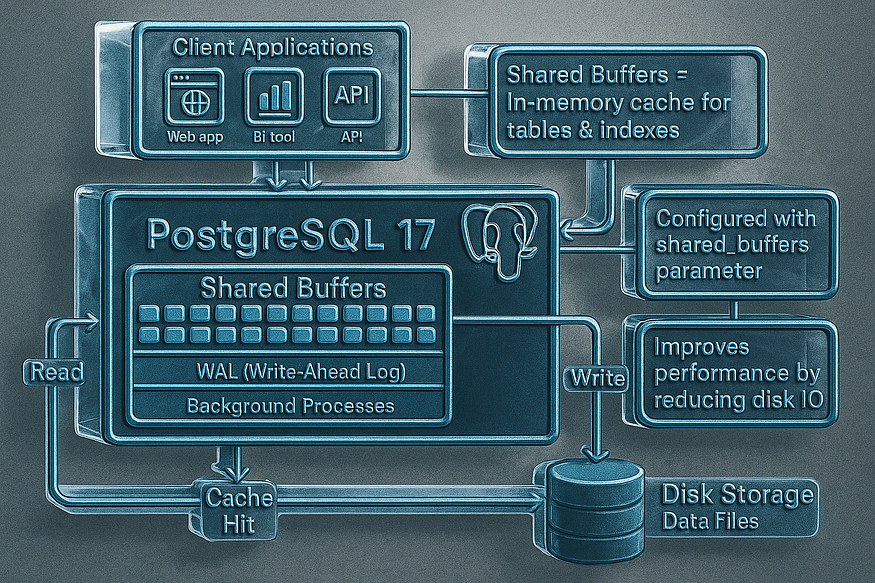
# **30 - PostgreSQL 17 Performance Tuning: The Role of Shared Buffers**



When you’re running PostgreSQL with millions of rows, performance tuning is no longer optional — it’s essential. Among all the settings available, ****shared buffers**** stand out as one of the most critical and most misunderstood. To see why, let’s walk through a realistic example.

Now our database resembles a bustling online shop with millions of products. Indexes are in place to help with search performance, but memory management is the hidden factor that determines how fast queries will really run.

## **What Exactly Are Shared Buffers?**

Shared buffers are PostgreSQL’s ****own dedicated memory cache****. Every time you query data, PostgreSQL checks this memory space first. If the data is there, it returns results immediately. If not, PostgreSQL pulls it from disk and stores it in shared buffers for the next time.

## **Analogy: The School Library Reading Room**

Think of shared buffers as the reading room of a school library. Students don’t fetch books from the basement archive every time — they use the books in the reading room, which contains the most requested titles. If “Mathematics 101” is checked out constantly, the librarian ensures it always has a place in the room. That’s exactly how shared buffers work — keeping frequently accessed data close at hand.

## **Why PostgreSQL Defaults Aren’t Enough**

Out of the box, PostgreSQL 17 sets shared\_buffers to a very small value in the postgresql.conf file. These defaults aren’t tuned for performance—they exist to guarantee PostgreSQL can run even on tiny systems.

But for a database with 10 million rows, this is like trying to run a whole school library with one small bookshelf. Sure, it works, but every student spends more time walking to the basement archive than actually reading.

## **How PostgreSQL Decides What to Keep**

PostgreSQL doesn’t just throw random data into shared buffers. It uses a ****clock-sweep algorithm****, which assigns each memory block a usage count:

* Every time a block is accessed, its score increases.
* Popular blocks (like our book category rows) stay in memory longer.
* Rarely used blocks eventually get evicted to make space for new ones.

## **Analogy: Classroom Wall Posters**

Imagine a teacher decorating a classroom wall. The multiplication chart is used every day, so it stays up. A poster from last year’s science fair might get removed when new material is needed. Shared buffers apply the same logic — popular data stays pinned in memory, while less relevant data gets cycled out.

## **Dirty Pages: Fast Updates, Delayed Writes**

When we update records — for example, changing clothing prices — PostgreSQL first changes the data in shared buffers. These modified blocks are called ****dirty pages****. Eventually, PostgreSQL writes them back to disk, but not right away.

## **Analogy: The Draft Notebook**

Students take notes on a draft notebook during class. The information is useful immediately, but at the end of the day, the notes must be copied neatly into the official class notebook. Shared buffers give PostgreSQL the same flexibility: quick in-memory updates now, with careful persistence to disk later.

## **The Balancing Act of Tuning Shared Buffers**

The hard part is knowing ****how much memory to allocate**** for shared buffers:

* ****Too small****: Queries keep going back to disk, wasting time and I/O.
* ****Too large****: PostgreSQL hogs memory and leaves little for the operating system to perform its own caching.

## **Analogy: The Cafeteria Problem**

If a cafeteria prepares too few meals in advance (small buffers), students wait in line for every order. If it prepares far too many (oversized buffers), the food goes cold and staff can’t keep up. The balance is crucial.

## **Why It Matters with 10 Million Rows**

For our items table with millions of products:

* With well-tuned shared buffers, hot queries like “find all books under $50 added this month” are lightning-fast because the required rows are already in memory.
* With poorly tuned buffers, even simple lookups repeatedly hit the disk, undoing the advantages of indexes.

In other words, ****shared buffers decide whether your database feels snappy or sluggish****.

## **Final Takeaway**

Shared buffers are PostgreSQL’s ****reading room, classroom wall, and draft notebook all rolled into one****. They control which parts of your massive dataset stay instantly accessible, and which must be fetched again from disk. For high-volume workloads like our 10 million-row items table, tuning this setting is one of the most important steps in unlocking true PostgreSQL performance.