Database Project

Joshua Grimmett

Version<2.0>

25/10/19

Database Design Structure

Relational database design was chosen as the structure for the client's application. A relational database is a database composed of one or more tables containing data points relating to one another. Data is stored in tables of columns and rows. Columns define the data types and relationships and rows contain instances of data containing the restricting columns.

Relational databases are ideal for relatively small to medium amounts of predictable data to be stored for quick creation, retrieval, updates and deletion. Relational databases do not scale horizontally which leaves them as a suboptimal option for large systems with millions of user accounts but perfect for the client's application.

Database Management System

The chosen database relational database management system is Microsoft SQL Server. Microsoft SQL Server is a toolbox of software which allows management of data with both UI and SQL. Microsoft SQL Server uses a Microsoft designed dialect of SQL, Transact-SQL (T-SQL). This database management system was chosen because of its extensive and detailed developer documentation as well as its quick and easy setup on a windows machine.

Data Normalisation

Initial Data

tbl_stock_names
stock_symbol
stock_name
stock_exchange

- stock_symbol
    - The stock identifier on the stock exchange
- stock_name
    - The company or stock name
- stock_exchange
    - The stock exchange which the stock is listed on

tbl_daily_prices
stock_exchange
stock_symbol
market_date
stock_price_open
stock_price_high
stock_price_low
stock_price_close
stock_volume
stock_price_adj_closed

- stock_exchange
    - The stock exchange which the stock is listed on
- stock_symbol
    - The stock identifier on the stock exchange
- market_date
    - The market date which the stock price was listed
- stock_price_open
    - The stock price at the opening for the market day
- stock_price_high
    - The price of the stock at its highest for the market day
- stock_price_low
    - The price of the stock at its lowest for the market day
- stock_price_close
    - The stock price at the closing of the market day
- stock_volume
    - The volume of the stock
- stock_price_adj_closed
    - The price of the stock after dividends on the closing of the market day

First Normal Form

For the database schema to satisfy the first normal form it must meet the following conditions:

- It must contain only atomic (single) values.
- It must also not contain repeating columns.

An atomic value in a database is a value that cannot be split into multiple columns. The initial data already satisfies this requirement and does not have any non-atomic values.

Repeating columns are a group of columns representing an index for each column and value. This would require a new column for every added stock to the stock list, which would make it very difficult to add any quantity of stocks other than the defined columns. This can be redesigned by creating a separate table for a list of stocks with a key to reference to each stock symbol. The initial data already satisfies this requirement by splitting the stock details and stock list into two separate tables.

Second Normal Form

For the database schema to satisfy the conditions of second normal form it must meet the following requirements:

- Already be in first normal form.
- And, contain no partial dependencies.

A partial dependency is a non-primary column which is dependent on a primary key. The first normal form version of the database design contains no partial dependencies and satisfies the second normal form.

Third Normal Form

For the database schema to satisfy the conditions of third normal form it must meet the following requirements:

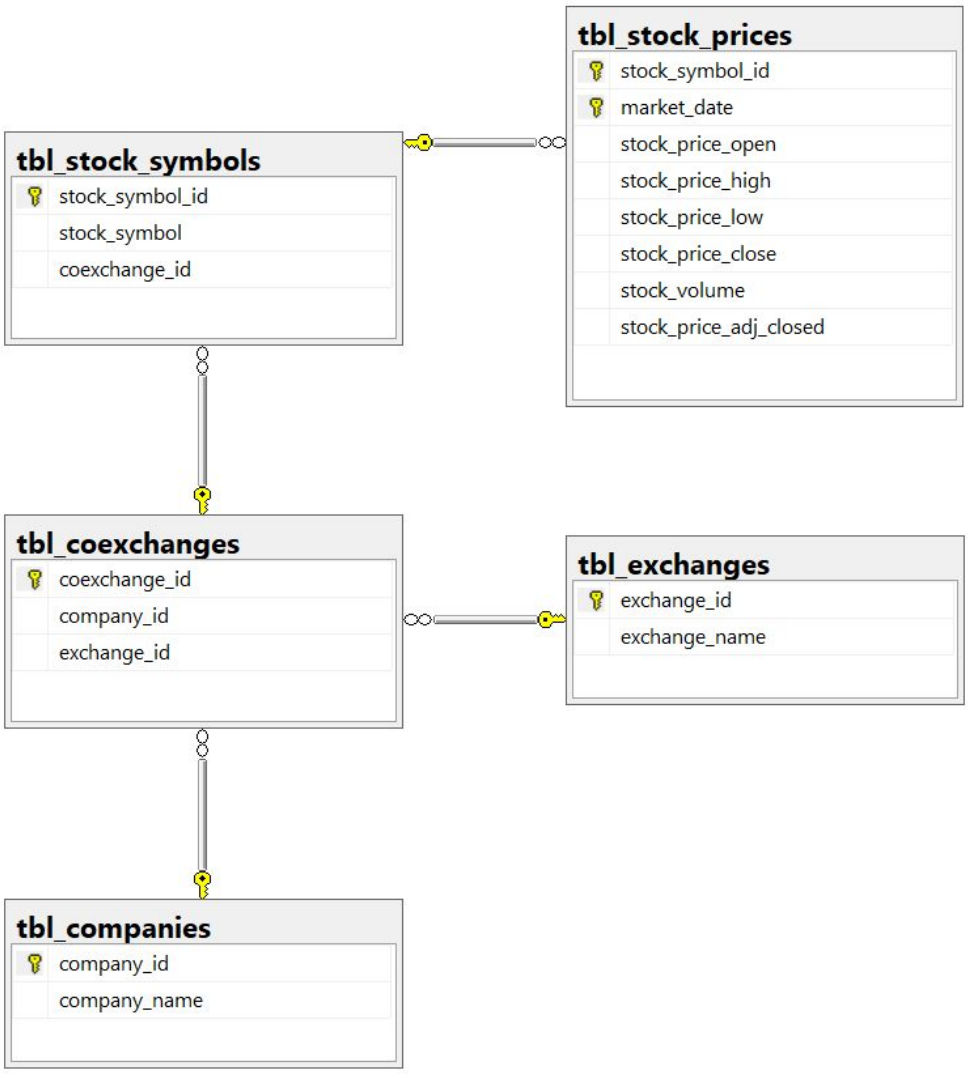
- Already be in second normal form.
- And, contain no transitive dependencies.

A transitive dependency is a non-prime attribute which is dependent on another non-prime attribute. The second normal form version of the database design already satisfies the requirements for third normal form. The second normal form version of the database does not contain any transitive dependencies and therefore is in third normal form.

Final Design

The initial data layout did not support inserting stock from a new stock exchange. It was difficult to update a stock to belong to another stock market or if two stocks with the same symbol on different markets were inserted, it would create an anomaly. To avoid this, the tbl_stock_names table was split into four tables: tbl_stock_symbols, tbl_companies, tbl_exchanges and tbl_coexchanges.

Entity Relationship Diagram



Tables

tbl_stock_symbols

The tbl_stock_symbols table is a table responsible for holding the stock symbols. The table has 3 columns: stock_symbol_id, stock_symbol and coexchange_id. The primary key for this table is the auto-incrementing integer, unique identifier column, stock_symbol_id. The table contains the stock_symbol column as a varchar type containing the stock symbol a.k.a stock ticker. The last column in the table is a foreign key: coexchange_id which is an integer referencing its corresponding coexchange record. The table has a one-to-many relationship with the tbl_stock_prices table relating to the tbl_stock_prices' foreign key, stock_symbol_id. The table also has a one-to-one relationship with tbl_coexchanges.

Field Name	Data Type	Nullable
stock_symbol_id	integer	No
stock_symbol	varchar(15)	No
coexchange_id	integer	No

tbl_exchanges

The tbl_exchanges table is a table containing the stock exchanges. The table only has 2 columns: exchange_id and exchange_name. The primary key is a unique auto-incrementing integer, exchange_id. Each record contains an exchange_name attribute corresponding to the name of the exchange.

Field Name	Data Type	Unique	Nullable
exchange_id	integer	Yes	No
exchange_name	varchar(50)	Yes	No

tbl_companies

The tbl_companies table is a table containing the records for each company. The table only has 2 columns: company_id and company_name. The primary key is a unique auto-incrementing integer, company_id. Each record contains a company_name attribute corresponding to the name of the company.

Field Name	Data Type	Unique	Nullable
company_id	integer	Yes	No
company_name	nvarchar(100)	Yes	No

tbl_coexchanges

The tbl_coexchanges table is a conjoining table between the tbl_companies and tbl_exchanges table. The table contains 3 attributes: coexchange_id, company_id and exchange_id. The primary key is a simple unique auto-incrementing integer, coexchange_id. The table contains two foreign keys. The first foreign key is the company_id attribute, representing a one-to-one relationship with the tbl_companies table. The second foreign key is the exchange_id attribute representing a one-to-one relationship with the tbl_exchanges table.

Field Name	Data Type	Nullable
coexchange_id	integer	No
company_id	integer	No
exchange_id	integer	No

tbl_stock_prices

The tbl_stock_prices table is a table containing the records of stocks across many days. The table has 8 columns: stock_symbol_id, market_date, stock_price_open, stock_price_high, stock_price_low, stock_price_close, stock_volume and stock_price_adj_close. The primary key is a candidate key between the stock_symbol_id column and the market_date column. The tbl_stock_prices table has a many-to-one relationship with the tbl_stock_symbols table, linked by the stock_symbol_id foreign key.

Field Name	Data Type	Date Format	Nullable
stock_symbol_id	integer		No
market_date	date	D/M/Y	No
stock_price_open	float		No
stock_price_high	float		No
stock_price_low	float		No
stock_price_close	float		No
stock_volume	integer		No
stock_price_adj_close	float		No

Table Creation (SQL)

Database Creation

```
CREATE DATABASE stock_exchange;
```

Tables Creation

```
USE [stock_exchange];

CREATE TABLE dbo.tbl_companies (
	company_id int IDENTITY (1,1) NOT NULL,
	company_name nvarchar(100) UNIQUE NOT NULL,
	PRIMARY KEY (company_id)
);

CREATE TABLE dbo.tbl_exchanges (
	exchange_id int IDENTITY (1,1) NOT NULL,
	exchange_name varchar(50) NOT NULL UNIQUE,
	PRIMARY KEY (exchange_id)
);

CREATE TABLE dbo.tbl_coexchanges (
	coexchange_id int IDENTITY (1,1) NOT NULL,
	company_id int NOT NULL,
	exchange_id int NOT NULL,
	PRIMARY KEY (coexchange_id),
	CONSTRAINT fk_coexchange_company FOREIGN KEY (company_id)
	REFERENCES tbl_companies(company_id),
	CONSTRAINT fk_coexchange_exchange FOREIGN KEY (exchange_id)
	REFERENCES tbl_exchanges(exchange_id)
);

CREATE TABLE dbo.tbl_stock_symbols (
	stock_symbol_id int IDENTITY(1,1) NOT NULL,
	stock_symbol varchar(15) NOT NULL,
	coexchange_id int NOT NULL,
	PRIMARY KEY (stock_symbol_id),
```

```
	CONSTRAINT fk_symbol_coexchange FOREIGN KEY (coexchange_id)
	REFERENCES dbo.tbl_coexchanges(coexchange_id)
);

CREATE TABLE dbo.tbl_stock_prices (
	stock_symbol_id int NOT NULL,
	market_date date NOT NULL,
	stock_price_open float NOT NULL,
	stock_price_high float NOT NULL,
	stock_price_low float NOT NULL,
	stock_price_close float NOT NULL,
	stock_volume int NOT NULL,
	stock_price_adj_closed float NOT NULL,
	PRIMARY KEY (stock_symbol_id, market_date),
);
```

Tables Population (SQL)

Two comma-separated values (CSV) files were provided to populate the tables: NYSE_daily_prices_A.csv and NYSE_stock_names.csv, containing the stock prices and the stock names, respectively. A problem occurred when trying to insert values from CSV, caused by values containing commas in the data. To avoid this, it was decided that the files would be converted to a tab-separated values (TSV) format.

The files were converted with a custom python CLI tool: format_csv.py. The following commands were executed to convert the files:

```
python format_csv.py NYSE_daily_prices_A.csv -f=TAB
-o=NYSE_daily_prices_A.tsv
```

```
python format_csv.py NYSE_stock_names.csv -f=TAB -o=NYSE_stock_names.tsv
```

To populate the existing tables in the database the TSV values were inserted into temporary tables: tbl_stock_names_temp and tbl_daily_prices_temp. They can be created with the following SQL:

```
CREATE TABLE tbl_stock_names_temp (
```

```
	stock_symbol varchar(5) NOT NULL,
	stock_name varchar(100) NOT NULL,
	stock_exchange varchar(15) NOT NULL,
	PRIMARY KEY (stock_symbol)
);

CREATE TABLE tbl_daily_prices_temp (
	stock_exchange char(4) NOT NULL,
	stock_symbol char(3) NOT NULL,
	market_date varchar(10) NOT NULL,
	stock_price_open float NOT NULL,
	stock_price_high float NOT NULL,
	stock_price_low float NOT NULL,
	stock_price_close float NOT NULL,
	stock_volume int NOT NULL,
	stock_price_adj_close float NOT NULL,
);
```

Then 2 bulk insert commands can be executed to insert the TSV values into the temporary tables. The from command must be modified to contain a valid path to the TSV files. The values are then transferred from the temporary tables into the final tables and then the temporary tables are deleted.

```
BULK INSERT tbl_stock_names_temp
FROM 'D:\temp\NYSE_stock_names.tsv'
WITH (
	FIRSTROW = 2,
	FIELDTERMINATOR = '\t',
	ROWTERMINATOR = '\n',
	TABLOCK
);

BULK INSERT tbl_daily_prices_temp
FROM 'D:\temp\NYSE_daily_prices_A.tsv'
WITH (
	FIRSTROW = 2,
	FIELDTERMINATOR = '\t',
	ROWTERMINATOR = '\n',
	TABLOCK
);
```

```
INSERT INTO tbl_exchanges (exchange_name)
SELECT DISTINCT stock_exchange as exchange_name from tbl_stock_names_temp;

INSERT INTO tbl_companies (company_name)
SELECT      stock_name AS [company_name] FROM tbl_stock_names_temp;

INSERT INTO tbl_coexchanges (company_id, exchange_id)
SELECT      tbl_companies.company_id, tbl_exchanges.exchange_id
FROM  tbl_companies, tbl_exchanges, tbl_stock_names_temp
WHERE tbl_companies.company_name = tbl_stock_names_temp.stock_name AND
            tbl_exchanges.exchange_name =
tbl_stock_names_temp.stock_exchange;

INSERT      INTO tbl_stock_symbols (stock_symbol, coexchange_id)
SELECT      tbl_stock_names_temp.stock_symbol,
tbl_coexchanges.coexchange_id
FROM  tbl_stock_names_temp, tbl_coexchanges
WHERE tbl_stock_names_temp.stock_name = (
                  SELECT tbl_companies.company_name
                  FROM tbl_companies
                  WHERE tbl_companies.company_id =
tbl_coexchanges.company_id
            );

SET DATEFORMAT dmy;

INSERT INTO tbl_stock_prices (
      stock_symbol_id,
      market_date,
      stock_price_open,
      stock_price_high,
      stock_price_low,
      stock_price_close,
      stock_price_adj_closed,
      stock_volume
)
SELECT      stock_symbol_id,
            CONVERT(date, market_date) AS market_date,
            stock_price_open,
            stock_price_low,
            stock_price_high,
```

```
            stock_price_close,
            stock_price_adj_close,
            stock_volume
FROM  tbl_daily_prices_temp as dp, tbl_stock_symbols as sy
WHERE       dp.stock_symbol = sy.stock_symbol;

DROP TABLE tbl_stock_names_temp;

DROP TABLE tbl_daily_prices_temp;
```