IEEE Data Engineering Bulletin, March 2000, Vol. 23, No.1, pp. 23-28.

Database Design for Real-World E-Commerce Systems[1]

Il-Yeol Song
College of Information Science and Technology
Drexel University
Philadelphia, PA 19104
song@drexel.edu

Kyu-Young Whang
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology (KAIST) and
Advanced Information Technology Research Center (AITrc)
Taejeon, Korea
kywhang@mozart.kaist.ac.kr

Abstract

This paper discusses the structure and components of databases for real-world e-commerce systems. We first present an integrated 8-process value chain needed by the e-commerce system and its associated data in each stage of the value chain. We then discuss logical components of a typical e-commerce database system. Finally, we illustrate a detailed design of an e-commerce transaction processing system and comment on a few design considerations specific to e-commerce database systems, such as the primary key, foreign key, outer join, use of weak entity, and schema partition. Understanding the structure of e-commerce database systems will help database designers effectively develop and maintain e-commerce systems.

1. Introduction

In this paper, we present the structure and components of databases for real-world e-commerce systems. In general, an e-commerce system is built by following one of two approaches. The first approach is the customization approach using a suite of tools such as IBM's WebSphere Commerce Suite [Shur99]. For example, the Commerce Suite provides tools for creating the infrastructure of a virtual shopping mall, including catalog templates, registration, shopping cart, order and payment processing, and a generalized database. The second approach is the bottom-up development of a system in-house by experts of an individual company. In this case, the developer is manually building a virtual shopping mall with mix-and-match tools. In addition, a database supporting the business model of the e-commerce system must be manually developed.

Whether a developer is using the customization or the bottom-up approach, understanding the structure of e-commerce database systems will help the database designers effectively develop and maintain the system. Our paper is based on our experience of building real-world e-commerce database systems in several different domains, such as an online shopping mall, an online service delivery, and an engineering application.

[1] *Copyright 2000 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.

The major issues of designing a database for e-commerce environments are [BD98, CFP99, KM00, LS98, SL99]:

- Handling of multimedia and semi-structured data;
- Translation of paper catalog into a standard unified format and cleansing the data;
- Supporting user interface at the database level (e.g., navigation, store layout, hyperlinks);
- Schema evolution (e.g., merging two catalogs, category of products, sold-out products, new products);
- Data evolution (e.g., changes in specification and description, naming, prices);
- Handling meta data;
- Capturing data for customization and personalization such as navigation data within the context.

In Section 2, we present our view of a value chain needed by the e-commerce system and its associated data in each stage of the value chain. In Section 3, we first discuss logical components of e-commerce database systems. We then present the detailed database schema of an e-commerce transaction processing (ECTP) system and discuss a few database design considerations specific to e-commerce systems. Section 4 concludes our paper with comments on the roles and future developments of e-commerce database systems.

2. An E-commerce Value Chain and Data Requirements

An e-commerce value chain represents a set of sequenced business processes that show interactions between online shoppers and e-commerce systems. A value chain helps us understand the business processes of e-commerce systems and helps identify data requirements for building operational database systems. Treese and Stewart [TS99] show a four-step value chain that consists of Attract, Interact, Act, and React. *Attract* gets and keeps customer interest. *Interact* turns interest into orders. *Act* manages orders; *React* services customers. The four-step chain could be considered as a minimal model for a working e-commerce system.

In this paper, we present a more detailed value chain that consists of eight business processes. The new value chain integrates steps such as personalization, which is usually performed by a separate add-on product. Figure 1 shows the integrated e-commerce value chain with the 8 business processes,

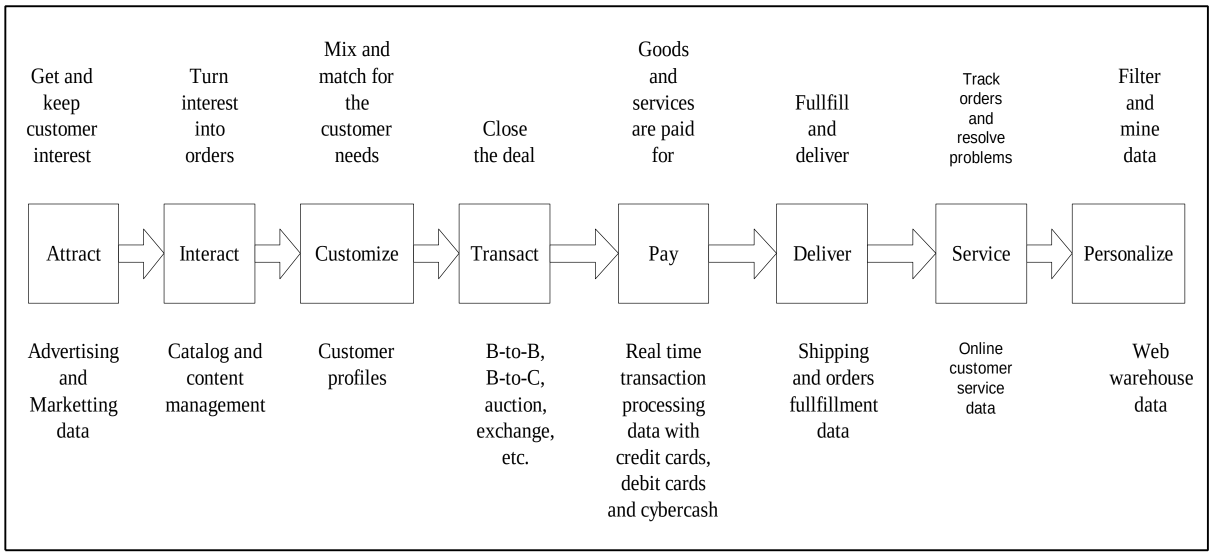
Get and keep customer interest
Turn interest into orders
Mix and match for the customer needs
Close the deal
Goods and services are paid for
Fullfill and deliver
Track orders and resolve problems
Filter and mine data
Attract
Interact
Customize
Transact
Pay
Deliver
Service
Personalize
Advertising and Marketting data
Catalog and content management
Customer profiles
B-to-B, B-to-C, auction, exchange, etc.
Real time transaction processing data with credit cards, debit cards and cybercash
Shipping and orders fullfillment data
Online customer service data
Web warehouse data

Figure 1. An e-commerce value chain with 8 business processes.

their goals and data requirements. We call each phase of the value chain a business process in that it is important in its own right and involves significant complexity. Each business process involves a set of interactions between online shoppers and e-commerce systems for achieving particular objectives. Each business process will have different data requirements based on underlying business models supported by an e-commerce system. For example, products, services and users of e-commerce systems would be different whether the system supports B-to-B, B-to-C, auction, or exchanges. A database designer must fully understand each business process and identify the data requirements needed to support each business process.

3. Database Schema for E-Commerce Systems

3.1 High-Level Logical Components

A database schema for a real-world e-commerce system is significantly complicated. Figure 2 shows a package diagram that shows logical components of a typical e-commerce database. The diagram uses the notation of Package used in UML [BRJ99]. A package in UML is a construct that groups inter-related modeling elements. In Figure 2, each package contains one or more related tables.

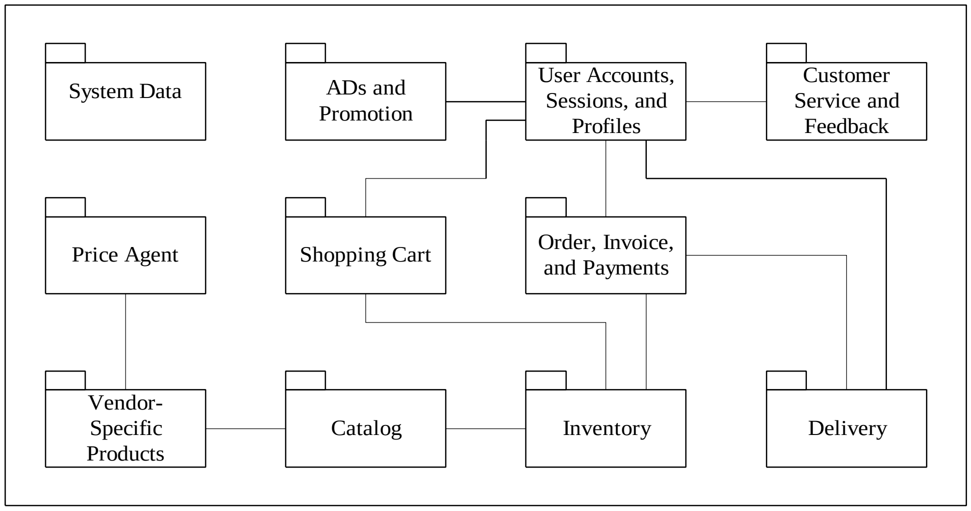

Figure 2. A Package diagram with logical components of e-commerce systems.

Package *User Accounts, Sessions, and Profiles (UASP)* records login ID, password, demographic data, credit card data, customer profiles, and usage history such as the total number of purchases, the total amount of payments, and the total number of returns. The package holds customization data, such as particular items to display, the amount of data to display, and the order of data presentation, and personalization data, such as user purchasing behavior, statistical data, and reporting data. The package could also be extended to include modules for capturing data for clickstream analysis [KM00]. The UASP package also keeps various user types such as individual customers, retail customers, user subgroups, buyers, sellers, and various affiliates, depending on the business model supported by the system.

Package *ADs and Promotion* involves tables that are related to advertising, promotion, and coupons. The package tracks which promotions are associated with which sessions, and which ADs are displayed in which sessions.

Package *Customer Service and Feedback* keeps tracks of customer feedback data for each user and order such as type, nature, status, and responses.

Package *Price Agent* contains data about competitions for each product type. The package allows the system to search other online websites that sell the same products and ranks the site names in the order of prices. Thus, the package may not exist if the site does not have any competitive sites in the same domain.

Package *Shopping Cart* models a shopping cart and inventory items in the shopping cart. Note that items in the shopping cart may or may not be actually ordered. Thus, the various states of the items must be kept track of.

Package *Order, Invoice, and Payment (OIP)* keeps track of actual orders that are transferred from a shopping cart. The package keeps track of individual order items, discounts, sale prices, commissions, taxes, cancels, and any changes in order states. The package also includes invoice history, payment history, and credit card processing modules.

Package *Delivery* includes shipping address, shipping methods and charges.

Packages *Inventory*, *Catalog,* and *Vendor-Specific Products* are closely related. *Vendor-Specific-Products* hold the vendor data and products. *Inventory* lists all the items that are available for sale and their associated tax data. *Catalog* is a standardized model that integrates various vendor-specific products in a domain. Since one vendor could use different terminology and format for the same product, *Catalog* protects the actual business model from the vendor-specific variations. *Catalog* plays a buffer between *Vendor-Specific Products* and *Inventory*. *Catalog* also needs a dynamic catalog management module to accommodate new vendors and new products that were not considered in advance. Thus, actual implementation of the program is done against *Catalog* and *Inventory*. Our experience shows that developing a proper schema for *Catalog* for a domain is most time-consuming and complex.

Package *System Data* holds various system parameters such as server configuration and access control parameters.

3.2 Schema for E-commerce Transaction Processing Systems (ECTP)

In this section, we present a database schema for an e-commerce transaction processing (ECTP) system that sells inventory items. Figure 3 shows the ECTP schema and includes some portions from UASP, Order-Invoice-Payment, Shopping Cart, Inventory, and Delivery packages. Even though a real-world schema would be much more complicated than the one shown in Figure 3, the schema illustrates a few interesting design considerations in e-commerce environments. The detailed schema will be different depending on the business rules and models supported by the site. Note that Figure 3 uses the UML notation. In order to create a relational schema from Figure 3, we need to add the primary key of the one-side to the many-side as a foreign key.

The proper selection of the primary key is very important. For most tables, we use a system-generated primary (surrogate) key to avoid dependence on data changes. Smart keys, which have embedded meanings, create dependency on those data. Thus, changes in those data cause changes in primary keys. All tables in Figure 3 uses surrogate keys.

Note that in order not to lose any data for the personalization procedure and report generation, there are cases that we do not declare an attribute, which is the primary key of another table, as a foreign key. For example, table ORDER_ITEM will capture attribute INV_ITEM# from INVENTORY_ITEM table in the relational schema. But we do not declare the attribute as a foreign key. Doing so will cause the referential integrity to be violated when the INV_ITEM# in the INVENTORY_ITEM table is deleted. When generating reports in this case, a join between ORDER_ITEM and INVENTORY_ITEM must use an outer join, rather than a natural join, so that the report include the order item with the deleted product.

In Figure 3, the relationship between *Order* and *OrderItem* is one-to-many. In typical data modeling situations, *OrderItem* is modeled as a weak entity of *Order*. Thus, the primary key of

OrderItem will be a combination of Order# and a sequence number of items within the order number. However, in this diagram, we created Order_Item# as the primary key of *OrderItem*. Order# in *OrderItem* will simply be a foreign key, without being a part of the primary key of *OrderItem*. This allows the e-commerce system to easily and flexibly handle individual order items independently of *Order*. If you want to count the number of times an item was placed in order, even if the order itself was cancelled, this technique becomes useful.

On the other hand, *OrderItem History* was modeled as a weak entity of *OrderItem*. In *OrderItem History*, the history of each *OrderItem* can be recorded separately. For every traceable

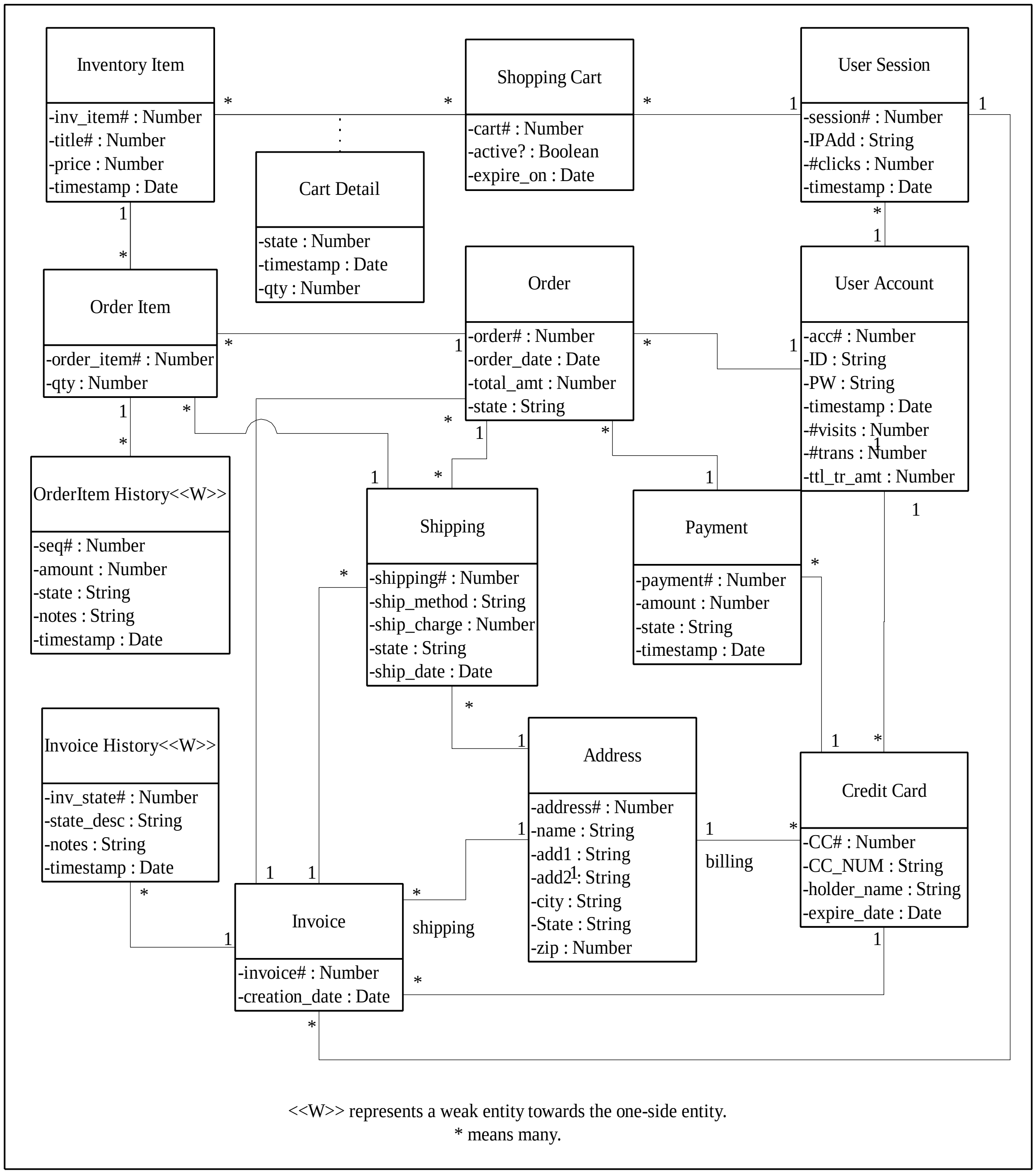

Figure 3. A simplified database schema for e-commerce transaction processing.

entities that could have changes in states, there must be a weak entity that keeps track of the history of the state changes. This facilitates the handling of various customer supports at the lowest level of grain. These traceable entities in Figure 3 include *User Account*, *Order*, *OrderItem*, *Invoice*, *Shipping*, and *Payment*. In Figure 3, we showed only *OrderItem History* and *Invoice History* as weak entities. *Invoice History* is a weak entity of *Invoice* and keeps track of all the changes in states of invoices.

The relationship between *Invoice* and *User Session* allows the system to compute the purchasing rate, which is the number of sessions that result in an invoice, and eventually an actual order. The relationship between *OrderItem* and *Shipping* allows the system to send individual order item to different shipping address. *User Account* has a few statistical attributes. Usually they are placed in a separate table. In Figure, we showed them in *User Account* for the lack of space.

A typical real-world database schema for e-commerce systems are divided into multiple databases run by different database instances for several reasons. For example, a static schema such as catalog could be separated from a more volatile schema such as order processing. This multiple schema approach requires connecting those separated schemas when processing queries that need to concurrently access those schemas. Thus, this approach somewhat slows down the performance. However, there are many advantages such as better stability, maintainability, load balancing, and security.

4. Conclusion

In this paper, we presented an e-commerce value chain with eight business processes, logical components of e-commerce databases, and the schema for an e-commerce transaction processing system. Most real-world e-commerce database schema will have a similar framework as we presented in this paper. Our experience shows that e-commerce tools can speed up the development, but still lack certain functionality such as partial orders, back orders, returns, and email notification to users. Therefore, understanding the structure of e-commerce database systems will help the database designers effectively develop and maintain the system, regardless of the approach taken.

From the database design point of view, an interesting research issue is what database structures are needed to support customization and personalization most effectively. For example, how and what data do we need to capture to build a web warehouse [KM00] for personalization, and then, how do we communicate with users of systems? The personalization process requires significant resources to capture clickstream data and user behavior patterns. Readers are referred to the reference [SL99] for a list of typical OLAP queries and data warehouse design in e-commerce environments.

References

[BRJ99] Booch, G, Rumbaugh, J., and Jacobson, I, *UML User's Guide*, Addison Wesley, 1999.

[BD98] Buchner,A. and Mulvenna, M, "Discovering Internet Market Intelligence through Online Analytical Web Usage Mining," *SIGMOD Record*, Vol. 27, No.4, December 1998, pp. 54-61.

[CFP99] Ceri, S., Fraternalim P., and Paraboschi, S, "Design Principles for Data-Intensive Web Sites," *SIGMOD Record*, Vol. 28, No.1, March 1999, pp. 84-89.

[KM00] Kimball, R. and Merz, R., *The Data Webhouse Toolkit*, Wiley, 2000.

[LS98] Lohse, G.L. and Spiller, P, "Electronic Shopping," *Communications of the ACM*, Vol.41, No.7, 1998, pp. 81-88.

[Shur99] Shurety, S, *E-Business with Net.Commerce*, Prentice Hall, 1999.

[SL99] Song, I-Y. And LeVan-Schultz, K., "Data Warehouse Design for E-Commerce Environments," *Lecture Notes in Computer Science*, Vol. 1727, Springer, pp. 374-388.

[TS99] Treese, G.W. and Stewart, L.C, *Designing Systems for Internet Commerce*, Addison Wesley, 1998.