# Tutorial 3
# Functional Dependencies and Normalization

CPSC 304
Introduction to Relational Databases

## Before you start...

- Please download vehicle_rental_specifications.pdf (from Tutorial 1) and Tutorial3_FDsNORM.pdf from Canvas.
- A subset of the entities and attributes in the PDF document will be used for this tutorial.
- You don’t need to submit anything. Tutorial solutions will be available at the same time as the next tutorial is released.

## SuperRentInfo

**Consider the following relational schema and functional dependencies (FDs):**

**SuperRentInfo**(customerID, customerName, email, startDate, endDate, city, confirmationNumber, branchName, typeName, paymentCode)

… with the following FDs:

1) customerID → customerName, email
2) confirmationNumber → customerID
3) branchName → city
4) confirmationNumber, branchName → typeName
5) confirmationNumber, branchName, typeName → startDate, endDate
6) paymentCode → customerID, confirmationNumber

## SuperRentInfo

To save yourself some writing time in the questions that follow, let us abbreviate the above schema to:

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) $I \rightarrow N, E$
2) $CN \rightarrow I$
3) $B \rightarrow C$
4) $CN, B \rightarrow T$
5) $CN, B, T \rightarrow S, ED$
6) $P \rightarrow I, CN$

## Task #1: Anomalies

Give an instance of **SuperRentInfo** (i.e., a relation with a few rows in it) that illustrates these **three anomalies**: insertion, deletion, and update. Explain how your table shows the three anomalies.

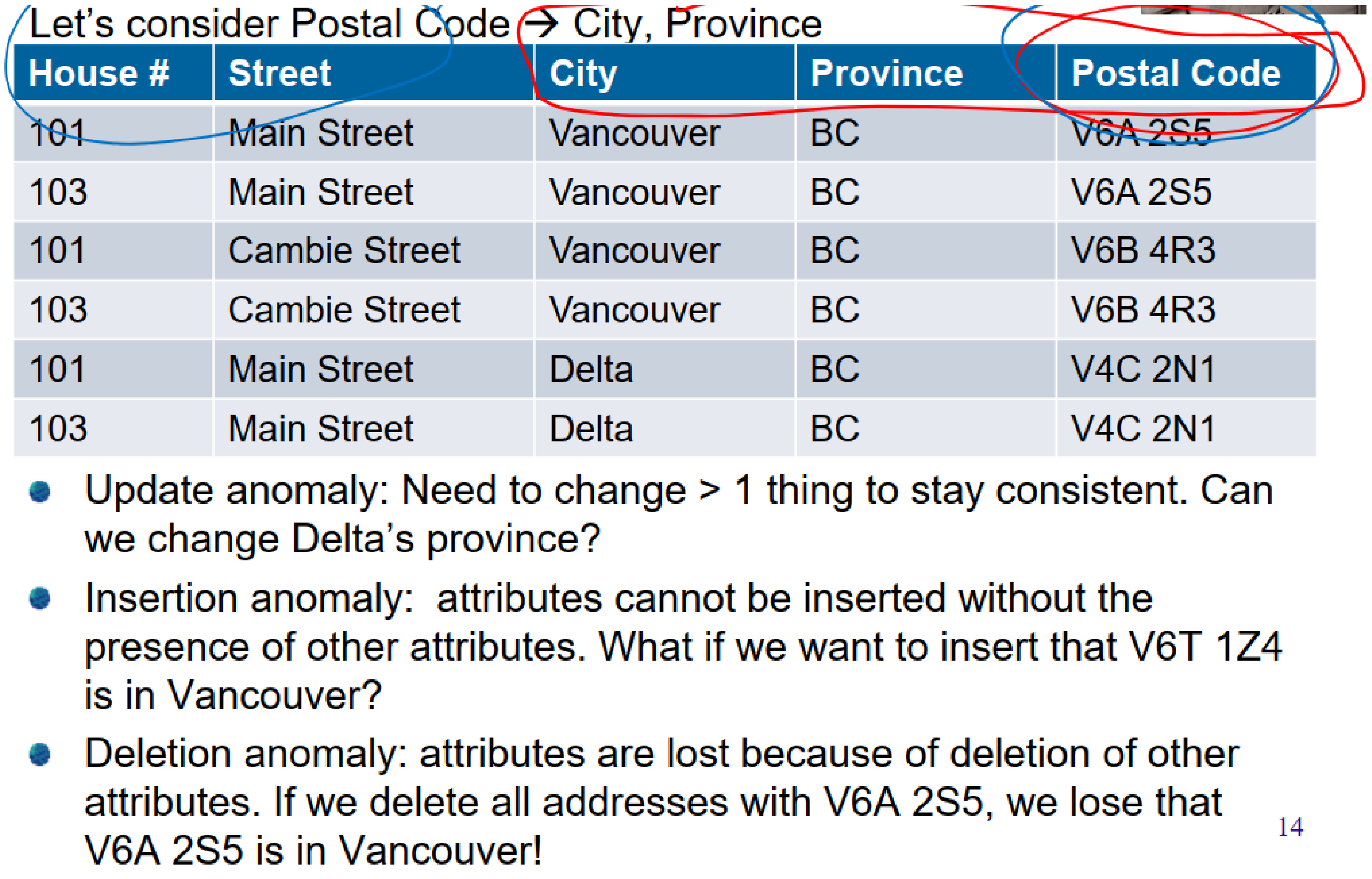
**SuperRentInfo**(customerID, customerName, email, startDate, endDate, city, confirmationNumber, branchName, typeName, paymentCode)

… with the following FDs:

1) customerID → customerName, email
2) confirmationNumber → customerID
3) branchName → city
4) confirmationNumber, branchName → typeName
5) confirmationNumber, branchName, typeName → startDate, endDate
6) paymentCode → customerID, confirmationNumber

## Task #1: Anomalies - solution

Recap on 3 types of anomalies:



Let's consider Postal Code → City, Province

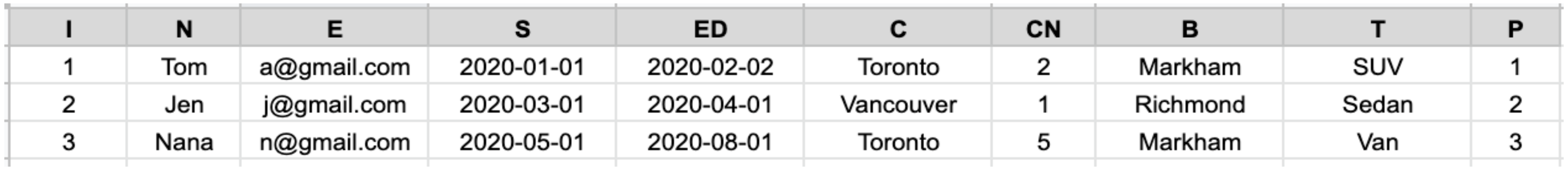
| House # | Street | City | Province | Postal Code |
|---|---|---|---|---|
| 101 | Main Street | Vancouver | BC | V6A 2S5 |
| 103 | Main Street | Vancouver | BC | V6A 2S5 |
| 101 | Cambie Street | Vancouver | BC | V6B 4R3 |
| 103 | Cambie Street | Vancouver | BC | V6B 4R3 |
| 101 | Main Street | Delta | BC | V4C 2N1 |
| 103 | Main Street | Delta | BC | V4C 2N1 |

- Update anomaly: Need to change > 1 thing to stay consistent. Can we change Delta's province?
- Insertion anomaly: attributes cannot be inserted without the presence of other attributes. What if we want to insert that V6T 1Z4 is in Vancouver?
- Deletion anomaly: attributes are lost because of deletion of other attributes. If we delete all addresses with V6A 2S5, we lose that V6A 2S5 is in Vancouver!

14

# Task #1: Anomalies - solution

Back to our task. Think about this example:



| I | N | E | S | ED | C | CN | B | T | P |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Tom | a@gmail.com | 2020-01-01 | 2020-02-02 | Toronto | 2 | Markham | SUV | 1 |
| 2 | Jen | j@gmail.com | 2020-03-01 | 2020-04-01 | Vancouver | 1 | Richmond | Sedan | 2 |
| 3 | Nana | n@gmail.com | 2020-05-01 | 2020-08-01 | Toronto | 5 | Markham | Van | 3 |

- **Insertion Anomaly**: Can you insert a customer Peter with his email address, but don't know what vehicle type is reserved for him?
    - The only way out is to introduce NULL values. However, NULLs are problematic; therefore, it is desirable to minimize their use.
- **Deletion Anomaly**: Suppose we delete Jen's customer information. Deleting information related to Jen forces us to accidentally lose information about our SuperRent branches and their location, as well as any reservations Jen made.

- **Update Anomaly**: Suppose the branch name for the branch in Toronto is updated from Markham to another name like York.  This means every record that contains that info in the above table must be updated; thus, a chain of updating redundancy needs to be handled.

## Task #2: Lossless-Join Decomposition

Consider the decomposition of the relation **SuperRentInfo** into

SI1(I, N, E, S, C, B) and

SI2(C, P, T, ED, CN, B)

Is this a lossy, or lossless-join, decomposition? Justify your answer.

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

*Lossless-Join: If we JOIN the X-part of r with the Y-part of r, the result is exactly r*

Reminder: in this tutorial, ”CN” is an independent attribute, not “C and N”. Same for ”ED”.

## Task #2: Lossless-Join Decomposition - Recap

*Criteria: the common attribute(s) have to be a (super)key in either X1 or X2.* [1]

$X1 \cap X2 \rightarrow X1$ *OR* $X1 \cap X2 \rightarrow X2$

*[1] https://byjus.com/gate/lossless-decomposition-in-dbms/*

## Task #2: Lossless-Join Decomposition - solution

Consider the common attributes between SI1(I, N, E, S, C, B) and SI2(C, P, T, ED, CN, B) (i.e., the intersection):

SI1 ∩ SI2 = { C, B }

We now check if that closure corresponds to SI1 or SI2, that is:

SI1 ∩ SI2 → SI1 **OR** SI1 ∩ SI2 → SI2

Let's check the closure by using the given FDs :

(C,B)+ = {C, B} // we can't use any FDs, we're stuck!

{C, B} does not include either SI1 or SI2; so, we conclude that this decomposition is lossy.

*Task 3 is similar. It will be left as your homework.*

## Task #4: Find Keys

Find **all** keys for **SuperRentInfo.**

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

## Task #4: Find Keys - review

**LHS-Both-RHS table method** to find the keys: (example from class)

"Find all the minimal keys for R(ABCD) where AB → C and C → BD."

**Step 1**: List all the attributes that **only** appear on the RHS of the FDs (i.e., things that can only be derived)

| Left | Middle | Right |
|---|---|---|
|  |  | D |

**Step 2**: List all the attributes that **only** ever appear on the LHS of a FD, or do not appear in a FD at all (i.e., things that have to be part of any key)

| Left | Middle | Right |
|---|---|---|
| A |  | D |

**Step 3**: List all the attributes that appear on the LHS and RHS of the FDs (i.e., things that are not obviously required and may help you derive new things)

| Left | Middle | Right |
|---|---|---|
| A | BC | D |

# Task #4: Find Keys - review

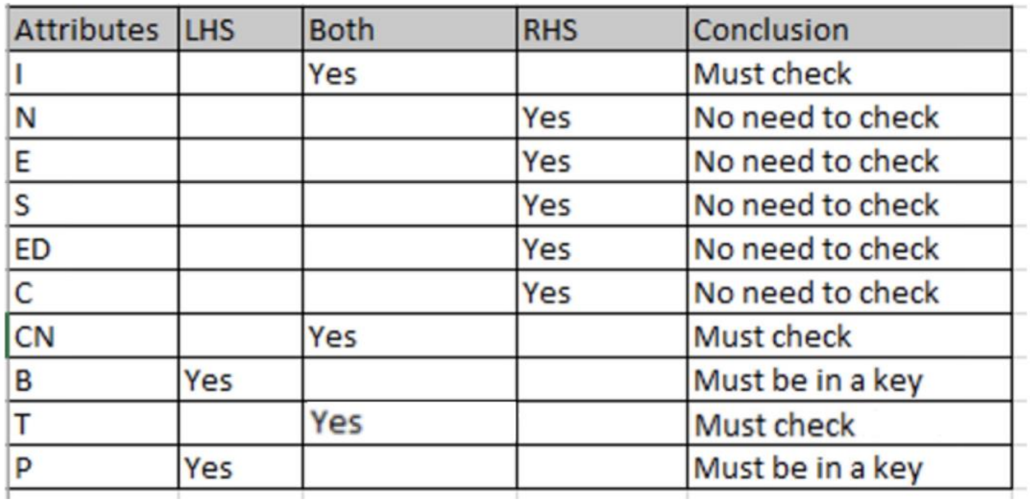
**Step 4**: Take the closure of the attributes in the left column. Are all of the attributes there? If so, you have found a minimal key. If not, start adding in attributes from the middle column to see if you can determine all the attributes of the relation.

{A}+ = {A} // not all attributes
{AB}+ = {ABCD} ← minimal key
{AC}+ = {ACBD} ← minimal key

| Left | Middle | Right |
|---|---|---|
| A | BC | D |

# Task #4: Find Keys - solution

We use the LHS-Both-RHS table to investigate:



| Attributes | LHS | Both | RHS | Conclusion |
|---|---|---|---|---|
| I | | Yes | | Must check |
| N | | | Yes | No need to check |
| E | | | Yes | No need to check |
| S | | | Yes | No need to check |
| ED | | | Yes | No need to check |
| C | | | Yes | No need to check |
| CN | | Yes | | Must check |
| B | Yes | | | Must be in a key |
| T | | Yes | | Must check |
| P | Yes | | | Must be in a key |

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

- Attributes B and P do not appear on the RHS and must be part of every key.
- Check the closure of { B, P } and see if you can get every attribute from it, using the FDs:
- {B,P}+ = { B, P, C, I, CN, T, S, ED, N, E } // It works! Note: if this does not get you all the attributes, then you would iteratively investigate LHS + "Both" – one by one – (i.e. {B, P, I}+ and {B, P, CN}+ and so on. )

Based on this, we claim that **BP** is the only key.

# Task #5: Lossless-Join BCNF Decomposition

Obtain a lossless-join, BCNF decomposition of **SuperRentInfo.**

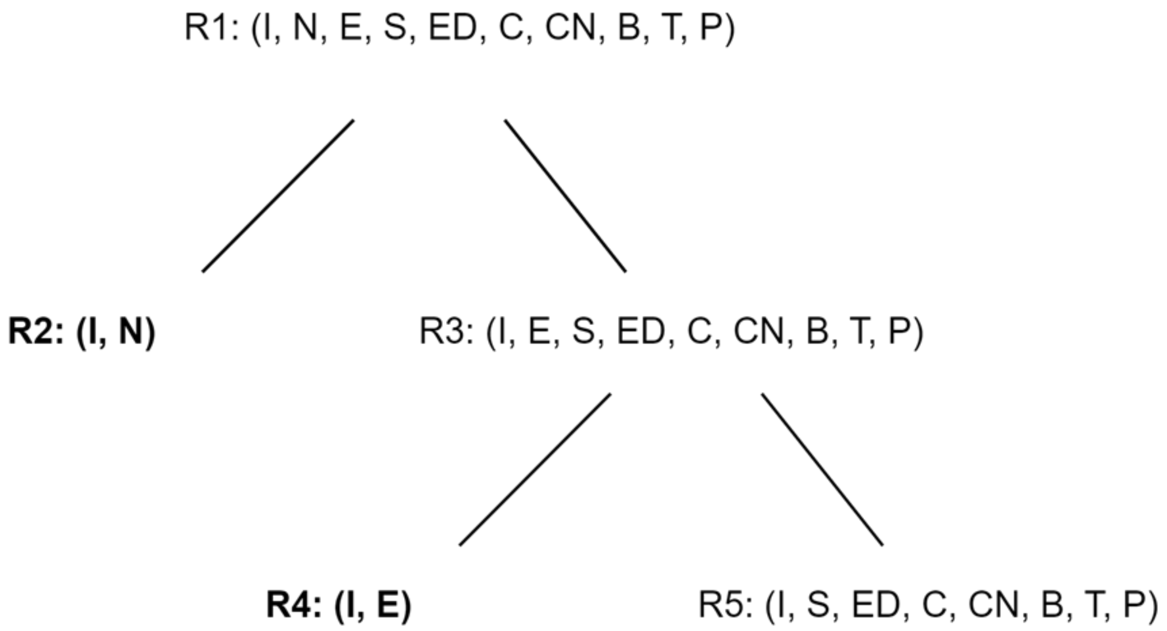
**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

## Task #5: Solution

- Currently, all FDs violate the BCNF condition. *(key is BP)*
- Pick any one of them, say FD 1: I → N, E. We will first split it up into I → N and I → E, then we can decompose the original relation with these 2 FDs separately.

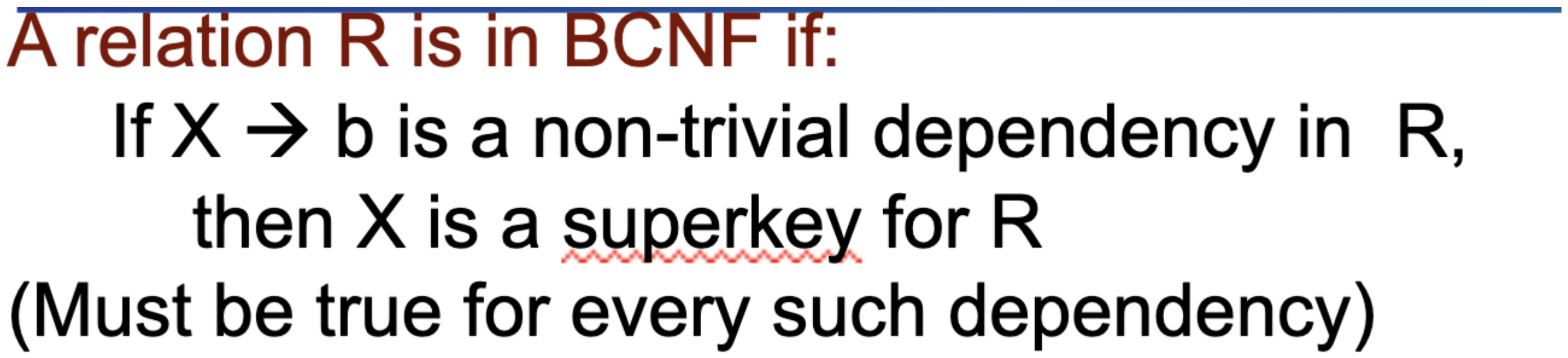

R1: (I, N, E, S, ED, C, CN, B, T, P)
**R2: (I, N)**
R3: (I, E, S, ED, C, CN, B, T, P)
**R4: (I, E)**
R5: (I, S, ED, C, CN, B, T, P)

Reminder, **BCNF MAY NOT BE DEPENDENCY PRESERVING.**

## Task #5: Solution

*Wait, but why we first split "*I → N, E" into I → N and I → E?

The definition of BCNF:



A relation R is in BCNF if:

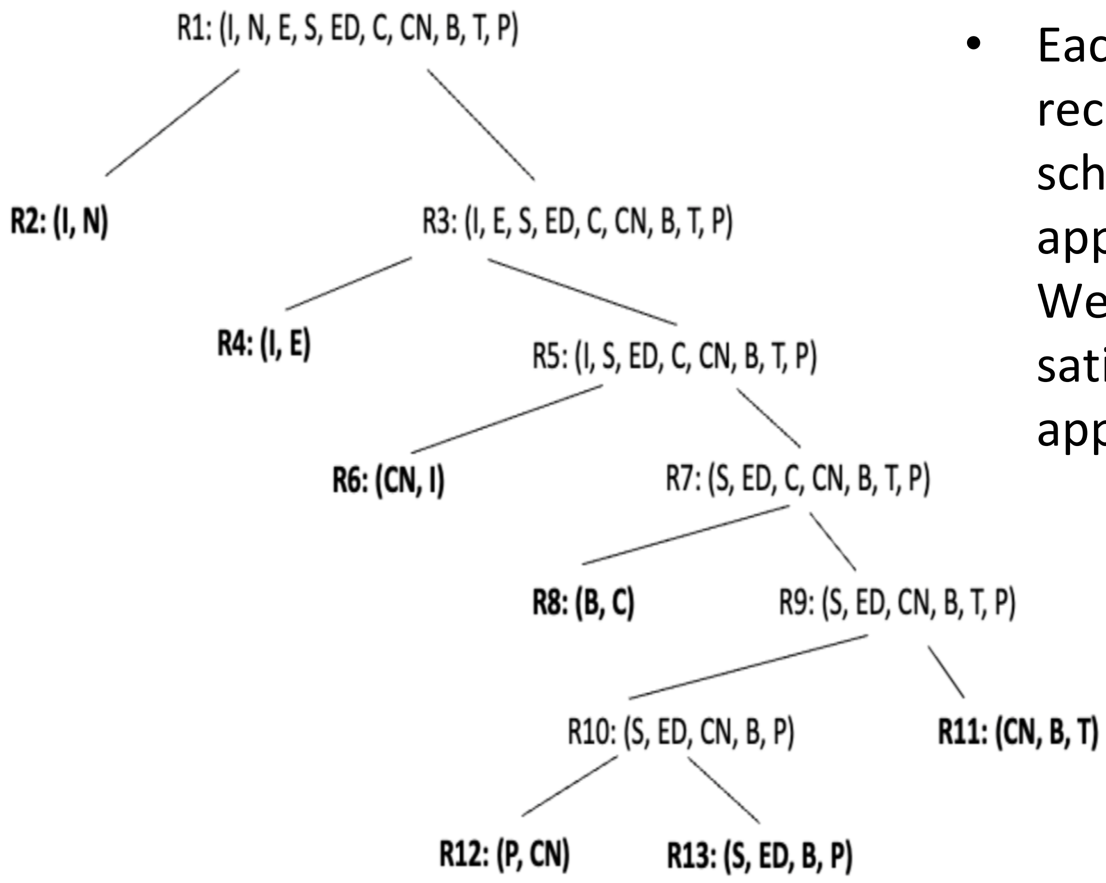
If X → b is a non-trivial dependency in R,

then X is a superkey for R

(Must be true for every such dependency)

Here, "X → b" means X (uppercase) can have multiple attributes, and b (lowercase) stands for a single attribute

# Task #5: Solution



- Each node is split into two children, recursively, whenever the relational schema at the node has an applicable FD that violates BCNF. We terminate when each leaf satisfies BCNF with respect to its applicable set of FDs.

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

Note:

- BCNF may **NOT** be dependency preserving.
- Different orders of the use of FDs may generate different results

## Task #6: Minimal cover

Find a minimal cover for this set of FDs.

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

Minimal cover G for a set of FDs F:

- Closure of F = closure of G (i.e., imply the same FDs)
- Right hand side of each FD in G is a single attribute
- If we delete an FD in G or delete attributes from an FD in G, the closure changes

# Task #6: Minimal cover - Recap

Recap: 3 steps to find a minimal cover

1. Put FDs in standard form (have only one attribute on RHS)
2. Minimize LHS of each FD
3. Delete Redundant FDs

# Task #6: Minimal cover - solution

Step 1: Split RHS attributes to obtain simpler FDs:

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. CN, B, T → S
7. CN, B, T → ED
8. P → I
9. P → CN

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

1) I → N, E
2) CN → I
3) B → C
4) CN, B → T
5) CN, B, T → S, ED
6) P → I, CN

## Task #6: Minimal cover - solution

Step 2: Can we delete any attributes from the LHS?

- Only **FDs 5, 6 and 7** are candidates for this because all other FDs have a single attribute on their LHS.

- FD 5: None of the attributes can be removed from the LHS because the closure fails to contain T when we drop an attribute.
  *For example, if we drop B, then $\{CN\}^+=\{CN, I, N, E\}$ and it doesn't contain B, so we cannot drop B in FD5. The same can be said if we drop CN.*

- FD 6: T can be dropped because FD 5 will compensate to generate T. Result: **CN, B -> S**
- FD 7: Same idea as above. We can remove T. Result: **CN, B -> ED**

(from 1st step)

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. **CN, B, T → S**
7. **CN, B, T → ED**
8. P → I
9. P → CN

## Task #6: Minimal cover - solution

Step 3: Is any FD redundant?

- Check one-by-one that no FD in the above set is redundant. For example, the closure of I, computed without using FD 1, does not contain N; therefore, FD 1 is not redundant.
- The only redundant FD is found to be P → I because the closure of P still contains I (using FD 9 and FD 3). We then eliminate that FD and come to our final result of a minimal cover:

(from 1st step)

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. CN, B, T → S
7. CN, B, T → ED
8. P → I
9. P → CN

(from 2nd step)

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. **CN, B -> S**
7. **CN, B -> ED**
8. P → I
9. P → CN

Minimal Cover

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. CN, B -> S
7. CN, B -> ED
8. P → CN

## Task #7 & #8: 3NF decomposition

- Obtain a lossless-join, dependency-preserving, 3NF decomposition of SuperRentInfo by using the **decomposition** method.
- Obtain a lossless-join, dependency-preserving, 3NF decomposition of SuperRentInfo by using the **synthesis** method.

**SuperRentInfo**(I, N, E, S, ED, C, CN, B, T, P)

… with the following FDs:

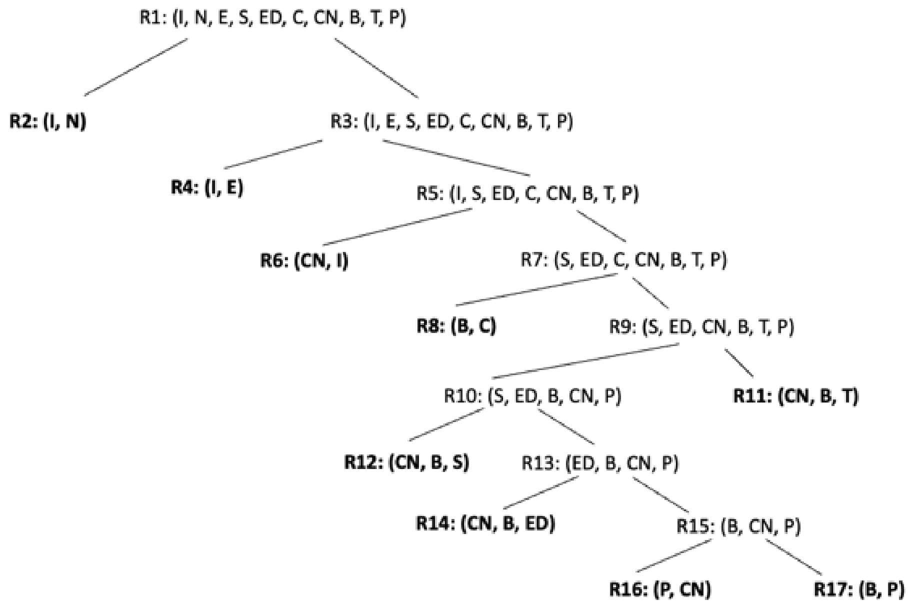
1) $I \rightarrow N, E$
2) $CN \rightarrow I$
3) $B \rightarrow C$
4) $CN, B \rightarrow T$
5) $CN, B, T \rightarrow S, ED$
6) $P \rightarrow I, CN$

# Task #7&8: decomposition method

- Intuition: first remove redundancy using lossless joins, then ensure that we maintain all functional dependencies
- Use results from task 5 (decomposition) and task 6 (minimal cover):

Minimal Cover

1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. CN, B -> S
7. CN, B -> ED
8. P → CN



- This is an example of a “perfect” scenario (rare case) where there is no need to add any relations to maintain this to be dependency preserving.
- If any of the minimal cover FDs are not preserved via decomposition, you have to add it at the very end to get your final set of 3NF abiding relations.

# Task #7&8: synthesis method

- We first create a relational schema corresponding to each FD in the minimal cover. This gives the database schema {INE, CN+I, BC, CN+BTS+ED, P+CN}—with each relational schema having the obvious FD and key. This set of relational schemas preserves all FDs, by construction/definition.
- If no relation in the set is a superkey of the original relation, then add a relation containing the key.
  We see that B,P is not in any of the above set of relations even though it's the key. No worries! Just add it in!

**Final Result:**

R1 = (<u>I</u>, N)
R2 = (<u>I</u>, E)
R3 = (<u>CN</u>, I)
R4 = (<u>B</u>, C)
R5 = (<u>CN, B</u>, T)
R6 = (<u>CN, B</u>, S)
R7 = (<u>CN, B</u>, ED)
R8 = (<u>P</u>, CN)
**R9 = (<u>B, P</u>)**

Minimal Cover
1. I → N
2. I → E
3. CN → I
4. B → C
5. CN, B → T
6. CN, B -> S
7. CN, B -> ED
8. P → CN

- Compare this to your answer for Q7. Are there any differences?