Due in two weeks, but start early—as soon as any portion has been covered
- ❖ Sign up for Gradiance
- ❖ Try our VM now
 - We can't help you on the due date if you run into last-minute installation problems
- ❖ Readings: see Tentative Syllabus on course website
- ❖ My office hour today is canceled

3

Relational model: review

- ❖ A database is a collection of relations (or tables)
- ❖ Each relation has a list of attributes (or columns)
- ❖ Each attribute has a domain (or type)
- ❖ Each relation contains a set of tuples (or rows)

Keys

4

- ❖ A set of attributes K is a key for a relation R if
 - In no instance of R will two different tuples agree on all attributes of K
 - That is, K is a “tuple identifier”
 - No proper subset of K satisfies the above condition
 - That is, K is minimal
- ❖ Example: *Student* (SID , $name$, age , GPA)
 - SID is a key of *Student*
 - age is not a key (not an identifier)
 - $\{SID, name\}$ is not a key (not minimal)

Schema vs. instance

5

Student

<u>SID</u>	<u>$name$</u>	<u>age</u>	<u>GPA</u>
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3

- ❖ Is $name$ a key of *Student*?
- ❖ Key declarations are part of the schema

More examples of keys

6

- ❖ *Enroll* (SID , CID)
 - ☞ A key can contain multiple attributes!
- ❖ *Address* ($street_address$, $city$, $state$, zip)
 - ☞ A relation can have multiple keys!
 - We typically pick one as the “primary” key, and underline all its attributes, e.g., *Address* ($street_address$, $city$, $state$, zip)

Usage of keys 7

- ❖ More constraints on data, fewer mistakes
- ❖ Look up a row by its key value
 - Many selection conditions are “key = value”
- ❖ “Pointers”
 - Example: *Enroll (SID, CID)*
 - *SID* is a key of *Student*
 - *CID* is a key of *Course*
 - An *Enroll* tuple “links” a *Student* tuple with a *Course* tuple
 - Many join conditions are “key = key value stored in another table”

Database design 8

- ❖ Understand the real-world domain being modeled
- ❖ Specify it using a database design model
 - More intuitive and convenient for schema design
 - But not necessarily implemented by DBMS
 - A few popular ones:
 - Entity/Relationship (E/R) model
 - Object Definition Language (ODL)
 - UML (Unified Modeling Language)
- ❖ Translate specification to the data model of DBMS
 - Relational, XML, object-oriented, etc.
- ❖ Create DBMS schema

But what about ORM? 9

- ❖ Automatic object-relational mappers are made popular by recent prevalence of rapid Web development frameworks
 - For example, in Django:
 - You declare Python classes and their relationships
 - Django automatically converts them into database tables
 - If you want, you can just work with Python objects, and never need to be aware of the database schema or write SQL
 - The design model sometimes isn't powerful enough
 - Designer discretion is still required in all but simple cases
 - Querying/modifying data in a general-purpose programming language isn't necessarily easier than SQL

Entity-relationship (E/R) model

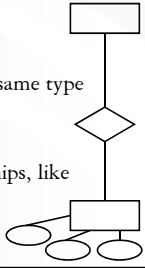
10

- ❖ Historically and still very popular
- ❖ Can think of as a “watered-down” object-oriented design model
- ❖ Primarily a design model—not directly implemented by DBMS
- ❖ Designs represented by E/R diagrams
 - We use the style of E/R diagram covered by GMUW; there are other styles/extensions
 - Very similar to UML diagrams

E/R basics

11

- ❖ Entity: a “thing,” like an object
- ❖ Entity set: a collection of things of the same type, like a relation of tuples or a class of objects
 - Represented as a rectangle
- ❖ Relationship: an association among entities
- ❖ Relationship set: a set of relationships of the same type (among same entity sets)
 - Represented as a diamond
- ❖ Attributes: properties of entities or relationships, like attributes of tuples or objects
 - Represented as ovals



An example E/R diagram

12

- ❖ Students enroll in courses

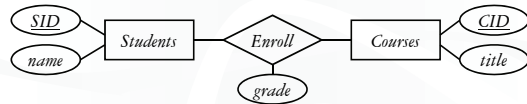


- ❖ A key of an entity set is represented by underlining all attributes in the key
 - A key is a set of attributes whose values can belong to at most one entity in an entity set—like a key of a relation

Attributes of relationships

13

- ❖ Example: students take courses and receive grades



- ❖ Where do the grades go?

- With *Students*?
- With *Courses*?
- With *Enroll*!

More on relationships

14

- ❖ There could be multiple relationship sets between the same entity sets

- Example: *Students Enroll Courses*; *Students TA Courses*

- ❖ In a relationship set, each relationship is uniquely identified by the entities it connects

- Example: Between Bart and CPS316, there can be at most one *Enroll* relationship and at most one *TA* relationship

- ☞ What if Bart took CPS316 twice and got two different grades?

Multiplicity of relationships

15

- ❖ E and F : entity sets

- ❖ Many-many: Each entity in E is related to 0 or more entities in F and vice versa

- Example:

- ❖ Many-one: Each entity in E is related to 0 or 1 entity in F , but each entity in F is related to 0 or more in E

- Example:

- ❖ One-one: Each entity in E is related to 0 or 1 entity in F and vice versa

- Example:

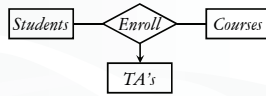
- ❖ “One” (0 or 1) is represented by an arrow \longrightarrow

- ❖ “Exactly one” is represented by a rounded arrow \longrightarrow

N-ary relationships

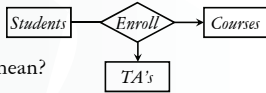
16

- ❖ Example: Each course has multiple TA's; in a course, each student is assigned to one TA



- ❖ Meaning of an arrow into E : Pick one entity from each of the other entity sets; together they must be related to either 0 or 1 entity in E

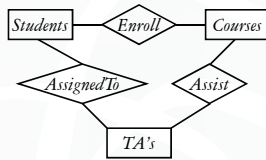
- E.g., hypothetically, what would the multiplicities on the right mean?



N-ary versus binary relationships

17

- ❖ Can we model n -ary relationships using just binary relationships?



Roles in relationships

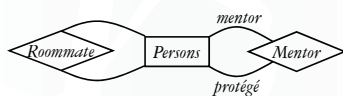
18

- ❖ An entity set may participate more than once in a relationship set

☞ May need to label edges to distinguish roles

- ❖ Examples

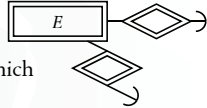
- People mentor others; label needed
- People are roommates of each other; label not needed



Weak entity sets

19

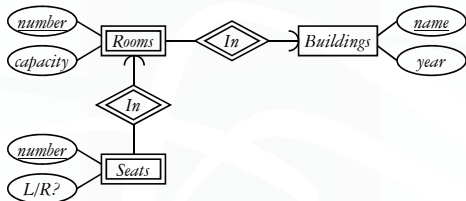
- ❖ Sometimes, the key of an entity set E comes not completely from its own attributes, but from the keys of other (one or more) entity sets
- ❖ E must link to them via many-one (or one-one) relationship sets
 - Example: *Rooms* inside *Buildings* are partly identified by *Buildings'* name
- ❖ E is called a weak entity set, drawn as a double rectangle
- ❖ The relationship sets through which E obtains its key are called supporting relationship sets, drawn as double diamonds



Weak entity set examples

20

- ❖ Seats in rooms in buildings

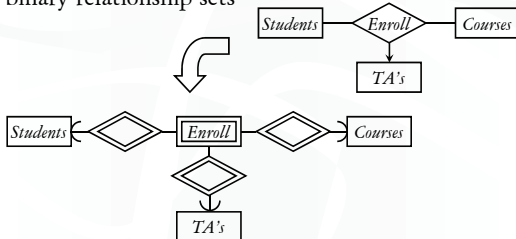


- ❖ Why must double diamonds be many-one/one-one?

Modeling n -ary relationships

21

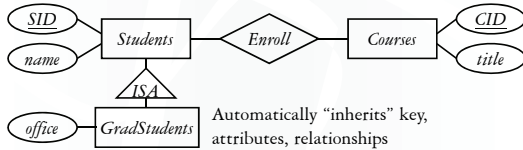
- ❖ An n -ary relationship set can be replaced by a weak entity set (called a connecting entity set) and n binary relationship sets



ISA relationships

22

- ❖ Similar to the idea of subclasses in object-oriented programming: subclass = special case, fewer entities, and possibly more properties
 - Represented as a triangle (direction is important)
- ❖ Example: Graduate students are students, but they also have offices



Summary of E/R concepts

23

- ❖ Entity sets
 - Keys
 - Weak entity sets
- ❖ Relationship sets
 - Attributes of relationships
 - Multiplicity
 - Roles
 - Binary versus n -ary relationships
 - Modeling n -ary relationships with weak entity sets and binary relationships
 - ISA relationships

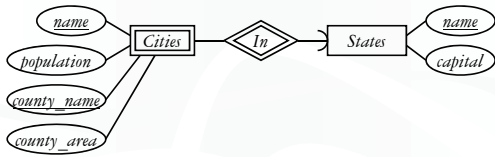
Case study 1

24

- ❖ Design a database representing cities, counties, and states
 - For states, record name and capital (city)
 - For counties, record name, area, and location (state)
 - For cities, record name, population, and location (county and state)
- ❖ Assume the following:
 - Names of states are unique
 - Names of counties are only unique within a state
 - Names of cities are only unique within a county
 - A city is always located in a single county
 - A county is always located in a single state

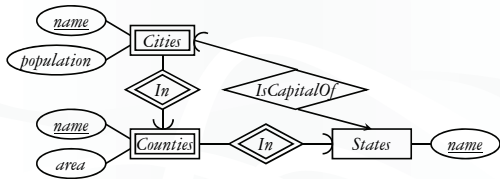
Case study 1: first design

25



Case study 1: second design

26



Case study 2

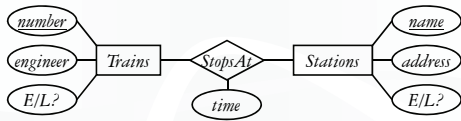
27

❖ Design a database consistent with the following:

- A station has a unique name and an address, and is either an express station or a local station
- A train has a unique number and an engineer, and is either an express train or a local train
- A local train can stop at any station
- An express train only stops at express stations
- A train can stop at a station for any number of times during a day
- Train schedules are the same everyday

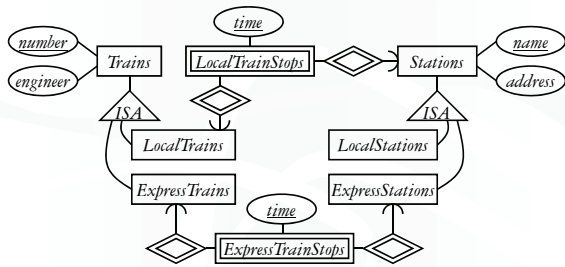
Case study 2: first design

28



Case study 2: second design

29



Is the extra complexity worth it?
