

A BCNF Normalisation Algorithm

Input:

- A *specification* containing:
 1. a relation schema, R , and
 2. a set of Functional Dependencies (FDs), F over R .
- An Entity-relationship Diagram (ERD) conforming to the specification.

Notes:

- $\text{schema}(R)$ is called the *universal set of attributes*.
- We assume the ERD satisfies the URSA.

Output: A database schema \mathbf{R} which is in BCNF with respect to F .

Notes:

- $\mathbf{R} = \{R_1, R_2, \dots, R_n\}$, where each R_i is a subset of $\text{schema}(\mathbf{R})$.
- The union of all $\text{schema}(R_i)$ is $\text{schema}(\mathbf{R})$.
- \mathbf{R} is called a **decomposition** of $\text{schema}(\mathbf{R})$.

[Home Page](#)[Title Page](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Page 3 of 14](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

Strategy

Step 1. Convert ERD into a database schema, \mathbf{S} .

Step 2. If any of the relation schemas in \mathbf{S} are *not* in BCNF with respect to F , then decompose them further, using the DECOMPOSE algorithm, given in this lecture.

Algorithm 1 (ERD-TO-BCNF(ERD, F))

1. **begin**

2. Convert ERD into a database schema $\mathbf{S} = \{S_1, S_2, \dots, S_m\}$;

3. **let** the output database schema \mathbf{R} be empty;

4. **for each** S_i in \mathbf{S} **do**

5. **if** S_i is in BCNF with respect to F

6. **add** S_i to \mathbf{R} ;

7. **else**

8. **merge** DECOMPOSE(S_i, F) and \mathbf{R} ;

9. **end for**;

10. **return** the decomposition \mathbf{R} ;

11. **end.**

[Home Page](#)

[Title Page](#)



Page 5 of 14

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

Example Entity Relationship Diagrams

ERD for Employee Database

ERD for Online-Bookshop Database

Convert these to database schemas each with a set of FDs.

[Home Page](#)

[Title Page](#)



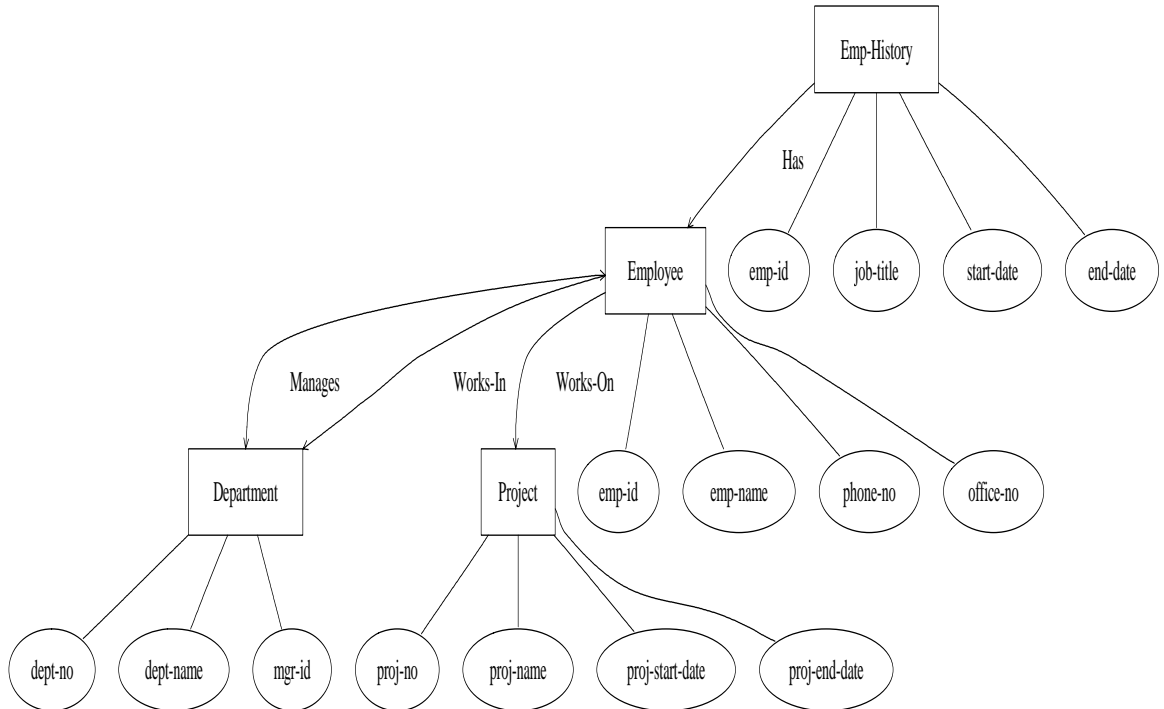
Page 6 of 14

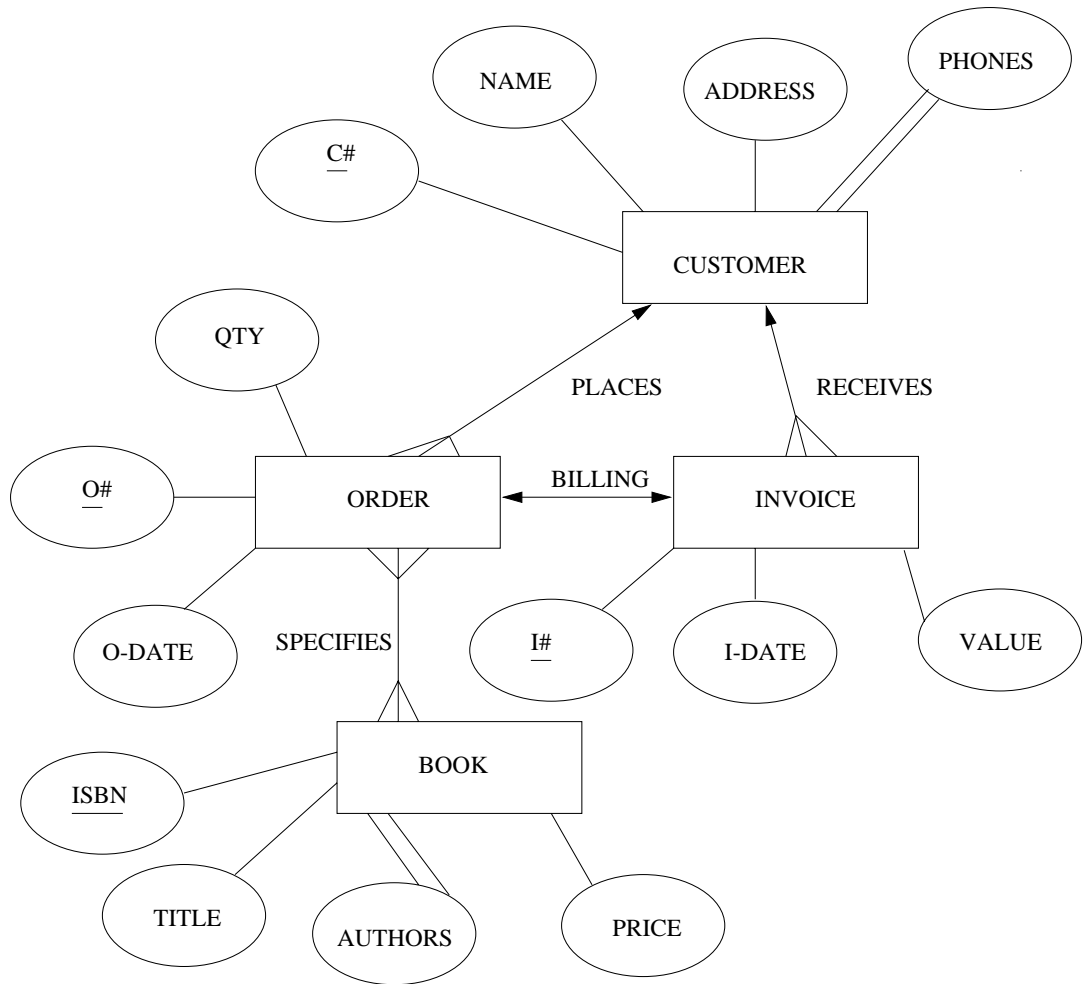
[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)





NEXT

Testing whether a database schema is in BCNF

Algorithm 2 (TEST-BCNF(R, F))

1. **begin**
2. **for each** $X \rightarrow Y$ **in** F **do**
3. **if** X **is not** a superkey with respect to F
4. **return** NO;
5. **end if**;
6. **end for**;
7. **return** YES;
8. **end.**

Decomposition Condition used by the Algorithm

- Let $\text{schema}(\text{PHONE}) = \{\text{cust-name}, \text{phone-num}, \text{phone-network}\}$.
- Let $F = \{\text{cust-name} \rightarrow \text{phone-num}, \text{phone-num} \rightarrow \text{phone-network}\}$.

Is PHONE in BCNF with respect to F ?

The Idea

So $\text{phone-num} \rightarrow \text{phone-network}$ **violates** BCNF.

- phone-num is the **left-hand** side of the violating FD and
- phone-network is the **right-hand** side of the violating FD.

Split PHONE into two relation schemas:

1. $R_1 = \text{NETWORK}$, with $\text{schema}(\text{NETWORK}) = \{\text{phone-num}, \text{phone-network}\}$, containing the attributes in the left and right hand sides of the **violating** FD, and F_1 containing $\text{phone-num} \rightarrow \text{phone-network}$.
2. $R_2 = \text{CUST}$, with $\text{schema}(\text{CUST}) = \{\text{cust-name}, \text{phone-num}\}$, containing the attributes in $\text{schema}(\text{PHONE})$ **except** those in the right-hand side of the **violating** FD (and not in its left-hand side), and F_2 containing $\text{cust-name} \rightarrow \text{phone-num}$.

Algorithm 3 (DECOMPOSE(R, F))

1. **begin**
2. **let** the output database schema **Out** be empty;
3. **if** TEST-BCNF(R, F) = YES **then**
4. **add** R to **Out**;
5. **else**
6. **let** $X \rightarrow Y$ in F be nontrivial (i.e. Y is **not** a subset of X)
 such that X is **not** a superkey with respect to F;
7. **let** R_1 be a relation schema,
 with schema(R_1) = X **merged** with Y;
8. **merge** DECOMPOSE(R_1 , F) and **Out**;
9. **let** R_2 be a relation schema,
 with schema(R_2) = schema(R) **except**
 the attributes that are in Y and **not** in X;
10. **merge** DECOMPOSE(R_2 , F) and **Out**;
11. **end if**
12. **return Out**;
13. **end.**

[Home Page](#)[Title Page](#)[◀](#) [▶](#)[◀](#) [▶](#)[Page 12 of 14](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

Result. $\text{DECOMPOSE}(R, F)$ returns a decomposition of $\text{schema}(R)$.

Result. The relations over $\text{DECOMPOSE}(R, F)$ can be joined naturally during query processing; this is known as the **lossless join** criterion.

An Example

Let $STUD$ be a relation schema, with $schema(STUD) = \{SNUM, POSTCODE, CITY, COUNTRY\}$, with FDs $\{SNUM \rightarrow POSTCODE, POSTCODE \rightarrow CITY, CITY \rightarrow COUNTRY\}$

- $CITY \rightarrow COUNTRY$ **violates** BCNF in $STUD$, so *decompose* $STUD$ into CC , with $schema(CC) = \{CITY, COUNTRY\}$, and $STUD1$, with $schema(STUD1) = \{SNUM, POSTCODE, CITY\}$
- CC is in BCNF while $POSTCODE \rightarrow CITY$ violates BCNF in $STUD1$, so *decompose* $STUD1$ into PC , with $schema(PC) = \{POSTCODE, CITY\}$, and $SINFO = \{SNUM, POSTCODE\}$.
- All the relation schemas in the database schema $\{CC, PC, SINFO\}$ is now in BCNF with respect to the specification.

Home Page

Title Page

◀ ▶

◀ ▶

Page 13 of 14

Go Back

Full Screen

Close

Quit

Another Example

Let \mathcal{R} be a relation schema, with $\text{schema}(\mathcal{R}) = \{C, T, H, R, S, G\}$.

- C stands for a course,
- T stands for a teacher,
- H stands for hour,
- R stands for room,
- S stands for student and
- G stands for grade.

An example set of FDs \mathcal{F} over \mathcal{R} :

1. $C \rightarrow T$,
2. $HR \rightarrow C$,
3. $HT \rightarrow R$,
4. $CS \rightarrow G$ and
5. $HS \rightarrow R$.

Decompose \mathcal{R} into BCNF.