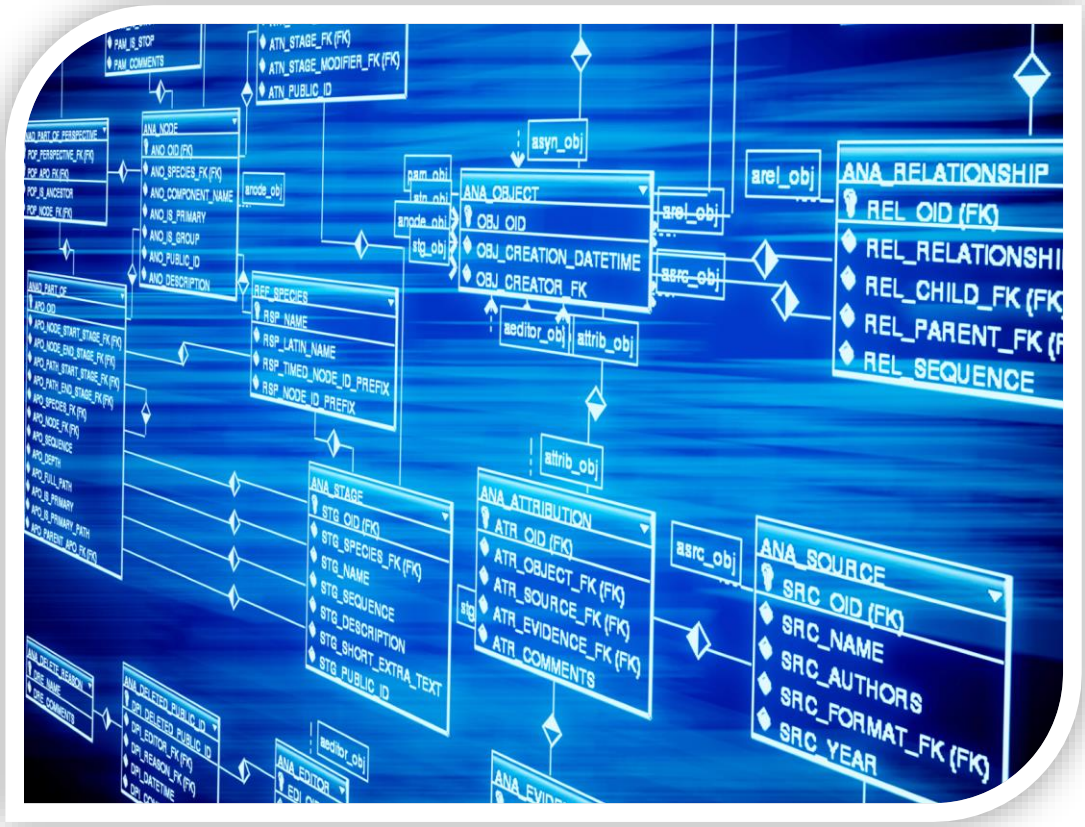
B2B DBMS Model



Doug Perez
Eloy Systems
www.linkedin.com/in/dougperez
www.eloysystems.com
doug.perez@gmail.com

Executive Summary

This project is an OLTP backend database for a B2B e-commerce platform. Included herein is a description of the data model, the implementation of the database, its features, and a discussion of the implementation of the requirements given with the assignment. For reference, please also see the B2BDBMSProject_ERDiagram.pdf for a visual representation of the data model, entities, and attributes.

The model is implemented with Microsoft SQL Server version 2012. A PostgreSQL version of the project is approximately 80% complete.

Contents

Design Conventions

The following section is an overview of the various design conventions that were adopted for this project. They give an overall structure to the design and function of the system and are helpful to understand prior to digesting the details of the data model and database features.

Object Naming

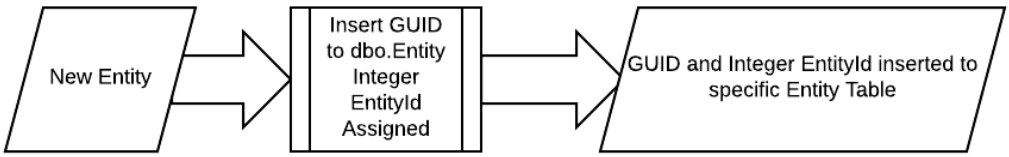
Objects are named according to the following conventions:

- Object names are singular, i.e. Order rather than Order***s.***
- Object names use CamelCase naming where each distinct word in a name is capitalized, but the entire object is named without space or special character interruption.
- Procedure and function names begin with the action. For example, a procedure to add a record to the database begins with 'Add' while a function that gets a list of subsidiaries for a company would begin with 'Get'.
- Views begin with the prefix 'vw' to differentiate them from tables in SELECT statements.
- Given the limited scope of the project, all objects were created within the dbo schema.
- All tables use an integer as a primary key for maximum performance, and all primary key columns are named <TableName>Id, for example AddressId, unless the Id value is part of the entity model described next.
- Index names include a prefix for type of index (XU for Unique, XNC for Nonclustered, etc.), the Table name being indexed, and each of the columns indexed.

Entity Model

Four key components of the data model are the Customer, Company, Subsidiary, and User. Per the requirements, the first three need to be uniquely identified and they share several demographic attributes such as address and email. The fourth attribute was added as a design decision in constructing the database. To maintain the use of Integer primary keys, the four components must share a single set of unique integers.

Therefore, before a record can be inserted in any of the four tables above, they must first be assigned a Globally Unique Identifier (GUID) and it must be assigned a unique integer to be used within the system. This process is achieved through the use of Instead of Insert triggers on dbo.Company, dbo.Subsidiary, dbo.Customer, and dbo.User.



Referential Integrity and Deletions

The entity model described above is one way the database reflects a fully normalized OLTP design. Rather than have redundant table entries for Company and Customer email, a single agnostic email table can service any entity that uses an email. Likewise, entities are free to have as many of these demographic entries as needed, rather than being limited to a single address or two phone numbers, thereby fully realizing the benefits of a relational database model.

As this database implementation is done for demonstration purposes and there is no information given about the application the database will interact with, relationships are enforced strictly using foreign and unique keys. It is very common for tables to have multiple foreign key references so that the use of the database reflects a sound B2B model per the requirements.

All database tables are assigned an Instead of Delete trigger which sets a field called IsDeleted = 1 rather than physically deleting the record from the database. This soft delete convention is useful for maintaining referential integrity in the records left behind after a delete.

Database Objects

In this section, each database object will be described in detail along with references and implementation notes.

Tables

All Tables

These fields and their use are common to all tables in the database.

Column Name	Data Type	Key(s)	Notes
IsDeleted	BIT		Flag for soft-deleting records
InsertDateTime	DATETIME		Date and time record was inserted
InsertUser	INT	FK	Foreign Key to dbo.User
LastUpdateDateTime	DATETIME		Date and time record was last updated
LastUpdateUser	INT	FK	Foreign Key to dbo.User

dbo.Address

All physical address records are stored in this table for any entity, Customer, Company, Subsidiary, or User.

Column Name	Data Type	Key(s)	Notes
AddressId	INT	PK	Auto-incrementing integer
EntityId	INT	FK	Foreign Key to dbo.Entity
AddressTypeId	INT	FK	Foreign Key to dbo.AddressType
AddressLine1	VARCHAR(200)		
AddressLine2	VARCHAR(200)		
City	VARCHAR(100)		
State	VARCHAR(2)		
PostalCode	VARCHAR(20)		
Country	VARCHAR(100)		

XU_Address_EntityId_AddressTypeId is a unique constraint allowing only one address type record per entity.

dbo.AddressType

This table holds the address type values that differentiate address records such as ‘Home’, ‘Office’, ‘Mailing’, etc.

Column Name	Data Type	Key(s)	Notes
AddressTypeId	INT	PK	Auto-incrementing Integer
AddressTypeDescription	VARCHAR(200)		Description of the address type

dbo.Company

This table stores the primary Company data. This is one of the key entities listed in the requirements.

Column Name	Data Type	Key(s)	Notes
CompanyEntityId	INT	PK	Also FK to dbo.Entity
CUIT	UNIQUEIDENTIFIER	FK	GUID FK to dbo.Entity
CompanyTypeId	INT	FK	Foreign Key to dbo.CompanyType
ActivityStartDate	DATETIME		Date and time company began business
Website	VARCHAR(2000)		Web address for the company

XU_Company_CompanyEntityID_CUIT enforces a unique constraint on the combination of EntityId and CUIT.

dbo.CompanyPrice

This table serves as a mapping between a company and a supplier's price. It allows the company to choose a particular product from a specific supplier, then set its own price for resale to end users.

Column Name	Data Type	Key(s)	Notes
CompanyPriceId	INT	PK	Auto-Incrementing integer
CompanyEntityId	INT	FK	Foreign Key to dbo.Company
SupplierProductId	INT	FK	Foreign Key to dbo.SupplierProduct
CompanyPricePerUnit	MONEY		The amount the company will charge.

XU_CompanyPrice_CompanyEntityId_SupplierProductId enforces a unique constraint between a company and a single supplier product.

dbo.CompanyType

This table holds the company type values that differentiate companies according to 'Company' and 'Supplier'. This convention allows both types to share the same table.

Column Name	Data Type	Key(s)	Notes
CompanyTypeId	INT	PK	Auto-incrementing Integer
CompanyTypeDescription	VARCHAR(200)		Description of the customer type

dbo.Customer

This table stores the primary Customer data. This is one of the key entities listed in the requirements.

Column Name	Data Type	Key(s)	Notes
CustomerEntityId	INT	PK	Also FK to dbo.Entity
COIT	UNIQUEIDENTIFIER	FK	Foreign Key to dbo.Entity
DocumentNumber	VARCHAR(200)		Identifying number given in requirements.
DocumentTypeId	INT	FK	Foreign Key to dbo.DocumentType
FullName	VARCHAR(200)		Customer's full name per requirements
DateOfBirth	DATE		Date of birth per requirements
DiscountPercent	NUMERIC(5,3)		A number between 0 and 1 representing any discount given to the customer

XU_Customer_CustomerEntityId_COIT enforces a unique constraint on the combination of EntityId and COIT.

dbo.DocumentType

This table holds the Document type values that differentiate identifying documents such as 'Driver License' and 'Passport'.

Column Name	Data Type	Key(s)	Notes
DocumentTypeId	INT	PK	Auto-incrementing Integer
DocumentTypeDescription	VARCHAR(200)		Description of the document type

dbo.Email

All email records are stored in this table for any entity, Customer, Company, Subsidiary, or User.

Column Name	Data Type	Key(s)	Notes
EmailId	INT	PK	Auto-incrementing integer
EntityId	INT	FK	Foreign Key to dbo.Entity
EmailTypeId	INT	FK	Foreign Key to dbo.EmailType
Email	VARCHAR(1000)		The email address

XU_Email_EntityId_EmailTypeId is a unique constraint allowing only one email type record per entity.

dbo.EmailType

This table holds the Email type values that differentiate Email records such as 'Home', 'Business', 'Other', etc.

Column Name	Data Type	Key(s)	Notes
EmailTypeId	INT	PK	Auto-incrementing Integer
EmailTypeDescription	VARCHAR(200)		Description of the Email type

dbo.Entity

This table is the centerpiece of the entity model. It allows Customer, Subsidiary, Company, and User records to share a common set of unique integer identifiers while preserving their globally unique identifiers, which may originate in a different system. By assigning these records all a unique integer identity, common demographic tables such as address, email, and telephone may be re-used, and each entity may have a virtually unlimited number of relationships to these tables.

Column Name	Data Type	Key(s)	Notes
EntityId	INT	PK	Auto-incrementing Integer
EntityTypeId	INT	FK	Foreign Key to dbo.EntityType
EntityNativeId	UNIQUEIDENTIFIER		The GUID uniquely identifying the entity globally.

XU_Entity_EntityId_EntityNativeId enforces a unique assignment of the Integer EntityId with the GUID EntityNativeId.

XU_Entity_EntityNativeId is an additional unique check on the Native ID to make sure that if any values are imported from an outside system, there is no duplication of GUID values.

dbo.EntityType

This table holds the Entity type values that differentiate Entity records such as 'Customer, 'Subsidiary, 'Company', etc.

Column Name	Data Type	Key(s)	Notes
EntityTypeId	INT	PK	Auto-incrementing Integer
EntityTypeDescription	VARCHAR(200)		Description of the Entity type

dbo.Order

This table stores the Order header data containing key order data per the requirements

Column Name	Data Type	Key(s)	Notes
OrderId	INT	PK	Auto-incrementing integer
CustomerEntityId	INT	FK	Foreign Key to dbo.Customer
SubsidiaryEntityId	INT	FK	Foreign Key to dbo.Subsidiary. Key listed requirement
OrderDate	DATETIME		Date order was placed
ShippingAddressId	INT	FK	Foreign Key to dbo.Address
OrderNotes	VARCHAR(MAX)		Notes about the order
OrderStatusId	INT	FK	Foreign Key to dbo.OrderStatus
TotalPrice	MONEY		Sum of amount of all order line items in dbo.OrderItem.

dbo.OrderItem

This table stores the Order line item data containing specifics about items ordered.

Column Name	Data Type	Key(s)	Notes
OrderItemId	INT	PK	Auto-incrementing integer
OrderId	INT	FK	Foreign Key to dbo.Order
CompanyPriceId	INT	FK	ForeignKey to dbo.CompanyPrice
OrderLineNumber	INT		The specific line number of the order
UnitsOrdered	NUMERIC(5,3)		Number of units ordered
ListPrice	MONEY		The base price of the item
DiscountPrice	MONEY		Price after customer discount percent applied
OverridePrice	MONEY		A manual override price, if needed

XU_OrderItem_OrderId_CompanyPriceId enforces a unique constraint that each product type may be listed only once per order.

dbo.OrderStatus

This table holds the Order Status records that define the current state of the order such as 'New', 'Processing', 'Shipped', etc.

Column Name	Data Type	Key(s)	Notes
OrderStatusId	INT	PK	Auto-incrementing Integer
OrderStatusDescription	VARCHAR(200)		Description of the Order Status

dbo.Product

Table containing base product-level information.

Column Name	Data Type	Key(s)	Notes
ProductId	INT	PK	Auto-incrementing integer
ProductDescription	VARCHAR(200)		Description of the product
ProductSKU	VARCHAR(50)		Stockkeeping Unit of the product

dbo.Subsidiary

This table stores the primary Subsidiary data. This is one of the key entities listed in the requirements.

Column Name	Data Type	Key(s)	Notes
SubsidiaryEntityId	INT	PK	Also FK to dbo.Entity
SUIT	UNIQUEIDENTIFIER	FK	Foreign Key to dbo.Entity
CompanyEntityId	INT	FK	Foreign Key to dbo.Company
Nickname	VARCHAR(200)		Nickname of the company, per the requirements

XU_Subsidiary_SubsidiaryEntityId_SUIT enforces a unique constraint on the combination of EntityId and SUIT.

dbo.SupplierProduct

Mapping table between supplier and product allowing a supplier to define its price and own SKU.

Column Name	Data Type	Key(s)	Notes
SupplierProductId	INT	PK	Auto-Incrementing integer
SupplierEntityId	INT	FK	Foreign Key to dbo.Company (Supplier Type)
ProductId	INT	FK	Foreign Key to dbo.Product
UnitTypeId	INT	FK	Foreign Key to dbo.Unit
SupplierPricePerUnit	MONEY		Supplier charge per unit of product
SupplierSKU	VARCHAR(50)		Supplier's own stockkeeping unit

XU_SupplierProduct_SupplierEntityId_ProductId_UnitTypeId enforces a unique constraint among a supplier, product, and unit type, ensuring only one price per product at a time.

dbo.Telephone

All telephone records are stored in this table for any entity, Customer, Company, Subsidiary, or User.

Column Name	Data Type	Key(s)	Notes
Telephone	INT	PK	Auto-incrementing integer
EntityId	INT	FK	Foreign Key to dbo.Entity
TelephoneTypeId	INT	FK	Foreign Key to dbo.TelephoneType
TelephoneNumber	VARCHAR(20)		The telephone number

XU_Telephone_EntityId_TelephoneTypeId is a unique constraint allowing only one telephone type record per entity.

dbo.TelephonelType

This table holds the Telephone type values that differentiate Email records such as ‘Home’, ‘Work’, ‘Fax’, etc.

Column Name	Data Type	Key(s)	Notes
TelephoneTypeId	INT	PK	Auto-incrementing Integer
TelephoneTypeDescription	VARCHAR(200)		Description of the Telephone type

dbo.UnitlType

This table holds the Unit type values that differentiate Email records such as ‘Unit’, ‘kg’, ‘lb’, etc.

Column Name	Data Type	Key(s)	Notes
UnitTypeId	INT	PK	Auto-incrementing Integer
UnitTypeDescription	VARCHAR(200)		Description of the Unit type

dbo.User

This table stores the primary User data used to track insert and update users and other security features.

Column Name	Data Type	Key(s)	Notes
UserEntityId	INT	PK	Also FK to dbo.Entity
USIT	UNIQUEIDENTIFIER	FK	Foreign Key to dbo.Entity
LastName	VARCHAR(100)		User’s last name
FirstName	VARCHAR(100)		User’s first name
Username	VARCAR(50)		Username for logging into system
FirstActiveDate	DATE		Date account became active

XU_User_UserEntityId_USIT enforces a unique constraint on the combination of UserEntityId and USIT.

Stored Procedures

All tables in the database with the exception of dbo.Entity are given a stored procedure to add values, one record at a time, to the database. Entity is not given such a procedure because of the tight entity model relationship and its associated triggers described above. The naming convention for these procedures is dbo.Add<TableName> and the arguments are the values to be inserted into the table for each record type.

In addition, there are 2 ‘Get’ stored procedures and 1 additional ‘Add’ stored procedure that satisfy specific project requirement and are discussed in detail below.

Functions

The database contains both table-valued and scalar-valued functions to pull lists or individual values quickly without having to recall all of the given relationships. All functions are named beginning with 'Get' and then what data they return. The list of functions follows:

Function Name	Table or Scalar	Input Parameter(s)	Returns
dbo.GetAddressFromEntityId	Table	@EntityId INT	Table with all addresses for given entity
dbo.GetEmailFromEntityId	Table	@EntityId INT	Table with all email addresses for given entity
dbo.GetProductsFromOrderId	Table	@OrderId INT	Table with all products ordered given OrderId
dbo.GetSubsidiariesByCompany	Table	@CompanyEntityId INT	Table with all of a company's subsidiaries given CompanyEntityId
dbo.GetTelephoneFromEntityId	Table	@EntityId INT	Table with all addresses for given entity
dbo.GetCOITFromCustomerEntityId	Scalar	@CustomerEntityId INT	UNIQUEIDENTIFIER of COIT
dbo.GetCOITFromDocumentNumber	Scalar	@DocumentNumber VARCHAR(200)	UNIQUEIDENTIFIER of COIT
dbo.GetCUITFromCompanyEntityId	Scalar	@CompanyEntityId INT	UNIQUEIDENTIFIER of CUIT
dbo.GetCUITFromSubsidiaryEntityId	Scalar	@SubsidiaryEntityId INT	UNIQUEIDENTIFIER of CUIT
dbo.GetDiscountPrice	Scalar	@CompanyPriceId INT, @OrderId INT	MONEY discount price of order item
dbo.GetListPrice	Scalar	@CompanyPriceId INT	MONEY base price of order item
dbo.GetOrderTotal	Scalar	@OrderId INT	MONEY total of all line items of Order

Views

A handful of friendly views were added to list some of more basic types of data retrieved with full names and descriptors, rather than integer identifiers. The views created are:

ViewName	Description
vwEntityAddress	Lists all entities with descriptive names and physical addresses
vwEntityEmail	Lists all entities with descriptive names and email addresses
vwEntityTelephone	Lists all entities with descriptive names and telephone number and type
vwOrderDetails	Descriptive view of orders and all line items