Converting the statement of work to ERD

# Requirements

* An Entity Relationship diagram (ERD) for the lab project needs to match the statement of work, including the identified entities, attributes, and relationships.
* Need to have 5 entities with attributes.
* Each attribute needs to belong to correct entity. For example, book title is an attribute of a book, and it should not be placed in the author entity.
* Specify the data type for each attribute (numeric, varchar, etc.)
* Each entity needs a numeric primary key. Follow the primary key guidelines in Chapter 5 of the Coronel book.
* All relationships are one-to-many. No need to implement the generalization hierarchies in this project.
* Many-to-many relationships need to be resolved with an associative entity.
* No redundant relationships. (The relationships don’t form a circle).
* Specify if the cardinality is mandatory or optional for each side of each relationship.

The step-by-step example below shows an approach to convert the sample lab project statement of work to ERD.

# Statement of Work from Sample Lab Project

This database supports the Book Inventory application and keeps track of books, their prices, authors, publishers, and the available number of copies of every book at each store. The purpose of the database is to support the customers with buying a book at the most convenient location and at the best price, and to support the managers with book orders for the store.

The book information includes the title, publisher, author, edition, hard cover, and ISBN. The store information includes the store name, street address, zip code, and manager’s name. For each author, the database keeps track of first, last name, nickname, and date of birth. The zip code table includes city and state. An inventory tracks the book price and the number of available copies at each store.

For this application, each book must have an author. If there is more than one author, the database stores only the first author’s name. Each author could have written zero or more books. A book can be available at zero or many stores, and each store could sell zero or more books. A zipcode location may have zero or more stores, and each store must have a location.

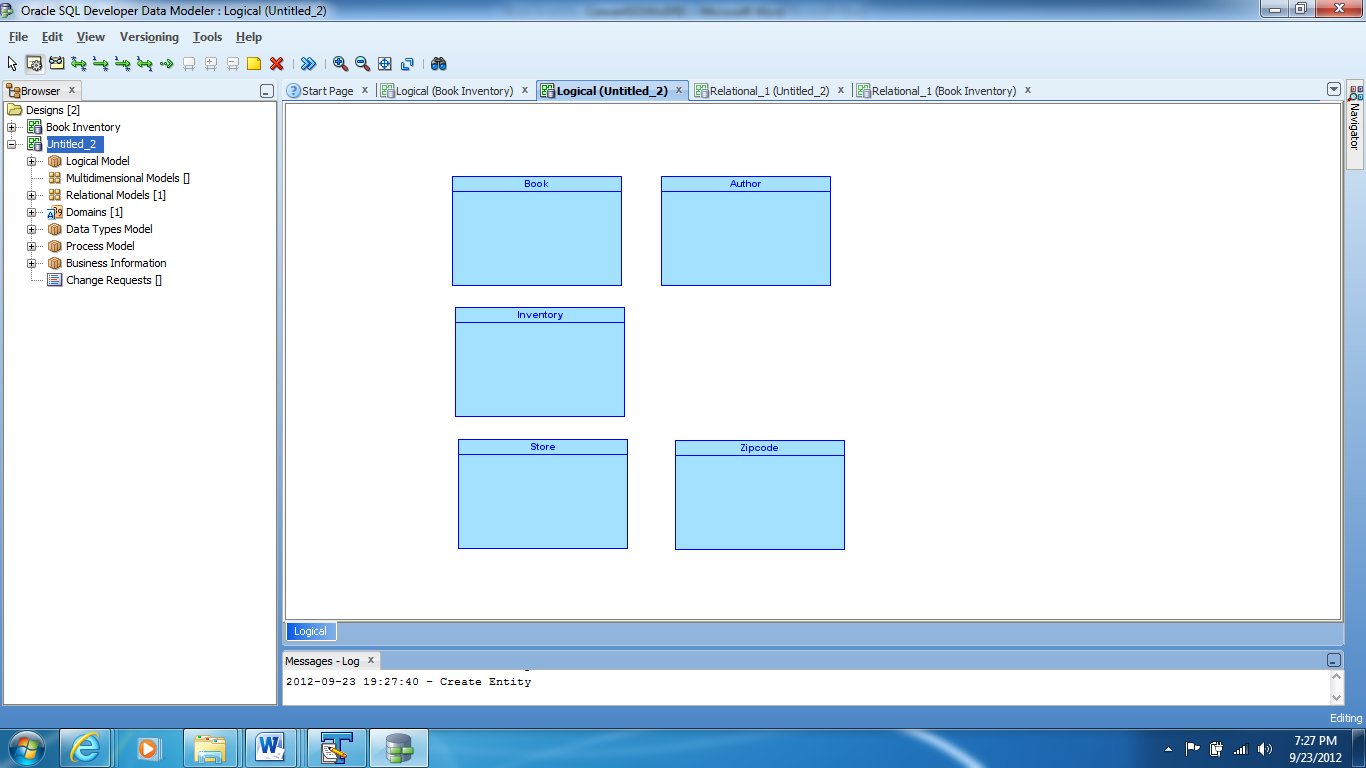
## Step 1 – Identify the entities

The entity names are highlighted in yellow: book, store, author, zip code, and inventory. Each entity becomes a separate box on an ERD diagram, and the top line in the box shows an entity name.

The **book** information includes the title, publisher, author, edition, hard cover, and ISBN. The **store** information includes the store name, street address, zip code, and manager’s name. For each **author**, the database keeps track of first, last name, nickname, and date of birth. The **zip code** table includes city and state. An **inventory** tracks the book price and the number of available copies at each store.

Figure 1 shows a box of each entity identified above. The top line in each box displays an entity name. The rest of the box is empty because we have not added the attributes yet. The entities are not connected because we have not defined the relationships.

Entity name



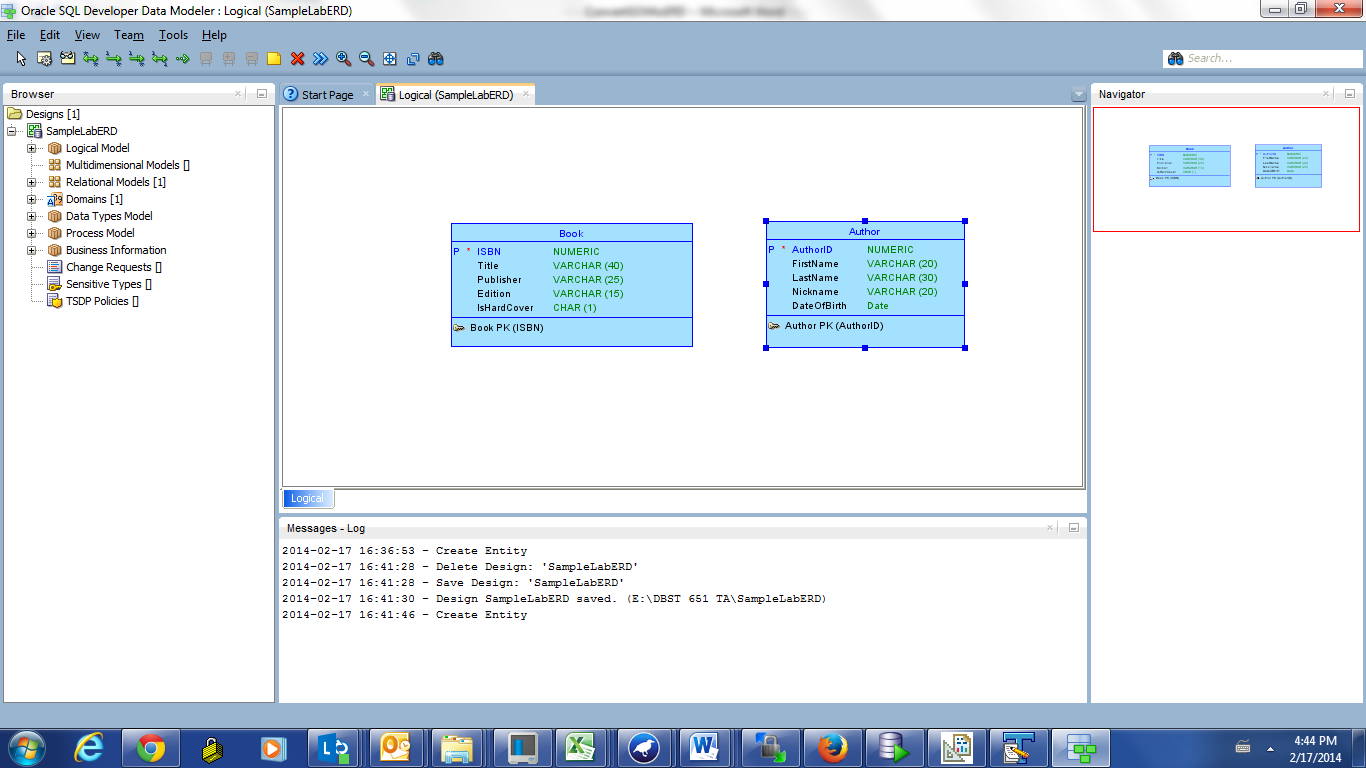
The attributes will be added to this area during the next step

Figure 1: Entities in Book Inventory statement of work

## Step 2 – Identify the attributes

For each entity highlighted in the step above, identify the list of characteristics. An attribute is an entity characteristic. For instance, title and publisher are the characteristics of a book.

Figure 2 shows the elements added to the book entity during the step 2.



Data types

Primary key constraint

Letter P indicates that an attribute is a primary key

Attributes

Figure 2: Book entity before the relationships are added

Table 1 shows an approach to add the attributes entities in the book inventory sample lab project.

|  |  |
| --- | --- |
| The book information includes the title, publisher, author, edition, hard cover, and ISBN.  Since ISBN identifies each book uniquely, and ISBN is numeric, we may choose this attribute for the primary key. |  |
| For each **author**, the database keeps track of first name, last name, nickname, and date of birth.  The name attributes do not identify an author uniquely, and the names are subject to change. Date of birth is not a unique identifier because more than one author can be born on the same day. Hence, we add a numeric attribute author\_id. |  |
| The store information includes the store name, street address, zip code, and manager’s name.  We add a numeric primary key StoreID.  We do not add zipcode at this point because the zipcode is a separate entity. The software will add the foreign key for us when we create the relationship between store and zipcode entities. |  |
| Each zipcode includes city and state.  For this exercise, assume that each city is located at only one zipcode. |  |
| An inventory tracks the book price and the number of available copies at each store.  We add an InventoryID primary key. Since book and store data are tracked in a different entity, we do not add ISBN and StoreID. The software will add those attributes for us once we define the relationships. |  |

Table 1: Adding attributes to entities

## Step 3 – Identify and create the relationships

The number of relationships=number of entities -1

If the book inventory has 5 entities, the diagram has 4 relationships, and each relationship is one-to-many.

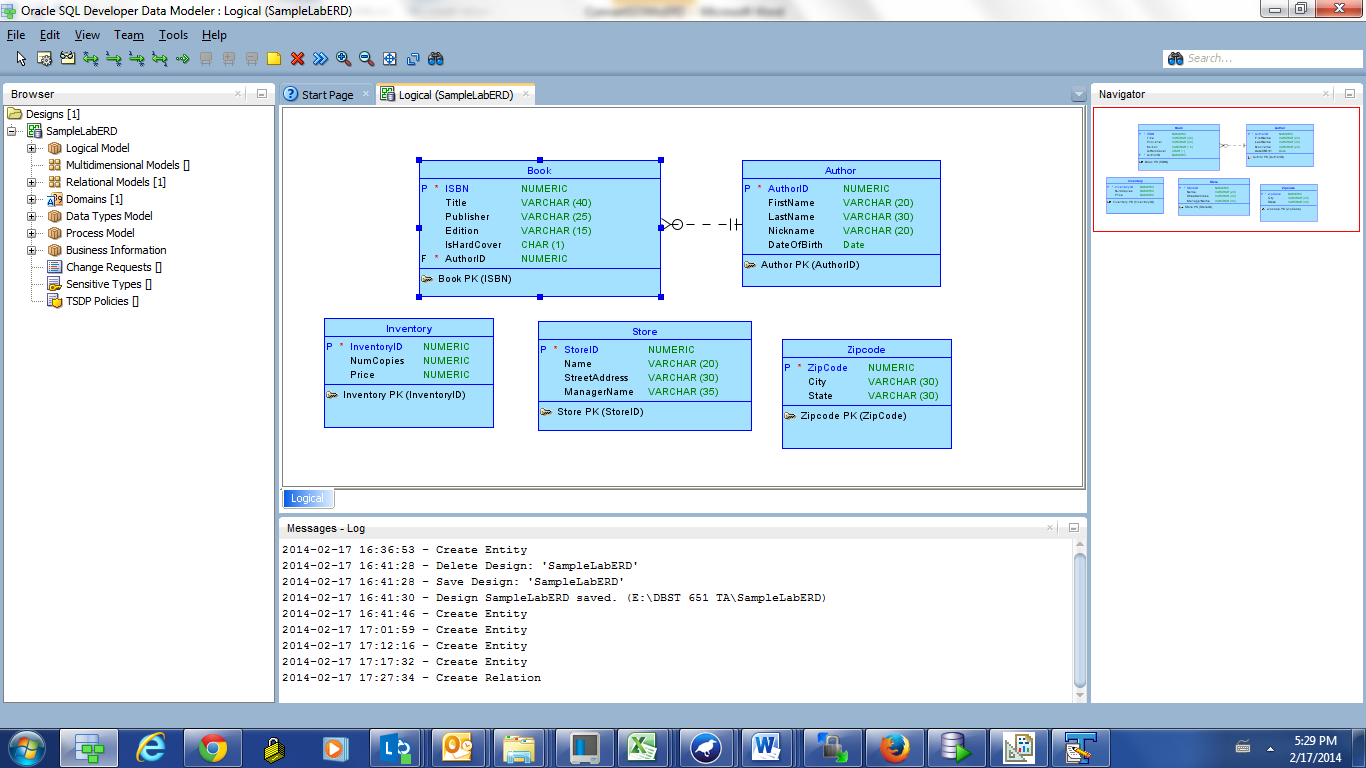
Look for words ‘zero or more’, ‘must have’, ‘may have’, ‘one or more’, ‘at least one.’

Figure 3 shows the relationship between book and author, where author is at one side, and book is at the many side. AuthorID becomes a foreign key in the book entity. **A foreign key is always added to an entity at the many side of the relationship.**

Each book must have an author. If there is more than one author, the database stores only the first author’s name. Each author could have written zero or more books

The book is optional. The cardinality is many.

An author is mandatory. The cardinality is one.



Foreign key AuthorID

Figure 3: Relationship between book and author

Figure 4 shows the relationship between the store and ziptode entities.

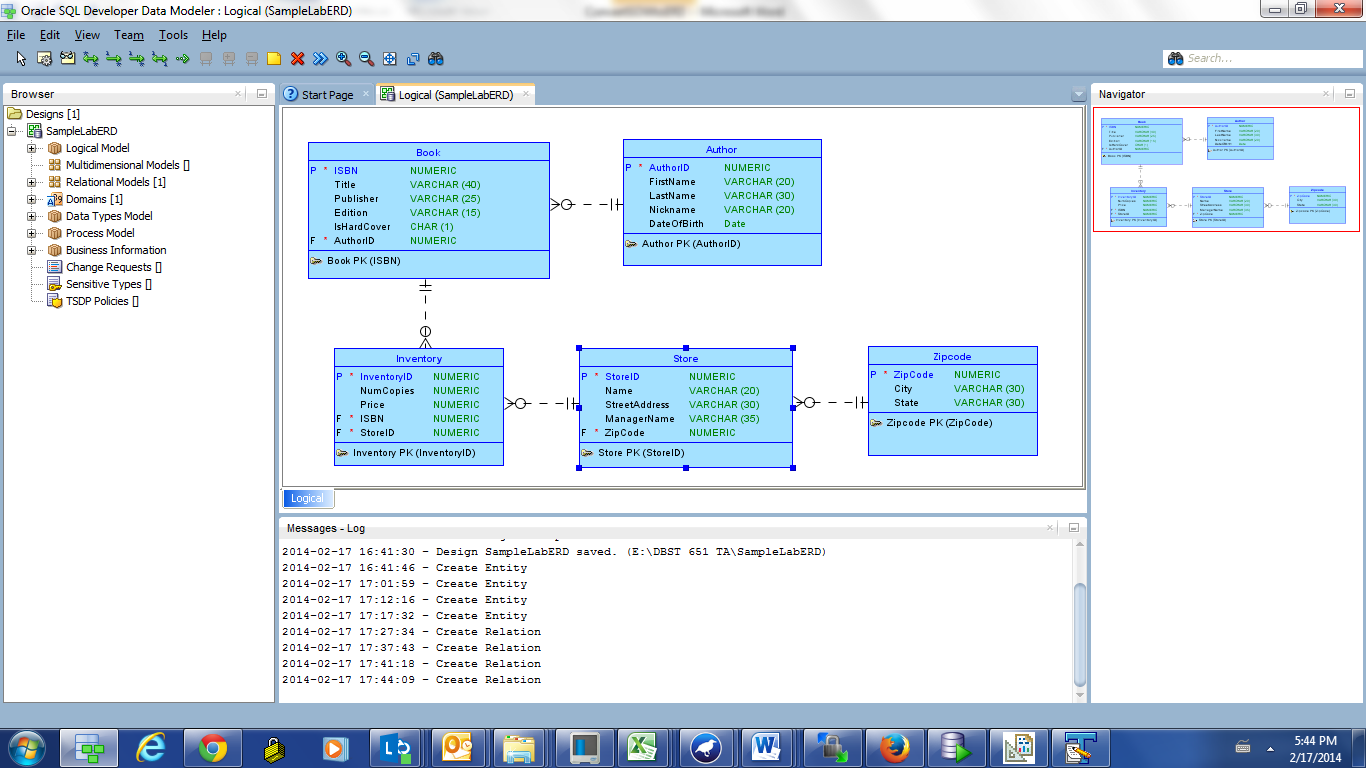
A zipcode location may have zero or more stores, and each store must have a location.

The store is optional

The store cardinality is many

The zipcode is mandatory

The zipcode cardinality is one

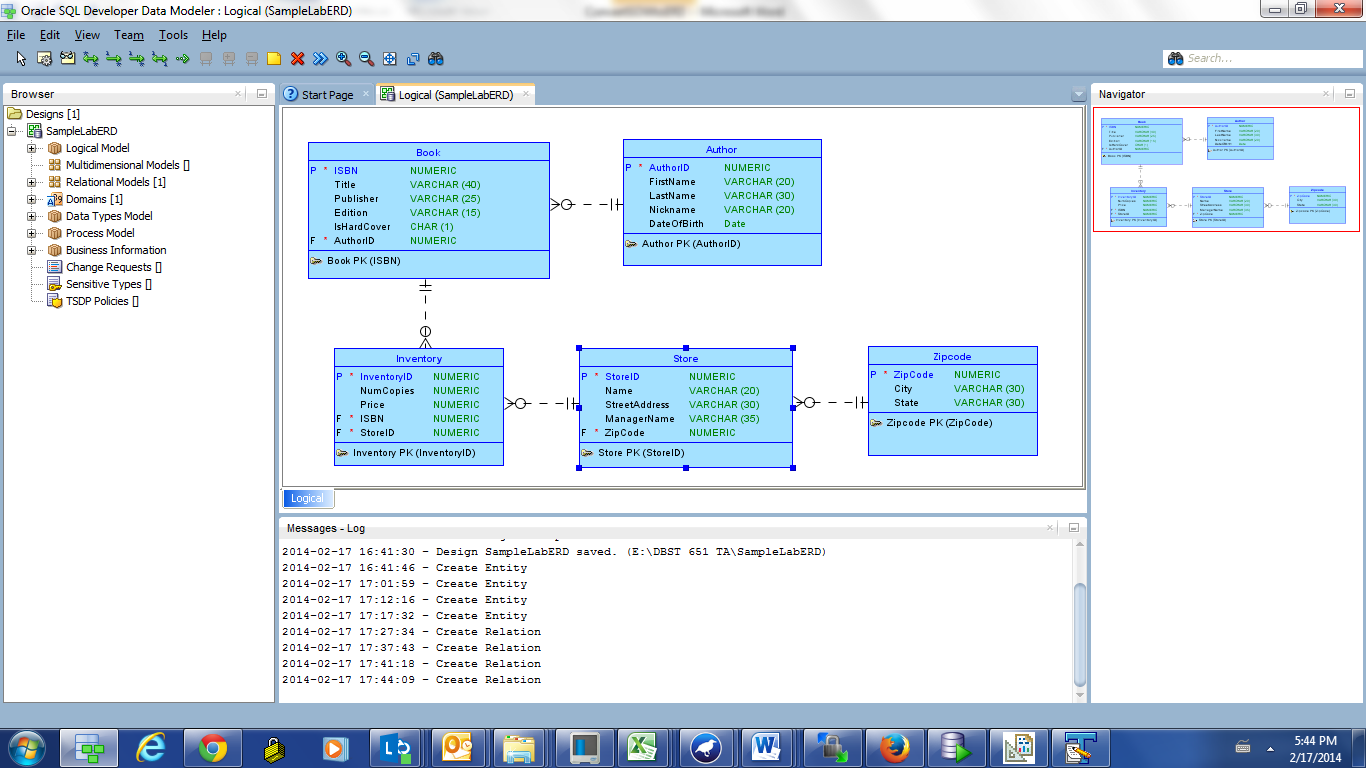


ZipCode foreign key was added the store entity

Figure 4: Relationship between Store and Zipcode

The statement “A book can be available at zero or many stores, and each store could sell zero or more books” indicates that the relationship between store and book is many-to-many. The complete diagram on Figure 5 shows that to resolve the many-to-many relationship, we add

1. an associative entity inventory
2. one-to-many relationship between inventory and store
3. one-to-many relationship between book and inventory



c

a

b

ISBN foreign key was added

StoreID foreign key was added

Figure 5: The diagram with all entities and all relationships