2012

Music Store

Phase I

This report contains an overview of database design, entity relationship diagrams, functional dependencies, normal forms, design justifications, assumptions, and other design related information relating to the database / music store application for the Principles of Database Systems course at Metropolitan State University of Denver.

Devin Brown & Brad Gill
Principles of Database Systems
10/23/2012

1. Project Summary

The Oracle music store serves a wide variety of musical interests. As a small store, it must maintain close control of its sale and inventory. The store keeps an inventory of music in stock. The store also keeps track of sales and customer special orders for items out of stock, which are placed through an employee of the store.

The primary goal when designing this database was to allow for ease of data accessibility, eliminate dependencies, minimize redundancies, while structuring the data in a form that was easily understood and still remained functionally orthogonal.

The overall goal of the Music Store Database / Application Interface is to enable maximum end-user, usability and flexibility, while allowing for queries and other DBA tasks to be assigned with relative ease. Another important requirement for this database was setting an infrastructure that allows for easy administration and user-access based restriction.

As is shown in the UML diagram, join tables are often the solution for allowing flexibility between option tables, which also contain the "one -to-many" type relationship. These join tables, for example, allow for **Customers** to have multiple **Transactions** of which contain multiple **Music**(s).

2. Assumptions

- Assume that the database is implemented for transactions on or after Jan. 1, 2000
- **Music**: Music titles can come in 3 different forms: cassette tape, compact disk, or sheet music. At any one time the store may have all three forms in stock for a particular music title. Therefore, the database must keep track of which forms of music are in stock for each music title
- **Vendors**: A vendor may publish several **music** titles; however, a particular music title can be published by one and only one vendor.
- Special **Orders** to **Vendors**: When placing a special order, all music by the same vendor may exist on the same order. Music from different vendors must be placed on different orders: one vendor per order. You can create a unique order ID by using the date as part of the order number. Date should be in the format YYMMDD. **Orders** are placed by a store employee.
- For special orders, recording **Customer** information is mandatory
- For in-stock purchases, **Customer** information is optional.
- **Employees** of the store will be the users of this database system, not the customers themselves. Information on every employee is in the database for management purposes.
- Each **Sale** is made by one **Employee** for one **Customer**. The sale can include multiple music titles, and should also include quantity & price of each item and a grand total. Upon making the sale, the system should *decrement* the music in stock appropriately.
- **Employee** and **Vendor** have required **Contact**(s), **Customer** is optional
- Each **Employee** and **Customer** may have multiple **Transactions**
- Each **Transaction** may have multiple **Music**(s)
- Each **Music** has one or more **Artists**
- Each **Order** may contain multiple **Music**(s)

- **Customers** may be linked to an **Order** (special order)
- An **Employee** may only have ONE **Job_Title** (wage, wage_type)
- Each **Music** may have zero or more **Promotions**
- **Vendors** have a **company_contact** AND **representative_contact**, for shipping and general information, respectively
- **Customers** may optionally subscribe for **Promotions** notifications
- **Employees** may have multiple **Order**(s) linked to **Vendors**
- Each **Vendor** may fulfill multiple **Order**(s)

3. Sample Statistics

Sample statistics relevant to the applications include the following. They would impact the real operational environment of the database system. Here, it is offered as general information only.

a. **CUSTOMERS:** 3,000 customers in the database
b. **QUERIES**: sales reports are done once weekly by music title
c. **MUSIC:** 5,000 different music titles, between 1 and 20 copies of any music title is in stock at any given time
d. **VENDORS:** 100 different vendors supply music
e. **CUSTOMER ORDERS/SALES:**
    1. 50 customer special orders per day are submitted (each order has on average 2 titles)
    2. 30 orders are queried per day to check on the status of the order
    3. 50 sale transactions per day in the store with an average of 3 titles per sale

4. Data Requirements

a. The Oracle music store sells music in 3 different forms: cassette, compact disk, and sheet music. A given music title has several attributes associated with it: examples are title, artist(s), producer, musical subject, physical type of music (cassette, compact disc, or sheet music), year of recording, etc. An employee can query the music in stock by searching on any one or more attributes of the music.
b. Music vendors can distribute several music titles; however, each music title can only be distributed by one vendor. Vendor information (name, address, phone, contact person, etc.) is also maintained in the database.
c. **Employees** are in charge of two tasks:
    i. selling music to customers
    ii. placing special order from vendors for customers.
d. **Employee** information maintained: ID, name, address, pay level, job title, etc.
e. The store can place **Orders** to **Vendors** when stock runs low or when a customer requests a special order.

f. A particular **Order** can only go to one vendor: this means that all items being ordered from a particular vendor can exist on the same order.
g. If items are distributed by differing **Vendors**, an order form for each vendor must be filled out. Example attributes of orders are the item(s) to be ordered, date of order, vendor to whom the order is going, and customer information if applicable.
h. **Customer** information is maintained when special orders are placed.
i. **Customer** information can also be maintained optionally when selling music: for example, customers can fill out a card to be put onto a mailing list.
j. Sales information must be maintained whenever music items are sold: amount of purchase, each item purchased, customer information (if applicable), the employee making the sale, etc.
k. **Music** may have one or more **Artists**
l. **Music** may have one or more **Promotions**

5. Functional Requirements

There are four types of processes that are relevant:

[query] This process allows store employees to query the database with regard to music in stock and music on order.

[order] This process generates special orders by customers for music. A store employee inputs orders.

[sell] This process modifies the database appropriately, regarding the item(s) being sold and the employee making the sale. It is typically the operation done at the cash register. An invoice is generated for every order placed by a customer. A receipt is printed for every in-store sale transaction.

[admin] This process modifies the database information about employees, customers, vendors, etc. It may have other management report features which are left out for this project.

6. Entity Relational Diagrams

(attached, last two pages)

7. Dependencies, Multi-Valued Dependencies, and Normal Forms

Employees dependency:

Key: active , ID

ID - > Employee_contact, Job_title

Job_title -> wage_type, wage

Nothing determines active.

BCNF:

{Job_title,wage_type, wage}

{ ID, Active, Employee_contact, Job_title}

Contact dependency:

Key: ID

ID -> Name, Address, City, State

Address, City, State -> zip

Name, Address -> email, phone

BCNF:

{Address, City, State, zip}

{name,Address,email,phone}

{ID,Name,city,State}

This would be worse than keeping them all together since we would have to update multiple tables when adding a contact.

Vendors Dependency:

Key: ID

ID -> Represntative_contact

Representative_contact -> company_contact

Normalized

Customer dependency:

Key: Customer_contact, promotions, ID

Normalized

Employee_transactions dependency:

Key: Transaction_ID

Transaction_ID -> employee_ID, Customer_ID

Normalized

Transaction dependency:

Key: ID

ID -> sale_amount, date_sold, employee_ID, music_ISBN

Normalized

Music Transactions dependency:

Key: Transaction_ID,Quantity

Transaction_ID -> Music_ISBN, Quantity

Normalized

Music dependency:

Key: ISBN, Type

ISBN -> producer, genre, year, vendor_ID

Type -> price, quantity

Normalized

Artists dependency:

Key: Music_ISBN

Music_ISBN -> artist name

Normalized

Order dependency:

Key: ID,Status

ID -> Music_ISBN, Date_Ordered

Music_ISBN -> Vendor_ID

Normalized

Music_Orders:

Key: Order_ ID, quantity

Order_ID -> Music_ISBN

Normalized

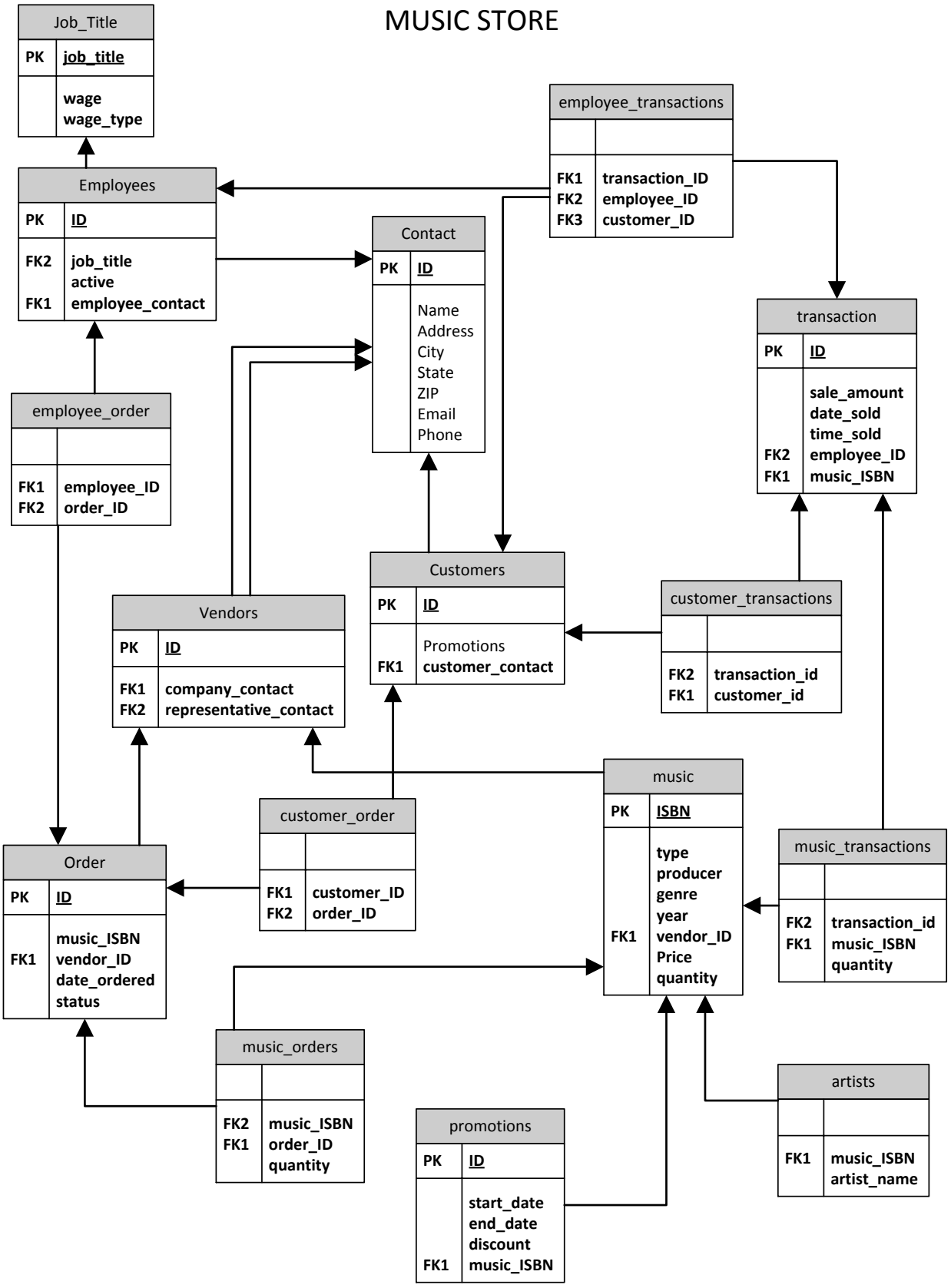
Promotions dependency:

Key: ID

ID -> start_date, End_date, Music_ISBN, Discount

Normalized

MUSIC STORE


Job_Title
PK job_title
wage
wage_type
employee_transactions
FK1 transaction_ID
FK2 employee_ID
FK3 customer_ID
Employees
PK ID
FK2 job_title
active
FK1 employee_contact
Contact
PK ID
Name
Address
City
State
ZIP
Email
Phone
transaction
PK ID
sale_amount
date_sold
time_sold
FK2 employee_ID
FK1 music_ISBN
employee_order
FK1 employee_ID
FK2 order_ID
Customers
PK ID
Promotions
FK1 customer_contact
customer_transactions
FK2 transaction_id
FK1 customer_id
Vendors
PK ID
FK1 company_contact
FK2 representative_contact
music
PK ISBN
type
producer
genre
year
FK1 vendor_ID
Price
quantity
customer_order
FK1 customer_ID
FK2 order_ID
music_transactions
FK2 transaction_id
FK1 music_ISBN
quantity
Order
PK ID
FK1 music_ISBN
vendor_ID
date_ordered
status
music_orders
FK2 music_ISBN
FK1 order_ID
quantity
promotions
PK ID
start_date
end_date
discount
FK1 music_ISBN
artists
FK1 music_ISBN
artist_name