Normalization 내용 :

		User	Follow	Tweet	Comment	BlackList
	1 . Primary Key인가	O	O	O	O	O
	2. repeat groups이 아닌가	O	O	O	O	O
1NF	3. Atomic columms인가	O	O	✓	✓	O
2NF	4. 부분종속성 (partial dependency) 없는가	O	O	✓	✓	O
3NF	5. No transitive dependencies (values depend only on Candidate keys)	O	O	✓	✓	O
	6. Every non-trivial functional dependency involves either a superkey or an elementary key’s subkey	O	O	✓	✓	O
BCNF	7. no redundancy from any functional dependency	O	O	✓	✓	O

Comment정규화 과정 :

[초기 테이블 구조]

```
CREATE TABLE Comment (
  CommentID varchar(20),
  TweetID varchar(20),
  UserID varchar(20),
  Content varchar(50),
  Timestamp varchar(25),
  PRIMARY KEY (CommentID),
  FOREIGN KEY (TweetID) REFERENCES Tweet(TweetID),
  FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

문제점 : UserID를 직접 참조하고 있으며, TweetID도 외래키로 참조하고 있음
즉, 중복된 데이터 가능성 및 TweetID없는 Comment있을 수 있음

```
-- 중복된 데이터
insert into InitialComment values('1', '1', '202235040', 'Content1', '2023.11.21.10:40');
insert into InitialComment values('2', '1', '202235040', 'Content2', '2023.11.21.10:41');

-- 이상점: TweetID가 없는 Comment
insert into InitialComment values('3', null, '202235041', 'Content3', '2023.11.21.10:42');
```

[문제해결]
-- Comment 테이블
```
CREATE TABLE Comment (
  CommentID varchar(20) PRIMARY KEY,
  TweetID varchar(20),
  UserID varchar(20),
  Content varchar(50),
  Timestamp varchar(25),
  FOREIGN KEY (TweetID) REFERENCES Tweet(TweetID),
  FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

Tweet 정규화 과정 :

[초기 테이블 구조]
```
CREATE TABLE Tweet (
  TweetID varchar(20),
  WriterID varchar(20),
  WriterName varchar(50),
  WriterEmail varchar(50),
  Content varchar(255),
  Timestamp varchar(25),
  PRIMARY KEY (TweetID),
  FOREIGN KEY (WriterID) REFERENCES User(UserID)
);
```

WriterName과 WriterEmail이 WriterID에 종속되어있음 – 종속성 문제 가능성

[1NF정규화]
```
CREATE TABLE Tweet (
  TweetID varchar(20),
  WriterID varchar(20),
  Content varchar(255),
  Timestamp varchar(25),
  PRIMARY KEY (TweetID),
  FOREIGN KEY (WriterID) REFERENCES User(UserID)
);
```
((WriteName과 WriterEmail을 제거해서 Atomic하도록함))

[2NF정규화]
```
CREATE TABLE Tweet (
  TweetID varchar(20),
  WriterID varchar(20),
  Content varchar(255),
  Timestamp varchar(25),
  PRIMARY KEY (TweetID),
  FOREIGN KEY (WriterID) REFERENCES User(UserID)
```

```
);

CREATE TABLE User (
  UserID varchar(20),
  Name varchar(50),
  Email varchar(50),
  PRIMARY KEY (UserID)
);
```
((WriterName과 WriterEmail이 WriterID에 완전 함수 종속이므로 User에서 가져온 것으로 WriterID 기본키로 사용))

[BCNF 정규화]
```
CREATE TABLE Tweet (
  TweetID varchar(20),
  WriterID varchar(20),
  Content varchar(255),
  Timestamp varchar(25),
  PRIMARY KEY (TweetID),
  FOREIGN KEY (WriterID) REFERENCES Writer(WriterID)
);

CREATE TABLE Writer (
  WriterID varchar(20) PRIMARY KEY,
  Name varchar(50),
  Email varchar(50)
);
```
((User 테이블에서 가져온 WriterID를 기본키로 사용하는 대신, Writer 테이블 따로 만들음 Writer과 WriterID가 후보키이고 모든 결정자가 후보키에만 의존하게함으로써 BCNF만족하게 함))

Follow 정규화 과정 :

[초기 테이블]

```
CREATE TABLE Follow (
  FollowID varchar(20) PRIMARY KEY,
  FollowerID varchar(20) REFERENCES User(UserID),
  FollowingID varchar(20) REFERENCES User(UserID),
  Timestamp varchar(25)
);
```

문제점 : 사용자에 대한 follower와 following 중복 데이터 가능성
(그 외 follower/following 목록 작업의 편리성 문제)

Follower와 Following으로 테이블 분리

```
CREATE TABLE Follower (
```

```
  UserID varchar(20),
  followingID varchar(20),
  Timestamp varchar(25),
  Primary key (UserID,FollowingID),
  Foreign key (followingID) references User(UserID),
  Foreign key(UserID) references USer(USerID)
);

CREATE TABLE Following (
  UserID varchar(20),
  FollowerID varchar(20),
  Timestamp varchar(25),
  Primary key (UserID, FollowerID),
  Foreign key (followerID) references User(UserID),
  Foreign Key(UserID) references User(UserID)
);
```

(해당 테이블은 BCNF를 만족)

BlackList 정규화 과정:

[초기 테이블]

```
CREATE TABLE BlackList (
  FollowID varchar(20),
  Timestamp varchar(25),
  PRIMARY KEY (FollowID)
);
```

문제점 : FollowID가 BlackList 테이블에 대해 명시적이지 않음(모호함), 차단 유저를 나타내는 key 누락 및 차단 관계를 알기 어려움.

```
CREATE TABLE BlackList (
  BlackListID varchar(20),
  BlockingUserID varchar(20),
  BlockedUserID varchar(20),
  Timestamp varchar(25),
  Primary Key (BlackListID),
  Foreign Key (BlockingUserID) references User(UserID),
  Foreign Key (BlockedUserID) references User(UserID)
);
```

(BlackListID로 변경, 차단 관계를 나타내기 위한 BlockingUserID와 BlockedUserID 추가)

[3NF 정규화]

이행종속성 문제 : primary key 수정 (BlackListID 삭제)

```
CREATE TABLE BlackList (
  BlockingUserID varchar(20),
  BlockedUserID varchar(20),
  Timestamp varchar(25),
  Primary Key (BlackingUserID, BlockedUserID),
  Foreign Key (BlockingUserID) references User(UserID),
  Foreign Key (BlockedUserID) references User(UserID)
);
```

(해당 테이블은 BCNF를 만족)

Tweet 정규화 과정:
[초기 테이블]

```
CREATE TABLE Tweet (
  TweetID varchar(20),
  WriterID varchar(20),
  Content varchar(255),
  Timestamp varchar(25),
  PRIMARY KEY (TweetID),
  Foreign key (writerID) references User(UserID)
);
```

반복 그룹이 WriterID 컬럼 형태로 존재 (하나의 트윗에 다수의 작성자 존재가능성)

[정규화]

```
CREATE TABLE Tweet (
  TweetID varchar(20),
  UserID varchar(20), -- Foreign key references User(UserID)
  Content varchar(255),
  Timestamp varchar(25),
  ㄴ(Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP 변경? Tweet 시각=현재시각)
  PRIMARY KEY (TweetID),
  FOREIGN KEY (UserID) REFERENCES User(UserID)
  FOREIGN KEY (RetweetID) REFERENCES Tweet(TweetID) (추가?)
);
```

(이제 tweet 테이블의 UserID 열은 User 테이블의 UserID 열을 참조하는 외부 키 역할. 그러면 tweet 테이블의 각 열에 원자 값이 포함되고 반복 그룹이 제거)
(Timestamp가 자동으로 Tweet 시각을 현재시각으로 저장)
(RetweetID 도 추적 가능)