# CMPT 354: Database Systems I - Exam Survival Guide

**Course Philosophy:** This course is not just about "how to write SQL." It is fundamentally about **how to store and retrieve data safely and efficiently.** The professor wants to verify if you understand the constant struggle to balance **Data Integrity (Correctness)** versus **Performance (Fast I/O).**

---

## 1. The Universal Cheat Codes: When asked "Why?"

If a short-answer question asks for the **advantages/disadvantages** or **reasoning** behind a technique, algorithm, or design, and you don't recall the specific answer, pivot to one of these three perspectives.

**Perspective A: "To Reduce Disk I/O" (The Strongest Argument)**

- **When to use:** Questions regarding **Indexes**, **Storage**, **Query Optimization**, or **Join Algorithms**.
- **The Logic:** The bottleneck in database performance is almost always the **Disk**. CPU and Memory are fast, but Disk is slow.
- **What to write:** "The core objective of this technique is to minimize the number of disk accesses (Disk I/O). Specifically, it aims to reduce expensive Random I/O and maximize Sequential I/O to optimize performance."

**Perspective B: "To Eliminate Redundancy and Ensure Integrity"**

- **When to use:** Questions regarding **Normalization**, **BCNF**, or **Database Design (E/R Diagrams)**.
- **The Logic:** If the same data is stored in multiple places, updating just one instance leads to data inconsistency, known as an **Anomaly**.
- **What to write:** "The goal is to eliminate data redundancy to prevent insertion, deletion, and update anomalies, thereby ensuring data integrity."

**Perspective C: "For Data Independence"**

- **When to use:** Questions regarding **Views**, **SQL features**, or **Data Modeling**.
- **The Logic:** Applications/Users should be able to query data without knowing exactly how or where it is physically stored on the hard drive.
- **What to write:** "This provides an abstraction layer that ensures physical and logical data independence, allowing the underlying storage structure to change without breaking the application."

---

## 2. "Partial Credit" Defense Logic by Topic

If a question comes from a specific chapter and you don't remember the details, write down these core concepts.

**1) Transactions (ACID)**

- **Keyword: ACID** (Atomicity, Consistency, Isolation, Durability).
- **Defense Logic:** 90% of transaction questions relate to **Concurrency Control**. The goal is to prevent data corruption when multiple users access the DB simultaneously.
- **What to write:**
  - "The schedule must be **Conflict-Serializable**. Even if transactions run concurrently, the result must be the same as if they were executed serially to maintain consistency."
  - **If about Locking:** "We use **Strict 2PL (Two-Phase Locking)** to ensure serializability and avoid cascading aborts."

**2) Normalization (BCNF)**

- **Keyword:** Functional Dependency (FD).
- **Defense Logic:** We split tables because we cannot allow **non-key attributes to determine other attributes**.
- **What to write:**
  - "This table violates BCNF because a determinant (LHS) is not a Super Key. To resolve this, we must perform a **Lossless Decomposition**."

**3) Indexing (B+ Trees)**

- **Keyword:** Trade-off (Search Speed vs. Update Cost).
- **Defense Logic:** An index is like a book's "Table of Contents." It makes finding things fast, but if you change the content, you have to rewrite the table of contents, which is slow.
- **What to write:**
  - "Using a B+ Tree is beneficial for **Range Queries**. However, having too many indexes incurs overhead because the indexes must be updated whenever data is inserted or modified."

**4) Query Optimization (Join Algorithms)**

- **Keyword:** Cost Estimation.
- **Defense Logic:** The DBMS isn't guessing; it looks at statistics (like row counts) to find the cheapest path.
- **What to write:**
  - "The optimizer estimates the I/O cost based on table size and available memory buffer (M). It selects an algorithm like Hash Join or Sort-Merge Join over Nested Loop Join to minimize these costs."

---

## 3. Quick Responses for Specific Problem Types

**Q. Design a Physical Query Plan.** (But you don't know how) * **Strategy:** Push the **Selection ($\sigma$)** down as far as possible. * **Reasoning:** "We perform Selection first to reduce the cardinality (number of rows) of the data participating in Joins early on, which reduces intermediate result sizes and improves performance."

**Q. Why use NoSQL/MongoDB?** (Compared to RDBMS) * **Strategy:** Mention **Flexibility** and **Scalability**. * **Reasoning:** "Because there is no fixed schema, data structure changes are easy (Flexibility). It also supports horizontal scaling (Scale-out) effectively for massive datasets, though it may sacrifice strict ACID consistency for Availability."

**Q. How to prevent SQL Injection? * Strategy: Prepared Statements**. * **Reasoning:** "By using Prepared Statements (or parameterized queries), user input is treated strictly as data parameters rather than executable SQL code, preventing malicious command execution."