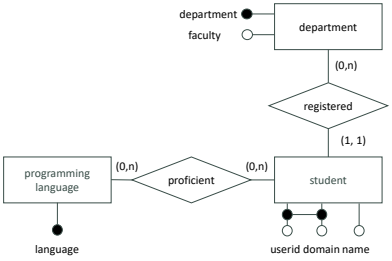
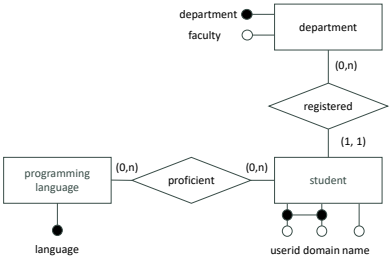
Anomalies and Boyce-Codd Normal Form

Stéphane Bressan

Sentosa University of Technology (SUT) records the programming language skills of the students of its different faculties and departments.





We have four tables: `programming_language(language)`, `student(userid, domain, name, department)`, `department(department, faculty)`, and `proficiency(userid, domain, language)`.

We store everything in one table (`proficiency`.) What can go wrong?

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

Each faculty is organised in several departments. A department belongs to one faculty only and there is no two different departments with the same name in Sentosa University of Technology.

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

$$\{department\} \rightarrow \{faculty\}$$

Students are identified by their email. The email of a student is composed of her userid and domain.

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

$$\{userid, domain\} \rightarrow \{name\}$$

We record the primary department of the student.

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

$$\{userid, domain\} \rightarrow \{department\}$$

For each student, the database records (only once) each programming language in which the student is proficient.

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

$$\{userid, domain, language\} \rightarrow \{name, userid, domain, department, faculty, language\}$$

How can we avoid repeating the same rows?

proficiency					
name	userid	domain	department	faculty	language
...					
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
...					

```
1 CREATE TABLE proficiency (
2 ...
3 PRIMARY KEY (name, userid, domain, department, faculty, language));
```

```
1 CREATE TABLE proficiency (
2 ...
3 PRIMARY KEY (userid, domain, language));
```

Let us call the table R (`proficiency`) and the columns A (`name`) , B (`userid`), C (`domain`), D (`department`), E (`faculty`), and F (`language`), respectively.

We have the following set of functional dependencies:
$\Sigma = \{\{B, C\} \rightarrow \{A\}, \{B, C\} \rightarrow \{D\}, \{D\} \rightarrow \{E\}, \{B, C, F\} \rightarrow \{A, B, C, D, E, F\}\}$

Calculate the attribute closures to find all the candidate keys.

$\{A\}^+ = \{A\}$.

...

$\{B, C\}^+ = \{A, B, C, D, E\}$.

...

$\{B, C, F\}^+ = \{A, B, C, D, E, F\}$.

...

The candidate key (there is only one) of R with Σ is $\{B, C, F\}$ (`userid`, `domain`, `language`).

We store everything in one table. What can go wrong?

If we do not enforce the functional dependencies, we can experience anomalies:

- redundant storage,
- update anomalies,
- deletion anomalies, and
- insertion anomalies.

Let us illustrate the four different anomalies with this design and the functional dependency:

$$\{department\} \rightarrow \{faculty\}$$

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

Redundant Storage

The faculty of a department is repeated for every student of the department and every time the student is proficient in a programming language.

Is it really an issue?

IBM 726 (rental USD$850 per month in 1952)

Seagate ST12000NE0008 IronWolf Pro Internal Hard Drive, 12TB, 3.5"

S$595.23

Get it **Tue, 29 Mar - Fri, 1 Apr**
FREE Delivery

$$\{department\} \rightarrow \{faculty\}$$

If we want to save space and some query time (or not), we can store the departments and their faculties in a separate table.

```
1 CREATE TABLE department (
2 department VARCHAR(64),
3 faculty VARCHAR(128));
```

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

Even then we still need to indicate the departments in the main table ...

proficiency				
name	userid	domain	department	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	R

We join the two tables to recover the original table.

```
1 SELECT p.name, p.userid, d.domain, p.department, p.faculty, p.language
2 FROM proficiency p, department d
3 WHERE p.department = d.faculty;
```

```
1 SELECT p.name, p.userid, d.domain, p.department, p.faculty, p.language
2 FROM proficiency p INNER JOIN department d ON p.department = d.faculty;
```

```
1 SELECT *
2 FROM proficiency p NATURAL JOIN department d
```

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	informatics	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

```
1 UPDATE proficiency SET faculty='informatics' WHERE userid='stan' AND domain='comp.sut.edu' ;
```

Update Anomaly

Two rows of the table have the same value for the column `department` but different values for the column `faculty`. This violates the functional dependency.

If we store the department and the faculty in a separate table and we make the column `department` the primary key of this new table, the functional dependency is enforced by the constraint!

```
1 CREATE TABLE department (
2 department VARCHAR(128) PRIMARY KEY,
3 faculty VARCHAR(128) NOT NULL);
```

$$\{department\} \rightarrow \{faculty\}$$

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

The new `proficiency` table has an `department` column but no `faculty` column. We make the `department` column in the new `proficiency` table a foreign key referencing the primary key of the `department` table.

proficiency				
name	userid	domain	department	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	C++
...				

```
1 CREATE TABLE proficiency (
2 ...
3 department VARCHAR(128) REFERENCES department(department);
4 ...);
```

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R

```
1 DELETE FROM proficiency WHERE userid='ami' AND domain='med.sut.edu';
```

Deletion Anomaly

When Amirah Moktar (`ami@med.sut.edu`) graduates, we may forget that we have a department of pharmacy and a faculty of medcine.

This cannot happen if we store the departments and their faculties in a separate table.

```
1 CREATE TABLE department (
2 department VARCHAR(128) PRIMARY KEY,
3 faculty VARCHAR(128) NOT NULL);
```

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
null	null	null	pharmacy	medecine	null

Insertion Anomaly

We cannot record the department of pharmacy and the faculty of medicine as there is no student from this department and faculty and we cannot have null values for `name`, `userid`, and `domain`, and `language` because they should be `NOT NULL` (some are prime attributes part of the `PRIMARY KEY`).

The solution , as we have seen, is to modify the `proficiency` table and create a separate `department` table.

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

We could use an outer join[a] to combine the two tables and recreate the original table with null values that we intended.

[a] Do we need a full, left,or right outer join?

```
1 SELECT *
2 FROM proficiency p FULL OUTER JOIN  department d ON p.department = d.department;
```

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
null	null	null	pharmacy	medecine	null

Verify that we have similar anomalies with the other non-trivial functional dependencies.

The purpose of normal forms is to recognise designs that enforce functional dependencies by means of the main SQL constraints (primary key, unique, not null, and foreign key constraints) and thus protect the data against anomalies.

The purpose of normalisation is to transform (decompose) a poor design into a design that enforces functional dependencies by meansthe main SQL constraints.

The candidate key is $\{userid, domain, language\}$, yet some attributes, like $name$ and $department$ depend only on a proper subset of a candidate key (they are not fully dependent on the primary key):

$$\{userid, domain\} \rightarrow \{name, department\}$$

The candidate key is $\{userid, domain, language\}$, yet some attributes, like $faculty$, also depend on other attributes (they are transitively dependent on the primary key):

$$\{department\} \rightarrow \{faculty\}$$

These attributes describe the student and the department and do not depend on or inform us about the relationship with the programming language. We are mixing several entities and relationships in the same table.

"A non-key field must provide a fact about the key[s], the whole key[s], and nothing but the key[s], [so help me Codd.]",
W. Kent in "A Simple Guide to Five Normal Forms in Relational Database Theory", Communication of the ACM, Volume 26, Number 2 (1983).

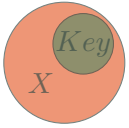
Theorem

A relation R with a set of functional dependencies Σ is in BCNF if and only if for every functional dependency $X \to \{A\} \in \Sigma^+$:

- $X \to \{A\}$ *is trivial or*
- X *is a superkey.*

It is sufficient to look at Σ.

For some candidate key, we must have one of the following:



X is a superset of the candidate key (X is a superkey).

X is the candidate key (X is a superkey).

proficiency					
name	userid	domain	department	faculty	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	computing	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	computing	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	computing	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	science	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	science	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	medecine	R

The candidate key is $\{userid, domain, language\}$.

$\{department\} \rightarrow \{faculty\}$ is non-trivial, its left-hand side is not a superkey. It does violate the two conditions of the theorem.

We have found a culprit. The table is not in BCNF.

The solution to avoid anomalies is to decompose the table into fragments..

proficiency				
name	userid	domain	department	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	Python
Goh Jin Wei	go	comp.sut.edu	information systems and analytics	Python
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	C++
Tan Hee Wee	tanhw	eng.sut.edu	computer engineering	Fortran
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	C++
Bjorn Sale	bjorn	eng.sut.edu	computer engineering	Fortran
Tan Hooi Ling	tanh	sci.sut.edu	physics	Julia
Tan Hooi Ling	tanh	sci.sut.edu	physics	Fortran
Roxana Nassi	rox	sci.sut.edu	mathematics	R
Amirah Mokhtar	ami	med.sut.edu	pharmacy	R

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

A solution is again to decompose the table into two.

proficiency				
name	userid	domain	department	language
Tan Hee Wee	tanh	comp.sut.edu	computer science	JavaScript
Tan Hee Wee	tanh	comp.sut.edu	computer science	Python
Tan Hee Wee	tanh	comp.sut.edu	computer science	C++
Stanley Georgeau	stan	comp.sut.edu	computer science	Python
...			s	

department	
department	faculty
computer science	computing
information systems and analytics	computing
computer engineering	engineering
physics	science
mathematics	science
pharmacy	medecine

We now need to study the two new tables and their (projected) functional dependencies. (Verify that they are in BCNF.)

A decomposition of a table R is a set of tables $\{R_1, \cdots, R_n\}$ such that $R = R_1 \cup \cdots \cup R_n$.

A binary decomposition of a table R is a pair of tables $\{R_1, R_2\}$ such that $R = R_1 \cup R_2$.

A binary decomposition is lossless-join if and only if the full outer natural join of its two fragments (the two tables resulting from the decomposition) equals the initial table (otherwise it is lossy.)

```
1  SELECT p.name, p.userid, p.domain, p.department, p.faculty, p.language
2  FROM proficiency p FULL OUTER JOIN department d ON p.department = d.department;
3
```

Theorem

A binary decomposition of R into R_1 and R_2 is lossless-join if $R = R_1 \cup R_2$ and $R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$.

Note that if $R_1 \cap R_2$ is the primary key of one of the two tables, then it can be a foreign key in the other referencing the primary key.

Theorem

A decomposition is lossless-join if there exists a sequence of binary lossless-join decomposition that generates that decomposition.

Consider a relation R with a set of functional dependencies Σ. A set Σ' of projected functional dependencies on R' from R with Σ, where $R' \subset R$, is the set of functional dependencies equivalent to the set of functional dependencies $X \rightarrow Y$ in Σ^+ such that $X \subset R'$ and $Y \subset R'$.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

What is a set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

What is a set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?

$\Sigma' = \{\{A, B\} \to \{D, E\}, \{B\} \to \{E\}, \{B\} \to \{D\}\}$

A decomposition of R with Σ into $R_1 \cdots R_n$ with the respective sets of projected functional dependencies $\Sigma_1 \cdots \Sigma_n$ is dependency preserving if and only if $\Sigma^+ = (\Sigma_1 \cup \cdots \cup \Sigma_n)^+$.

$R = \{A, B, C\}$ with $\Sigma = \{\{C\} \rightarrow \{B\}\}$ (the candidate key is $\{A, C\}$)}) is decomposed into a dependency preserving lossless-join decomposition $R_1 = \{A, C\}$ with the set of projected functional dependenncies $\Sigma_1 = \emptyset$ and $R_2 = \{B, C\}$ with the set of functional dependencies $\Sigma_2 = \{\{C\} \rightarrow \{B\}\}$.

$R = \{A, B, C\}$ with $\Sigma = \{\{C\} \to \{B\}, \{A, B\} \to \{C\}\}$ (the candidate keys are $\{A, C\}$ and $\{A, B\}$) is decomposed into a non dependency preserving lossless-join decomposition
$R_1 = \{A, C\}$ with the set of functional dependencies $\Sigma_1 = \emptyset$ and
$R_2 = \{B, C\}$ with the set of functional dependencies $\Sigma_2 = \{\{C\} \to \{B\}\}$.
The functional $\{A, B\} \to \{C\}$ is lost!

$R = \{A, B, C\}$ with $\Sigma = \{\{A\} \to \{B\}, \{B\} \to \{C\}, \{A\} \to \{C\}\}$ (the candidate key is $\{A\}$) is decomposed into a dependency preserving lossless-join decomposition $R_1 = \{A, B\}$ with the set of functional dependencies $\Sigma_1 = \{\{A\} \to \{B\}\}$ and $R_2 = \{B, C\}$ with the set of functional dependencies $\Sigma_2 = \{\{B\} \to \{C\}\}$.
The functional $\{A\} \to \{C\}$ is not lost!

When a relation is not in BCNF [a], we can pick one of the functional dependencies violating the BCNF definition and use it to decompose the relation into two relations. We continue decomposing until every fragment is in BCNF.

[a] The same algorithm work for other normal forms.

The decomposition algorithm is guaranteed to find a lossless decomposition in BCNF.

The decomposition may not be dependency preserving.

Decomposition

Let $X \to Y$ be a functional dependency in Σ that violates the BCNF definition (it is not trivial and X is not a superkey). We use it decompose R into the following two relations R_1 and R_2.

- $R_1 = X^+$,
- $R_2 = (R - X^+) \cup X$.

We must now check whether R_1 and R_2 with the respective sets of projected functional dependencies Σ_1 and Σ_2 are in BCNF and continue the decomposition if they are not.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

$\Sigma'' = \{\{B\} \to \{C\}, \{C\} \to \{B, D, E\}\}$

Is R with Σ in BCNF?

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

The two candidate keys are $\{A, B\}$ and $\{A, C\}$.

$\{C\} \to \{D\}$ is non-trivial and $\{C\}$ is not a superkey key. Therefore it is not in BCNF.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

Let us decompose R with Σ into a lossless decomposition in BCNF.

$\{B\} \to \{C\}$ violates the BCNF definition (it is non-trivial and $\{B\}$ is not a superkey.

We decompose into two fragments.

$R_1 = \{B\}^+ = \{B, C, D, E\}$ with the projected functional dependencies
$\Sigma_1 = \{\{B\} \to \{C, D, E\}, \{C\} \to \{B\}\}$.

$R_2 = (R - \{B\}^+) \cup \{B\} = \{A, B\}$ with the projected functional dependencies $\Sigma_2 = \emptyset$.

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Yes they are in BCNF. We can stop here. Otherwise we would have to continue decomposing whichever fragments are not in BCNF.

Is the decomposition lossless?

The decomposition is guaranteed to be lossless (by properties of the algorithm).

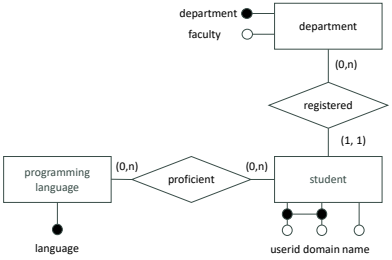
Have we lost any functional dependency?

No, we can recover all the functional dependencies in Σ from $\Sigma_1 \cup \Sigma_2$. The decomposition is dependency preserving.

Can we choose another dependency to decompose and reach a different result?

Yes ... do it.

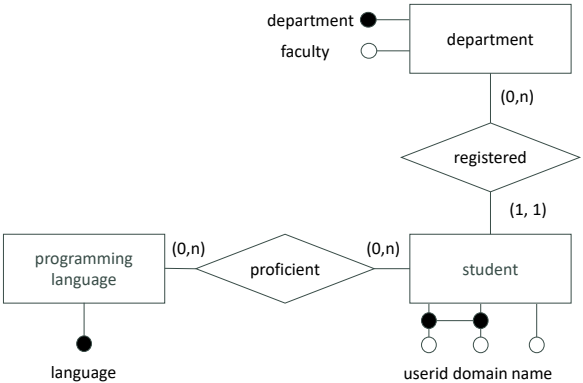
What about our case?



What about our case?

Let us call the table R (`proficiency`) and the columns A (`name`) , B (`userid`), C (`domain`), D (`department`), E (`faculty`), and F (`language`), respectively.

$R = \{A, B, C, D, E, F\}$
$\Sigma = \{\{B, C\} \to \{A\}, \{B, C\} \to \{D\}, \{D\} \to \{E\}, \{B, C, F\} \to \{A, B, C, D, E, F\}\}$


department
faculty
department
(0,n)
registered
(1, 1)
programming language
(0,n)
proficient
(0,n)
student
language
userid domain name

We get the following lossless dependency preserving BCNF decomposition.

$R_{1.1} = \{B, C, F\}$ with $\Sigma_{1.1} = \emptyset$,
$R_{1.2} = \{A, B, C, D\}$ with $\Sigma_{1.2} = \{\{B, C\} \to \{A, D\}\}$,
$R_2 = \{D, E\}$ with $\Sigma_2 = \{\{D\} \to \{E\}\}$.

With A (`name`) , B (`userid`), C (`domain`), D (`department`), E (`faculty`), and F (`language`), respectively.