JACOBS UNIVERSITY

The Relational Model

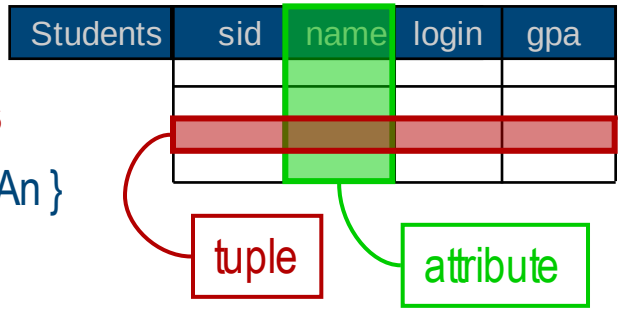
Relational Database: Definitions

- Technically: Relation made up of 2 parts:
  - Schema: specifies name of relation, plus name and type of each column
    - Ex: Students(sid: string, name: string, login: string, gpa: real)
  - Instance: a table, with rows and columns
    - *# rows = cardinality, # fields = degree / arity*

does not change often

changes all the time

- Mathematically:
  - Let A1, …, An (n>0) be value sets, called attribute domains
  - relation $R \subseteq A1 \times \ldots \times An = \{ (a1,\ldots,an) \mid a1 \in A1, \ldots, an \in An \}$



- Can think of a relation as a set of rows or tuples
  - i.e., all rows are distinct = no duplicates (hmm…)
  - atomic attribute types only – no fancies like sets, trees, …
- Relational database: a set of relations

Querying Relational Databases

- major strength of relational model: simple, powerful *querying* of data
  - Data organised in tables, query results are tables as well
  - Small set of generic operations, work on any table structure
- Query describes structure of result ("what"), not algorithm how result is achieved ("how")
  - data independence, optimizability
- Queries can be written intuitively, DBMS responsible for efficient evaluation
  - key: precise (mathematical) semantics for relational queries
  - Allows optimizer to extensively re-order operations

SQL, Structured English Query Language

- Here: DML = Data Manipulation Language
- "all students with GPA<3.6"
  - SELECT *
    FROM Students S
    WHERE S.gpa < 3.6

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

- To find just names and logins, replace the first line:
  - SELECT S.name, S.login

~~sid~~	name	login	~~gpa~~
53666	Jones	jones@cs	3.4
53688	Smith	smith@eecs	3.2

name	login
Jones	jones@cs
Smith	smith@eecs

SQL Joins: 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*:

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

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53666	Topology112	A
53688	History105	B

- …we get?

DML: Modifying the Database

- insert a single tuple:
  - INSERT INTO Students(sid, name, login, gpa)
    VALUES (53688, ‘Smith’, ‘smith@ee’, 3.2)
- delete all tuples satisfying some condition:
  - DELETE FROM Students S
    WHERE S.name = ‘Smith’
- change all tuples satisfying some condition:
  - UPDATE Students S
    SET gpa = 3.0
    WHERE S.name = ‘Smith’

Integrity Constraints

- Integrity constraint = 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
- Primary Key, Referential Integrity, Multiplicity, CHECK constraints, …
- A legal instance of a relation is one that satisfies all specified ICs
- Goal: data more faithful to real-world meaning
  - Also: avoid some data entry errors

Primary Key, Foreign Key, Referential Integrity

- Primary Key = (set of) attributes identifying tuple in a relation
- Foreign key = (set of) attributes `referring' to tuple in another relation
  - Aka `logical pointer'
- Example: sid is a foreign key referring to *Students*:
  - Enrolled(sid: string, *cid*: string, *grade*: string)
- If all foreign key constraints enforced: referential integrity
  - Can you name a model w/o referential integrity?

Referential Integrity in SQL

- SQL/92 and SQL:1999 options on deletes and updates:
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
    REFERENCES  Students
    ON DELETE CASCADE
    ON UPDATE SET DEFAULT )
```

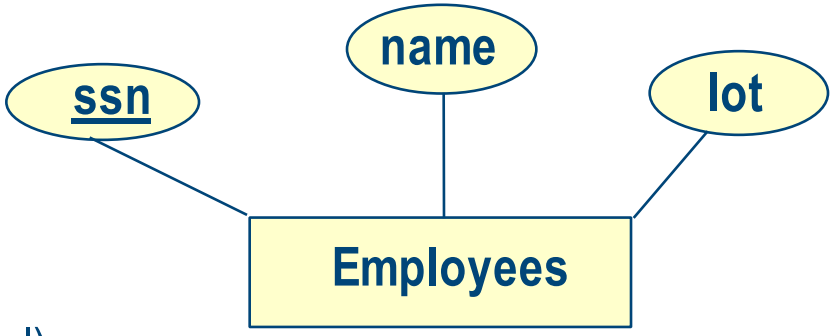
treat corresponding Enrolled tuple when Students (!) tuple is deleted

Where do ICs Come From?

- semantics of real-world enterprise described in database relations
- can check database to see if IC is violated,
  but can NEVER infer that an IC is true by looking at an instance
  - IC = statement about all possible instances!
- Key & foreign key ICs most common; plus more general ICs (later)

ER → Relational: Entity Sets

- Entity sets to tables:
  - ER attribute → table attribute
    (can do that because ER constrained
    to simple types, same as in relational model)
  - Declare key attribute "Primary key"




```
CREATE TABLE Employees
 ( ssn CHAR(11),
   name CHAR(20),
   lot  INTEGER,
   PRIMARY KEY (ssn) )
```

- Best practice (not followed by book):
  Add "abstract" identifying key attribute
  - No further semantics
  - System generated
  - use only this as primary key & for referencing

```
CREATE TABLE Employees
 ( sid INTEGER,
   ssn CHAR(11) UNIQUE,
   …,
   PRIMARY KEY (sid) )
```

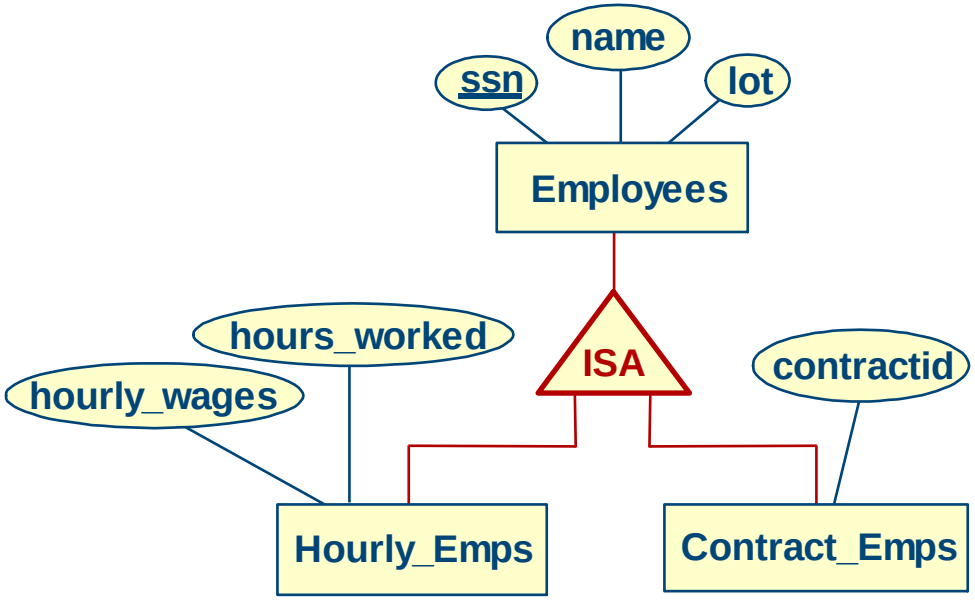
ER → Relational: Relationship Sets

- Keys for each participating entity set (as foreign keys)
- All descriptive attributes

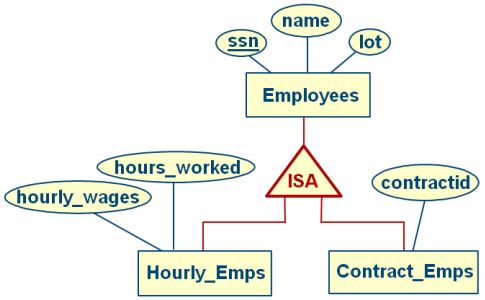
```
CREATE TABLE Works_In
( ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
      REFERENCES Employees,
  FOREIGN KEY (did)
      REFERENCES Departments )
```

ER → Relational: ISA Hierarchies

- H ISA E: every H entity is also E entity
- Mapping to Relations
  - Several choices
  - Constraints determine



ISA Hierarchies: Mapping Variants



- #1: 3 tables
  - Create table E(eid, ssn, name, lot)
    create table H(eid, hwg, hw)
    create table C(eid, cid)
  - „hourly emps“: SELECT ssn, name, lot, hwg, hw FROM E, H WHERE E.eid = H.eid
    „all emps“: SELECT ssn, name, lot FROM E
- #2: 2 tables
  - EH(eid, ssn, name, lot, hwg, hw)
    EC(eid,ssn, name, lot, cid)
  - „hourly emps“: SELECT ssn, name, lot, hwg, hw FROM EH
    „all emps“: (SELECT ssn, name, lot FROM EH) UNION (SELECT ssn, name, lot FROM EC)
- #3: 1 table
  - EHC(eid, ssn, name, lot, isH, hwg, hw , cid) ex: <42, 123213, „John Doe“, 5, false, NULL, NULL, 17>
  - „hourly emps“: SELECT ssn, name, lot, hwg, hw FROM EHC WHERE isH=true
    „all emps“: SELECT ssn, name, lot FROM EHC
  - x

Views

- view = relation, described by query (not stored data)

```
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
```

- Deletion:

```
DROP VIEW YoungActiveStudents
```

Views and Security

- Views useful for personalized information (or a summary), while hiding details in underlying relation(s)
- Given *YoungStudents*, but not *Students* or *Enrolled*, we can find students who are enrolled
- …but not the cid's of the courses they are enrolled in

Relational Model: Summary

- A tabular representation of data
- Simple & intuitive, most widely used
- Integrity constraints can be specified by the DBA, based on application semantics; DBMS checks for violations
  - primary and foreign keys + domain constraints + …
- SQL query language for generic set-oriented table handling
  - Attribute selection ("projection"); set-oriented tuple grabbing ("selection"); joins
- Rules to translate ER to relational model
  - Not all concepts translate 1:1