Data Modeling in NoSQL (C*) - Examples

Big Data Systems

Dr. Rubi Boim

Motivation (for this course)

- Learn modeling techniques by examples

- In real life you will blend strategies from each

Use cases

- Shopping cart
- Sensor data
- Gmail
- Instagram

Shopping cart

Requirements

- Customers can add items from a catalog
- Fast read performance

Shopping cart

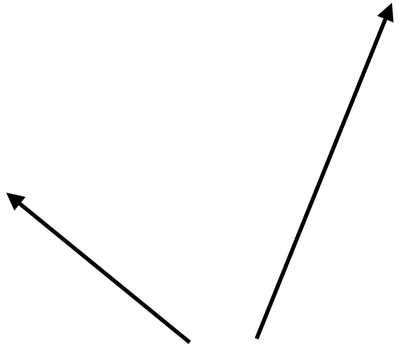
shoppingcart		
user_id	BIGINT	K
item_id	BIGINT	🔻
count	INT	
time_added	TIMESTAMP	
title	TEXT	
cost	DOUBLE	

items		
item_id	BIGINT	K
title	BIGINT	
cost	INT	
inventory_count	TIMESTAMP	

Shopping cart

shoppingcart		
user_id	BIGINT	K
item_id	BIGINT	▼
count	INT	
time_added	TIMESTAMP	
title	TEXT	
cost	DOUBLE	

items		
item_id	BIGINT	K
title	BIGINT	
cost	INT	
inventory_count	TIMESTAMP	



Denormalization for read speed (without joins)

Sensor data (time series)

Requirements

- Sensors can write different measures per second
- Write heavy (1m+ writes per second)

Sensor data (time series) - option 1

sensordata		
date	DATE	K
ts	TIMESTAMP	🔻
serial_number	TEXT	🔻
type	TEXT	🔻
value	TEXT	

Sensor data (time series) - option 1

sensordata		
date	DATE	K
ts	TIMESTAMP	🔻
serial_number	TEXT	🔻
type	TEXT	🔻
value	TEXT	

If the number of sensors will grow we will have a problem

Sensor data (time series) - option 2

sensordata		
date	DATE	K
serial_number	TEXT	K
ts	TIMESTAMP	🔻
type	TEXT	🔻
value	TEXT	

Sensor data (time series) - option 2

sensordata		
date	DATE	K
serial_number	TEXT	K
ts	TIMESTAMP	🔻
type	TEXT	🔻
value	TEXT	

Data is will always added after the previous data

- SSTable compactions can be optimized
  You can change the compaction strategies

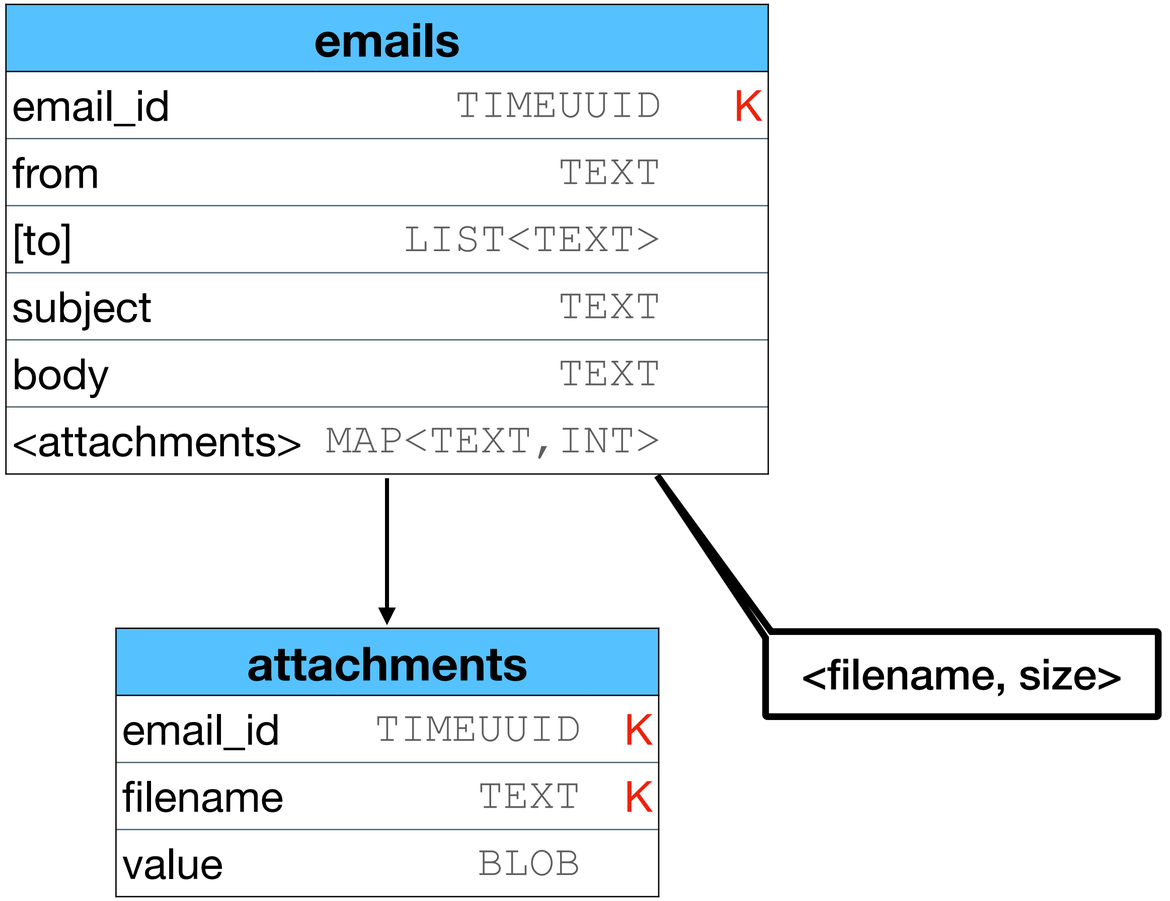
Gmail

Requirements

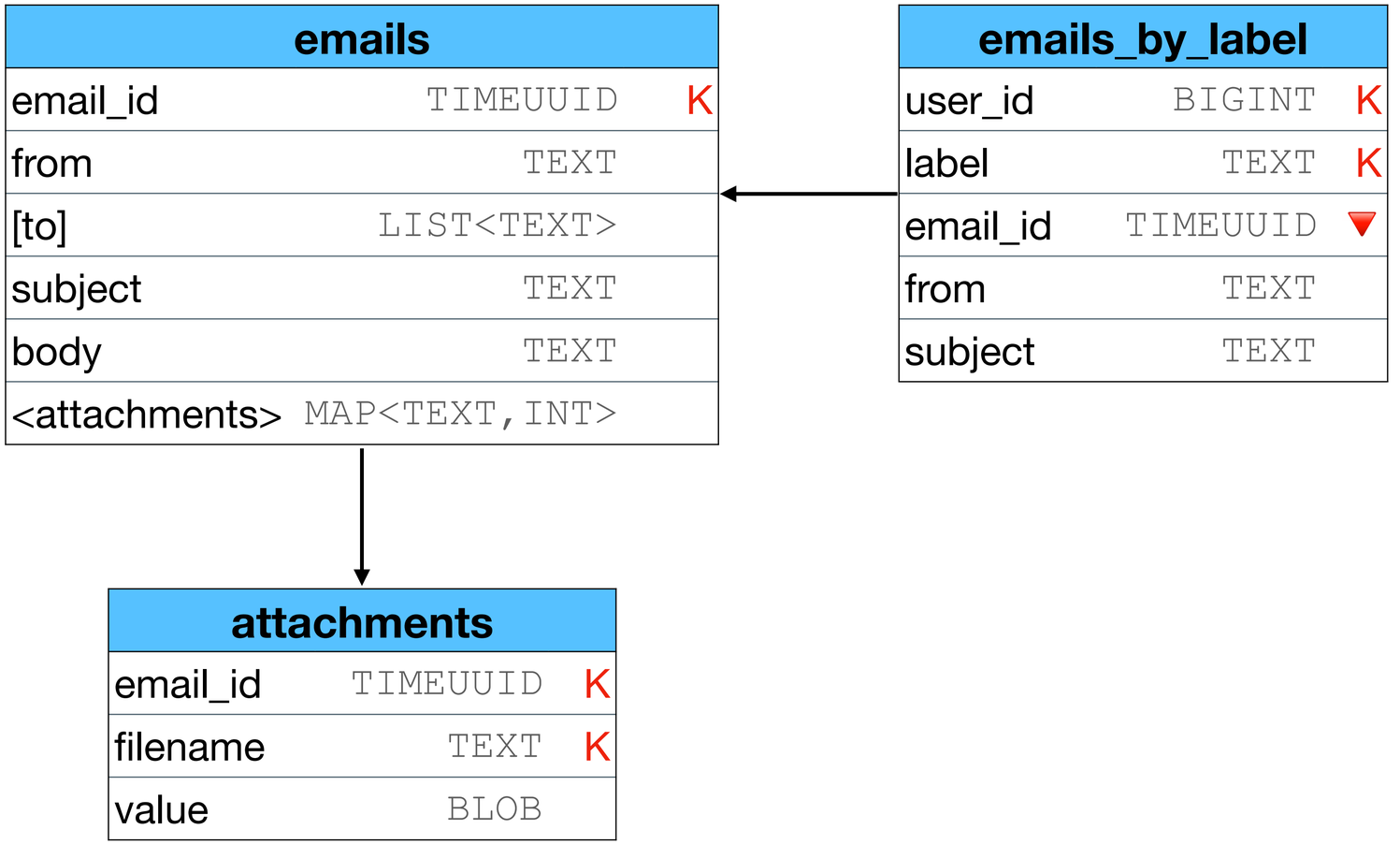
- Manage emails by labels
- Support attachments

*speculation

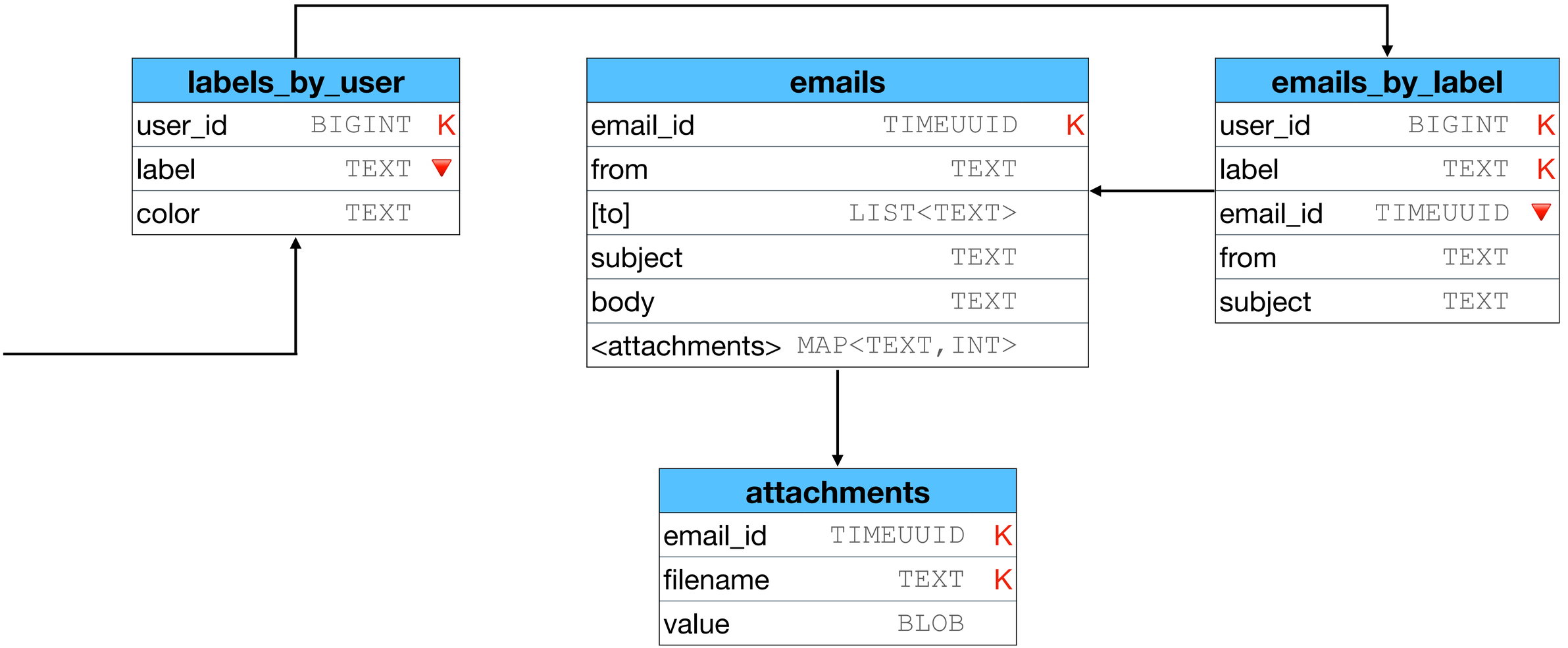
Gmail


emails
email_id TIMEUUID K
from TEXT
[to] LIST<TEXT>
subject TEXT
body TEXT
<attachments> MAP<TEXT,INT>
attachments
email_id TIMEUUID K
filename TEXT K
value BLOB
<filename, size>

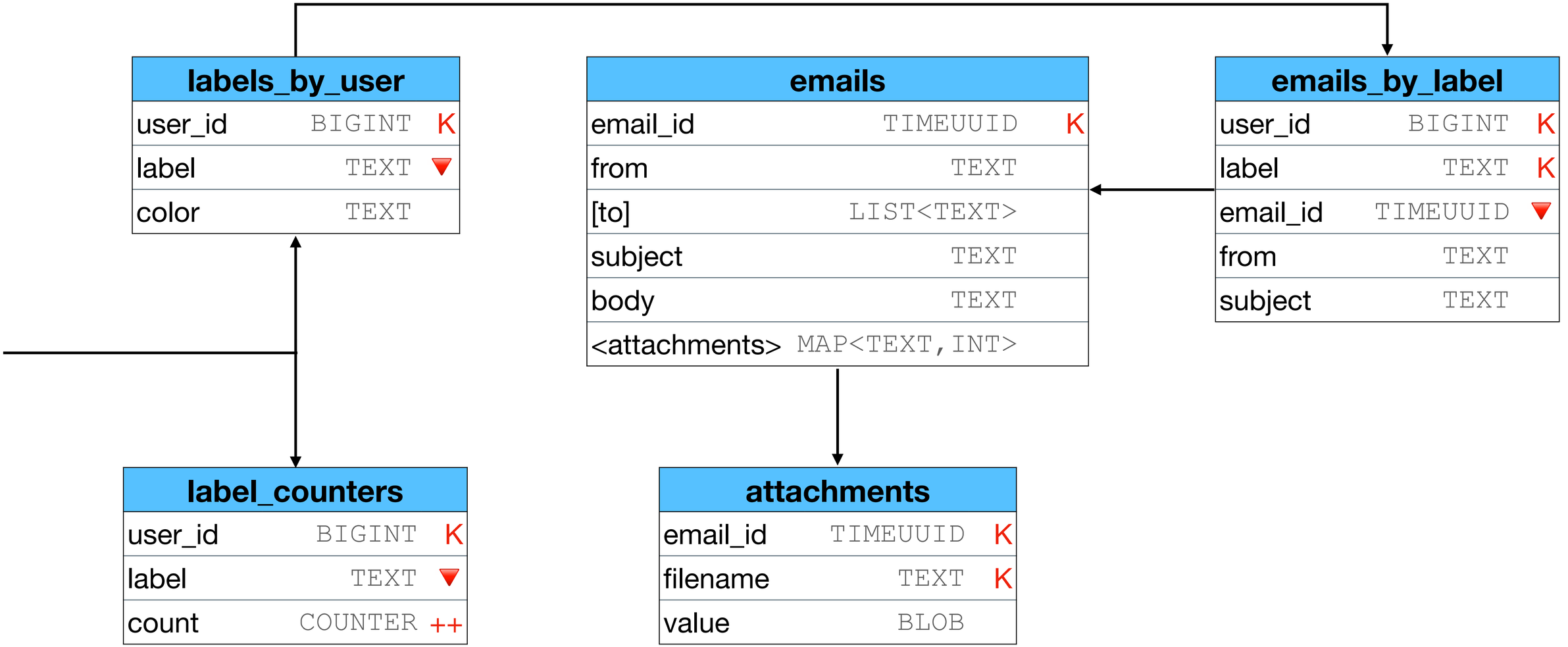
Gmail


emails
email_id TIMEUUID K
from TEXT
[to] LIST<TEXT>
subject TEXT
body TEXT
<attachments> MAP<TEXT,INT>
emails_by_label
user_id BIGINT K
label TEXT K
email_id TIMEUUID
from TEXT
subject TEXT
attachments
email_id TIMEUUID K
filename TEXT K
value BLOB

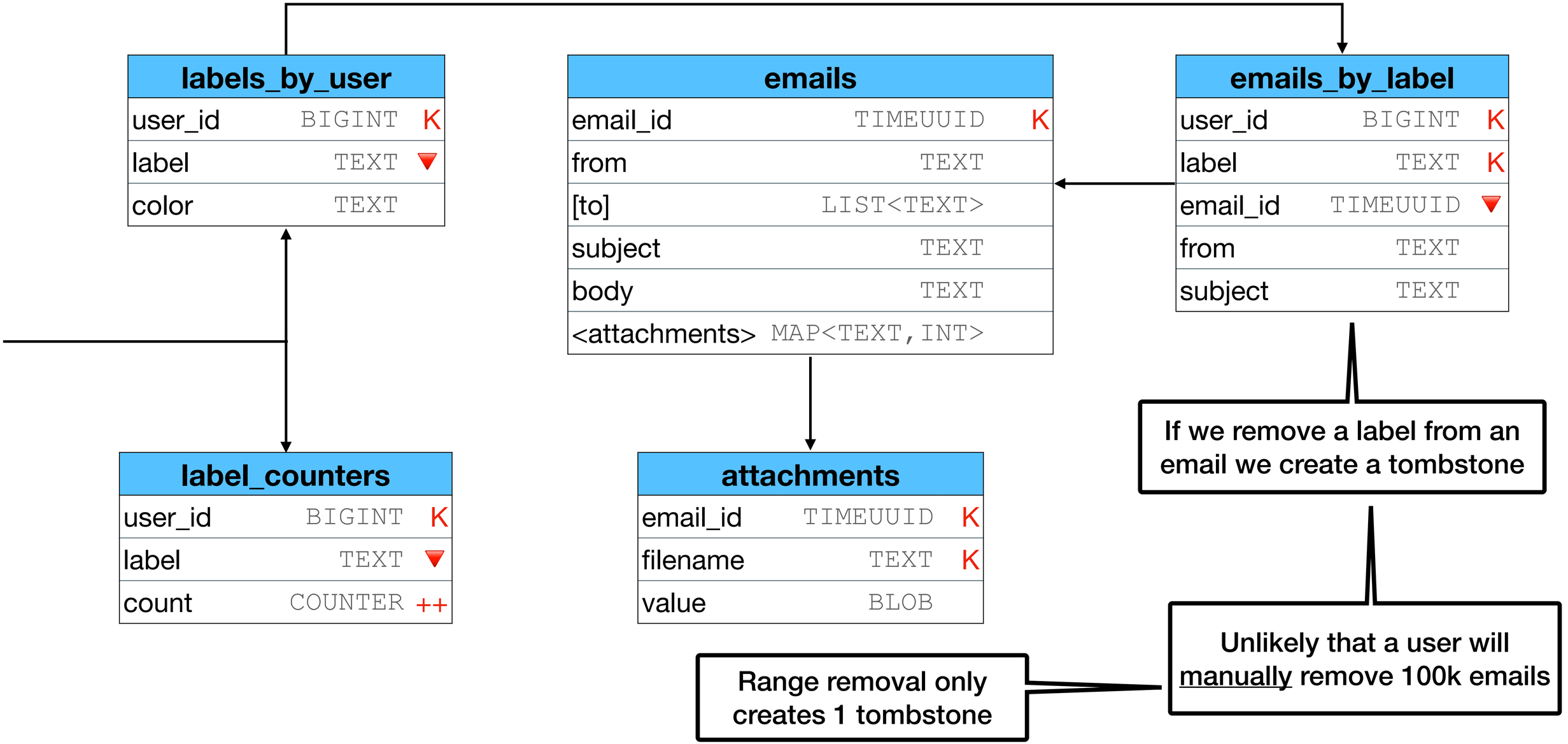
Gmail


labels_by_user
user_id BIGINT K
label TEXT
color TEXT
emails
email_id TIMEUUID K
from TEXT
[to] LIST<TEXT>
subject TEXT
body TEXT
<attachments> MAP<TEXT,INT>
emails_by_label
user_id BIGINT K
label TEXT K
email_id TIMEUUID
from TEXT
subject TEXT
attachments
email_id TIMEUUID K
filename TEXT K
value BLOB

Gmail


labels_by_user
user_id BIGINT K
label TEXT ▼
color TEXT
emails
email_id TIMEUUID K
from TEXT
[to] LIST<TEXT>
subject TEXT
body TEXT
<attachments> MAP<TEXT,INT>
emails_by_label
user_id BIGINT K
label TEXT K
email_id TIMEUUID ▼
from TEXT
subject TEXT
label_counters
user_id BIGINT K
label TEXT ▼
count COUNTER ++
attachments
email_id TIMEUUID K
filename TEXT K
value BLOB

Gmail


labels_by_user
user_id BIGINT K
label TEXT ▼
color TEXT
emails
email_id TIMEUUID K
from TEXT
[to] LIST<TEXT>
subject TEXT
body TEXT
<attachments> MAP<TEXT,INT>
emails_by_label
user_id BIGINT K
label TEXT K
email_id TIMEUUID ▼
from TEXT
subject TEXT
label_counters
user_id BIGINT K
label TEXT ▼
count COUNTER ++
attachments
email_id TIMEUUID K
filename TEXT K
value BLOB
If we remove a label from an email we create a tombstone
Unlikely that a user will manually remove 100k emails
Range removal only creates 1 tombstone

Instagram

Requirements (basic)

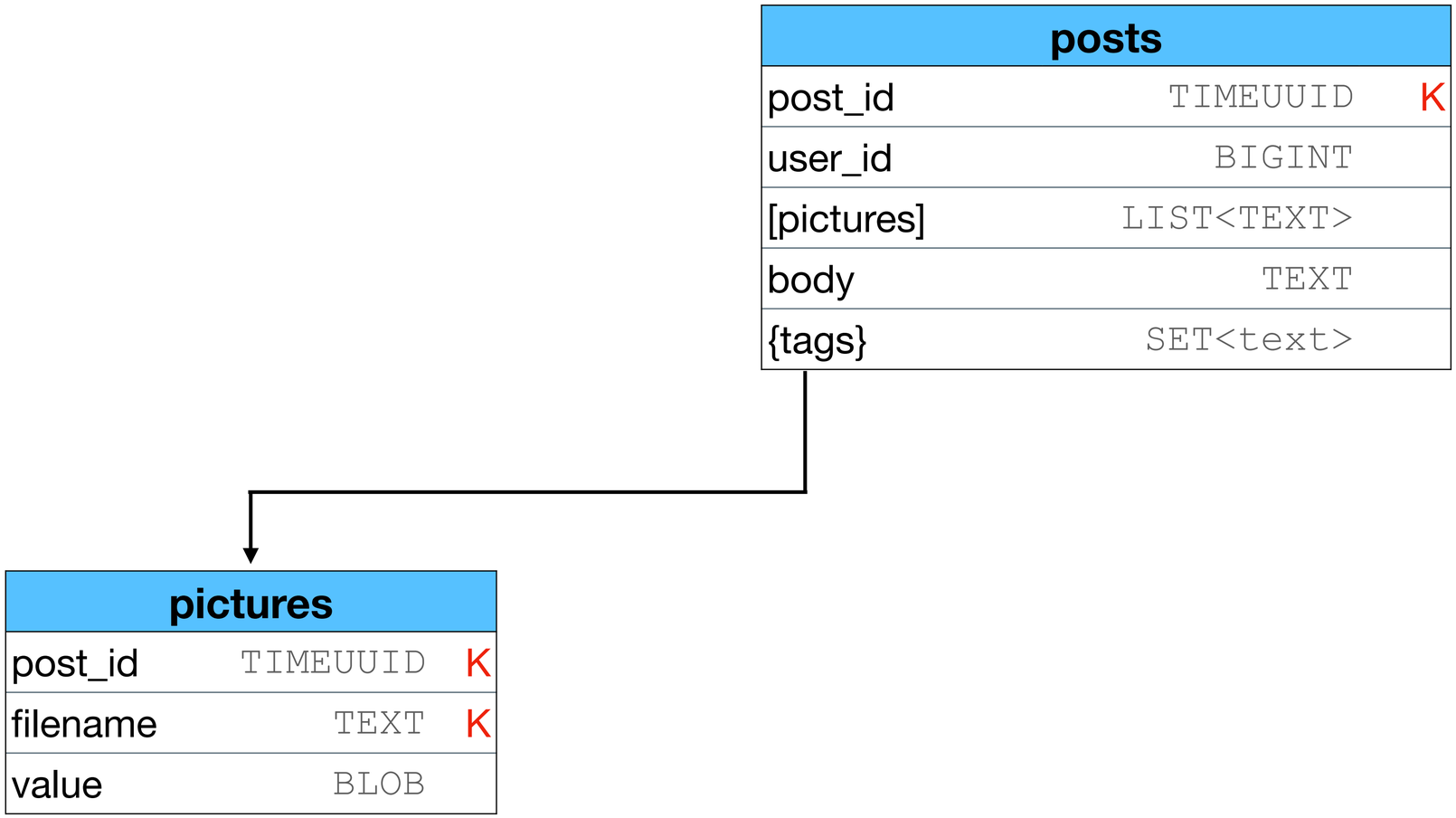
- Follow users
- Post
- Like, comment

*speculation

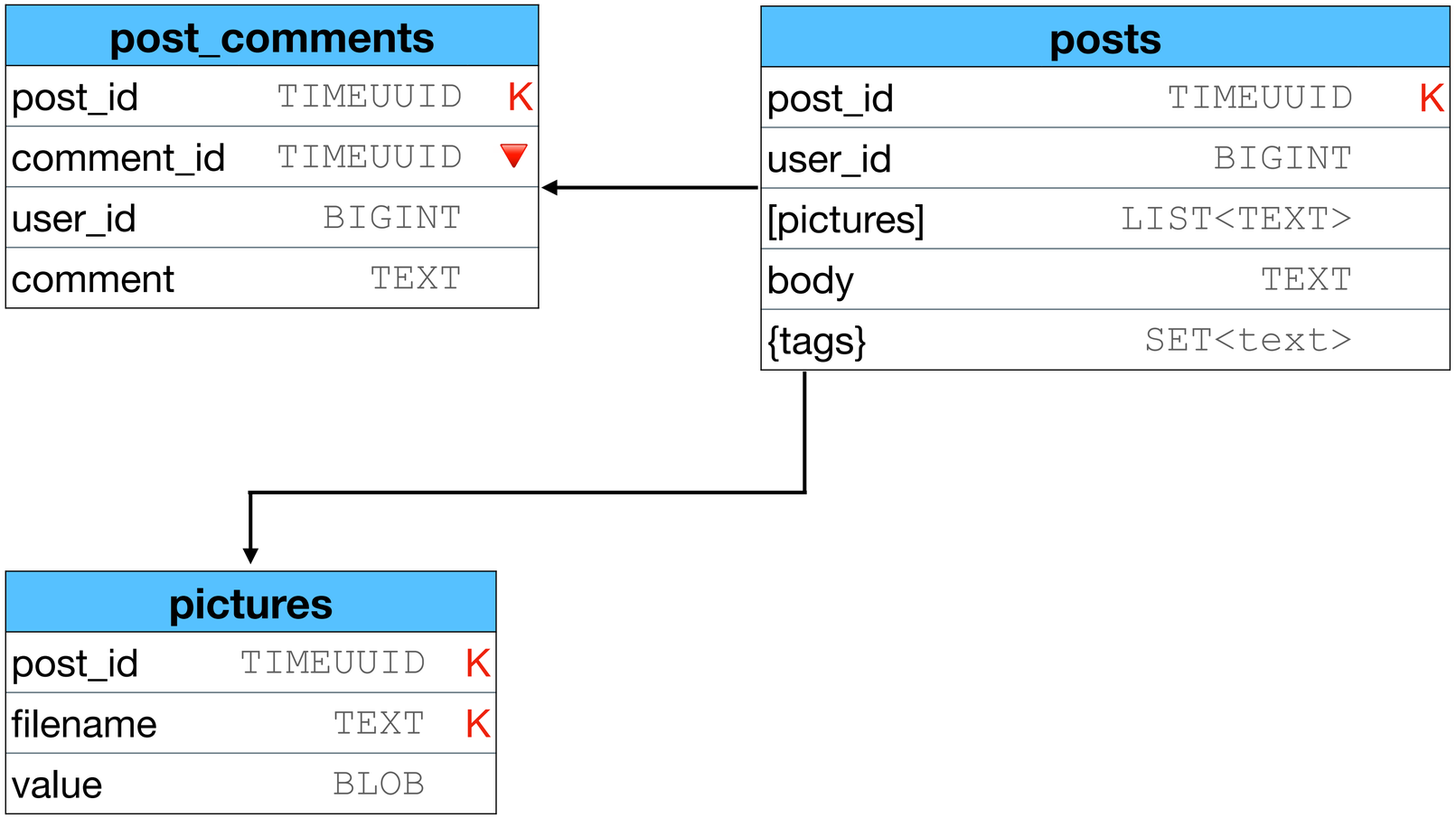
Instagram

posts		
post_id	TIMEUUID	K
user_id	BIGINT	
[pictures]	LIST<TEXT>	
body	TEXT	
{tags}	SET<text>	

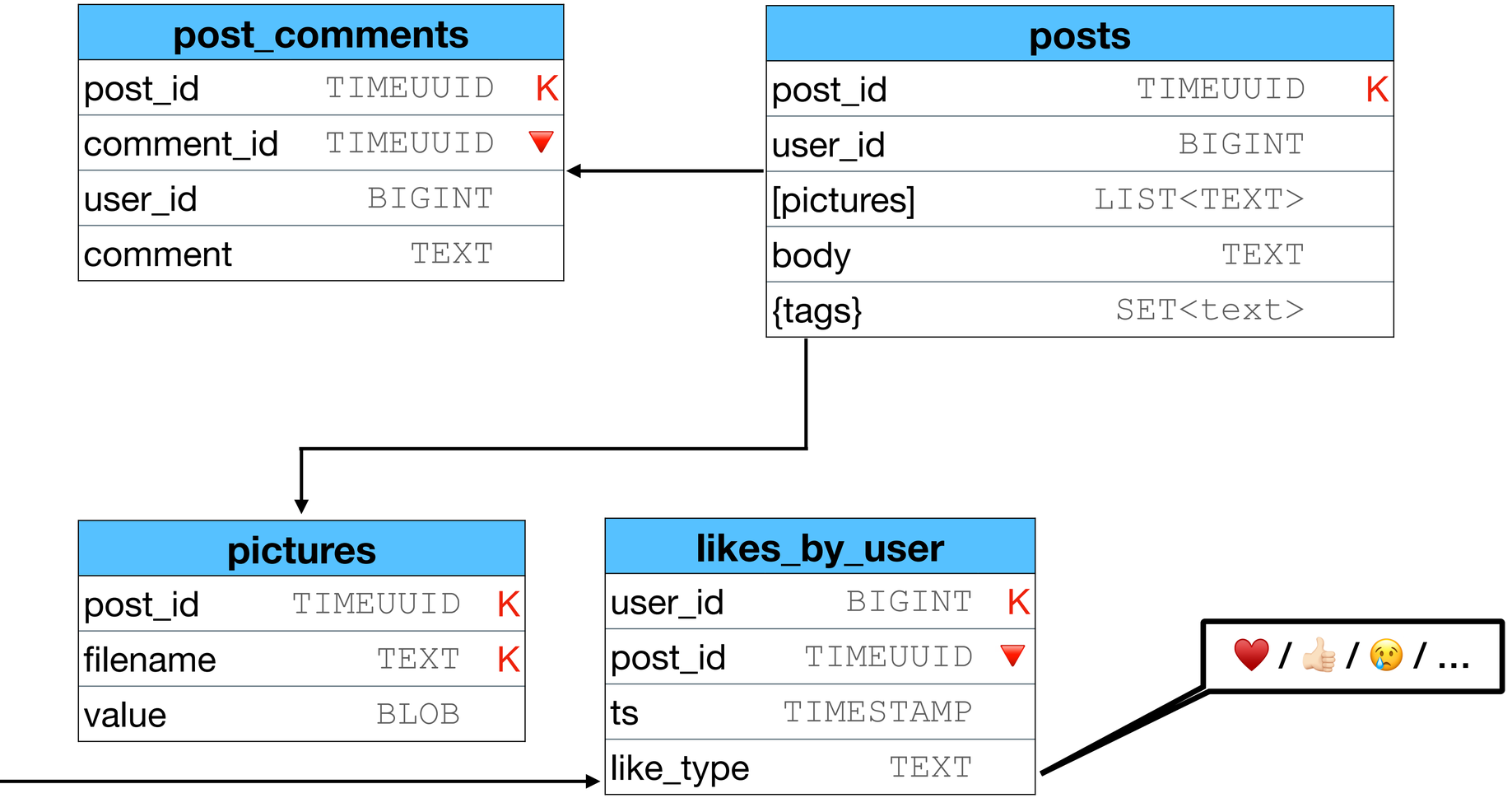
Instagram


posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB

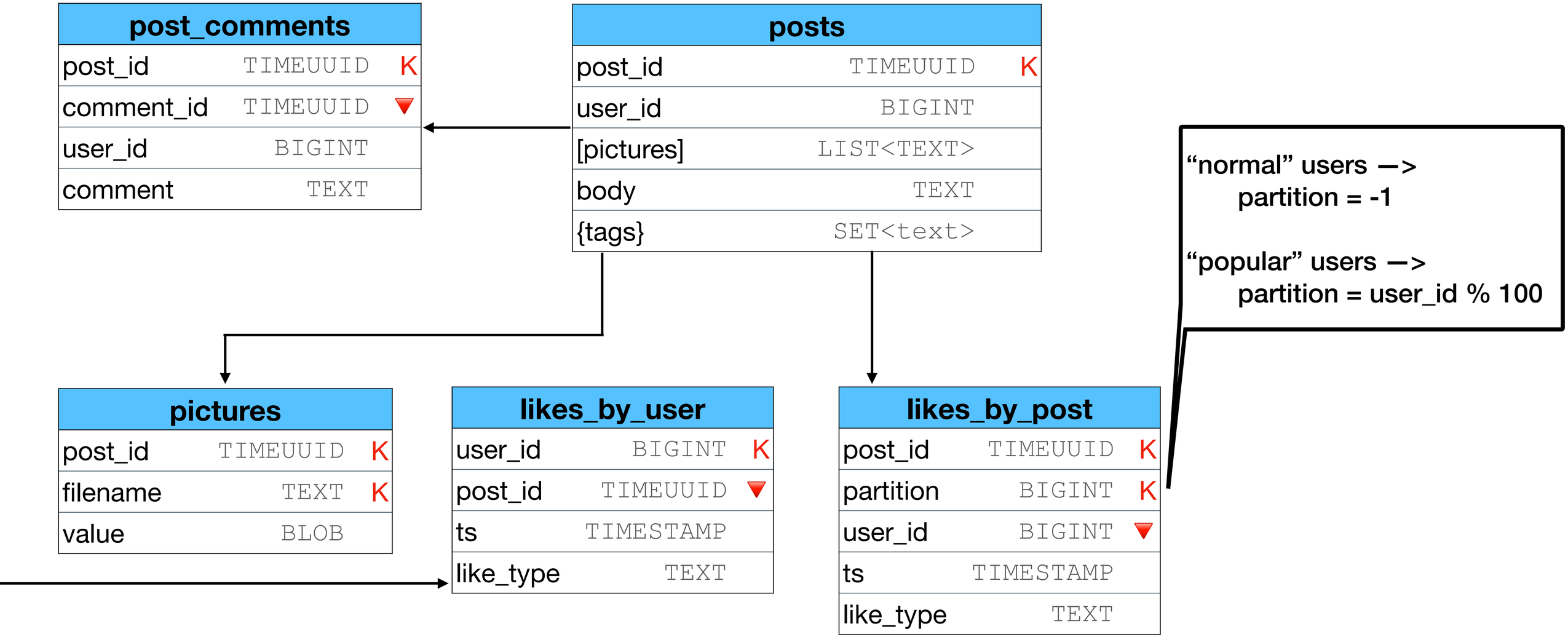
Instagram


post_comments
post_id TIMEUUID K
comment_id TIMEUUID
user_id BIGINT
comment TEXT
posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB

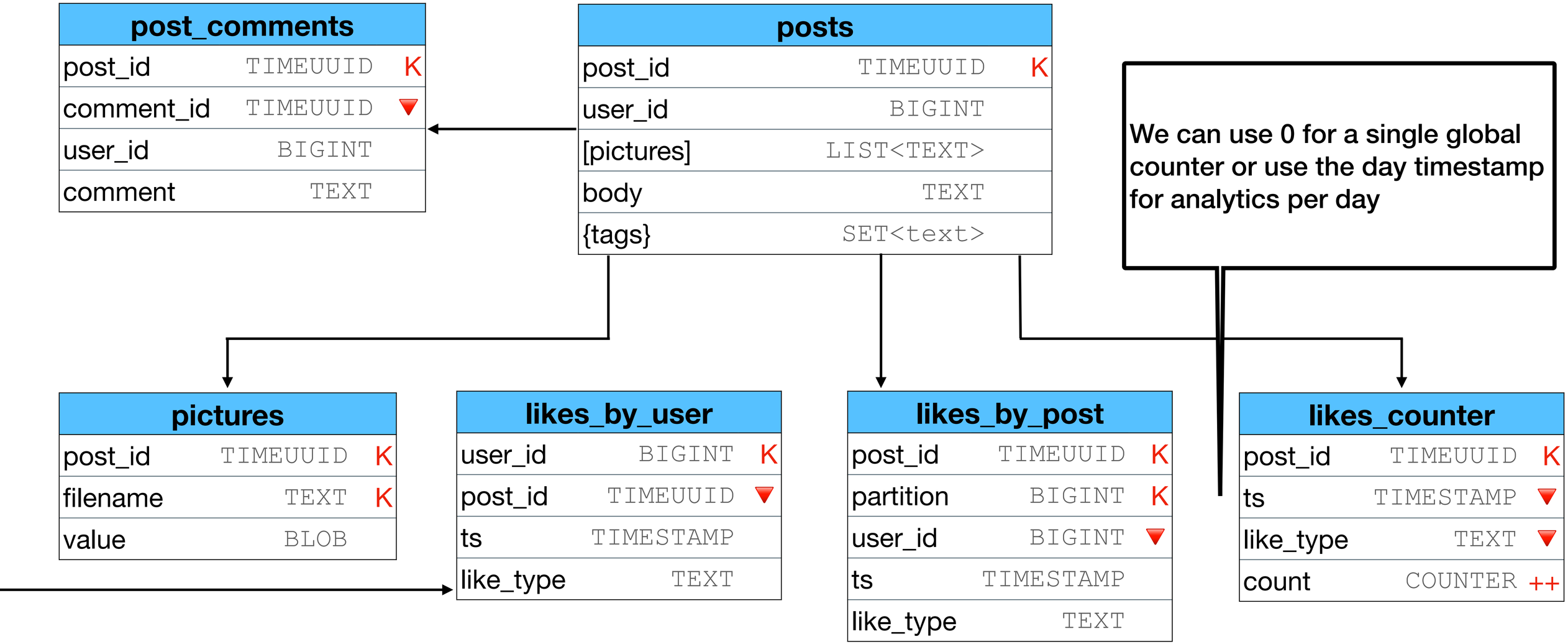
Instagram


post_comments
post_id TIMEUUID K
comment_id TIMEUUID 🔻
user_id BIGINT
comment TEXT
posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB
likes_by_user
user_id BIGINT K
post_id TIMEUUID 🔻
ts TIMESTAMP
like_type TEXT
❤️ / 👍🏻 / 😢 / ...

Instagram


post_comments
post_id TIMEUUID K
comment_id TIMEUUID
user_id BIGINT
comment TEXT
posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB
likes_by_user
user_id BIGINT K
post_id TIMEUUID
ts TIMESTAMP
like_type TEXT
likes_by_post
post_id TIMEUUID K
partition BIGINT K
user_id BIGINT
ts TIMESTAMP
like_type TEXT
“normal” users —>
partition = -1
“popular” users —>
partition = user_id % 100

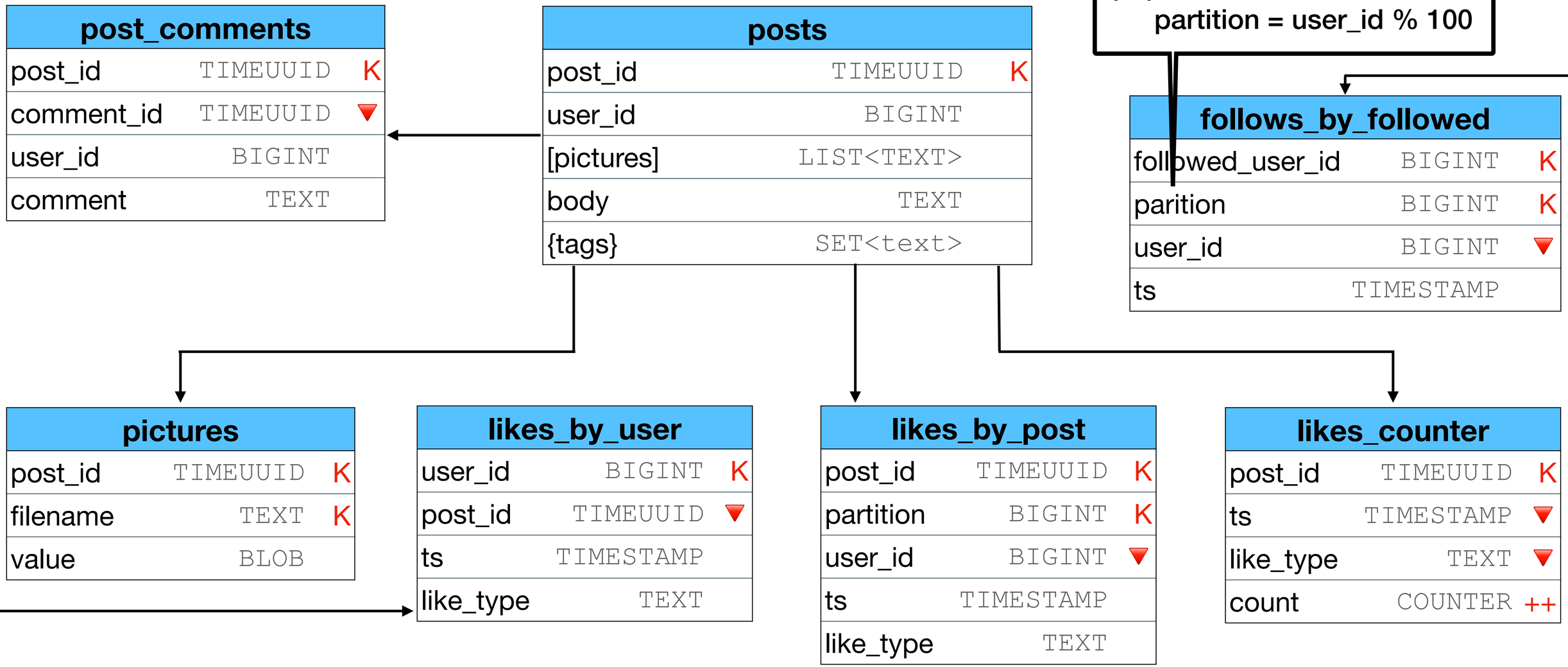
Instagram


post_comments
post_id TIMEUUID K
comment_id TIMEUUID ▼
user_id BIGINT
comment TEXT
posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
We can use 0 for a single global counter or use the day timestamp for analytics per day
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB
likes_by_user
user_id BIGINT K
post_id TIMEUUID ▼
ts TIMESTAMP
like_type TEXT
likes_by_post
post_id TIMEUUID K
partition BIGINT K
user_id BIGINT ▼
ts TIMESTAMP
like_type TEXT
likes_counter
post_id TIMEUUID K
ts TIMESTAMP ▼
like_type TEXT ▼
count COUNTER ++

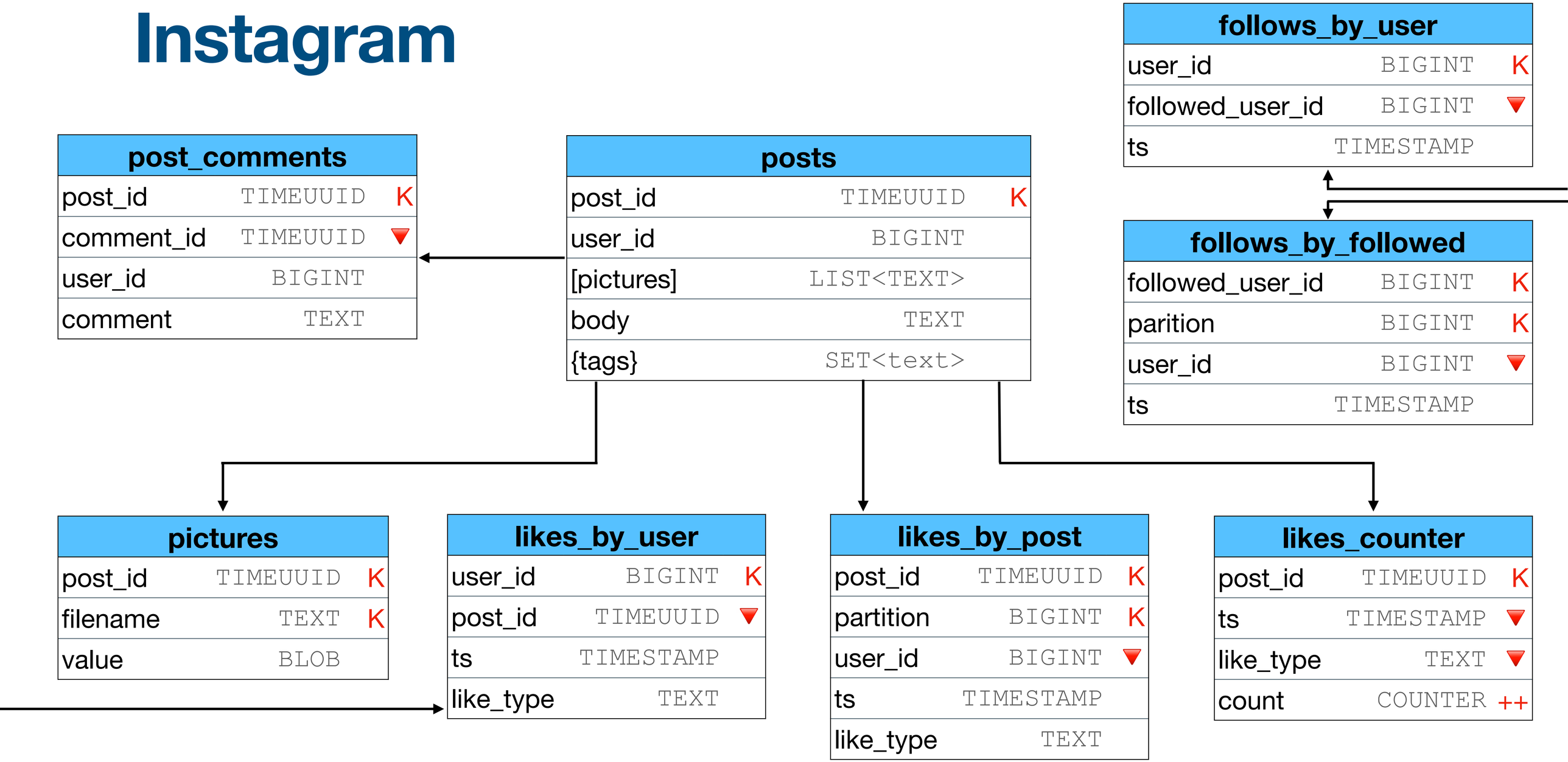
Instagram

same logic as likes_by_post
“normal” users —>
partition = -1

“popular” users —>
partition = user_id % 100



Instagram


follows_by_user
user_id BIGINT K
followed_user_id BIGINT
ts TIMESTAMP
follows_by_followed
followed_user_id BIGINT K
parition BIGINT K
user_id BIGINT
ts TIMESTAMP
post_comments
post_id TIMEUUID K
comment_id TIMEUUID
user_id BIGINT
comment TEXT
posts
post_id TIMEUUID K
user_id BIGINT
[pictures] LIST<TEXT>
body TEXT
{tags} SET<text>
pictures
post_id TIMEUUID K
filename TEXT K
value BLOB
likes_by_user
user_id BIGINT K
post_id TIMEUUID
ts TIMESTAMP
like_type TEXT
likes_by_post
post_id TIMEUUID K
partition BIGINT K
user_id BIGINT
ts TIMESTAMP
like_type TEXT
likes_counter
post_id TIMEUUID K
ts TIMESTAMP
like_type TEXT
count COUNTER ++