Oracle SQL을 사용한 데이터베이스 구현

야구 통계 DB

김영범

INDEX

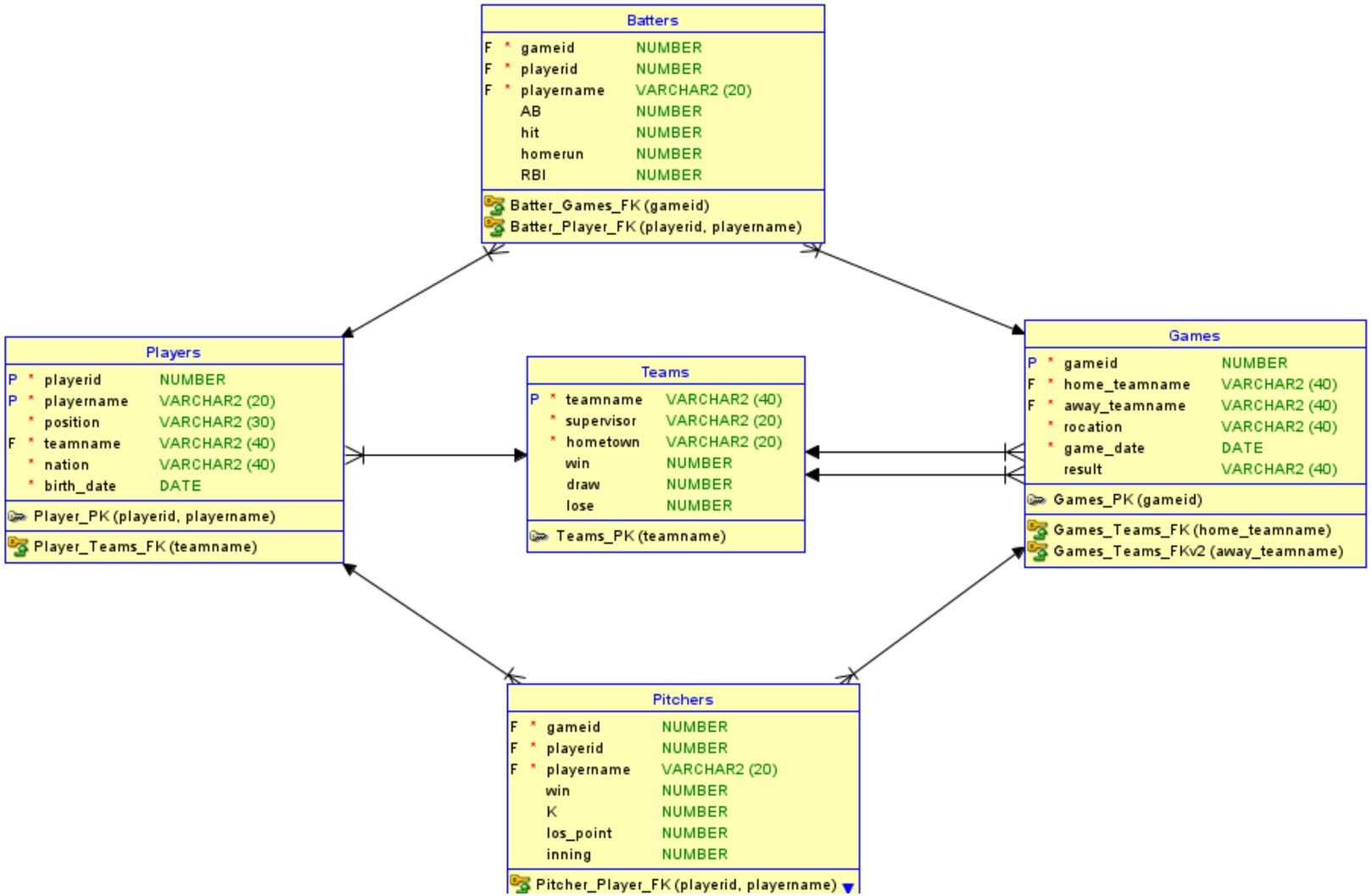
기능 설명

기능

1. 야구 팀과 야구 선수 및 경기 정보를 제공하는 테이블 구현
2. 야구 팀은 팀 정보와 경기의 결과에 따른 통계를 가진다.
3. 경기는 경기 일정 및 정보 경기의 결과를 제공한다.
4. 선수들의 생년월일, 소속 팀 등 간단한 정보를 제공하는 테이블을 구현
5. 야구 선수는 타자 및 투수로 구분하여 경기 마다의 성적 정보를 가진다.

데이터 모델링

데이터 모델링


Batters
F * gameid NUMBER
F * playerid NUMBER
F * playername VARCHAR2 (20)
AB NUMBER
hit NUMBER
homerun NUMBER
RBI NUMBER
Batter_Games_FK (gameid)
Batter_Player_FK (playerid, playername)
Players
P * playerid NUMBER
P * playername VARCHAR2 (20)
* position VARCHAR2 (30)
F * teamname VARCHAR2 (40)
* nation VARCHAR2 (40)
* birth_date DATE
Player_PK (playerid, playername)
Player_Teams_FK (teamname)
Teams
P * teamname VARCHAR2 (40)
* supervisor VARCHAR2 (20)
* hometown VARCHAR2 (20)
win NUMBER
draw NUMBER
lose NUMBER
Teams_PK (teamname)
Games
P * gameid NUMBER
F * home_teamname VARCHAR2 (40)
F * away_teamname VARCHAR2 (40)
* rocation VARCHAR2 (40)
* game_date DATE
result VARCHAR2 (40)
Games_PK (gameid)
Games_Teams_FK (home_teamname)
Games_Teams_FKv2 (away_teamname)
Pitchers
F * gameid NUMBER
F * playerid NUMBER
F * playername VARCHAR2 (20)
win NUMBER
K NUMBER
los_point NUMBER
inning NUMBER
Pitcher_Player_FK (playerid, playername)

설명

1. 팀(Teams) 테이블: 팀 테이블은 팀명으로 구분하고 감독, 연고지, 승, 무, 패의 정보를 가진다.

2. 경기(Games) 테이블: 경기 테이블은 경기id로 구별하고 홈팀, 원정팀, 장소, 경기날짜, 승패결과의 정보를 가진다.

3. 선수(Players) 테이블: 선수 테이블은 선수id로 구별하고 이름, 포지션, 소속 팀, 국적, 생년월일의 정보를 가진다.

4. 타자(Batters) 테이블: 타자 테이블은 경기별 타석(AB), 안타(hit), 홈런(homerun), 타점(RBI)의 정보를 가진다.

5. 투수(Pitchers) 테이블: 투수 테이블은 경기별 삼진(K), 실점(kos_point), 소화한 이닝(inning)의 정보를 가진다.

제약 조건: 대부분의 데이터는 not null의 제약을 가지지만, 경기를 해야 데이터가 나오는 팀 테이블의 승, 무, 패와 타자 테이블의 타석, 안타, 홈런, 타점과 투수 테이블의 승, 삼진, 실점, 소화 이닝은 default를 0으로 설정한다.
또, 경기 결과도 default는 '경기 예정'으로 설정한다.

DDL

DDL

아래와 같이 5개의 테이블 생성

```
-- tabla 생성
-- teams
CREATE TABLE teams (
    teamname   VARCHAR2(40) NOT NULL,
    supervisor VARCHAR2(20) NOT NULL,
    hometown   VARCHAR2(20) NOT NULL,
    win        NUMBER DEFAULT 0,
    draw       NUMBER DEFAULT 0,
    lose       NUMBER DEFAULT 0
);

ALTER TABLE teams ADD CONSTRAINT teams_pk PRIMARY KEY ( teamname );
-- games
CREATE TABLE games (
    gameid        NUMBER NOT NULL,
    home_teamname VARCHAR2(40) NOT NULL,
    away_teamname VARCHAR2(40) NOT NULL,
    rocation      VARCHAR2(40)
        CONSTRAINT notnull NOT NULL,
    game_date     DATE
        CONSTRAINT "not null" NOT NULL,
    result        VARCHAR2(40) DEFAULT '경기 예정'
);

ALTER TABLE games ADD CONSTRAINT games_pk PRIMARY KEY ( gameid );

ALTER TABLE games
    ADD CONSTRAINT games_teams_fk FOREIGN KEY ( home_teamname )
        REFERENCES teams ( teamname );

ALTER TABLE games
    ADD CONSTRAINT games_teams_fkv2 FOREIGN KEY ( away_teamname )
        REFERENCES teams ( teamname );
```

```
-- players
CREATE TABLE players (
    playerid   NUMBER NOT NULL,
    playername VARCHAR2(20) NOT NULL,
    position   VARCHAR2(30) NOT NULL,
    teamname   VARCHAR2(40) NOT NULL,
    nation     VARCHAR2(40) NOT NULL,
    birth_date DATE NOT NULL
);

ALTER TABLE players ADD CONSTRAINT player_pk PRIMARY KEY ( playerid,
                                                           playername );
ALTER TABLE players
    ADD CONSTRAINT player_teams_fk FOREIGN KEY ( teamname )
        REFERENCES teams ( teamname );
-- batters
CREATE TABLE batters (
    gameid     NUMBER NOT NULL,
    playerid   NUMBER NOT NULL,
    playername VARCHAR2(20) NOT NULL,
    ab         NUMBER DEFAULT 0,
    hit        NUMBER DEFAULT 0,
    homerun    NUMBER DEFAULT 0,
    rbi        NUMBER DEFAULT 0
);

ALTER TABLE batters
    ADD CONSTRAINT batter_games_fk FOREIGN KEY ( gameid )
        REFERENCES games ( gameid );

ALTER TABLE batters
    ADD CONSTRAINT batter_player_fk
        FOREIGN KEY ( playerid,
                      playername )
            REFERENCES players ( playerid,
                                 playername );
```

```
-- pitchers
CREATE TABLE pitchers (
    gameid     NUMBER NOT NULL,
    playerid   NUMBER NOT NULL,
    playername VARCHAR2(20) NOT NULL,
    win        NUMBER DEFAULT 0,
    inning     NUMBER DEFAULT 0,
    los_point  NUMBER DEFAULT 0,
    k          NUMBER DEFAULT 0
);

ALTER TABLE pitchers
    ADD CONSTRAINT pitcher_games_fk FOREIGN KEY ( gameid )
        REFERENCES games ( gameid );

ALTER TABLE pitchers
    ADD CONSTRAINT pitcher_player_fk
        FOREIGN KEY ( playerid,
                      playername )
            REFERENCES players ( playerid,
                                 playername );
```

DDL

생성한 테이블에 아래와 같은 양식으로 values 삽입

```
-- 데이터 삽입
-- teams
insert into teams (teamname, supervisor, hometown) values('롯데 자이언츠', '김태형', '부산');
-- games
insert into games values(4, '롯데 자이언츠', '한화 이글스', '사직 야구장', '2024/04/02', '롯데 승');
-- players
insert into players values(26, '윤동희', 'CF', '롯데 자이언츠', '대한민국', '2003/09/18');
-- batters
insert into batters values(23, 26, '윤동희', 6, 2, 1, 2);
-- pitchers
insert into pitchers values (9, 28, '윌커슨', 1, 9, 0, 10);
-- 경기를 뛰지 않은 선수
insert into batters (gameid, playerid, playername) values(37, 6, '강민호');
```

위와 같은 형태로 5개의 테이블에 해당하는 데이터를 삽입(총합 대략 350개)

DML

DML

기본적인 select부터 update, 조인, 부속질의 , 그룹함수, 순위함수 등 지금까지 공부했던 내용들을 내가 구현한 DB에 맞춰 사용

Update

```
-- update
-- 승
update teams set win = (select count(*) from games where result = '롯데 승') where teamname = '롯데 자이언츠';
-- 무
update teams set draw =
(select count(*) from games where result = '무승부' and
(home_teamname = '롯데 자이언츠' or away_teamname = '롯데 자이언츠'))
where teamname = '롯데 자이언츠';
-- 패
update teams set lose =
(select count(*) from games where (result != '롯데 승' and result != '무승부'
and result != '경기 예정' and result != '우천 취소')
and (home_teamname = '롯데 자이언츠' or away_teamname = '롯데 자이언츠'))
where teamname = '롯데 자이언츠';
```

Teams 테이블 update에 Games 테이블의 경기 결과(result)를 사용하여 update

DML

select(basic)

```
-- basic
-- between
-- 90~99년생 선수들 출력 (생년월일 빠른 순 정렬)
select * from players where birth_date between '90/01/01' and '99/12/31' order by birth_date;
-- date -> char로 변환
-- 생일이 화요일인 선수들 출력
select playername, TO_CHAR(birth_date, 'DAY') from players
where TO_CHAR(birth_date, 'DAY') = '화요일';
-- 8월에 생일인 선수들 출력
select playername, TO_CHAR(birth_date, 'MM') as month from players
where TO_CHAR(birth_date, 'MM') = '08';
-- 생일이 같은 사람들 출력
select p1.playername, p2.playername, TO_CHAR(birth_date, 'MM/DD') as birthday
from players p1, (select playername, TO_CHAR(birth_date, 'MM/DD') as birthday  from players) p2
where TO_CHAR(birth_date, 'MM/DD') = p2.birthday and p1.playername != p2.playername;
-- 기본 연산
-- 각 팀들의 승률(승 / (승 + 무 + 패))출력:
select teamname, (win / (win + draw + lose)) as "win_rate" from teams;
-- 정렬
-- 위의 sql을 승률 순으로 정렬
select teamname, (win / (win + draw + lose)) as "win_rate" from teams order by "win_rate" desc;
-- 최대, 최소(max, min)
-- 위의 sql에서 최대 최소 승률 출력
select max((win / (win + draw + lose))) "최고 승률",
min((win / (win + draw + lose))) "최저 승률" from teams;
```

```
-- 문자열
-- 문자열 연결
-- 홈 팀 및 경기장 이름
select home_teamname, rocation from games;
-- games의 rocation은 home_team의 구장 이름이다 이를 이용하여 문자열을 연결하면
select distinct home_teamname || '의 구장은 ' || rocation as stadium from games;
-- like 문자열
-- 경기장이 '파크'로 끝나는 구장을 가진 팀이름 출력
select distinct home_teamname, rocation from games where rocation like '%파크';
-- 이름이 두글자인 선수들 출력
select playername from players where playername like '__';
-- '스'로 끝나는 팀 이름들 출력
select teamname from teams where teamname like '%스';
```

기초적인 내용(between, 문자열, 연산 등..)부터

DML

순위 함수

```
-- 순위 함수
-- 야구는 승패마진(승-패)로 순위를 결정한다. 이에따라 현재 순위 출력
select teamname, win, draw, lose, (win - lose) as gains, rank() over(order by (win - lose) desc) as rank from t
-- 위를 이용해 1위 팀의 연고지를 출력
select rownum as rank, teamname, hometown
from (select teamname, rank() over(order by (win - lose) desc) as rank, hometown from teams order by rank)
where rownum = 1;
-- 또 위를 이용하여 1위 팀의 선수를 출력(부속질의와 palyers의 조인)
select p.teamname, p.playername, p.position from players p,
(select rownum as rank, teamname, hometown
from (select teamname, rank() over(order by (win - lose) desc) as rank, hometown from teams order by rank)
where rownum = 1) t
where p.teamname = t.teamname;
-- 문자열에서 이용한 문장을 순위함수와 함께 이용하여 스로 끝나는 팀들의 순위 출력
select t.teamname, r.rank from
(select teamname,(win - lose) as gains, rank() over(order by (win - lose) desc) as rank from teams) r,
(select teamname from teams where teamname like '%스') t
where t.teamname = r.teamname;
```

	TEAMNAME	WIN	DRAW	LOSE	GAINS	RANK
1	롯데 자이언츠	6	1	1	5	1
2	삼성 라이온즈	6	0	2	4	2
3	기아 타이거즈	5	1	2	3	3
4	LG 트윈스	5	0	3	2	4
5	NC 다이노스	3	1	4	-1	5
6	한화 이글스	3	2	4	-1	5
7	키움 히어로즈	3	1	5	-2	7
8	KT 위즈	3	0	5	-2	7
9	두산 베어스	1	2	5	-4	9
10	SSG 랜더스	2	0	6	-4	9

	RANK	TEAMNAME	HOMETOWN
1	1	롯데 자이언츠	부산

	TEAMNAME	PLAYERNAME	POSITION
1	롯데 자이언츠	레이예스	RF
2	롯데 자이언츠	윤동희	CF
3	롯데 자이언츠	반즈	SP
4	롯데 자이언츠	윌커슨	SP

	TEAMNAME	RANK
1	LG 트윈스	4
2	NC 다이노스	5
3	한화 이글스	5
4	두산 베어스	9
5	SSG 랜더스	9

야구 팀의 순위를 순위 함수를 사용하여 출력(앞에서 했던 연산이나 문자열 등을 이용)

DML

조인을 사용하여 경기 일정 파악이나 투수, 타자들의 기록 등을 확인할 수 있고, 이를 부속질의와 함께 이용하면 상대전적, 경기에 나올 선수들의 성적 등 다양한 결과를 얻을 수 있다.

조인

```
-- 조인
-- 문자열에서 했던 취소되었거나 or 치뤄지지 않은 경기 확인
select * from games where result like '%취소%' or result like '%예정%';
-- 위를 이용하여 players와 조인하여 치뤄지지 않은 경기에 출전할 선수들 출력
select p.teamname, p.playername, p.position from players p, teams t,
(select * from games where result like '%예정%') g
where (g.home_teamname = t.teamname or g.away_teamname = t.teamname)
and t.teamname = p.teamname;

-- gameid를 이용해 투수 '반즈'선수가 경기를 했던 경기에 타자들의 기록을 출력
select p.playername, b.playername, b.ab, b.hit, b.homerun, b.rbi
from batters b, pitchers p
where p.gameid = b.gameid and p.playername = '반즈';
-- 위의 sql문에서 롯데 자이언츠 타자들의 gmaeid를 제외하면 상대팀의 타자들의 기록
-- 즉, 상대기록을 볼 수 있다.
select p.playername, b.playername, b.ab, b.hit, b.homerun, b.rbi
from batters b, pitchers p
where p.gameid = b.gameid and p.playername = '반즈'
and b.playerid not in (select playerid from players where teamname = '롯데 자이언츠');
```

```
-- 사직 야구장에서 경기 예정인 선수(타자)들의 성적 출력
select b.playername, sum(ab) "타수", sum(hit) "안타", sum(homerun) "홈런", sum(rbi)"타점"
from batters b,
(select p.teamname, p.playername, p.position from players p, teams t,
(select * from games where result like '%예정%' and rocation = '사직 야구장') g
where (g.home_teamname = t.teamname or g.away_teamname = t.teamname)
and t.teamname = p.teamname) ep
where ep.playername = b.playername
group by b.playerid, b.playername;
```

1	기아 타이거즈	김도영	3B
2	기아 타이거즈	최형우	DH
3	기아 타이거즈	양현종	SP
4	기아 타이거즈	네일	SP
5	LG 트윈스	오스틴	RF
6	LG 트윈스	홍창기	RF
7	LG 트윈스	임찬규	SP
8	LG 트윈스	에르난데스	SP
9	삼성 라이온즈	구자욱	RF
10	삼성 라이온즈	강민호	C
11	삼성 라이온즈	레예스	SP
12	삼성 라이온즈	원태인	SP
13	두산 베어스	양의지	C
14	두산 베어스	정수빈	CF
15	두산 베어스	곽빈	SP
16	두산 베어스	최원준	SP

	PLAYERNAME	PLAYERNAME_1	AB	HIT	HOMERUN	RBI
1	반즈	강민호	5	1	0	0
2	반즈	김도영	3	1	1	1
3	반즈	노시환	4	1	0	0
4	반즈	강백호	5	1	0	1
5	반즈	최형우	3	0	0	0
6	반즈	로하스	4	1	1	1
7	반즈	구자욱	3	1	0	1

	PLAYERNAME	타수	안타	홈런	타점
1	강백호	36	8	0	6
2	레이예스	40	15	0	11
3	윤동희	37	11	2	6
4	로하스	37	14	3	11

DML

그룹함수

그룹 함수를 사용하여 선수들의 성적을 집계할 수 있고 이를 활용하여 타격 1위, 홈런 1위나 평균 이상의 선수들을 출력하는 등의 결과를 얻을 수 있다.

```
-- 그룹 함수
-- batters와 pitchers 테이블을 만든 이유는 각 경기마다 선수의 성적을 집계하기 위함이다.
-- 이를 이용하여 각 선수의 기록들을 출력
-- 평균보다 많은 이닝을 던진 투수 출력
select playerid, playername, avg(inning) from pitchers p
group by playerid, playername
having avg(inning) > (select avg(inning) from pitchers) order by avg(inning) desc;

-- 타자들의 기록들의 합 출력
select playername, sum(ab), sum(hit), sum(homerun), sum(rbi) from batters group by playerid, playername;
-- 타자의 타율은 타석당 안타개수(hit/ab)이다. 이를 이용하여 타율 순으로 정렬하여 출력
select playername, (sum(hit)/sum(ab)) as "batting_average"
from batters group by playerid, playername order by "batting_average" desc;
-- 문자열 표현
-- 위의 sql 문장의 batting_average는 소수점 표현이 너무 길다. TO_CHAR를 통해 소수점 3자리까지만 출력하자
select playername, TO_CHAR((sum(hit)/sum(ab)), '0.999') as "batting_average"
from batters group by playerid, playername order by "batting_average" desc;
```

	PLAYERID	PLAYERNAME	AVG(INNING)
1	35	하트	7.5
2	27	반즈	7.25
3	28	윌커슨	7
4	7	레예스	7
5	4	네일	6.75
6	24	앤더슨	6.75
7	15	곽빈	6.2

	PLAYERNAME	SUM(AB)	SUM(HIT)	SUM(HOMERUN)	SUM(RBI)
1	강백호	36	8	0	6
2	김혜성	47	17	2	10
3	강민호	24	6	1	2
4	홍창기	37	14	0	1
5	에레디아	37	16	0	7
6	레이예스	40	15	0	11
7	박민우	39	12	0	4
8	오스틴	39	11	2	21
9	윤동희	37	11	2	6
10	송성문	29	9	1	7
11	김도영	36	12	8	13
12	최형우	38	11	1	4
13	로하스	37	14	3	11
14	노시환	23	6	1	3
15	양의지	37	9	5	12
16	최정	36	9	5	6
17	구자욱	39	14	4	16
18	정수빈	37	10	1	4
19	데이비슨	41	11	10	16

	PLAYERNAME	batting_average
1	에레디아	0.432
2	로하스	0.378
3	홍창기	0.378
4	레이예스	0.375
5	김혜성	0.362
6	구자욱	0.359
7	김도영	0.333
8	송성문	0.310
9	박민우	0.308
10	윤동희	0.297
11	최형우	0.289
12	오스틴	0.282
13	정수빈	0.270
14	데이비슨	0.268
15	노시환	0.261
16	최정	0.250
17	강민호	0.250
18	양의지	0.243
19	강백호	0.222

DML

그룹함수2

그룹 함수를 사용하여 선수들의 성적을 집계할 수 있고 이를 활용하여 타격 1위, 홈런 1위나 평균 이상의 선수들을 출력하는 등의 결과를 얻을 수 있다.

```
-- 위의 함수들을 이용하여 타자들의 타율, 안타, 홈런, 타점의 각 1위 선수들을 출력
-- 타율 왕
select playername as "타율 왕", TO_CHAR((sum(hit)/sum(ab)), '0.999') as "타율"
from batters group by playerid, playername having TO_CHAR((sum(hit)/sum(ab)), '0.999') =
(select max(TO_CHAR((sum(hit)/sum(ab)), '0.999')) from batters group by playerid, playername);
-- 타격 왕
select playername as "타격 왕", sum(hit) as "안타"
from batters group by playerid, playername having sum(hit) =
(select max(sum(hit)) from batters group by playerid, playername);
-- 홈런 왕
select playername as "홈런 왕", sum(homerun) as "홈런"
from batters group by playerid, playername having sum(homerun) =
(select max(sum(homerun)) from batters group by playerid, playername);
-- 타점 왕
select playername as "타점 왕", sum(rbi) as "타점"
from batters group by playerid, playername having sum(rbi) =
(select max(sum(rbi)) from batters group by playerid, playername);

-- 위를 이용하여 players와 조인하면 소속팀까지 출력가능
-- ex) 타점 왕
select  p.playername "타점 왕", p.teamname "소속 팀", b.rbi "타점"
from players p,
(select playername, sum(rbi) as rbi from batters group by playerid, playername having sum(rbi) =
(select max(sum(rbi)) from batters group by playerid, playername)) b
where p.playername = b.playername;

-- 투수도 위의 예와 같이 삼진 왕을 출력
select  p.playername "삼진 왕", p.teamname "소속 팀", pc.k "삼진"
from players p,
(select playername, sum(k) as k from pitchers group by playerid, playername having sum(k) =
(select max(sum(k)) from pitchers group by playerid, playername)) pc
where p.playername = pc.playername;
```

각 부문 별 1위 선수 출력

	타점 왕	타점
1	오스틴	21

	홈런 왕	홈런
1	데이비슨	10

	타격 왕	안타
1	김혜성	17

	타율 왕	타율
1	에레디아	0.432

	삼진 왕	소속 팀	삼진
1	반즈	롯데 자이언츠	45

DML

앞에서 했던 내용들을 응용하여 야구에 존재하는 '규정 이닝', '규정 타석' 과 나의 임의대로 설정한 '골든 글러브'를 구현

응용

골든 글러브

```
-- 골든글러브
-- 야구에서는 매년 각 포지션에서 가장 뛰어난 활약을 보인 선수에게 골든글러브라는 상을 수여한다.
-- 안타와 타점은 1점, 홈런은 2점의 가산점을 부여한다고 가정하고 각 포지션 별 골든글러브 출력
-- 3루수 골든글러브 수상자 출력
select p.playername as "수상자", p.position, b.point from players p,
(select playerid, playername, (sum(hit) * 1 + sum(homerun) * 2 + sum(rbi) * 1) as point
from batters group by playername, playerid order by point desc ) b
where p.playerid = b.playerid and p.position = '3B' and rownum = 1;
-- 투수 골든 글러브 삼진 + (실점/이닝 * -1)
select p.playername as "수상자", p.position, pc.point from players p,
(select playerid, playername, floor(sum(k) * 1 + (sum(los_point)/(sum(inning) * (-1)))) as point
from pitchers group by playername, playerid order by point desc ) pc
where p.playerid = pc.playerid and p.position = 'SP' and rownum = 1;
```

	수상자	POSITION	POINT
1	김도영	3B	41

	수상자	POSITION	POINT
1	반즈	SP	44

DML

앞에서 했던 내용들을 응용하여 야구에 존재하는 '규정 이닝', '규정 타석' 과 나의 임의대로 설정한 '골든 글러브'를 구현

응용

규정 이닝, 규정 타석

```
-- 야구에는 '규정 이닝', '규정 타석'이 존재한다.
-- 규정 이닝(타석)이란 정식 기록으로 인정할 수 있는 이닝(타석) 수이고,
-- 이는 진행한 경기수에 따라 정해진다.
-- 규정 이닝은 소속된 팀이 진행한 경기 수 만큼 이닝을 소화해야 하고
-- 규정 타석은 소속된 팀이 진행한 경기 수 * 3.1에서 소수점을 뺀 수 만큼 타석을 진행해야 한다.
```

```
-- 한화 이글스팀 소속 선수의 규정타석 출력
select floor(count(*) * 3.1) as "규정 타석" from games
where (home_teamname = '한화 이글스' or away_teamname = '한화 이글스')
and (result != '우천 취소' and result != '경기 예정');
-- 위를 이용하여 한화 이글스팀 소속의 선수 중 규정 타석을 채우지 못한 선수(타자) 출력
select playername, ab as "타석 수",
(select floor(count(*) * 3.1) as "규정 타석" from games
where (home_teamname = '한화 이글스' or away_teamname = '한화 이글스')
and (result != '우천 취소' and result != '경기 예정')) as "규정 타석"
from
(select playerid from players where teamname = '한화 이글스') p,
(select playerid, playername, sum(ab) ab from batters group by playerid, playername
having sum(ab) < (select floor(count(*) * 3.1) as "규정 타석" from games
where (home_teamname = '한화 이글스' or away_teamname = '한화 이글스')
and (result != '우천 취소' and result != '경기 예정'))) b
where p.playerid = b.playerid;
```

	PLAYERNAME	타석 수	규정 타석
1	노시환	23	27

```
-- 한화 이글스팀 소속의 선수 중 규정 이닝을 채운 선수(투수) 출력
select playername, inning as "이닝 수",
(select count(*) as "규정 이닝" from games
where (home_teamname = '한화 이글스' or away_teamname = '한화 이글스')
and (result != '우천 취소' and result != '경기 예정')) as "규정 이닝"
from
(select playerid from players where teamname = '한화 이글스') p,
(select playerid, playername, sum(inning) inning from pitchers group by playerid, playername
having sum(inning) >= (select count(*) as "규정 이닝" from games
where (home_teamname = '한화 이글스' or away_teamname = '한화 이글스')
and (result != '우천 취소' and result != '경기 예정'))) pc
where p.playerid = pc.playerid;
```

	PLAYERNAME	이닝 수	규정 이닝
1	문동주	18	9
2	류현진	29	9

DML

delete

Plyaers에서 데이터가 삭제되면 해당하는 Batters나 Pitchers에서 그에 따라 함께 삭제되어야 한다.

```
-- alter
-- players에서 데이터가 사라지면 batters와 pitchers의 데이터도 함께 사라지도록 설정
-- 기존 제약조건 삭제
ALTER TABLE batters
    drop CONSTRAINT batter_player_fk;
ALTER TABLE pitchers
    drop CONSTRAINT pitcher_player_fk;
-- 제약조건 새로 추가
ALTER TABLE batters
    ADD CONSTRAINT batter_player_fk
        FOREIGN KEY ( playerid,
                      playername )
            REFERENCES players ( playerid,
                                 playername ) on delete cascade;

ALTER TABLE pitchers
    ADD CONSTRAINT pitcher_player_fk
        FOREIGN KEY ( playerid,
                      playername )
            REFERENCES players ( playerid,
                                 playername ) on delete cascade;
-- delete
-- players와 함께 batters도 29번(페라자) 삭제
delete from players where playerid = 29;
-- players와 함께 pitchers도 39번(후라도) 삭제
delete from players where playerid = 39;

select * from players;
select * from batters;
select * from pitchers;
```

Batters의 29번 데이터의 정보도 삭제 되었다.

42	27	7	레예스	1	9	0	10
43	27	27	반즈	0	8	1	14
44	28	4	네일	1	6	3	7
45	28	32	문동주	0	4	6	2
46	29	40	헤이수스	0	6	1	7
47	29	16	최원준	1	7	1	8
48	30	19	고영표	0	6	1	5

Pitchers의 39번 데이터의 정보도 삭제 되었다.

42	27	7	레예스	1	9	0	10
43	27	27	반즈	0	8	1	14
44	28	4	네일	1	6	3	7
45	28	32	문동주	0	4	6	2
46	29	40	헤이수스	0	6	1	7
47	29	16	최원준	1	7	1	8
48	30	19	고영표	0	6	1	5

View

View

Players에서 nation을 이용하여 외국인 선수들로만 구성된 View를 생성 또, 생성한 View를 이용하여 새로운 View도 생성

```
-- view
-- players에서 우리나라 국적이 아닌 선수들을 뽑아 외국인선수(foreign_players)라는 새로운 view를 생성
create view foreign_players as select * from players where nation != '대한민국';
select * from foreign_players;

-- 생성한 view를 조인해 외국인 타자들만 출력
select distinct f.playername, f.nation from batters b, foreign_players f
where b.playerid = f.playerid;

-- 외국인 타자들 중 홈런 개수 순으로 출력
select f.playername, b.homerun
from (select playerid, sum(homerun) as homerun from batters group by playerid) b,
foreign_players f
where b.playerid = f.playerid order by b.homerun desc;

-- 위의 sql문과 위에서 했던 순위 함수를 사용하여 가장 많은 홈런을 친 외국인 선수를 출력
select playername as "외국인 타자 중 최다 홈런", homerun from (select f.playername as playername, b.homerun as homerun
from (select playerid, sum(homerun) as homerun from batters group by playerid) b,
foreign_players f
where b.playerid = f.playerid order by b.homerun desc)
where rownum = 1;
```

	PLAYERID	PLAYERNAME	POSITION	TEAMNAME	NATION	BIRTH_DATE
1	4	네일	SP	기아 타이거즈	미국	93/02/08
2	7	레예스	SP	삼성 라이온즈	도미니카 공화국	96/11/02
3	9	오스틴	RF	LG 트윈스	미국	93/10/14
4	12	에르난데스	SP	LG 트윈스	베네수엘라	95/05/03
5	18	로하스	RF	KT 위즈	미국	90/05/24
6	21	에레디아	LF	SSG 랜더스	쿠바	91/01/31
7	25	레이예스	RF	롯데 자이언츠	베네수엘라	94/10/05
8	27	반즈	SP	롯데 자이언츠	미국	95/10/01
9	28	윌커슨	SP	롯데 자이언츠	미국	89/05/24
10	34	데이비슨	1B	NC 다이노스	미국	91/03/26
11	35	하트	SP	NC 다이노스	미국	92/11/23

	PLAYERNAME	NATION
1	에레디아	쿠바
2	로하스	미국
3	오스틴	미국
4	레이예스	베네수엘라
5	데이비슨	미국

	외국인 타자 중 최다 홈런	HOMERUN
1	데이비슨	10

```
-- 국적(nation)별로 몇명의 선수가 있는지 출력
select nation, count(*) from foreign_players group by nation;
-- 위의 sql문과 foreign_players를 활용해
-- 가장 많은 인원을 가진 나라의 선수들의 정보를 가진 view를 생성(view를 사용한 새로운 view)
create view most_numbers_of_foreign (playerid, playername, position, teamname, nation, birth_date)
as
(select f.playerid, f.playername, f.position, f.teamname, f.nation, f.birth_date
from foreign_players f,
(select nation, count(*) from foreign_players group by nation
having count(*) = (select max(count(*)) from foreign_players group by nation)) c
where c.nation = f.nation);

select * from most_numbers_of_foreign;
```

	PLAYERID	PLAYERNAME	POSITION	TEAMNAME	NATION	BIRTH_DATE
1	4	네일	SP	기아 타이거즈	미국	93/02/08
2	9	오스틴	RF	LG 트윈스	미국	93/10/14
3	18	로하스	RF	KT 위즈	미국	90/05/24
4	27	반즈	SP	롯데 자이언츠	미국	95/10/01
5	28	윌커슨	SP	롯데 자이언츠	미국	89/05/24
6	34	데이비슨	1B	NC 다이노스	미국	91/03/26
7	35	하트	SP	NC 다이노스	미국	92/11/23

후기

후기 및 통합SQL문

후기

책에 나오는 내용이나 예제를 따라하는 것 보다 이렇게 내가 좋아하는 분야에 직접 데이터를 만들어 DB를 구현하니 집중도 잘 되었고 공부했던 내용들이 머리에 더 잘 저장된 것 같습니다. DB를 구현하며 기본적인 DDL이나 DML의 처리 능력이 올랐고 특히, 부속 질의나 그룹 함수를 주로 사용하며 해당 로직에 대한 이해도가 많이 올랐고, 위의 두 SQL문을 사용할 때 다양하게 발생한 문제들을 직접 해결해보고, 습득하면서 능동적인 학습 성취를 이룰 수 있었습니다. 이 DB를 데이터베이스에 대한 재미를 느꼈고, 구현하면서 다음에는 훨씬 더 많은 양의 데이터도 처리해보고 싶다는 생각이 들었습니다.
이 DB에서도 간단하게 구현하였기 때문에 야구에 대한 세부적인 통계까지는 다루지 못하였고, 예매나 팬 홈페이지 등 다른 기능들도 추가하여 작성해보고 싶습니다.

통합 SQL문은 아래 github 링크를 작성하였으니, 참조해주시길 바랍니다.

DB구현 깃허브 링크
https://github.com/bbang-beom/baseballstats