(c)Normalization

ITEM: **BCNF**; Every other attributes dependent on the single primary key “barcode”.

PUBLISHER: **BCNF**; This table only has one attribute and it’s the primary key.

STORE: **BCNF**; Every other attributes dependent on the single primary key “store\_ID”.

BOOK: **BCNF**; “book\_category” is dependent on superkey.

SOFTWARE: **BCNF**; Every other attributes dependent on the single primary key “barcode”.

JOURNAL: **BCNF**; Every other attributes dependent on the single primary key “barcode”.

CD/DVD: **BCNF**; Every other attributes dependent on the single primary key “barcode”.

AUTHOR/ARTIST: **BCNF**; Every other attributes dependent on the single primary key “a\_ID”.

TRANSACTION: **BCNF**; Every other attributes dependent on the single primary key “transaction\_ID”.

STORES: **BCNF**; {store\_ID, barcode} determine the number of stores and sold.

WRITTEN \_BY: **BCNF**;

CONTAINED\_IN: **BCNF**;

COMPOSED\_BY: **BCNF**;

CUSTOMER: **BCNF**; Every other attributes dependent on the single primary key “customer\_ID”.

(d) Indexes

1. Use clustering for ITEM, and make the item-name sorted for easier searching.

2. Use Tree-based for Inventory in table Stores, easy to check some range for inventory.

3. Use Hash-based index for Time in table Transaction, easy to check the number of transactions made in one specific day.

4. Use clustering for last\_name in table Author/Artist, and for publisher\_name in table Publisher, and for issue in table Journal, and for a\_ID in table Composed\_by, and for last\_name in table Customer.

5. Use Hash-based index for genre in table CD/DVD. It’s easy to check what we have in each genre, and for book\_category in table Book, and for software\_category in table Software.

6. Use Tree-based index for quantity in table Contained\_in. It’s fast to search for some range for quantity of each transaction made.