

UNIT 8

Query Processing and Security

(5 Hours) — Database Management System (CAC255) · 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.

8.1 Overview of Query Processing

Query processing is the set of activities involved in extracting data from a database. A high-level SQL query is translated into low-level operations and then executed. The main steps are:

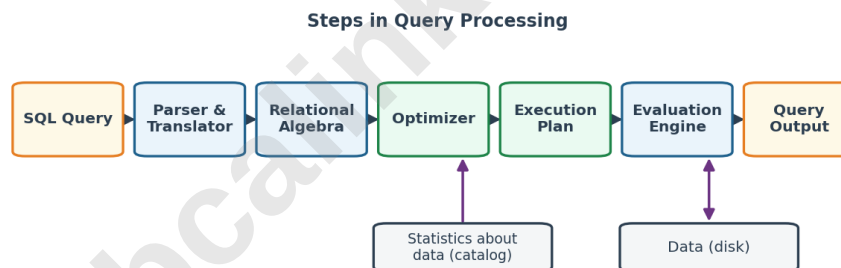


Figure 8.1 — Steps in query processing

1. **Parsing and translation** — check syntax and names, then convert the SQL into an internal form such as a relational-algebra expression.
2. **Optimization** — the optimizer considers many equivalent evaluation plans and picks the one with the lowest estimated cost, using catalog statistics.
3. **Evaluation** — the query-execution engine runs the chosen plan and returns the result.

Example 8.1 — Equivalent expressions

SQL: `SELECT Name FROM Employee WHERE Salary > 50000;`

Expression 1: $\pi(\text{Name})(\sigma(\text{Salary} > 50000)(\text{Employee}))$

Expression 2: $\sigma(\text{Salary} > 50000)(\pi(\text{Name}, \text{Salary})(\text{Employee}))$ — an equivalent but possibly slower plan; the optimizer chooses between them.

Measures of Query Cost

The cost of a query is mainly the **disk access time** (much slower than CPU). It is estimated from: number of **seeks**, number of **block transfers**, and sometimes CPU and network cost.

Cost formula

$$\text{Cost} = (\text{number of block transfers} \times tT) + (\text{number of seeks} \times tS)$$

where tT = time to transfer one block and tS = time for one disk seek (tS is much larger than tT).

Let br = number of blocks of relation r , nr = number of tuples and M = number of memory buffer blocks.

8.2 Selection Operation

Algorithm	Condition	Cost (approx.)
A1 — Linear search	Any condition, any file	Scan all blocks: br transfers + 1 seek. On a key equality: average $br/2$.
A2 — Binary search	Equality on the ordering attribute of a sorted file	$\lceil \log_2(br) \rceil$ block accesses (seeks + transfers).
A3 — Primary index, equality on key	Index on key, one record	Height of the index + 1 block.
A4 — Primary index, equality on non-key	Many matching records stored together	Index height + number of blocks holding the matches.
A5 — Secondary index, equality	Index on non-ordering attribute	Index height + 1 (key) or one block per matching record (non-key).
A6 — Comparison ($<$, $>$, $\leq$, $\geq$)	Range on an ordered/indexed attribute	Use index to find the starting point, then scan sequentially.

For **complex selections**: *conjunction* ($\sigma\theta_1 \wedge \theta_2$) — use the cheapest condition first or an intersection of pointers; *disjunction* ($\theta_1 \vee \theta_2$) — union of results or linear scan; *negation* — usually a linear scan.

8.3 Sorting

Sorting is needed for ORDER BY, merge join, duplicate removal and GROUP BY. If the relation fits in memory, use quick-sort; otherwise **external sort-merge** is used:

1. **Create sorted runs**: read M blocks at a time, sort them in memory and write each sorted *run* to disk.
2. **Merge runs**: merge up to $(M - 1)$ runs at a time (one buffer block is for output) in passes until one sorted run remains.

Cost of external sort-merge

Number of merge passes = $\lceil \log(M-1)(br/M) \rceil$.

Total block transfers = $br \times (2 \times \lceil \log(M-1)(br/M) \rceil + 1)$ (final write not counted).

Note: Formula written as in common textbooks (e.g. Silberschatz et al.); please verify it with your own textbook because notation and counting of the last write vary.

8.4 Join Operation and Evaluation of Expressions

Join algorithm	Idea	Block transfers (r outer, s inner)
Nested-loop join	For each tuple of r, scan the whole of s.	$nr \times bs + br$
Block nested-loop join	For each block of r, scan s once.	$br \times bs + br$
Indexed nested-loop join	Use an index on the join attribute of s instead of scanning.	$br + nr \times c$ (c = cost of an index lookup)
Merge (sort-merge) join	Sort both on the join attribute and merge.	$br + bs$ (if already sorted) + sort cost
Hash join	Partition both relations with a hash function, then join matching partitions.	$\approx 3(br + bs) + 4nh$

Merge join and hash join are good for equi-joins on large relations; block nested-loop is the general fallback.

Evaluation of Expressions

- **Materialization:** evaluate one operation at a time; store each intermediate result in a temporary relation on disk and use it as the input of the next operation. Simple but costs extra disk writes.
- **Pipelining:** pass tuples from one operation directly to the next without storing the intermediate result. Saves disk I/O; may be *demand-driven* (pull) or *producer-driven* (push).

Query Optimization

Query optimization selects the best plan among many equivalent ones. Two approaches (often combined):

- **Heuristic (rule-based) optimization** applies transformation rules that almost always help (listed below).

- **Cost-based optimization** estimates the cost of alternative plans from statistics (table size, number of distinct values, index availability) and chooses the cheapest.

Common heuristic rules:

- Perform **selection as early as possible** (push σ down the tree) to reduce the number of tuples.
- Perform **projection early** to remove unwanted columns.
- Replace a Cartesian product followed by a selection with a **join**.
- Choose the **join order** so that the smallest intermediate results appear first.

Some equivalence rules

$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$ | $E_1 \bowtie E_2 = E_2 \bowtie E_1$ (join is commutative) | $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$ (associative)

8.5 Database Administrator (DBA)

The **DBA** is the person or team with central control over the database system.

Roles and Responsibilities of a DBA
Schema definition — creating the original database schema using DDL.
Storage structure and access-method definition — deciding file organization and indexes.
Schema and physical-organization modification — making changes as needs evolve.
Granting authorization — deciding who may access what data.
Backup and recovery — regular backups and restoring after failures.
Performance monitoring and tuning — responding to changes in requirements.
Routine maintenance — disk space, upgrades, patches, integrity checks.
Security enforcement and user training — auditing and helping users.

8.6 Database Security

Security Issues

Database security means protecting data against accidental or intentional loss, destruction, misuse or unauthorized access. Major threats: unauthorized access, SQL injection, privilege abuse, data theft, malware, weak passwords, insider misuse and physical damage. The goals are known as the **CIA triad**: **Confidentiality** (only authorized users see data), **Integrity** (only authorized changes) and **Availability** (data is accessible when needed).

Types (Levels) of Security

Level	Protection
Physical	Guard servers against theft, fire, floods and unauthorized entry.
Human	Screen employees; prevent giving access to wrong persons; training against social engineering.
Operating system	OS login, file permissions, patches.
Network	Firewalls, SSL/TLS, VPN, encryption of communication.
Database system	Authentication, authorization (privileges), views, encryption, auditing.

Access Protection, User Accounts and Database Audits

- **Authentication** — proving identity using user name and password (or biometrics, tokens).
- **User accounts** — every user gets an account with defined privileges; the DBA creates/removes accounts. The **principle of least privilege** means giving only the access that is necessary.
- **Database audit** — keeping an **audit trail (log)** of who accessed or changed what and when, so that misuse can be detected and traced.

Creating a user account

```
CREATE LOGIN ram_login WITH PASSWORD = 'Str0ng#Pass';    -- server login (T-SQL)
CREATE USER  ram FOR LOGIN ram_login;                  -- database user
-- MySQL:   CREATE USER 'ram'@'localhost' IDENTIFIED BY 'Str0ng#Pass';
```

Discretionary Access Control (DAC)

In **DAC** the **owner** of an object decides who gets which privileges, and may pass privileges on to others. It is flexible but weaker against Trojan horses. SQL supports it with **GRANT** and **REVOKE**.

GRANT and REVOKE

```
GRANT SELECT, INSERT ON Student TO ram;           -- give privileges
GRANT UPDATE (Age) ON Student TO sita;            -- column-level privilege
GRANT SELECT ON Student TO hari WITH GRANT OPTION; -- hari can pass it on
REVOKE INSERT ON Student FROM ram;                -- take away
```

Mandatory Access Control (MAC)

In **MAC** access is decided by the system using **security classes**. Each object has a classification and each user a clearance, e.g. **Top Secret (TS) > Secret (S) > Confidential (C) > Unclassified (U)**. The **Bell-LaPadula model** enforces:

- **Simple security property:** a subject may read an object only if clearance $\geq$ classification (“no read up”).
- **Star (*) property:** a subject may write an object only if clearance $\leq$ classification (“no write down”).

Point	DAC	MAC
Who controls access	Owner of the object	System / security administrator
Flexibility	High	Low, rigid
Security level	Lower	Higher (military, government)
Based on	Privileges (GRANT/REVOKE)	Clearance and classification labels

8.7 Data Encryption and Decryption

Encryption converts readable data (**plaintext**) into unreadable form (**ciphertext**) using an algorithm and a key; **decryption** converts it back. Even if data is stolen, it stays useless without the key.

Type	Key usage	Examples
Symmetric	Same secret key encrypts and decrypts; fast.	DES, 3DES, AES
Asymmetric (public-key)	Public key encrypts, private key decrypts; slower.	RSA, ECC
Hashing (one-way)	Cannot be decrypted; used for passwords and integrity.	SHA-256, MD5 (weak)

- **Transparent Data Encryption (TDE)** encrypts the database files on disk automatically.
- **Column-level encryption** protects sensitive columns such as card numbers.

- **Encryption in transit** uses SSL/TLS between client and server.

Key Takeaways — Unit 8

- ✓ Query processing: parsing/translation → optimization → evaluation.
- ✓ Cost is mainly disk seeks and block transfers; selection uses linear search, binary search or indexes.
- ✓ External sort-merge sorts large relations; join algorithms include nested-loop, block nested-loop, indexed, merge and hash join.
- ✓ Optimization pushes selections/projections down and picks the cheapest plan using statistics.
- ✓ DBA handles schema, authorization, backup, tuning and security.
- ✓ DAC uses GRANT/REVOKE at the owner's discretion; MAC uses security classes (no read up, no write down).
- ✓ Encryption (symmetric, asymmetric) protects data; audits record user activity.

— *End of Unit 8* —

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