PERFORMANCE X-RAY
DIAGNOSTIC REPORT

Beyond the Index Range Scan

Diagnosing Access vs. Filter Predicates in Oracle

ACCESS EFFICIENCY

98%
(OPTIMAL)

```
-----------------------------------------------------------------------------------------
| Id | Operation                    | Name          | Rows | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|  0 | SELECT STAT                  |               |      |       |     5 (100)|          |
|  1 |   TABLE AC  BY INDEX ROWID   | CUSTOMERS     |    1 |   120 |     5   (0)| 00:00:01 |
|* 2 |    INDEX '  E SCAN           | CUST_IDX      |    1 |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Inform n (identified by operation id):
---------------------------------------------------
   2 - access("C".  TOMER_ID"=1001)
   2 - filter("C."  IS"='ACTIVE' AND "C"."REGION"
```

EFFICIENT ACCESS: Direct Path, Low IO Cost

WASTE DETECTED: CPU/IO Filter, Post-Index Check

FILTER OVERHEAD

15%
(CPU INTENSIVE)

ACCESS PREDICATES (CYAN)	FILTER PREDICATES (ORANGE)
• Direct Data Location • Index Key Match • Minimal I/O	• Post-Access Check • Row-by-Row Evaluation • Increased CPU/IO Usage

The False Comfort of the Index Range Scan

Seeing INDEX RANGE SCAN in an execution plan feels like a victory. But ignoring the Predicate Information section leads to ""tuning by coincidence." You might be using an index, but still burning CPU and I/O on unnecessary work.

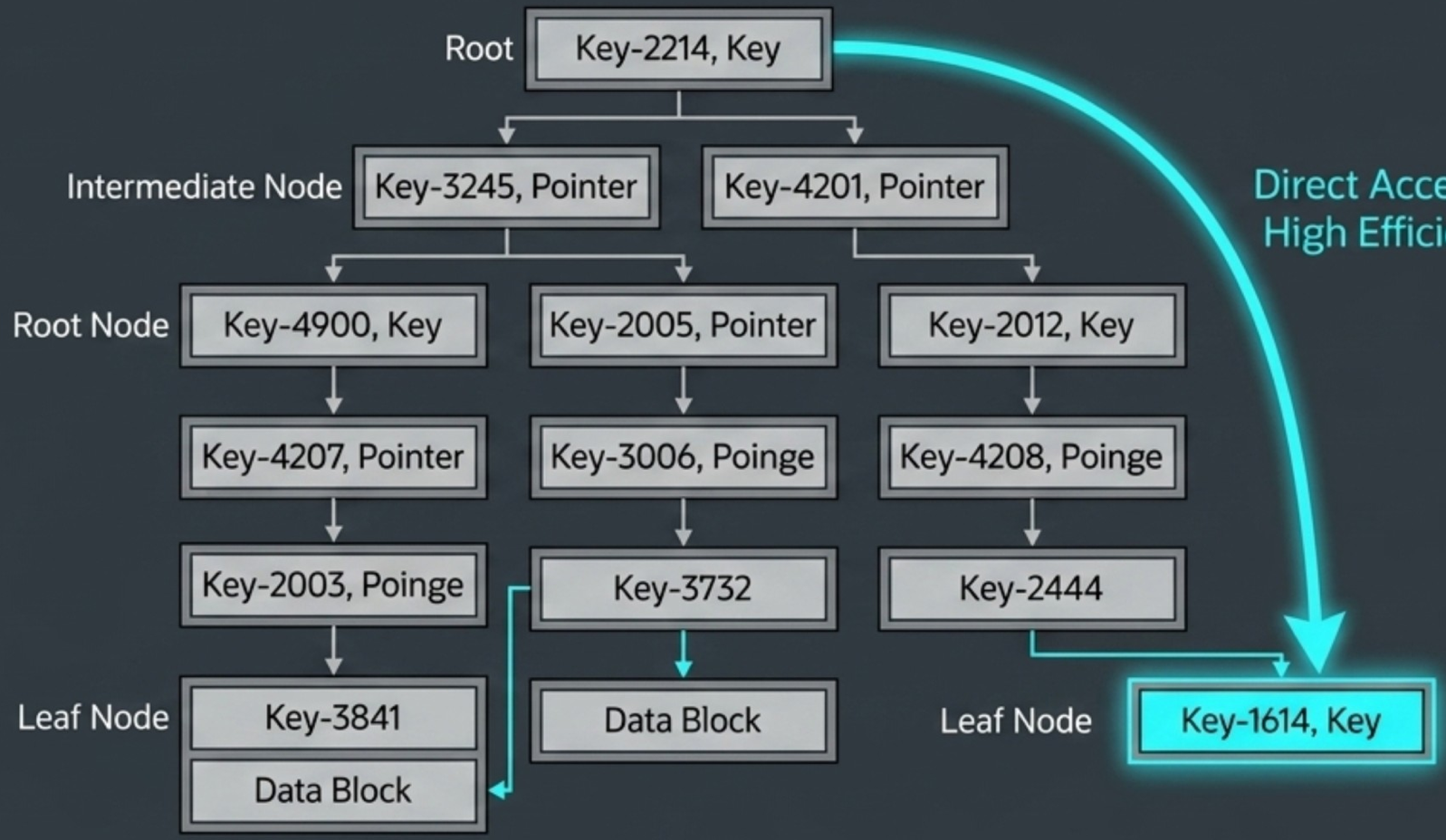
INDEX RANGE SCAN

Predicate Information: Filter

CPU & I/O Waste Spikes
Hidden Filter Predicates
Unnecessary Data Traversal

Baseline: Traversing the B-Tree

To understand predicate performance, we must look at how **Oracle navigates** data. An efficient query traverses **intermediate nodes** directly to the specific leaf node block containing the target data.

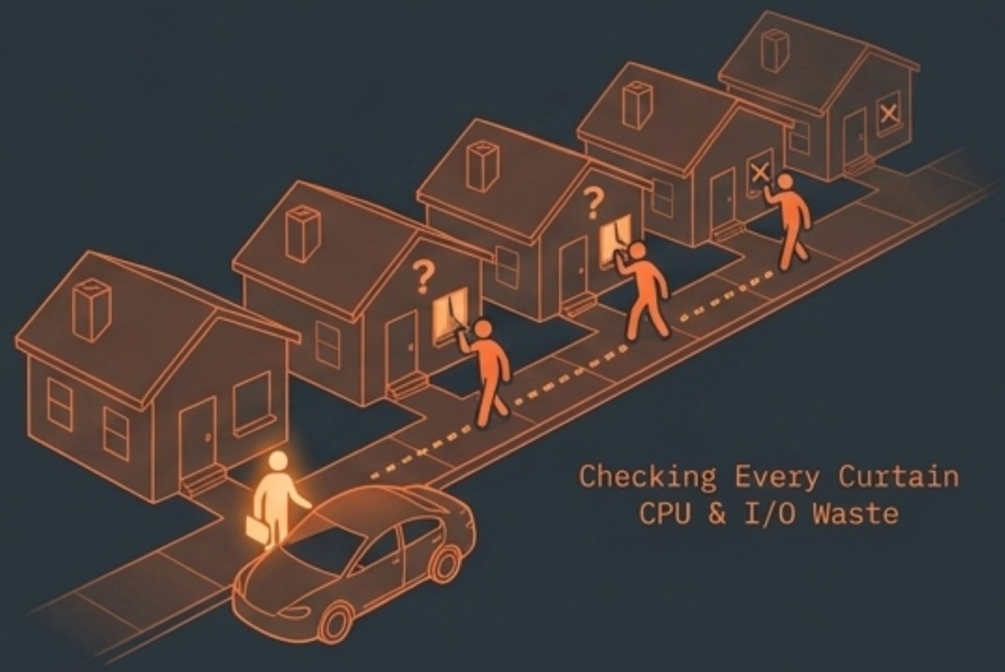


The Core Conflict: GPS vs. Checking Curtains

When querying an index, Oracle uses two vastly different methods to evaluate your WHERE clauses. The difference between them is the difference between driving directly to an address and wandering a neighborhood.

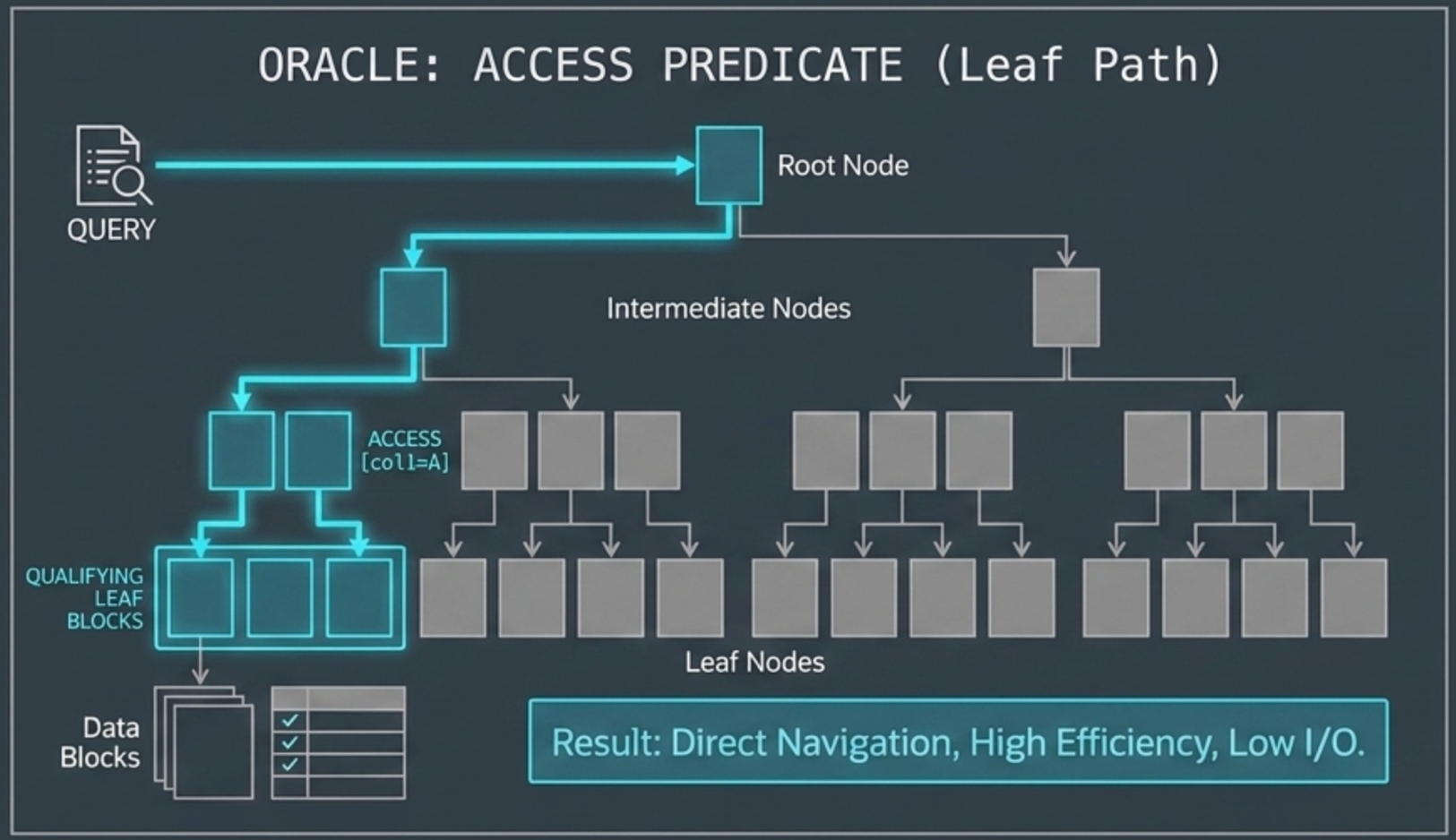
Access Predicate

Filter Predicate



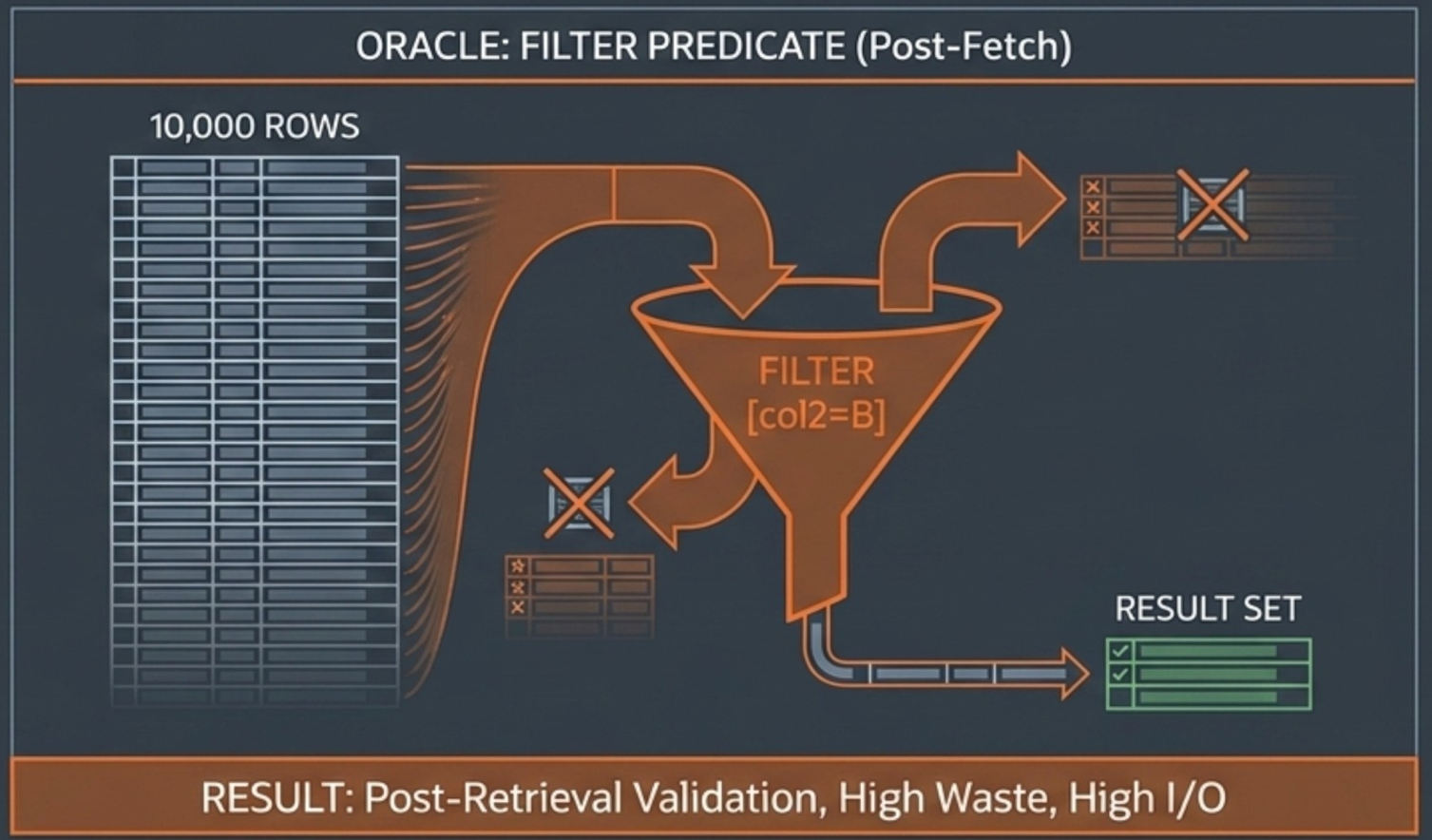
Access Predicates: The Navigation Path

Access predicates are conditions Oracle uses **to traverse the index structure** to "**jump**" directly to a **specific data range**. They are the highly efficient drivers of index range scans, unique scans, or skip scans.



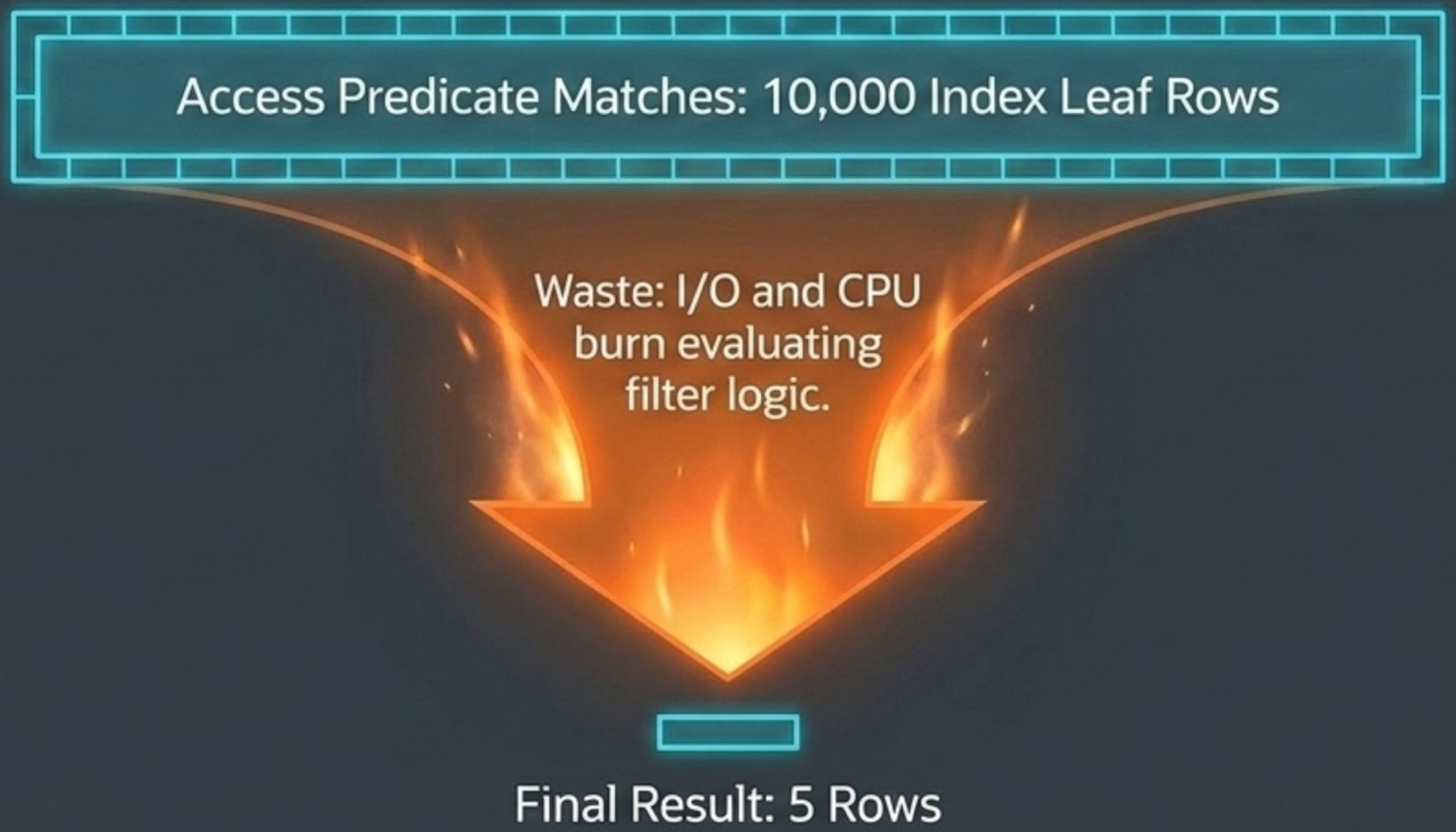
Filter Predicates: Post-Retrieval Validation

Filter predicates act as “checks” performed after data is retrieved from a structure. Every row fetched must be evaluated. If it doesn’t match the condition, it is discarded.



The Architecture of Waste

A query heavily reliant on filter predicates wastes massive amounts of system resources by fetching data it ultimately throws away.



Diagnostic Matrix: Access vs. Filter

	Access Predicate	Filter Predicate
Role	Navigation / B-Tree Search	Post-retrieval validation
Efficiency	High (jumps to data range)	Lower (tests every retrieved row)
Mechanism	Drives Index Range, Unique, Skip Scans	Applies conditions on non-indexed columns or mismatched indexes
Impact	Minimizes I/O	Increases CPU; high row drops indicate index mismatch
Metaphor	The GPS to the exact house	Checking the curtains on every house

Moving from Theory to Memory

To diagnose predicate behavior, theoretical EXPLAIN PLAN outputs aren’t enough. We need the actual execution statistics pulled directly from Oracle’s memory.

```
-- Run the query with the gathering hint
SELECT
 /*+ GATHER_PLAN_STATISTICS */
 *
FROM sales_data
WHERE region > 'EAST'
  AND sale_date = TRUNC(SYSDATE);
```

Extracting the Performance X-Ray

Immediately after executing your query, run DBMS_XPLAN.DISPLAY_CURSOR. The +PREDICATE formatting flag is the critical key that forces Oracle to reveal how it applied your WHERE clauses.

```
-- Pull the plan for the last executed statement
SELECT * FROM TABLE(
  DBMS_XPLAN.DISPLAY_CURSOR(
    format=>'ALLSTATS LAST +PREDICATE'
  )
);
```

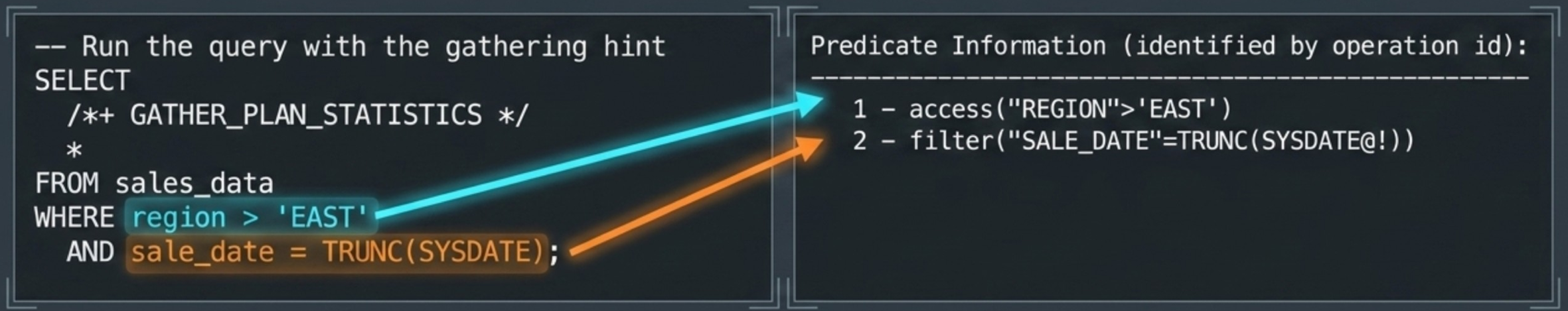
The crucial flag to reveal Access/Filter behavior.

Reading the Predicate Output

Oracle clearly separates how it navigates (access) versus how it checks (filter). Connect your WHERE clauses to the output to identify structural inefficiencies.

```
-- Run the query with the gathering hint
SELECT
  /*+ GATHER_PLAN_STATISTICS */
  *
FROM sales_data
WHERE region > 'EAST'
  AND sale_date = TRUNC(SYSDATE);
```

```
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("REGION">'EAST')
   2 - filter("SALE_DATE"=TRUNC(SYSDATE@!))
```



The Tuning Resolution

When your diagnostic X-Ray reveals an index scan burdened by a heavy filter predicate, the query is working too hard. The INDEX RANGE SCAN is a trap. It’s time to re-architect.