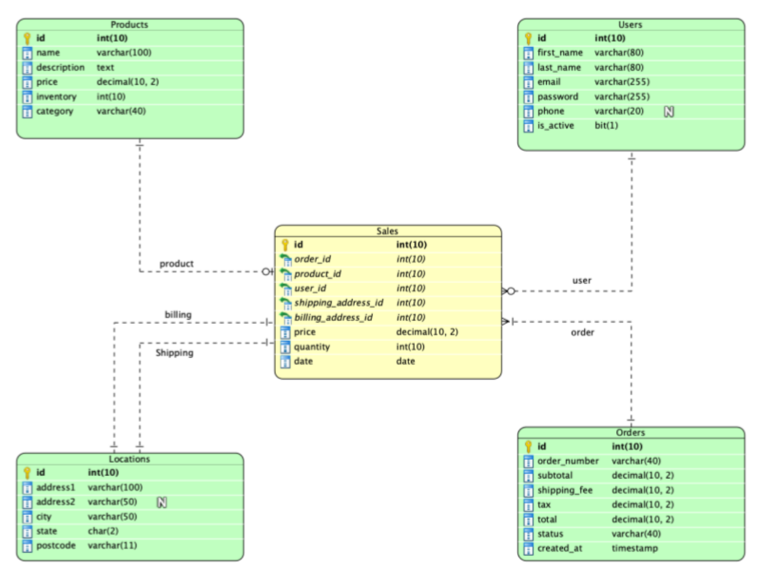
Determine the most efficient query

So far, you have learned about the factors that affect database performance and database query optimization. This is an important part of a BI professional's work because it allows them to ensure that the tools and systems their team is using are as efficient as possible. Now that you're more familiar with these concepts, you'll review an example.

The scenario

Francisco's Electronics recently launched its home office product line on its e-commerce site. After receiving a positive response from customers, company decision-makers chose to add the rest of their products to the site. Since launch, they have received more than 10,000,000 sales. While this is great for the business, such a massive catalog of sales records has affected the speed of their database. The sales manager, Ed, wanted to run a query for the number of sales created after November 1, 2021, in the "electronics" category but was unable to because the database was too slow. He asked Xavier, a BI analyst, to work with the database and optimize a query to speed up the sales report generation.

To begin, Xavier examined the **sales_warehouse** database schema shown below. The schema contains different symbols and connectors that represent two important pieces of information: the major tables within the system and the relationships among these tables.



The **sales_warehouse** database schema contains five tables—Sales, Products, Users, Locations, and Orders—which are connected via keys. The tables contain five to eight columns (or attributes) ranging in data type. The data types include varchar or char (or character), integer, decimal, date, text (or string), timestamp, and bit.

The foreign keys in the Sales table link to each of the other tables. The "product_id" foreign key links to the Products table, the "user_id" foreign key links to the Users table, the "order_id" foreign key links to the Orders table, and the "shipping_address_id" and "billing_address_id" foreign keys link to the Locations table.

Examining the SQL query

After considering the details, Xavier found that the following request needed optimization:

```
SELECT
        COUNT(*) AS number_of_sales
FROM
        Sales AS oi
INNER JOIN
        Products AS p ON p.id = oi.product_id
WHERE
        p.category = 'electronics'
        AND oi.date > '2021-11-01'
```

Optimizing the query

To make this query more efficient, Xavier started by checking if the "date" and "category" fields were indexed. He did this by running the following queries:

```
SHOW
        INDEXES
FROM
        Sales
WHERE
        Column_name = 'date';


SHOW
        INDEXES
FROM
        Products
WHERE
        Column_name = 'category';
```

Without indexes in the columns used for query restrictions, the engine did a full table scan that processed all several million records and checked which ones had date >= "2021-11-01" and category = "electronics."

Then, he indexed the "date" field in the Sales table and the "category" field in the Products table using the following SQL code:

```
CREATE INDEX idx_sales_date ON Sales (date);

CREATE INDEX idx_products_category ON Products (category);
```

Unfortunately, the query was still slow, even after adding the indices. Assuming that there were only a few thousand sales created after "2021-11-01," the query still created a very large virtual table (joining Sales and Products). It had millions of records before filtering out sales with a date after "2021-11-01" and sales with products in "electronics" categories. This resulted in an inefficient and slow query.

To make the query faster and more efficient, Xavier modified it to first filter out the sales with dates after "2021-11-01." The query "(**SELECT** product_id **FROM** Sales **WHERE** date > '2021-11-01') **AS** oi" returned only a few thousand records, rather than millions of records. He then joined these records with the Products table.

Xavier's final optimized query was:

```
SELECT
        COUNT(*) AS number_of_sales
FROM
        (
                SELECT
                        product_id
                FROM
                        Sales
                WHERE
                        date > '2021-11-01'
        ) AS oi
INNER JOIN
        Products AS p ON p.id = oi.product_id
WHERE
        p.category = 'electronics'
```

Key takeaways

Optimizing queries will make your pipeline operations faster and more efficient. In your role as a BI professional, you might work on projects with extremely large datasets. For these projects, it's important to write SQL queries that are as fast and efficient as possible. Otherwise, your data pipelines might be slow and difficult to work with.

Mark as completed

Like Dislike Report an issue