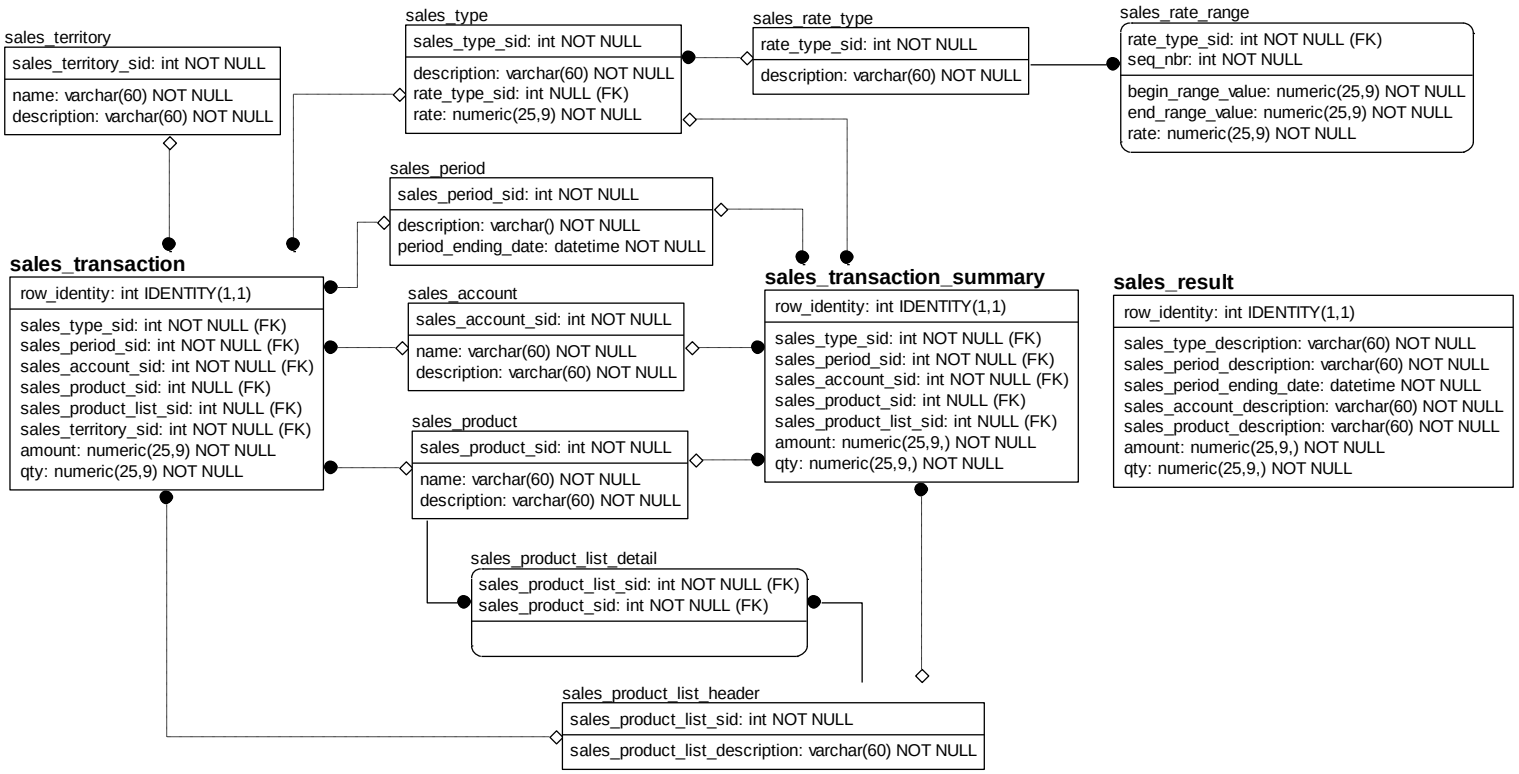
This entity relationship diagram defines some tables and relationships in a relational database.



General Information:

Table "sales_transaction" consists of a row of data for every sales transaction. Each row is made up of columns that define characteristics for each sales transaction. Table "sales_transaction" only allows INSERTs and DELETEs. Data is either Imported (Inserted) into this table, or Cleared (Deleted) from this table. Data is Cleared if additional processing shows that the imported data had mistakes. INSERTS and DELETES happen at different times.

Table "sales_transaction_summary" allows INSERTs, UPDATEs and DELETEs, and consists of data summarized from the "sales_transaction" table. It is summarized based upon several characteristics of each sale. Those characteristics are sales type, sales period, sales account, product and product list. So, if 100 rows exist in table "sales_transaction", and there are only three combinations of sales type, sales period, sales account, sales product and sales product list, then there would be three rows in table "sales_transaction_summary". The values in columns amount and qty would equal the sum of those columns in "sales_transaction", for rows with matching characteristics. This table data only changes when data in table "sales_transaction" changes. In both "sales_transaction" and "sales_transaction_summary", each transaction refers to EITHER one product (when column sales_product_sid is Not Null) OR one product List (when column sales_product_list_sid is Not Null). Only one of these columns (sales_product_sid or sales_product_list_sid) will have a value in any given row. A product list is defined in table "sales_product_list_header", and the products contained in the list are defined in table "sales_product_list_detail", which is associated with table "sales_product" that defines the product.

Assumptions:

Tables "INSERTED" and "DELETED" have the same definition as table "sales_transaction".
Tables "INSERTED" and "DELETED" are automatically populated by SQL Server AFTER the Insert or Delete.
Table "INSERTED" contains new rows that were just Inserted into table "sales_transaction" for an Import.
Table "DELETED" contains rows that were just Deleted from table "sales_transaction" for an Import Clear (Delete).
Updates are not allowed on table "sales_transaction", but are allowed on table "sales_transaction_summary".
Row_Identity columns are system-generated unique, sequential integer values that define the primary key for each particular table. These row_identity columns are not related to row_identity columns in other tables.
SID columns are unique, sequential integer values used as Primary Keys in their respective base table.

In any given row, Sales Product Sid or Sales Product List Sid will have a value, not both.
Zero values and negative values for columns "amount" and/or "qty" in the summary table are valid values.
Temporary tables are ok to use.
Table "sales_transaction" is a very large table, with tens of millions of rows of data.
Table "sales_transaction_summary" is a large table, with several million rows of data.
At any point in time, table "sales_transaction_summary" only contains data for corresponding rows that exist in table "sales_transaction". That is, rows should not exist in table "sales_transaction_summary" if there are no rows in table "sales_transaction" with matching characteristics.
Solving the problems correctly is the number one priority, but coding efficiency is also considered in grading.

Task #1:

Write code (choose a language with which you are familiar) that will keep the "sales_transaction_summary" table synchronized and summarized with the main transaction table "sales_transaction". This code would effectively execute every time data in table "sales_transaction" is Inserted or Deleted. In each Import (Insert) or Clear (Delete) for table "sales_transaction", there may be any number of rows affected. The summarization is based on transaction characteristics, as defined in the General Information section above. Define any assumptions.

Task #2:

Sample Sales Types:

Sales_type_sid	Description	Rate_type_sid	Rate
1	Normal	Null	1.0
2	Sell-through	2	1.1
3	Barter	1	1.2
4	Rental	Null	1.3

Sample Sales Rate Range:

Rate_type_sid	Seq_nbr	Begin_range_value	End_range_value	Rate
1	1	0.1	100.0	1.25
1	2	100.1	500.0	1.35
1	3	500.1	99999.0	1.45

Write code (choose a language with which you are familiar) that will create entries in the sales_result table. This code will be run once per month, for a specific Period. Define any assumptions. Use the following requirements.

- The data going into the sales_result table is primarily based upon the sales transaction summary table.
- Data in "sales_result" should also be summarized based upon the same characteristics as the sales transaction summary table.
- Sales result rows will correspond one-for-one with the sales transaction summary table, except when the summary rows refer to a product list. These list product rows must be exploded. That is, if a product list contains N products, then the one summary row would result in N sales result rows. Each of which would contain the same data except for the product description (from table "sales_product" via associated product list tables), amount and qty columns. The amount and qty columns would be set to their average values by dividing the values in the summary table by the number of products in the product list.
- The description columns and the period ending date column need to be filled with data from their associated tables, based upon their sids in the summary table.
- Finally, all amount values must be multiplied by the rate defined by the sales type. This must be based upon the sales type sid in the summary table. The final rate to be used is determined by the following:
    - If the sales_type table column rate_type_sid has no value (Is Null), then use the rate defined in that table.
    - If the sales_type table column rate_type_sid has a value (is Not Null), then the rate that should be used is selected from table sales_rate_range. The sales_rate_range rate is found first, by matching the rate_type_sid, and then by finding the row in which the sales qty is between the begin_range_value and the end_range_value, inclusive. If no row is found in table sales_rate_range for the rate_type_sid and the qty, then use the rate defined in column rate in the sales_type table.