Running Head: ERD DATA MODEL

Order Transaction System: ERD Data Model

Jered McClure

Walden University

Order Transaction System: ERD Data Model



 The PPI order transaction database is a fairly straight forward schema to think about. Each object connects to the next object in a logical manner, without too many incongruities to muck up the overall design. This results in a model that contains only ten classes: CUSTOMER, PAYMENT, SHIPPING, ORDER, EMPLOYEE, PRODUCT, STOCK, STOCK\_LINK, PROD\_LINK, and CUS\_LINK.

 The CUSTOMER table is required for the tracking of customer information. It provides all data to understand PPI’s customer base, as well as, being abstract enough that further customer details can be added in the future without breaking the structural independence of the ERs. Since the data contained in this table will be heavy, it is kept as far from the ORDER table as possible. In essence, the layers between the order and the customer maintain database efficiencies.

 The PAYMENT and STOCK tables are required for the overall management of business information. While payments and inventory are managed in a separate system, having their data stored in the order transaction database provides instant information for order processing, as well as, historical report management. Furthermore, the order entry system can use this data to ensure accurate and timely shipping, and product management.

 Customer service representatives have their details stored in the EMPLOYEE table for tracking their representation on invoice enquiries. This data is kept fundamentally basic as internal business processes will determine where to manage full employee details. However, having this data in this database means that any customer queries can quickly and easily be addressed by the representative who handled their order in the initial instance.

 ORDER and SHIPPING both maintain overall linkages for customer orders placed in the system. Also, they both preserve historical references to customer orders such that customer service representatives are able to view past customer orders. Since the data in both of these tables is likely to grow as PPI gain more customers, the number of columns is kept relatively small.

 Finally, CUS\_LINK, STOCK\_LINK, and PROD\_LINK ensure there are no many-to-many relations between the CUSTOMER and ORDER table, the ORDER and PRODUCT table, as well as, the STOCK and PRODUCT table. While the links can be made manually through a query language, by having these two tables present within the schema, systems and analysts can produce queries without bringing the entire database to a screeching halt. Additionally, since the CUSTOMER and STOCK tables are likely to be data heavy, having a layer of abstraction improves the overall efficiency of system operations.

 One thing that could be done differently is to separate out the customer’s address details into its own table. This would decrease the overall amount of data managed in the CUSTOMER table. However, since the address is only gathered upon order shipment or through business information queries, splitting the data out seems overkill and adds very little performance improvement. That being said, if this is required in the future, the data could be split out without affecting more than one other table in the schema, SHIPPING.

Relationships and Why

|  |  |  |  |
| --- | --- | --- | --- |
| **Tables** | **Relationship** | **Modality** | **Why** |
| CUSTOMER to SHIPPING | 1 to Many | Required | Each shipping entry is related to only a single customer, but each customer can have multiple shipments. |
| CUSTOMER to CUS\_LINK | 1 to Many | Optional | Customers and Orders is a many-to-many relation, as such, CUS\_LINK breaks this up so that there is only ever a 1 to many relation. |
| ORDER to CUS\_LINK | 1 to Many |
| CUSTOMER TO PAYMENT | 1 to Many | Required | Each customer can have many payment options, but each payment option can have only one customer. |
| PAYMENT TO ORDER | 1 to Many | Required | Each payment can have many orders but each order can have only one payment. |
| ORDER to EMPLOYEE | Many to 1 | Required | Each order can have one employee but each employee can have many orders. |
| SHIPPING to ORDER | 1 to 1 | Required | Each order and each shipment can only have one reference to the other. |
| ORDER to PROD\_LINK | 1 to Many | Optional | Each order and each product is a many-to-many relation, as such PROD\_LINK breaks this up so that there is only ever a 1 to many relation. |
| PRODUCT to PROD\_LINK | 1 to Many |
| PRODUCT to STOCK\_LINK | 1 to Many | Optional | A many-to-many relation exists between products and their stock. Therefore, STOCK\_LINK ensures there is only ever a 1 to many relation in the database. |
| STOCK to STOCK\_LINK | 1 to Many |

Reference

Coronel, C., Morris, S., & Rob, P. (2012). *Database Systems: Design, Implementation, and Management* (10th ed.). Boston, MA, USA: Cengage Learning. Retrieved 2012