

UNIT 3

Relational Database Model

(4 Hours) — Database Management System (CACS255) · BCA II Year / IV Semester

⚠ Important Note — Please Read

These notes were prepared quickly from the course syllabus and standard textbook knowledge. **They may contain errors, typing mistakes or incomplete points.** Please take care: cross-check definitions, formulas, SQL syntax and examples with your class notes, the prescribed textbook and your teacher before relying on them for exams or practicals.

3.1 Structure of RDBMS and Terminology

The **relational model**, proposed by **E. F. Codd (1970)**, represents data as a collection of two-dimensional tables called **relations**. A DBMS based on it is an **RDBMS**.

Formal term	Common term	Meaning
Relation	Table	A set of tuples with the same attributes
Tuple	Row / Record	One entry of the table
Attribute	Column / Field	A named property of the relation
Domain	Data type / value set	The set of permitted values for an attribute
Degree	Number of columns	Count of attributes in the relation
Cardinality	Number of rows	Count of tuples in the relation
Relation schema	Table definition	Name of the relation and its attributes, e.g. Student(SID, Name, Dept)
Relation instance	Table contents	Data present in the relation at a given time

Properties of a Relation

- Each relation has a unique name; each attribute name in it is unique.
- All values in a column come from the same domain and are **atomic** (single, indivisible).
- The order of rows and the order of columns are not important.
- No two tuples are identical (duplicates are not allowed in the pure model).

Example 3.1 — Relation Student

Student(SID, Name, Dept, Age) has **degree** = 4. If it holds 3 rows, its **cardinality** = 3.

SID	Name	Dept	Age
1	Ram	CS	20
2	Sita	IT	21
3	Hari	CS	22

Database Schema and Schema Diagram

A **database schema** is the logical design of the whole database (all relation schemas plus constraints). A **schema diagram** is a picture of it: each relation is drawn as a box with its name on top and its attributes listed below; the primary key is underlined and arrows run from each foreign key to the primary key it references.

Example 3.2 — Schema of a college database

Department(DeptID, DeptName)

Student(SID, Name, Age, *DeptID*) — DeptID → Department.DeptID

Course(CID, Title, Credits)

Enrollment(*SID*, CID, Grade) — SID → Student.SID, CID → Course.CID

3.2 Keys and Relationships

Key	Definition	Example (Student)
Super key	Any set of attributes that uniquely identifies a tuple (may contain extra attributes).	{SID}, {SID, Name}, {Email}
Candidate key	A <i>minimal</i> super key — no attribute can be removed.	{SID}, {Email}
Primary key	The candidate key chosen to identify tuples; cannot be NULL or duplicate.	SID
Alternate key	Candidate keys not chosen as primary key.	Email
Composite key	A key made of two or more attributes.	{SID, CID} in Enrollment
Foreign key	An attribute referring to the primary key of another (or the same) relation.	Student.DeptID → Department.DeptID
Unique key	Enforces uniqueness; allows NULL (once in most systems).	Email

Integrity Constraints

- **Entity integrity:** the primary key can never be NULL.
- **Referential integrity:** a foreign key value must either match an existing primary key value in the referenced table or be NULL.
- **Domain integrity:** values must lie in the attribute's domain.

Relationships between tables

- **One-to-one (1:1)** — a Person has one Passport.
- **One-to-many (1:N)** — one Department has many Students (foreign key on the many side).
- **Many-to-many (M:N)** — Students and Courses; implemented with a third (junction) table.

3.3 Relational Algebra

Relational algebra is a **procedural query language**. It has operators that take one or two relations as input and produce a new relation as output. Basic operators: **select** (σ), **project** (π), **union** ($\cup$), **set difference** ($-$), **Cartesian product** ($\times$), **rename** (ρ). Others (intersection, joins, division) can be derived from them.

Sample data used in the examples below:

SID	Name	DeptID
1	Ram	10
2	Sita	20
3	Hari	NULL

Relation Student

DeptID	DName
10	BCA
20	BBA
30	BIM

Relation Dept

1. Select (σ)

Selects the **rows** (tuples) that satisfy a condition. Notation: $\sigma(\text{condition})(R)$.

Example 3.3 — Select

$\sigma(\text{DeptID} = 10)(\text{Student})$ returns the tuples of students in department 10.

SID	Name	DeptID
1	Ram	10

2. Project (π)

Selects specified **columns** and removes duplicate rows. Notation: $\pi(A1, A2, \dots)(R)$.

Example 3.4 — Project

$\pi(\text{Name})(\text{Student})$ returns the column Name only: {Ram, Sita, Hari}.

3. Union ($\cup$), Intersection ($\cap$) and Set Difference ($-$)

These need **union-compatible** relations (same number of attributes, same domains).

Example 3.5 — Set operations

Let $R = \{1, 2, 3\}$ and $S = \{3, 4\}$.

$R \cup S = \{1, 2, 3, 4\}$ $R \cap S = \{3\}$ $R - S = \{1, 2\}$

4. Cartesian Product ($\times$)

Combines every tuple of R with every tuple of S . If R has m tuples and S has n tuples, $R \times S$ has $m \times n$ tuples and the attributes of both.

Example 3.6 — Cartesian product

$\text{Student} \times \text{Dept}$ has $3 \times 3 = 9$ tuples and $3 + 2 = 5$ attributes; most of the rows are meaningless, which is why we usually follow it with a select on DeptID (that combination gives a join).

5. Natural Join ($\bowtie$)

Combines tuples of two relations that have equal values on all **common attributes**, and keeps the common attribute only once. Equivalent to Cartesian product + select + project.

Example 3.7 — Student ⋈ Dept

Common attribute: DeptID. Ram(10) matches BCA, Sita(20) matches BBA. Hari has NULL, so there is no match.

SID	Name	DeptID	DName
1	Ram	10	BCA
2	Sita	20	BBA

6. Outer Joins

An outer join keeps the tuples that have no match and fills the missing side with NULL. Textbooks draw the join sign with open (extended) left, right or both sides; here they are written as $\bowtie(L)$, $\bowtie(R)$ and $\bowtie(F)$ for left, right and full.

Type	Symbol	Keeps	Result for our data
Left outer join	Student $\bowtie(L)$ Dept	All tuples of the left relation	Ram-BCA, Sita-BBA, Hari-NULL
Right outer join	Student $\bowtie(R)$ Dept	All tuples of the right relation	Ram-BCA, Sita-BBA, NULL-BIM
Full outer join	Student $\bowtie(F)$ Dept	All tuples of both relations	Ram-BCA, Sita-BBA, Hari-NULL, NULL-BIM

7. Rename (ρ) and Division ($\div$)

- **Rename** $\rho(X)(R)$ gives the relation R the new name X (useful in self-joins).
- **Division** $(R \div S)$ finds the tuples of R associated with *all* tuples of S, e.g. “students who enrolled in *all* courses”.

Example 3.8 — Write in relational algebra

Query: names of students who study in BCA.

Solution

$\pi(\text{Name})(\sigma(\text{DName} = \text{'BCA'})(\text{Student} \bowtie \text{Dept}))$

Step 1: join Student and Dept; Step 2: select rows where DName = BCA; Step 3: project Name → **Ram**.

Operator	Symbol	Works on	Result
Select	σ	Rows	Subset of rows
Project	π	Columns	Subset of columns (no duplicates)
Union	$\cup$	Two compatible relations	All tuples of both
Set difference	$-$	Two compatible relations	In first but not in second
Cartesian product	$\times$	Any two relations	All combinations
Natural join	$\bowtie$	Two relations	Matching on common attributes

Key Takeaways — Unit 3

- ✓ Relation = table, tuple = row, attribute = column; degree = columns, cardinality = rows.
- ✓ Keys: super $\supseteq$ candidate $\supseteq$ primary; foreign key links two tables and gives referential integrity.
- ✓ Primary key cannot be NULL (entity integrity); foreign key must match a primary key or be NULL.
- ✓ Relational algebra is procedural: σ picks rows, π picks columns, $\cup - \times$ combine relations.
- ✓ Natural join matches on common attributes; outer joins keep unmatched rows and fill NULL.
- ✓ Convert every M:N relationship into a third table with both keys.

— *End of Unit 3* —

Reminder: these notes may contain errors — verify with your textbook and teacher.