DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment 4

Name: Vanshit Kumar
UID: 23BCS10087
Branch: BE-CSE
Section/Group: KRG - 1A
Semester: 5th
Date of Performance: 11 Sep, 2025
Course Name: ADBMS
Course Code: 23CSP-333

1. AIM

1) Consider a relation R having attributes as R(ABCD), functional dependencies are given below: AB->C, C->D, D->A Identify the set of candidate keys possible in relation R. List all the set of prime and non-prime attributes.
2) Relation R(ABCDE) having functional dependencies as : A->D, B->A, BC->D, AC->BE Identify the set of candidate keys possible in relation R. List all the set of prime and nonprime attributes.
3) Consider a relation R having attributes as R(ABCDE), functional dependencies are given below: B->A, A->C, BC->D, AC->BE. Identify the set of candidate keys possible in relation R. List all the set of prime and non-prime attributes.
4) Consider a relation R having attributes as R(ABCDEF), functional dependencies are given below: A->BCD, BC->DE, B->D, D->A Identify the set of candidate keys possible in relation R. List all the set of prime and non-prime attributes.
5) Designing a student database involves certain dependencies which are listed below: X ->Y, WZ ->X, WZ ->Y, Y ->W, Y ->X, Y ->Z. The task here is to remove all the redundant FDs for efficient working of the student database management system.

3. Solution

Problem 1 — R(A B C D), FDs: AB → C, C → D, D → A

Compute closures / keys:

- AB+AB+AB+: AB → C ⇒ have A,B,C. C → D ⇒ have D. So AB+={A,B,C,D}AB+ = {A,B,C,D}AB+={A,B,C,D}. ⇒ **AB** is a key. It's minimal (neither A nor B alone is a key).
- BC+BC+BC+: B,C → from C→D and D→A gives A,B,C,D ⇒ **BC** is a key.
- BD+BD+BD+: B,D → D→A gives A, with A and B we get C via AB→C ⇒ all ⇒ **BD** is a key.
- No single attribute is a key; other 2-combinations not containing B produce no B, so not keys.

Candidate keys: { **AB**, **BC**, **BD** }
Prime attributes: attributes that occur in some candidate key = A, B, C, D (all of them)
Non-prime attributes: none

Problem 2 — R(A B C D E), FDs: A → D, B → A, BC → D, AC → BE

Find candidate keys:

- AC+AC+AC+: AC → BE gives B and E; B → A (already), A → D gives D ⇒ AC+={A,B,C,D,E}AC+ = {A,B,C,D,E}AC+={A,B,C,D,E}. So **AC** is a key (minimal).
- BC+BC+BC+: BC → D, B → A, then AC → BE (since we have A and C) gives E ⇒ BC+={A,B,C,D,E}BC+ = {A,B,C,D,E}BC+={A,B,C,D,E}. So **BC** is a key (minimal).
- Check single attributes: B+ = {B,A,D} (missing C,E) → not key. A+ = {A,D} → not key. So only AC and BC are minimal keys.

Candidate keys: { **AC**, **BC** }
Prime attributes: attributes appearing in some candidate key = A, B, C
Non-prime attributes: D, E

Problem 3 — R(A B C D E), FDs: B → A, A → C, BC → D, AC → BE

This is similar but ordering yields different minimal keys:
Compute closures:

- A+A+A+: A → C. With A and C, AC → BE gives B and E. With B and C, BC → D gives D. So A+={A,B,C,D,E}A+ = {A,B,C,D,E}A+={A,B,C,D,E}. So **A** alone is a key.
- B+B+B+: B → A, then A→C, AC→BE gives E, and BC→D (we have B and C) gives D ⇒ B+={A,B,C,D,E}B+ = {A,B,C,D,E}B+={A,B,C,D,E}. So **B** alone is also a key.
- C alone not key.

Candidate keys: { **A**, **B** } (both single-attribute)
Prime attributes: A, B
Non-prime attributes: C, D, E

Problem 4 - R(A B C D E F), FDs: A → B C D, BC → D E, B → D, D → A

Observations: D → A and A → B,C,D means A and D are mutually determining (cycle).
Also B → D → A, so B implies A as well (via D).
So any attribute that implies A (i.e. A, B, or D) together with F gives all attributes:

- AF+A F+AF+: A → B,C,D; BC → DE gives E; include F ⇒ all. ⇒ **AF** is a key.
- BF+B F+BF+: B → D → A, A → B,C,D; BC → DE gives E; include F ⇒ all. ⇒ **BF** is a key.
- DF+D F+DF+: D → A → B,C,D; BC → DE gives E; include F ⇒ all. ⇒ **DF** is a key.

Check minimality: A, B, D alone do not contain F, F alone is not a key. So each pair (A/B/D) with F is minimal.

Candidate keys: { **AF**, **BF**, **DF** }
Prime attributes: attributes appearing in some candidate key = A, B, D, F
Non-prime attributes: C, E

Problem 5 — Student DB: FDs

X → Y, WZ → X, WZ → Y, Y → W, Y → X, Y → Z
We want a minimal (redundant-free) FD set (minimal cover).
Steps & observations:

1. Notice WZ→Y WZ \to YWZ→Y is implied by WZ→XWZ \to XWZ→X together with X→YX \to YX→Y, because WZ→XWZ \to XWZ→X and X→YX \to YX→Y imply WZ→YWZ \to YWZ→Y. So **WZ → Y** is **redundant** and can be removed.
2. Notice Y→XY \to XY→X is implied by Y→WY \to WY→W and Y→ZY \to ZY→Z together with WZ→XWZ \to XWZ→X: from Y we get W and Z, and WZ → X gives X. So **Y → X** is **redundant** and can be removed.
3. After removing those, check remaining FDs: X→Y, WZ→X, Y→W, Y→Z. All RHSs are single attributes; check LHS extraneous attributes:
    - In WZ→X neither W nor Z is extraneous (neither W→X nor Z→X is derivable alone).
    - The Y → … FDs have single-attribute LHS, so nothing to reduce there.
    - X→Y is necessary (not implied by the others).

So a minimal (redundant-free) set of FDs is:
Minimal cover:

- X → Y
- WZ → X
- Y → W
- Y → Z

(From these, Y → X is derivable (Y→W and Y→Z give WZ, and WZ→X), and WZ→Y is derivable (WZ→X and X→Y).)

4. Learning Outcomes

1. Learned to compute candidate keys using attribute closure.
2. Understood how to classify prime and non-prime attributes.
3. Identified partial dependencies and their effect on normalization.
4. Gained practical insight into reducing redundancy and anomalies in database design.