Normalisation Example

CS2312

Normalisation Example

BEER_DATABASE

beer	brewery	strength	city	region	ware house	quantity
Choice	Websters	XX	York	North West	1	200
Choice	Websters	XX	York	North West	4	100
Choice	Websters	XX	York	North West	8	200
Old Bob	Websters	XXX	York	North West	1	300
Old Bob	Websters	XXX	York	North West	2	300
Landlord	Taylors	XXX	Leeds	North West	8	200
Landlord	Taylors	XXX	Leeds	North West	3	190
Directors	Fremlins	X	London	South East	6	400
Directors	Fremlins	X	London	South East	4	290
Wobbly Joe	Sam Smith	XXXX	York	North West	4	90
Watery	Whitbread	O	London	South East	null	null

Additional Notes: Warehouses are shared by breweries.

Each beer is unique to the brewer. Each brewery is based in a city.

Minimal Sets of Functional Dependencies

- A set of functional dependencies F is minimal if:

1. Every dependency F has a single determined attribute A
2. We cannot remove any dependency from F and still have a set of dependencies equivalent to F
3. We cannot replace and dependency X $\rightarrow$ A in F with a dependency A$\rightarrow$ X, where A $\subset$ X and still have a set of dependencies that is equivalent to F

I.e. a canonical form with no redundancies

(beer, brewery, strength, city, region, warehouse, quantity)

- beer$\rightarrow$ brewery
- beer$\rightarrow$ strength
- brewery $\rightarrow$ city
- city $\rightarrow$ region
- beer, warehouse, $\rightarrow$ quantity

Relational Synthesis Algorithm into 3NF: (beer, brewery, strength, city, region, {warehouse, quantity})

set D := { R } ; P. 426, P. 431

1. Find a minimal cover G for F
2. For each determinant X of a functional dependency that appears in G
   create a relation schema { X $\cup$ A1, X $\cup$ A2…X $\cup$ Am} in D where
   X $\rightarrow$ A1, X $\rightarrow$ A1, … X $\rightarrow$ A1m are the only dependencies in G with X as the determinant;
3. Place any remaining (unplaced) attributes in a single relation to ensure attribute preservation property so we don't lose anything.
4. If none of the relations contains a key of R, create one more relation that contains attributes that form a key for R.

beer$\rightarrow$ brewery	(beer, brewery, strength)
beer$\rightarrow$ strength	
brewery $\rightarrow$ city	(brewery, city)
city $\rightarrow$ region	(city, region)
beer, warehouse, $\rightarrow$ quantity	(beer, warehouse, quantity)

Step-wise normalisation:
(<u>beer</u>, brewery, strength, city, region, {<u>warehouse</u>, quantity})

- beer→ brewery, strength — *partial dependency*
- brewery → city — *transitive dependency*
- city → region — *transitive dependency*
- beer, warehouse, → quantity — *repeating group*

1NF remove repeating group
(<u>beer</u>, brewery, strength, city, region, {<u>warehouse</u>, quantity})

(<u>beer, warehouse</u>, quantity)
beer, warehouse, → quantity

transitive dependency
transitive dependency

(<u>beer</u>, brewery, strength, city, region)

beer→ brewery, strength
brewery → city
city → region

(<u>beer</u>, brewery, strength, city, region)

- beer→ brewery, strength
- brewery → city — *transitive dependency*
- city → region — *transitive dependency*

- **2NF** **no partial dependencies**
- **3NF/BCNF** **no transitive dependencies**

(<u>beer</u>, brewery, strength, city, region)

Take the most indirect transitive dependencies

(city, region)
city → region

(<u>beer</u>, brewery, strength, city)
beer→ brewery, strength
brewery → city

(<u>brewery</u>, city)
brewery → city

(<u>beer</u>, brewery, strength)
beer→ brewery, strength

Using BNCF decomposition algorithm:
(<u>beer</u>, brewery, strength, city, region, <u>warehouse</u>, quantity)

- beer→ brewery, strength *partial dependency*
- brewery → city *transitive dependency*
- city → region *transitive dependency*
- beer, warehouse, → quantity

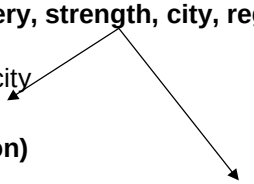
Directly to BCNF

take a violating dependency and form a relation from it.

First choose a direct transitive dependency and its *closure*

(<u>beer</u>, brewery, strength, city, region, <u>warehouse</u>, quantity)

brewery → city



(<u>brewery</u>, city, region)

brewery → city

city → region *transitive dependency*

(<u>beer</u>, brewery, strength, <u>warehouse</u>, quantity)

beer→ brewery, strength *partial dependency*

beer, warehouse, → quantity

Using BNCF decomposition algorithm:
(<u>beer</u>, brewery, strength, city, region, <u>warehouse</u>, quantity)

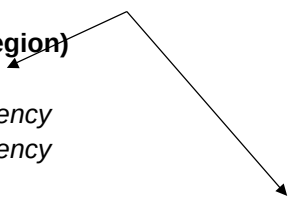
- beer→ brewery, strength *partial dependency*
- brewery → city *transitive dependency*
- city → region *transitive dependency*
- beer, warehouse, → quantity

take a violating dependency and form a relation from it.

First the partial dependency and its *closure*

(<u>beer</u>, brewery, strength, city, region, <u>warehouse</u>, quantity)

beer→ brewery, strength



(<u>beer</u>, brewery, strength, city, region)

beer→ brewery, strength

brewery → city *transitive dependency*

city → region *transitive dependency*

normalise as before...

(<u>beer</u>, <u>warehouse</u>, quantity)

beer, warehouse, → quantity