Introduction to Data Management
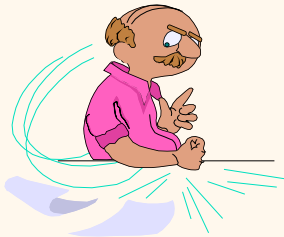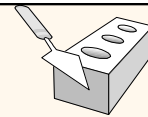
Lecture #6
(Relational Algebra)

Instructor: Mike Carey
mjcarey@ics.uci.edu


Announcements

❖ HW#1 is due *right now*!
- https://grape.ics.uci.edu/wiki/asterix/wiki/cs122a-2014-spring#HomeworkAssignments
- Project Part 1 – again, please register your two-person teams by following Young-Seok's instructions (if you haven't already).
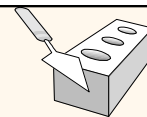
❖ Today's plan:
- Quick peek at what I added to last time's PPT
- Time to move on – next up are relational query languages!

An Example: Putting It Together
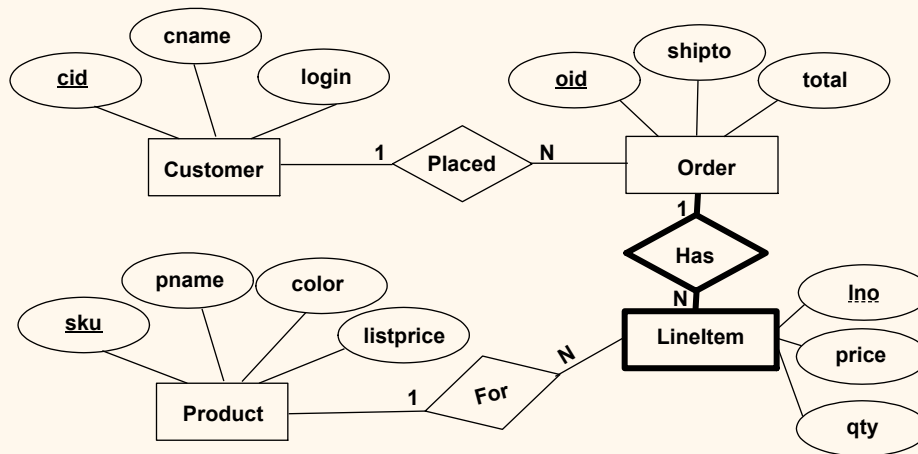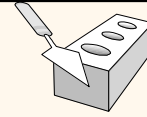
Putting It Together (Cont'd.)

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))

Putting It Together (Cont'd.)
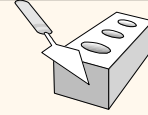
Customer

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
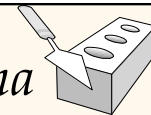
❖ Relations: a tabular representation of data.

❖ Simple and intuitive, also (very!) widely used.

❖ Integrity constraints can be specified by the DBA, based on application semantics. DBMS then checks for violations.
  ▪ Important ICs: primary key, foreign key, unique.
  ▪ In addition, we *always* have domain constraints.

❖ Powerful and natural query languages exist (next!)

❖ Rules to translate E-R schemas to relational schemas
  ▪ Can be done by a human or automatically (using a tool)

Relational Query Languages
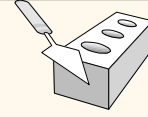
❖ *Query languages:* Allow manipulation and retrieval of data from a database.
❖ Relational model supports simple, powerful QLs:
  ▪ Strong formal foundation based on logic.
  ▪ Allows for much optimization.
❖ Query Languages ≠ programming languages!
  ▪ QLs not expected to be "Turing complete".
  ▪ QLs not intended to be used for complex calculations.
  ▪ QLs support easy, efficient access to large data sets.

Formal Relational Query Languages
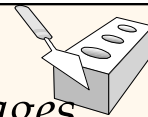
❖ Two mathematical Query Languages form the basis for "real" languages (e.g., SQL), and for implementation:
  ▪ *Relational Algebra*: More operational, very useful for representing execution plans.
  ▪ *Relational Calculus*: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

Preliminaries
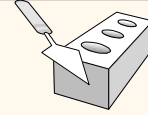
❖ A query is applied to *relation instances*, and the result of a query is also a relation instance.

  ▪ *Schemas* of input relations for a query are fixed (but query will run regardless of instance!)
  ▪ The schema for the *result* of a given query is also fixed! Determined by definition of query language constructs.

❖ Positional vs. named-field notation:

  ▪ Positional notation easier for formal definitions, named-field notation more readable.
  ▪ Both used in SQL (but try to avoid positional stuff!)

Example Instances

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

❖ "Sailors" and "Reserves" relations for our examples.

❖ We'll use positional or named field notation, and assume that names of fields in query results are "inherited" from names of fields in query input relations.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Relational Algebra
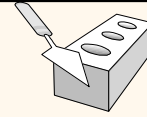
❖ Basic operations:
- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
- *Cross-product* ($\times$) Allows us to combine two relations.
- *Set-difference* ($-$) Tuples in reln. 1, but not in reln. 2.
- *Union* ($\cup$) Tuples in reln. 1 and in reln. 2.

❖ Additional operations:
- Intersection, *join*, division, renaming: Not essential, but (very!) useful. (I.e., don't add expressive power, but…)

❖ Since each operation returns a relation, operations can be ***composed!*** (Algebra is "closed".)

Projection

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$
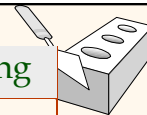
❖ Removes attributes that are not in *projection list*.

❖ *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.

❖ Relational projection operator has to eliminate *duplicates*! (Why??)
- Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (*Q:* Why not?)

age
35.0
55.5

$$\pi_{age}(S2)$$

Selection

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

- ❖ Selects rows that satisfy a *selection condition*.
- ❖ No duplicates in result! (Why?)
- ❖ *Schema* of result identical to schema of (only) input relation.
- ❖ *Result* relation can be the *input* for another relational algebra operation! (This is *operator composition*.)
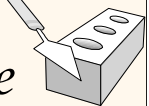
$$\sigma_{rating>8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke   13

Union, Intersection, Set-Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

- ❖ All of these operations take two input relations, which must be *union-compatible*:
  - Same number of fields.
  - "Corresponding" fields are of the same type.
- ❖ What is the *schema* of result?

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

Q: Any issues w/duplicates?

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke   14

Cross-Product
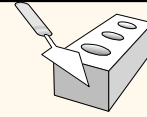
- ❖ Each row of S1 is paired with each row of R1.
- ❖ *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  - ▪ *Conflict*: Both S1 and R1 have a field called *sid*.

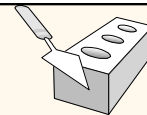| (sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Result relation name

Attribute renaming list

Source expression (anything!)

  - ▪ <u>*Renaming operator*</u>: $\rho\,(C(1 \to sid1, 5 \to sid2),\ S1 \times R1)$

Renaming

- ❖ *Conflict*: S1 and R1 both had *sid* fields, giving:

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
…	…	…	…	…	…	…
58	rusty	10	35.0	58	103	11/12/96

- ❖ Several renaming options available:
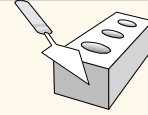
$\rho\,(S1R1(1 \to sid1),\ S1 \times R1)$ ← Positional renaming

$\rho\,(TempS1(sid \to sid1),\ S1)$ ← Name-based renaming
$TempS1 \times R1$

Generalized projection

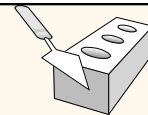$(\pi_{sid \to sid1, sname, rating, age}(S1)) \times R1$

Joins

❖ *Condition Join*: $R \bowtie_c S = \sigma_c(R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{sid<sid} R1$$

❖ *Result schema* same as that of cross-product.

❖ Fewer tuples than cross-product, so might be able to compute more efficiently

❖ Sometimes (often!) called a *theta-join*.
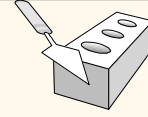

More Joins

❖ *Equi-Join*: A special case of condition join where the condition *c* contains only *equalities.*

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

$$S1 \bowtie_{sid} R1$$

❖ *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.

❖ *Natural Join*: An equijoin on *all* commonly named fields.

To Be Continue…

❖ We'll continue from here next time
- Relational division operator
- Relational algebra query examples
- Then on to relational calculus…