L18: Database Normalization

Dr Kieran T. Herley

Semester One, 2023-24

School of Computer Science & Information Technology
University College Cork

Summary

The perils of redundancy in DB design. Normalization as methodology for reducing redundancy. Functional dependencies.

More on Database Design

- Designs produced by ER approach may need further refinement and tuning
- Why?
  - Eliminate certain undesirable redundancy patterns (normalization)
  - (Also performance tuning to enhance query performance etc.)

A Simple Database

suppliers

snum	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	10	Paris
S4	Clark	20	London
S5	Adams	30	Athens

parts

pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

ordered_from

snum	pnum	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

A simple (but not perfect) design for DB for managing a company's orders for spare parts:

- Keeps track of suppliers we use and items we require
- Also track details of outstanding orders

Assumption: "status" captures some city-dependent characteristic such as delivery charge.

A Not-So-Good Design

suppliers′

snum	sname	city
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris
S4	Clark	London
S5	Adams	Athens

parts

pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

ordered_from′

snum	pnum	quantity	status
S1	P1	300	20
S1	P2	200	20
S1	P3	400	20
S1	P4	200	20
S1	P5	100	20
S1	P6	100	20
S2	P1	300	10
S2	P2	400	10
S3	P2	200	30
S4	P2	200	20
S4	P4	300	20
S4	P5	400	20

This variation contains undesirable *redundancy*

- Supplier status info. replicated many times in `ordered_by`′
- May complicate efforts to modify this info.

What's So Bad About Redundancy?

Design Principle

Avoid redundancy as far as possible

What's So Bad About Redundancy?

Design Principle

Avoid redundancy as far as possible

Watchword

One fact in one place

What's So Bad About Redundancy?

Design Principle

Avoid redundancy as far as possible

Watchword

One fact in one place

Redundancy Pitfalls

Space Usage Duplicating information in several places wastes space

Anomalies

Update Anomalies If we change one copy of a datum, we need to change them all, otherwise inconsistencies arise

Also insertion and deletion anomalies (examples later)

Normalization

- Normal forms capture desirable traits that reduce risk of certain DB inconsistensies
- Form hierarchy of increasing stringency:

$$1\text{NF} < 2\text{NF} < 3\text{NF} \cdots$$

First Normal Form

R is in First Normal Form (1NF) if and only if it contains atomic values only (i.e. no multi-valued attributes).

Our first Design (Poor)

first				
snum	status	city	pnum	quantity
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

parts			
pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

Basically, we mash suppliers and orders information together

What's Wrong With This Design?

first

snum	status	city	pnum	quantity
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

parts

pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

Update Anomaly City value for suppliers replicated; on updates need to modify all copies, else . . .

Insertion Anomaly Cannot record supplier's details (city) until that supplier supplies at least one part.

Deletion Anomaly If we delete last tuple for a supplier, we lose all info. about that supplier.

Functional Dependencies

Dependencies Dependencies capture how different attributes in our table(s) interrelate

Definition

Given relation R, attribute Y is functionally dependent on attribute X (denoted $X \rightarrow Y$) if and only if, whenever two tuples of R agree on their X-value, they are also guaranteed to agree also on their Y-value.

Normalization Design methodology guided by careful analysis of these can help reduce undesirable redundancy patterns

Note Dependencies are dictated (directly or indirectly) by the semantics of the database attributes.

FDs in First Design

Definition

Given relation R, attribute Y is functionally dependent on attribute X if and only if, whenever two tuples of R agree on their X-value, they are also guaranteed to agree also on their Y-value.

Dependencies:

```
parts:
pnum -> pname
pnum -> weight
pnum -> colour

first:
pnum, snum -> status, city, qty
snum -> status, city
city -> status
```

first

snum	status	city	pnum	quantity
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

parts

pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

Note: each attribute dependent on table key, but there may be others

Towards a Better Design

`first` **table**

`first(snum, status, city, pnum, quantity)`

Diagnosis Attributes status, city in `first` “relate” only to part of key (snum) not the whole key (snum, pnum)

```
first:
pnum, snum -> status, city, qty
snum -> status, city
city -> status
```

Such “non-full dependencies” are problematic

Prescription Remove such dependencies by splitting `first` into

```
second(snum, status, city)
ordered_from(snum, pnum, quantity)
```

Schemas without non-full dependencies are in Second Normal Form (2NF)

Our second Design (Better)

second

snum	status	city
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

ordered_from

snum	pnum	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Solves the three specific anomalies mentioned earlier, but . . .

Problems With Second Design

second		
snum	status	city
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

ordered_from		
snum	pnum	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Update Anomaly The status value for each city is replicated many times; if the status for London changes, then either we must update all the tuples for that city or worse have inconsistent status info. for that city in our table.

Insertion Anomaly Cannot enter status values for city until we have a supplier located in that city.

Deletion Anomaly If we delete the only tuple for a particular city, we lose the status info for that city.

Dependencies In Second Design

`second` **table** `second(snum, status, city)`

Diagnosis "indirect" (transitive) dependence of status on snum via city

```
snum -> city
city -> status
snum -> status
```

Such dependencies are problematic

Prescription To eliminate problem split `second` table into

```
suppliers(snum, city)
cities(city, status)
```

Schemas with no transitive dependecies are in Third Normal Form (3NF)

The Final Design

```
ordered_from(snum, pnum, quantity)
parts(pnum, pname, colour, weight)
suppliers(snum, city)
cities(name, status)
```

suppliers

snum	sname	city
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris
S4	Clark	London
S5	Adams	Athens

parts

pnum	pname	colour	weight
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Red	12
P6	Cog	Red	19

Cities

status	name
London	20
Paris	10
Athens	30

ordered_from

snum	pnum	quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

In a Nutshell

“ [Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key [so help me Codd] ”
(*Attributed to Bill Kent*)

Notes and Acknowledgements

The suppliers-Part example is adapted from “An Introduction to Database Systems” (3rd ed.) by C. J. Date.