**센서 빅데이터 처리 과제 2 problem 1**

**DB Design**

201921041 김대훈

1.DB Design

요구사항 분석

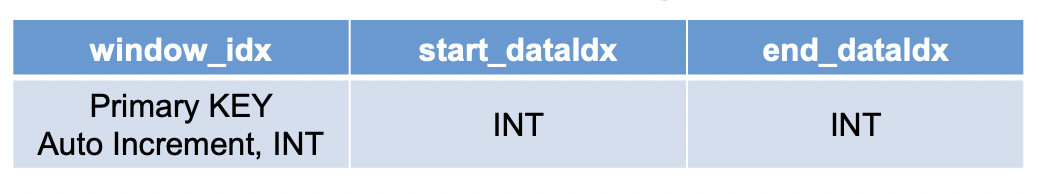
1. Sensor information을 확인 할 수 있는 entitiy 가 필요하고, accelerator 가 튜플의 형태로 존재해야 한다.

* 센서 테이블과 엑셀러레이터 테이블을 연결하여 구현한다.

1. Register information을 확인 할 수 있는 entity가 필요하고, ‘PORT4\_CHN1’,’PORT6\_CHN1’,’PORT7\_CHN1’이 튜플 형태로 존재해야 한다.

* 레지스터 테이블과 센서 테이블이 foreign key로 연결되어야한다.

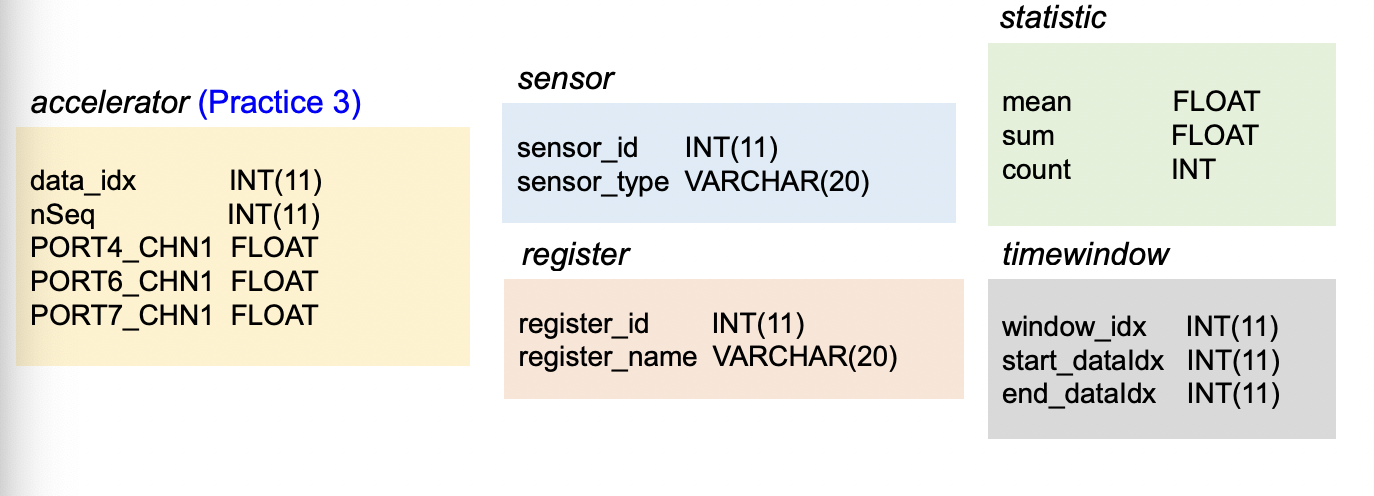
1. time window information 을 확인 할 수 있는 entity 가 존재한다.



다음과 같은 속성을 만족해야 한다.

* 이때 300개의 섹션(window\_idx 크기) 와 함께 start/end data Idx 를 추가하여 타임윈도우 테이블에 넣어야 한다.

1. Statistic information 을 알 수 있는 entity가 필요하다. Window\_idx 와 레지스터 파라미터를 활용해 AVG() SUM() COUNT() 를 저장하여 스태티스틱 테이블에 담아야 한다.



전반적인 형태는 다음과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 다이어그램은 툴을 사용에 직접 설계한 다이어 그램이다.

A query tool 을 사용해서 웹에서 그렸는데, 컴퓨터 껏다가 키면서 날라가 다음과 같은 이미지만 남게 되었다.

압축파일에 이미지 다시 파일 첨부하겠습니다.

먼저 센서 테이블 부터 확인해 보겠다. 센서 테이블은 문제의 요구사항 대로 센서 정보를 담고 있어야 되고, 해당하는 센서인 ‘accelerator’를 담고 있어야 한다. 따라서 sensor\_type 에 accelerator, sensor\_id 에 임의로 ‘1’ 을 튜플 형태로 하여 집어 넣는 방식으로 구현한다. Sensor\_type 의 경우 문자가 들어가기 때문에 varchar, sensor\_id 는 1이 들어가기 때문에 int 형을 준다. 또한 PK 인 sensor\_id 에 AI 지정 유무는 크게 상관이 없다. 다른 센서의 정보가 들어오게 된다면 AI 가 활용될 것이다. AI는 안 해줘도 상관없다.두번째로 accelerator 테이블을 살펴 보면, 먼저 실습과 동일한 부분으로 구성을 시키고, 1번 요구조건을 만족하기 위해 sensor\_id 를 추가한다. 이를 FK 로 하여 sensor 테이블의 sensor\_id를 참조하고, 여기에는 ‘1’이 들어가게 된다는 특징이 존재한다. 레지스터 테이블의 경우, 레지스터의 정보를 담고 있어야 되는데, 요구사항을 만족 시키기 위해 register\_id register\_name를 튜플화를 통해 구축한다. Register\_id 에는 1,2,3을 각각 지정하고 register\_name에는 ‘PORT4\_CHN1’ ‘PORT6\_CHN1’ ‘PORT7\_CHN1’ 를 넣어준다. 마찬가지로 id 는 1,2,3 이 들어가므로 INT형을 선언해주고 name 부분에는 문자열이 들어가니 VARCHAR 이 선언된다. 추가적으로 과제 요구사항을 만족하기 위해 센서 테이블과 레지스터 테이블을 FK로 연결해줘야 한다. Register 테이블에 sensor\_id column 을 만들고, sensor 테이블의 sensor\_id를 참조하게 해준다. 이렇게 되면 register 테이블의 sensor\_id 에는 1이 마찬가지로 들어가게 된다.

Timewindow 테이블 같은 경우는 문제에서 주어진 조건에 따라 window\_idx, start\_idx, end\_idx를 만들어준다. 따로 다른 테이블을 참조하지는 않고, start\_idx, end\_idx를 설정하는 기준이 필요한데 300개의 window\_idx와 함께, accelerator 의 data\_idx 범위 내에서 랜덤한 수를 2개 뽑아 작은 수를 start\_dataIdx, 큰 수를 end\_dataIdx로 지정해줄 것이다. 이런 방식으로 300개의 start-end dataIdx 섹션을 만든다. 마지막으로 statistic table 같은 경우, 먼저 PK 인 statistic\_idx를 만들고, INT 형으로 선언한다. AI를 통해 자동으로 인덱스 값이 1씩 증가하게 한다. Reg\_id를 통해 레지스터를 분류할 것이고 이를 위해 레지스터 테이블의 register\_id를 참조한다.(FK) 또한 각 레지스터 별로 타임 윈도우 테이블에서 랜덤으로 파싱한 start-end 인덱스 구간 300개를 참조해 각각 윈도우들의 구간 만큼의 AVG() SUM() COUNT() 값을 구하게 한다. 따라서 window\_idx column을 만들고, timewindow 테이블의 window\_idx를 참조하게 한다.(FK) 그렇게 되면 statistic table은 레지스터 3개와 윈도우의 300개의 섹션이 곱해져 총 900개의 인덱스가 생성된다. Mean column 같은 경우는 레지스터에 대한 윈도우 구간의 평균값이며 소수점이 나올 수 있으니 FLOAT 형태로 선언한다. Sum column같은 경우는 레지스터에 대한 윈도우 구간 값들의 합이며 마찬가지로 소수점이 나올 수 있으니 FLOAT으로 선언한다. COUNT는 레지스터에 대한 윈도우 구간 수를 저장하는 column 이고, 소수점이 나올 일이 없으니 INT 형으로 한다. 강의 시간에 언급되었지만, statistic table 에서 이미 register table을 참조 하고 있고, register table 은 sensor table을 참조하고 있으니 굳이 statistic table 에서 sensor table을 또 참조할 필요가 없어 sensor\_id 를 따로 attribute로 추가해 참조하지 않았다.

데이터 베이스 구축 결과

텍스트, 전자기기, 스크린샷, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

* Accelerator table

텍스트, 전자기기, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Register table

텍스트, 스크린샷, 전자기기이(가) 표시된 사진

자동 생성된 설명

* Sensor table

텍스트, 전자기기, 컴퓨터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Statistic table

텍스트, 전자기기, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Timewindow table