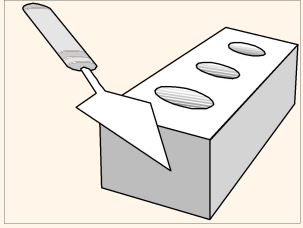


Introduction to Data Management

Lecture #4
E-R Model, Still Going…

Instructor: Mike Carey
mjcarey@ics.uci.edu

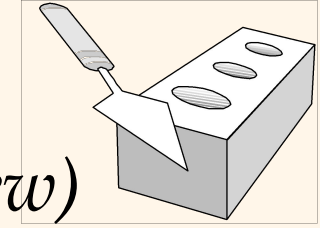
Today's Reminders

- Continue to follow the course wiki page
  - http://www.ics.uci.edu/~cs122a/
- Also follow (and live by) the Piazza page
  - https://piazza.com/uci/spring2019/cs122a/home
- The first HW assignment is underway
  - Conceptual (E-R) database design for ***PHLOG***
- This week's quiz will be on E-R modeling
  - Also bring any lingering HW #1 related questions

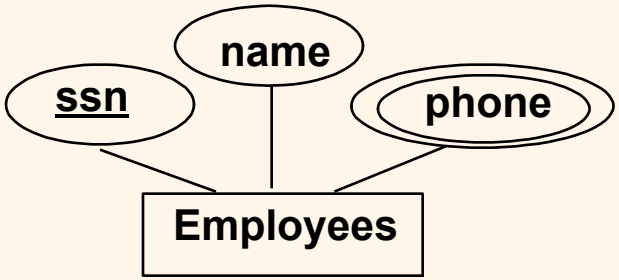
More on Assignment 1

- Due date: April 12th @ 5PM
- Late submission: April 13th @ 5PM with 20% point deduction
- Use software to draw your E-R diagram in the provided template and follow the instructions carefully (including Entity placement!)
- Submit your assignment to Gradescope
- Everyone should have received Gradescope (if not, join using the following code: MZYZW6)

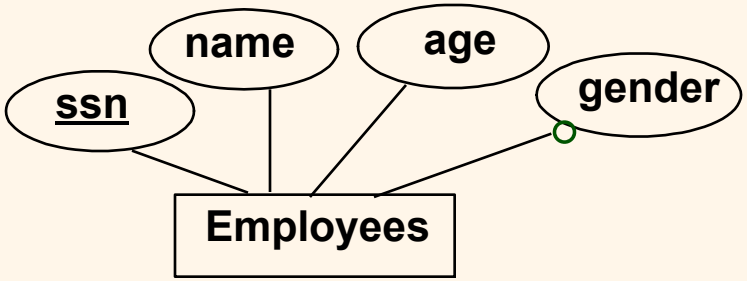
Add'l Advanced ER Features (Review)



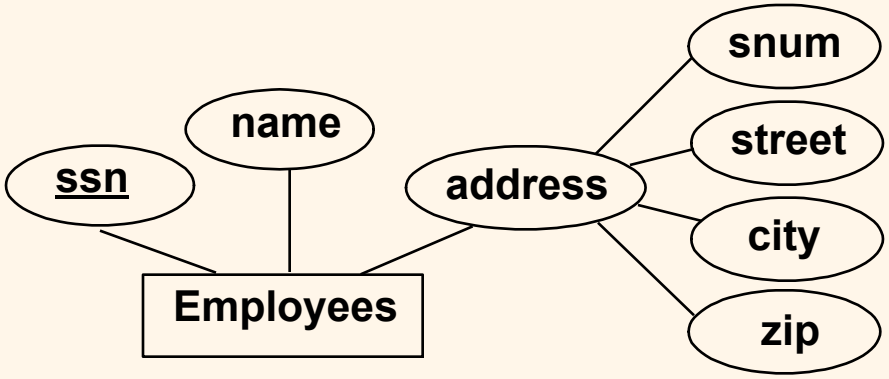
- Multi-valued (vs. single-valued) attributes



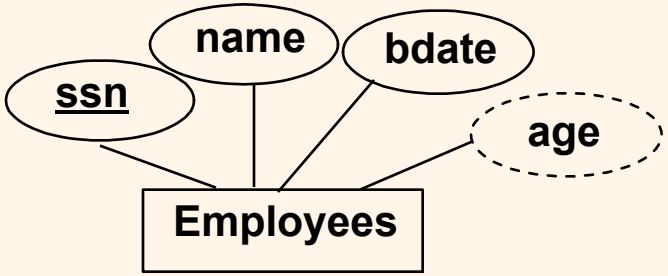
- Optional attributes

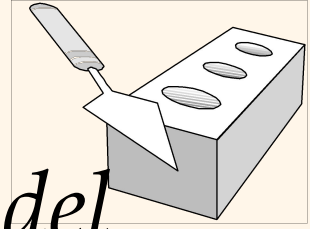


- Composite (vs. atomic) attributes



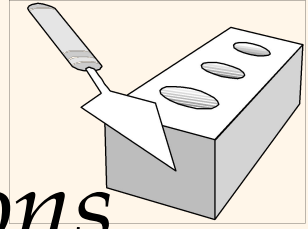
- Derived (vs. base/stored) attributes





Conceptual Design Using the ER Model

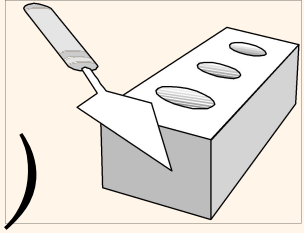
- Design choices:
  - Should a given concept be modeled as an entity or as an attribute?
  - Should a given concept be modeled as an entity or as a relationship?
  - Characterizing relationships: Binary or ternary? Aggregation? …
- Constraints in the ER Model:
  - A lot of data semantics (语义) can (and should) be captured
  - But, not all constraints can be captured by ER diagrams. (*Ex:* Department heads from earlier…!)



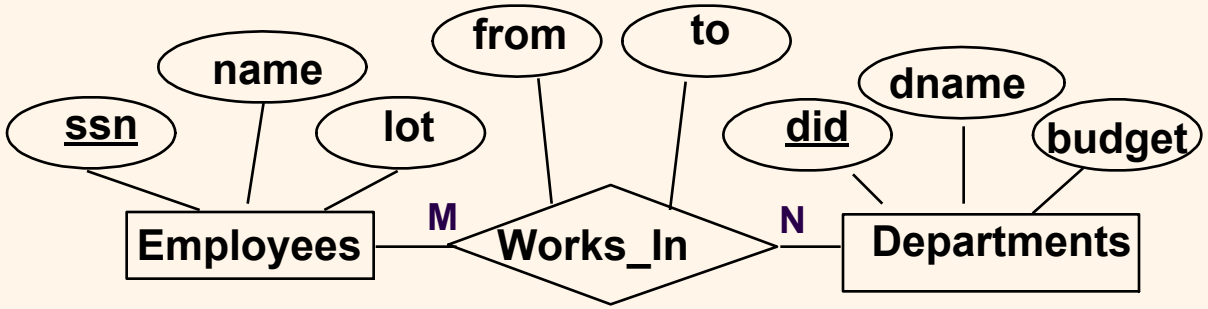
Advanced Attribute Considerations

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends on how we want to use addresses, on the data semantics, and what model features we use:
  - If we have several addresses per employee, *address* would be a multivalued attribute or an entity if we stick only to basic E-R concepts (as they can't be set-valued w/o advanced modeling).
  - If address structure (city, street, etc.) is important, e.g., to query for employees in a given city, *address* should be modeled as a composite attribute or an entity (as attribute values are *atomic* otherwise) – i.e., it shouldn't just be an address string.
  - *If the address itself is logically separate (e.g., representing a property that's located there) and refer-able, it's rightly an entity in any case!*

Attribute Considerations (Cont'd.)

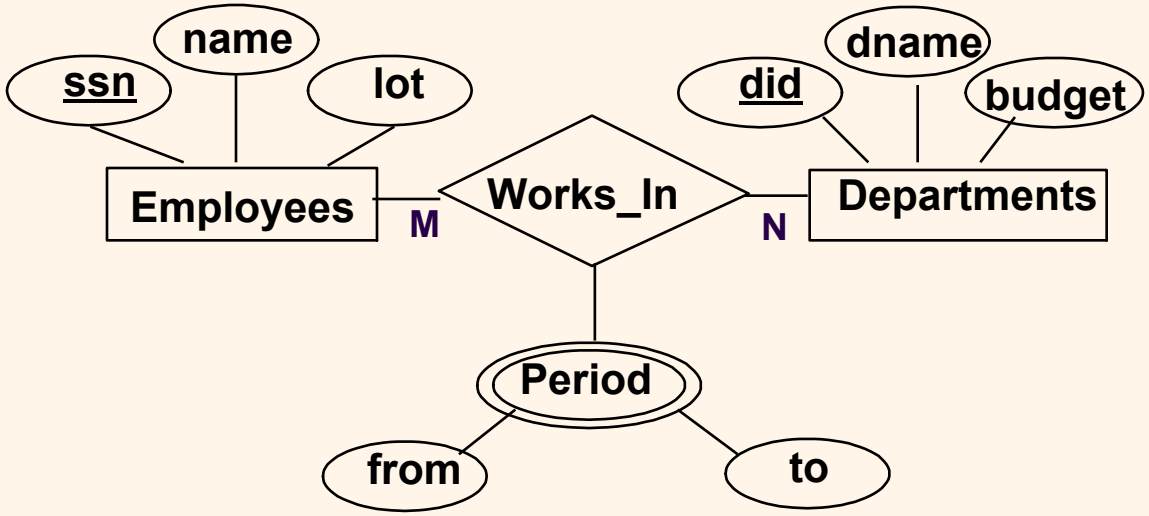


- Works_In here does not allow an employee to work in a department for two or more periods. ***(Q: Why...?)***



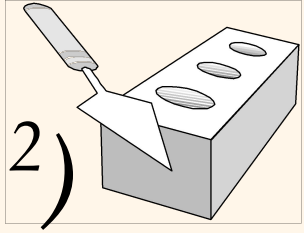
因为他只有一个from一个to

- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this "relationship".* Could be accomplished by having a *multivalued* relationship attribute:



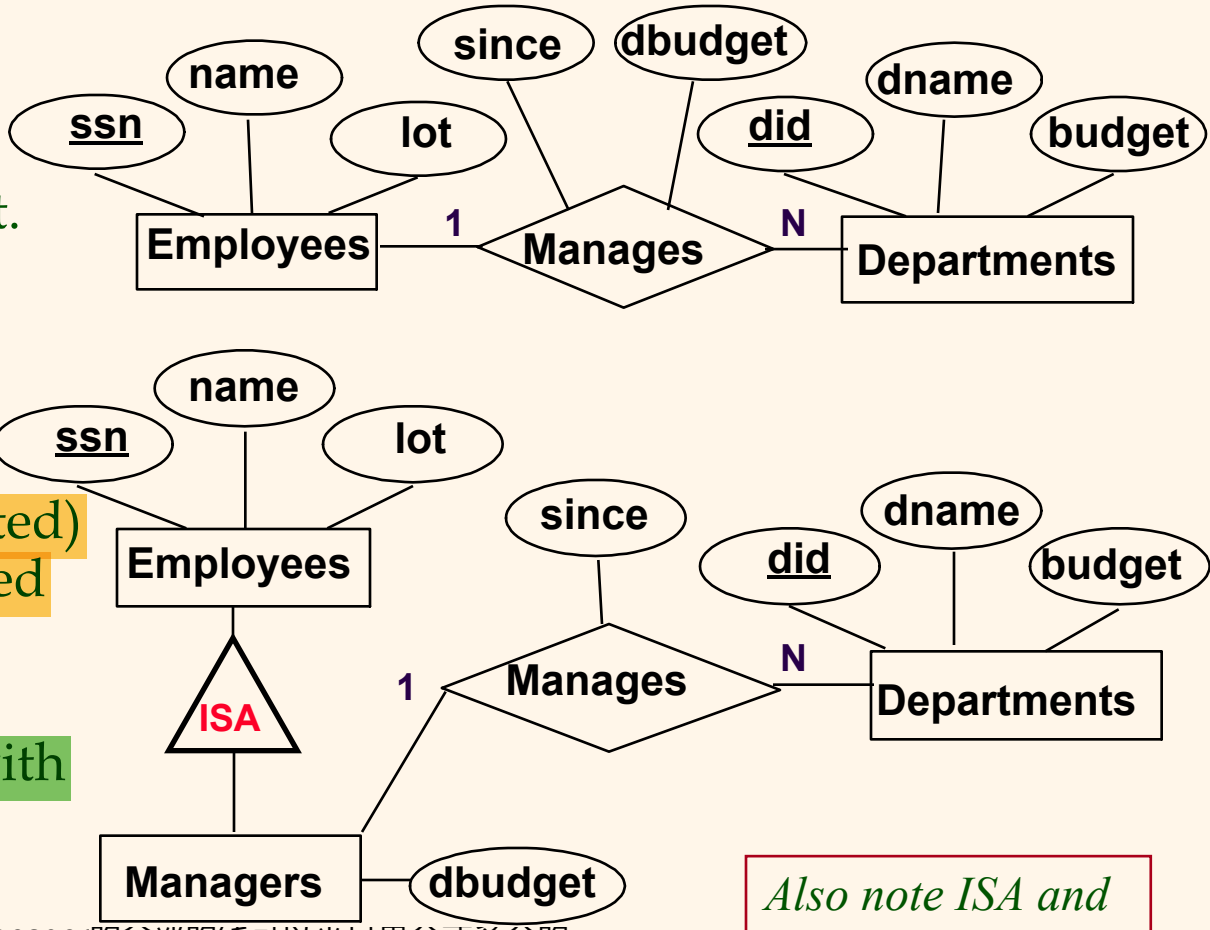
relationship直接的动作会产生副产品，就讲这个副产品为动作的attribute

Attribute Considerations (Cont'd.[2])



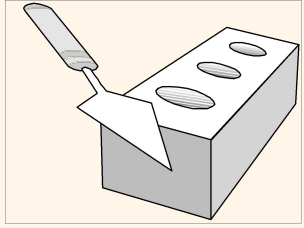
这个dbudget是某个employyes对某个department的dbudget，如果这个员工还有其他department的话，也有其他department的dbuget，所以这里的dbuget是指某个员工对于某个单独一个的department的钱

- ER diagram on the right is okay if a manager gets a separate discretionary budget (dbudget) per dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* could be stored (repeated) with each dept managed by the manager.
  - Misleading: Suggests *dbudget* is associated with department-mgr combination.



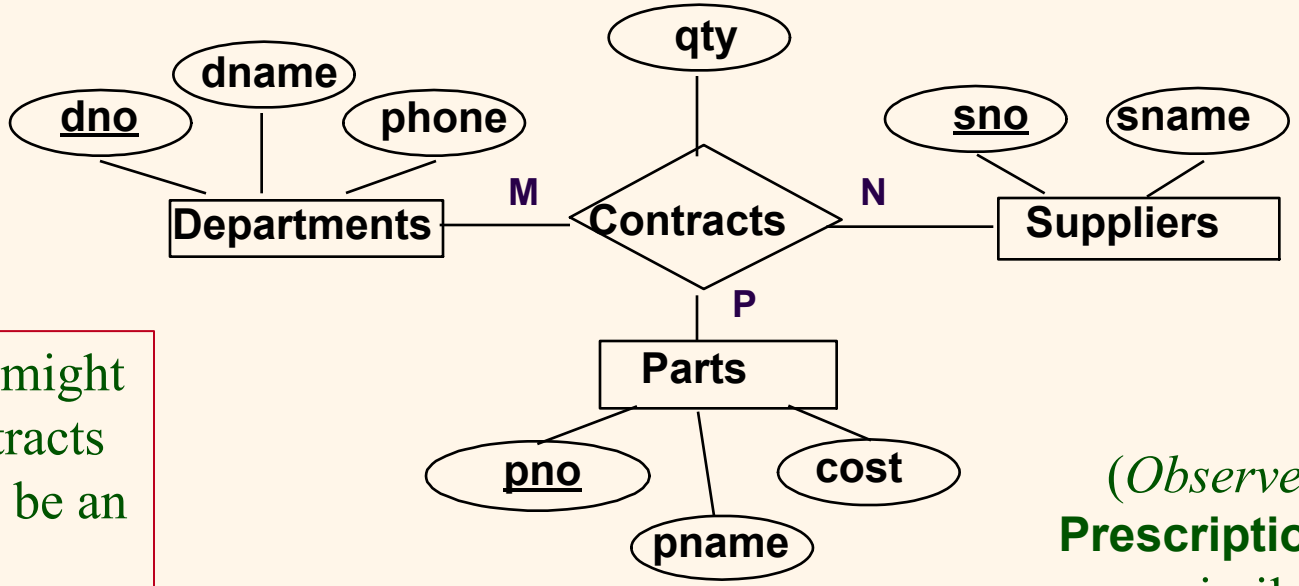
这个是manager的全部的钱可以来自单个或多个的department的钱

Also note ISA and the relationship...



Binary vs. Ternary Relationships

- An example where ternary is needed: a ternary relation Contracts relates the entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*:

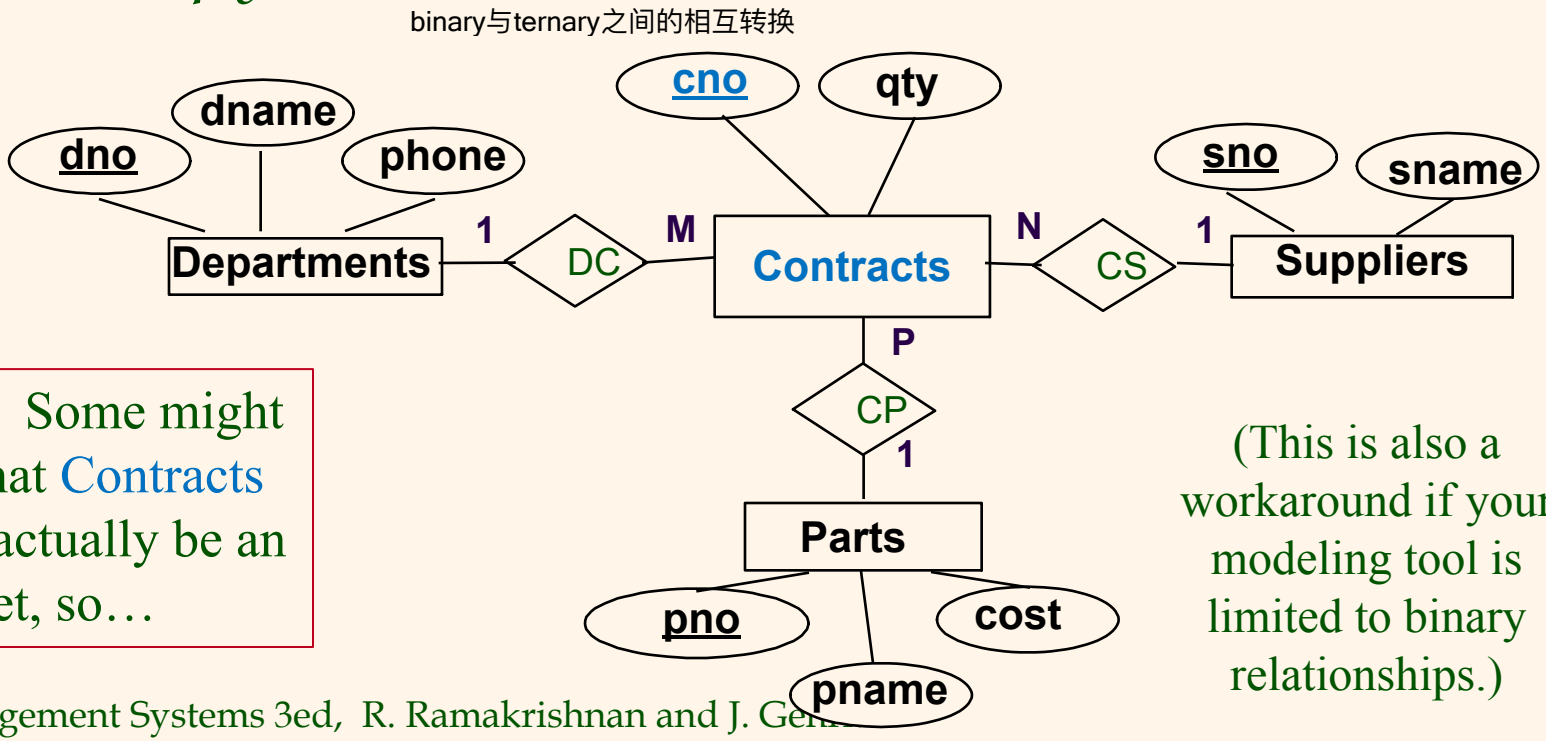


NOTE: Some might argue that Contracts should actually be an entity set…

(*Observe*: **Prescriptions** was similar)

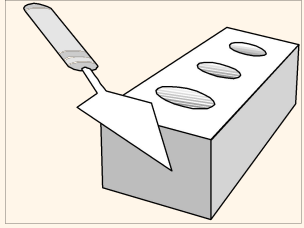
Binary vs. Ternary Relationships (Cont'd.)

- What the entity set perspective would lead to: an entity set Contracts related to the entity sets Parts, Departments and Suppliers, with the descriptive attribute *qty*:



NOTE: Some might argue that Contracts should actually be an entity set, so…

(This is also a workaround if your modeling tool is limited to binary relationships.)

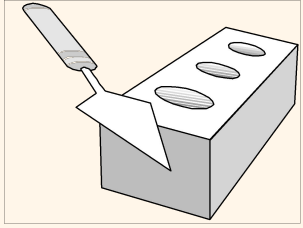


Datatase Design Process (Flow)

- Requirements Gathering (interviews)
- Conceptual Design (using ER)
- Platform Choice (which DBMS?)
- Logical Design (for target data model)
- Physical Design (for target DBMS, workload)
- Implement (and test, of course ☺)

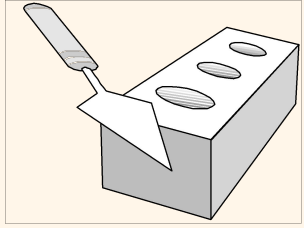
database 设计的步骤

(Expect backtracking, iteration, and also incremental adjustments – and, we will actually be giving you a bit of practice with that last one in the next few HW assignments...! ☺)



Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*
  - Yields a high-level description of data to be stored
- *ER model* popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Additionally: *weak entities*, *ISA hierarchies*, *aggregation*, and *multi-valued*, *composite* and/or *derived* attributes.
- Note: Many variations on the ER model (and many notations in use as well) – and also, UML...

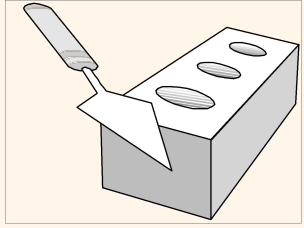


Summary of ER (Cont'd.)

完整性约束

- Several kinds of integrity constraints can be expressed in the ER model: *cardinality constraints*, *participation constraints*, also *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set (more about key constraints will be coming soon).
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

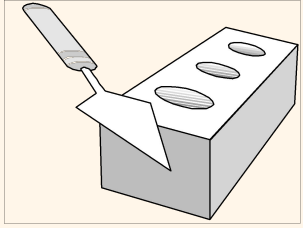
企业



Summary of ER (Cont'd.)

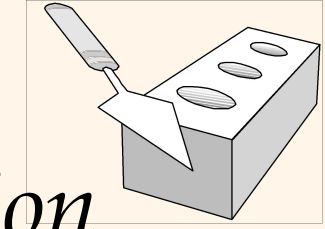
- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use an ISA hierarchy, and whether or not to use aggregation.
- Ensuring good database design: The resulting relational schema should be analyzed and refined further. For this, FD information and normalization techniques are especially useful (coming soon).

FD: funtional dependence



Relational Database: Definitions

- Relational database: a set of *relations*
- Relation: consists of 2 parts:
  - ***Instance***: a table, with rows and columns. #Rows = *cardinality*, #fields = *degree* or *arity*.
  - ***Schema***: specifies name of relation, plus name and type of each column.
    - E.g. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
- Can think of a relation as a *set* of rows or *tuples* (i.e., all rows are distinct) in the pure relational model (*vs.* reality of SQL ☺)



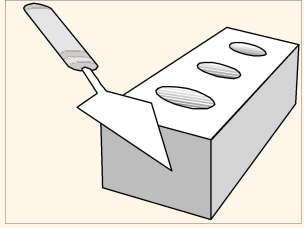
Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5, all rows distinct

行数是 Cardinality = 3，就是这个图总共能有多少种不同

degree是列，表示有多少个attribute

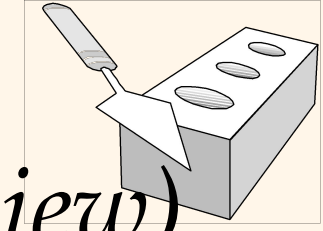


Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise (and set-based) semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

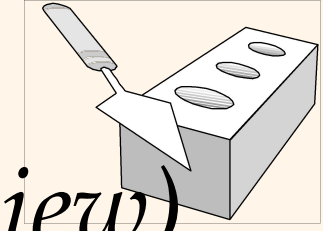
直观的

关系查询的精确（和基于集合）语义



The SQL Query Language (Preview)

- Developed by IBM (System R) in the 1970s
- Need for a standard, since it is used by many vendors (Oracle, IBM, Microsoft, …)
- ANSI/ISO Standards:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision, very widely supported)
  - SQL-99 (major extensions, current standard)



The SQL Query Language (Preview)

- To find all 18 year old students, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18
```

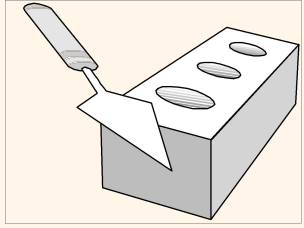
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

select是指就找到什么，就把那一整列列出来，相符的都列出来

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

如果只要名字和login的话，那就可以是select S.name和S.login



Querying Multiple Relations

- What does the following query compute?

```
SELECT  S.name, E.cid
FROM  Students S, Enrolled E
WHERE  S.sid=E.sid AND E.grade="A"
```

Given the following instances of Students and Enrolled:

这个是 Student S

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

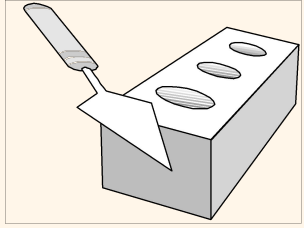
这个是Enrolled E

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

b不符合

We will get:

S.name	E.cid
Smith	Topology112



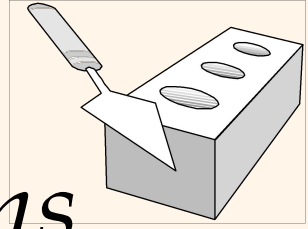
Creating Relations in SQL

这样的可以create一个table

- Create the Students relation. Observe that the type (*domain*) of each field is specified and enforced by the DBMS whenever tuples are added or modified.
- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Students
    (sid CHAR(20),
     name CHAR(20),
     login CHAR(10),
     age INTEGER,
     gpa REAL)
```

```
CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(20),
     grade CHAR(2))
```



Destroying and Altering Relations

DROP TABLE Students

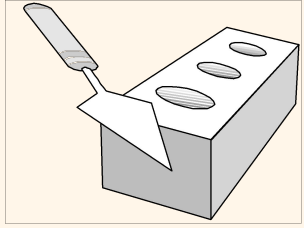
drop table 用来delete

- Destroys the relation Students. The schema information *and* the tuples are deleted.

ALTER TABLE Students
ADD COLUMN firstYear integer

加一个alter table 可以往后面加一个column（列），然后值都为null

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.



Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

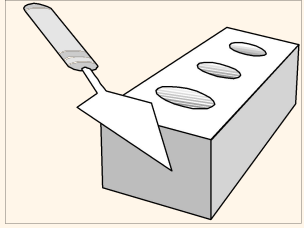
可以增加一行数据

- Can delete all tuples satisfying some condition (e.g., name = Smith):

可以减去所有tuple满足以下关系

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

- *Powerful variants of these commands are available; more later!*



Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors (centrally), too!