DBMS Project Report

PES University

Database Management Systems

Microsoft services Recommendation System

Submitted By,

Nikhil Reddy M

19BCT0108

Introduction

We live in a world that had many technological improvements in the last century. Most of the million-dollar companies like Microsoft, Google, Apple and Amazon today get their market or provide market to other companies through recommendation systems and make money.

Recommendation systems relies completely on database and trained machine learning and deep learning models to achieve their market and user’s trust and awareness of products of either organizations or companies.

Recommendations can be done through several processes like proving Ads through Websites, Search suggestions, Product Recommendations. Companies store the data in their data base and train models to show relative ads and similar or compact product recommendations.

The link for the queries is given below:

https://1drv.ms/u/s!AtmotjFzhXrLbj1LTloyAMpxREo?e=Mm7xCx

Data Model

Microsoft\_Services

|  |  |
| --- | --- |
| Type\_of\_Service | Product |

Product\_Id\_Architecture

|  |  |  |
| --- | --- | --- |
| Product\_A | System\_Architecture | Product\_Id |

Product\_Version

|  |  |
| --- | --- |
| Product\_Id\_V | Product\_Versional |

Recommendation\_System

|  |  |  |
| --- | --- | --- |
| User\_Id | Surfing\_Keywords | Recommended\_Products\_WA |

Abstract for Data Model:

Microsoft\_Services Table defines wide range of products from its different services. These products are given product id’s based on their architecture type in Product\_Architecture Table. Each product id can have different versions of products in Product\_Version Table. Every time user surfs or visits a website gets a user id generated. Through the keywords in the search or the web address the user gets the recommended Microsoft products in Recommendation\_System Table.

ER Diagram

Product\_Id\_Architecture

Microsoft\_Services

Product\_Version

1 N

Version

Compatibility

N 1

Recommendation\_System

Recommends

1

1

Abstract for ER Diagram:

In the above ER Diagram, Microsoft\_Services Table has Product as a primary key – every product in any service is unique. Each Product differs in compatibility Table Product\_Id\_Architecture has Product\_Id as primary key to uniquely identify them and has Product\_A as a foreign key from Microsoft\_Services Table, having Compatibility as a N:1relationship. Every Product\_Id has different product versions. Thus, Product\_Version Table has Product\_Versional as primary key and Product\_id\_V as a foreign key from Product\_Id\_Architecture having Version as a N:1 relationship.

Recommandation\_System Table has a User\_Id as a primary key since each time user enters a website or surfs the net, user gets unique User\_Id. It has Recommends as a 1:1 relationship with Product\_Version Table.

Normalization

We try to remove the duplicate data either existing or to be created after joining tables or using DML commands. Hence, to achieve data redundancy and improve data integrity we make sure to follow maximum number of normal forms.

First Normal form: All the tables in the dataset are meaningful single valued attributes with proper domain values in the dataset. Thus, all tables satisfy First Normal form.

Ex: Table Microsoft\_Services – Attributes Type\_of\_Service and Product are single valued. If there was Service\_Address it would fail 1NF.

Second Normal form: Since there is no composite key attribute in any of the tables and each table has a maximum of three attributes partial dependency does not exist in this database.

Ex: Table Product\_Version – Attributes Product\_Id\_V and Product\_Versional does not have any partial dependency since it has only two attributes.

Third Normal form: Transitional dependency does not exist in any of the tables, thus all tables satisfy Third Normal form.

Ex: Tables Product\_Id\_Architecture and Product\_Version –

Product\_A and System\_Architecture produces unique Product \_Id, Product\_Id\_V produces unique Product\_Versional but Product\_A and System\_Architecture does not produce any values in attribute Product\_Versional.

BCNF: There isn’t even a table that has composite key, thus non-prime key concept of candidate keys are ruled out.

Ex: Table Recommendation\_System has unique primary key User\_Id thus follow BCNF. If User\_Id and Time was created as a composite key, then same data is stored many a times so there needs a creation of other table to create a unique User\_Id.

Fourth Normal form: There is no Multivalued dependency in any of the tables since there are no two attributes which are not related to each other but depends on primary key.

Ex: If there was Product\_leader for each Version of the Product\_Id\_V then it would have violated 4NF since there may be same project leader for more than one version or for another Product\_Id\_V.

Fifth Normal form: Join Dependency is a feature that defines Fifth Normal form. Only one table doesn’t fall back on 5 NF.

Ex: In System\_Id\_Architecture Table any of the attributes can be seperated then joined without any changes but in Table Recommendation\_System Surfing\_Keywords and Recommended\_Products\_WA can be separated from User\_Id, while re-joining those two attributes with User\_Id we do not get the same table.

Functional Dependencies

1. Product 🡪 {Type\_of\_Service}
2. Product\_Id 🡪 {Product\_A , System\_Architecture}
3. Product\_Versional 🡪 {Product\_Id\_V}
4. User\_Id 🡪 {Surfing\_Keywords , Recommend\_Products\_WA}

DDL

Table: Microsoft\_Services

CREATE TABLE Microsoft\_Services(

Type\_of\_Service varchar(20) NOT NULL,

Product varchar(20) NOT NULL,

PRIMARY KEY(Product));

Table: System\_Id\_Architecture

CREATE TABLE System\_Id\_Architecture(

Product\_A varchar(20) NOT NULL,

System\_Architecture varchar(20) NOT NULL,

Product\_Id INT NOT NULL,

PRIMARY KEY(Product\_Id),

FOREIGN KEY(Product\_A) REFERENCES Microsoft\_Services(Product) ON DELETE CASCADE);

Table: Product\_Version

CREATE TABLE Product\_Version(

Product\_Id\_V INT NOT NULL,

Product\_Versional varchar(20) NOT NULL,

PRIMARY KEY(Product\_Versional),

FOREIGN KEY(Product\_Id\_V) REFERENCES System\_Id\_Architecture(Product\_Id) ON DELETE CASCADE);

Table: Recommendation\_System

CREATE TABLE Recommendation\_System(

User\_Id SERIAL,

Surfing\_Keywords varchar(20) NOT NULL,

Recommended\_Products\_WA varchar(20) NOT NULL,

PRIMARY KEY(USER\_Id));

TRIGGERS

1. Trigger to avoid Product\_id and Product\_Versional mismatch during updation.

CREATE OR REPLACE FUNCTION Raise\_Mismatch()

RETURNS trigger AS

$$

BEGIN

IF NEW.Product\_Versional NOT LIKE (CAST(New.Product\_Id\_V as varchar(20))||'\\_' || '%') THEN

RAISE EXCEPTION 'Part of Product version prior to \_ symbol should be same as product id';

END IF;

RETURN NEW;

END;

$$

Language plpgsql;

CREATE TRIGGER Id\_Version\_Mismatch

BEFORE INSERT OR UPDATE

ON Product\_Version

FOR EACH ROW

EXECUTE PROCEDURE Raise\_Mismatch();

Update Product\_Version

SET Product\_Id\_V=6

WHERE Product\_Versional='1\_1.0';

SELECT \*

FROM Product\_Version

WHERE Product\_Versional='1\_1.0';

2.Trigger to log the insertion or updation of Product name.

CREATE TABLE IF NOT EXISTS Changes(

Idd SERIAL PRIMARY KEY,

Product\_Change varchar(20) NOT NULL,

Changed\_on TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP);

CREATE OR REPLACE FUNCTION log\_Change\_don()

RETURNS trigger AS

$$

BEGIN

IF (New.Product IS NOT NULL) OR TG\_OP='DELETE' THEN

INSERT INTO CHANGES (Idd, Product\_Change,Changed\_on)

VALUES (DEFAULT, NEW.Product,DEFAULT);

ELSE

INSERT INTO Changes(Idd,Product\_Change,Changed\_on)

VALUES( DEFAULT, OLD.Product,DEFAULT);

END IF;

RETURN NEW;

END;

$$

Language plpgsql;

CREATE TRIGGER Name\_changes\_don

BEFORE INSERT OR UPDATE

ON Microsoft\_Services

FOR EACH ROW

EXECUTE PROCEDURE log\_Change\_don();

INSERT INTO Microsoft\_Services(Type\_of\_Service, Product)

VALUES

('Mobile\_Services','Simple');

SELECT \* FROM Microsoft\_Services;

SELECT \* from Changes;

Update Microsoft\_Services

SET Product='Sample'

WHERE Product='Simple';

SELECT \* from Changes;

SQL Queries

1.Show all Product Versions of a Bing.

Select Product\_Versional

From Product\_Version

where product\_id\_v IN

(

select product\_id\_v

from product\_version,system\_id\_architecture

where product\_a='Bing' AND

Product\_id=Product\_id\_v

)

order by Product\_Versional;

2.Given Product\_Versional determine type of Service.

select DISTINCT type\_of\_service

from microsoft\_services,system\_id\_architecture

where product In

(select distinct product\_a

from system\_id\_architecture,product\_version

where

product\_id=product\_id\_v AND

product\_versional='29\_1.2');

3.Given a type of service find number of product\_ids available

select count(product\_id)

from microsoft\_services, system\_id\_architecture

where type\_of\_service='Mobile\_Services' and

product\_a=product;

4.Given a set of recommended\_websites determine what user id who was first to access any of the above ones.

select min(user\_id)

from recommendation\_system

where recommended\_products\_wa='visual.com' or

recommended\_products\_wa='visualcode.com';

1. Join System\_id\_architecture and product\_version table

select \* from system\_id\_architecture

full outer join product\_version

on product\_id=product\_id\_v;

6.Name the Products along with their frequncy of Recommendation/ads which are most seen

Select Product, count(Recommended\_Products\_WA)

from Recommendation\_System, Microsoft\_Services

where Product||'.com' ILIKE Recommended\_Products\_WA

GROUP BY Recommended\_Products\_WA,Product

ORDER BY count(Recommended\_Products\_WA) DESC;

7.Name the Products on which the company hasn’t given Recommendations yet or the products on which they have to give recommendations on assuming every product has same importance.

SELECT Product

from Microsoft\_Services

where lower(product)||'.com' not in

(select Recommended\_Products\_WA

from Recommendation\_System);

CONCLUSION

Recommendation systems have boosted revenue for big companies for about 20-30%. It brings awareness about their products. Thus, Big Data and AI have become outgrowing every year. Companies spend a lot of Money to store data and handle secure Servers with Databases.

BILOGRAPHY

1.studytonight.com

2.postgresqltutorial.com