CSC261 Project Milestone 3

Project Details

Name of the project: JsTube
Team number: 17
Team name: Jays Club
Group members:

- Junfei Liu (jliu137)
- Jinghan Lu (jlu54)
- Jinghao Jiang (jjiang27)

Task A[1]

Relation 1. `USER`

User_id	User_name	Email	Password	Membership

For relation `USER`

- The **primary key** is `User_id`
- The rest of the candidate keys (**secondary keys**) are `User_name` and `Email`[2]

The set of Functional Dependencies are

- `User_id → User_name`
- `User_id → Email`
- `User_id → Password`
- `User_id → Membership`
- `User_name → User_id`
- `User_name → Email`
- `User_name → Password`
- `User_name → Membership`
- `Email → User_id`
- `Email → User_name`
- `Email → Password`
- `Email → Membership`

[1] All functional dependencies listed in Task A are **minimal functional dependencies**
[2] By our design, each `email` address can only be used to register for one account and `user_name` cannot be the same for different users

All other attributes `Password` and `Membership` cannot functionally determine other attributes within the relation:

- Two users (different `User_id, User_name,` and `Email`) may accidentally choose the same `Password`, so `Password` cannot functionally determine other attributes within this relation.
- `Membership` is a boolean variable, so according to the pigeonhole principle, whenever there are more than two tuples in this table, there will be different users (with different `User_id, User_name,` and `Email`) that have the same `Membership` status.

For each functional dependency FD, the left hand side attribute is itself a candidate key (also a superkey), so the relation is already in **BCNF**.

Relation 2. `VIDEO`

Video_name[3]	Video_id	User_id	Upload_time[4]	Membership_ requirement	Link

For relation `VIDEO`

- The **primary key** is `Video_id`
- Since `Link` is unique to all videos, it is a **secondary key** in this relation

Set of Functional Dependencies:

- `Video_id → Video_name`
- `Video_id → User_id`
- `Video_id → Upload_time`
- `Video_id → Membership_requirement`
- `Video_id → Link`
- `Link → Video_id`
- `Link → Video_name`
- `Link → User_id`
- `Link → Upload_time`
- `Link → Membership_requirement`

All attributes other than candidate keys cannot form functional dependencies on any attributes:

- `Video_name`: Different users can upload videos with the same title. So it is impossible for Video_name to functionally determine other attributes.
- `User_id`: The same user can upload videos on the same day, so in this case, they will get the same values for `Upload_day`, `Upload_month` and `Upload_year`. But

[3] By our design, different video may have the same `video_name`
[4] We allow the same user to upload more than one video each day

attributes `Video_name`, `Video_id`, `Membership_requirement` and `Link` are different in these tuples. So `User_id` cannot functionally determine other attributes.
- `Upload_time`: Same as `User_id`, the same person can upload several videos on the same day. But attributes `Video_name, Video_id, Membership_requirement` and `Link` are different in these tuples.
- `Membership_requirement`: This is a boolean attribute. Same reason as Membership in USER relation.

For each functional dependency, the LHS are all candidate keys which are superkeys of the relation, so they are in BCNF.

Relation 3. `MEMBERSHIP`

User_id	Level	Valid_to_time

For relation `MEMBERSHIP`
- The **primary key** is `User_id`

Set of Functional Dependencies:
- `User_id` → `Level`
- `User_id` → `Valid_to_time`

All attributes other than candidate keys cannot form functional dependencies on any attributes:
- `Level`: Same level will have many user tuples, all with different values for `User_id`, `Valid_to_time`. So `Level` cannot functionally determine any attributes.
- `Valid_to_time`: On the same date, there might be many users whose membership has expired. So these three attributes cannot form functional dependencies with other attributes.

For each functional dependency, the LHS are all candidate keys which are superkeys of the relation, so they are in BCNF.

Relation 4. `ADMINISTRATOR`

Administrator_id	Email	Password

For relation `ADMINISTRATOR`
- The **primary key** is `Administrator_id`
- The rest of the candidate keys (**secondary keys**) are `Email`

Set of Functional Dependencies:

- {Administrator_id, Email} → Password

All other attributes Password cannot functionally determine other attributes within the relation:

- Two administrators (different Administrator_id and Email) may accidentally choose the same Password, so Password cannot functionally determine other attributes within this relation.

For each functional dependency, the LHS are all candidate keys which are superkeys of the relation, so they are in BCNF.

Relation 5. CASE

Case_id	Administrator_id	Video_id	Type	Report_user_id

For relation CASE

- The **primary key** is Case_id

Set of Functional Dependencies:

- Case_id → Administrator_id
- Case_id → Video_id
- Case_id → Type
- Case_id → Report_user_id

All other attributes Administrator_id, Video_id, Type, Report_user_id cannot functionally determine other attributes within the relation:

- One administrator may have access to multiple cases, so Administrator_id cannot functionally determine other attributes.
- A video could be reported more than once, so there will be difference cases with the same Video_id.
- The Type of cases could be the same for different reported cases.
- A user may report many cases, so the Report_user_id cannot determine the specific case.

For each functional dependency, the LHS are all candidate keys which are superkeys of the relation, so they are in BCNF.

Relation 6. VIEW

Video_id	User_id	Times

For relation VIEW

- The **primary key** is the combination of `Video_id` and `User_id`

The set of Functional Dependencies are

- `{Video_id, User_id} → Times`

All other attributes `Times` cannot functionally determine other attributes within the relation:

- The number of times a user has watched a video is not unique. A user could watch two different videos once each. Therefore, `Times` cannot functionally determine the other two attributes.

For each functional dependency, the LHS are all candidate keys which are superkeys of the relation, so they are in BCNF.