2주차 과제

마이그레이션이란?

우선 과제를 진행하기 앞서 `마이그레이션` 이 무엇인지 명확하게 하였습니다.

> 💡 **마이그레이션**(migration)이란 한 종류의 데이터베이스에서 다른 종류의 데이터베이스로 데이터를 옮기는 것을 의미한다. DB 마이그레이션, DB 이전, DB 이관, DB 이행이라고도 불린다.

1. 서비스 구조 파악

현재 무코챗은 `UserEntity`, `ChatJoinEntity`, `ChatRoomEntity`, `ChatEntity`, `ChatLikeEntity`, `ContentEntity`, `ContentLikeEntity` 등 7개의 엔티티가 포함되어 있습니다.

각 테이블의 간략한 설명은 아래와 같습니다.

UserEntity

사용자 정보를 관리하는 테이블입니다. 각 사용자는 고유한 ID를 가지며, 이 ID는 다른 테이블에서 사용자를 참조할 때 사용됩니다.

ChatRoomEntity

채팅방 정보를 관리하는 테이블입니다. 각 채팅방은 고유한 ID와 이름을 가지며, 채팅방에 공지사항으로 설정된 채팅 메시지(`noticeChat`)를 가질 수 있습니다.

ChatJoinEntity

사용자가 채팅방에 참여한 정보를 관리하는 테이블입니다. 각 레코드는 사용자 ID(`userId`)와 채팅방 ID(`roomId`)의 조합으로, 어떤 사용자가 어떤 채팅방에 참여했는지를 나타냅니다.

ChatEntity

채팅 메시지를 관리하는 테이블입니다. 각 메시지는 보낸 사람(`senderId`), 속한 채팅방(`roomId`), 메시지 내용(`content`) 등의 정보를 가집니다.

ChatLikeEntity

채팅 메시지의 좋아요 정보를 관리하는 테이블입니다. 각 레코드는 사용자 ID(`userId`)와 채팅 메시지 ID(`chatId`)의 조합으로, 어떤 사용자가 어떤 메시지를 좋아했는지를 나타냅니다.

ContentEntity

콘텐츠(게시물) 정보를 나타내는 테이블입니다. 각 콘텐츠는 고유한 ID, 제목, 내용 그리고 생성자 정보를 가집니다. `creatorId`는 게시물을 작성한 사용자의 ID를 참조하며, `creator`는 이를 연결하는 관계를 나타냅니다. `createdAt`은 게시물이 생성된 날짜를 나타냅니다.

ContentLikeEntity

콘텐츠에 대한 좋아요 정보를 나타내는 테이블입니다. 각 레코드는 좋아요를 누른 사용자의 ID(`userId`)와 대상 콘텐츠의 ID(`contentId`)를 가집니다. `user`와 `post`는 각각 사용자와 콘텐츠 엔티티를 참조하는 관계를 나타냅니다. `createdAt`은 좋아요가 등록된 날짜를 나타냅니다.

2. 마이그레이션 대상 테이블 선정

데이터베이스 마이그레이션 대상 테이블을 선정할 때는 다음과 같은 기준을 고려하는 것이 좋다고 합니다.

서비스 중요도

서비스의 핵심 기능과 밀접하게 연관된 테이블을 우선적으로 고려해야 합니다.

데이터 양

데이터 양이 많은 테이블을 마이그레이션하면 시간이 오래 걸릴 수 있으므로, 이를 고려해야 합니다. 데이터 양이 적거나 데이터 변경이 자주 일어나지 않는 테이블은 마이그레이션하기 좋습니다.

테이블 간의 연관성

테이블 간의 연관 관계를 고려해야 합니다. 특히, 다른 테이블과의 관계가 복잡한 테이블은 마이그레이션 시 주의가 필요합니다.

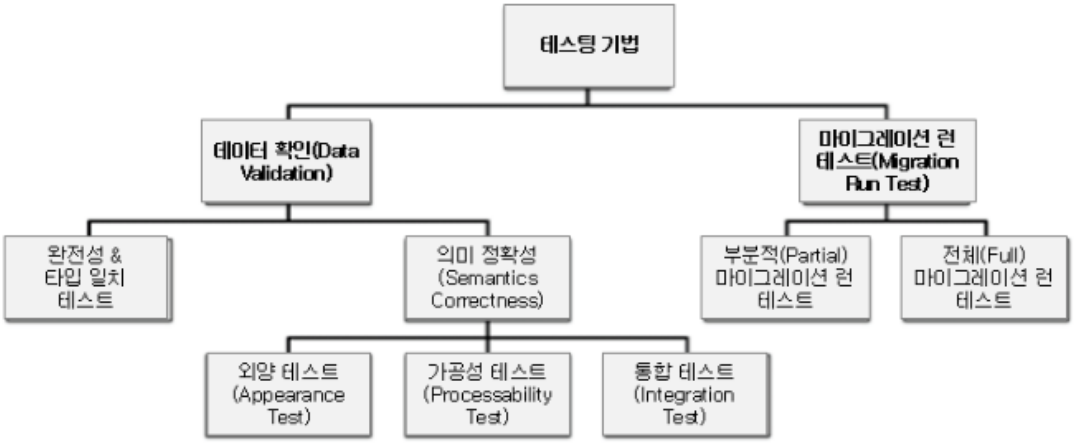
마이그레이션 복잡도

테이블의 구조가 복잡하거나 특정 데이터베이스 기능에 의존하는 경우, 마이그레이션 복잡도가 높아질 수 있습니다. 이런 테이블은 마이그레이션 시 문제가 발생할 확률이 높으므로 신중히 고려해야 합니다.

위의 내용을 바탕으로 채팅과 관련된 `ChatEntity`, `ChatRoomEntity`, `ChatJoinEntity`, `ChatLikeEntity`가 핵심 테이블로 보입니다. 따라서 4가지 테이블을 선정하고 싶습니다.

3. 마이그레이션을 검증할 수 있는 테스트

마이그레이션을 검증할 수 있는 테스트 유형은 아래 그림과 같습니다.



출처 : https://peimsam.tistory.com/231

구분	테스트 유형	설명
데이터 유효성 테스트	완전성 테스트	- 타겟 데이터베이스에서 누락된 오브젝트를 식별
	외양 테스트	- 그래픽 사용자 인터페이스(GUI) 수준에서 오브젝트의 외양에 주안점 - 테스터가 소스와 타겟 애플리케이션 GUI를 보고 수동으로 오브젝트 비교
	통합 테스트	- 마이그레이션 후에 애플리케이션 간의 연결성 (Connectivity) 및 링킹(Linking)이 제대로 작동하는지 확인 - End-to-End 테스트
	가공성 테스트	- 마이그레이션된 데이터를 처리하고, 타겟 비즈니스 애플리케이션과 새롭게 유입된 데이터 간의 조화롭고 성공적인 상호작용을 보장하는 테스트
마이그레이션 실행 테스트	전체 마이그레이션 테스트	- 전체 데이터셋을 가지고 모든 마이그레이션 프로그램을 실행하는 테스트
	부분 마이그레이션 테스트	- 적은 수의 비즈니스 오브젝트를 마이그레이션 하여 시험 마이그레이션(Trial Migration) 속도를 향상시키는 테스트

출처 : https://peimsam.tistory.com/231

4. 마이그레이션 계획

계획은 gpt를 참고해서 작성해보았습니다.

준비 단계

먼저, 마이그레이션을 위한 계획을 수립합니다. 이 계획에는 마이그레이션 대상 테이블, 마이그레이션 순서, 필요한 리소스, 예상되는 마이그레이션 시간 등이 포함되어야 합니다. 테스트 환경에서 마이그레이션을 먼저 실시하여 예상치 못한 문제를 찾아내는 것이 좋습니다.

병렬 운영

새 데이터베이스를 준비하고, 기존 데이터베이스와 새 데이터베이스를 동시에 운영합니다. 새로운 데이터는 두 데이터베이스에 모두 작성되어야 하며, 이를 위해 애플리케이션 코드를 일시적으로 수정할 수 있습니다.

데이터 이동

기존 데이터베이스의 데이터를 새 데이터베이스로 복사합니다. 이 과정 중에는 서비스를 중지하지 않습니다. 데이터의 양이 많을 경우에는 단계적으로 이동하거나, 특정 시간(트래픽이 적은 시간)에 진행하는 것이 좋습니다.

동기화 확인

모든 데이터가 새 데이터베이스로 정상적으로 이동되었는지 확인합니다. 이때, 두 데이터베이스의 데이터가 일치하는지 검증해야 합니다.

스위칭

애플리케이션의 데이터베이스 연결을 새 데이터베이스로 변경합니다. 이 작업은 가능한 짧은 시간 안에 이루어져야 하며, 이 시간 동안에는 짧은 서비스 중단이 발생할 수 있습니다.

테스트 및 모니터링

마이그레이션 후에는 서비스가 정상적으로 작동하는지 확인해야 합니다. 또한, 성능 이슈나 다른 문제가 없는지 지속적으로 모니터링해야 합니다.

참고

- http://wiki.hash.kr/index.php/마이그레이션
- https://peimsam.tistory.com/231
- https://gc.hosting.kr/blog-msa_migration-3/
- https://peimsam.tistory.com/232