julakanti harihanth (s4)

inagalla lohiya (s4)

mandala sai kushal (s4)

rahul nikki y (s5)

MULTIMEDIA WEB APPLICATION

COURSE INCHARGE | AMIT KUMAR

[Year]

**Contents:**

* **Problem Statement**
* **Application Overview**
* **Management and utilization perspectives**
* **Implementation and design of database management systems**
* **Entities used**
* **ER Diagram**
* **Relational Schema**
* **Table schema**
* **My SQL Executable code for tables**
* **Conversion table**
* **Functional dependencies**
* **Normalization**
* **Sample output of final normalized database table**

**Application Overview**

Multimedia web application is an online portal where a user have a unique id, username and password and for security reasons hash password also must be saved can upload media in which divides into images, videos, music in which each has a unique video\_id, data\_name for recognition which is given by the user and has address pointing to its location in his database and view which specifies whether certain media should be shown or hidden. Each type of media has its own section. Where he can upload images, videos, text and audio.

Store your media online and view media of our friends and this has separate sections of photos and videos and music and some media and can be made private and some other features.

A user can send a friend request for another user and can like, comment, share a particular media. User can also add the images from facebook, videos from YouTube, text from Gmail, music from sound cloud which have same options like the other media.

**Management and Utilization Perspectives:**

The traditional lifestyles of people have changed- Is your target audience still reading printed newspapers or magazines? Are they still watching that one popular show where the entire family gathers on a Sunday morning? So where are they then? On the world wide web- probably liking or sharing a post on facebook, enquiring regarding finding accommodation on twitter, commenting on a movie review posted on a blog, repining an amazing dress design they saw on their pin board or watching an advertisement released on youtube.

So By this project we want to widen the features like the user can directly upload his favourite videos, Music, and images from youtube, Facebook and Soundcloud respectively directly by pasting its address in the user input and can have whole media organized at one place so no need to visit all sites again and again and integrity, security and authentication is given importance for the user.

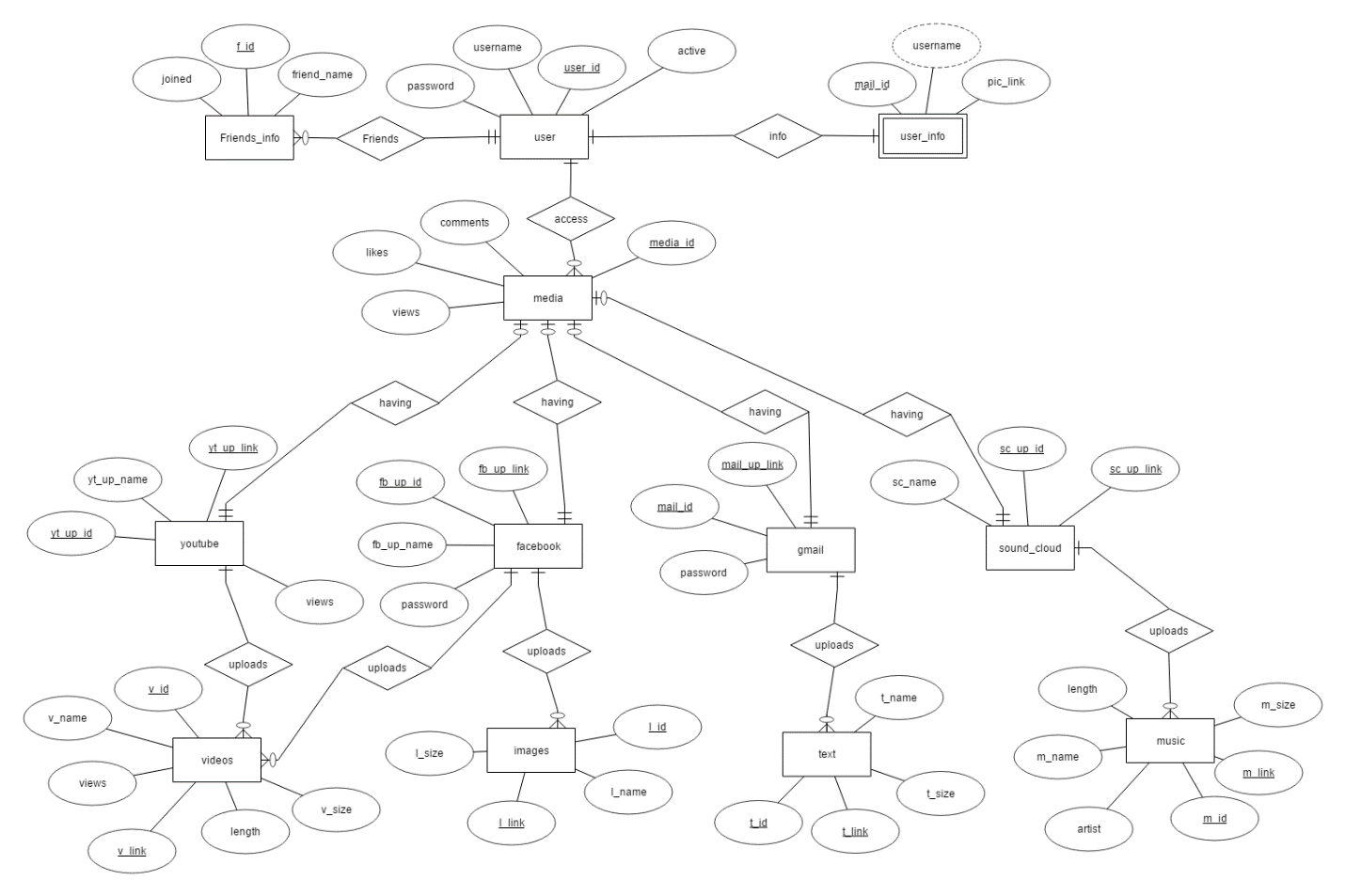
**Implementation and design of database management Systems:**

As there will be large number of tables and databases for all the users there might be concurrency and deadlock in some situations.So,the tables are normalized to at least to 3NF and some tables to BCNF according to the functional dependencies and decomposition of tables.

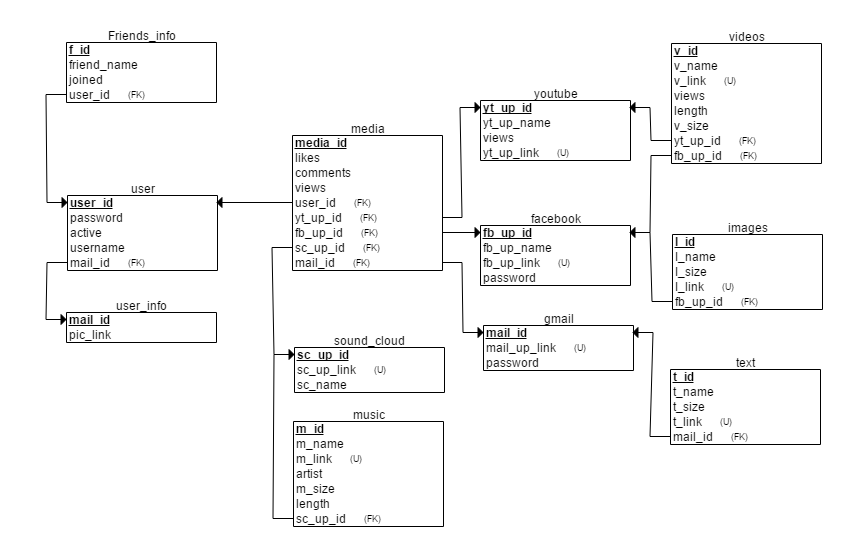
**Entities used:**

* user(user\_id, username, active,password).
* user\_info(mail\_id,username,pic\_link).
* friends\_info(f\_id,friend\_name,joined).
* media(media\_id,comments,likes,views).
* youtube(yt\_up\_id,yt\_up\_link,yt\_up\_name,views).
* facebook(fb\_up\_id,fb\_up\_link,fb\_up\_name,password).
* gmail(mail\_id,mail\_up\_link,password).
* soundcloud(sc\_up\_id,sc\_up\_link,sc\_name).
* videos(v\_id,v\_link,v\_name,length,v\_size,views).
* images(I\_id,I\_link,I\_name,I\_size).
* text(t\_id,t\_link,t\_name,t\_size).
* music(m\_id,m\_link,m\_name,m\_size,length,artist)

**ER DIAGRAM:**

****

**Relational Schema:**



**Table schema:**

1. user (user\_id, username, active,password).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| user id | Int | Primary Key |
| Username | Varchar (50) | Not Null |
| Password | Varchar (50) | Not Null |
| active | Varchar (50) | Not Null |

1. user\_info(mail\_id,username,pic\_link).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| mail\_id | Int | Primary Key |
| username | Varchar (50) | Not Null |
| pic\_link | Varchar (50) | Not Null |

1. friends\_info(f\_id,friend\_name,joined).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| f\_id | Int | Primary Key |
| friend\_name | Varchar (50) | Not Null |
| joined | Varchar (50) | Not Null |

1. media(media\_id,comments,likes,views).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| media\_id | Int | Primary Key |
| comments | Varchar (100) | Not Null |
| likes | Int | Not Null |
| views | Int | Not Null |

1. youtube(yt\_up\_id,yt\_up\_link,yt\_up\_name,views).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| yt\_up\_id | Int | Primary Key |
| yt\_up\_link | Int | Candidate Key |
| yt\_up\_name | Varchar (50) | Not Null |
| Views | Int | Not Null |

1. facebook(fb\_up\_id,fb\_up\_link,fb\_up\_name,password).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| fb\_up\_id | Int | Primary Key |
| fb\_up\_link | Int | Candidate Key |
| fb\_up\_name | Varchar (50) | Not Null |
| password | Varchar (50) | Not Null |

1. gmail(mail\_id,mail\_up\_link,password).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| mail\_id | Int | Candidate Key |
| mail\_up\_link | Int | Primary Key |
| password | Varchar (50) | Not Null |

1. soundcloud(sc\_up\_id,sc\_up\_link,sc\_name).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| sc\_up\_id | Int | Primary Key |
| sc\_up\_link | Int | Candidate Key |
| sc\_name | Varchar (50) | Not Null |
| Size | Int | Not Null |

1. videos(v\_id,v\_link,v\_name,length,v\_size,views).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| v\_id | Int | Primary Key |
| v\_link | Int | Foreign Key |
| v\_name | Varchar (50) | Not Null |
| length | Int | Not Null |
| v\_size | Int | Not Null |
| Views | Int | Not Null |

1. images(I\_id,I\_link,I\_name,I\_size).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| I\_id | Int | Primary Key |
| I\_link | Int | Foreign Key |
| I\_name | Varchar (50) | Not Null |
| I\_size | Int | Not Null |

1. text(t\_id,t\_link,t\_name,t\_size).

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| t\_id | Int | Primary Key |
| t\_link | Int | Foreign Key |
| t\_name | Varchar (50) | Not Null |
| t\_size | Int | Not Null |

1. music(m\_id,m\_link,m\_name,m\_size,length,artist)

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| m\_id | Int | Primary Key |
| m\_link | Int | Foreign Key |
| m\_name | Varchar (50) | Not Null |
| m\_size | Int | Not Null |
| length | Int | Not Null |
| artist | Varchar (50) | Not Null |

**MY SQL Executable code for tables:**

CREATE TABLE user

(

password INT NOT NULL,

active INT NOT NULL,

user\_id INT NOT NULL,

username INT NOT NULL,

mail\_id INT NOT NULL,

PRIMARY KEY (user\_id),

FOREIGN KEY (mail\_id) REFERENCES user\_info(mail\_id)

);

CREATE TABLE user\_info

(

pic\_link INT NOT NULL,

mail\_id INT NOT NULL,

PRIMARY KEY (mail\_id)

);

CREATE TABLE Friends\_info

(

f\_id INT NOT NULL,

friend\_name INT NOT NULL,

joined INT NOT NULL,

user\_id INT NOT NULL,

PRIMARY KEY (f\_id),

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

CREATE TABLE media

(

likes INT NOT NULL,

comments INT NOT NULL,

views INT NOT NULL,

media\_id INT NOT NULL,

user\_id INT NOT NULL,

yt\_up\_id INT NOT NULL,

fb\_up\_id INT NOT NULL,

sc\_up\_id INT NOT NULL,

mail\_id INT NOT NULL,

PRIMARY KEY (media\_id),

FOREIGN KEY (user\_id) REFERENCES user(user\_id),

FOREIGN KEY (yt\_up\_id) REFERENCES youtube(yt\_up\_id),

FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),

FOREIGN KEY (sc\_up\_id) REFERENCES sound\_cloud(sc\_up\_id),

FOREIGN KEY (mail\_id) REFERENCES gmail(mail\_id)

);

CREATE TABLE youtube

(

yt\_up\_name INT NOT NULL,

yt\_up\_id INT NOT NULL,

views INT NOT NULL,

yt\_up\_link INT NOT NULL,

PRIMARY KEY (yt\_up\_id),

UNIQUE (yt\_up\_link)

);

CREATE TABLE facebook

(

fb\_up\_name INT NOT NULL,

fb\_up\_id INT NOT NULL,

fb\_up\_link INT NOT NULL,

password INT NOT NULL,

PRIMARY KEY (fb\_up\_id),

UNIQUE (fb\_up\_link)

);

CREATE TABLE gmail

(

mail\_id INT NOT NULL,

mail\_up\_link INT NOT NULL,

password INT NOT NULL,

PRIMARY KEY (mail\_id),

UNIQUE (mail\_up\_link)

);

CREATE TABLE sound\_cloud

(

sc\_up\_id INT NOT NULL,

sc\_up\_link INT NOT NULL,

sc\_name INT NOT NULL,

PRIMARY KEY (sc\_up\_id),

UNIQUE (sc\_up\_link)

);

CREATE TABLE videos

(

v\_id INT NOT NULL,

v\_name INT NOT NULL,

v\_link INT NOT NULL,

views INT NOT NULL,

length INT NOT NULL,

v\_size INT NOT NULL,

yt\_up\_id INT NOT NULL,

fb\_up\_id INT NOT NULL,

PRIMARY KEY (v\_id),

FOREIGN KEY (yt\_up\_id) REFERENCES youtube(yt\_up\_id),

FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),

UNIQUE (v\_link)

);

CREATE TABLE images

(

I\_id INT NOT NULL,

I\_name INT NOT NULL,

I\_size INT NOT NULL,

I\_link INT NOT NULL,

fb\_up\_id INT NOT NULL,

PRIMARY KEY (I\_id),

FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),

UNIQUE (I\_link)

);

CREATE TABLE text

(

t\_name INT NOT NULL,

t\_size INT NOT NULL,

t\_id INT NOT NULL,

t\_link INT NOT NULL,

mail\_id INT NOT NULL,

PRIMARY KEY (t\_id),

FOREIGN KEY (mail\_id) REFERENCES gmail(mail\_id),

UNIQUE (t\_link)

);

CREATE TABLE music

(

m\_id INT NOT NULL,

m\_name INT NOT NULL,

m\_link INT NOT NULL,

artist INT NOT NULL,

m\_size INT NOT NULL,

length INT NOT NULL,

sc\_up\_id INT NOT NULL,

PRIMARY KEY (m\_id),

FOREIGN KEY (sc\_up\_id) REFERENCES sound\_cloud(sc\_up\_id),

UNIQUE (m\_link)

);

**Conversion table:**

|  |  |  |
| --- | --- | --- |
| **ENTITIES** | **RELATION** | **MySQL EXECUTABLE CODE** |
| user | User to user\_info:  *One to One*  User to friends\_info:  *Many to One*  User to media:  *Many to One* | CREATE TABLE user  (  password INT NOT NULL,  active INT NOT NULL,  user\_id INT NOT NULL,  username INT NOT NULL,  mail\_id INT NOT NULL,  PRIMARY KEY (user\_id),  FOREIGN KEY (mail\_id) REFERENCES user\_info(mail\_id)  ); |
| User\_info | User\_info to user:  *One to One* | CREATE TABLE user\_info  (  pic\_link INT NOT NULL,  mail\_id INT NOT NULL,  PRIMARY KEY (mail\_id)  ); |
| Friends\_info | Friends\_info to user:  One to Many | CREATE TABLE Friends\_info  (  f\_id INT NOT NULL,  friend\_name INT NOT NULL,  joined INT NOT NULL,  user\_id INT NOT NULL,  PRIMARY KEY (f\_id),  FOREIGN KEY (user\_id) REFERENCES user(user\_id)  ); |
| Media | Media to youtube:  *One to One*  Media to facebook:  *One to One*  Media to gmail:  *One to One*  Media to sound cloud:  *One to One* | CREATE TABLE media  (  likes INT NOT NULL,  comments INT NOT NULL,  views INT NOT NULL,  media\_id INT NOT NULL,  user\_id INT NOT NULL,  yt\_up\_id INT NOT NULL,  fb\_up\_id INT NOT NULL,  sc\_up\_id INT NOT NULL,  mail\_id INT NOT NULL,  PRIMARY KEY (media\_id),  FOREIGN KEY (user\_id) REFERENCES user(user\_id),  FOREIGN KEY (yt\_up\_id) REFERENCES youtube(yt\_up\_id),  FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),  FOREIGN KEY (sc\_up\_id) REFERENCES sound\_cloud(sc\_up\_id),  FOREIGN KEY (mail\_id) REFERENCES gmail(mail\_id)  ); |
| Youtube | Youtube to media:  *One to One*  Youtube to videos:  *Many to One* | CREATE TABLE youtube  (  yt\_up\_name INT NOT NULL,  yt\_up\_id INT NOT NULL,  views INT NOT NULL,  yt\_up\_link INT NOT NULL,  PRIMARY KEY (yt\_up\_id),  UNIQUE (yt\_up\_link)  ); |
| Facebook | Facebook to media:  *One to One*  Facebook to videos:  *Many to One*  Facebook to images:  *Many to One* | CREATE TABLE facebook  (  fb\_up\_name INT NOT NULL,  fb\_up\_id INT NOT NULL,  fb\_up\_link INT NOT NULL,  password INT NOT NULL,  PRIMARY KEY (fb\_up\_id),  UNIQUE (fb\_up\_link)  ); |
| Gmail | Gmail to media:  *One to One*  Gmail to text:  *Many to One* | CREATE TABLE gmail  (  mail\_id INT NOT NULL,  mail\_up\_link INT NOT NULL,  password INT NOT NULL,  PRIMARY KEY (mail\_id),  UNIQUE (mail\_up\_link)  ); |
| Sound cloud | Sound cloud to media:  *One to One*  Sound cloud to music:  *Many to One* | CREATE TABLE sound\_cloud  (  sc\_up\_id INT NOT NULL,  sc\_up\_link INT NOT NULL,  sc\_name INT NOT NULL,  PRIMARY KEY (sc\_up\_id),  UNIQUE (sc\_up\_link)  ); |
| Videos | Videos to youtube:  *One to Many*  Videos to facebook:  *One to Many* | CREATE TABLE videos  (  v\_id INT NOT NULL,  v\_name INT NOT NULL,  v\_link INT NOT NULL,  views INT NOT NULL,  length INT NOT NULL,  v\_size INT NOT NULL,  yt\_up\_id INT NOT NULL,  fb\_up\_id INT NOT NULL,  PRIMARY KEY (v\_id),  FOREIGN KEY (yt\_up\_id) REFERENCES youtube(yt\_up\_id),  FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),  UNIQUE (v\_link)  ); |
| Images | Images to facebook:  *One to Many* | CREATE TABLE images  (  I\_id INT NOT NULL,  I\_name INT NOT NULL,  I\_size INT NOT NULL,  I\_link INT NOT NULL,  fb\_up\_id INT NOT NULL,  PRIMARY KEY (I\_id),  FOREIGN KEY (fb\_up\_id) REFERENCES facebook(fb\_up\_id),  UNIQUE (I\_link)  ); |
| Text | Text to gmail:  *One to Many* | CREATE TABLE text  (  t\_name INT NOT NULL,  t\_size INT NOT NULL,  t\_id INT NOT NULL,  t\_link INT NOT NULL,  mail\_id INT NOT NULL,  PRIMARY KEY (t\_id),  FOREIGN KEY (mail\_id) REFERENCES gmail(mail\_id),  UNIQUE (t\_link)  ); |
| Music | Music to sound cloud:  *One to Many* | CREATE TABLE music  (  m\_id INT NOT NULL,  m\_name INT NOT NULL,  m\_link INT NOT NULL,  artist INT NOT NULL,  m\_size INT NOT NULL,  length INT NOT NULL,  sc\_up\_id INT NOT NULL,  PRIMARY KEY (m\_id),  FOREIGN KEY (sc\_up\_id) REFERENCES sound\_cloud(sc\_up\_id),  UNIQUE (m\_link)  ); |

**Functional dependencies:**

1. **User:**

user\_id -> username,password,active.

Username,password ->user\_id,active.

1. **friends\_info:**

f\_id-> friend\_name,joined.

1. **Media**:

Media\_id-> likes,comments,views.

1. **user\_info:**

mail\_id,username ->pic\_link.

1. **Youtube:**

yt\_up\_link-> yt\_up\_id,yt\_up\_name,views.

yt\_up\_id-> yt\_up\_link,yt\_up\_name,views.

yt\_up\_name-> views.

1. **Facebook:**

fb\_up\_id -> fb\_up\_link, fb\_up\_name,password.

fb\_up\_link-> fb\_up\_id, fb\_up\_name,password.

fb\_up\_name,password -> fb\_up\_id, fb\_up\_link.

1. **Soundcloud:**

sc\_up\_link -> sc\_name,sc\_up\_id.

sc\_up\_id -> sc\_up\_name,sc\_up\_link.

1. **Gmail:**

mail\_id -> mail\_up\_link,password.

mail\_up\_link ->mail\_id,password.

1. **Videos:**

v\_id -> v-link,v\_name,size,length,views.

v\_link ->v\_id,v\_name,size,length,views.

v\_name,size ->length,views.

1. **Images:**

I\_link ->I\_id,I\_name,I\_size.

I\_id ->I\_link,I\_name,size.

1. **Text:**

t\_id ->t\_link,t\_name,t\_size.

t\_link ->t\_id,t\_name,t-size.

1. **Music:**

m\_id ->m\_link,m\_name,length,artist,m\_size.

m\_link ->m\_id,m\_name,length,artist,m\_size.

m\_name,m\_size-> length,artist.

**Normalization:**

1. **User:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { user\_id}, { password,username}, The set of key attributes are: { user\_id,password,username }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: user\_id --> username,active,password

checking FD: username,password --> user\_id,active

**3NF**

find all cadnidate keys. The candiates keys are { user\_id}, { password,username}, The set of key attributes are: { user\_id,password,username }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency user\_id --> username,active,password

checking functional dependency username,password --> user\_id,active

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **friends\_info:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { f\_id}, The set of key attributes are: { f\_id }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: f\_id --> friend\_name,joined

checking FD: f\_id,joined --> friend\_name

checking FD: f\_id,friend\_name --> joined

**3NF**

find all cadnidate keys. The candiates keys are { f\_id}, The set of key attributes are: { f\_id }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency f\_id --> friend\_name,joined

checking functional dependency f\_id,joined --> friend\_name

checking functional dependency f\_id,friend\_name --> joined

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Media:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { media\_id}, The set of key attributes are: { media\_id }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: media\_id --> comments,likes,views

**3NF**

find all cadnidate keys. The candiates keys are { media\_id}, The set of key attributes are: { media\_id }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency media\_id --> comments,likes,views

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **user\_info:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { mail\_id,username}, The set of key attributes are: { mail\_id,username }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: mail\_id,username --> pic\_link

**3NF**

find all cadnidate keys. The candiates keys are { mail\_id,username}, The set of key attributes are: { mail\_id,username }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency mail\_id,username --> pic\_link

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Youtube:**

**1NF:**

All the attributes are atomic in nature.

**2NF:**

All the attributes are dependent on primary key only.

**3NF:**

Original schema { yt\_up\_id,yt\_up\_link,yt\_up\_name,views}

Primary key -> {yt\_up\_id, yt\_up\_link}

Yt\_up\_name -> views

Yt\_up\_id -> yt\_up\_name

Hence not in 3NF

So new table scheme

Scheme1 {yt\_up\_link, yt\_up\_name, views}

Scheme2 {yt\_up\_id, yt\_up\_name}

**BCNF**:

There is no inter relation among the candidate keys

1. **Facebook:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { fb\_up\_id}, { fb\_up\_link}, { fb\_up\_name,password}, The set of key attributes are: { fb\_up\_id,fb\_up\_link,fb\_up\_name,password }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: fb\_up\_id --> fb\_up\_link,fb\_up\_name,password

checking FD: fb\_up\_link --> fb\_up\_id,fb\_up\_name,password

checking FD: fb\_up\_name,password --> fb\_up\_id,fb\_up\_link

**3NF**

find all cadnidate keys. The candiates keys are { fb\_up\_id}, { fb\_up\_link}, { fb\_up\_name,password}, The set of key attributes are: { fb\_up\_id,fb\_up\_link,fb\_up\_name,password }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency fb\_up\_id --> fb\_up\_link,fb\_up\_name,password

checking functional dependency fb\_up\_link --> fb\_up\_id,fb\_up\_name,password

checking functional dependency fb\_up\_name,password --> fb\_up\_id,fb\_up\_link

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Soundcloud:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { sc\_up\_id}, { sc\_up\_link}, The set of key attributes are: { sc\_up\_id,sc\_up\_link }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: sc\_up\_id --> sc\_up\_link,sc\_name

checking FD: sc\_up\_link --> sc\_up\_id,sc\_name

**3NF**

find all cadnidate keys. The candiates keys are { sc\_up\_id}, { sc\_up\_link}, The set of key attributes are: { sc\_up\_id,sc\_up\_link }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency sc\_up\_id --> sc\_up\_link,sc\_name

checking functional dependency sc\_up\_link --> sc\_up\_id,sc\_name

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Gmail:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { mail\_id}, { mail\_up\_link}, The set of key attributes are: { mail\_id,mail\_up\_link }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: mail\_id --> mail\_up\_link,password

checking FD: mail\_up\_link --> mail\_id,password

**3NF**

find all cadnidate keys. The candiates keys are { mail\_id}, { mail\_up\_link}, The set of key attributes are: { mail\_id,mail\_up\_link }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency mail\_id --> mail\_up\_link,password

checking functional dependency mail\_up\_link --> mail\_id,password

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Videos:**

**1NF:**

All the attributes are atomic in nature.

**2NF:**

All the attributes are dependent on primary key only.

**3NF:**

Original schema { v\_id, v\_link, v\_name, length, v\_size, views }

Primary key -> {v\_id, v\_link}

v\_id -> v\_name, v\_size

v\_name, v\_size -> length, views

Hence not in 3NF

So new table scheme

Scheme1 { v\_id, v\_name, v\_size}

Scheme2 {v\_name, v\_size, length, views}

**BCNF**:

There is no inter relation among the candidate keys.

1. **Images:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { I\_id}, { I\_link}, The set of key attributes are: { I\_id,I\_link }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: I\_link --> I\_id,I\_name,I\_size

checking FD: I\_id --> I\_link,I\_name,I\_size

**3NF**

find all cadnidate keys. The candiates keys are { I\_id}, { I\_link}, The set of key attributes are: { I\_id,I\_link }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency I\_link --> I\_id,I\_name,I\_size

checking functional dependency I\_id --> I\_link,I\_name,I\_size

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Text:**

(It satisfies 1NF,2NF,3NF,BCNF)

**1NF:**

All the attributes are atomic in nature.

**2NF**

find all candidate keys. The candiates keys are { t\_id}, { t\_link}, The set of key attributes are: { t\_id,t\_link }

for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes

checking FD: t\_id --> t\_link,t\_name,t\_size

checking FD: t\_link --> t\_id,t\_name,t\_size

**3NF**

find all cadnidate keys. The candiates keys are { t\_id}, { t\_link}, The set of key attributes are: { t\_id,t\_link }

for each FD, check whether the LHS is superkey or the RHS are all key attributes

checking functional dependency t\_id --> t\_link,t\_name,t\_size

checking functional dependency t\_link --> t\_id,t\_name,t\_size

**BCNF**

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

1. **Music:**

**1NF:**

All the attributes are atomic in nature.

**2NF:**

All the attributes are dependent on primary key only.

**3NF:**

Original schema { m\_id, m\_link, m\_name, m\_size, length, artist }

Primary key -> {m\_id, m\_link}

m\_id -> m\_name, m\_size

m\_name, m\_size -> length, views

Hence not in 3NF

So new table scheme

Scheme1 { m\_id, m\_name, m\_size}

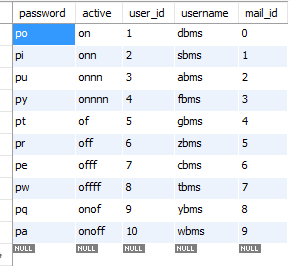
Scheme2 {m\_name, m\_size, length, views}

**BCNF**:

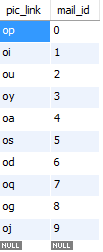
There is no inter relation among the candidate keys.

**Sample output of final normalized database table:**

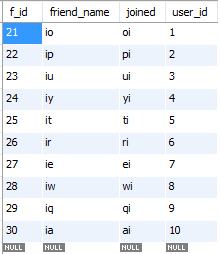
1. User



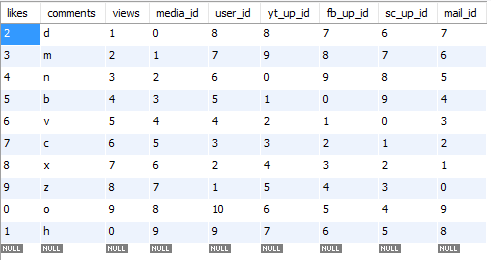
1. User\_info



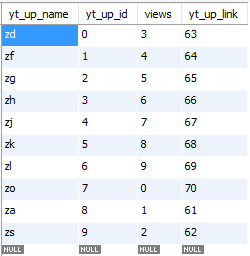
1. Friends\_info



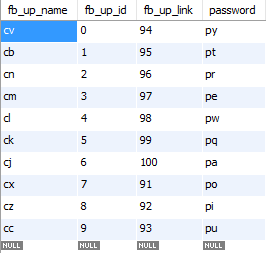
1. Media



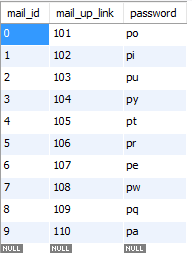
1. Youtube



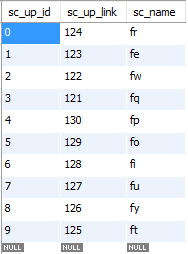
1. Facebook



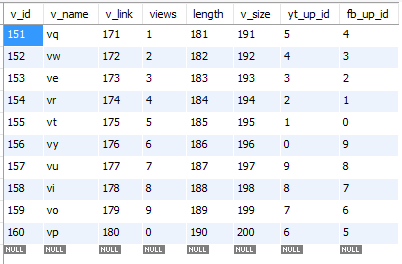
1. Gmail



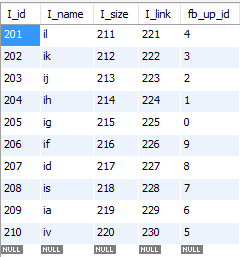
1. Sound\_cloud



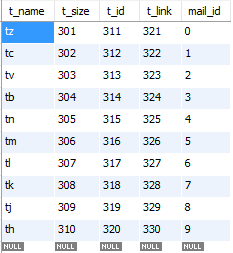
1. Videos



1. Images



1. Text



1. Music

