**Design Concept**

1. **Member** has all users’ profile including: name, gender, date of birth; which is listed by primary key: member\_ID
2. **Privileged Member** which is a subset of **Member** stores the payment confirmation\_ID

(confirmation ID should not be included in **Member;** this table might be locked for security access)

Example:

Table1: Member

|  |  |  |  |
| --- | --- | --- | --- |
| Member\_ID(PK) | Name | Gender | Birth Date |
| 1001 | Namo | Male | … |
| 1002 | Jason | Male | … |
| 1003 | Joanna | Female | … |
| 1004 | Jessy | Male | … |
| 1005 | Min | Male | … |

Table2: Privileged Member

|  |  |
| --- | --- |
| Member\_ID(PK) | Confirmation\_ID |
| 1001 | abcxxx |
| 1005 | Defyyy |

1. All pictures are grouped into albums. Each picture belongs to only one album and each album is owned by a single user.

(= a user can have multiple albums and each album can have multiple pictures)

Example:

Table3: Album

|  |  |
| --- | --- |
| album\_ID(PK) | Member\_Id(FK) |
| A | 1001 |
| B | 1001 |
| C | 1003 |

Table4: Picture

|  |  |  |
| --- | --- | --- |
| pic\_ID(PK) | Link(addr) | album\_ID(FK) |
| i. | … | A |
| ii. | … | C |
| iii. | … | B |
| iV. | … | A |

1. A picture may contain persons and users can be tagged by other users in the picture. Each tag can refer to only one user in the picture. A picture can have multiple tags when there are may users in it, while a tag can belong to only one picture.

Table6: Tags

|  |  |  |
| --- | --- | --- |
| Tag\_ID(PK) | member\_Id(FK) | pic\_Id(FK) |
| 1 | 1001 | i. |
| 2 | 1001 | ii. |
| 3 | 1004 | i. |
| 4 | 1005 | v. |

1. A regular user can post a status with text and multiple pictures. A privileged user can post an advanced status with text, multiple pictures and a video. For each status a user posts, including advanced status, any user can comment on it with text and one picture. Assume one picture can only appear in one status (including advanced status) or in one comment.

(= a disjoint relationship between **Picture** vs **Status**, **Advanced Status** and **Comment**)

Table7: SinglePic (1:1)

|  |  |
| --- | --- |
| pic\_ID(PK) | comment\_ID(FK) |
| 1 | 1 |
| 2 | 4 |
| 3 | 7 |

Table 8: MultiplePicToAdvStatus (N:1)

|  |  |
| --- | --- |
| pic\_ID(PK) | advStatus\_Id(FK) |
| 4 | 100005 |
| 5 | 100005 |
| 6 | 100001 |

Table 9: MultiplePicToStatus (N:1)

|  |  |
| --- | --- |
| pic\_ID(PK) | status\_Id(FK) |
| 7 | 201 |
| 8 | 201 |
| 9 | 203 |

1. Users have friendships, which are bi-directional

Assume that Member 1001 and 1002 are friends. It can be presented by two different ways.

Method1: redundant but search fast!

|  |  |  |
| --- | --- | --- |
| Friendship\_Id(PK) | Member\_id(FK1) | Member\_Id(FK2) |
| 1 | 1001 | 1002 |
| 2 | 1002 | 1001 |

Method2: only one item left in the table but search slow

|  |  |  |
| --- | --- | --- |
| Friendship\_Id(PK) | Member\_id(FK1) | Member\_Id(FK2) |
| 1 | 1001 | 1002 |

For ER diagram, there is no difference.

Example:

|  |  |  |
| --- | --- | --- |
| Friendship\_Id(PK) | Member\_id(FK1) | Member\_Id(FK2) |
| 1 | 1001 | 1002 |
| 2 | 1001 | 1003 |
| 3 | 1001 | 1007 |
| 4 | 1002 | 1005 |
| 5 | 1002 | 1008 |
| 6 | 1005 | 1008 |