CS352: Advanced Database Systems

Final Project

Trina VanderLouw

Professor Carol Segura

Colorado Technical University Online

March 25, 2013

Highlights: Normalization, Database Design, Database Administration, SQL

Programs Used: MySQL Workbench, SQL Server, Microsoft Access

Instructor Feedback: Hi Trina, wonderful, clean, organized code all annotated correctly.

Excellent job here, nicely organized and good depth throughout.

Final Grade: A
Table of Contents

The Database Models, Languages, and Architecture.......................................................... 3

ANSI Architecture Model........................................................................................................ 3

Data Independence.................................................................................................................. 3

Data Administrator.................................................................................................................. 4

Database Administrator ......................................................................................................... 4

Pros and Cons of DAs and DBAs .......................................................................................... 5

Database System Development Life Cycle.............................................................................. 6

Database Management Systems............................................................................................. 8

Advanced SQL........................................................................................................................ 12

Web and Data Warehousing and Mining in the Business World ........................................... 16

Extraction Transformation Loading (ETL)............................................................................ 18

References.............................................................................................................................. 20
The Database Models, Languages, and Architecture

ANSI Architecture Model

The 3 level ANSI SPARC architecture model stands for the American National Standard Institute Standard Planning and Requirements Committee (Jackson, 2013). This architecture model consists of an external layer, conceptual layer, and internal layer. Each layer represents a different relationship to the data in the database. The external layer contains the schemas that show the use of the data. This is where the users can view the data that is relevant to them. The conceptual layer is the community view of the data and represents the meaning of data. This layer shows what data is stored and what its relationship is to the database and other data. The internal layer represents the storage of data. This is literally the physical representation of the location of the database on the computer. This layer also describes how the data is stored in the database. (Segura, 2013).

The ANSI-SPARC architecture is important because it allows the users to access data that is relevant to them. They do not have to understand how the data is stored physically because they are working at the external level and not the internal level. This type of architecture is also useful to the DBAs who may need to change the structure of the database storage because the changes will not affect the users and their views. (Jackson, 2013). It is vital that business owners and managers recognize the importance of careful planning from the beginning. The planning process will save time in the long run and will allow them to have the best possible system for their needs (Segura, 2013).

Data Independence

Data independence is promoted by the ANSI model and has two types of independence. They are the physical and logical data types. The logical is used to define the immunity which
exists between the external schemas and the conceptual schemas. If a change is made to the conceptual schema it will not have an impact on the external. The physical independence is the ability for the conceptual schema to be immune from changes in the internal schemas. The physical refers to changes made in the internal schema. (Segura, 2013).

**Data Administrator**

A data administrator (DA) is hired by a company to work on large-scale developments that are strategic and managerial (Segura, 2013). They will often work on moving subsystems between platforms and will work on developing and implementing policies and procedures. The DAs are responsible for naming conventions used within the database and how documentation will be kept and used. They will plan out the usage systems and train others how to effectively use the resources. The DA will also often work closely with the DBA to select the data management and development tools. They will work together on any issues that arise and model the flow of information for the organization. (Data Administrator, 2002).

**Database Administrator**

A database administrator’s (DBA) job is to ensure that data is available to all users and that it is secured against unauthorized use. They use the software to store and organize data and their main role is technical in nature. They are tasked with identifying the needs of the users and they create and maintain databases to meet those needs. DBAs are also responsible for any errors that occur and the resolution of those errors. They create and test the modifications that are necessary to maintain the database security. Updating permissions and backing up the data are also part of the DBAs role. They will work with management, and sometimes a DA, to understand the needs of the company and to plan the goals of the database to meet those needs. (What Database Administrators Do, 2013).
Pros and Cons of DAs and DBAs

There are many organizations that do not have both a DA and a DBA. The ideal situation would be to have two people, one DA and one DBA, so that they can work together on both the strategic and technical levels. Having two people gives the company a better foundation for their data management because both of them are trained to understand different points of view. The DA might have ideas and suggestions that relate to the overall strategy of the company but may not know how to implement those. Working with a DBA they can come up with the best possible solution to meet the needs of the company on a technical and strategic level. Likewise, if there is no DA the DBA is lacking a valuable dynamic that sheds light on the strategic business side of the data management. They may be able to effectively maintain the database, but the company would have potential losses in their competitive advantage because of the lack of strategic data capabilities.
This ERD shows the entities, attributes, and relationships for the organization’s database. The enhanced ERD tools are used to show multiplicity and the relationships that exist between entities. In this scenario, an employee can help zero to many customers, but the customer can be helped by only one employee. Both the employee and the customer can have only one address, but one address could belong to many customers or employees. For example, one business office could have 12 customers that all share the same address.

Employees must be either customer facing or internal support, but not both. This relationship is shown as being mandatory, with only one type selected. The employee types are subclasses of the employee superclass.

A customer can buy or sell products and can have many orders. They can have zero to many of each type of order. Each order can be bought or sold by only one customer. Each order contains products and can have one-to-many on each order. A product can be on zero to many orders.
Database Management Systems

Original Form

<table>
<thead>
<tr>
<th>Charity ID</th>
<th>Charity Name</th>
<th>Charity Location</th>
<th>POC ID</th>
<th>Tel Extn.</th>
<th>Customer ID</th>
<th>Customer Name</th>
<th>Date Started</th>
<th>No of Month</th>
<th>Date Placed</th>
<th>Expected Contribution End</th>
</tr>
</thead>
</table>

1<sup>st</sup> NF

Normalization is used to show the minimal number of attributes needed with minimal redundancy (Connolly & Begg, 2010, pg. 366). The goal of first normal form (1<sup>st</sup> NF) is to make sure there is only one value in each cell. It also ensures there are no repeating groups.

CharityDonation (**CharityID**, CharityName, CharityLocation, POCName, POCID, TelExtn, CustomerID, CustomerFirstName, CustomerLastName, StartDate, NumberOfMonths, DatePlaced, ExpectedEndDate)

2<sup>nd</sup> NF

The second form (2<sup>nd</sup> NF) is used to place only the essential fields into the table that are dependent on the primary key and to use additional tables as needed. To get this example into 2<sup>nd</sup> NF tables have been created for the Charity, POC, Customer, and Donations. These tables allow the attributes to be dependent on the primary key.

Charity (**CharityID**, CharityName, CharityLocation)

POC (**POCID**, POCName, TelExtn)

Customer (**CustomerID**, FirstName, LastName)

Donations (**CustomerID**, DatePlaced, StartDate, NumberOfMonths, ExpectedEndDate)
3\textsuperscript{rd} NF

Third form (3\textsuperscript{rd} NF) is complete if there are no transitive dependencies (Connolly & Begg, 2010, pg. 388). In this scenario, the tables are already in 3\textsuperscript{rd} NF because each of the attributes is dependent on the primary key and we would just need to add a linking table. This would look like this:

Contributions (\textit{DonationID, CustomerID, CharityID, POCID})
Charity (\textit{CharityID, CharityName, CharityLocation})
POC (\textit{POCID, POCName, TelExtn})
Customer (\textit{CustomerID, FirstName, LastName})
Donations (\textit{DonationID, DatePlaced, StartDate, NumberOfMonths, ExpectedEndDate})

Boyce-Codd

Boyce-Codd Normal Form (BCNF) is an advanced normalization form that uses additional constraints to ensure that all determinants are candidate keys. The determinants are attributes that other attributes are fully functionally dependent on. It is important to note that sometimes it is better to leave it in 3\textsuperscript{rd} NF to preserve dependencies (Connolly & Begg, 2010, pg. 400). The only part of the 3\textsuperscript{rd} NF that I could find that was not dependent was the DatePlaced in the Donations table. This could be fixed by changing the date to the Contributions table. The Charity table has only an ID, the charity name, and location. Each of the attributes are dependent to the CharityID. The POC has only and ID, name, and telephone. The Customer has only ID, first name, and last name. The Donations now only has a start date, number of months, and expected end date.
Contributions (DatePlaced, DonationID, CustomerID, CharityID, POCID)
Charity (CharityID, CharityName, CharityLocation)
POC (POCID, POCName, TelExtn)
Customer (CustomerID, FirstName, LastName)
Donations (DonationID, StartDate, NumberOfMonths, ExpectedEndDate)

Part 2

The following diagram shows the updated customer/employee ERD. Each of the tables is in 3rd NF. The attributes in the Employee table are all dependent on the Employee ID. The AddressID is a foreign key that links to the Address table. The Address table has an AddressID primary key that is used not only for the Employee table, but also the Customer table. The employee helps the customer and the customer table has a primary key of CustomerID. This is a unique identifier for the customer. The Customer can be a purchaser, supplier, or both. They will be given a PurchaserID and/or SupplierID so that vendors and customers can be identified in separate tables. The Purchaser and Supplier tables will use the CustomerID foreign key to link to the customer’s personal information. The tables will also link to the orders table through the OrderID foreign key. The Orders table has OrderID as a primary key. This identifier will record whether the order was a purchase or a sale, what product was bought or sold, the date it was bought/sold, the total, and if the product is a resale item. The product table uses ProductID as a primary key that links to the orders table. The attributes inside of this table are all used to further identify the product.
Advanced SQL

The following are MySQL statements for creating the tables for the ERD above.

CREATE TABLE Address (  
    AddressID INT NOT NULL,  
    Address1 VARCHAR(45) NOT NULL,  
    Address2 VARCHAR(20) NULL,  
    City VARCHAR(20) NOT NULL,  
    State VARCHAR(2) NOT NULL,  
    Zip INT(5) NOT NULL,  
    PRIMARY KEY (AddressID)) ;

CREATE TABLE Employee (  
    EmployeeID INT NOT NULL,  
    FirstName VARCHAR(20) NOT NULL,  
    LastName VARCHAR(20) NOT NULL,  
    SSN INT NOT NULL,  
    AddressID INT NOT NULL,  
    Phone VARCHAR(12) NOT NULL,  
    Email VARCHAR(45) NOT NULL,  
    StartDate DATE NOT NULL,  
    EndDate DATE NULL,  
    PRIMARY KEY (EmployeeID),  
    FOREIGN KEY (AddressID) REFERENCES Address (AddressID)) ;

CREATE TABLE CustomerFacing (  
    EmployeeID INT NOT NULL,  
    ProductSpecialty VARCHAR(20) NULL,  
    TrainingHours INT NULL,  
    CommissionRate DECIMAL(4,2) NOT NULL,  
    PRIMARY KEY (EmployeeID),  
    FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID)  
    REFERENCES Address (AddressID)) ;

CREATE TABLE InternalSupport (  
    EmployeeID INT NOT NULL,  
    Salary DECIMAL(6,2) NOT NULL,  
    SupportArea VARCHAR(20) NOT NULL,  
    PRIMARY KEY (EmployeeID),  
    FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID)  
    REFERENCES Address (AddressID)) ;

CREATE TABLE Customer (  
    CustomerID INT NOT NULL,  
    CustomerName VARCHAR(45) NOT NULL,  
    CustomerCity VARCHAR(20) NOT NULL,  
    CustomerState VARCHAR(2) NOT NULL,  
    CustomerZip INT(5) NOT NULL,  
    PRIMARY KEY (CustomerID)) ;
CREATE TABLE Customers (  
FirstName VARCHAR(20) NOT NULL,  
LastName VARCHAR(20) NOT NULL,  
Phone VARCHAR(12) NOT NULL,  
Email VARCHAR(45) NOT NULL,  
AddressID INT NOT NULL,  
EmployeeID INT NOT NULL,  
PRIMARY KEY (CustomerID),  
FOREIGN KEY (AddressID)  
    REFERENCES Address (AddressID),  
FOREIGN KEY (EmployeeID)  
    REFERENCES CustomerFacing (EmployeeID)) ;

CREATE TABLE Purchasers (  
PurchaserID INT NOT NULL,  
CustomerID INT NOT NULL,  
PRIMARY KEY (PurchaserID),  
FOREIGN KEY (CustomerID)  
    REFERENCES Customer (CustomerID)) ;

CREATE TABLE Suppliers (  
SupplierID INT NOT NULL,  
CustomerID INT NOT NULL,  
PRIMARY KEY (SupplierID),  
FOREIGN KEY (CustomerID)  
    REFERENCES Customer (CustomerID)) ;

CREATE TABLE Product (  
ProductID INT NOT NULL ,  
Name VARCHAR(20) NOT NULL ,  
Color VARCHAR(10) NOT NULL ,  
Size INT NOT NULL ,  
Price DECIMAL(4,2) NOT NULL ,  
Quantity INT NOT NULL ,  
Description TEXT NOT NULL ,  
PRIMARY KEY (ProductID)) ;

CREATE TABLE Orders (  
OrderID INT NOT NULL,  
PurchaserID INT NULL,  
SupplierID INT NULL,  
ProductID INT NOT NULL,  
Date DATE NOT NULL,  
Total DECIMAL(4,2) NOT NULL,  
Resale BINARY NULL,  
PRIMARY KEY (OrderID),  
FOREIGN KEY (PurchaserID)  
    REFERENCES Purchasers (PurchaserID),  
FOREIGN KEY (SupplierID)  
    REFERENCES Suppliers (SupplierID),  
FOREIGN KEY (ProductID)  
    REFERENCES Product (ProductID)) ;
FOREIGN KEY (PurchaserID)
    REFERENCES Purchasers (PurchaserID),
FOREIGN KEY (SupplierID)
    REFERENCES Suppliers (SupplierID),
FOREIGN KEY (ProductID)
    REFERENCES Product (ProductID))
;
The following MySQL statements are for **adding data** to the tables.

**INSERT INTO Address**
(AddressID, Address1, Address2, City, State, Zip)
VALUES (1,'1300 Sunnyside Drive', NULL, 'Cerritos', 'CA', 90703),
(2,'9795 Riverside Park','Suite 302','Overland Park','KS',66212)
;

**INSERT INTO Employee**
(EmployeeID, FirstName, LastName, SSN, AddressID, Phone, Email, StartDate, EndDate)
VALUES (1, 'Josie', 'Palmer', 607529877, 1, '702-651-8513', 'Josie.Palmer@marketworld.com', '2012-05-01', NULL)
;
**INSERT INTO CustomerFacing**
(EmployeeID, ProductSpecialty, TrainingHours, CommissionRate)
VALUES (1, 'Shoes', 20, 10)
;
**INSERT INTO Customer**
(CustomerID, FirstName, LastName, Phone, Email, AddressID, EmployeeID)
VALUES (1,'Sierra','Smith','952-752-9574','S.Smith205@gmail.com',2,1)
;
**INSERT INTO Purchasers**
(PurchaserID, CustomerID)
VALUES (1,1)
;
The following is an **update** to the commission rate for an employee (25% increase)

**UPDATE CustomerFacing**
SET CommissionRate=CommissionRate*1.25
WHERE EmployeeID=1
;

The following are MySQL statements for **selecting** the customer and employee data.

**SELECT ***
FROM Customer
;
**SELECT ***
FROM Employee
;
**SELECT Employee.FirstName AS EmployeeFirstName, Employee.LastName AS EmployeeLastName,
Customer.FirstName AS CustomerFirstName,
Customer.LastName AS CustomerLastName
FROM Employee
INNER JOIN Customer
ON Employee.EmployeeID=Customer.EmployeeID
;
The following are MySQL statements for **deleting** data
DELETE FROM Purchasers
WHERE CustomerID=1
;
DELETE FROM Customer
WHERE CustomerID=1
;
DELETE FROM customerfacing
WHERE employeeID=1
;
DELETE FROM customer
WHERE employeeID=1
;
DELETE FROM employee
WHERE employeeID=1
;
Web and Data Warehousing and Mining in the Business World

Star Schema

MySQL code for the Dimensions and Fact Table

CREATE TABLE Product (
  ProductID INT NOT NULL,
  ProductName VARCHAR(25) NOT NULL,
  SKU INT(16) NOT NULL,
  UnitPrice DECIMAL(6,2) NOT NULL,
  Size VARCHAR(5) NULL,
  Color VARCHAR(20) NULL,
  Description VARCHAR(45) NULL,
  Department VARCHAR(10) NOT NULL,
  PRIMARY KEY (ProductID))
;

CREATE TABLE Customer (
  CustomerID INT NOT NULL,
  CustomerName VARCHAR(35) NOT NULL,
  CustomerType VARCHAR(10) NOT NULL,
CREATE TABLE Customer (
  CustomerID INT NOT NULL,
  City VARCHAR(20) NOT NULL,
  State VARCHAR(10) NOT NULL,
  Zip INT(5) NOT NULL,
  Region VARCHAR(20) NOT NULL,
  Country VARCHAR(20) NOT NULL,
  PRIMARY KEY (CustomerID)) ;

CREATE TABLE Employee (
  EmployeeID INT NOT NULL,
  EmployeeFirstName VARCHAR(20) NOT NULL,
  EmployeeLastName VARCHAR(25) NOT NULL,
  EmployeeType VARCHAR(10) NOT NULL,
  Address VARCHAR(20) NOT NULL,
  City VARCHAR(20) NOT NULL,
  State VARCHAR(20) NOT NULL,
  Zip INT(5) NOT NULL,
  PRIMARY KEY (EmployeeID)) ;

CREATE TABLE Time (
  TimeID INT NOT NULL,
  Day INT(2) NOT NULL,
  Week INT(2) NOT NULL,
  Month INT(2) NOT NULL,
  Year INT(4) NOT NULL,
  PRIMARY KEY (TimeID)) ;

CREATE TABLE Orders (
  TimeID INT NOT NULL,
  CustomerID INT NOT NULL,
  ProductID INT NOT NULL,
  EmployeeID INT NOT NULL,
  OrderTotal DECIMAL(6,2) NOT NULL,
  NumberOfItems INT NOT NULL,
  FOREIGN KEY (TimeID)
    REFERENCES Time (TimeID),
  FOREIGN KEY (CustomerID)
    REFERENCES Customer (CustomerID),
  FOREIGN KEY (ProductID)
    REFERENCES Product (ProductID),
  FOREIGN KEY (EmployeeID)
    REFERENCES Employee (EmployeeID)) ;
Extraction Transformation Loading (ETL)

ETL is the term used to describe the process of getting data from its original source and into a data warehouse. This process is completed by extracting data from one source and loading it into another. One problem with this process is that the data types from one database, or source, do not match the data types for the other. It could also be a problem that the data to be transferred is not in a database format to begin with, such as text files, spreadsheets, and old legacy systems (Segura, 2013). For example, if a company was using MS Access and wanted to send the information to a MySQL warehouse, they would have to make some changes before loading the data. One example of a conflicting data type is “currency” in Access. There is no such data type in MySQL, so the information would need to be transformed from “currency” into a DECIMAL format (Curtiss, 2007). Once the data is extracted and transformed, it can then be loaded into the new data warehouse.

The data in the data warehouse can be updated as often as needed. Some companies update data once every six months, while others update data every hour. The frequency of the data additions will be dependent on the functional purpose of the data warehouse. As the data is added into the new warehouse or data mart, it is important to note that professionals will need to be involved in the process. As part of this process they should ensure the data transfers are automated, well documented, and easily changeable (Segura, 2013). The documentation will be used to record what changes occurred and which data types were transformed. To complete the operation a separate database should be used as a transitional database, called a staging database. This is done to avoid causing corruption in either the source or the destination database.

To complete the ETL process it is important to have a logical data map that is used prior to the physical data being transformed. This map will provide information on the relationships in
the extreme starting points and the extreme ending points. This can be shown using either a table or a spreadsheet. One good idea is to use layers that show the different sources, operating systems, hardware, and detailed fields and data types (Segura, 2013). Once everything is done, and the data is clean, the last step is to load the data.
References


