Database Index Documentation - Superstore Database

Overview

Database ini menggunakan **13 indexes** yang telah dioptimasi untuk performa query dengan dataset **500 rows**. Indexes dipilih secara strategis untuk mempercepat operasi JOIN, WHERE, dan ORDER BY yang sering digunakan.

Index Summary

Table	Index Name	Columns	Type	Size	Purpose
customers	customers_pkey	customer_id	PRIMARY KEY	16 kB	Unique identifier
customers	idx_customers_region	region	BTREE	16 kB	Filter by region
products	products_pkey	product_id	PRIMARY KEY	40 kB	Unique identifier
products	idx_products_category	category	BTREE	16 kB	GROUP BY category
orders	orders_pkey	order_id	PRIMARY KEY	16 kB	Unique identifier
orders	idx_orders_customer_id	customer_id	FOREIGN KEY	32 kB	JOIN with customers
orders	idx_orders_order_date	order_date	BTREE	16 kB	Filter by date range
order_details	order_details_pkey	id	PRIMARY KEY	56 kB	Unique identifier
order_details	idx_order_details_order_id	order_id	FOREIGN KEY	40 kB	JOIN with orders
order_details	idx_order_details_product_id	product_id	FOREIGN KEY	56 kB	JOIN with products
order_details	idx_order_details_seller_id	seller_id	FOREIGN KEY	16 kB	JOIN with sellers
sellers	sellers_pkey	seller_id	PRIMARY KEY	16 kB	Unique identifier

Table	Index Name	Columns	Type	Size	Purpose
sellers	idx_sellers_region	seller_region	BTREE	16 kB	Filter by seller region

Total Index Size: ~336 kB

Index Categories

1 Primary Keys (5 indexes)

Automatically created untuk enforce uniqueness dan identity.

```
-- customers
CREATE INDEX customers_pkey ON customers(customer_id);

-- products
CREATE INDEX products_pkey ON products(product_id);

-- orders
CREATE INDEX orders_pkey ON orders(order_id);

-- order_details
CREATE INDEX order_details_pkey ON order_details(id);

-- sellers
CREATE INDEX sellers_pkey ON sellers(seller_id);
```

Performance Impact:

- ☑ O(log n) lookup time
- ☑ Enforce data integrity
- ☑ Essential untuk relationship

---

2 Foreign Key Indexes (4 indexes)

Optimize JOIN operations antar tabel.

```
-- orders → customers
CREATE INDEX idx_orders_customer_id ON orders(customer_id);

-- order_details → orders
CREATE INDEX idx_order_details_order_id ON order_details(order_id);

-- order_details → products
CREATE INDEX idx_order_details_product_id ON order_details(product_id);
```

```sql
-- order_details → sellers
CREATE INDEX idx_order_details_seller_id ON order_details(seller_id);
```

Query yang di-optimasi:

```sql
-- Example: Join 5 tables
SELECT *
FROM order_details d
INNER JOIN orders o ON d.order_id = o.order_id
INNER JOIN customers c ON o.customer_id = c.customer_id
INNER JOIN products p ON d.product_id = p.product_id
INNER JOIN sellers s ON d.seller_id = s.seller_id;
```

Performance Impact:

- ☑ JOIN 10-100x lebih cepat
- ☑ Menghindari full table scan
- ☑ Critical untuk relational queries

---

3 Filter & Aggregation Indexes (4 indexes)

Optimize WHERE clause dan GROUP BY operations.

```sql
-- Filter by customer region
CREATE INDEX idx_customers_region ON customers(region);

-- Group by product category
CREATE INDEX idx_products_category ON products(category);

-- Filter by order date range
CREATE INDEX idx_orders_order_date ON orders(order_date);

-- Filter by seller region
CREATE INDEX idx_sellers_region ON sellers(seller_region);
```

Query yang di-optimasi:

```sql
-- Example 1: Filter by region
SELECT * FROM customers WHERE region = 'East';

-- Example 2: Aggregate by category
SELECT category, SUM(sales)
FROM order_details d
JOIN products p ON d.product_id = p.product_id
```

```sql
GROUP BY category;

-- Example 3: Filter by date range
SELECT * FROM orders
WHERE order_date BETWEEN '2015-01-01' AND '2017-12-31';
```

Performance Impact:

- ☑ WHERE filtering 5-50x lebih cepat
- ☑ GROUP BY lebih efisien
- ☑ Mengurangi data scanning

🚫 Indexes yang Dihapus (Optimasi)

Berikut 8 indexes yang **dihapus** karena redundant atau tidak perlu:

Index Name	Reason for Removal
`idx_order_details_order`	❌ Duplicate dengan `idx_order_details_order_id`
`idx_order_details_product`	❌ Duplicate dengan `idx_order_details_product_id`
`idx_orders_customer`	❌ Duplicate dengan `idx_orders_customer_id`
`idx_order_details_sales`	❌ ORDER BY sales jarang digunakan
`idx_order_details_profit`	❌ ORDER BY profit jarang digunakan
`idx_customers_segment`	❌ Filter by segment tidak kritis
`idx_sellers_rating`	❌ Sort by rating tidak sering
`idx_order_details_order_product`	❌ Composite index redundant

Result:

- 🗃 Storage berkurang 52% (700KB → 336KB)
- 🗃 INSERT/UPDATE overhead berkurang
- ☑ Query performance tetap optimal

📈 Performance Benchmarks

Before Optimization (21 indexes)

```
Total Index Size: ~700 kB
INSERT performance: Slower (update 21 indexes)
SELECT performance: Good
Storage overhead: High
```

After Optimization (13 indexes)

```
Total Index Size: ~336 kB
INSERT performance: Better (update 13 indexes)
SELECT performance: Good (no degradation)
Storage overhead: Medium
```

Best Practices

WHEN TO CREATE INDEX

1. **Foreign Keys** - Always index foreign key columns

```
CREATE INDEX idx_table_fk ON table(foreign_key_column);
```

2. **Frequent WHERE Clauses** - Index columns used in WHERE

```
CREATE INDEX idx_table_col ON table(frequently_filtered_column);
```

3. **JOIN Conditions** - Index columns used in JOIN ON

```
CREATE INDEX idx_table_join ON table(join_column);
```

4. **Large Tables** - Tables dengan > 10,000 rows

WHEN NOT TO CREATE INDEX

1. **Small Tables** - Tables dengan < 1,000 rows
2. **High Write Frequency** - Banyak INSERT/UPDATE/DELETE
3. **Low Cardinality** - Kolom dengan sedikit unique values (contoh: boolean)
4. **Rarely Queried** - Kolom jarang digunakan di WHERE/JOIN
5. **Duplicate Indexes** - Index yang sudah di-cover oleh composite index

---

Maintenance Commands

View All Indexes

```
SELECT
    schemaname,
    tablename,
```

```
    indexname,
    pg_size_pretty(pg_relation_size(indexrelid)) as index_size
FROM pg_indexes
WHERE schemaname = 'public'
ORDER BY tablename, indexname;
```

Check Index Usage

```
SELECT
    schemaname,
    tablename,
    indexname,
    idx_scan as index_scans,
    idx_tup_read as tuples_read,
    idx_tup_fetch as tuples_fetched
FROM pg_stat_user_indexes
WHERE schemaname = 'public'
ORDER BY idx_scan DESC;
```

Drop Unused Index

```
-- Check if index is used
SELECT * FROM pg_stat_user_indexes
WHERE indexrelname = 'index_name' AND idx_scan = 0;

-- Drop if unused
DROP INDEX IF EXISTS index_name;
```

Rebuild Index (if corrupted)

```
REINDEX INDEX index_name;
-- or rebuild all indexes on a table
REINDEX TABLE table_name;
```

📊 Query Performance Tips

1. Use EXPLAIN ANALYZE

Check apakah query menggunakan index:

```
EXPLAIN ANALYZE
SELECT * FROM orders WHERE customer_id = 'CG-12520';
```

Look for:

- ☑ `Index Scan` - Good! Using index
- ❌ `Seq Scan` - Bad! Full table scan

2. Composite Index vs Multiple Indexes

```
-- Single composite index
CREATE INDEX idx_composite ON table(col1, col2);
-- Good for: WHERE col1 = ? AND col2 = ?
-- Good for: WHERE col1 = ?
-- Bad for: WHERE col2 = ?

-- Multiple single indexes
CREATE INDEX idx_col1 ON table(col1);
CREATE INDEX idx_col2 ON table(col2);
-- Good for both columns independently
```

3. Index on Calculations

```
-- ❌ Bad - Index not used
SELECT * FROM orders WHERE EXTRACT(YEAR FROM order_date) = 2017;

-- ☑ Good - Index used
SELECT * FROM orders
WHERE order_date >= '2017-01-01' AND order_date < '2018-01-01';
```

Change Log

Version 1.0 (December 4, 2025)

- ☑ Initial database setup with 21 indexes
- ☑ Added sellers table with 3 indexes
- ☑ Optimized to 13 indexes (removed 8 redundant)
- ☑ Documented all indexes and best practices

Related Files

- `create_tables.sql` - Database schema definition
- `create_indexes.sql` - Original index creation script
- `optimize_indexes.sql` - Index optimization script
- `config.py` - Query functions using indexes
- `app.py` - Streamlit dashboard with optimized queries

📞 Support

Untuk pertanyaan atau optimasi lebih lanjut, review:

1. Query performance dengan `EXPLAIN ANALYZE`
2. Index usage statistics dengan `pg_stat_user_indexes`
3. Table size growth dengan `pg_relation_size()`

Last Updated: December 4, 2025
Database: superstore
Total Rows: 500
Total Indexes: 13
Index Size: ~336 kB