PetsDumb  
not a petsmart affiliate

February 11, 2018

# Overview

## Project Background and Description

Database Engineer Assignment  
   
I want to open a pet shop and I want to have my own inventory/point of sale system. I plan to sell pets, toys, supplies, and food. Please create a DB definition that will allow me to track purchases, inventory, vendors, holiday sales, and taxes.  
   
BTW, I plan to be extremely successful and have the highest volume pet shop in the state, maybe even start taking internet orders in the future.

## Project Scope

We need an architectural layout of the database with source code, documentation, a diagram, and a plan / writeup for a proof of concept to be used for interview purposes. The code should adhere to best practices and be scalable, the documentation should be informative and consistent while easy to read and navigate for reference, the diagram should be clear and helpful, and the writeup should explain the project in detail.

## High-Level Requirements

**Business Requirements**

* Must track inventory.
* Must be usable for point of sale.
* Must track and sell pets, toys, supplies, and food.
* Vendors must be tracked and associated to products.
* Must track and accommodate for tax.
* Must track and accommodate for holidays and sales.
* Must accept large volumes of data and be scalable to local and national web traffic.

**Architecture**

* Tables must be efficient, enforce data integrity, lack redundancy, and be sensible.

## Deliverables

* **Source Code**
  + SQL Server 2016 Developer edition SQL file: Create\_PetsDumb.sql
* **Documentation**
  + Using Redgate trial software to generate documentation on the PetsDumb architecture: Documentation\_PetsDumb.pdf
* **ERD Diagram**
  + Using Toad Datamodeler trial software to generate a PDF diagram of the PetsDumb Architecture: ERD\_PetsDumb.jpg
* **LoadTest\_PetsDumb.docx**
  + Using Redgate data generator to map out foreign key dependencies and dim tables, I had it generate fake data within the constraints of the database and bench marked performance for as quick of input based on business logic as possible on my laptop. Summary of findings will be included in this report.
* **Project Plan**
  + A summary of the project and walk through of the proof of concept solution: ProjectPlan\_PetsDumb.docx

## In Scope

**Business requirements**

* No RDBMS was mentioned nor inability to use specific toolsets. I have decided to work with SQL Server 2016 and using Toad Datamodeler + Redgate toolbelt to provide an extremely detailed high performing data model and documentation.
* Capture of historical data and data change over time.
* Tax application:
  + Schema and data was provided and created by using the data dump in reference to:
    - <https://www.taxrates.com/download-tax-tables/>
* Zip code application:
  + To apply tax rates correctly, zip code data and schema was generated using this data dump in reference to:
    - <http://federalgovernmentzipcodes.us/>
* Holiday tracking:
  + To apply correct holiday discounts and references, a date table was created to maintain referential integrity with dates and times. They were created in reference to:
    - <http://www.sqlservercentral.com/scripts/Data+Warehousing/65762/>

**Architecture**

* Primary key and foreign key constraints and references should be created.
  + Seed increment and identity must be planned.
  + Smallest data type must be used.
  + Data types, naming conventions, and schema design should adhere to best practice.
* Temporal tables for later analysis.
* Data types should be carefully planned.
* Nulls should be carefully planned.
* Dim tables should utilize columnstore indexes for performance increase.
* Schemas should be used to keep tables compartmentalized.
  + Also to help maintain security.
* Data must be encrypted at rest, TDE AES 256 must be applied.
* Must be able to scale outward for millions of records.

## Not in Scope

**Business Requirements**

* Business logic has not been created. This is an architectural layout and proof of concept design for a POS / Inventory system for Pet Store. There are no triggers / stored procedures / security / file group / data file / partitioning / filestream / xml / json / jobs / HA or DR planning implemented or shown.
* Credit card storage is not implemented.

**Architecture**

* The way the application is designed in coordination to the sql generated would dictate index creation. Therefore, indexes are not created as the application must be designed to design the indexes. The only indexes created are the columnstore for the dim tables since those are rarely changing tables.

## Implementation Summary and Thoughts

I created four dimension tables:

* Date
* TaxRates
* Time
* ZipCode

They are rarely changing. I have populated the TaxRates and ZipCode tables with data dumps online and used a Date and Time generator to generate a dimension table for each. They have all had columnstore indexes created to increase performance.

The tables can accommodate for large amounts of data and all states. It may be advisable to create a database clone for each state and run them in unison that way while partitioning the files and filegroups to different drives. You could just partition the table itself by state via files and filegroups and along with the indexes.

The app schema is the portion the application will interact with the most. I created four app tables:

* HolidayDiscount
* Members
* Purchases
* Vendors

The HolidayDiscount is generated with the Date table.

The inv schema is the portion the backend piece would integrate with the application. I created four tables here:

* Food
* Pets
* Supplies
* Toys

I then generated data in the Members table, Vendors Table, and then began populating the inventory tables. Once I had enough data, I was able to create records in the Purchases table.

I used the Redgate data generator tool trial to create a fake data import. Below is a summary of the speed and quantity of records inserted. This test was ran on my Lenovo Elitebook T410 laptop:

Max memory for SQL Server: 2GB

Intel® Core™ i5-2520M CPU @ 2.50GHz

SSD HD

|  |  |  |
| --- | --- | --- |
| Table | Records | Time |
| HolidayDiscount | 54,788 | 0:00:02 |
| Members | 100000 | 0:00:07 |
| Vendors | 100000 | 0:00:01 |
| Food | 100000 | 0:00:05 |
| Pets | 100000 | 0:00:06 |
| Supplies | 100000 | 0:00:06 |
| Toys | 100000 | 0:00:09 |
| Purchases | 1000000 | 0:01:43 |
| Average | 206,849 | 0:00:17 |
| Total | 1,654,788 | 0:02:19 |

During testing, records were successfully being manipulated through the temporal tables as expected.

On a server with more memory and better hardware, it should handle concurrent high transactional processing while maintaining referential integrity and creating a rich OLAP historical trend to analyze at a later period. It is easy and independent to add or remove additional features or inventory to the database backend.