CSc 484
Database Management Systems

Ken Gamradt

Spring 2024

Normalization (I)

Goal of Relational Database Design

- Generate a set of relation schemas that allows us to
  - store information without unnecessary redundancy
  - retrieve information easily
- A suitable set of relation schemas has minimal redundancy
  - Each attribute represented only once
    - Important exception of attributes that form all or part of foreign keys,
      - Essential for the joining of related relations
- Data redundancy may cause update anomalies
  - Insertion
  - Deletion
  - Modification

Relational Schemas for the University Database

- Generated from the E-R model

classroom(*<u>building</u>*, *<u>room_number</u>*, *capacity*)
department(*<u>dept_name</u>*, *building*, *budget*)
course(*<u>course_id</u>*, *title*, *dept_name*, *credits*)
instructor(*<u>ID</u>*, *name*, *dept_name*, *salary*)
section(*<u>course_id</u>*, *<u>sec_id</u>*, *<u>semester</u>*, *<u>year</u>*, *building*, *room_number*, *time_slot_id*)
teaches(*<u>ID</u>*, *<u>course_id</u>*, *<u>sec_id</u>*, *<u>semester</u>*, *<u>year</u>*)
student(*<u>ID</u>*, *name*, *dept_name*, *tot_cred*)
takes(*<u>ID</u>*, *<u>course_id</u>*, *<u>sec_id</u>*, *<u>semester</u>*, *<u>year</u>*, *grade*)
advisor(*<u>s_ID</u>*, *i_ID*)
time_slot(*<u>time_slot_id</u>*, *<u>day</u>*, *<u>start_time</u>*, *end_time*)
prereq(*<u>course_id</u>*, *<u>prereq_id</u>*)

- The goodness (or badness) of the resulting schema depends on how good the E-R design was

Repetition-of-Information

- Suppose we have this from the E-R diagram
  - *inst_dept (ID, name, salary, dept_name, building, budget)*

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

Decomposition

- The only way to avoid the repetition-of-information problem in the ***inst_dept*** schema is to decompose it into two schemas
  - ***instructor***
  - ***department***

    inst_dept (ID, name, salary, dept_name, building, budget)

  into

    instructor (ID, name, salary, dept_name)

    department (dept_name, building, budget)

- **Lossless decomposition**
  - There is no loss of information by replacing a relation schema with two smaller relation schemas

Decomposition

- Lossless decomposition
  - If R1, R2 are two relation schemas decomposed from relation schema R, we say R1 and R2 form a lossless decomposition of R:
    - R1 ∩ R2 forms a superkey
    - R1 ∩ R2 forms a superkey for R2

R *inst_dept (ID, name, salary, dept_name, building, budget)*

R1 *instructor (ID, name, salary, dept_name)*

R2 *department (dept_name, building, budget)*

superkey (primary key) R1 ∩ R2 = *dept_name*

Decomposition

- Example

R *employee (ID, name, street, city, salary)*

R1 *employee1 (ID, name)*

R2 *employee2 (name, street, city, salary)*

R1 ∩ R2 = *name*

NOT a superkey for any decomposed relation schema

- **Lossy decomposition**
  - May cause the loss of information

Functional Dependencies

- Describes the relationship between attributes in a relation
- For a relation schema R (A, B, C, …), B **is functionally dependent on** A if each value of A is associated with exactly one value of B
  - A and B may each consist of one or more attributes
  - In A $\rightarrow$ B, A is called **determinant**
  - A $\rightarrow$ B is also described as "**A functionally determines B**"
- The main method to identify functional dependencies is based on the user's requirement specification or similar sources
  - The meaning of attributes
  - The relationships between attributes

Functional Dependencies

- Identify functional dependencies
  - Based on a sample relation instance which represent all possible data values that the database may hold

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

$A \rightarrow C$

$D \rightarrow B$

Functional Dependencies

- Identify functional dependencies
  - Based on the information provided by the enterprise (requirement analysis)
    - The meaning of each attribute
    - The relationships between the attributes
  - Common sense or experience

inst_dept (ID, name, salary, dept_name, building, budget)

Functional dependencies

$ID \rightarrow$ *name, salary, dept_name, building, budget*

Requirements:

1. Instructors are uniquely identified by their ID
2. Each instructor has only one name
3. Each instructor works for only one department
4. Each department has only one budget, and only one building

Functional Dependencies

student (ID, name, dept_name, tot_cred)

$\Downarrow$

ID → name, dept_name, tot_cred

course(course_id, title, dept_name, credits)

$\Downarrow$

course_id → title, dept_name, credits

Functional Dependencies

- Trivial dependency
  - A → B is trivial if B ⊆ A
  - ⊆ means subset of

student (ID, name, dept_name, tot_cred)

Trivial dependency: -- right side is a subset to the left side

- ID, name → name
- name → name
- …

Functional Dependencies

- Full functional dependency:
  - A → B is a **full functional dependency** if removal of any attribute from A results in the dependency no longer existing
  - A → B is a **partial functional dependency** if some attributes can be removed from A and the dependency still holds

student (<u>ID</u>, name, dept_name, tot_cred)

- *ID, name* → *dept_name* is a **partial** dependency because *ID* → *dept_name*

classroom (<u>building</u>, <u>room_number</u>, capacity)

- *building, room_number* → *capacity* is a **full** dependency because the removal of attributes from the left side makes the dependency not exist

Functional Dependencies

- When specifying the functional dependencies of a relation schema, **only** list **non-trivial, fully functional dependencies**
- Functional dependencies can be used to **identify the primary key** for the relation schema

inst_dept (ID, name, salary, dept_name, building, budget)

- Functional dependencies:

  ID → *name, salary, dept_name, building, budget*

  dept_name → *building, budget*

- *ID* is a primary key (candidate key) of *inst_dept* because all other attributes are functionally dependent on *ID*

Functional Dependencies

- **Armstrong's axioms** (rules of inference)
  - The set of inference rules specifies how functional dependencies can be inferred from given one

Reflexivity	if $B \subseteq A$, then $A \rightarrow B$
Augmentation	if $A \rightarrow B$, then $A, C \rightarrow B, C$
Transitivity	if $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
Self-Determination	$A \rightarrow A$
Decomposition	if $A \rightarrow B, C$, then $A \rightarrow B$ and $A \rightarrow C$
Union	if $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow B, C$

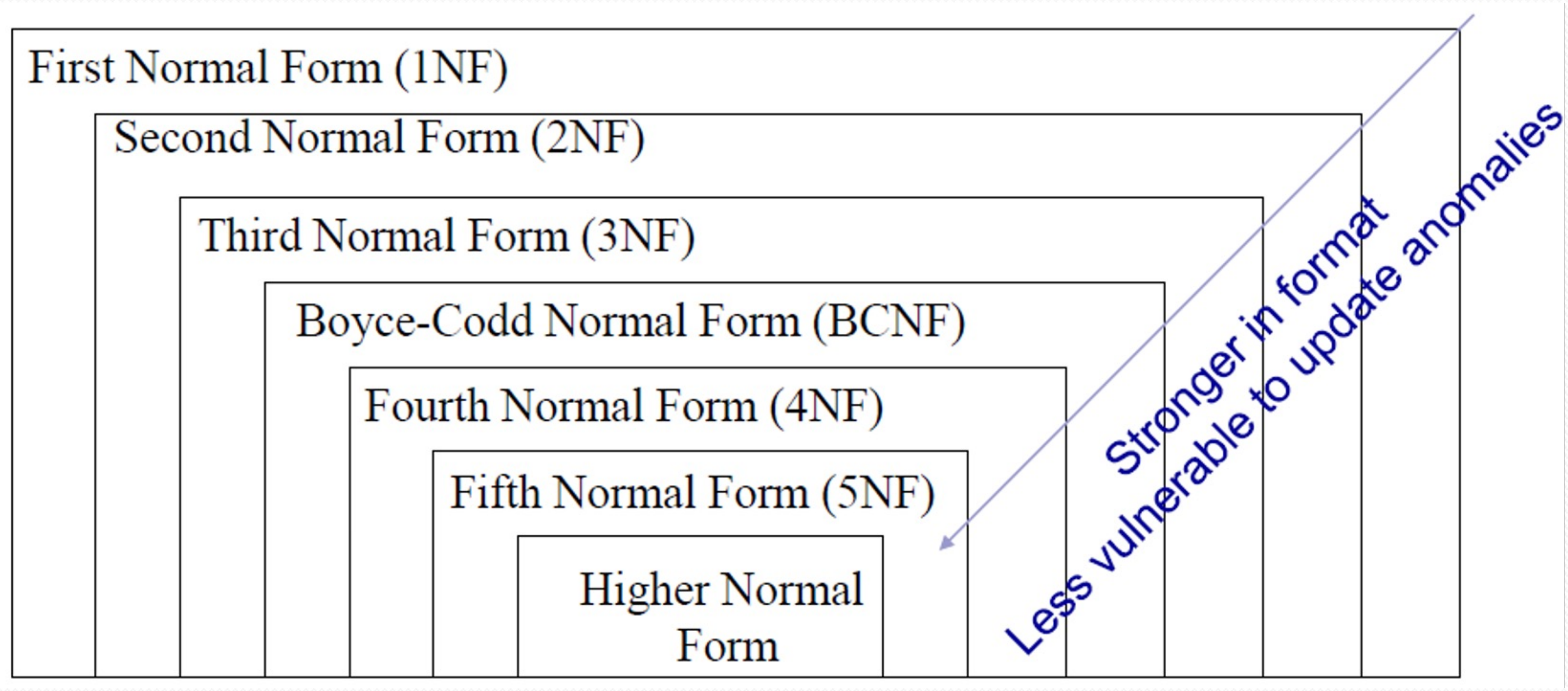
Normalization

- A technique for producing a set of relations with desirable attributes
  - Given the data requirements of an enterprise
- The process of normalization
  - Decide if a given relation schema is in “good form”
    - There are a number of different forms (called normal forms)
  - If a given relation schema is not in “good form”, then we decompose it into a number of smaller relation schemas, each of which is in an appropriate normal form
    - The decomposition **must** be a **lossless** decomposition

Normalization

- Can be used as a validation technique to check the structure of relations
  - Such as relation schemas gotten from an E-R design
- Can be used as a standalone database design technique
  - A bottom-up design approach
  - Start with a single or several relation schemas containing all attributes that are of interest
  - Decompose the relation schemas into smaller schemas with no data redundancy

Normal Forms



Normalization

- Each normal form has some requirements to test the relation schema
- Relation schemas in higher normal forms are less likely to have redundant information
- All normal forms except 1NF are based on functional dependencies
- Normalization is often executed as a series of steps
  - Each step corresponds to a specific normal form
  - In general, process until relation schemas in 3NF or BCNF

Normalization

- Start with a set of relation schema
- Test relation schemas to determine whether or not they satisfy or violate requirements of a given normal form
- If not satisfied
  - Decompose into smaller relation schemas

- Purpose:
  - Guarantees no redundancy
  - Guarantees no update anomalies

Normalization as a Validation technique

- Example

inst_dept (<u>ID</u>, name, salary, dept_name, building, budget)

- Relation schema for the resulting E-R design
- Choose BCNF as its normal form

Boyce-Codd Normal Form (BCNF)

- A relation schema R is in BCNF if for all functional dependencies of the form $A \rightarrow B$, where $A \subseteq R$ and $B \subseteq R$, holds at least one of the following:
  - $A \rightarrow B$ is a **trivial** functional dependency
    - I.e., $B \subseteq A$
  - A is a superkey for schema R

- A database design is in BCNF if each member of the set of relation schemas that make up the design is in BCNF

- **Note**: functional dependencies are said to be **trivial** because they are satisfied by all relations

BCNF

- Example of a relational schema that is **NOT** in BCNF

inst_dept (<u>ID</u>, name, salary, dept_name, building, budget)

- The functional dependency *dept_name* → *budget* holds on *inst_dept*
  - *dept_name* is **NOT** a **superkey**

BCNF

- The *instructor* schema is in BCNF
- All nontrivial functional dependencies hold

$\boldsymbol{ID}$ $\rightarrow$ name, dept_name, salary -- superkey & primary key

- The *department* schema is in BCNF
- All nontrivial functional dependencies hold

dept_name $\rightarrow$ *building, budget* -- superkey & primary key

BCNF

- We now state a general rule for decomposing schemas that are not in BCNF
- Let *R* be a schema that is not in BCNF
- Then there is at least one nontrivial functional dependency $\alpha \rightarrow \beta$ such that α is **NOT** a **superkey** for *R*
- We replace *R* in our design with two schemas:
  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$

BCNF

- Example

  inst_dept (<u>*ID*</u>, ***name, salary, dept_name, building, budget***)

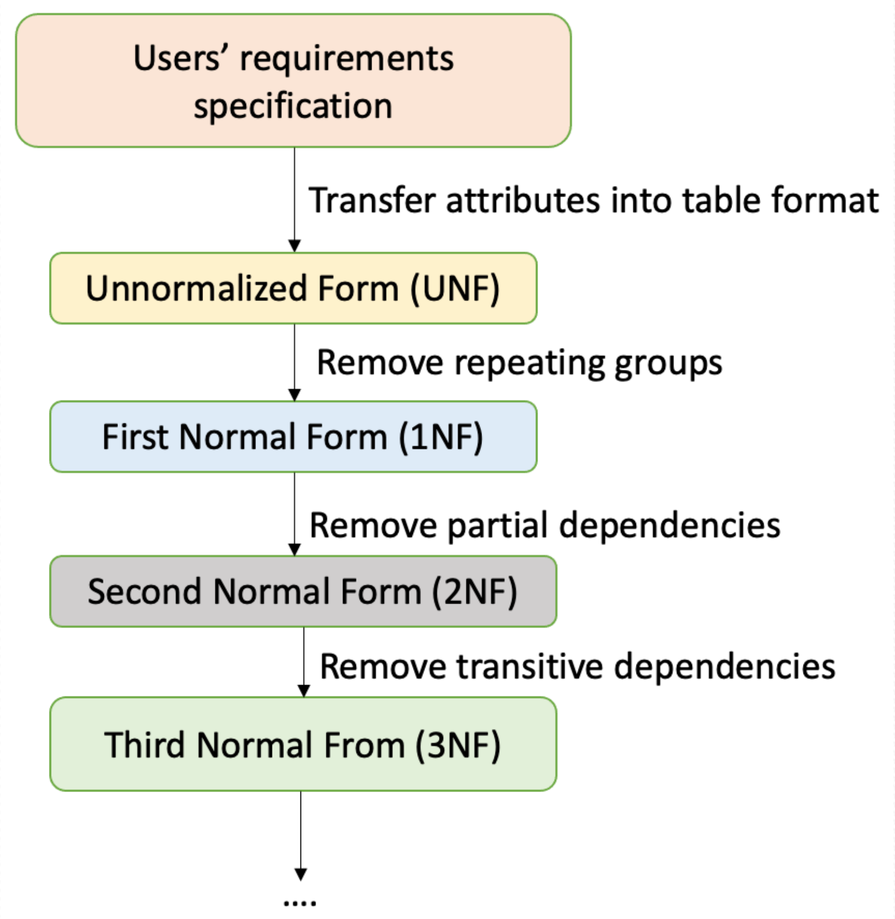
- Functional dependencies:
  - *ID* → *name, salary, dept_name, building, budget* R
  - ***dept_name*** → ***building, budget*** $A \rightarrow B$

  departmart (***dept_name***, ***building, budget***) = (A U B)

  instructor (***ID, name, salary, dept_name***) = (R – (B – A))

  = (R – B + A)

Database Design using Normalization



Unnormalized Form (UNF)

- Transfer attributes into table form

Course_Section (course_id, title, credits, sec_id, semester, year, building, room_number, time_slot_id, day, start, end)

Course_Section

Course_ID	Title	Credits	Sec_id	Semester	Year	Building	Room_num	Capacity	Time_slot_id	Day	Start	End
CSC346	OOP	3	01 01	Spring Fall	2018 2017	SDEH SDEH	109 201	30 45	A B	Tue Wed	9:00 12:00	10:00 13:00
CSC484	DBMS	3	01 02	Fall Spring	2017 2018	SDEH SDEH	302 118	30 42	A C	Tue Thu	9:00 11:00	10:00 12:00

Notes:

- *Room_num* should be *Room_number*

Unnormalized Form (UNF)

- A relation contains one or more repeating group
- **Key**: refer to the attribute(s) that uniquely identify each row within the UNF

Course_Section

Course_ID	Title	Credits	Sec_id	Semester	Year	Building	Room_num	Capacity	Time_slot_id	Day	Start	End
CSC346	OOP	3	01 01	Spring Fall	2018 2017	SDEH SDEH	109 201	30 45	A B	Tue Wed	9:00 12:00	10:00 13:00
CSC484	DBMS	3	01 02	Fall Spring	2017 2018	SDEH SDEH	302 118	30 42	A C	Tue Thu	9:00 11:00	10:00 12:00

- To simplify, one *time_slot_id* relates to only one time slot

First Normal Form (1NF)

- A relation in which the intersection of each row and column contains one and only one value
- From UNF to 1NF: remove the repeating groups
  - Approach 1:
    - Enter appropriate data in the empty columns of rows containing the repeating data

Course_Section

Course_ID	Title	Credits	Sec_id	Semester	Year	Building	Room_num	Capacity	Time_slot_id	Day	Start	End
CSC346	OOP	3	01	Spring	2018	SDEH	109	30	A	Tue	9:00	10:00
CSC346	OOP	3	01	Fall	2017	SDEH	201	45	B	Wed	12:00	13:00
CSC484	DBMS	3	01	Fall	2017	SDEH	302	30	A	Tue	9:00	10:00
CSC484	DBMS	3	02	Spring	2018	SDEH	118	42	C	Thu	11:00	12:00

First Normal Form (1NF)

- Specify the primary key by functional dependencies

Course_Section (course_id, title, credits, sec_id, semester, year, building, room_number, time_slot_id, day, start, end)

- Functional dependencies:

  course_id → *title, credits*

  course_id, sec_id, semester, year →

  building, room_number, capacity, time_slot_id, day, start, end

  building, room_number → *capacity*

  time_slot_id → *day, start, end*

- Primary key: *(course_id, sec_id, semester, year)*

First Normal Form (1NF)

- Approach 2:
  - Place the repeating data in a separate relation
    - Along with the original key attribute(s)

course (<u>course_id</u>, title, credits)

section (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room_number, time_slot_id, day, start, end)

Course

Course_ID	Title	Credits
CSC346	OOP	3
CSC346	OOP	3
CSC484	DBMS	3
CSC484	DBMS	3

Section

Course_ID	Sec_id	Semester	Year	Building	Room_num	Capacity	Time_slot_id	Day	Start	End
CSC346	01	Spring	2018	SDEH	109	30	A	Tue	9:00	10:00
CSC346	01	Fall	2017	SDEH	201	45	B	Wed	12:00	13:00
CSC484	01	Fall	2017	SDEH	302	30	A	Tue	9:00	10:00
CSC484	02	Spring	2018	SDEH	118	42	C	Thu	11:00	12:00

Second Normal Form (2NF)

- A relation that is currently in first 1NF
  - Every non-primary key attribute is fully functionally dependent on the primary key
  - If a partial dependency exists, remove the partially dependent attribute(s) from the relation by placing them in a new relation
    - Along with a copy of their determinant

Second Normal Form (2NF)

- 1NF → 2NF

Course_Section (*<u>course_id</u>*, *title*, *credits*, *<u>sec_id</u>*, *<u>semester</u>*, *<u>year</u>*, *building*, *room_number*, *time_slot_id*, *day*, *start*, *end*) -- 1NF

- Functional properties: -- dependencies ???

  course_id → title, credits -- (title, credits) partially dependent on primary key

  -- course_id, sec_id, semester, year → title, credits

  course_id, sec_id, semester, year → title, credits, building, room_number, capacity, time_slot_id, day, start, end

  building, room_number → capacity

  time_slot_id → day, start, end

First Normal Form (1NF)

- 1NF → 2NF

course (course_id, title, credits)

section (course_id, sec_id, semester, year, building,

room_number, time_slot_id, day, start, end)

Course

Course_ID	Title	Credits
CSC346	OOP	3
CSC346	OOP	3
CSC484	DBMS	3
CSC484	DBMS	3

Section

Course_ID	Sec_id	Semester	Year	Building	Room_num	Capacity	Time_slot_id	Day	Start	End
CSC346	01	Spring	2018	SDEH	109	30	A	Tue	9:00	10:00
CSC346	01	Fall	2017	SDEH	201	45	B	Wed	12:00	13:00
CSC484	01	Fall	2017	SDEH	302	30	A	Tue	9:00	10:00
CSC484	02	Spring	2018	SDEH	118	42	C	Thu	11:00	12:00

Third Normal Form (3NF)

- A relation that is currently in 1NF and 2NF
  - No non-primary-key attribute is transitively dependent on the primary key
  - If a transitive dependency exists, remove the transitively dependent attribute(s) from the relation by placing the attributes(s) in a new relation
    - Along with a copy of the determinant

Transitive Dependency

- If $A \rightarrow B$, $B \rightarrow C$, then $A \rightarrow C$, we say C is **transitively dependent** on A

  inst_dept(ID, name, salary, dept_name, building, budget)

- Functional dependencies:

  ID → *name, salary,* ***dept_name****, building, budget* $A \rightarrow B$

  dept_name → ***building, budget*** $B \rightarrow C$

  $A \rightarrow C$

- *(**building, budget**)* is transitively dependent on ***ID***

Third Normal Form (3NF)

- 2NF → 3NF

course (<u>course_id</u>, title, credits) -- 2NF

section (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room_number, time_slot_id, date, start, end)

course (<u>course_id</u>, title, credits) -- 3NF

- Functional dependencies:

  course_id → title, credits -- meets the requirements for 3NF

Third Normal Form (3NF)

- 2NF → 3NF

section (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room_number, time_slot_id, date, start, end)

- Functional dependencies:

  course_id, sec_id, semester, year → ***building, room_number****, capacity, time_slot_id, day, start, end* — ***A → B***

  building, room_number → ***capacity*** — ***B → C***

  course_id, sec_id, semester, year → *building, room_number, capacity,* ***time_slot_id****, day, start, end* — ***A → B***

  time_slot_id → ***day, start, time*** — ***B → C***

- *(**capacity**)* is transitively dependent on the primary key — ***A → C***
- *(**day, start, end**)* is transitively dependent on the primary key — ***A → C***

Third Normal Form (3NF)

- Relation schemas for the database system: -- in 3NF

course (course_id, title, credits)

section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

building (building, room_number, capacity)

time_slot (time_slot_id, day, start, end)

Course

Course_ID	Title	Credits
CSC346	OOP	3
CSC346	OOP	3
CSC484	DBMS	3
CSC484	DBMS	3

Section

Course_ID	Sec_id	Semester	Year	Building	RoomNum	Time_slot_id
CSC346	01	Spring	2018	SDEH	109	A
CSC346	01	Fall	2017	SDEH	201	B
CSC484	01	Fall	2017	SDEH	302	A
CSC484	02	Spring	2018	SDEH	118	C

Building

Building	RoomNum	Capacity
SDEH	109	30
SDEH	201	45
SDEH	302	30
SDEH	118	42

Time_slot

Time_slot_it	Date	Start	End
A	Tue	9:00	10:00
B	Wed	12:00	13:00
A	Tue	9:00	10:00
C	Thu	11:00	12:00

General Definition of 2NF and 3NF

- Second Normal Form (2NF)
  - A relation that is currently in 1NF
  - Every non-candidate-key attribute is fully functionally dependent on any candidate key
- Third Normal From (3NF)
  - A relation that is currently in 1NF and 2NF
  - No non-candidate-key attribute is transitively dependent on any candidate key

Remember

- 1NF – split composite attributes into distinct attributes
  - *address → street, city, state, zip*
  - *name → first_name, middle_initial, last_name*
  - Split combined entities into distinct entities

- 3NF – separate functional dependencies
  - If an attribute is functionally dependent on something other than the primary key, then break it off and form a new relation
    - Basically, separate entities
      - If you think there might be two entities hiding in 1 relation, then separate them!

Acknowledgements