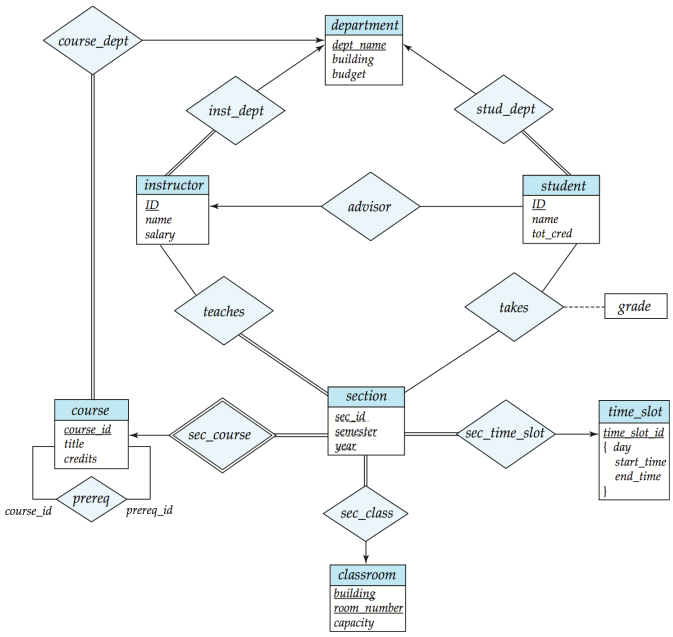
Normal Forms

CISC637, Lecture #9

Ben Carterette



Review: Translating an E-R Diagram to Relational Schema

- Remember the basic steps:
  1. Define a relation for each entity set
     - Primary key of relation = unique identifier of entity
  2. Define a relation for each relationship set
     - Primary key of relation = concatenation of unique identifiers of entity sets involved
     - Define foreign key to relations representing entity sets
  3. Modify keys of relations defined in step 2 to capture mapping constraints as necessary
  4. Eliminate redundant relations
     - Relations that have exactly the same fields and keys
  5. Merge relations with the same keys to capture participation constraints as necessary
  6. Check normal forms and normalize relations as needed

3

Students and Advisors

- University requirements:
  - instructors are identified by an ID num, and we need to store their name and salary
    - every instructor must also be associated with exactly one department
  - students are identified by an ID num, and we need to store their name and total credits
    - every student must also be associated with exactly one department
  - a student can have at most one instructor as advisor
    - instructors can advise any number of students
- FDs to capture these requirements:
  1. i_ID → name, salary
  2. i_ID → dept_name
  3. s_ID → name, tot_cred
  4. s_ID → dept_name
  5. s_ID → i_ID

4

Students and Advisors

- University requirements:
  - instructors are identified by an ID num, and we need to store their name and salary
    - every instructor must also be associated with exactly one department
  - students are identified by an ID num, and we need to store their name and total credits
    - every student must also be associated with exactly one department
  - a student can have at most one instructor as advisor
    - instructors can advise any number of students
- FDs to capture these requirements:
  1. i_ID → name, salary, dept_name
  2. s_ID → name, tot_cred, dept_name
  3. s_ID → i_ID

Students and Advisors

- Change to university requirements:
  - Each instructor is an advisor for at most one department
  - Students can have majors in more than one department
  - Students can have multiple advisors, but at most one advisor per department
- FDs:
  1. i_ID → name, salary
  2. i_ID → dept_name
  3. s_ID → name, tot_cred
  3. s_ID, dept_name -> i_ID
     - given student ID and department, there is at most one possible advising instructor

Normal Forms

- A **normal form** is a set of criteria used to examine a relation for purposes of *optimal* RDB design
  - meant to help identify redundancies in relation
  - determine whether it is subject to logical inconsistencies and anomalies
- There is a series of normal forms of increasing "strictness" and lower chance of inconsistency
  - First normal form (1NF)
  - ~~Second normal form (2NF)~~
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
  - Fourth, fifth, sixth normal forms (4NF, 5NF, 6NF)

A Way to Model New Advising Req

- Possible relational schema:
  - Student(s_ID:int, name:string, tot_cred:int, advisors:list, departments:list)
  - Instructor(i_ID:int, name:string, salary:real, dept_name:string)
- Possible problems:
  - Difficult to maintain
  - Can't enforce referential integrity
  - High potential for invalid information to creep in

First Normal Form

- 1NF is the simplest normal form
- A relation is in 1NF if all of its fields contain only single values
  - No lists or sets
- Importance of 1NF?
  - Avoiding *update anomalies*
  - Maintaining referential integrity
  - Reduces possibility of redundant data

Higher Normal Forms

- Normal forms higher than 1NF are defined in terms of FDs and are concerned with degrees of redundancy
- We will say *a relation is in xNF* if the criteria of *x* normal form are true for every possible instance of that relation
  - e.g. a relation with no list/set attributes is in 1NF
- Hierarchy of normal forms:
  - 4NF ➔ BCNF ➔ 3NF ➔ 2NF ➔ 1NF
  - Every relation in 4NF is also in BCNF, 3NF, 2NF, 1NF by definition

Redundancy?

- `inst_dept` relation

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Functional Dependencies and Redundancy

- inst_dept functional dependencies:
  - Candidate key FD
    - i_ID -> i_ID, name, salary, dept_name, building, budget
  - Superkey FDs
    - i_ID, name -> i_ID, name, salary, dept_name, building, budget
    - i_ID, salary -> i_ID, name, salary, dept_name, building, budget
    - Every possible FD with ID on the left and all fields on the right
  - Trivial FDs
    - i_ID, name -> name
    - name, salary -> salary
    - Any FD where fields on the right are a subset of fields on the left
  - dept_name -> building, budget
    - Not a candidate key, not a superkey, not trivial

Boyce-Codd Normal Form

- List **all** functional dependencies that hold on R
- Each FD should either be:
  - trivial, or
  - a superkey/candidate key
- If any FD on R is non-trivial AND not a key, then R is *not* in Boyce-Codd Normal Form

BCNF?

- `inst_dept` relation

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Boyce-Codd Normal Form

- Intuitively, for a relation to be in BCNF, every field of the relation should be determined by exactly one key
  - That key can have more than one field though
- If any field can be determined from some field that is not part of the key, the possibility of redundancy in data exists
  - Every relation in BCNF implies no redundancy
- BCNF is the most important normal form
  - If all relations are in BCNF you will have a decent relational design
  - But BCNF cannot necessarily capture all requirements

BCNF's Power to Capture Req's

- Change to university requirements:
  - Each instructor is an advisor for at most one department
  - Students can have majors in more than one department
  - Students can have multiple advisors, but at most one advisor per department
- FDs to capture these requirements:
  1. i_ID → name, salary
  2. i_ID → dept_name
  3. s_ID → name, tot_cred
  3. s_ID, dept_name -> i_ID
     - given student ID and department, there is at most one possible advising instructor

Possible Relational Schema

- Possible relational schema #1:
  - Student(s_ID, name, tot_cred)
  - Instructor(i_ID, name, salary, dept_name)
  - DeptAdvisor(s_ID, dept_name, i_ID)
- All relations in BCNF?

Possible Relational Schema

- Possible relational schema #1:
  - Student(s_ID, name, tot_cred)
  - Instructor(i_ID, name, salary, dept_name)
  - DeptAdvisor(s_ID, dept_name, i_ID)
- All relations in BCNF?
  - No!
  - Notice that i_ID → dept_name holds on two separate tables
    - Implies redundancy
  - On DeptAdvisor, that FD is non-trivial, and i_ID is not a key

Possible Relational Schema

- Possible relational schema #2:
  - Student(s_ID, name, tot_cred)
  - StudentDept(s_ID, dept_name)
  - Instructor(i_ID, name, salary, dept_name)
  - Advisor(s_ID, i_ID)
- Are all relations in BCNF?

Possible Relational Schema

- Possible relational schema #2:
  - Student(s_ID, name, tot_cred)
  - StudentDept(s_ID, dept_name)
  - Instructor(i_ID, name, salary, dept_name)
  - Advisor(s_ID, i_ID)
- Are all relations in BCNF?
  - Yes!
- Are FDs preserved?

Possible Relational Schema

- Possible relational schema #2:
  - Student(s_ID, name, tot_cred)
  - StudentDept(s_ID, dept_name)
  - Instructor(i_ID, name, salary, dept_name)
  - Advisor(s_ID, i_ID)
- Are all relations in BCNF?
  - Yes!
- Are FDs preserved?
  - No!
    - s_ID, dept_name → i_ID does not hold on any one relation
  - Three-table join required to check that the advisor is in one of the departments the student is in
  - Cannot easily check that a student has at most one advisor from each department

BCNF Tradeoff

- We can have relations in BCNF (which is good) but lose some of the required dependencies (which is bad)
- We can preserve all of the dependencies (good) but have relations that are not in BCNF (bad)
- We can't always have it both ways

Third Normal Form

- Third Normal Form (3NF) provides a way to retain FDs that cannot be captured in BCNF
  - With minimal additional redundancy
- Checking for 3NF:
  - List all functional dependencies that hold on R
  - Each FD should meet one of the following criteria:
    - be trivial, or
    - be a superkey/candidate key, or
    - have at least one field on the right side that is part of some candidate key for R (and that is not on the left side of the same FD)
  - If any FD violates all three criteria, relation is not in 3NF

Possible Relational Schema

- Possible relational schema #1:
  - Student(s_ID, name, tot_cred)
  - Instructor(i_ID, name, salary, dept_name)
  - DeptAdvisor(s_ID, dept_name, i_ID)
- All relations in 3NF?

Possible Relational Schema

- Possible relational schema #1:
  - Student(s_ID, name, tot_cred)
  - Instructor(i_ID, name, salary, dept_name)
  - DeptAdvisor(s_ID, dept_name, i_ID)
- All relations in 3NF?
  - Yes! Student and Instructor are in BCNF, therefore in 3NF by definition
  - What about DeptAdvisor, which is not in BCNF?
    - i_ID → dept_name holds on DeptAdvisor
      - dept_name on the right is part of candidate key
        s_ID, dept_name → s_ID, dept_name, i_ID

Another Example

- Employees and managers
  - Employees have ID numbers, names, and salaries
  - An employee can split time between different departments
  - An employee is managed by exactly one manager in each department
  - A manager only manages employees in one department
- Some FDs:
  - empID → name, salary
  - mgrID → dept
  - empID, dept → mgrID, time
- Employee(empID, name, salary) is in BCNF
- Is schema EmpMgr(empID, dept, mgrID, time) in 3NF?
- Is schema EmpMgr(empID, dept, mgrID, time) in 3NF?

Another Example

- Employees and managers
  - Employees have ID numbers, names, and salaries
  - An employee can split time between different departments
  - An employee is managed by exactly one manager in each department
  - A manager only manages employees in one department
  - **Every employee must have at least one manager**
- FDs:
  - empID → name, salary
  - mgrID → dept
  - empID, dept → mgrID, time
- Possible schema:
  - Employee(empID, name, salary, dept, mgrID, time), with mgrID NOT NULL
  - 3NF?

Another Example

- Employees and managers
  - Employees have ID numbers, names, and salaries
  - An employee can split time between different departments
  - An employee is managed by exactly one manager in each department
  - A manager only manages employees in one department
  - **Every employee must have at least one manager**
- FDs:
  - empID → name, salary
  - mgrID → dept
  - empID, dept → mgrID, time
- Possible schema:
  - Employee(empID, name, salary, dept, mgrID, time), with mgrID NOT NULL
  - 3NF?
    - No: empID → name, salary is not a candidate key, and the right-hand fields are not part of any candidate key

Another Example

- New university requirement
  - All students must have exactly one advisor
  - All instructors must advise exactly one student
  - Student and instructor in same department
- FDs:
  - s_ID → name, tot_cred
  - i_ID → name, salary
  - s_ID → i_ID
  - i_ID → s_ID
  - s_ID, i_ID → dept
- Possible schema:
  - StudentInstructor(s_ID, s_name, tot_cred, i_ID, i_name, salary, dept)
    … or StudentInstructor(s_ID, s_name, tot_cred, i_ID, i_name, salary, dept)
  - 3NF? Yes to both! BCNF? Yes to both! Good design? Probably not.

Database Design Recap So Far

- Redundancy: bad
  - Introduces possibility of invalid or inconsistent data
- Incompleteness: bad
  - Database that doesn't capture requirements shifts work to application designers, users
- Functional dependencies are a formal way to state requirements
  - Requirements -> functional dependencies
  - An FD is *true* iff it correctly capture some requirement
  - The truth of an FD has nothing to do with relational schema
  - An FD may or may not *hold* on a relation
- Normal forms are a formal way to talk abt redundancy/possibility for inconsistency
  - 1NF -> lots of potential for redundancy, but at least we can maintain referential integrity
  - 3NF -> a little bit of redundancy so that some non-candidate key FDs can hold
  - BCNF -> [almost] no redundancy, FDs that hold are primary key FDs
- Tradeoff (illustrated in university advising example):
  - We can avoid redundancy, but not all FDs hold
  - We can ensure all FDs hold, but have possibility of redundancy