CIS 451/551

week 3: ER models
entity relationship modeling

• a first step in the design of the structure of a database
• many different notations, at varying levels of detail

• entity
  • the main items to be modeled
  • ex: classes, teachers, departments, students

• relationship
  • connections between the entities
  • ex: teach, assign, take
different ER styles

• we will use two in this class
• Chen ER
  • used by this text
  • also many variations

• Crowsfoot
  • one option in MySQL WorkBench
  • closer to the table structure
entity (Chen)

- entity in box
- attributes in ovals
- key(s) underlined
entity (Crowsfoot)

- key has “key” symbol
- attributes have types shown
- filled-in diamonds indicate attribute non-null
relationship (Chen)
cardinality (Chen)

Arrow indicates that car participates in assign relationship at most once; in other words, car assigned to at most one employee.
cardinality 2 (Chen)

- Employee has at most one car
- So when we convert to table, license will be a foreign key in the employee table
cardinality (Crowsfoot)

- car has one employee
- employee can have many cars
total participation (Chen)

bold arrow (or double lined) indicates that all cars participate in assign relationship; so here each car is assigned to exactly on employee
weak entities

- an entity may not have enough information to be uniquely identified
- for example, a dependent in company has a name but we also need to know the essn from the employee table
- a check written on a bank account has a check number, but a bank might process many checks (from different people) with the same check number
- to identify a check we need the check number and the account number, which belongs to the account entity
- here we would say that check is weak and is owned by the account (Chen)
- or that check is in an identifying relationship with account (Crowsfoot)
example bank-check

- bolded entity box means it is *weak*
- dotted underline means *partial key*
- in the table (later), key for check will be (acct#, cnumber)
- bold diamond means *owning* relationship
- check must participate in it, hence the bold arrow

note: the attributes of check belong only to check, the acct# cannot be put there since it belongs to account only
owning relationship

• indicates relationship from the weak entity to the entity whose key is needed to fully identify it

• acct# is needed to identify check (partial key: cnumber)

• essn is needed to identify dependent (partial key: dependent_name)

• in a university, a class might have a class_num (such as 451) but need a dept_code (such as CIS) from the department table
  • so class would be weak and owned by department

• in MySQL WorkBench these are called identifying relationships
bank-check in Crowsfoot

- the solid line is an identifying relationship
- acct_id is brought to check over as a foreign key
- let workbench do that!!
resolving many-to-many relationships

- customer can have many accounts
- account can be held by many customers
- track the date a customer was added to an account
create a bridge table to represent “holds”

CUSTOMER
  cust_id
  cust_name

ACCOUNT
  acct#
  type
  balance

CustAcct
  date
what that would look like as tables

• CUSTOMER: cust_id, cust_name
  • PK: cust_id

• ACCOUNT: acct_num, acct_type, acct_balance
  • PK: acct_num

• CUST_ACCT: cust_id, acct_num, cust_acct_date
  • PK: cust_id, acct_num
  • FK: cust_id references CUSTOMER
  • FK: acct_num references ACCOUNT
bank-check-customer in Crowsfoot

- all relationships are identifying here
- slightly different attributes
three-way relationships

- makes sense if the texts used by a class depend on the instructor
- duplication if they do not depend on instructor (why?)
- fourth normal form (later)
better if text does not depend on instructor:
IS-A Hierarchy

- subclasses are a form of a weak entity
- key to hourly employee is the key of employee
Convert Chen ER Model to Relational Schema

1. create a table for each strong entity
2. create a table for each weak entity, incorporating owning keys as appropriate
   • include IS-A hierarchies here
3. represent all one-to-many relationships with foreign keys
4. create tables for all many-to-many relationships
   • these are the bridge tables
5. create tables for all other multi-way relationships