CSC7072: Databases, fall 2015

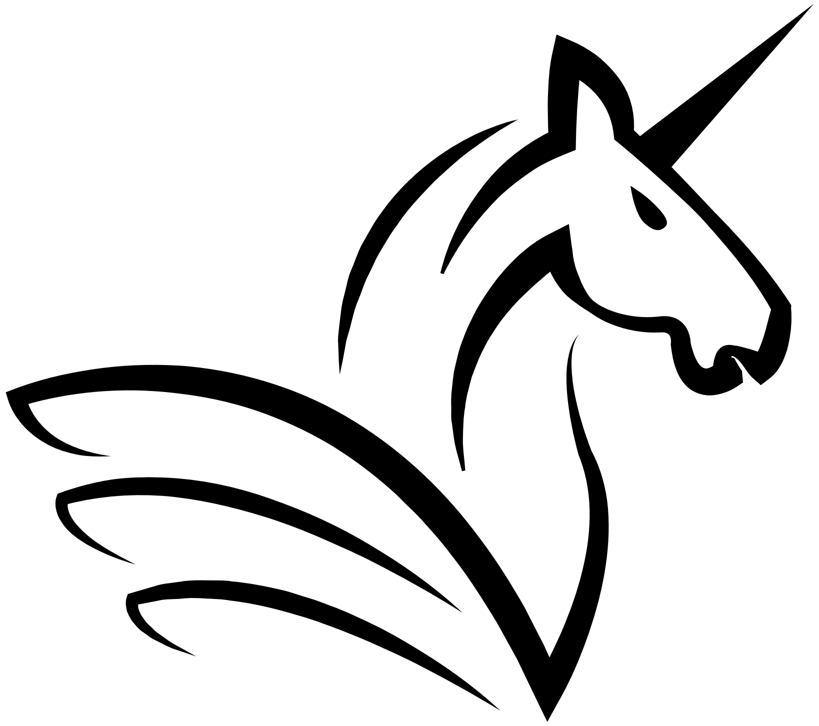
Dr. Kim Bauters

normalisation

Normalisation

adding tables using SQL

we learned how to convert an ER model to a relation model



why do you believe that this is a good approach?

Normalisation

normalisation levels

what is normalisation?

it is a formal process to identify a good database,
and to improve a database design if any issues are found

1st normal form (1NF)
2nd normal form (2NF)
3rd normal form (3NF)
3.5th normal form (BCNF)
4th normal form (4NF)
5th normal form (5NF)

gets harder: more conditions to satisfy

gets better: less anomalies with update/insert/delete

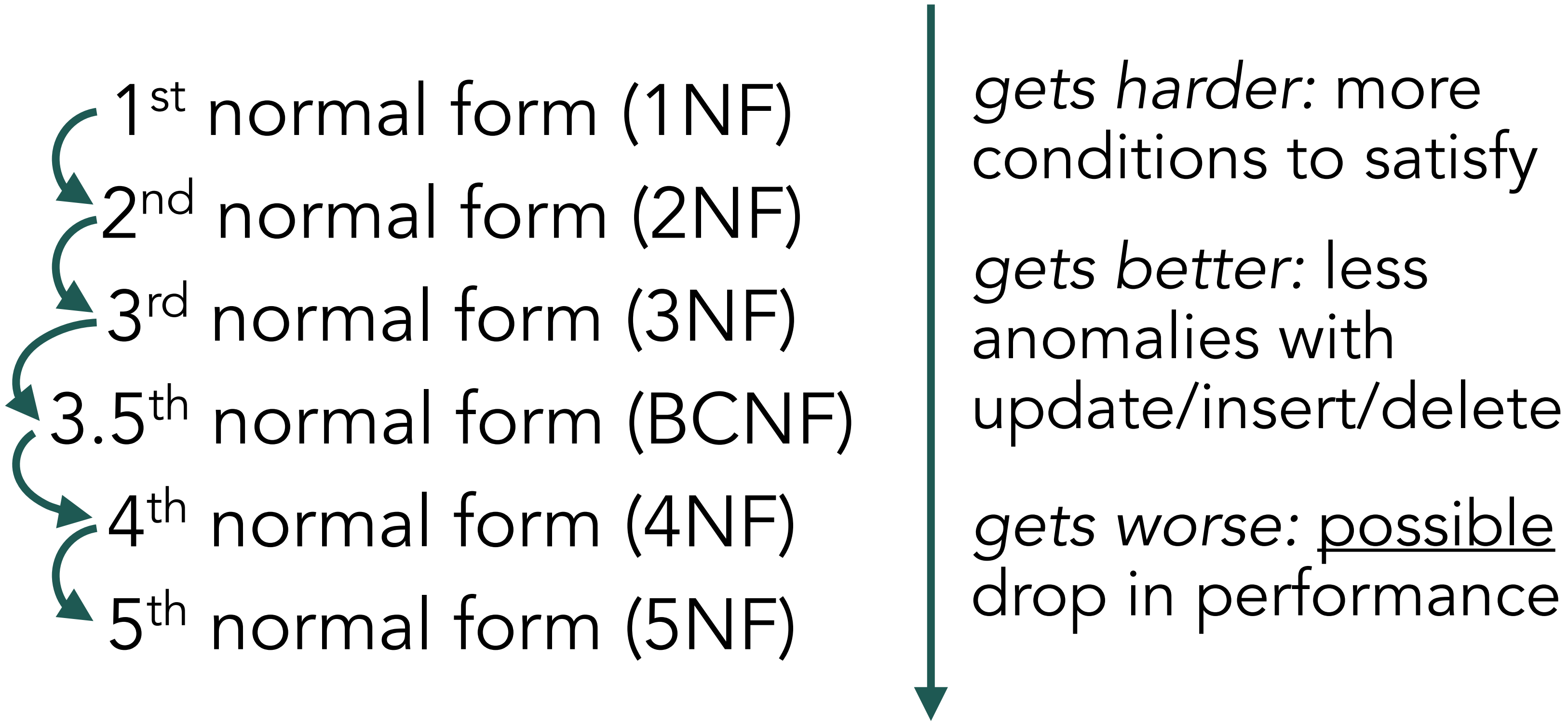
gets worse: possible drop in performance

Normalisation

normalisation levels

what is normalisation?

it is a formal process to identify a good database,
and to improve a database design if any issues are found



1st normal form (1NF)

2nd normal form (2NF)

3rd normal form (3NF)

3.5th normal form (BCNF)

4th normal form (4NF)

5th normal form (5NF)

gets harder: more conditions to satisfy

gets better: less anomalies with update/insert/delete

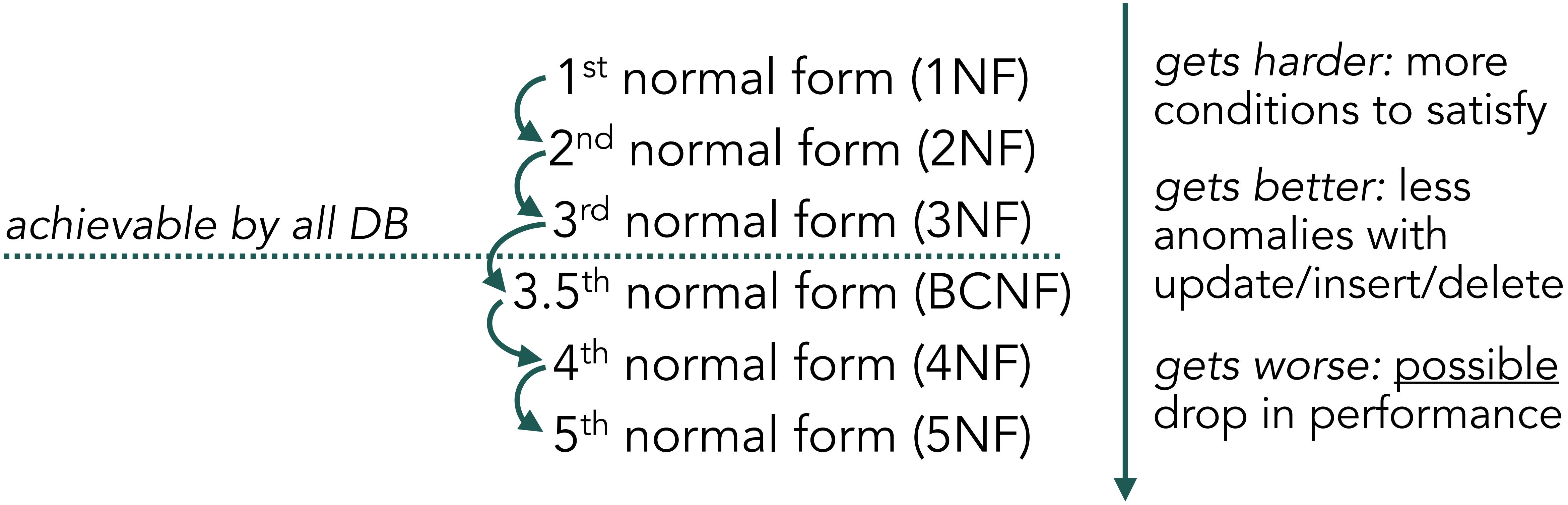
gets worse: possible drop in performance

Normalisation

normalisation levels

what is normalisation?

it is a formal process to identify a good database,
and to improve a database design if any issues are found



1st normal form (1NF)

2nd normal form (2NF)

3rd normal form (3NF)

3.5th normal form (BCNF)

4th normal form (4NF)

5th normal form (5NF)

achievable by all DB

gets harder: more conditions to satisfy

gets better: less anomalies with update/insert/delete

gets worse: possible drop in performance

Normalisation

problems we try to avoid

what do we want to achieve?

- we want to minimise (even eliminate) redundant information
  having redundant information would make updates more difficult as we need to make sure *everything* is changed correctly

- we want to achieve representational power
  we want to be able to put all our required data in the database!

- we want to avoid loss of information
  we can keep on splitting tables into smaller ones, but as some point we won't be able to piece the information back together!

Normalisation

problems we try to avoid

what do we want to achieve?

- we want to minimise (even eliminate) redundant information
  having redundant information would make updates more difficult as we need to make sure *everything* is changed correctly

id	name	dept_name	salary	dept_budget
45039	John	Comp. Sci.	65000	450000
30594	Selma	Comp. Sci.	75000	450000
30492	Paul	Elec. Eng.	90000	720000
20996	Tisan	Elec. Eng.	80000	720000

better not forget to update both!

Normalisation

problems we try to avoid

what do we want to achieve?

- we want to minimise (even eliminate) redundant information
  having redundant information would make updates more difficult as we need to make sure *everything* is changed correctly

id	name	dept_name	salary	dept_budget
45039	John	Comp. Sci.	65000	450000
30594	Selma	Comp. Sci.	75000	450000
30492	Paul	Elec. Eng.	90000	720000
20996	Tisan	Elec. Eng.	80000	940000

better not forget to update both!

and you will …

Normalisation

problems we try to avoid

what do we want to achieve?

John can be reached on 074 1000 1000, or 028 1000 1000

oops …

- we want to achieve representational power
  we want to be able to put all our required data in the database!

id	name	dept_name	salary	tel_no
45039	John	Comp. Sci.	65000	074 1000 1000
30594	Selma	Comp. Sci.	75000	074 2000 2000
30492	Paul	Elec. Eng.	90000	074 3000 3000
20996	Tisan	Elec. Eng.	80000	074 4000 4000

Normalisation

problems we try to avoid

what do we want to achieve?

- we want to avoid loss of information
  we can keep on splitting tables into smaller ones, but as some point we won't be able to piece the information back together!

id	name	dept_name	salary
45039	Kim	Comp. Sci.	40000
30594	Kim	Elec. Eng.	50000

id	name
45039	Kim
30594	Kim

name	dept_name	salary
Kim	Comp. Sci.	40000
Kim	Elec. Eng.	50000

Normalisation

problems we try to avoid

avoiding loss of information

id	name
45039	Kim
30594	Kim

name	dept_name	salary
Kim	Comp. Sci.	40000
Kim	Elec. Eng.	50000

↘ join ↙

id	name	dept_name	salary
45039	Kim	Comp. Sci.	40000
45039	Kim	Elec. Eng.	50000
30594	Kim	Comp. Sci.	40000
30594	Kim	Elec. Eng.	50000

lossy decomposition: no longer gives us original table after join

Normalisation

why they exist

main idea of normalisation:

- verify if a schema is in a *"good"* form
- if not, decompose the relation in multiple relations using a lossless decomposition

some notions we will use:

a *non-key* attribute is an attribute that is not a part of the primary key or any candidate key

a *functional dependency*, written as X → Y … Z, means that the values of Y … Z are determined by the values of X (the *determinant*).

in other words: X would be a candidate key for table(X, Y, Z)
or also: the same student_id will always give us the same row

Normalisation

1NF: atomic values

first normal form (1NF)

definition

A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

a domain is *atomic* if its elements are considered *indivisible* units

examples of non-atomic data:

- a set of telephone numbers, a composite attribute
- CS7052: this is department CS and id 7052 !

Normalisation

1NF: atomic values

first normal form (1NF)

definition
A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

always assumes data to be atomic, but also:

- there are no duplicate rows/columns
- all records have the same number of attributes

(but this really is by definition of using relation schemas)

Normalisation

2NF: non-key attributes depend on the entire PK

second normal form (2NF)

definition

A relation is in second normal form when we do not have a *non-key attribute* that depends on a *strict subset* of the *primary key*.

<u>**part**</u>	<u>**warehouse**</u>	quantity	address
1021	alpha	512	infinity loop
0023	alpha	256	infinity loop
0587	beta	192	redmond
1021	gamma	1024	mountain view

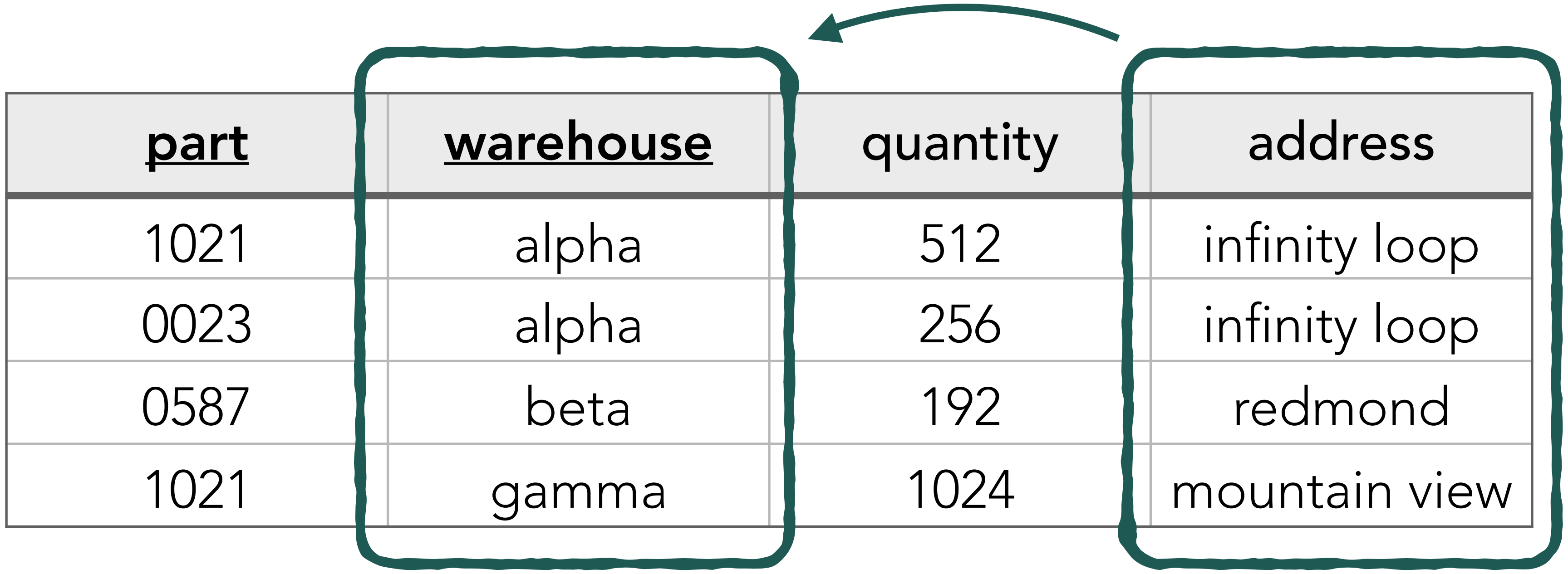
Normalisation

2NF: non-key attributes depend on the entire PK

second normal form (2NF)

definition

A relation is in second normal form when we do not have a *non-key attribute* that depends on a *strict subset* of the *primary key*.



part	**warehouse**	quantity	address
1021	alpha	512	infinity loop
0023	alpha	256	infinity loop
0587	beta	192	redmond
1021	gamma	1024	mountain view

Normalisation

2NF: non-key attributes depend on the entire PK

second normal form (2NF)

definition

A relation is in second normal form when we do not have a *non-key attribute* that depends on a *strict subset* of the *primary key*.

problem:

a functional dependency: warehouse → address

indicates the need to decompose!

solution:

inventory_part(part, warehouse, quantity)

inventory_warehouse(warehouse, location)

Normalisation

3NF: non-key attributes depend *only* on the entire PK

third normal form (3NF)

definition

A relation is in third normal form when we do not have a *non-key attribute* that is a fact about another *non-key attribute*

employee_id	department	location
12345	Comp. Sci.	ECS
10001	Comp. Sci.	ECS
34021	Elec. Eng.	ECIT
60520	Elec. Eng.	ECIT

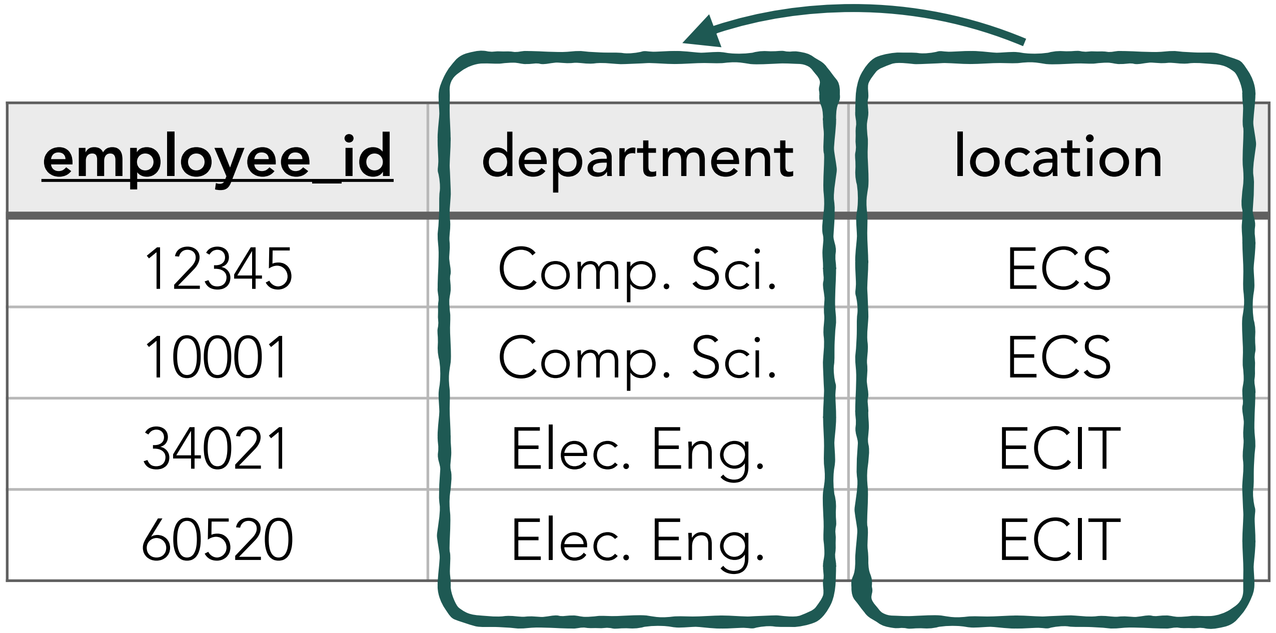
Normalisation

3NF: non-key attributes depend *only* on the entire PK

third normal form (3NF)

definition

A relation is in third normal form when we do not have a *non-key attribute* that is a fact about another *non-key attribute*



<u>employee_id</u>	department	location
12345	Comp. Sci.	ECS
10001	Comp. Sci.	ECS
34021	Elec. Eng.	ECIT
60520	Elec. Eng.	ECIT

department → location

so … decompose!

Normalisation

3NF: non-key attributes depend *only* on the entire PK

third normal form (3NF)

definition

A relation is in third normal form when we do not have a *non-key attribute* that is a fact about another *non-key attribute*

solution:

employee_department(employee_id, department)

department(department, location)

conclusion: a schema is in (2nd and) 3rd normal form when every field is either part of the primary key or provides a single-valued fact *about the whole key and nothing else*

Normalisation

BCNF: all determinants must be candidate keys

Boyce-Codd normal form (BCNF): *overlapping candidate keys*

definition

A relation is in Boyce-Codd normal form when every attribute provides a fact about the whole key, *or*, if and only if every determinant is a candidate key.

example:

supervision(project, branch, manager)

❶ manager → branch	each manager works in one branch
❷ project, branch → manager	each project has several managers and spans several branches; a project has a unique manager for every branch

Normalisation

BCNF: all determinants must be candidate keys

Boyce-Codd normal form (BCNF): *overlapping candidate keys*

definition

A relation is in Boyce-Codd normal form when **every attribute provides a fact about the whole key**, *or*, if and only if every determinant is a candidate key.

example:

supervision(project, branch, manager)

❶ manager → branch	each manager works in one branch
❷ project, branch → manager	each project has several managers and spans several branches; a project has a unique manager for every branch

not in BCNF!

Normalisation

BCNF: all determinants must be candidate keys

Boyce-Codd normal form (BCNF): *overlapping candidate keys*

definition

A relation is in Boyce-Codd normal form when every attribute provides a fact about the whole key, *or*, if and only if **every determinant is a candidate key.**

example:

supervision(project, branch, manager)

❶ manager → branch

❷ project, branch → manager

not in BCNF!

candidate keys:

project, branch

project, manager

Normalisation

BCNF: all determinants must be candidate keys

Boyce-Codd normal form (BCNF): *overlapping candidate keys*

definition

A relation is in Boyce-Codd normal form when every attribute provides a fact about the whole key, *or*, if and only if every determinant is a candidate key.

example:

supervision(project, branch, manager)

❶ manager → branch	each manager works in one branch
❷ project, branch → manager	each project has several managers and spans several branches; a project has a unique manager for every branch

prevents decomposition

Normalisation

strong entity, composite and derived attributes

3NF or Boyce-Codd normal form?

always possible to convert a DB into 3NF where:

- the decomposition is lossless;
- all functional dependencies are preserved; but where
- some anomalies occur …
  e.g. a branch is involved in multiple projects; a manager changes branch …

always possible to convert a DB into BCNF where:

- the decomposition is lossless; but only
- ***some*** functional dependencies are preserved …

Normalisation

strong entity, composite and derived attributes

goal for a relational database design:

- achieve BCNF; with
- lossless decomposition; *and*
- full functional dependency preservation

if that is impossible, then choose:

- stick to 3NF and full functional dependency preservation
  be wary of the anomalies!
- get BCNF but miss out on some functional dependencies
  might not be able to represent all information!

Normalisation

4NF: non-key attributes must depend on each other and the PK

fourth normal form (4NF)

definition

A relation is in fourth normal form when it does not have more than one multivalued dependency.

restaurant	variety	delivery_area
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City

multivalued dependency:

if we choose a restaurant, then we know the variety (which does not depend on the delivery area), and vice versa

Normalisation

4NF: non-key attributes must depend on each other and the PK

fourth normal form (4NF)

definition

A relation is in fourth normal form when it does not have more than one multivalued dependency.

restaurant	**variety**	**delivery_area**
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City

multivalued dependency:

restaurant ↠ variety

restaurant ↠ delivery_area

Normalisation

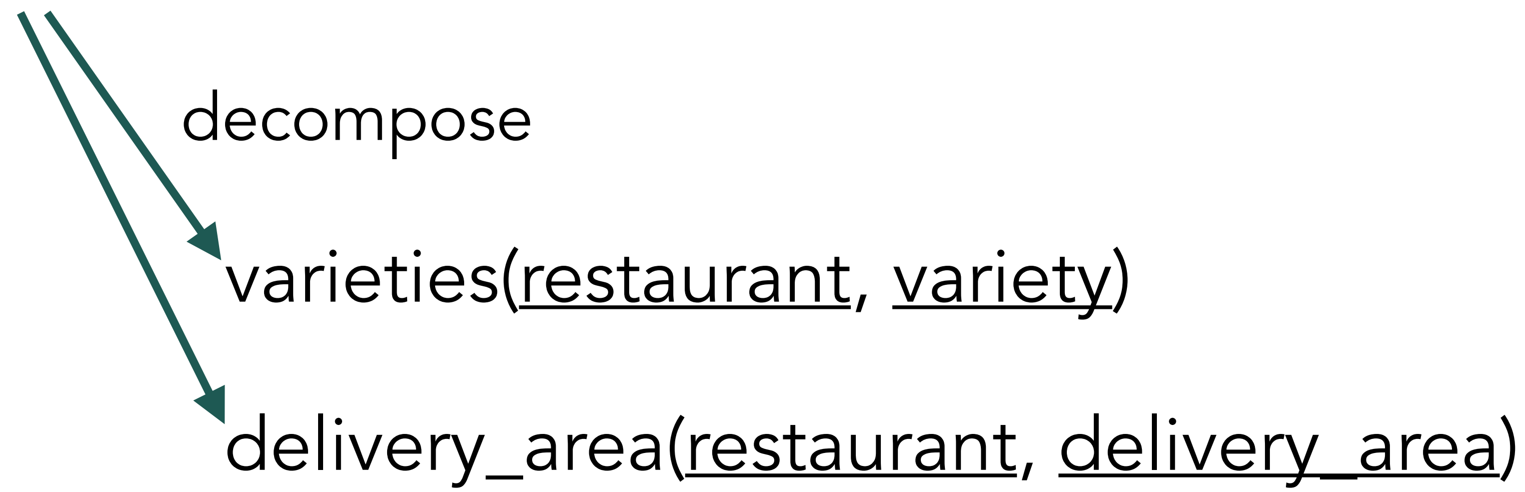
4NF: non-key attributes must depend on each other and the PK

fourth normal form (4NF)

definition

A relation is in fourth normal form when it does not have more than one multivalued dependency.

delivery(restaurant, variety, delivery_area)



decompose

varieties(restaurant, variety)

delivery_area(restaurant, delivery_area)

Normalisation

5NF: the relation consists only of a PK and a non-key attribute

fifth normal form (5NF)

definition

A relation is in fifth normal form if decomposing it in any possible way would not remove any redundancies.

by far, one of the most elusive normal forms to understand

easiest is just to try out all possible decompositions, and make sure none of them are lossy decompositions

it tends to create too many small tables, which is great for updating and absolutely horrible for performance (umpteen joins!)

Normalisation

ER models and normalisation

is normalisation needed for E-R models?

- **when** an E-R model is well-designed (*i.e.* all entities and relations are correctly identified) **then** the tables generated from an E-R model should not need normalisation

- in the real world, the design is imperfect and some functional dependencies will still be left
  e.g. employee(id, name, dept_name, building)
  where dept_name → building

so E-R model is *generation*, normalisation is *verification*

Normalisation

how best to use normalisation

how to design a good database

1. start from a relation model:
   - we can obtain this as the result of an E-R model; or
   - we can have a single relation with all relevant attributes *i.e.* one big table, called the universal relation
2. normalise this relation model to break it into smaller relations
3. stop when the desired normalisation form is reached
   - no excuse to stop before 3NF!
   - always apply 3.5NF if possible
   - 4NF and 5NF provide trade-offs: anomalies/performance

Normalisation

recap

normalisation in a nutshell:

each level depends on previous level(s), *i.e.* 3NF requires 2NF, 1NF

if a schema is not in the required NF; decompose!

all DB can be turned into 3NF; not all can be turned into BCNF

levels are:

- 1NF: atomic, no duplicate rows, equal no. of attributes
- 2NF: do not depend on only a part of the PK
- 3NF: do not depend on another non-key
- BCNF: do not provide a fact about only part of the key
- 4NF: depend on all other non-keys *and* on PK

Normalisation

recap

what level is this schema in?

student_id	name	telno1	telno2
1001	James	028 1234 1021	
1002	Thomas	028 9876 1021	
1003	Laura	028 1021 1234	
1004	Lindsey	028 1021 9876	0478 11 04 123
1005	Francis	028 1201 1234	
1006	Paul	028 9876 1201	

Normalisation

recap

what level is this schema in?

student_id	name	telno1	telno2
1001	James	028 1234 1021	
1002	Thomas	028 9876 1021	
1003	Laura	028 1021 1234	
1004	Lindsey	028 1021 9876	0478 11 04 123
1005	Francis	028 1201 1234	
1006	Paul	028 9876 1201	

0NF, as not the same number of attributes for all rows

Normalisation

recap

what level is this schema in?

student_id	name	telno
1001	James	028 1234 1021
1002	Thomas	028 9876 1021
1003	Laura	028 1021 1234
1004	Lindsey	028 1021 9876 0478 11 04 123
1005	Francis	028 1201 1234
1006	Paul	028 9876 1201

Normalisation

recap

what level is this schema in?

student_id	name	telno
1001	James	028 1234 1021
1002	Thomas	028 9876 1021
1003	Laura	028 1021 1234
1004	Lindsey	028 1021 9876 0478 11 04 123
1005	Francis	028 1201 1234
1006	Paul	028 9876 1201

0NF, as some attributes are not atomic (telno)

Normalisation

recap

what level is this schema in?

student_id	name	school	location
1001	James	EEECS	Belfast
1002	Thomas	EEECS	Belfast
1003	Laura	EEECS	Belfast
1004	Lindsey	EEECS	Belfast
1005	Francis	EEECS	Belfast
1006	Paul	POLSOC	Lisburn

Normalisation

recap

what level is this schema in?

student_id	name	school	location
1001	James	EEECS	Belfast
1002	Thomas	EEECS	Belfast
1003	Laura	EEECS	Belfast
1004	Lindsey	EEECS	Belfast
1005	Francis	EEECS	Belfast
1006	Paul	POLSOC	Lisburn

assume: school → location

Normalisation

recap

what level is this schema in?

<u>student_id</u>	name	school	location
1001	James	EEECS	Belfast
1002	Thomas	EEECS	Belfast
1003	Laura	EEECS	Belfast
1004	Lindsey	EEECS	Belfast
1005	Francis	EEECS	Belfast
1006	Paul	POLSOC	Lisburn

2NF, as *location* only depends on the non-key *school*

Normalisation

recap

what level is this schema in?

school	advisor	topic	project_name
EEECS	Kim	CS	planning under uncertainty
EEECS	Kim	CS	automated digitisation
EEECS	Weiru	CS	handling inconsistency
EEECS	Anna	DM	mining Twitter
POLSOC	Janosch	POL	conflict and resolution
POLSOC	Paul	POL	children in war

Normalisation

recap

what level is this schema in?

school	**advisor**	**topic**	**project_name**
EEECS	Kim	CS	planning under uncertainty
EEECS	Kim	CS	automated digitisation
EEECS	Weiru	CS	handling inconsistency
EEECS	Anna	DM	mining Twitter
POLSOC	Janosch	POL	conflict and resolution
POLSOC	Paul	POL	children in war

assume: advisor → topic

Normalisation

recap

what level is this schema in?

school	**advisor**	**topic**	**project_name**
EEECS	Kim	CS	planning under uncertainty
EEECS	Kim	CS	automated digitisation
EEECS	Weiru	CS	handling inconsistency
EEECS	Anna	DM	mining Twitter
POLSOC	Janosch	POL	conflict and resolution
POLSOC	Paul	POL	children in war

1NF, as *topic* only depends on part of the key, namely on *advisor*

Normalisation

recap

what level is this schema in?

student_no	advisor	adv-room	class1	class2	class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	201-01	211-02	214-01

Normalisation

recap

what level is this schema in?

student_no	advisor	adv-room	class1	class2	class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	201-01	211-02	214-01

0NF, as we have repeating groups (*i.e.* not atomic)
can be turned into 1NF by adding row for each class for student

Normalisation

recap

what level is this schema in?

student_no	advisor	adv-room	class
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

Normalisation

recap

what level is this schema in?

student_no	**advisor**	**adv-room**	**class**
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

1NF, with the need for a composite key – class only depends on stu_no
can be turned into 2NF by splitting the table

Normalisation

recap

what level is this schema in?

student_no	advisor	adv-room
1022	Jones	412
4123	Smith	216

student_no	class
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

Normalisation

recap

what level is this schema in?

student_no	advisor	adv-room
1022	Jones	412
4123	Smith	216

student_no	class
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

2NF, as *adv-room* only depends on the non-key *advisor*
can be turned into 3NF by splitting the left table

Normalisation

recap

what level is this schema in?

student_no	advisor
1022	Jones
4123	Smith

advisor	adv-room
Jones	412
Smith	216

student_no	class
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

Normalisation

recap

what to expect for exam:

1. definitions of the different normal forms,
   and examples that *do not* satisfy that normal form

2. general knowledge on normal forms
   e.g. what is normalisation? when do we use normalisation?
   how do we transform a database to a higher normal form? …

3. identify normal form of a schema,
   and convert a schema to a given normal form