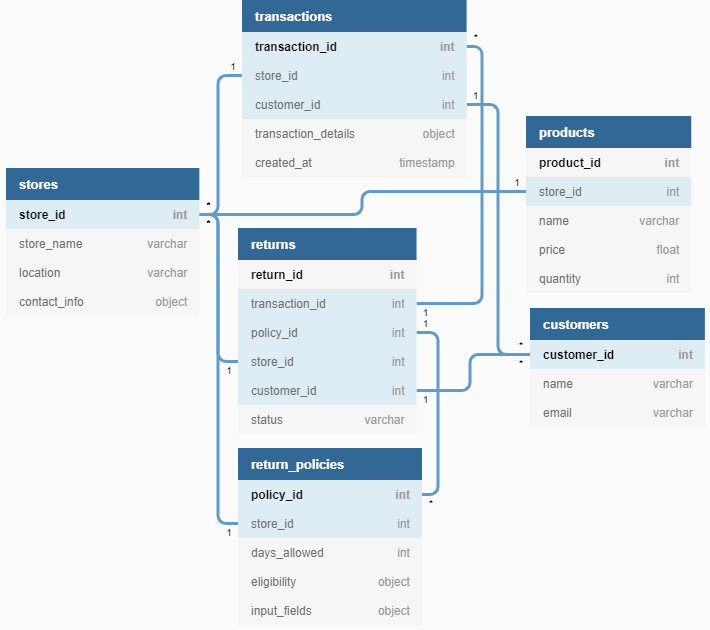
Design a database for a product return application. The application will contain many different stores, each store will have different return policy and product, but all of them require the customer to fill out a return form, which may be slightly different in every store. Your database will need to keep track of all these data, and expected to sufficiently perform the following operations:

* Notify the store about their approved return order so they can refund their customer
* Notify the customer about the status of their return order
* Display the information of a return order for approval process

You can use any type of database, such as SQL and NoSQL. Something good to consider when you design your database would be future scaling and additional operations by new features. Please provide an explanation for each of your decisions.

Note: You do not need to write any code, simply lay down the structure of your database. Write down all your assumptions. If you have any questions, you can make an assumption and write it down.



**Assumptions**

1. Assuming that all tables have a created\_at timestamp whenever a value is inserted. I only specified it in the transaction table to have a clear indication about the number of days after purchase.
2. Database structure works for both SQL and NoSQL. NoSQL databases like MongoDB can take objects as mixed variables. If not, then values that are listed as objects can be converted VARCHAR via JSON string.

**Explanation**

Stores table

* Each store has its own id and information to identify itself

Return policy table

* Each store may have multiple return policies so a reference is attached to locate all active policies
* Return policies will have a date range which the transaction will be eligible for return.
* The ***eligibility*** object will consist of fields that are specified by the store that determines whether products are deemed returnable.
* The ***input\_fields*** object will consist of the fields that the customer will have to fill out for the return.

Transaction table

* Each transaction will have a ***store\_id*** and ***customer\_id*** in order to notify the correct parties.
* The ***transaction\_details*** object will contain product information including varying fields from ***product\_id***to pricing, quantity, on sale, etc.
* The ***created\_at***timestamp is there to specify whether the transaction has exceeded the return period.

Products table

* An inventory containing details of all the products available for the stores to manage and which stores has ownership of the product.

Customers table

* Contains customer information in order to provide updates on the return status.

Returns table

* A list of return requests that has references to the correct return policy and transaction.
* Status will be either one of “pending”, “approved” or “rejected”.

Scalability

* If multiple stores are under one branch manager, the manager’s id can be attached to the stores that are associated.
* The ***store\_id*** value in the products table can be changed to an array of ids to allow multiple stores that own the same product.

Events

* **Notify the store about their approved return order so they can refund their customer:** Once a return has been approved, the store information can be located via the reference to the ***store\_id*** in the returns table.
* **Notify the customer about the status of their return order:** The customer will be able to access the ***status*** of their return through the ***customer\_id*** on the returns table. An email can be sent out when the return request is created “pending” and another can be sent out once it is either “approved” or “rejected”
* **Display the information of a return order for approval process:** Information can be displayed by retrieving all the data through the references in the return table. Everything from customer info to store info, product info and transaction info can be accessible from that data.