jsupskills

Database Indexing & Scan Strategies

Understanding Index Types and How Queries Use Them

Presented by :
jsupskills.dev

Types of Database Indexes

- **B-Tree Index** (Default for most DBs)
- **Hash Index** (Optimized for = searches)
- **Bitmap Index** (Best for low-cardinality columns)
- **Full-Text Index**(Optimized for text search)
- **GiST, GIN**, and **SP-GiST** (Used in PostgreSQL)

B-Tree Index

- Default index type in most databases
- Supports equality (=) and range queries (<, >, BETWEEN)
- Optimized for fast lookups (O(log N))

Example:

```
CREATE INDEX idx_emp_id ON employees(id);
```

Hash Index

- Optimized for exact match queries (WHERE id = 20)
- NOT suitable for range queries (<, >)
- Supported in PostgreSQL (not MySQL)

Example:

```
CREATE INDEX idx_emp_id ON employees USING HASH(id);
```

jsupskills

Bitmap Index

- Efficient for columns with low cardinality (few distinct values, e.g., gender).
- Used in data warehouses and read-heavy applications.
- ❌ Not commonly used in OLTP databases (like MySQL, PostgreSQL).

Full-Text Index

- Optimized for searching text data (LIKE '%word%' is slow without it).
- Used in MySQL (FULLTEXT), PostgreSQL (GIN/GiST indexes).

Example:

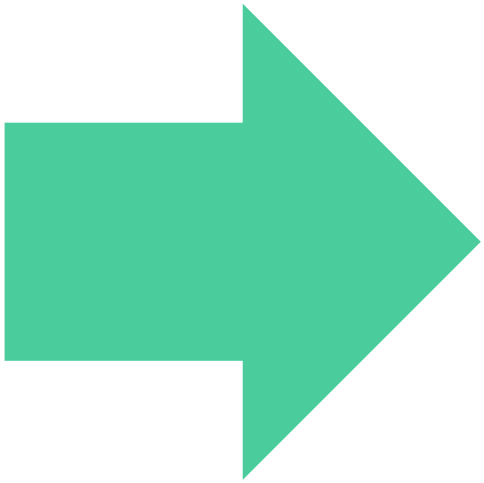
```
CREATE FULLTEXT INDEX idx_emp_name ON employees(name);
```

GiST, GIN, and SP-GiST Indexes

- **GiST (Generalized Search Tree):** Used for full-text search and geometric data.
- **GIN (Generalized Inverted Index):** Optimized for text search and JSONB in PostgreSQL.
- **SP-GiST (Space-Partitioned GiST):** Used for highly unbalanced data.

Database Scan Strategies (Using Indexes)

When an index exists, the database chooses different strategies to access data efficiently.



jsupskills

Index Scan (B-Tree)

- The database uses an index to find rows efficiently.
- Works well when only a few rows match the condition.

Example:

```
EXPLAIN ANALYZE SELECT id FROM employees WHERE id = 20;
```

Index-Only Scan

- When an index contains all the requested columns, the database avoids accessing the table.
- Faster than an Index Scan.

Example (PostgreSQL):

```
CREATE INDEX idx_emp_id_name ON employees(id, name);

EXPLAIN ANALYZE SELECT id, name FROM employees
WHERE id = 20;
```

jsupskills

Bitmap Index Scan

- Used when multiple conditions require different indexes.
- Efficiently merges results from multiple indexes.

Example (PostgreSQL):

```
EXPLAIN ANALYZE SELECT * FROM employees
WHERE id = 20 OR department = 'HR';
```

Sequential Scan (Table Scan)

A Sequential Scan (Seq Scan) occurs when the database reads the entire table row by row. The query is very slow without an index. Use indexes to improve performance and speed up queries.

- The database reads the entire table row by row.
- Happens when:
  - No index exists.
  - The query retrieves most rows (index would be inefficient).

Choosing the Right Scan Strategy

- **Use indexes** to avoid full table scans.
- **Analyze execution plans** before optimizing.
- **Consider composite indexes** for multi-column queries.
- **Use** EXPLAIN ANALYZE to check query performance.

jsupskills

Thank You for Reading!

- Optimize queries with the right indexing strategy.
- Use EXPLAIN ANALYZE to understand query performance.
- Avoid sequential scans for large datasets.
- Keep learning and optimizing!

www.jsupskills.dev