Introduction to Data Management

Lecture #6
E-R Relational Mapping (Cont.)

Instructor: Mike Carey
mjcarey@ics.uci.edu

It's time again for....

Friday Nights with Databases

Starbucks in Macau!

Brought to you by…

Today's Reminders

- Continue to follow the course wiki page
  - http://www.ics.uci.edu/~cs122a/
- Continue to live by the Piazza page
  - https://piazza.com/uci/spring2018/cs122a/home
- First HW assignment is due today
  - Up to 24 hours to finish with a 20% late penalty
- Next HW assignment is available now
  - Translate E-R **PHLOG** schema into relational form
  - Use our solution schema (out tomorrow at 5pm)

Today's Reminders (Cont.)

- Be careful on what you post in Piazza:
  - ✖ how should I model X in the HW?
    - Propose more general questions
    - Use non-HW related examples
  - Avoid asking repeated questions(especially near the deadline!)
- Maximum class size expanded to 447

From ISA Hierarchies to Relations

- ***Most general and "clean" approach*** (recommended)**:**
  - 3 relations: Employees, Hourly_Emps, and Contract_Emps.
    - *Hourly_Emps*: Every employee recorded in Employees. For hourly emps, ***extra*** info recorded in Hourly_Emps (*hourly_wages, hours_worked, ssn*); delete Hourly_Emps tuple if referenced Employees tuple is deleted.
    - Queries about all employees easy; those involving just Hourly_Emps require a join to access the extra attributes.
- Another alternative: Hourly_Emps and Contract_Emps.
  - ***Ex:*** *Hourly_Emps(ssn, name, lot, hourly_wages, hours_worked)*
  - If each employee must be in one of the two subclasses…
    (Q: Can we always do this, then? A: Not w/o redundancy!)

ISA Hierarchy Translation Options

- I. "Delta table" approach (recommended):
  - *Emps(**ssn**, name, lot)* □ (All Emps partly reside here)
  - *Hourly_Emps(ssn, wages, hrs_worked)*
  - *Contract_Emps(ssn, contractid)*
- II. "Union of tables" approach:
  - *Emps(**ssn**, name, lot)*
  - *Hourly_Emps(ssn, name, lot, wages, hrs_worked)*
  - *Contract_Emps(ssn, name, lot, contractid)*
- III. "Mashup table" approach:
  - *Emps(kind, ssn, name, lot, wages, hrs_worked, contractid)*

Things to consider:
- Expected queries?
- PK/unique constraints?
- Relationships/FKs?
- Overlap constraints?
- Space/time tradeoffs?

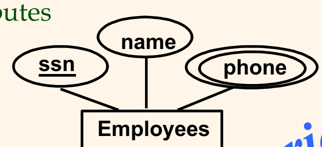
ISA Considerations (cont'd.)

- Query convenience
  - *Ex:* List the names of all Emps in lot 12A
- PK enforcement PK： Primary Key
  - *Ex:* Make sure that ssn is unique for all Emps
- Relationship targets
  - *Ex:* Lawyers table REFERENCES Contract_Emps
- Handling of overlap constraints
  - *Ex:* Sally is under a contract for her hourly work
- Space and query performance tradeoffs
  - *Ex:* List all the info about hourly employee 123
  - *Ex:* What if most employees are "just plain employees"?

如果是某个attribute去开一个table，那么那个attribute所拥有的才是primary key

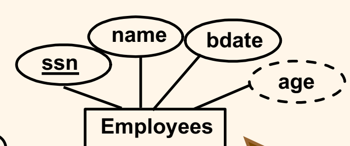
Mapping Advanced ER Features

- Multi-valued (vs. single-valued) attributes



Employees_phones(ssn, phone)
- ssn is an FK in this table
- (ssn, phone) is its PK外面的是属与foreignmkey

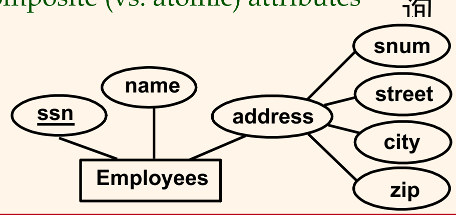
- Derived (vs. base/stored) attributes



Queries...!

查询

- Composite (vs. atomic) attributes



Employees(ssn, name, address_snum, address_street, address_city, address_zip)

SQL Views (and Security)

Primary Key
Unique Key
Foreign Key

- A *view* is just a relation, but we store its *definition* rather than storing the (materialized) set of tuples.

```
CREATE VIEW YoungActiveStudents (name, grade)
    AS SELECT S.name, E.grade
       FROM Students S, Enrolled E
       WHERE S.sid = E.sid and S.age < 21
```

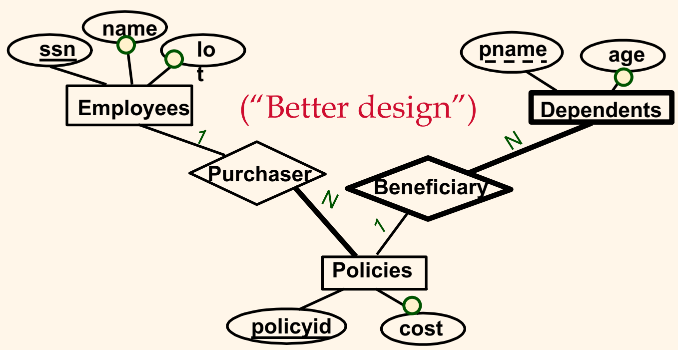
- Views can be used to present needed information while hiding details of underlying table(s). view可以用作只显示某部分data而隐藏重要data
  - Given YoungStudents (but not Students or Enrolled), we can see (young) students *S* who have are enrolled but not see the *cid's* of their courses.

SQL Views (Cont'd.)

- Other view uses in our ER translation context might include:
  - ***Derived attributes, e.g., age (vs. birthdate)***
  - Simplifying/eliminating join paths (for SQL)
  - Beautifying the "Mashup table" approach (to ISA)

```
CREATE VIEW EmployeeView (ssn, name, bdate, age)
    AS SELECT E.ssn, E.name, E.bdate,
              TIMESTAMPDIFF(YEAR, E.bdate, CURDATE( ))
       FROM Employees E
```

Another Mapping Example: Binary vs. Ternary Relationships



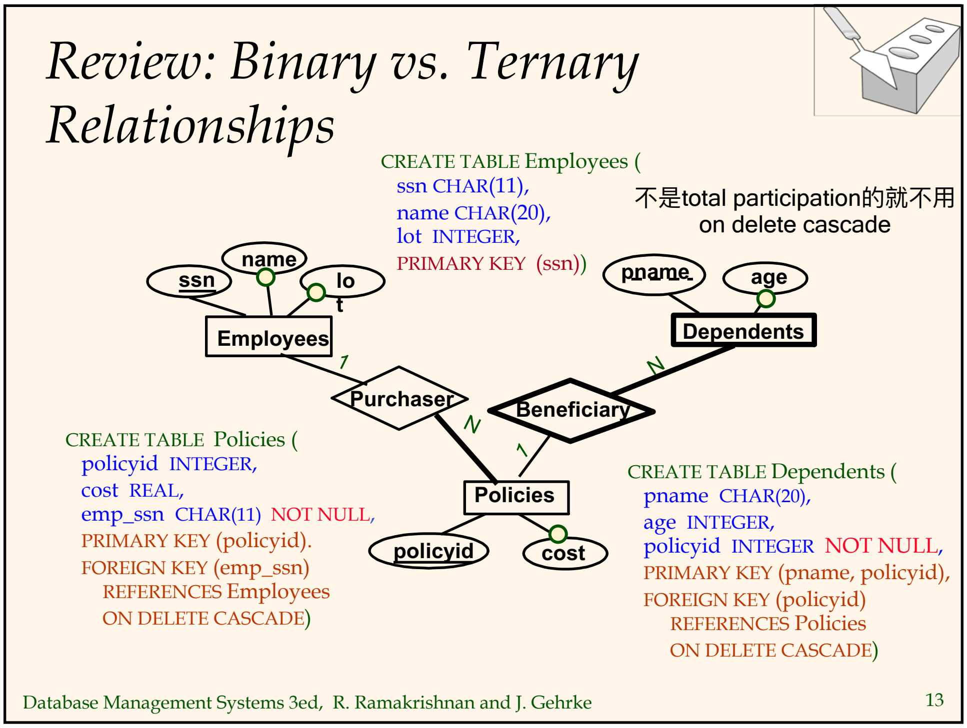
Binary vs. Ternary Relationships (Cont'd.)

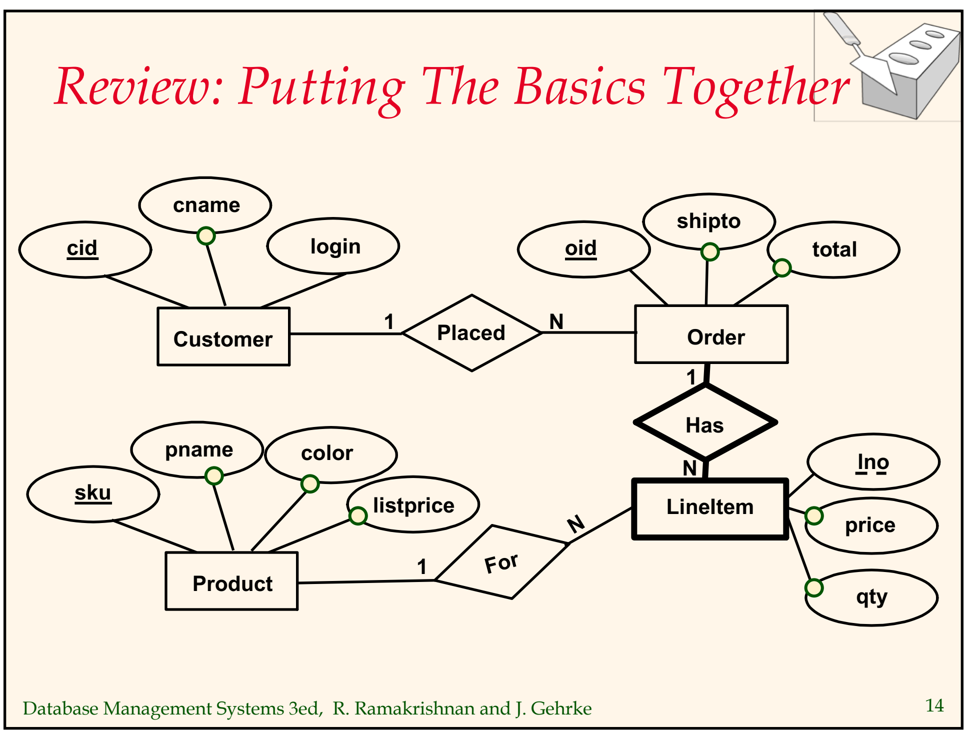
- The key constraints let us combine *Purchaser with Policies* and *Beneficiary with Dependents.*
- Participation constraints lead to NOT NULL constraints.
  (*Note:* Primary key attributes are all NOT NULL as well – check documentation to see if that's implicit or explicit!)

```
CREATE TABLE Policies (
  policyid INTEGER,
  cost REAL,
  emp_ssn CHAR(11) NOT NULL,
  PRIMARY KEY (policyid),
  FOREIGN KEY (emp_ssn) REFERENCES Employees
    ON DELETE CASCADE)
```

```
CREATE TABLE Dependents (
  pname CHAR(20),
  age INTEGER,
  policyid INTEGER NOT NULL,
  PRIMARY KEY (pname, policyid),
  FOREIGN KEY (policyid) REFERENCES Policies
    ON DELETE CASCADE)
```

primary key 和forergn key can be overlap


Review: Binary vs. Ternary Relationships
CREATE TABLE Employees (
ssn CHAR(11),
name CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))
不是total participation的就不用
on delete cascade
name
ssn
lo
t
pname
age
Employees
Dependents
1
N
Purchaser
Beneficiary
N
1
CREATE TABLE Policies (
policyid INTEGER,
cost REAL,
emp_ssn CHAR(11) NOT NULL,
PRIMARY KEY (policyid).
FOREIGN KEY (emp_ssn)
REFERENCES Employees
ON DELETE CASCADE)
Policies
policyid
cost
CREATE TABLE Dependents (
pname CHAR(20),
age INTEGER,
policyid INTEGER NOT NULL,
PRIMARY KEY (pname, policyid),
FOREIGN KEY (policyid)
REFERENCES Policies
ON DELETE CASCADE)
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke
13


Review: Putting The Basics Together
cname
cid
login
oid
shipto
total
Customer
1
Placed
N
Order
1
Has
N
pname
color
sku
listprice
lno
LineItem
price
qty
Product
1
For
N
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke
14

Review: Putting It Together (Cont'd.)

```
CREATE TABLE Customer (
  cid INTEGER,
  cname VARCHAR(50),
  login VARCHAR(20)
    NOT NULL,
  PRIMARY KEY (cid),
  UNIQUE (login))

CREATE TABLE Product (
  sku INTEGER,
  pname VARCHAR(100),
  color VARCHAR(20),
  listprice DECIMAL(8,2),
  PRIMARY KEY (sku))

CREATE TABLE Order (
  oid INTEGER,
  custid INTEGER,
  shipto VARCHAR(200),
  total DECIMAL(8,2),
  PRIMARY KEY (oid),
  FOREIGN KEY (custid) REFERENCES Customer))

CREATE TABLE LineItem (
  oid INTEGER,
  lno INTEGER,
  price DECIMAL(8,2),
  qty INTEGER,
  sku INTEGER,
  PRIMARY KEY (oid, lno),
  FOREIGN KEY (oid) REFERENCES Order
     ON DELETE CASCADE),
  FOREIGN KEY (sku) REFERENCES Product))
```


Reminder: Putting It Together (Cont'd.)

Customer

cid	cname	login
1	Smith, James	jsmith@aol.com
2	White, Susan	suzie@gmail.com
3	Smith, James	js@hotmail.com

Product

sku	pname	color	listprice
123	Frozen DVD	null	24.95
456	Graco Twin Stroller	green	199.99
789	Moen Kitchen Sink	black	350.00

Order

oid	custid	shipto	total
1	3	J. Smith, 1 Main St., USA	199.95
2	1	Mrs. Smith, 3 State St., USA	300.00

LineItem

oid	lno	price	qty	item
1	1	169.95	1	456
1	2	15.00	2	123
2	1	300.00	1	789

Relational Model and E-R Schema Translation: Summary

数据的表格表示

- Relational model: a tabular representation of data.
- Simple and intuitive, also widely used.
- Integrity constraints can be specified by the DBA based on application semantics. DBMS then checks for violations.
  - Two important ICs: Primary and foreign keys (PKs, FKs).
  - In addition, we *always* have domain constraints.
- Powerful and natural query languages exist (soon!)
- Rules to translate E-R to relational model
  - Can be done by a human, or automatically (using a tool)

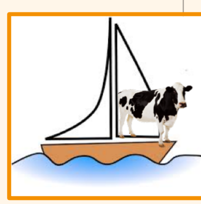
Relational Database Design

- Two aspects to the RDB design problem:
  - *Logical* schema design: We just saw one approach, namely, doing E-R modeling followed by an E-R □ relational schema translation step
  - *Physical* schema design: Later, once we learn about indexes, when should we utilize them?
- We will look at both problem aspects this term, starting first with relational schema design
  - Our power tools will be functional dependencies (FDs) and normalization theory
  - Note: FDs also play an important role in other contexts as well, e.g., SQL query optimization

So, Given a Relational Schema…

- How do I know if my relational schema is a "good" logical database design or not?
  - What might make it "not good"?
  - How can I fix it, if indeed it's "not good"?
  - How "good" is it, after I've fixed it?
- Note that your relational schema might have come from one of several places
  - You started from an E-R model (but maybe that model was "wrong" or incomplete in some way?)
  - You went straight to relational in the first place
  - It's not your schema – you inherited it! ☺

Ex: Wisconsin Sailing Club



Proposed schema design #1:

sid	sname	rating	age	date	bid	bname	color
22	Dustin	7	45.0	10/10/98	101	Interlake	blue
22	Dustin	7	45.0	10/10/98	102	Interlake	red
22	Dustin	7	45.0	10/8/98	103	Clipper	green
22	Dustin	7	45.0	10/7/98	104	Marine	red
31	Lubber	8	55.5	11/10/98	102	Interlake	red
31	Lubber	8	55.5	11/6/98	103	Clipper	green
31	Lubber	8	55.5	11/12/98	104	Marine	red
…	…	…	…	…	…	…	…

Q: Do you think this is a "good" design? (Why or why not?)

nctional dependency属性关系

寻找那些你相同我也相同，你不同我也不同的，然后加上一个主要的key就可以

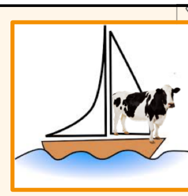
属性之间有三种关系，但并不是每一种关系都存在函数依赖。设R(U)是属性集U上的关系模式，X、Y是U的子集：

如果X和Y之间是1：1关系（一对一关系），如学校和校长之间就是1:1关系，则存在函数依赖X → Y和Y →X。

如果X和Y之间是1：n关系（一对多关系），如年龄和姓名之间就是1:n关系，则存在函数依赖Y → X。

如果X和Y之间是m：n关系（多对多关系），如学生和课程之间就是m:n关系，则X和Y之间不存在函数依赖。

Ex: *Wisconsin Sailing Club*



Proposed schema design #2:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
...	...	...	...

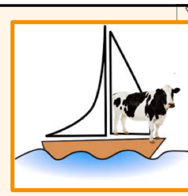
sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
...	...	...

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Q: What about *this* design?

- Is #2 "better than #1...? Explain!
- Is it a "best" design?
- How can we go from design #1 to this one?

Ex: *Wisconsin Sailing Club*



Proposed schema design #3:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
...	...	...	...

sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
...	...	...

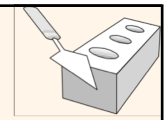
bid	bname
101	Interlake
102	Interlake
103	Clipper
104	Marine

bid	color
101	blue
102	red
103	green
104	red

Q: What about *this* design?

- Is #3 "better" or "worse" than #2...?
- What sort of tradeoffs do you see between the two?

The Evils of Redundancy (or: The Evils of Redundancy)



- *Redundancy* is at the root of several problems associated with relational schemas:
  - Redundant storage (space)
  - Insert/delete/update anomalies
- *Functional dependencies* can help in identifying problem schemas and suggesting refinements.
- Main refinement technique: *decomposition*, e.g., replace R(ABCD) with R1(AB) + R2(BCD).
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - Does the decomposition cause any problems?

Good rule to follow:
"One fact, one place!"