

INFS1200 — Week 11 Notes

Table of contents

Normalisation and Relational Database Schema Design	2
Today's outline	2
Key concepts recap	2
Approaching normality	2
First Normal Form (1NF)	3
Full and partial functional dependencies	4
Second Normal Form (2NF)	4
Transitive dependency	4
Third Normal Form (3NF)	5
Boyce-Codd Normal Form (BCNF)	5
Question 8 — Partial vs. transitive dependency	6
Question 9 — Validating normal form	6
Question 10 — Validating normal form	7
Question 11 — Validating normal form	7
Question 12 — Validating normal form	7
Question 13 — BCNF and 3NF	7
Relational database schema design	8
Question 14 — Determining which FDs apply	8
Minimal cover	9
Question 15 — Minimal cover	10
3NF synthesis algorithm	10
Question 16 — 3NF synthesis	10
BCNF decomposition algorithm	11
Question 17 — BCNF decomposition	11
Question 18 — BCNF decomposition (with a lossy-join check)	12
Denormalisation	12
Summary	13
Applied Class 10: Normalisation, BCNF Decomposition and 3NF Synthesis	13
Section A — BCNF decomposition (top-down)	13

Section B — 3NF synthesis (bottom-up)	15
Case Study 9: Payroll System (BCNF Decomposition and 3NF Synthesis)	16
The corrected payroll schema	17
Section A — BCNF decomposition	17
Section B — 3NF synthesis	20

Normalisation and Relational Database Schema Design

Module 4, part 2. Continues 2026-04-27-database-design-guidelines-and-functional-dependencies¹ — now that we can identify FDs and keys, we use them to formally measure and improve schema quality via **normalisation**.

Today's outline

- Normalisation: 1NF, 2NF, 3NF, BCNF
- Relational database schema design: BCNF decomposition, minimal cover, 3NF synthesis

Key concepts recap

- **Superkey** — a set of attributes such that no two tuples share the same values for them (its closure covers every attribute in R).
- **(Candidate) key** — a *minimal* superkey (removing any attribute breaks the superkey property). Minimal — shortest; a relation can have several candidate keys, and every superkey contains at least one candidate key.
- **Primary key** — one candidate key, chosen as *the* key (only one per relation).
- **Prime attribute** — an attribute that belongs to *any* candidate key (not necessarily the primary key).
- **Non-prime attribute** — belongs to no candidate key.

Approaching normality

Functional dependencies reveal redundancy: in $R [A, B, C]$ with no FDs, there's no redundancy; but given $A \rightarrow B$, several tuples can share the same A value, and whenever they do, they're *forced* to repeat the same B value too — that repetition is the redundancy **normalisation** identifies and removes.

¹courses/infos1200/2026-04-27-database-design-guidelines-and-functional-dependencies.pdf

Normalisation is the process of taking a relational schema through a series of tests, using FDs and (candidate/primary) keys, to certify which **normal form** it satisfies. Schemas that fail a test are decomposed into smaller relations with better properties.

Normal forms overview

NF	Outcome	Test (for every non-trivial $X \rightarrow A$ in F)
1NF	Identifies non-atomic values	Relation has no multivalued attributes or nested relations
2NF	Identifies partial dependencies (removes some anomalies)	X is <i>not</i> a proper subset of a candidate key, or A is a prime attribute
3NF	Identifies partial + transitive dependencies (removes most anomalies)	X is a superkey, or A is a prime attribute
BCNF	Identifies <i>all</i> anomalies (at the cost of not preserving all FDs)	X is a superkey

BCNF 3NF 2NF 1NF — each normal form is strictly more restrictive than the last.

First Normal Form (1NF)

A relation schema is in 1NF if every attribute's domain contains only **atomic (simple, indivisible) values** — no set of values, no tuple of values (nested attributes), and no combination of both.

Non-1NF example — an `items` column holding a comma-separated list, or a nested `name = {firstName, familyName}` sub-structure, both violate 1NF. Normalizing to 1NF either:

- **repeats** the non-nested key columns per value (e.g. one row per (`customerName`, `orderNum`, `item`) combination) — introduces redundancy; or
- **flattens** into fixed columns (`item1`, `item2`, ...) — introduces a lack of flexibility (a hard cap on how many items one order can have, and `nulls` for orders with fewer items).

Neither is fully satisfactory — this motivates 2NF/3NF/BCNF, which address *how* to further decompose an already-1NF relation.

Full and partial functional dependencies

- $X \rightarrow Y$ is a **full functional dependency** if removing *any* attribute from X breaks the dependency (no proper subset of X determines Y).
- $X \rightarrow Y$ is a **partial dependency** if *some* attribute can be removed from X and the FD still holds.

Example — Address [houseNum, street, postcode, state, value], $F = \{\text{houseNum, street, postcode} \rightarrow \{\text{state, value}\}, \text{postcode} \rightarrow \text{state}\}$:

- $\{\text{houseNum, street, postcode}\} \rightarrow \text{value}$ is a **full** dependency (no subset of the LHS determines value).
- $\{\text{houseNum, street, postcode}\} \rightarrow \text{state}$ is a **partial** dependency, since $\text{postcode} \rightarrow \text{state}$ alone already suffices.

Second Normal Form (2NF)

A relation schema R is in 2NF if every **non-prime** attribute is **fully** functionally dependent on the primary key — informally, *no partial dependency*.

In the Address example: the (only) key is {houseNum, street, postcode}; state is non-prime and only partially dependent on the key (via $\text{postcode} \rightarrow \text{state}$) — this violates 2NF, so Address is **not** in 2NF.

Fixing it: decompose into Address [houseNum, street, postcode, value] and Postcodes [postcode, state] (Address.postcode $\rightarrow$ Postcodes.postcode). 2NF identifies anomalies caused by *partial* dependencies — but **not** anomalies caused by *transitive* dependencies, covered next.

2NF doesn't fix everything: Employee [ID, name, level, salary] with $\text{level} \rightarrow \text{salary}$ doesn't violate 2NF at all (level isn't even a *proper subset* of the key {ID} — it's a whole non-key attribute), so Employee is in 2NF — yet it still has all the same anomalies from before (updating one developer's salary is still inconsistent, deleting the only "Administration" employee still loses that level's salary, etc.). 2NF alone isn't enough.

Transitive dependency

$X \rightarrow Y$ in R is a **transitive dependency** if some attribute set Z exists such that Z is neither a candidate key nor a subset of one, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

Example — Employee [ID, name, level, salary], $F = \{\text{level} \rightarrow \text{salary}\}$: consider $\text{ID} \rightarrow \text{salary}$. level is neither a candidate key nor a subset of one; $\text{ID} \rightarrow \text{level}$ and $\text{level} \rightarrow \text{salary}$ both hold; therefore $\text{ID} \rightarrow \text{salary}$ is a transitive dependency (salary depends on ID *only through* the intermediate level).

Third Normal Form (3NF)

A relation schema R with FD set F is in 3NF iff, for every non-trivial FD $X \rightarrow A$ in F : X is a superkey, **or** A is a prime attribute. Equivalently — for any non-trivial $X \rightarrow A$ where A is non-prime, X must be a superkey.

In the `Employee` example: `level` $\rightarrow$ `salary` — `level` is not a superkey, and `salary` is not a prime attribute — **violates 3NF**. Fixing it: decompose into `StaffAppointment` [`ID`, `name`, `level`] and `StaffIncome` [`level`, `salary`]. Most 3NF relations are anomaly-free — but not *all* of them, as the next example shows.

3NF still isn't always enough — `Teach` [`studentID`, `courseID`, `lecturer`], $F = \{\{\text{studentID}, \text{courseID}\} \rightarrow \text{lecturer}, \text{lecturer} \rightarrow \text{courseID}\}$ (keys: $\{\text{studentID}, \text{courseID}\}, \{\text{studentID}, \text{lecturer}\}$). `Teach` is in 3NF (`lecturer` $\rightarrow$ `courseID`'s RHS, `courseID`, is a prime attribute — so it doesn't violate 3NF even though `lecturer` isn't a superkey). Yet it still has anomalies:

- **Deletion anomaly:** deleting the only row for a given (`studentID`, `courseID`) pair can delete the *only* record of which lecturer teaches that course.
- **Insertion anomaly:** can't record which lecturer teaches a course that currently has no enrolled students.

Boyce-Codd Normal Form (BCNF)

A relation schema R with FD set F is in BCNF iff, for every non-trivial FD $X \rightarrow A$ in F : X is a superkey. Informally: whenever a set of attributes determines another attribute, it must determine *all* attributes of R .

In `Teach`: `lecturer` $\rightarrow$ `courseID` — `lecturer` is **not** a superkey — violates BCNF, so `Teach` is **not** in BCNF (even though it is in 3NF).

The catch: BCNF can lose dependencies

Decomposing `Teach` to fix the BCNF violation: split into [`lecturer`, `courseID`] (from `lecturer` $\rightarrow$ `courseID`) and [`studentID`, `lecturer`]. Each piece individually satisfies its own local FDs and is in BCNF. But now rejoin them:

lecturer	courseID	studentID	lecturer
John	INFS1200	1234	John
Jane	INFS1200	1234	Jane

↓ join ↓

studentID	courseID	lecturer
1234	INFS1200	John

The rejoined table **violates** the original FD $\{\text{studentID}, \text{courseID}\} \rightarrow \text{lecturer}$ (both rows share $\{1234, \text{INFS1200}\}$ but disagree on `lecturer`) — even though neither decomposed piece violated *any* FD on its own. **Decomposing into BCNF can fail to preserve all the original FDs.** A **dependency-preserving** decomposition is one where, if R is split into $R_1, \dots, R_n$ with FD sets $F_1, \dots, F_n$ (each F_i containing the FDs of F whose attributes fall entirely within R_i), then $(F_1 \dots F_n) = F$ — i.e. no FD information is lost by the split.

BCNF vs 3NF trade-off:

	BCNF	3NF
Removes all anomalies		
Preserves all FDs		

Most organisations aim for one or the other depending on which guarantee matters more for their use case.

Question 8 — Partial vs. transitive dependency

What's the difference between partial and transitive dependency?

Solution

Partial dependency: an attribute depends on only a *subpart* of the primary key. **Normalizing to 2NF** solves this. **Transitive dependency:** a non-prime attribute depends on *other non-prime attributes* (rather than directly on the key). **Normalizing to 3NF** solves this.

Question 9 — Validating normal form

Given $R [A, B, C, D, E, F]$ with $\{A, B\} \rightarrow \{C, D, E\}$, $C \rightarrow F$, $E \rightarrow \{A, B\}$ — what is the highest normal form?

Solution

2NF. Keys: $\{A, B\}$, $\{E\}$. No partial dependencies exist (both $\{A, B\} \rightarrow \{C, D, E\}$ and $E \rightarrow \{A, B\}$ have a *whole* key as LHS), so 2NF holds. But $C \rightarrow F$: C is not a superkey, and F is non-prime — violates both 3NF and BCNF.

Question 10 — Validating normal form

Given $R [A, B, C, D]$ with $\{A, B\} \rightarrow \{C, D\}$, $\{C, D\} \rightarrow A$, $D \rightarrow B$ — what is the highest normal form?

Solution

3NF. Keys: $\{A, B\}$, $\{C, D\}$, $\{A, D\}$ — every attribute is prime. Since every attribute is prime, 2NF and 3NF automatically hold (every FD's RHS is a prime attribute). But $D \rightarrow B$: D is not a superkey — violates BCNF.

Question 11 — Validating normal form

Given $R [A, B, C, D, E]$ with $B \rightarrow \{C, D\}$, $A \rightarrow E$ — what is the highest normal form?

Solution

1NF. Key: $\{A, B\}$. Both A and B are proper subsets of the key — $A \rightarrow E$ and $B \rightarrow \{C, D\}$ are both **partial dependencies**, violating 2NF.

Question 12 — Validating normal form

Given $R [A, B, C, D, E]$ with $A \rightarrow \{B, C, D, E\}$, $E \rightarrow A$ — what is the highest normal form?

Solution

BCNF. Keys: $\{A\}$, $\{E\}$. The LHS of every FD (A and E) is itself a superkey — satisfies BCNF, the strictest test.

Question 13 — BCNF and 3NF

Given $R [A, B, C, D]$ with $\{A, C, D\} \rightarrow B$, $\{A, C\} \rightarrow D$, $D \rightarrow C$, $\{A, C\} \rightarrow B$ — which is true?

A. Neither BCNF nor 3NF B. BCNF but not 3NF C. 3NF but not BCNF D. Both BCNF and 3NF

i Solution

C. Keys: $\{A, C\}$, $\{A, D\}$. $D \rightarrow C$: D is not a superkey, so **BCNF is violated**. But C is a prime attribute (appears in key $\{A, C\}$), so this same FD does **not** violate 3NF — R is in 3NF but not BCNF.

Relational database schema design

Two complementary algorithms turn a *universal relation* (all attributes lumped together, plus a set of FDs) into a well-designed multi-relation schema:

1. **Decomposition** (top-down): break the universal relation apart to reach **lossless-join, anomaly-free BCNF** schemas.
2. **Synthesis** (bottom-up): build up from individual attributes to reach **lossless-join, dependency-preserving 3NF** schemas.

flowchart LR

```
U["Universal Relation + FDs"] -->|decomposition| RDB["Relational DB schema"]
E["EER Diagram (from UoD)"] -->|mapping| RDB
A["All attributes + FDs"] -->|synthesis| RDB
```

Determining which FDs apply to a decomposed relation

For a decomposed relation S and an FD $X \rightarrow Y$ from the original F : $X \rightarrow Y$ holds in S if $X \rightarrow Y$ holds in S 's attributes **and** $Y \subseteq X$ (computed using *all* of F , even attributes not in S).

Question 14 — Determining which FDs apply

Given $R[A, B, C, D, E]$ with $\{A, B\} \rightarrow C$, $\{B, C\} \rightarrow D$, $\{C, D\} \rightarrow E$, $\{D, E\} \rightarrow A$, $\{A, E\} \rightarrow B$ — which FDs hold in $S[A, B, C, D]$?

A. $A \rightarrow B$ B. $\{A, B\} \rightarrow E$ C. $\{A, E\} \rightarrow B$ D. $\{B, C, D\} \rightarrow A$ E. None of the above

i Solution

D. $\{B, C, D\}$ (computed against the *full* F , using E as an intermediate even though it's not in S): $\{B, C\} \rightarrow D$ gives nothing new; $\{C, D\} \rightarrow E$ adds $E \rightarrow \{B, C, D, E\}$; $\{D, E\} \rightarrow A$ adds $A \rightarrow \{A, B, C, D, E\}$ — covers everything, and $A \subseteq S$, so $\{B, C, D\} \rightarrow A$ holds in S . A fails ($A = \{A\}$ alone — doesn't even reach B). B and C both fail because their RHS, E , isn't even an attribute of S — an FD “holding in S ” requires both sides to be within S 's attributes.

Minimal cover

To decompose into 3NF (synthesis) — or just to simplify reasoning about BCNF — we first reduce F to a **minimal cover** G : an equivalent, “as small as possible” set of FDs. G is a minimal cover for F iff:

- $F = G$ (equivalent implied FDs);
- every FD in G has a single attribute on its RHS;
- deleting any FD in G , or any attribute from any FD’s LHS, changes G (i.e. every FD, and every LHS attribute, is *necessary*).

Finding a minimal cover — 3 steps

1. **RHS simplification** — split every FD so its RHS has exactly one attribute ($X \rightarrow \{Y, Z\}$ becomes $X \rightarrow Y$ and $X \rightarrow Z$).
2. **LHS simplification** — for each FD $X \rightarrow A$ where X has multiple attributes including some B : if $X = (X - \{B\})$ (i.e. B is redundant in the LHS), replace it with $(X - \{B\}) \rightarrow A$.
3. **FD set simplification** — delete any FD $X \rightarrow A$ if X (computed *without* that FD) still includes A — i.e. the FD is implied by the rest of F and can be dropped.

Worked example — $R [A,B,C,D,E,F,G,H]$, $F = \{A \rightarrow B, \{A,B,C,D\} \rightarrow E, \{E,F\} \rightarrow G, \{E,F\} \rightarrow H, \{A,C,D,F\} \rightarrow \{E,G\}\}$:

Step 1: RHS split	Step 2: LHS simplify	Step 3: delete redundant
$A \rightarrow B$	$A \rightarrow B$	$A \rightarrow B$
$\{A,B,C,D\} \rightarrow E$	$\{A,C,D\} \rightarrow E$ (B redundant, since $A \rightarrow B$)	$\{A,C,D\} \rightarrow E$
$\{E,F\} \rightarrow G$	$\{E,F\} \rightarrow G$	$\{E,F\} \rightarrow G$
$\{E,F\} \rightarrow H$	$\{E,F\} \rightarrow H$	$\{E,F\} \rightarrow H$
$\{A,C,D,F\} \rightarrow E$	$\{A,C,D,F\} \rightarrow E$	(<i>dropped</i> — $\{A,C,D,F\}$ already includes E without this FD)
$\{A,C,D,F\} \rightarrow G$	$\{A,C,D,F\} \rightarrow G$	(<i>dropped</i> — same reason)

Minimal cover: $\{A \rightarrow B, \{A,C,D\} \rightarrow E, \{E,F\} \rightarrow G, \{E,F\} \rightarrow H\}$ (often written with the last two combined as $\{E,F\} \rightarrow \{G,H\}$).

Question 15 — Minimal cover

Given $R [A, B, C, D, E, F]$ with $\{D, E, F\} \rightarrow C$, $\{A, B\} \rightarrow \{D, C\}$, $D \rightarrow F$ — which is a minimal cover?

- A. $\{D, E, F\} \rightarrow C$, $\{A, B\} \rightarrow \{D, C\}$, $D \rightarrow F$ B. $\{D, E, F\} \rightarrow C$, $\{A, B\} \rightarrow D$, $D \rightarrow F$ C. $\{D, E\} \rightarrow C$, $\{A, B\} \rightarrow D$, $\{A, B\} \rightarrow C$, $D \rightarrow F$ D. $\{D, E\} \rightarrow C$, $\{A, B\} \rightarrow \{C, D\}$

Solution

C. Step 1 (RHS split): $\{D, E, F\} \rightarrow C$, $\{A, B\} \rightarrow D$, $\{A, B\} \rightarrow C$, $D \rightarrow F$. Step 2 (LHS simplify): $\{D, E, F\} \rightarrow C$ simplifies to $\{D, E\} \rightarrow C$, since $D \rightarrow F$ already makes F redundant in that LHS ($\{D, E\}$ already includes F via $D \rightarrow F$, so $\{D, E, F\} = \{D, E\}$). Step 3: nothing more to remove — all four remaining FDs are necessary. A keeps the redundant F in the first FD's LHS. B drops the necessary $\{A, B\} \rightarrow C$ FD entirely. D drops the necessary $D \rightarrow F$ FD.

3NF synthesis algorithm

```
S := ;
Compute a minimal cover G of F;
Combine all FDs in G with the same LHS into one;
For each  $X \rightarrow Y$  in G:
    if no relation in S contains  $X \rightarrow Y$ :
        add a relation with schema  $X \rightarrow Y$  to S;
if any candidate key is missing from the relations:
    add a relation containing all prime attributes;
Eliminate redundant relations (one whose attributes are a
subset of another relation already in S).
```

Example — $R [A, B, C, D, E], F = \{\{A, B\} \rightarrow C, C \rightarrow D\}$ (already minimal); key: $\{A, B, E\}$. Synthesis: $R_1 [A, B, C]$ (from $\{A, B\} \rightarrow C$), $R_2 [C, D]$ (from $C \rightarrow D$); the key $\{A, B, E\}$ isn't contained in either, so add $R_3 [A, B, E]$. None of $R_1/R_2/R_3$ is a subset of another — final answer: $R_1 [A, B, C]$, $R_2 [C, D]$, $R_3 [A, B, E]$.

Question 16 — 3NF synthesis

Given the minimal cover $F = \{\{A, C\} \rightarrow E, \{B, D\} \rightarrow A, A \rightarrow B, E \rightarrow \{C, F\}\}$ for $R [A, B, C, D, E, F]$ — which is a correct 3NF synthesis?

- A. $R_1 [A, C, E]$, $R_2 [D, B, A]$, $R_3 [A, B]$, $R_4 [E, C, F]$ B. $R_1 [A, C, E]$, $R_2 [A, B, D]$, $R_3 [A, B]$, $R_4 [E, C, F]$, $R_5 [A, C, D]$ C. $R_1 [A, C, E]$, $R_2 [B, D, A]$, $R_3 [E, C, F]$, $R_4 [A, C, D]$ D. $R_1 [E, C, F]$, $R_5 [A, B, C, D, E]$

Solution

D. Candidate keys are $\{A, C, D\}$, $\{A, D, E\}$, $\{B, C, D\}$, $\{B, D, E\}$ — note F never appears in any key. Synthesising one relation per FD gives $[A, C, E]$ ($\{A, C\} \rightarrow E$), $[A, B, D]$ ($\{B, D\} \rightarrow A$), $[A, B]$ ($A \rightarrow B$), $[E, C, F]$ ($E \rightarrow \{C, F\}$). $[A, B]$ is a subset of $[A, B, D]$ — redundant, drop it. None of the remaining three relations contains a *full* candidate key (all four keys include D , but neither $[A, C, E]$ nor $[E, C, F]$ has D , and $[A, B, D]$ is missing C/E), so add R_5 with all **prime** attributes: $\{A, B, C, D, E\}$ (the union of all candidate keys). Now $[A, C, E]$ and $[A, B, D]$ are both subsets of R_5 $[A, B, C, D, E]$ — redundant, drop both. $[E, C, F]$ is **not** a subset of R_5 (F isn't in it) — keep. Final: $[E, C, F]$ and R_5 $[A, B, C, D, E]$.

BCNF decomposition algorithm

```
D := {R};
while (some relation Q in D is not in BCNF):
    find an FD  $X \rightarrow Y$  in Q that violates BCNF;
    replace Q in D with  $Q_1 [Q - Y]$  and  $Q_2 [X \ Y]$ ;
```

(The split point: Q_1 keeps everything except the “extra” attributes Y ; Q_2 becomes the offending FD’s own little BCNF-satisfying table, since X is now trivially a key for it.) The algorithm terminates because every 2-attribute relation is automatically in BCNF (with 0 or 1 non-trivial FDs, the LHS is always a key). Results can vary depending on which order violating FDs are processed in — that’s fine, multiple correct decompositions can exist.

Worked example — $R [A, B, C, D]$, $F = \{B \rightarrow C, D \rightarrow A\}$: keys — $\{B, D\} = \{A, B, C, D\}$, the only key. $B \rightarrow C$ violates BCNF (B isn’t a key) $\rightarrow$ split into $R_1 [A, B, D]$, $R_2 [B, C]$. $D \rightarrow A$ now violates BCNF in R_1 (D isn’t a key for R_1) $\rightarrow$ split into $R_3 [B, D]$, $R_4 [A, D]$. **Final:** $R_2 [B, C]$, $R_3 [B, D]$, $R_4 [A, D]$.

Implicit FDs matter: when checking whether a relation is in BCNF during decomposition, you must also check *implied* FDs, not just the explicitly given ones — e.g. given $A \rightarrow B$ and $B \rightarrow C$, the implicit $A \rightarrow C$ might be the one that actually violates BCNF in a sub-relation, even though it was never written down explicitly.

Question 17 — BCNF decomposition

Given $R [A, B, C, D, E]$ with $\{A, D\} \rightarrow B$, $C \rightarrow \{D, E\}$, $A \rightarrow E$ — which is a correct BCNF decomposition tree?

i Solution

The correct order is: apply $\{A,D\} \rightarrow B$ first (splitting off R1 $[A,B,D]$, leaving R2 $[A,C,D,E]$), **then** apply $C \rightarrow \{D,E\}$ to R2 (splitting it into R3 $[C,D,E]$ and R4 $[A,C]$). **Final:** R1 $[A,B,D]$, R3 $[C,D,E]$, R4 $[A,C]$.

Common mistakes to watch for: writing R3 as $[C,E]$ instead of the full $[C,D,E]$ (dropping an attribute of the violating FD's RHS); writing the intermediate relation as $[A,B,C,E]$ instead of the correct $[A,C,D,E]$ (forgetting $A \rightarrow E$ only removes B, not D, from the first split); or decomposing R2 further using $A \rightarrow E$ when A is already a key for R2 at that point (no violation left to fix).

Question 18 — BCNF decomposition (with a lossy-join check)

Given $R [A, B, C, D]$ with $A \rightarrow B$, $C \rightarrow D$, $\{A,D\} \rightarrow C$, $\{B,C\} \rightarrow A$ — which is a lossless-join BCNF decomposition?

- A. $\{\{A,B\}, \{A,C\}, \{B,D\}\}$ B. $\{\{A,B\}, \{A,C\}, \{C,D\}\}$ C. $\{\{A,B\}, \{A,C\}, \{B,C,D\}\}$ D. All of the above E. None of the above

i Solution

B. Keys: $\{A,D\}$, $\{A,C\}$, $\{B,C\}$ (all three give closure $\{A,B,C,D\}$). $A \rightarrow B$ violates BCNF $\rightarrow$ split into R1 $[A,C,D]$, R2 $[A,B]$. $C \rightarrow D$ violates BCNF in R1 $\rightarrow$ split into R3 $[A,C]$, R4 $[C,D]$. **Final:** $\{\{A,B\}, \{A,C\}, \{C,D\}\}$.

Why not A ($\{\{A,B\}, \{A,C\}, \{B,D\}\}$)? Concretely: take R rows $(1,2,5,6)$, $(1,2,3,7)$, $(8,2,9,4)$. Decomposing per option A and rejoining produces two *extra* rows not in the original R (e.g. $(1,2,3,4)$ appears in the rejoined result despite never being an actual tuple of R) — this is a **lossy join**, since $\{B,D\}$ isn't actually a valid join key here ($B=2$ recurs across all three original rows with different D values, so joining on B alone reintroduces spurious combinations).

Denormalisation

Since anomalies come from redundancy, it's tempting to decompose as aggressively as possible — but heavily decomposed schemas need more JOINS to answer queries. **Denormalisation** is the *deliberate*, controlled process of relaxing a normal form to improve query performance: fewer joins (faster queries), fewer foreign keys (less storage/maintenance overhead) — useful for analytical workloads or when specific frequent queries need pre-joined results. It must be a controlled, deliberate trade-off, not an accident of poor initial design.

Summary

You should now be able to test a relation schema against 1NF/2NF/3NF/ BCNF given a set of FDs, decompose a universal relation into lossless-join anomaly-free BCNF relations (top-down), and compute a minimal cover to synthesise a lossless-join dependency-preserving 3NF schema (bottom-up). This completes Module 4. Next module: Database Security. See [week11-tutorial-applied-class-10-normalisation-bcnf-3nf-synthesis²](#) and [week11-tutorial-case-study-9-payroll-system³](#) for practice.

Reading: Elmasri & Navathe Chapters 14 (up to 14.6) and 15 (up to 15.5).

Applied Class 10: Normalisation, BCNF Decomposition and 3NF Synthesis

Practice for [2026-05-04-normalisation-and-relational-database-schema-design⁴](#). Note: “consider the FDs in the order provided” is a *default* guideline — only FDs that actually **violate** BCNF get split on; other correct decompositions can exist depending on split order.

Section A — BCNF decomposition (top-down)

Question A.1

$R [A, B, C, D, E]$ with $\{A\} \rightarrow \{C\}$, $\{A, B\} \rightarrow \{D, E\}$. Decompose into BCNF.

Working

$\{A, B\} = \{A, B, C, D, E\}$ — $\{A, B\}$ is the only candidate key.

$\{A\} \rightarrow \{C\}$ violates BCNF (A isn't a superkey) $\rightarrow$ split into $R_1 [A, C]$ and $R_2 [A, B, D, E]$. Within R_2 , $\{A, B\}$ is a key (it already was for the whole relation), so $\{A, B\} \rightarrow \{D, E\}$ doesn't violate BCNF there — no further splitting needed.

Final answer: $R_1 [A, C]$ ($\{A\} \rightarrow \{C\}$), $R_2 [A, B, D, E]$ ($\{A, B\} \rightarrow \{D, E\}$).

Question A.2

$R [A, B, C, D, E]$ with $\{A\} \rightarrow \{B\}$, $\{C\} \rightarrow \{D, E\}$. Decompose into BCNF.

²[courses/infs1200/week11-tutorial-applied-class-10-normalisation-bcnf-3nf-synthesis.pdf](#)

³[courses/infs1200/week11-tutorial-case-study-9-payroll-system.pdf](#)

⁴[courses/infs1200/2026-05-04-normalisation-and-relational-database-schema-design.pdf](#)

Working

$\{A, C\} = \{A, B, C, D, E\}$ — $\{A, C\}$ is the only candidate key.

$\{A\} \rightarrow \{B\}$ violates BCNF (A alone isn't a superkey) $\rightarrow$ split into $R_1 [A, B]$ and $R_2 [A, C, D, E]$. In R_2 , $\{C\} \rightarrow \{D, E\}$ violates BCNF (C alone isn't a superkey for R_2) $\rightarrow$ split into $R_3 [C, D, E]$ and $R_4 [A, C]$.

Final answer: $R_1 [A, B]$, $R_3 [C, D, E]$, $R_4 [A, C]$ (no non-trivial FD).

Question A.3

$R [A, B, C, D, E, F]$ with $\{A\} \rightarrow \{B, C\}$, $\{C\} \rightarrow \{D, E\}$, $\{E\} \rightarrow \{F\}$. Decompose into BCNF.

Working

$\{A\} = \{A, B, C, D, E, F\}$ — A is the only candidate key. $A \rightarrow \{B, C\}$ does **not** violate BCNF here, since A *is* the key — so we skip it and look for an FD whose LHS genuinely isn't a superkey.

$\{C\} \rightarrow \{D, E\}$ violates BCNF (C isn't a superkey) $\rightarrow$ split into $R_1 [C, D, E]$ and $R_2 [A, B, C, F]$. Within R_2 , the *explicit* FDs don't obviously violate BCNF — but the **implicit** FD $C \rightarrow F$ (derivable transitively via $C \rightarrow E$ and $E \rightarrow F$) does (C isn't a superkey for R_2 either) $\rightarrow$ split R_2 into $R_3 [C, F]$ and $R_4 [A, B, C]$.

Final answer: $R_1 [C, D, E]$ ($\{C\} \rightarrow \{D, E\}$), $R_3 [C, F]$ ($\{C\} \rightarrow \{F\}$, implicit), $R_4 [A, B, C]$ ($\{A\} \rightarrow \{B, C\}$).

Note: $\{E\} \rightarrow \{F\}$ is **lost** by this decomposition — E and F never end up together in any final relation, so that FD can no longer be checked directly against the decomposed schema.

Question A.4

$R [A, B, C, D, E]$ with $\{A\} \rightarrow \{B\}$, $\{C\} \rightarrow \{D, E\}$, $\{A, D, E\} \rightarrow \{C\}$, $\{B, C\} \rightarrow \{A\}$. Decompose into BCNF.

Working

$\{A, C\}$, $\{A, D, E\}$ and $\{B, C\}$ are all candidate keys.

$\{A\} \rightarrow \{B\}$ violates BCNF $\rightarrow$ split into $R_1 [A, B]$ and $R_2 [A, C, D, E]$. Within R_2 , $\{C\} \rightarrow \{D, E\}$ violates BCNF $\rightarrow$ split into $R_3 [C, D, E]$ and $R_4 [A, C]$.

Final answer: $R_1 [A, B]$, $R_3 [C, D, E]$, $R_4 [A, C]$ — *the exact same result as Question A.2*, even though A.4 started with twice as many FDs. $\{A, D, E\} \rightarrow \{C\}$ and $\{B, C\} \rightarrow \{A\}$ are preserved.

$C\} \rightarrow \{A\}$ are both **lost** in this decomposition (their attributes never all end up together in one final relation) — this is worth noticing: extra FDs don't necessarily survive decomposition, and a differently-ordered decomposition might have preserved different (but not necessarily *more*) FDs.

Section B — 3NF synthesis (bottom-up)

Question B.1

$R [A, B, C, D, E, F, G, H]$ with $\{A, B\} \rightarrow \{C\}$, $\{C, D\} \rightarrow \{E\}$, $\{D\} \rightarrow \{F, G\}$. Synthesise into 3NF.

Working

$\{A, B, D, H\} = \{A, B, C, D, E, F, G, H\}$ — the only candidate key. Prime attributes: A, B, D, H.

This FD set is already a minimal cover (RHS already single-attribute per FD after splitting $\{D\} \rightarrow \{F, G\}$ into $\{D\} \rightarrow F$ and $\{D\} \rightarrow G$; no LHS or whole-FD redundancy).

Synthesis: R1 $[A, B, C]$ ($\{A, B\} \rightarrow C$), R2 $[C, D, E]$ ($\{C, D\} \rightarrow E$), R3 $[D, F, G]$ ($\{D\} \rightarrow \{F, G\}$), and — since the candidate key $\{A, B, D, H\}$ isn't contained in any of these — R4 $[A, B, D, H]$ (no non-trivial FD). All four are in 3NF.

Question B.2

$R [A, B, C, D, E, F]$ with $\{A, B\} \rightarrow \{C\}$, $\{C\} \rightarrow \{D, E\}$. Synthesise into 3NF.

Working

$\{A, B, F\} = \{A, B, C, D, E, F\}$ — the only candidate key. Prime attributes: A, B, F. Already a minimal cover.

Synthesis: R1 $[A, B, C]$ ($\{A, B\} \rightarrow C$), R2 $[C, D, E]$ ($\{C\} \rightarrow \{D, E\}$), R3 $[A, B, F]$ (no non-trivial FD, added since the key isn't otherwise present). All three are in 3NF.

Question B.3

$R [A, B, C, D, E]$ with $\{A\} \rightarrow \{D, E\}$, $\{D\} \rightarrow \{A\}$, $\{B\} \rightarrow \{C\}$, $\{B, C\} \rightarrow \{A, D\}$, $\{E, A\} \rightarrow \{D\}$. Synthesise into 3NF.

i Working

$\{B\} = \{A, B, C, D, E\}$ — $\{B\}$ is the only candidate key. Prime attributes: B.

Minimal cover: RHS-splitting and removing redundant LHS attributes (e.g. $\{E, A\} \rightarrow \{D\}$ simplifies to $\{A\} \rightarrow \{D\}$, since E is redundant once A is present — A alone already reaches D) and redundant whole FDs ($\{B\} \rightarrow \{D\}$ becomes derivable via $\{B\} \rightarrow \{A\} \rightarrow \{D\}$ so it's dropped; the duplicate simplified $\{A\} \rightarrow \{D\}$ collapses into one) leaves: $\{A\} \rightarrow \{D, E\}$, $\{B\} \rightarrow \{A, C\}$, $\{D\} \rightarrow \{A\}$.

Synthesis: R1 [A, D, E] ($\{A\} \rightarrow \{D, E\}$, $\{D\} \rightarrow \{A\}$), R2 [A, B, C] ($\{B\} \rightarrow \{A, C\}$) — the candidate key $\{B\}$ is already present in R2, so no extra all-prime-attributes relation is needed. Both are in 3NF.

Question B.4

R [A, B, C, D, E, F] with $\{A\} \rightarrow \{B, C, D, E, F\}$, $\{B, C\} \rightarrow \{A\}$, $\{D, E\} \rightarrow \{B\}$, $\{C\} \rightarrow \{D\}$. Synthesise into 3NF.

i Working

$\{A\}$, $\{B, C\}$ and $\{C, E\}$ are all candidate keys. Prime attributes: A, B, C, E.

Minimal cover: RHS-splitting $\{A\} \rightarrow \{B, C, D, E, F\}$ and simplifying — $\{A\} \rightarrow \{B\}$ is redundant (derivable via $\{A\} \rightarrow \{D\}$, $\{A\} \rightarrow \{E\}$, $\{D, E\} \rightarrow \{B\}$); $\{A\} \rightarrow \{D\}$ is redundant (derivable via $\{A\} \rightarrow \{C\}$, $\{C\} \rightarrow \{D\}$) — leaves: $\{A\} \rightarrow \{C, E, F\}$, $\{B, C\} \rightarrow \{A\}$, $\{D, E\} \rightarrow \{B\}$, $\{C\} \rightarrow \{D\}$.

Synthesis: R1 [A, C, E, F] ($\{A\} \rightarrow \{C, E, F\}$), R2 [B, C, A] ($\{B, C\} \rightarrow \{A\}$), R3 [D, E, B] ($\{D, E\} \rightarrow \{B\}$), R4 [C, D] ($\{C\} \rightarrow \{D\}$). The candidate key $\{A\}$ is already present in R1, so no extra relation is needed. All four are in 3NF.

Case Study 9: Payroll System (BCNF Decomposition and 3NF Synthesis)

Practice for 2026-05-04-normalisation-and-relational-database-schema-design⁵. Continues week10-tutorial-case-study-8-dirt-road-driving⁶ — Peter has now resent Dirt Road Driving's payroll schema *with* the full set of functional dependencies included (his earlier email accidentally omitted them).

⁵courses/infs1200/2026-05-04-normalisation-and-relational-database-schema-design.pdf

⁶courses/infs1200/week10-tutorial-case-study-8-dirt-road-driving.pdf

The corrected payroll schema

Employee [id, firstName, lastName, role]

Project [name, description, funding, projectLeader]

Project.projectLeader references Employee.id

TimeLog [employeeID, projectName, date, hoursWorked, approved]

TimeLog.employeeID references Employee.id

TimeLog.projectName references Project.name

AssetUse [employeeID, assetID, timestamp, useDuration, assetType, purchaseDate, insuranceValue]

assetID → assetType, purchaseDate

assetType → insuranceValue

AssetUse.employeeID references Employee.id

Department [code, name, manager, buildingID, buildingName, buildingLocation, floor]

buildingID → buildingLocation, buildingName

buildingName → buildingLocation, buildingID

Department.manager references Employee.id

EmployeeHistory [employeeID, departmentCode, seniorityLevel, baseSalary, securityLevel]

seniorityLevel, securityLevel → baseSalary

EmployeeHistory.employeeID references Employee.id

EmployeeHistory.departmentCode references Department.code

TripExpenseAllocations [tripName, expenseType, quantity, organiser, startDate, endDate, location]

tripName, expenseType → quantity, description

tripName → startDate, endDate, location, organiser

expenseType → allowance, restrictions

TripExpenseAllocations.organiser references Employee.id

TravelInsuranceHistory [tripName, approved, insuranceLevel, description, maxCoverage, advised]

tripName → approved, insuranceLevel, description

insuranceLevel → description, maxCoverage

TravelInsuranceHistory.tripName references TripExpenseAllocations.tripName

Section A — BCNF decomposition

Decompose (if needed) AssetUse, Department, EmployeeHistory, TripExpenseAllocations, and TravelInsuranceHistory into BCNF, clearly stating any new tables and all foreign keys.

AssetUse

Working

CK = {employeeID, assetID, timestamp}. FDs: {employeeID, assetID, timestamp} → {useDuration, assetType, purchaseDate, insuranceValue}, {assetID} → {assetType, purchaseDate}, {assetType} → {insuranceValue}, and the implicit {assetID} → {insuranceValue}. Highest NF: **1NF**.

Final answer:

- Asset [assetID, assetType, purchaseDate] — {assetID} → {assetType, purchaseDate}
- Insurance [assetID, insuranceValue] — {assetID} → {insuranceValue}, FK assetID references Asset.assetID
- AssetUse [employeeID, assetID, timestamp, useDuration] — no non-trivial FD, FK assetID references Asset.assetID

Lost: {assetType} → {insuranceValue} (once decomposed, assetType and insuranceValue never appear together in one relation, so this FD can no longer be directly enforced/checked).

Department

Working

CK = {code}. FDs: {code} → {name, manager, buildingID, buildingName, location, floor}, {buildingID} → {buildingLocation, buildingName}, {buildingName} → {buildingLocation, buildingID}. Highest NF: **2NF**.

Final answer:

- Building [buildingID, buildingLocation, buildingName] — {buildingID} → {buildingLocation, buildingName}, {buildingName} → {buildingID, buildingLocation}
- Department [code, name, manager, buildingID, floor] — {code} → {name, manager, buildingID, floor}, FK buildingID references Building.buildingID

EmployeeHistory

Working

CK = {employeeID, departmentCode}. FDs: {employeeID, departmentCode} → {seniorityLevel, baseSalary, securityLevel}, {seniorityLevel, securityLevel} → {baseSalary}. Highest NF: **2NF**.

Final answer:

- Salary [seniorityLevel, securityLevel, baseSalary] — {seniorityLevel, securityLevel} → {baseSalary}
- EmployeeHistory [employeeID, departmentCode, seniorityLevel, securityLevel] — {employeeID, departmentCode} → {seniorityLevel, securityLevel}, FKs to Employee.id and Department.code

TripExpenseAllocations

Working

CK = {tripName, expenseType}. FDs: {tripName, expenseType, quantity} → {organiser, startDate, endDate, location, allowance, restrictions, description}, {tripName, expenseType} → {quantity, description}, {tripName} → {startDate, endDate, location, organiser}, {expenseType} → {allowance, restrictions}. Highest NF: **1NF**.

Final answer:

Relation	FD
Trip [tripName, startDate, endDate, location, organiser]	{tripName} → {startDate, endDate, location, organiser}
Expense [expenseType, allowance, restrictions]	{expenseType} → {allowance, restrictions}
TripExpenseAllocations [tripName, expenseType, quantity, description]	{tripName, expenseType} → {quantity, description}, FKs to Trip.tripName and Expense.expenseType

TravelInsuranceHistory

Working

CK = {tripName} (the schema's stated primary key {tripName, approved, insuranceLevel} is **not minimal** — tripName alone is already a key).
FDs: {tripName, approved, insuranceLevel} → {description, maxCoverage, advisedPrecautions}, {tripName} → {approved, insuranceLevel, description}, {insuranceLevel} → {description, maxCoverage}. Highest NF: **2NF**.

Final answer:

- InsuranceCoverage [insuranceLevel, description, maxCoverage] — {insuranceLevel} → {description, maxCoverage}
- TravelInsuranceHistory [tripName, approved, insuranceLevel, advisedPrecautions] — {tripName} → {approved, insuranceLevel}, {tripName, approved, insuranceLevel} → {advisedPrecautions}, FK insuranceLevel references InsuranceCoverage.insuranceLevel

Section B — 3NF synthesis

Decompose (if needed) Department and TripExpenseAllocations into 3NF via minimal-cover synthesis.

Department

Working

CK = {code}. Highest NF (from Section A): 2NF.

Minimal cover: RHS-splitting {code} → {name, manager, buildingID, buildingName, buildingLocation, floor} and removing redundant attributes — code → buildingName and code → buildingLocation are both redundant (derivable transitively via code → buildingID then buildingID → buildingName/buildingLocation); buildingID → buildingLocation is redundant (derivable via buildingID → buildingName then buildingName → buildingLocation). **Simplified cover:** {code} → {name, manager, buildingID, floor}, {buildingID} → {buildingName}, {buildingName} → {buildingLocation, buildingID}.

Synthesis:

- Department [code, name, manager, buildingID, floor] — {code} → {name, manager, buildingID, floor}

- Building [buildingName, buildingLocation, buildingID] —
 $\{\text{buildingID}\} \rightarrow \{\text{buildingName}\}, \quad \{\text{buildingName}\} \rightarrow \{\text{buildingLocation}, \text{buildingID}\}$ (a candidate relation [buildingID, buildingName] from $\{\text{buildingID}\} \rightarrow \{\text{buildingName}\}$ alone is redundant — it's a subset of this Building relation, so it's dropped.)

TripExpenseAllocations

i Working

CK = {tripName, expenseType} (the stated primary key is not minimal). Highest NF (from Section A): 1NF.

Minimal cover: after RHS-splitting and removing every attribute already implied by {tripName} or {expenseType} alone from the {tripName, expenseType, quantity} → ... FD (all seven of its RHS attributes turn out to be redundant there, since each is already determined by tripName or expenseType individually), the simplified cover is exactly: {tripName, expenseType} → {quantity, description}, {tripName} → {startDate, endDate, location, organiser}, {expenseType} → {allowance, restrictions}.

Synthesis:

- TripExpenseAllocations [tripName, expenseType, quantity, description] — {tripName, expenseType} → {quantity, description}
- Trip [tripName, startDate, endDate, location, organiser] —
 $\{\text{tripName}\} \rightarrow \{\text{startDate}, \text{endDate}, \text{location}, \text{organiser}\}$
- Expense [expenseType, allowance, restrictions] — {expenseType} → {allowance, restrictions}

(Same final relations as the BCNF decomposition in Section A — in this case, decomposition and synthesis happen to converge on the same schema.)