Phase 3

Team Databaes

1 ER to Relation Model

In this section, we describe the changes made to the various entities to convert the ER diagram to a Relation model

1.1 Multivalued Attribute

1. Relation COURSE_INSTRUCTOR created to deal with CourseInstructor being a multivalued attribute in the Course entity. The new relation has the attribute InstructorName

2. BLOGTAG relation created as it was a multivalued attribute of BLOG entity. Added attribute Tag to identify each tag uniquely.

3. PINS relation created as it was a multivalued attribute of STUDY_GROUP. Added attribute PinDetails to identify each Pin uniquely

1.2 M:N Binary Relations

USED_FOR, FRIENDS_WITH, KNOWS, CONTAINS, PREREQUISITE are all M:N binary relationships, so created additional relations for each of them containing the primary keys of the entities they link and their respective relationship attributes.

1.3 Quaternary Relationship

A relation was made for the quaternary relationship PARTICIPATES_IN including foreign keys from all 4 entity types involved.

1.4 Handling Subclasses

1.4.1 Tradeoffs of modelling subclasses in POST

- Notes
  - Notice that BlogTag is multivalued, so it will require a new relation anyway.
  - Notice that Review has a relationship too
  - Last option (Single relation with multiple type attributes) not considered as the subclasses are disjoint and total.
- Disadvantages of **multiple relations - superclass and subclasses**
  - BlogTag already has a separate relation so Blog as a relation feels redundant.
  - The complex 3-valued key for POST (which is a weak entity) has to be copied for both REVIEW and BLOG
- Disadvantages of **multiple relations - subclass relations only**
  - BlogTag already has a separate relation so Blog as a relation feels redundant.
  - The relationship of POST along with it's 4 attributes and 2 owner-pk-attributes need to be copied for both REVIEW and BLOG, which feels very redundant
- Disadvantages of **single relation with one type attribute**
  - NULL in REVIEW attributes for both: CourseID foreign key and ReviewRating

  Finally, **Single relation with one type attribute** was chosen due to seemingly lesser disadvantages.

1.4.2 Tradeoffs of modelling subclasses in SG_EVENT

- Notes
  - Unlike POST this is less complicated, each class having 2 simple attributes.
  - Last option (Single relation with multiple type attributes) not considered as the subclasses are disjoint and total.
- Disadvantages of **multiple relations - superclass and subclasses**
  - The 2-valued key for SG_EVENT (which is a weak entity) has to be copied for both MEET and TARGET
- Disadvantages of **multiple relations - subclass relations only**
  - The 2 pk of SG_EVENT (foreign + partial) along with it's 2 attributes would have to be copied to both MEET and TARGET. This feels redundant
- Disadvantages of **single relation with one type attribute**
  - Both subclasses have 2 attributes each which would lead to 2 NULL values at all times.

Finally, **Multiple relations - superclass and subclasses** was chosen as it minimizes copied information while avoiding NULL values.

1.5 Handling Weak Entity Types

POST and SG_EVENT were the two weak entity types in our ER model, so a new relation was created copying the owner entity type's primary keys.

1.6 Relational Model after conversion from ER Diagram

The diagram can found here. The link to the DBML file has also been provided for reference
https://dbdiagram.io/d/5f670ef77da1ea736e2e894a
Note that this is already in 2NF form.

2 Normal Forms

2.1 Functional Dependencies

Besides the non-trivial functional dependencies and the functional dependencies with the primary key on the left-hand side, the only functional dependencies in the model are -

1. In LANGUAGE:
   $LangName \rightarrow LangCode$
2. In COURSE:
   $(CourseName, CourseOrg, CoursePlatform) \rightarrow CourseDifficulty$

2.2 Notes

In this section, we offer the rationale behind the existence, or lack thereof, of certain hypothetical functional dependencies.

1. In USER, a single Email can be associated with multiple accounts (UserName, DNum pairs) so it is not a candidate key. Moreover, Emails can also be fake so none of the other attributes can be assumed to be functionally dependent on Email.
2. In POST, Multiple clicks / scripts may end up leading to multiple posts (including duplicates) being made at the same timestamp. Thus, timestamp has not been included in any candidate keys. The only true discriminator (key) is thus the Postnumber, along with UserName, DNum (as postnumbers can be the same across users).
3. In SG_EVENT Similar to Post, 'duplicate' SG_EVENTs can exist for the same study group (identified by SgUrl) with the only difference being the EventNum. As EventNums can match across different study groups, the only key is (SgUrl, EventNum).
4. In STUDY_GROUP, SgUrl is the only key here as again duplicate Study Groups might get initialized (with 0 rating, 1 user and same create/update timestamps.), possibly using scripts.

5. In COURSE, the only key is CourseID because even for courses having the same CourseName (eg: 18.06 Linear Algebra), CourseOrg (eg: MIT OCW) and CoursePlatform (eg: Youtube) - due to there being multiple editions of the course (say across years) it's necessary to distinguish these. Thus CourseID was introduced as an identifying primary key by us, and also is the only key. Notice that while rating, duration, no. of hours required can change across different editions of a course, the difficulty (defined by us earlier to take one value among: "Beginner", "Intermediate" or "Advanced") will not change across these offerings. This leads to a functional dependency.

2.3 Changes introduced for normalization

2.3.1 1NF

The model obtained after conversion from ER model is already in 1NF, without any additional changes. Every attribute takes only atomic values. Multivalued attributes have already been handled by creating additional relations for them.

2.3.2 2NF

The model obtained after conversion from ER model is already in 1NF, without any additional changes. No attribute depends on a proper subset of a key. Our initial design was modular enough to ensure this.

2.3.3 3NF

The relation COURSE violates the 3NF property. We handle this by splitting it into two relations - COURSE and COURSE_DIFFICULTY. CourseDifficulty is removed from the prior relation while the latter contains the attributes CourseName, CourseOrg, CoursePlatform and CourseDifficulty.
Note that in LANGUAGE, LangName is a candidate key so despite the $LangName \rightarrow LangCode$ functional dependency, it does not violate 3NF (or BCNF).

3 Relational Database after normalizing to 3NF

The diagram can found here. The link to the DBML file has also been provided for reference
https://dbdiagram.io/d/5f64e9937da1ea736e2e6ee1
Note that this is also in BCNF

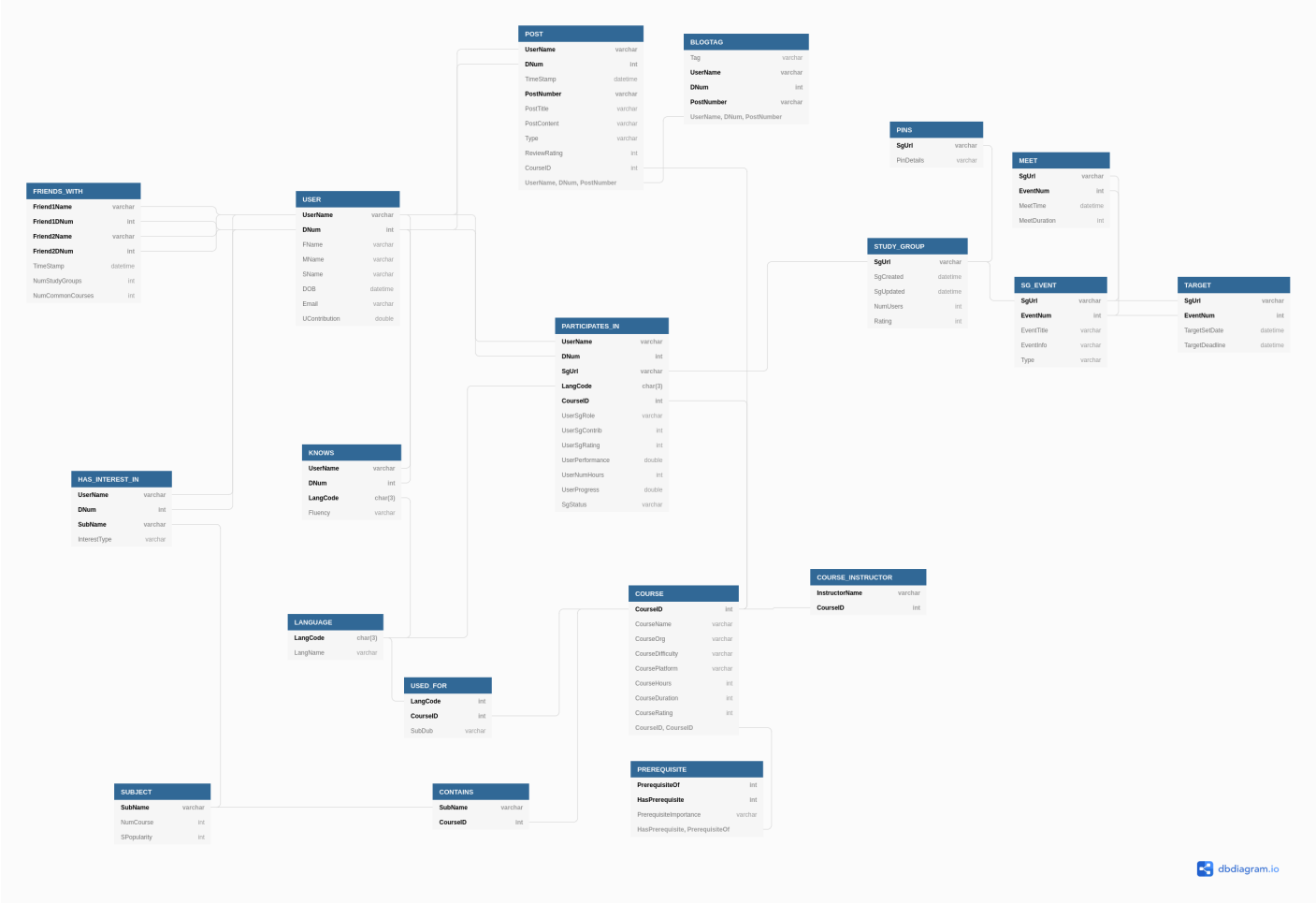

Figure 1: Relational database after conversion from ER model. Note that this is in 2NF

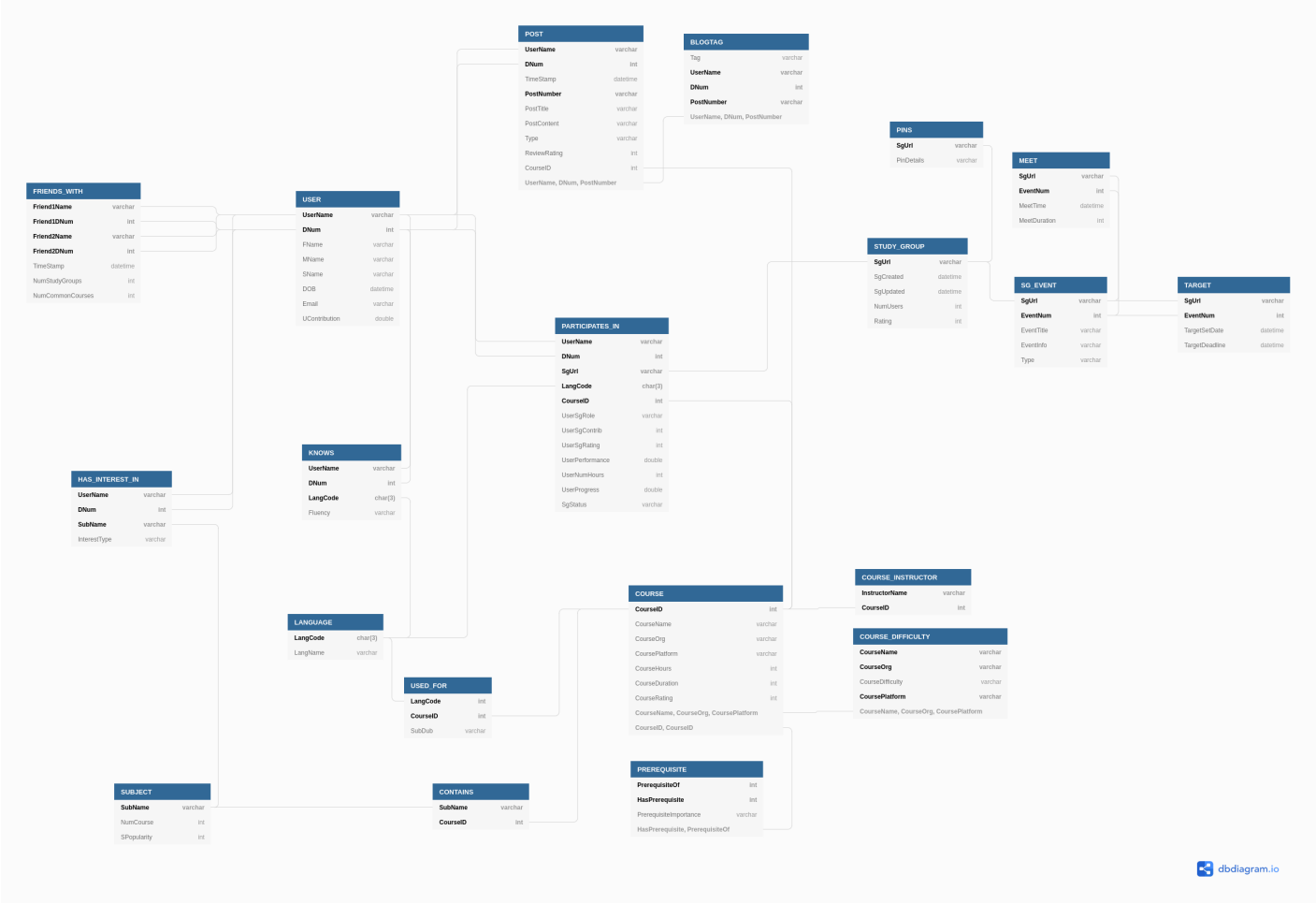

Figure 2: Relational database after normalizing to 3NF. Note that this is also in BCNF