TG ACADEMIC RESOURCE SERIES

DATABASE MANAGEMENT SYSTEMS

Functional Dependencies

Concepts, Attribute Closure & Candidate-Key Analysis

SUBJECT	DBMS / Relational Database Design
LEVEL	BTech / BCA / MCA / BSc / MSc / MTech
RESOURCE TYPE	Sample learning note
PUBLICATION ID	DBMS-001 | Version 1.0

Prepared by Tanmoy Gupta

Computer Science Educator & Industry Professional

This sample introduces functional dependencies through a practical enrolment relation. It demonstrates attribute closure and the identification of a candidate key, with a worked, step-by-step derivation.

A focused introduction—not a complete unit on normalisation, minimal covers or decomposition.

1. Learning Objectives

After studying this note, you should be able to:
• Explain the meaning of a functional dependency.
• Read and interpret an FD written as X → Y.
• Distinguish trivial and non-trivial functional dependencies.
• Calculate the closure of an attribute set.
• Use closure to test whether an attribute set is a superkey and identify a candidate key.

2. What Is a Functional Dependency?

A **functional dependency (FD)** is a constraint between sets of attributes in a relation. An FD **X** → **Y** means that whenever two rows agree on all attributes in X, they must also agree on all attributes in Y.

In other words, the value of X uniquely determines the value of Y within the relation, under the stated business rules.

Example

Suppose each StudentID identifies exactly one student name. Then:

StudentID → StudentName

This does not mean StudentName must be unique. Two different students may share the same name. The dependency says that one StudentID cannot be associated with two different StudentName values.

3. Trivial and Non-Trivial FDs

An FD X → Y is **trivial** when Y is a subset of X. For example, {StudentID, CourseID} → StudentID is trivial.

An FD X → Y is **non-trivial** when Y is not a subset of X. For example, StudentID → StudentName is non-trivial.

The dependency must be justified by the meaning of the data—not merely by a small sample of rows that happens to show the same pattern.

4. Worked Scenario: Student Enrolment

Consider the relation ENROLMENT with the following attributes:

ENROLMENT(StudentID, StudentName, CourseID, CourseName, InstructorID, InstructorName, Grade)

Assume the following business rules:

1. Each StudentID identifies one student and therefore determines StudentName.
2. Each CourseID identifies one course and its assigned instructor.
3. Each InstructorID identifies one instructor name.
4. A student's grade is recorded separately for each course in which the student is enrolled.

The functional dependencies are therefore:

FD	Business meaning
StudentID → StudentName	A student ID determines the student's name.
CourseID → CourseName, InstructorID	A course ID determines its title and assigned instructor.
InstructorID → InstructorName	An instructor ID determines the instructor's name.
StudentID, CourseID → Grade	A student's enrolment in a course determines that enrolment's grade.

Important modelling assumption

This example assumes a course has one assigned instructor at a time and that a student has at most one enrolment record per course. If the real business rules allow multiple sections or repeated attempts, the relation and dependencies may need to change.

5. What Is Attribute Closure?

The **closure of an attribute set X**, written X■, is the set of all attributes that can be functionally determined by X using the given FD set.

Closure is useful because:

• If X■ contains every attribute of the relation, X is a superkey.
• If X is a superkey and no proper subset of X is a superkey, X is a candidate key.

6. Complete Worked Example: Calculate (StudentID, CourseID)■

Let X = {StudentID, CourseID}. Start with the attributes in X and repeatedly apply every FD whose left-hand side is already contained in the current set.

Initial set

X■ = {StudentID, CourseID}

Step	FD applied	New attribute(s)	Closure after step
0	Initial set	—	{StudentID, CourseID}
1	StudentID → StudentName	StudentName	{StudentID, CourseID, StudentName}
2	CourseID → CourseName, InstructorID	CourseName, InstructorID	{StudentID, CourseID, StudentName, CourseName, InstructorID}
3	InstructorID → InstructorName	InstructorName	{StudentID, CourseID, StudentName, CourseName, InstructorID, InstructorName}
4	StudentID, CourseID → Grade	Grade	All seven attributes

At the end, no further FD can add a new attribute. The closure is:

X■ = {StudentID, StudentName, CourseID, CourseName, InstructorID, InstructorName, Grade}

The closure contains all seven attributes of ENROLMENT. Therefore, {StudentID, CourseID} is a **superkey**.

7. Is It a Candidate Key?

To establish minimality, test the proper subsets:

- StudentID■ = {StudentID, StudentName}. It does not contain all attributes.
- CourseID■ = {CourseID, CourseName, InstructorID, InstructorName}. It does not contain all attributes.
- The empty set cannot determine StudentID or CourseID under the stated FD set.

Neither single-attribute subset is a superkey. Thus {StudentID, CourseID} is minimal and is a **candidate key**.

8. Why Functional Dependencies Matter

Functional dependencies help database designers understand which facts determine other facts. They are a foundation for identifying keys and analysing redundancy during relational schema design.

In the enrolment example, CourseID determines CourseName and InstructorID, while InstructorID determines InstructorName. Storing all these facts in every student-course row can create repeated data. This is one reason database designers study normalisation.

A hint of the next topic

The dependency chain CourseID → InstructorID → InstructorName indicates that CourseID can also determine InstructorName by transitivity. Analysing such dependencies in relation to candidate keys is important in higher normal forms.

9. Check Your Understanding

Practice Questions

Q1. If every EmployeeID identifies exactly one EmployeeName, write the corresponding functional dependency.

Q2. Is {A, B} → A trivial or non-trivial? Give the reason.

Q3. Given R(A, B, C) and F = {A → B, B → C}, calculate A■.

Q4. If X■ contains every attribute of relation R, what can be concluded about X?

Suggested Answers (brief)

A1. EmployeeID → EmployeeName.

A2. Trivial, because the right-hand-side attribute A is already a member of the left-hand-side set {A, B}.

A3. Start with {A}; use A → B to add B; then use B → C to add C. Therefore A■ = {A, B, C}.

A4. X is a superkey of R. To call it a candidate key, minimality must also be established.

Where to Go Next

A complete DBMS learning sequence can continue with Armstrong's axioms, attribute-closure problem sets, minimal cover, candidate-key identification, 2NF/3NF/BCNF, lossless decomposition and dependency preservation. These are separate, deeper learning topics.

In guided classes: students can practise deriving closures, testing minimality, proving candidate keys and solving progressively harder normalisation problems with individual feedback.

About this resource
TG Academic Resource Series | DBMS-001 | Version 1.0
Prepared by Tanmoy Gupta — Computer Science Educator & Industry Professional

Original educational material prepared for student learning and revision. Redistribution or commercial reproduction without permission is not authorised.