# 프로젝트 보고서

Team05: 1876092 김지수, 1785054김진유

목차

1. 팀 구성 및 진행일정
   1. 팀 구성
   2. 진행 일정
2. 프로젝트 소개
   1. 프로젝트 선정 계기
   2. 프로젝트 의의 및 목적
3. Database schema
4. 역할분배
5. PHP code overview
   1. 페이지 구성
   2. Requirements
6. 팀 구성 및 진행일정
   1. 팀구성

1) 팀 번호 : 05

2) 팀 명 : 빅뎅잉텅응용

3) 팀원 : 1876092 김지수, 1785054김진유

1.2 진행일정

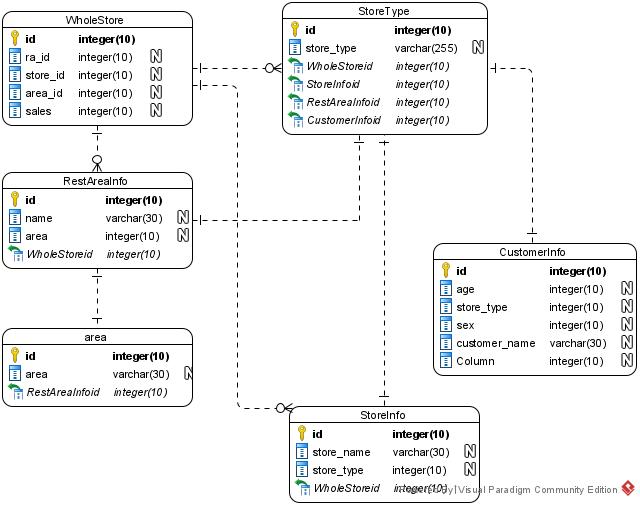
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 일자 |  | 10/13 | 10/14-20 | 10/21-27 | 10/27-11/3 | 11/3-9 | 11/10-16 | 11/17-22 |
| 주제 선정 및  proposal 작성 |  |  |  |  |  |  |  |  |
| DB 모델링 및 구성 |  |  |  |  |  |  |  |  |
| 웹페이시 구상 및 요구사항 작성 |  |  |  |  |  |  |  |  |
| DB제작 및 초기화  Script 작성 |  |  |  |  |  |  |  |  |
| Index.html 제작 |  |  |  |  |  |  |  |  |
| Rest.area, customer\_ranking등 php script 제작 |  |  |  |  |  |  |  |  |
| Advanced Analysis 기능 구현 |  |  |  |  |  |  |  |  |
| 문서작업 |  |  |  |  |  |  |  |  |

1. 프로젝트 소개

2.1 프로젝트 선정 계기

2.2 프로젝트 의의 및 목적

1. DATABASE SCHEMA



1. 역할분배
   1. 김지수

- 프로젝트 주제 선정

- Table creation and Initialization (StoreType, StoreInfo, RestAreaInfo)

- login PHP session

- customer survey ( INSERT )

- customer\_correct ( DELETE, UPDATE )

- 매장타입별 전국 분포 Pivot Table

- 연령 별 선호 매장 타입 1~5위

- 해당 매장 타입을 선호하는 소비자들의 평균 연령

- rollup (휴게소 별 매출 합계)

* 1. 김진유

- project proposal

- Table creation and initialization (WholeStore, CustomerInfo, Area)

- 전체 휴게소별 매장 목록

- 전국 휴게소 목록

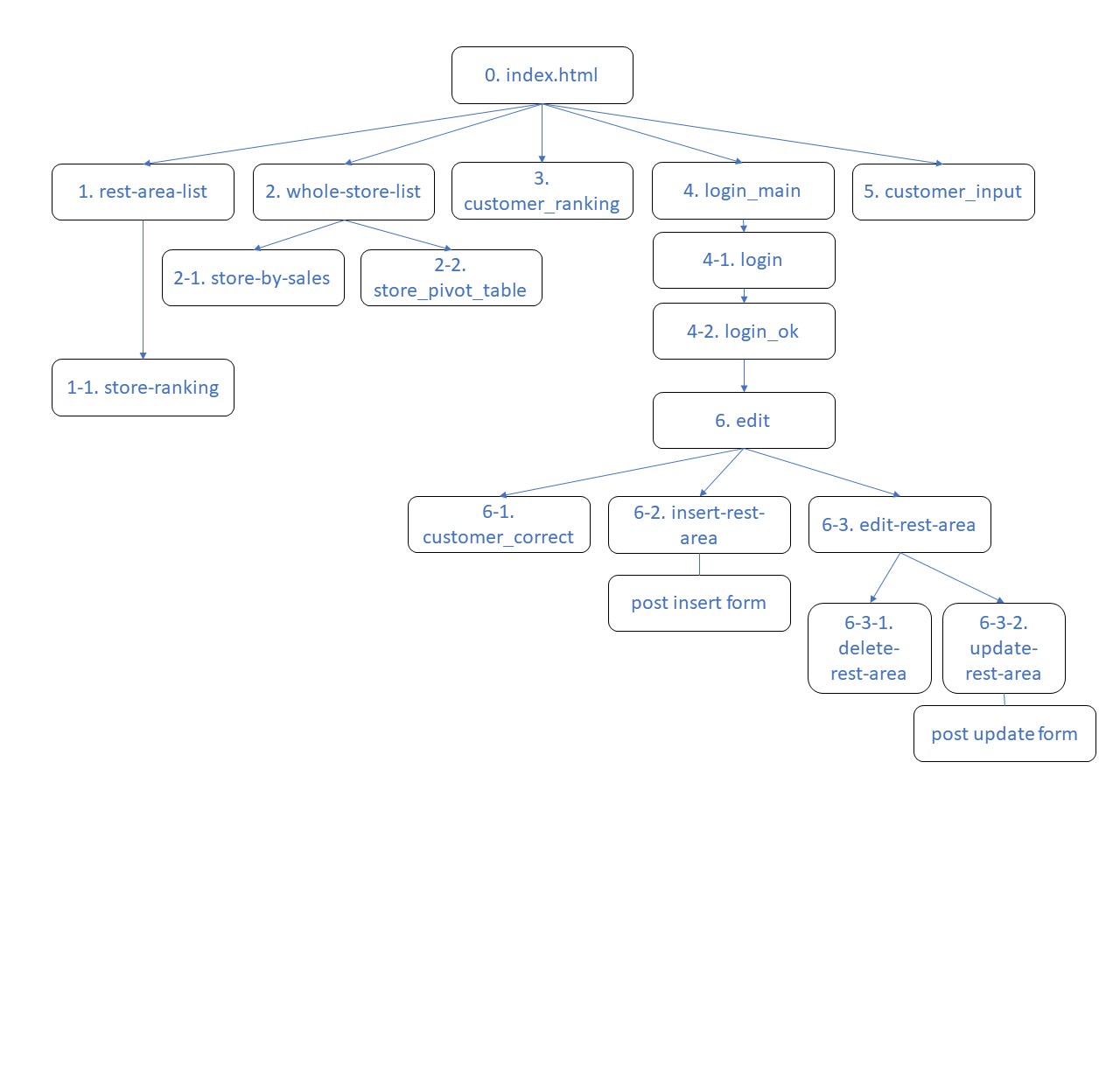
- 휴게소 추가 및 수정 ( INSERT, DELETE, UPDATE )

- 휴게소 내 매장 별 매출 순위 확인

- 매출 구간 별로 휴게소 매장 확인

- Data transaction implementation

1. PHP code overview



* 1. 페이지 정보

0. index : 시작 페이지

* + 1. rest-area-list : 휴게소 정보 ( id, name, area, sales ) 출력
       1. store-ranking : 휴게소 내 매장별 매출 순위 확인
    2. whole-store-list : 휴게소 별 전체 매장을 출력. foreign key로 설정된 휴게소id, 매장id, 지역id 를 통해 각각의 이름을 가져옴
       1. store-by-sales : 사용자가 알기를 원하는 매출 구간에 있는 매장 목록을 불러옴
       2. store-pivot-table : 매장 종류에 따른 지역 별 분포정도를 표시함
    3. customer\_ranking : 사용자 연령에 따른 선호 매장 타입을 알 수 있음
    4. login\_main : 로그인 과정을 거쳐 권한이 있는 사용자만 정보 수정 및 추가 가능
    5. customer\_input : 새로운 소비자 정보를 입력하는 페이지
    6. edit : 소비자 정보 및 휴게소 정보 수정 및 삭제
       1. customer\_correct : 소비자 정보 수정 및 삭제
       2. insert-rest-area : 휴게소 정보 추가
       3. edit-rest-area : 휴게소 정보 변경 및 삭제
  1. Requirements
     1. Theme of the project can be one of the following

- 휴게소 음식점에 관한 연령 별 소비 경향과 매출 추이를 비교 분석할 수 있다.

* + 1. Database creation scripts and data insert scripts should be created (SQL scripts)

- table creation sql : 아래의 WholeStore를 포함한 총 6개 table생성

- data insert sql : 아래의 값을 포함하여 총 238개의 record 생성

CREATE TABLE WholeStore(

                id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT, ra\_id INT, store\_id INT, area\_id INT, sales INT,

                FOREIGN KEY(ra\_id) REFERENCES RestAreaInfo (id) ON DELETE CASCADE ON UPDATE CASCADE,

                FOREIGN KEY(store\_id) REFERENCES StoreInfo (id) ON DELETE CASCADE ON UPDATE CASCADE,

                FOREIGN KEY(area\_id) REFERENCES RestAreaInfo (area)

ON DELETE CASCADE ON UPDATE CASCADE)

INSERT INTO WholeStore (ra\_id, store\_id, area\_id, sales) VALUES

        (14,1,5,31300),(13,30,5,46500),(15,43,4,21200),(16,23,4,69800), …

* + 1. INSERT, DELETE, UPDATE, SELECT related functions should be implemented in the Web site.
    2. Some SELECT queries should include SUM, AVG, etc (aggregation operations) and also GROUP BY statements

- SUM : 휴게소의 전체 매출액 구하기 위해 해당 휴게소에 속한 매장의 매출 합을 SUM으로 구했다.

SELECT area, name,store\_name, SUM(sales) FROM(SELECT area, name, store\_name,sales

        FROM(SELECT sub1.area\_id,area.area, sub1.store\_id, sub1.name, sub1.sales

        FROM(SELECT wholestore.area\_id, wholestore.store\_id AS store\_id,restareainfo.name

AS name, sales, restareainfo.id as ra\_id FROM wholestore

INNER JOIN  restareainfo ON wholestore.ra\_id=restareainfo.id)sub1

INNER JOIN area ON sub1.area\_id=area.id ORDER BY sub1.ra\_id)sub2

INNER JOIN storeinfo ON storeinfo.id = sub2.store\_id)sub3 GROUP BY name, store\_name WITH ROLLUP

- AVG, GROUP BY : 특정 매장 타입을 선호하는 연령의 평균을 구하기 위해 AVG와 GROUP BY를 사용했다.

SELECT storetype.store\_type, AVG(customerinfo.age) FROM customerinfo

INNER JOIN storetype ON customerinfo.store\_type = storetype.id GROUP BY store\_type

* + 1. Provide at least 5 kinds of advanced analysis functions to users for the data that is stored in the database

1) CASE/WHEN/THEN : 매장을 매출 구간 별로 나누기 위해 CASE문을 사용하였다.

2) RANKING : 연령대 별 선호하는 매장 타입의 순위를 가져오기 위해 RANK()를 사용하였다.

SELECT \* FROM (SELECT sub1.srange, sub1.id, ra.name, s.store\_name, sub1.sales

    FROM (SELECT id, ra\_id, store\_id, sales,

            (CASE

        WHEN sales <= 20000 THEN '0-20,000'

        WHEN sales > 20000 AND sales <= 40000 THEN '20,000-40,000'

        WHEN sales > 40000 AND sales <= 60000 THEN '40,000-60,000'

        WHEN sales > 60000 AND sales <= 80000 THEN '60,000-80,000'

        WHEN sales > 80000 THEN 'more than 80,000'

          END) as srange

    FROM wholestore) as sub1

    inner join restareainfo as ra on sub1.ra\_id=ra.id

    INNER JOIN storeinfo as s on sub1.store\_id=s.id

    ORDER by sub1.srange) as sub2

    WHERE sub2.srange='$range' ORDER BY sub2.sales

SELECT sub1.store\_type, COUNT(sub1.id), RANK() OVER (ORDER BY COUNT(sub1.id) DESC)

FROM (SELECT storetype.store\_type, storetype.id, age FROM customerinfo

INNER JOIN storetype ON customerinfo.store\_type = storetype.id

WHERE (age BETWEEN {$bottom} AND {$top})) AS sub1

GROUP BY sub1.store\_type ORDER BY RANK() OVER (ORDER BY COUNT(sub1.id) DESC)

3) PIVOT TABLE : 지역을 row로 매장 타입을 column으로 가져와 어떤 매장 종류가 전국에 분포되어 있는지를 나타냈다.

4) ROLLUP grouping : 이거 적어줘

SELECT area,

sum(CASE WHEN store\_type='한식' THEN cnt ELSE 0 END)AS 한식,

sum(CASE WHEN store\_type='카페' THEN cnt ELSE 0 END)AS 카페,

sum(CASE WHEN store\_type='분식' THEN cnt ELSE 0 END)AS 분식,

sum(CASE WHEN store\_type='중식' THEN cnt ELSE 0 END)AS 중식,

SUM(CASE WHEN store\_type='양식' THEN cnt ELSE 0 END)AS 양식,

sum(CASE WHEN store\_type='일식' THEN cnt ELSE 0 END)AS 일식,

sum(CASE WHEN store\_type='간식' THEN cnt ELSE 0 END)AS 간식,

SUM(CASE WHEN store\_type='편의점' THEN cnt ELSE 0 END)AS 편의점,

SUM(CASE WHEN store\_type='기타' THEN cnt ELSE 0 END) AS 기타

FROM (SELECT ex2.area,d.store\_type, COUNT(ex2.area) AS cnt,d.id

FROM (SELECT ex1.area, c.store\_type FROM(SELECT a.area\_id, a.store\_id, b.area

FROM wholestore AS a JOIN area AS b ON a.area\_id=b.id) ex1

JOIN StoreInfo as c ON c.id = ex1.store\_id)ex2 JOIN storetype as d ON d.id= ex2.store\_type

GROUP BY ex2.area, d.store\_type)sample GROUP BY area

SELECT area, name,store\_name, SUM(sales) FROM(SELECT area, name, store\_name,sales

        FROM(SELECT sub1.area\_id,area.area, sub1.store\_id, sub1.name, sub1.sales

        FROM(SELECT wholestore.area\_id, wholestore.store\_id AS store\_id,restareainfo.name

AS name, sales, restareainfo.id as ra\_id FROM wholestore

INNER JOIN  restareainfo ON wholestore.ra\_id=restareainfo.id)sub1

INNER JOIN area ON sub1.area\_id=area.id ORDER BY sub1.ra\_id)sub2

INNER JOIN storeinfo ON storeinfo.id = sub2.store\_id)sub3 GROUP BY name, store\_name WITH ROLLUP

5) DENSE\_RANK() OVER (PARTITION BY ra\_id ORDER BY sales) : 한 휴게소 내의 매장 별 매출 순위를 나타내기 위해 사용하였다.

SELECT w.id, ra.name, s.store\_name, w.sales, (DENSE\_RANK() OVER (PARTITION BY ra\_id ORDER BY sales)) as ranking FROM wholestore w

          INNER JOIN restareainfo as ra on w.ra\_id=ra.id

          INNER JOIN storeinfo as s on w.store\_id=s.id";

* + 1. User input should include various types of controls such as text box, buttons, etc. (Not just a single text box and single submit button!)

- 휴게소 목록을 추가하는 html form 양식 : 사용자에게 휴게소 이름과 지역 id를 받아서 POST한다.

- 이외에 소비자 정보를 추가하는 customer\_input.php 생성.

<form method="POST">

<input type="text" name="name" value="<?php echo $data['name'] ?>" placeholder="Enter Name" Required>

        <select name="area" size=5 value="<?php echo $data['area']?>">

                        <option value="1">서울경기</option>

                        <option value="2">부산경상</option>

                        <option value="3">강원</option>

                        <option value="4">충청</option>

                        <option value="5">전라</option>

                    </select>

        <input type="submit" name="update" value="Update">

</form>

* + 1. Use PHP sessions to store and retrieve information needed between some pages displayed

- 정보를 수정하는 페이지에 들어가기 위해 로그인 기능을 추가하였다. login 상태를 유지하기 위해 session을 사용한다.

session\_start();

$\_SESSION['user\_id'] = $user\_id;

* + 1. Use at least one transaction in the database for some part of the project.

- 사용자 정보를 수정할 때 잘못된 값을 입력받는 경우 그 정보가 db에 저장되지 않도록 transaction을 사용하였다. 잘못된 값이 들어오는 경우 rollback으로 되돌린다.

$conn->begin\_transaction();

            try {

                $id=$\_POST["id"];

                $age=$\_POST["age"];

                $store\_type=$\_POST["customer\_combobox"];

                $sex=$\_POST["sex"];

                $name=$\_POST["customer\_name"];

                $sql="UPDATE CustomerInfo SET age=".$age.", store\_type=".$store\_type.", sex=".$sex.", customer\_name='".$name."' WHERE id=".$id.";";

                mysqli\_query($conn,$sql);

                $conn->commit();

            } catch (TypeError $ex) {

                echo $ex->getMessage();

                $conn->rollback();

                throw $ex;

            }