















GAME_DB 파일 구성 및 실행 순서

2014147033 손명현

-  Alter Constraints.sql
-  CharacterConnect.csv
-  CharacterPayment.csv
-  Create Procedure DungeonCheck.sql
-  Create Procedure RFM_Analysis.sql
-  Create Procedure UserCharcterInsertion.sql
-  Create Tables.sql
-  Create Trigger UpdateLastConnect.sql
-  Create Views.sql
-  DataGeneratingProc.ipynb
-  Insert Data.sql
-  InsertData.xlsx
-  Prereq_Create Procedure LastRecordUpdate.sql
-  Query Examples.sql

SQL 파일, 그리고 Data insert를 위한 csv 및 xlsx 파일이 있습니다.

- 실행 순서

1. 새로운 Schema 생성
2. Data Insertion
 - a. MySQL WorkBench에서 Create Tables.sql, Insert Data.sql을 차례대로 실행
 - b. 네비게이터에서 Table Data Import Wizard 기능을 이용하여 CharacterConnect.csv의 데이터를 CHARACTER_CONNECT Table에 INSERT함
 - c. 마찬가지로 CharacterPayment.csv의 데이터를 CHARACTER_PAYMENT Table에 INSERT함
 - d. Prereq_Create Procedure LastRecordUpdate.sql 실행
 - e. Alter Constraint.sql 실행
3. Create Views.sql 실행
4. Trigger와 Procedure 정의가 담긴 Create Procedure ~.sql, Create Trigger ~.sql 실행
5. Query Examples.sql에서 응용 서비스 작동을 확인

- 구현한 응용 서비스

1. 3명 또는 4명이 입장할 수 있는 Dungeon에서 입장하는 캐릭터들이 Dungeon의 입장 조건을 충족하는지를 판단하고 Accept할지, Reject할지를 판단
2. User_Character의 게임 캐릭터 생성 프로시저 : 하나의 User는 6개보다 많은 캐릭터를 생성할 수

없습니다.

3. User_Character에서 마지막 접속 날짜 및 마지막 접속 시간을 Character_Connect Table 새로 추가되는 Instance를 이용해 갱신

4. RFM Analysis

InsertData.xlsx에는 USER, USER_CHARACTER, CLASS Table에 들어갈 데이터가 있으며, 이는 InsertData.sql문에 구현하였습니다.

CHARACTER_CONNECT, CHARACTER_PAYMENT에 들어갈 데이터는 수작업으로 생성하기 어려워, 기존에 존재하는 USER, USER_CHARACTER, CLASS Table의 값에 따라 적절한 데이터를 무작위로 생성하였습니다. 데이터 생성 프로세스는 DataGeneratingProc.ipynb에 담겨 있습니다(ipynb파일을 다시 실행할 필요는 없습니다.).

Prereq_Create Procedure LastRecordUpdate.sql은 응용 서비스가 아니며, 예시 데이터를 생성하여 DB를 구축하는 과정이 존재하기에 적합성이 맞지 않는 부분이 존재하기에 그 부분을 수정하기 위한 프로시저입니다. 예를 들어, USER_CHARACTER Table에는 LastConnectDate, LastPlayedHour라는, 가장 마지막에 게임을 플레이한 날짜와 시간이 담겨 있는데 이는 CHARACTER_CONNECT에서 Instance가 생성될 때마다 Update 됩니다. 그러나 현재는 일괄적으로 모든 데이터를 INSERT하였기 때문에, USER_CHARACTER Table의 LastConnectDate, LastPlayedHour가 모두 NULL값을 가지고 있습니다. LastRecordUpdate 프로시저는 그러한 불완전한 부분을 CHARACTER_CONNECT Table에서의 가장 마지막 접속 시의 값으로 채우는 과정입니다.

모든 데이터가 채워지면, Alter Constraint.sql을 실행해서 AUTO INCREMENT 조건과 FK 조건을 테이블들에 추가합니다.

Create Views.sql에는 각 User의 게임 접속 내역에 대한 View와 결제 내역에 대한 View를 정의하는 코드가 담겨 있습니다. 두 개의 View를 이용해 이후 RFM Analysis를 수행하는 프로시저를 작성했습니다.

Create Procedure DungeonCheck.sql에는 Dungeon에 입장하려는 캐릭터들의 모임이 Dungeon의 입장 조건을 충족하는지 여부를 판단하는 Procedure가 구현되어 있습니다. 캐릭터들의 Class 조합은 균형 잡힌 조합으로 이루어져야 하며, 이는 CLASS Table에 있는, 각 Class들의 능력치와 Procedure 내부의 기준에 따라 결정됩니다. 자세한 내용은 Query Examples.sql에 담겨 있습니다.

Create Procedure UserCharacterInsertion.sql에는 USER_CHARACTER에 새로운 데이터가 추가됐을 때, 즉 특정 User가 새로운 게임 캐릭터를 생성했을 때 실행되는 Procedure를 구현하였습니다. Procedure는 만약 User가 이미 6개의 게임 캐릭터를 가지고 있다면 캐릭터 생성을 Reject하며, 그렇지 않다면 LastConnectDate와 LastPlayedHour 컬럼을 NULL로 하여 USER_CHARACTER에 새로운 캐릭터에 관한 instance를 추가합니다.

Create Trigger UpdateLastConnect.sql에는 CHARACTER_CONNECT에 새로운 데이터가 추가됐을 때, 즉 특정 User가 특정 Character로 게임에 접속하여 플레이를 했을 때 실행됩니다. CHARACTER_CONNECT에 특정 캐릭터의 새로운 데이터가 추가되면, 추가된 데이터의 ConnectDate와 ConnectTime이 USER_CHARACTER 테이블에서 그 캐릭터의 LastConnectDate, LastPlayedHour로 기록됩니다.

Create Procedure RFM_Analysis.sql에는 User가 얼마나 최근에 게임에 접속했는지, 얼마나 많은 시간 플레이하였는지, 그리고 얼마나 많은 돈을 게임에 사용했는지를, 위의 Create Views.sql을 이용해 만든 두 개의 View를 이용하여 분석하는 Procedure가 구현되어 있습니다. 수업 때 배웠던 것과 마찬가지로, R, F, M 각 기준에 따라 상위 20%를 Rank 1로 하여, 20%씩 증가할수록 Rank가 낮아집니다. USER_RFM Table에 RFM

Analysis를 위한 데이터가 생성됩니다.