Student Name

Student Email

Abstract

This document provides an example outline for assignments or papers for the course CS779. The format of this document is not required for the assignments in CS779 but is here to show an example of formalization and can be used as a template. The real purpose of this document is to show a good example of an organized document.

Assignment Template

MET CS 779 Assignment Template

MET CS 779 Assignment Template

Table of Contents

[1. Introduction 2](#_Toc8753632)

[2. Some Heading 1 2](#_Toc8753633)

[1) Some Sub-heading 1 2](#_Toc8753634)

[2) Some Sub-heading 2 2](#_Toc8753635)

[3. Some Heading 2 2](#_Toc8753636)

[1) Some Sub-heading 1 2](#_Toc8753637)

[2) Some Sub-heading 2 2](#_Toc8753638)

[4. Conclusion 2](#_Toc8753639)

[5. Revision History 3](#_Toc8753640)

[Appendices 4](#_Toc8753641)

[Appendix A 4](#_Toc8753642)

[Bibliography 5](#_Toc8753643)

# Introduction

A data warehouse is a collection of subject-oriented, integrated, time-varying, but relatively stable information itself, used to support the management decision-making process.

Four characteristics of data warehouse:

Topic-oriented: Data warehouses are based on a specific topic, only data related to the topic is required, and other irrelevant details will be excluded

Integrated: collect data from different data sources to the same data source, there will be some ETL operations in this process

Change over time: Implicit or explicit time-based changes in key data

The data of the data warehouse is not updatable: after the data is loaded, only the query operation is generally performed, and there is no addition, deletion or modification of the traditional database. The data of the data warehouse reflects the content of historical data over a long period of time, a collection of database snapshots at different points in time, and the derived data based on these snapshots for statistics, synthesis and reorganization, rather than data processed online.

In this assignment, I made a olap schema based on an OLTP database of Netflix. Here I design some business questions based on the Netflix seniario, and in order to figure out these questions, I design the two fact tables and six dimensions and contains a slowly changing dimension. At last, I implemented the data warehouse using sql and using olap to analyze the seven questions.

# Some key business questions

## Business questions

Which state has rented the most dvds?

Which rating of dvd is the members who wanted to rent?

How many dvds of every genre has been rented and in every month ?

Which months of these years that has the most rentals ?

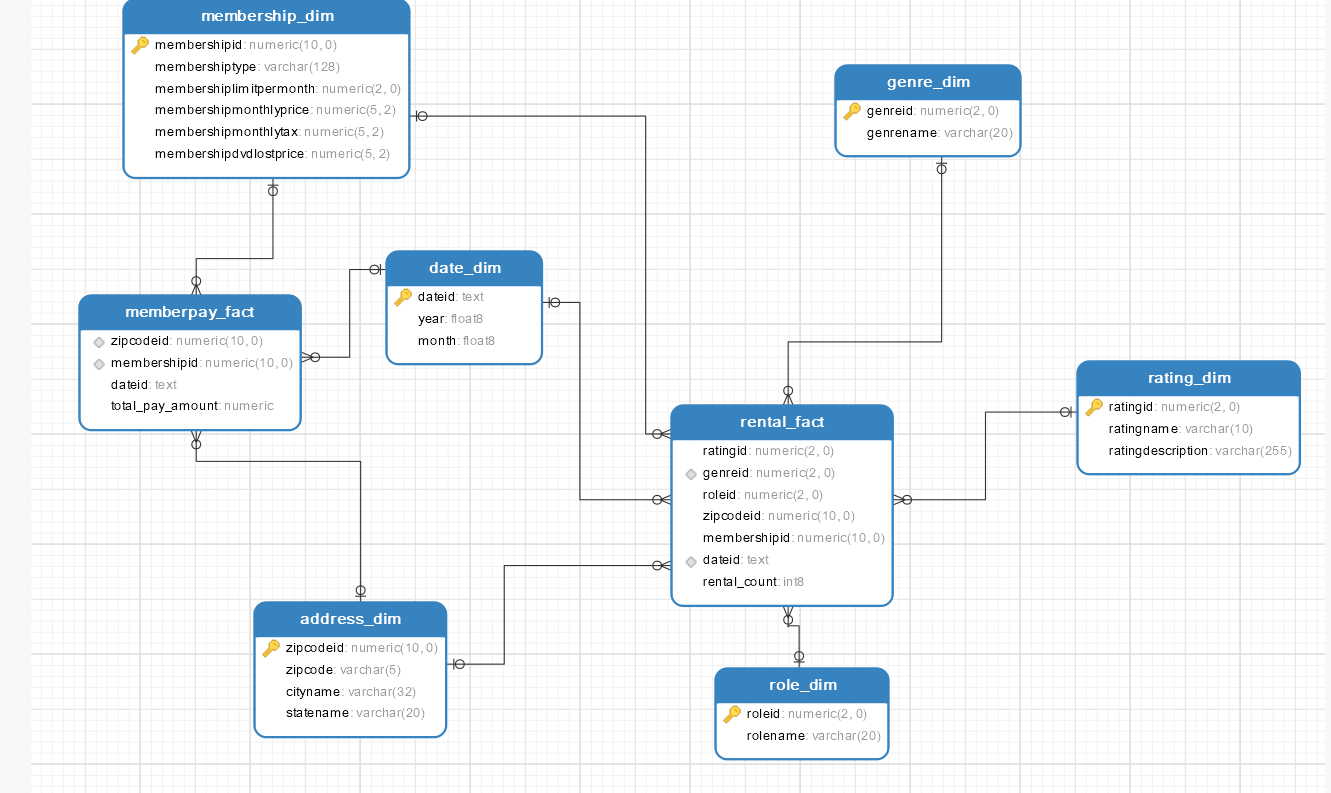
Which months of these years that has the most member fee?

How much money of these cities’ members has paid for memberships and which city is the most ?

Which member type has paid for membership most and which member type has paid for membership least ?

# Design data warehouse

## ERD of data warehouse



## Design facts tables and measures

I design two facts tables: rental\_fact table and memberpay\_fact table.

In the rental\_fact table, I design total\_pay\_amount measure which is measure the total of membership’s fee.

In the member\_pay\_fact table, I design rental\_count measure which is measure the number of dvds’ rental.

## Design dimensions

Dimension tables that are designed as follows:

(1) genre\_dim: genre dimension table, which is represent the genre dimension.

(2) role\_dim: role dimension table, which is represent the role dimension.

(3) rating\_dim: rating dimension table, which is represent the dvd’s rating information.

(4) date\_dim: date dimension table, which is represent the rental’s date or membership purchase’ date.

(5) membership\_dim: membership dimension table, which is represent the membership type and other information and it’s a slowly changing dimension table for it’s membership fee may changing over time.

(6) address\_dim: address dimension table, which is represent the zipcode and city and state of members.

## Design three indexes

In the designed data warehouse above, I design three indexes:

(1) idx\_memship: this index is created for fact table member\_pay\_fact from (membershipid)

The reason is the business question about querying the different membership’s total payment.

(2) idx\_date\_genre: this index is created for fact table rental\_fact from (dateid, genreid)

The reason is the business question of querying the number of the different genre in every month.

(3) idx\_zipcodeid: this index is created for fact table member\_pay\_fact from (zipcodeid)

The reason is the business question about querying the different address’s total payment.

# Implement data warehouse

## Creating dimensions tables

create table dw.genre\_dim  
as  
select \* from genre;  
  
create table dw.role\_dim  
as  
select \* from role;  
  
create table dw.rating\_dim  
as  
select \* from rating;  
  
  
create table dw.date\_dim  
as  
select distinct  
concat(extract(year from r.rentalrequestdate),'-', extract(month from r.rentalrequestdate)) as dateid,  
extract(year from r.rentalrequestdate) as year,  
extract(month from r.rentalrequestdate) as month  
from rental r  
union  
select  
distinct  
concat(extract(year from p.amountpaiddate),'-', extract(month from p.amountpaiddate)) as dateid,  
extract(year from p.amountpaiddate) as year,  
extract(month from p.amountpaiddate) as month  
from payment p  
union  
select  
distinct  
concat(extract(year from p.amountpaiduntildate),'-', extract(month from p.amountpaiduntildate)) as dateid,  
extract(year from p.amountpaiduntildate) as year,  
extract(month from p.amountpaiduntildate) as month  
from payment p;  
  
  
create table dw.membership\_dim  
as  
select \*  
from membership;  
  
create table dw.address\_dim  
as  
select z.zipcodeid, z.zipcode, c.cityname, s.statename  
from zipcode z  
join state s on z.stateid=s.stateid  
join city c on c.cityid=z.cityid;

## Creating facts tables

create table dw.rental\_fact  
as  
select rr.ratingid, g.genreid, ro.roleid, z.zipcodeid, ms.membershipid,  
concat(extract(year from r.rentalrequestdate),'-', extract(month from r.rentalrequestdate)) as dateid,  
count(r.rentalid) as rental\_count  
from rental r  
join dvd d on d.dvdid=r.dvdid  
join rating rr on rr.ratingid=d.ratingid  
join genre g on g.genreid=d.genreid  
join moviepersonrole mr on mr.dvdid=d.dvdid  
join role ro on ro.roleid=mr.roleid  
join member m on m.memberid=r.memberid  
join zipcode z on z.zipcodeid=m.memberaddressid  
join membership ms on ms.membershipid=m.membershipid  
group by rr.ratingid, g.genreid, ro.roleid, z.zipcodeid, ms.membershipid,  
concat(extract(year from r.rentalrequestdate),'-', extract(month from r.rentalrequestdate));  
  
  
create table dw.memberpay\_fact  
as  
select z.zipcodeid, ms.membershipid,  
concat(extract(year from p.amountpaiddate),'-', extract(month from p.amountpaiddate)) as dateid ,  
sum(p.amountpaid) as total\_pay\_amount  
from payment p  
join member m on m.memberid=p.memberid  
join zipcode z on z.zipcodeid=m.memberaddressid  
join membership ms on ms.membershipid=m.membershipid  
group by z.zipcodeid, ms.membershipid,  
concat(extract(year from p.amountpaiddate),'-', extract(month from p.amountpaiddate));

## Creating indexes

* Primary key and foreign key constraints

alter table dw.address\_dim add primary key (zipcodeid);

alter table dw.date\_dim add primary key (dateid);

alter table dw.genre\_dim add PRIMARY key (genreid);

alter table dw.membership\_dim add primary key (membershipid);

alter table dw.rating\_dim add primary key (ratingid);

alter table dw.role\_dim add primary key (roleid);

alter table dw.memberpay\_fact add foreign key (zipcodeid) references dw.address\_dim(zipcodeid);

alter table dw.memberpay\_fact add foreign key (dateid) references dw.date\_dim(dateid);

alter table dw.memberpay\_fact add foreign key (membershipid) references dw.membership\_dim(membershipid);

alter table dw.rental\_fact add foreign key (zipcodeid) references dw.address\_dim(zipcodeid);

alter table dw.rental\_fact add foreign key (dateid) references dw.date\_dim(dateid);

alter table dw.rental\_fact add foreign key (membershipid) references dw.membership\_dim(membershipid);

alter table dw.rental\_fact add foreign key (genreid) references dw.genre\_dim(genreid);

alter table dw.rental\_fact add foreign key (ratingid) references dw.rating\_dim(ratingid);

alter table dw.rental\_fact add foreign key (roleid) references dw.role\_dim(roleid);

* Three indexes

create index idx\_memship on dw.memberpay\_fact(membershipid);

create index idx\_date\_genre on dw.rental\_fact(dateid, genreid);

create index idx\_zipcodeid on dw.memberpay\_fact(zipcodeid);

## OLAP making queries for business questions

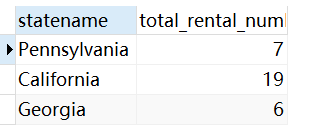
* Which state has rented the most dvds?

select a.statename, sum(r.rental\_count) as total\_rental\_number

from dw.rental\_fact r

join dw.address\_dim a on a.zipcodeid=r.zipcodeid

group by a.statename;



* Which rating of dvd is the members who wanted to rent?

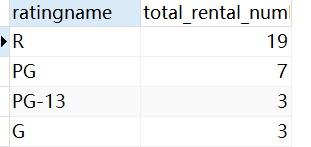
select rd.ratingname, sum(r.rental\_count) as total\_rental\_number

from dw.rental\_fact r

join dw.rating\_dim rd on r.ratingid=rd.ratingid

group by rd.ratingname

order by total\_rental\_number desc;



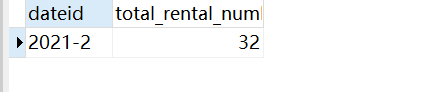
* How many dvds of every genre has been rented and in every month ?

select d.dateid, sum(r.rental\_count) as total\_rental\_number

from dw.rental\_fact r

join dw.date\_dim d on d.dateid=r.dateid

group by d.dateid;



* Which months of these years that has the most rentals ?

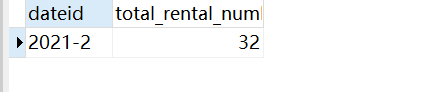
select d.dateid, sum(r.rental\_count) as total\_rental\_number

from dw.rental\_fact r

join dw.date\_dim d on d.dateid=r.dateid

group by d.dateid

order by total\_rental\_number desc limit 1;



* Which months of these years that has the most member fee?

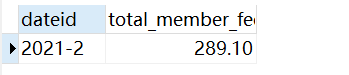
select d.dateid, sum(f.total\_pay\_amount) as total\_member\_fee

from dw.memberpay\_fact f

join dw.date\_dim d on d.dateid=f.dateid

group by d.dateid

order by total\_member\_fee desc limit 1;



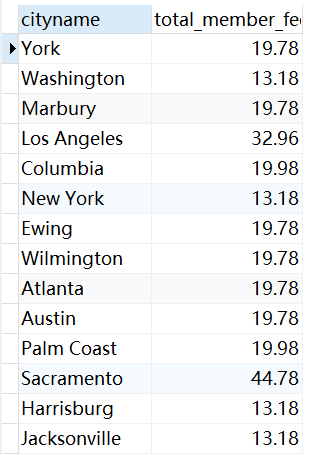
* How much money of these cities’ members has paid for memberships and which city is the most ?

select d.cityname, sum(f.total\_pay\_amount) as total\_member\_fee

from dw.memberpay\_fact f

join dw.address\_dim d on d.zipcodeid=f.zipcodeid

group by d.cityname;



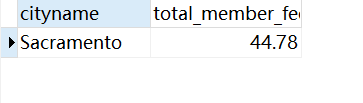
select d.cityname, sum(f.total\_pay\_amount) as total\_member\_fee

from dw.memberpay\_fact f

join dw.address\_dim d on d.zipcodeid=f.zipcodeid

group by d.cityname

order by total\_member\_fee desc limit 1;



* Which member type has paid for membership most and which member type has paid for membership least ?

select \*

from

(

select d.membershiptype, sum(f.total\_pay\_amount) as total\_member\_fee

from dw.memberpay\_fact f

join dw.membership\_dim d on d.membershipid=f.membershipid

group by d.membershiptype

order by total\_member\_fee desc limit 1

) t

union

select \*

from

(

select d.membershiptype, sum(f.total\_pay\_amount) as total\_member\_fee

from dw.memberpay\_fact f

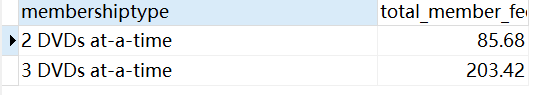
join dw.membership\_dim d on d.membershipid=f.membershipid

group by d.membershiptype

order by total\_member\_fee limit 1

) t2

;



# Conclusion

In this assignment, I have focused on the data warehouse designing and implementing, besides, I tried to figure out the data warehouse’ concept and learn how to choose the dimensions and fact measures.

In the process of completing the assigment, I read the books and materials related to the data warehouse, and realized the magical effect of the data warehouse in practice.

# 5. Revision History

A history of things you added and why, not required but nice to have.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Version** | **Description** |
| Your Name | 01/09/19 | 1.0 | Initial Document Creation |
| Your Name | 01/15/19 | 1.1 | Added more headers to provide better example |

# Appendices

## Appendix A

Fundamentals of Data Warehouses

The data warehouse lifecycle toolkit : expert methods for designing, developing, and deploying data warehouses

# Bibliography

Ballve, M. (2014, January 18). Apps Rule The Phone, But The Mobile Web Is Still Alive And Well. Retrieved from http://www.businessinsider.com/html5-and-the-web-is-still-relevant-2014-1

Darlington, K. W. (2011). Designing for Explanation in Health Care Applications of Expert Systems. *SAGE Open*, 1-9.

Medlock, S. e. (2011). LERM (Logical Elements Rule Method): A method for assessing and formalizing clinical rules for decision support. *International Journal of Medical Informatics*, 286-295.

Whitten, J. L. (2007). *Systems Analysis and Design Methods.* New York: McGraw-Hill/Irwin.