This week because of U-days
DV Thu 11:00-14:00
1110-216 Undervisningslokale

Normalization

Databases

Ira Assent

ira@cs.au.dk

Data-Intensive Systems group, Department of Computer Science, Aarhus University, DK

Intended learning outcomes

- Be able to
  - decompose tables
  - analyse decompositions of tables
  - Determine 1st, 2nd and 3rd normal forms

ira@cs.au.dk

Recap: Anomalies and FDs

- Storing natural joins of base relations leads to **update anomalies** (data integrity issues) and redundancy

Arrangements

meetid	topic	date	userid	name	group	office
34716	dDB	2014-08-28	ira	Ira Assent	vip	Ny-357
34717	dDB	2013-08-22	ira	Ira Assent	vip	Ny-357
42835	TA meeting	2014-08-18	aas	Annika Schmidt	phd	NULL

- Analyse anomalies with functional dependencies (FDs)
- FD is of the form $a_1,\ldots,a_n \rightarrow b$ for some attributes a_i and b
  - values of $a_1,\ldots,a_n$ determine the value of b in any row
  - Generalizes notion of *key*
- (Super) key is set of attributes $a_1,\ldots,a_n$
  - for any attribute b and any row, values of $a_1,\ldots,a_n$ determine value of b
  - A primary key or candidate key is always minimal

ira@cs.au.dk

Which FDs hold? And what would be a key?

Title	year	length	genre	studio	star
Star Wars	1977	121	SciFi	Fox	Carrie Fisher
Star Wars	1977	121	SciFi	Fox	Mark Hamill
Star Wars	1980	124	SciFi	Fox	Mark Hamill
Avatar	2009	162	SciFi	Fox	Zoe Saldana
Avatar	2022	192	SciFi	Fox	Zoe Saldana
Avatar	2022	192	SciFi	Fox	Sigourney Weaver

A. Title year $\longrightarrow$ star

B. Title year $\longrightarrow$ length

C. Title star $\longrightarrow$ year

D. Title year star $\longrightarrow$ studio

ira@cs.au.dk

Determining keys using FDs

- We can use the attribute closure to determine keys
  - If the attribute closure X^+ contains all attributes of the relation, the attributes X are a superkey
    - A primary key / candidate key has to be minimal
- FD $a_1,\ldots,a_n \rightarrow b$ is **full functional dependency** if $a_1,\ldots,a_{i-k},a_{i+k},\ldots a_n \rightarrow b$ is not a FD for any k
  - Minimal in the sense that there are no "unnecessary" attributes on the left hand-side
  - If a table has a FD, then it also has a full FD
  - Otherwise, **partial dependency**

- So, a primary key / candidate key is the left-hand side of full FD where the closure of the left-hand side is set of all attributes of the relation
- Full functional dependency also called **nonreducible**

Remember the plan

- We want to find all FDs
  - Indicative of potential anomalies
- Use them to decompose problematic tables
  - Understand when and how to do that
  - Should retain the same information when used in queries

ira@cs.au.dk

The broken table revisited

- Arrangements broken
  - redundant, anomalies can occur
  - Some examples of dependencies
    - FD: userid → userid, name, group, office
    - FD: meetid → meetid, topic, date, userid, name, group office

- We want to decompose the table

Arrangements

meetid	topic	date	userid	name	group	office
34716	dDB	2014-08-28	ira	Ira Assent	vip	Ny-357
34717	dDB	2013-08-22	ira	Ira Assent	vip	Ny-357
42835	TA meeting	2014-08-18	aas	Annika Schmidt	phd	NULL

Trying to fix the broken table

- Decomposition

meetid	topic	date	userid	group
34716	dDB	2014-08-28	ira	vip
34717	dDB	2013-08-22	ira	vip
42835	Logic	2014-08-18	fra	vip

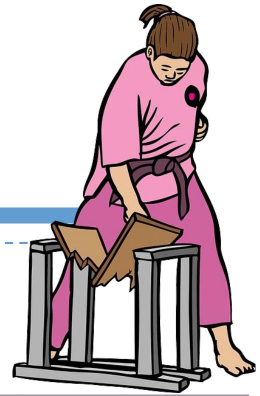
name	group	office
Ira Assent	vip	Ny-357
Ira Assent	vip	Ny-357
Frank Roehr	vip	Ho-010

- But recall, e.g. NATURAL JOIN generates spurious tuples
- We have lost some information
  - The second table with name, group, office lacks a key
  - A proper decomposition should make use of functional dependencies
    - FD: userid $\rightarrow$ userid, name, group, office

meetid	topic	date	userid	name	group	office
42835	Logic	2014-08-18	fra	Ira Assent	vip	Ny-357

ira@cs.au.dk

Nonadditive Join Decomposition



- Relation $R(a_1,\ldots,a_n,b_1,\ldots,b_k,c_1,\ldots,c_m)$ can be **decomposed** into tables R_1 $(a_1,\ldots,a_n,b_1,\ldots,b_k)$, R_2 $(a_1,\ldots,a_n,c_1,\ldots,c_m)$
  $R_1 = \pi_{a_1,\ldots,a_n,b_1,\ldots,b_k}(R)$, $R_2 = \pi_{a_1,\ldots,a_n,c_1,\ldots,c_m}(R)$
  where Greek letter π denotes a projection to some attributes

a	b	c
1	33	117
2	42	335

a	b
1	33
2	42

a	c
1	117
2	335

- If $a_1,\ldots,a_n$ is a superkey for R_1 or for R_2
  then this is a **nonadditive join** decomposition
  - "connects" the two tables
  - Avoids spurious tuples
- E.g. decompose Student (sid, cid, add) with
  student id, course id, club id into tables
  Attends (sid, cid) and Lives (sid, add)
  If sid is a superkey for Lives, nonadditive join decomposition

sid	cid	add
1	33	117
2	42	335

ira@cs.au.dk

Nonadditive Join Decomposition for the broken table

- Meetings, Owners form a nonadditive join decomposition of the broken Arrangements table since userid is a superkey for Owners

Meetings

meetid	date	topic	userid
34716	2014-08-28	dDB	ira
34717	2013-08-22	dDB	ira
42835	2014-08-18	TA meeting	aas

Owners

userid	name	group	office
ira	Ira Assent	vip	Ny-357
aas	Annika Schmidt	phd	NULL

Arrangements

meetid	topic	date	userid	name	group	office
34716	dDB	2014-08-28	ira	Ira Assent	vip	Ny-357
34717	dDB	2013-08-22	ira	Ira Assent	vip	Ny-357
42835	TA meeting	2014-08-18	aas	Annika Schmidt	phd	NULL

Lost in decomposition

- Example
  - R = (location, city, artist)
  - FDs: location → city; artist, city → location
  - Any decomposition of R will fail to preserve artist, city $\rightarrow$ location
  - If decomposed, need to maintain dependency manually, which is difficult and error-prone

location	city	artist
VoxHall	Aarhus	Illdisposed
Fængslet	Horsens	Sting

ira@cs.au.dk

Dependency-preserving decomposition

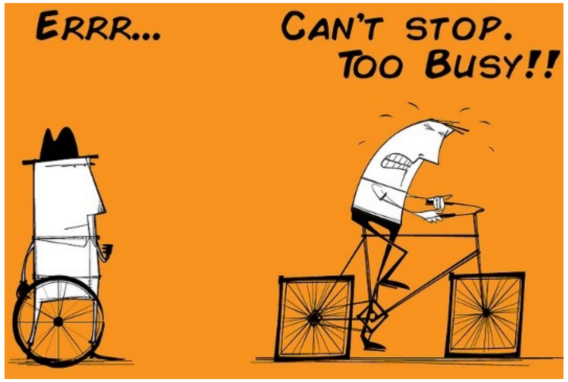
- Given relation $R(a_1,\ldots,a_n,b_1,\ldots,b_k,c_1,\ldots,c_m)$, decomposition into R_1 $(a_1,\ldots,a_n,b_1,\ldots,b_k)$, R_2 $(a_1,\ldots,a_n,c_1,\ldots,c_m)$
  - Let F_i be the set of dependencies in F^+ that include only attributes in R_i
    - i.e., the projection to R_i, written $F_i = \pi(R_i(F^+))$
  - If $(F1 \cup F2)^+ = F^+$

    then this is a **dependency-preserving** decomposition
    - i.e., each functional dependency in F either appears in in one of the relation schemas R_i or can be inferred from the dependencies that appear in some R_i
    - Otherwise, issue with referential integrity across decomposed tables

ira@cs.au.dk

Example

- Table Supervision(Teacher, Subject, Student)
  - FDs: student, subject $\rightarrow$ teacher, teacher $\rightarrow$ subject
- Consider the following decomposition:
  - R_1(Teacher, Subject), R_2(Student, Teacher)
    - Check each (original) functional dependency:
      - Teacher $\rightarrow$ Subject preserved in R_1
      - Student, Subject $\rightarrow$ Teacher lost
        - Not in any of the tables, cannot be inferred from the ones in the tables (here: only the first FD)
- Decomposition not dependency-preserving
  - If such decomposition is used, a lot of work necessary to collect data and check dependency on result (e.g. in program code) → NOT practical



ira@cs.au.dk

Decomposition

- $R = (A, B, C)$ $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B), R_2 = (A, C)$

1. Nonadditive-join decomposition: yes, dependency preserving: no
2. Nonadditive-join decomposition: yes, dependency preserving: yes
3. Nonadditive-join decomposition: no, dependency preserving: no
4. Nonadditive-join decomposition: no, dependency preserving: yes

ira@cs.au.dk

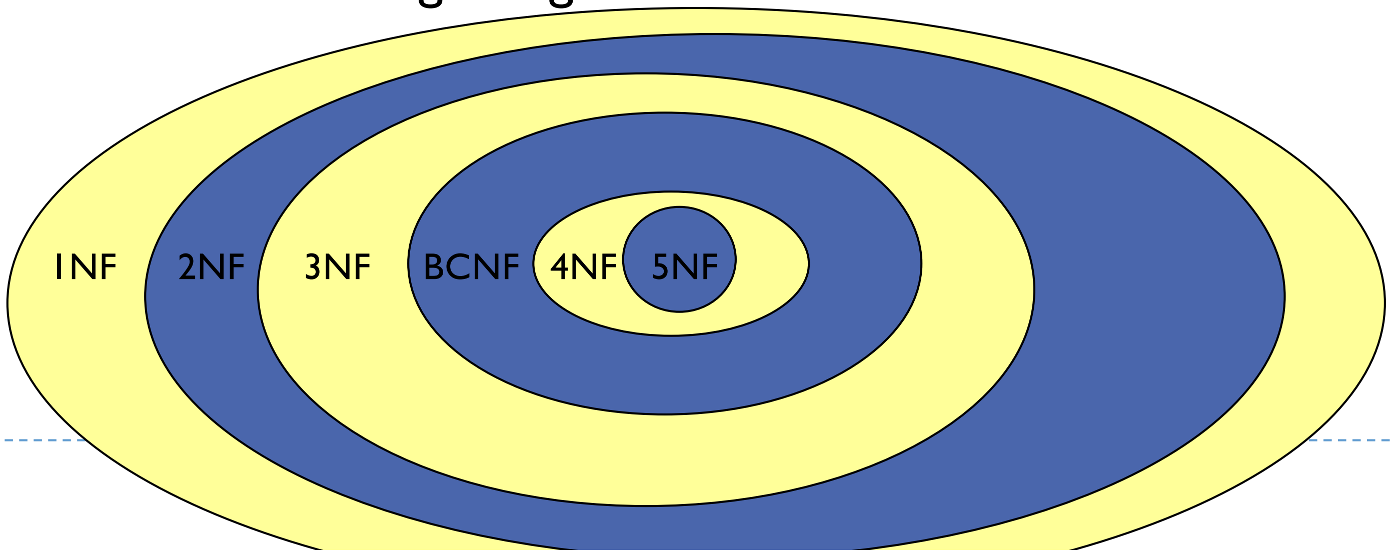
And still remember the plan

- We want to find all FDs
  - Indicative of potential anomalies
- Use them to decompose problematic tables
  - Understand when and how to do that
  - Should retain the same information when used in queries

ira@cs.au.dk

Normalization of Relations

- Takes a relation schema through a series of tests
  - Certify whether it satisfies a certain normal form
  - Proceeds in a top-down fashion
- The **normal form** of a relation is the highest normal form condition that it meets
- Normal Forms organized hierarchically from 1st to 4th plus BCNF (more exist, but rarely used)
  - We start from the beginning



ira@cs.au.dk

First Normal Form (1NF)

- Now part of the formal definition of a relation in the basic (flat) relational model
  - So, you should not encounter any non-1NF in the wild!
- Only attribute values permitted are single **atomic** (or **indivisible**) values
  - I.e., no relation within relation (**nested relation**)
  - And no relations as attribute values within tuples
  - E.g. Student (id, name, courses) with row (1, Ann, {DB, Prog})

- Techniques to achieve first normal form
  - Remove attribute and place in separate relation
    - E.g. Student (id, name), Attends(id, cid)
  - Expand the key
    - E.g. Student (id, cid, name, cname)
  - Use several atomic attributes if maximum number is known
    - E.g. Student (id, name, study_address, home_address)

ira@cs.au.dk

Only attribute values permitted are single **atomic** (or **indivisible**) values 1NF

Textbook example First Normal Form

a) Not in 1NF

b) In Dlocations, we have several values

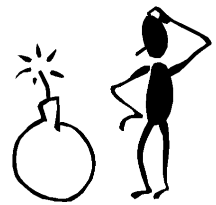
c) To achieve 1NF, separate values in Dlocations

- Single value per tuple
- introduces redundancy across tuples

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations



(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

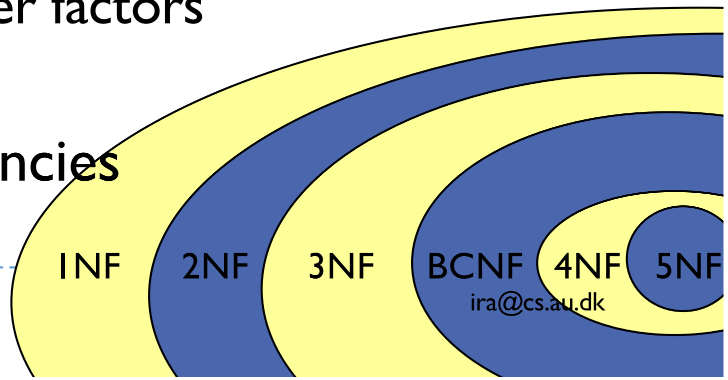
Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

ira@cs.au.dk

Higher normal forms

- 2nd, 3rd,… normal forms all target dependencies that may cause anomalies
  - form of decomposition to remove such dependencies
  - as we move up the hierarchy, we remove more such issues
- Properties that the relational schemas should have:
  - **Nonadditive join property**
    - Extremely critical
    - Otherwise, spurious tuples possible
      - Means incorrect data
        - Besides the point of having a database!
  - **Dependency preservation property**
    - Desirable but sometimes sacrificed for other factors

- After 1NF, now 2NF targets partial dependencies



ira@cs.au.dk

Second Normal Form

- **Prime attribute** is an attribute that is part of a candidate key
  - Otherwise, **non-prime**
  - The second normal form ensures that attributes are fully dependent on the key, unless they are part of a candidate key themselves

- Relation is in **2NF** if every non-prime attribute is fully functional dependent on candidate key
  - So, "other" attributes that are not part of candidate keys are right side of minimal FD with candidate key on left side
- In Courses, Location depends on University, not on Course and University:
  - University → Location
  - Location not part of a key, not in 2NF
  - Issue: whenever you mention university, you need to list the location for each of the courses (mixed semantics) → redundancy, risk of update anomalies!

Courses

<u>Course</u>	<u>University</u>	ECTS	Location
DB	AU	5	Aarhus
Prog	AU	10	Aarhus

ira@cs.au.dk

Obtaining Second Normal Form

- In Courses, Location depends on University, not on Course and University:
  - University → Location
- Decompose to obtain 2NF
  - take this "violating" FD
  - create a new relation UniLoc (University, Location)
  - remove the dependent attribute Location from Courses to get Courses2(Course, University, ECTS)

Courses

Course	University	ECTS	Location
DB	AU	5	Aarhus
Prog	AU	10	Aarhus

UniLoc

University	Location
AU	Aarhus

Every non-prime attribute is fully functional dependent on candidate key 2NF

Courses2

Course	University	ECTS
DB	AU	5
Prog	AU	10

ira@cs.au.dk

Second Normal Form?

Every non-prime attribute is fully functional dependent on candidate key 2NF

EmployeeSkills

Name	Skill	Task	Work Location
Chris	Typing	Letters	Åbogade
Chris	Shorthand	Dictation	Åbogade
Eve	Typing	Data Entry	Finlandsgade
Tom	Alchemy	Gold Creation	Finlandsgade
Tom	Juggling	Show Performance	Finlandsgade

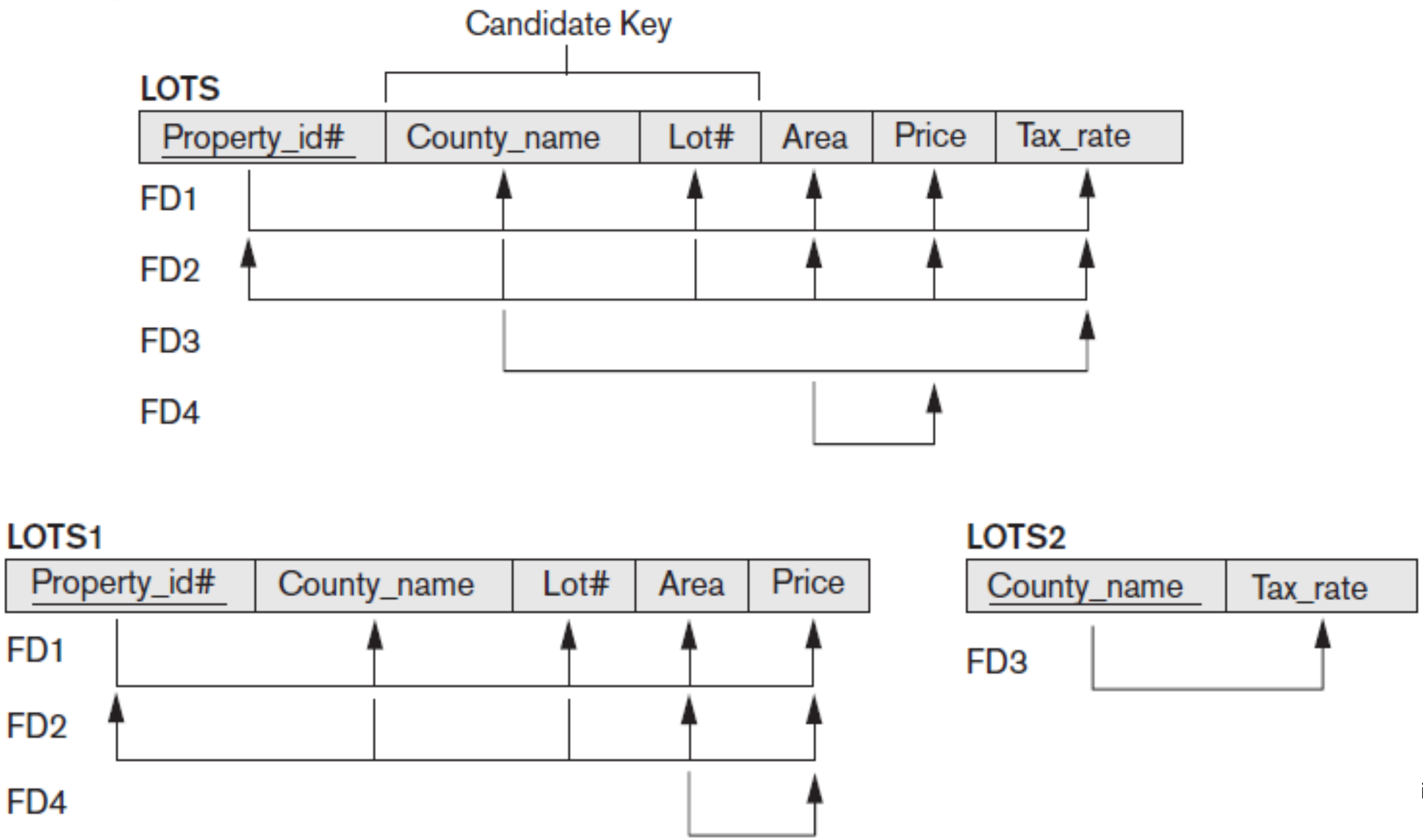
1. Yes.
2. No, there is partial dependency on a candidate key.
3. No, there is a full dependency on a candidate key.
4. I don’t know.

ira@cs.au.dk

Textbook example 2NF

Every non-prime attribute is fully functional dependent on candidate key 2NF

- Miniworld: prices of parcels of land for sale
- LOTS relation not in 2NF because Tax_rate partially dependent on candidate key {County_name, Lot#} (FD3)
- Decompose into LOTS1 and LOTS2



ira@cs.au.dk

Third Normal Form

- Relation is in **3NF** if for every nontrivial FD X → A, X superkey or A prime attribute **3NF**
  - So, no non-key attributes determine other non-key attributes and no proper subset of key determines non-key attributes

Student

id	name	zip	city
1	Ann	8000	Aarhus
2	Tom	8000	Aarhus

- Removes further redundancy and potential anomalies
  - get rid of transitive dependencies: A→B, B→C, thus A→C
  - Student(id, name, zip, city), FD zip→city
    - zip not a superkey for Student, city not prime (not part of candidate key for Student)
  - Decompose into Student2(id, name, zip), Address (zip, city)

Student2

id	name	zip
1	Ann	8000
2	Tom	8000

Address

zip	city
8000	Aarhus

ira@cs.au.dk

Third Normal Form example

- Concerts:

<u>**location**</u>	<u>**city**</u>	**artist**
VoxHall	Aarhus	Illdisposed
Jakobshof	Aachen	Ina Deter

- FDs: location → city; artist city → location
  - Candidate keys: artist, city; artist, location

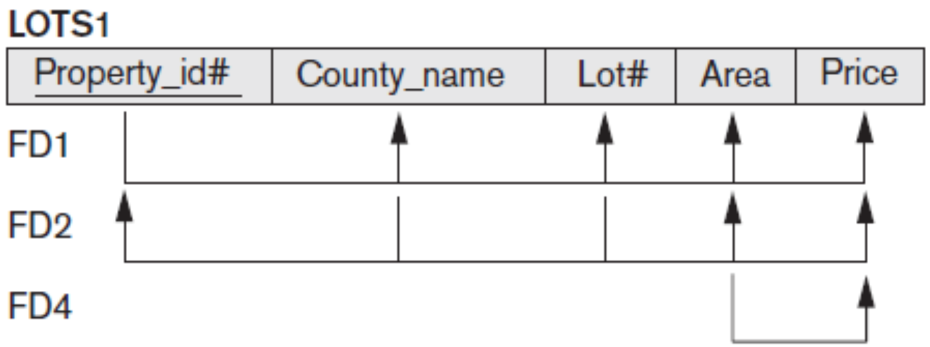
For every nontrivial FD X → A, X superkey or A prime attribute 3NF

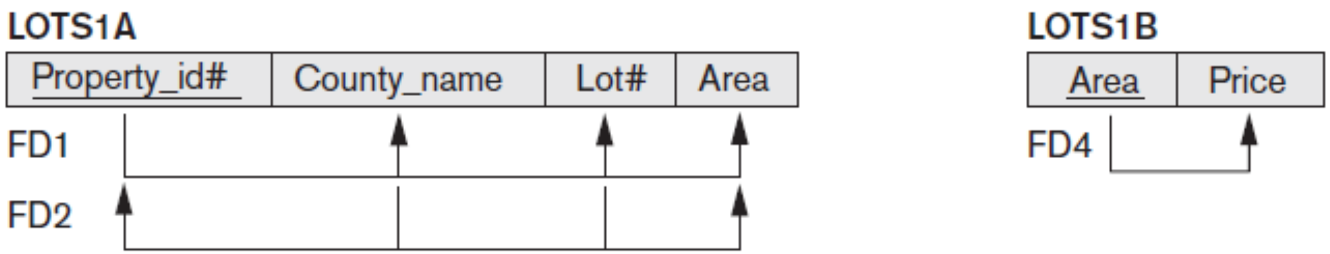
  - R is in 3NF
    - location→ city: city is contained in a candidate key (city is prime)
    - artist, city → location: artist, city is a candidate key (superkey)

ira@cs.au.dk

Textbook example 3NF

- Miniworld: prices of parcels of land for sale
- LOTS1 not in 3NF because of FD4: Area not a superkey, Price not prime
- Decompose into LOTS1 and LOTS2





ira@cs.au.dk

Third Normal Form?

For every nontrivial FD X → A, X superkey or A prime attribute 3NF

Winners

<u>Tournament</u>	<u>Year</u>	Winner	Date of Birth
Australian Open	2018	Roger Federer	8 August 1981
Australian Open	2022	Rafael Nadal	3 June 1986
Wimbledon	2017	Roger Federer	8 August 1981
French Open	2017	Rafael Nadal	3 June 1986
Australian Open	2020	Novak Djokovic	22 May 1987

1. Yes.
2. No, a non-superkey determines another attribute.
3. No, a FD determines a non-prime attribute.
4. I don't know.

ira@cs.au.dk

Normal Forms so far

Table 15.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

ira@cs.au.dk

Second Normal Form (2NF)?

Every non-prime attribute is fully functional dependent on candidate key 2NF

- Consider the relational schema (A,B,C,D) with the FDs
  - $A,B \rightarrow C,D$ and
  - $A \rightarrow D$

1. Yes.
2. No, there is partial dependency on a candidate key.
3. No, there is a full dependency on a candidate key.
4. I don't know.

ira@cs.au.dk

Third Normal Form (3NF)?

For every nontrivial FD X → A, X superkey or A prime attribute 3NF

- Consider the relational schema (A,B,C,D) with the FDs:
  - A,B → C,D and
  - C → D

1. Yes.
2. No, a non-superkey determines another attribute.
3. No, a FD determines a non-prime attribute.
4. I don't know.

ira@cs.au.dk

Summary

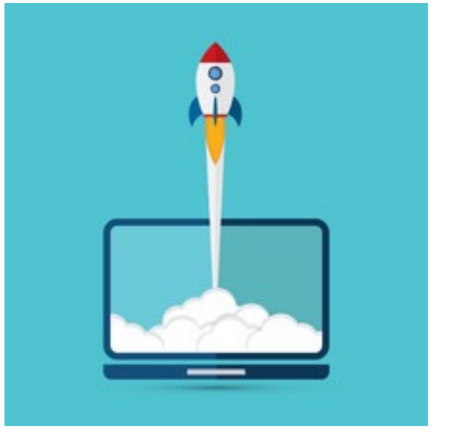
- Intended learning outcomes
- Be able to
  - decompose tables
  - analyse decompositions of tables
  - Determine 1st, 2nd and 3rd normal forms

ira@cs.au.dk

Where to go from here?

- We know how to test for issues and dependencies that violate 1NF, 2NF, 3NF and that decomposition can avoid these
  - We know that nonadditive join decomposition and dependency preservation are important

- Next, we'll see more issues and learn about multi-valued dependencies and 4NF, as well as BCNF normal form



ira@cs.au.dk

What was this all about?

Guidelines for your own review of today's session

- A full functional dependency…
  - Else, we call it…
- We use normal forms to define…
  - Normal forms differ with respect to the …
- A relation is in 1NF if…
  - If it is not in 1NF, we may have the following issue…
- A relation is in 2NF if…
  - If it is not in 2NF, we may have the following issue…
- A relation is in 3NF if…
  - If it is not in 3NF, we may have the following issue…

ira@cs.au.dk