ICCS240 Database Management

Normalization

Many slides in this lecture are either from or adapted from slides provided by

Jeff Ullman, Stanford U

Werner Nutt, Free U of Bozen-Bolzano

Goal: CODIFY COMMON SENSE

Design “sensible” database schemas by using the theory of normalization to guide and validate your design.

Is your database design good?

Approach to Database Design

Approach #1:

- Draw E/R diagram & translate into tables
- Subjective. *How do we know if this is good?*

Approach #2:

- Start with **universal** relations and decompose using functional dependencies

Approach #3:

- Draw E/R diagram & translate into tables
- Validate/refine using functional dependencies

Example: Universal Relation

Book	Author	Author Nationality	Book Type	Price	Subject	Pages	Thickness	Publisher	Publisher Country	Publication Type	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL, Database, Design	520	Thick	Apress	USA	E-book	1	Tutorial

The book “Beginning MySQL…” was authored by Chad Russell, who is American, in Hardcover; the price is 49.99; there are 520 pages, which makes it a Thick book; and it’s published by Apress in the USA; …

In terms of design:

+ Fast queries (no JOIN)

- Redundancy, potential update/delete anomalies

More Example of Bad Design – redundancy

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

name	Addr	beersLiked	manf	favBeer
Alice	USA	Bud	A.B.C.	WickedAle
Alice	???	Ale	X.Y.	???
Bob	China	Bud	???	FreshBud

Data is **redundant**,
because each of the ???'s can be figured out by using FDs:
name → addr favBeer and beersLiked → manf.

More Example of Bad Design – anomalies

Drinkers(name, addr, beersLiked, manf, favBeer)

name	Addr	beersLiked	manf	favBeer
Alice	USA	Bud	A.B.C.	WickedAle
Alice	USA	Ale	X.Y.	WickedAle
Bob	China	Bud	A.B.C.	FreshBud

Update **anomaly**: If `Alice` is transferred to `Brazil`, will we remember to update/change each of her tuples?

Deletion **anomaly**: If nobody likes Bud, we lose track of the fact that A.B.C. manufactures Bud.

To DECOMPOSE, or NOT to DECOMPOSE

Denormalized
(Flat, redundant, ...)

Normalized
(Lean, indirect, …)

1. **Lossless Reconstruction:** Can reconstruct the big relation by joining small ones

2. **Dependency Preservation:** Integrity check should be cheap (requires no join across table to assert simple dependencies)

3. **Redundancy Avoidance:** Avoid unnecessary data duplication.

4. **Good Performance:** Retain good performance suitable for the application.

Lossless and Lossy Decomposition

Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

Decomposition is **lossless** if it is feasible to reconstruct relation R from decomposed tables using (natural) Joins, more formally:

Suppose you decompose a relation R into relations R_1, R_2, the decomposition is *lossless* if $R_1 \bowtie R_2 = R$.

Example – lossless decomposition

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

=

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

⋈

<DeptDetails>

Dept_ID	Emp_ID	Dept_Name
Dpt1	E001	Operations
Dpt2	E002	HR
Dpt3	E003	Finance

Example from https://www.tutorialspoint.com/Lossless-and-Lossy-Decomposition-in-DBMS

Example – lossy decomposition

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

deompose into …

$\neq$

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

$\bowtie$

<DeptDetails>

Dept_ID	Dept_Name
Dpt1	Operations
Dpt2	HR
Dpt3	Finance

Example from https://www.tutorialspoint.com/Lossless-and-Lossy-Decomposition-in-DBMS

Lossless-Join Decomposition Criteria

- Verify join explicitly: $R = R_1 \bowtie R_2$

- Using FDs…

  Let R be a relational schema and
  $\mathcal{F}$ be a set of functional dependencies on R

  Let R_1 and R_2 form a decomposition of R

  The decomposition is lossless if at least one of the following functional dependencies are in $\mathcal{F}^+$ (closure of every attribute or attribute sets in $\mathcal{F}$).

  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

  $R_1 \cap R_2$: Attributes in both R_1 and R_2

Practically: How many tables? What tables?

ORDER_ITEM

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	100200	1	300.00	300.00
6	3000	101100	2	50.00	100.00
7	3000	101200	1	50.00	50.00

SKU_DATA

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

SKU_ITEM

	OrderNumber	SKU	Quantity	Price	SKU_Description	Department	Buyer
1	1000	201000	1	300.00	Half-dome Tent	Camping	Cindy Lo
2	1000	202000	1	130.00	Half-dome Tent Vestibule	Camping	Cindy Lo
3	2000	101100	4	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	2000	101200	2	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	3000	100200	1	300.00	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
6	3000	101100	2	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
7	3000	101200	1	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Should we store these two tables as they are,

or

should we combine them into one table in our new database?

Normal Forms

- **1NF** – Every attribute is atomic.
- **2NF** – A relation is in 2NF if all non-key attributes are independent on the whole of every candidate key.
- **3NF** – A relation is in 3NF if it is in 2NF and has no determinants except primary key.
- **BCNF** (Boyce-Codd Normal Form) – A relation is in BCNF if every determinant is a candidate key.

1NF

1NF: ensure atomic values

Book	Author	Author Nationality	Book Type	Price	Subject	Pages	Thickness	Publisher	Publisher Country	Publication Type	Genre ID	Genre Name
Beginning MySQL Database Design and Optimization	Chad Russell	American	Hardcover	49.99	MySQL, Database, Design	520	Thick	Apress	USA	E-book	1	Tutorial

Design Choice #1:

Subject 1	Subject 2	Subject 3
MySQL	Database	Design

Design Choice #2:

Repeat this row 3 times, each with a different subject.

Design Choice #3:

Subject

Subject ID	Subject name
1	MySQL
2	Database
3	Design

Book - Subject

Book	*Subject ID*
Beginning MySQL Database Design and Optimization	*1*
Beginning MySQL Database Design and Optimization	*2*
Beginning MySQL Database Design and Optimization	3

2NF

2NF: all non-key attributes must be dependent on the whole of every candidate key

Book

Book	Book Type	Author	Author Nationality	Price	Pages	Thickness	Genre ID	Genre Name	*Publisher ID*
Beginning MySQL Database Design and Optimization	Hardcover	Chad Russell	American	49.99	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	Paperback	E.F.Codd	British	39.99	538	Thick	2	Popular science	*2*
Beginning MySQL Database Design and Optimization	E-book	Chad Russell	American	22.34	520	Thick	1	Tutorial	*1*
The Relational Model for Database Management: Version 2	E-book	E.F.Codd	British	13.88	538	Thick	2	Popular science	*2*

Candidate Key: (Book, Book Type)

The term **non-key attribute** means an attribute that is neither
(1) a candidate key itself, nor
(2) part of a composite candidate key.

Book Type - Prices

Book	Book Type	Price
Beginning MySQL Database Design and Optimization	Hardcover	49.99
The Relational Model for Database Management: Version 2	Paperback	39.99
Beginning MySQL Database Design and Optimization	E-book	22.34
The Relational Model for Database Management: Version 2	E-book	13.88

Analysis:

- Only Price (and perhaps Pages) depends on both (Book, Book Type)
- The other (non-key) attributes only depend on Book

2NF

Originally defined by by E.F. Codd in 1971

A relation is in the **second normal form (2NF)** if:

- It is in first normal form (1NF)
- It does not have any non-key attribute that is functionally dependent on any proper subset of any candidate key of the relation.

Note that there is no restriction on the non-key to non-key attributes dependency. (addressed in 3NF)

BCNF

Recall: Functional Dependency

- A functional dependency occurs when the value of one (set of) attribute(s) determines the value of a second (set of) attribute(s)

- The attribute on the left side of the FD is called **determinant**.

BCNF: every determinant is a candidate key

We say a relation R is in BCNF if
whenever $X \to A$ is a nontrivial FD that holds in R,
then X is a superkey.

- Note: nontrivial means $A \notin X$.
- Note: superkey is any superset of a key (not necessarily a proper superset)

Example: *not BCNF*

name	Addr	beersLiked	manf	favBeer
Alice	USA	Bud	A.B.C.	WickedAle
Alice	USA	Ale	X.Y.	WickedAle
Bob	China	Bud	A.B.C.	FreshBud

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

FDs: name → addr favBeer, beersLiked → manf

- The only key is {name, beersLiked}
- In each FD above, the left side is ***not*** a superkey

⇒ Any one of these FDs shows Drinkers is not in BCNF

Each of the above FDs is a **partial dependency**

i.e., the right side depends only on a part of the key.

Another example: *not BCNF*

Beers(<u>name</u>, manf, manfAddr)

FDs: name → manf, manf → manfAddr

- The only key is {name}
- Here, name → manf does not violate BCNF

  … but manf → manfAddr does

The second FD is a transitive dependency, because manfAddr departs on manf, which is not part of any key.

Decomposition into BCNF

Given a relation R with FDs $\mathcal{F}$

Goal: decompose R into relations $R_1, \ldots, R_n$ such that

- Each R_i is a **projection** of R
- Each R_i is in *BCNF* (w.r.t. the projection of $\mathcal{F}$)
- R is the natural join of $R_1, \ldots, R_n$

Intuition: R is broken into pieces,

- that contain the same information as R,
- *but* are free of redundancy

The Algorithm: Divide and Conquer

- Look in $\mathcal{F}$ for an FD $X \rightarrow B$ that *violates* BCNF

  (If any FD following from $\mathcal{F}$ violates BCNF, then there is surely an FD in $\mathcal{F}$ itself that violates BCNF.)

- Compute X^+

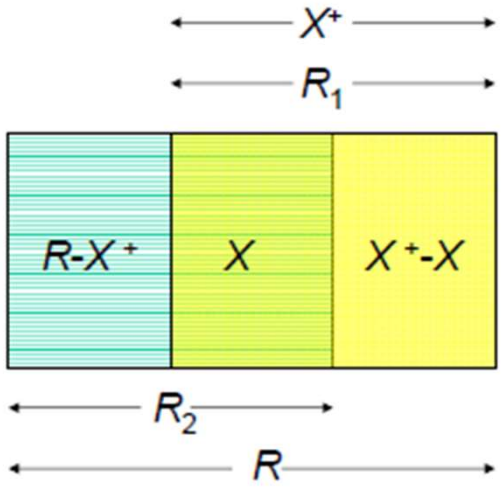
  (Note that X^+ does not contain all attributes of R, otherwise, X would be superkey.)

- Decompose R using $X \rightarrow B$,
  i.e., replace R by relations with schemas

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

- Compute the projection $\mathcal{F}_1, \mathcal{F}_2$ of $\mathcal{F}$ on R_1, R_2

- Continue with $R_1, \mathcal{F}_1$ and $R_2, \mathcal{F}_2$



Example

Not yet in BCNF

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

FDs: name → addr favBeer, beersLiked → manf

$X \to B$

- Pick the BCNF violation: name → addr
- Closure of the left sdie: $\{\text{name}\}^+ = \{\text{name}, \text{addr}, \text{favBeer}\}$
- Decomposed relations:

X^+

Drinkers1(name, addr, favBeer)

Drinkers2(name, beersLiked, manf)

$(R - X^+) \cup X$

Example (cont.)

- For Drinkers1(name, addr, favBeer), the only relevant FDs are name → addr and name → favBeer:
  - The only key is {name}
  - Hence, Drinkers1 is in BCNF.

- For Drinkers2(name, beersLiked, manf), the only relevant Fds are beersLiked → manf:
  - The only key is {name, beersLiked}
  - Hence, **violate BCNF**.

- Continue with Drinkers2, …

Example (cont.)

Drinkers2(name, beersLiked, manf)

FDs: $\mathcal{F}_2 = \{\text{beersLiked} \rightarrow \text{manf}\}$

- Pick the BCNF violation: $\text{beersLiked} \rightarrow \text{manf}$
- Closure of the left side: $\{\text{beersLiked}\}^+ = \{\text{beersLiked}, \text{manf}\}$
- Decomposed relations:

  Drinkers3(beersLiked, manf)

  Drinkers4(name, beersLiked)

Example (cont.)

- For Drinkers3(beersLiked, manf), the only relevant FDs are beersLiked → manf:
  - The only key is {beersLiked}
  - Hence, Drinkers3 is in BCNF.

- ForDrinkers4(name, beersLiked), there is no relevant FD:
  - The only key is {name, beersLiked}
  - Hence, Drinkers4 is in BCNF.

Example (concluded)

We have decomposed

Drinkers(name, addr, beersLiked, manf, favBeer)

into

- Drinkers1(name, addr, favBeer)
- Drinkers3(beersLiked, manf)
- Drinkers4(name, beersLiked)

Notice: Drinkers1 is about drinkers,
Drinkers3 is about beers,
Drinkers4 is about the relationship between drinkers and the beers they like

3NF

3NF - motivation

- There is one structure of FD's that causes trouble when we decompose.

$$AB \rightarrow C \text{ and } C \rightarrow B.$$

Example: A = street address, B = city, C = zip code.

- There are two keys: $\{A, B\}$ and $\{A, C\}$.
- Here, $C \rightarrow B$ is a BCNF violation,
  so we must decompose into AC, BC.

If we decompose ABC into AC and BC, then
we cannot enforce $AB \rightarrow C$ by checking FDs in the decomposed relations.

An unenforceable FDs

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

Join tuples with equal zip codes.

street	city	zip
545 Tech Sq.	Cambridge	02138
545 Tech Sq.	Cambridge	02139

Although no FD's were violated in the decomposed relations,
FD streetAddr city $\rightarrow$ zip is violated by the database as a whole.

3NF – Let's avoid this trouble!

3rd Normal Form (3NF) modifies the BCNF condition
so we do not have to decompose in this problem situation.

- An attribute is ***prime*** if it is a member of any key.

- $X \rightarrow A$ violates 3NF **iff**
  - X is not a superkey, and also
  - A is not prime.

Example

$X \to A$ violates 3NF **iff**
X is not a superkey, and
A is not prime.

- In our problem situation with FDs,

  $AB \to C$ and $C \to B$

  Note that, we have keys AB and AC
- Thus, A, B and C are each prime.

- Although $C \to B$ violates BCNF, it does not violate 3NF.

3NF: No non-primes determined by another non-primes

Another definition of 3NF [Carlo Zaniolo, 1982]:

A relation R is in 3NF **iff** for each of its FDs $X \rightarrow A$,
at least one of the following conditions holds:

- X contains A ($X \rightarrow A$ is trivial functional dependency)
- X is a superkey
- Every element of $A \setminus X$, is a prime attribute.

A clear sense of the difference between 3NF and BCNF.
That is, BCNF simply eliminates the 3rd alternative.

What 3NF and BCNF give you …

There are two important properties of a decomposition:

- **(1) Losslessness**:
  It should be possible to project the original relation onto the decomposed schema, and then reconstruct the original.

- **(2) Dependency Preservation**:
  It should be possible to check in the projected relations whether all the given FDs are satisfied.

What 3NF and BCNF give you … (cont.)

- We can get (1) with a BCNF decomposition
- We can get both (1) and (2) with a 3NF decomposition
- But we cannot always get (1) and (2) with a BCNF decomposition

"street-city-zip" is an example

Exercise

Let R be a relation with attributes $ABCD$.
Consider the following combinations of FDs on R:

- $AB \to C, C \to D, D \to A$
- $B \to C, B \to D$
- $AB \to C, BC \to D, CD \to A, AD \to B$
- $A \to B, B \to C, C \to D, D \to A$

For each collection of FDs do the following:

1. Indicate all the BCNF violations
2. Decompose the relations into collection of relations that are in BCNF
3. Are the decompositions dependency preserving?