Databases and Information Systems
CS303

Normalization
06-09-2023

Recap

- E-R Diagrams
- Translate E-R Diagrams to Relational Schema

Goal

- Generate Relational Schema:
  - Without redundancy
  - Allows easy information retrieval

- Achieved using Normal Forms

Design Alternatives: Larger Schemas

- inst_dept (ID, name, salary, dept_name, building, budget)
  - Represents Natural Join of instructor and department
    - Reduces join in some queries

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

Design Alternatives : Larger Schemas

- inst_dept (ID, name, salary, dept_name, building, budget)
  - building and budget are redundant
  - Department with no instructors cannot be represented (unless we use nulls)

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

Design Alternatives : Smaller Schemas

- inst_dept (ID, name, salary, dept_name, building, budget)
  - Can we break this into two smaller schemas?
  - How to spot redundancy?

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

Design Alternatives : Smaller Schemas

- inst_dept (ID, name, salary, dept_name, building, budget)
  - Enterprise should specify Functional Dependency

    dept_name $\longrightarrow$ building
  - This information can be used to decompose the database
  - There is a well defined way to perform such decompositions

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

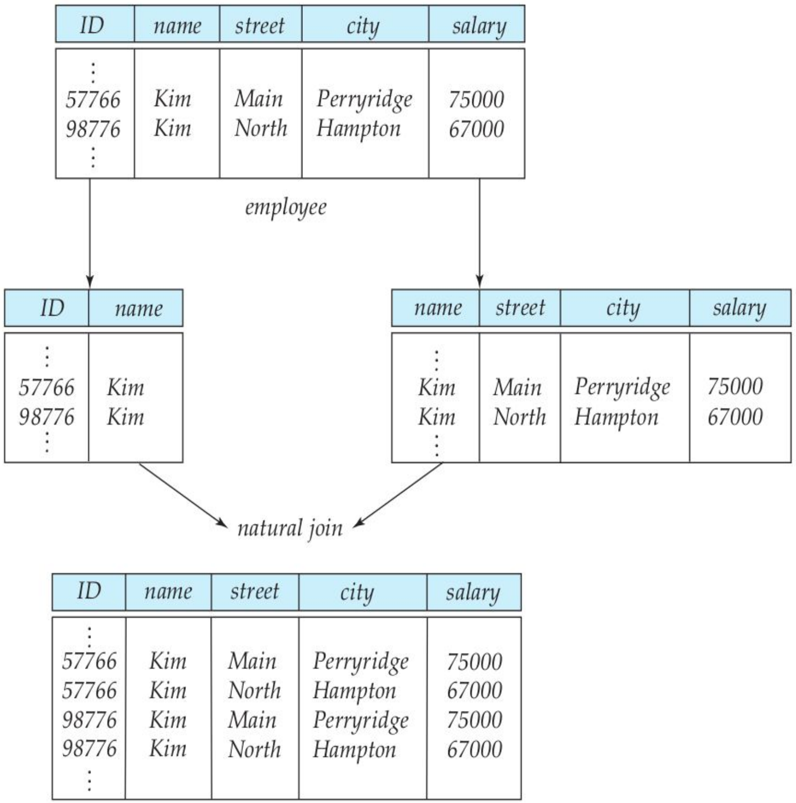
Design Alternatives : Smaller Schemas

- Not all decompositions are useful:
  - employee (ID, name, street, city, salary)
    - employee1 (ID, name)
    - employee2 (name, street, city, salary)

- There is a loss of information
  - Which Kim is in Main ?
  - Which Kim gets salary 75000 ?

- Such decompositions are called Lossy decompositions

- Goal is to always obtain Lossless decompositions



Lossless decomposition

- Let $r(R)$ be a relational schema with functional dependency set F
  - Let R_1 and R_2 be the decomposition of R
  - The decomposition is lossless if there is no information loss when $r(R)$ is replaced with $r_1(R_1)$ and $r_2(R_2)$
  - Formally : Decomposition of R to R_1 and R_2 is lossless if :

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

  - Otherwise, the decomposition is a lossy decomposition

First Normal Form : Atoms

- E-R Diagrams allow identification of Multivalued Attributes and Composite Attributes

- Relational Schema eliminates the substructures and creates attributes with atomic domains

- Relational Schema R is in First Normal Form (1NF) if all the attributes have atomic domain

- Other Normal Forms need better understanding of Functional Dependency

Functional Dependency

- Real world data comes with many constraints:
    - Students and instructors are uniquely identified by their ID
    - Each student and instructor has only one name
    - Each instructor and student is (primarily) associated with only one department
    - Each department has only one value for its budget, and only one associated building
- Legal Instance satisfies all the constraints
- Most common type of constraints are Key constraints and Functional Dependencies

Functional Dependency

- Superkey: A set of attributes K of a relational schema $r(R)$ is said to be a superkey if for any two tuples t_1 and t_2 if $t_1 \neq t_2$ then $t_1[K] \neq t_2[K]$

- Functional Dependency allows us to specify constraints over certain values of the attributes

- Let R be the set of all attributes of the relational schema $r(R)$ and let $\alpha \subseteq R$ and $\beta \subseteq R$. Given an instance of r,
we say $\alpha \rightarrow \beta$ if for any two tuples t_1 and t_2 if $t_1[\alpha] = t_2[\alpha]$ then $t_1[\beta] = t_2[\beta]$

- Example: inst_dept (ID, name, salary, dept name, building, budget)
  - dept_name → building, budget
  - ID, dept_name → name, salary, building, budget

Functional Dependency : Uses

- To test instances of relations to see whether they satisfy a given set F of functional dependencies

- To specify constraints on the set of legal relations

Functional Dependency

- $A \rightarrow C$?
- $C \rightarrow A$?
- $A,D \rightarrow B$?
- Some Trivial functional dependencies:
  - $A \rightarrow A$
  - $AB \rightarrow A$
- In general if $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$ always holds

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4

Functional Dependency

- room_number → capacity
  - Holds for this instance, but need not hold in general
- building, room_number → capacity
  - More Natural

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Functional Dependency

- If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$
- For a set of attributes F, denote F^+ to be the closure of F
  (set of all functional dependencies that can be inferred from F)

Boyce Codd Normal Form

- Eliminates redundancies based on functional dependencies
- A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$: at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$)
  - α is a superkey for schema R
- A database design is in BCNF if every relational schema of the design is in BCNF

Boyce Codd Normal Form

- inst_dept (ID, name, salary, dept_name, building, budget)
    - dept_name → building
        - This is a non-trivial dependency and dept_name is not a superkey

- instructor (ID, name, salary, dept_name)
    - This is in BCNF
- department (dept_name, building, budget)
    - This is in BCNF

- We need a generic way to do such a decomposition

Boyce Codd Normal Form : General decomposition Rule

- Let r(R) be a schema that is not in BCNF
  (There is a non-trivial functional dependency $\alpha \rightarrow \beta$ where α is not a superkey)

- Replace r(R) with two new schemas
  - $\alpha \cup \beta$
  - $(R - (\beta - \alpha))$

- Example: inst_dept (ID, name, salary, dept_name, building, budget)
  - dept_name → building, budget

- Application of the rule might result in smaller relations that are not in BCNF
  - Repeat for procedure for smaller relations till the resulting schema is in BCNF

Boyce Codd Normal Form : Dependency Preservation

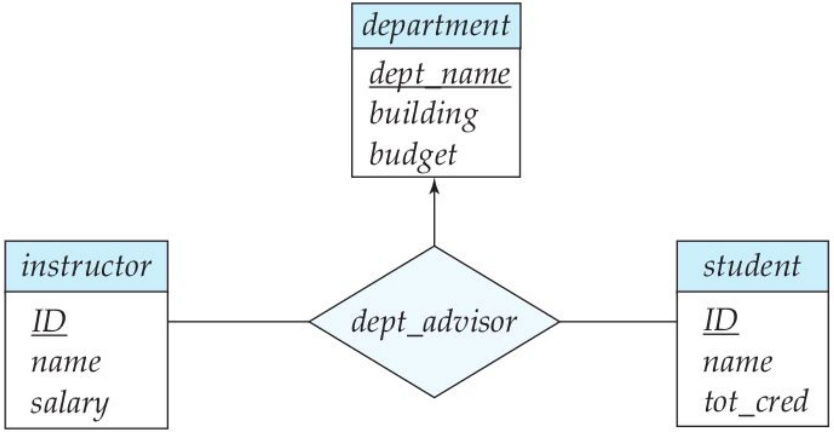
- Database comes with many constraints

- Suppose each student can have double majors (major/minor) and a student can have one advisor per department.

- dept_advisor (s_ID, i_ID, dept_name)

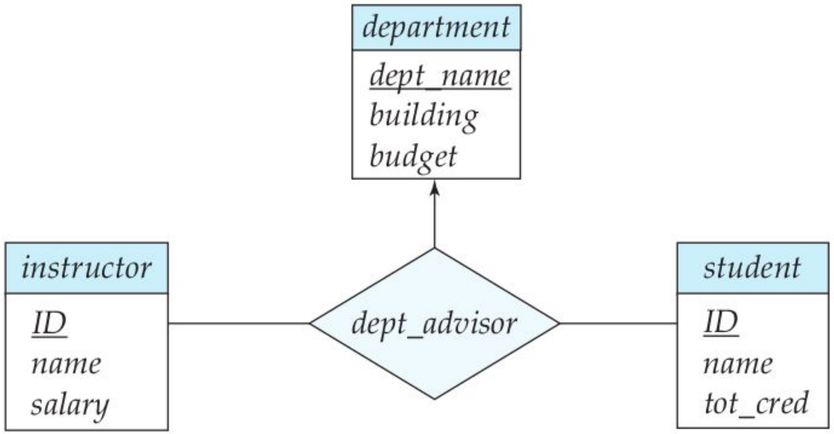
    - s_ID, dept_name → i_ID
    - i_ID → dept_name
      (every instructor can be advisor
       for a single department)

    - Not in BCNF because i_ID is not superkey



Boyce Codd Normal Form : Dependency Preservation

- dept_advisor (s_ID, i_ID, dept_name)
  - s_ID, dept_name → i_ID
  - i_ID → dept_name
    (every instructor can be advisor
    for a single department)
  - Not in BCNF because i_ID is not superkey
  - (s_ID, i_ID)
  - (i_ID, dept_name)
  - s_ID, dept_name → i_ID
    decomposition makes it computationally hard to enforce this functional dependency



- BCNF is not dependency preserving
  - 3NF (weaker than BCNF) is always dependency preserving

Third Normal Form

- A relation schema r(R) is in Third Normal Form (3NF) with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$: at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$).
  - α is a superkey for schema R.
  - Each attribute A in $\beta - \alpha$ is contained in some candidate key of R

- Each such A can be part of different candidate keys

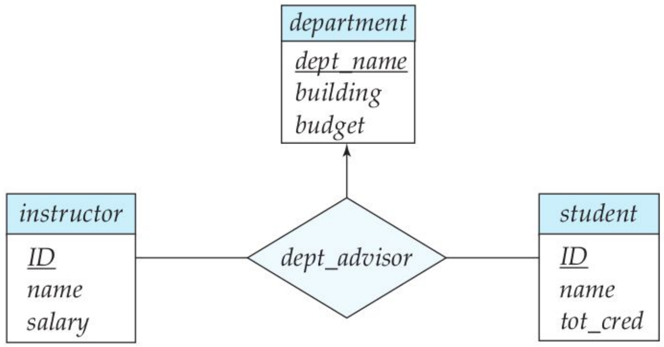
- Any schema that is in BCNF is also in 3NF

Third Normal Form

- A relation schema r(R) is in Third Normal Form (3NF) with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$: at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$).
  - α is a superkey for schema R.
  - Each attribute A in $\beta - \alpha$ is contained in some candidate key of R

- dept_advisor (s_ID, i_ID, dept_name) is in 3NF
  - s_ID, dept_name $\rightarrow$ i_ID
  - i_ID $\rightarrow$ dept_name

    (every instructor can be advisor for a single department)



Decomposition Algorithms

- Real life schema diagrams are typically Huge
- Cannot be done manually
- We need Algorithms that take a schema as an input and decompose them into BCNF or 3NF
- For that we need subroutines to compute F^+, Check if a set of attributes is a superkey