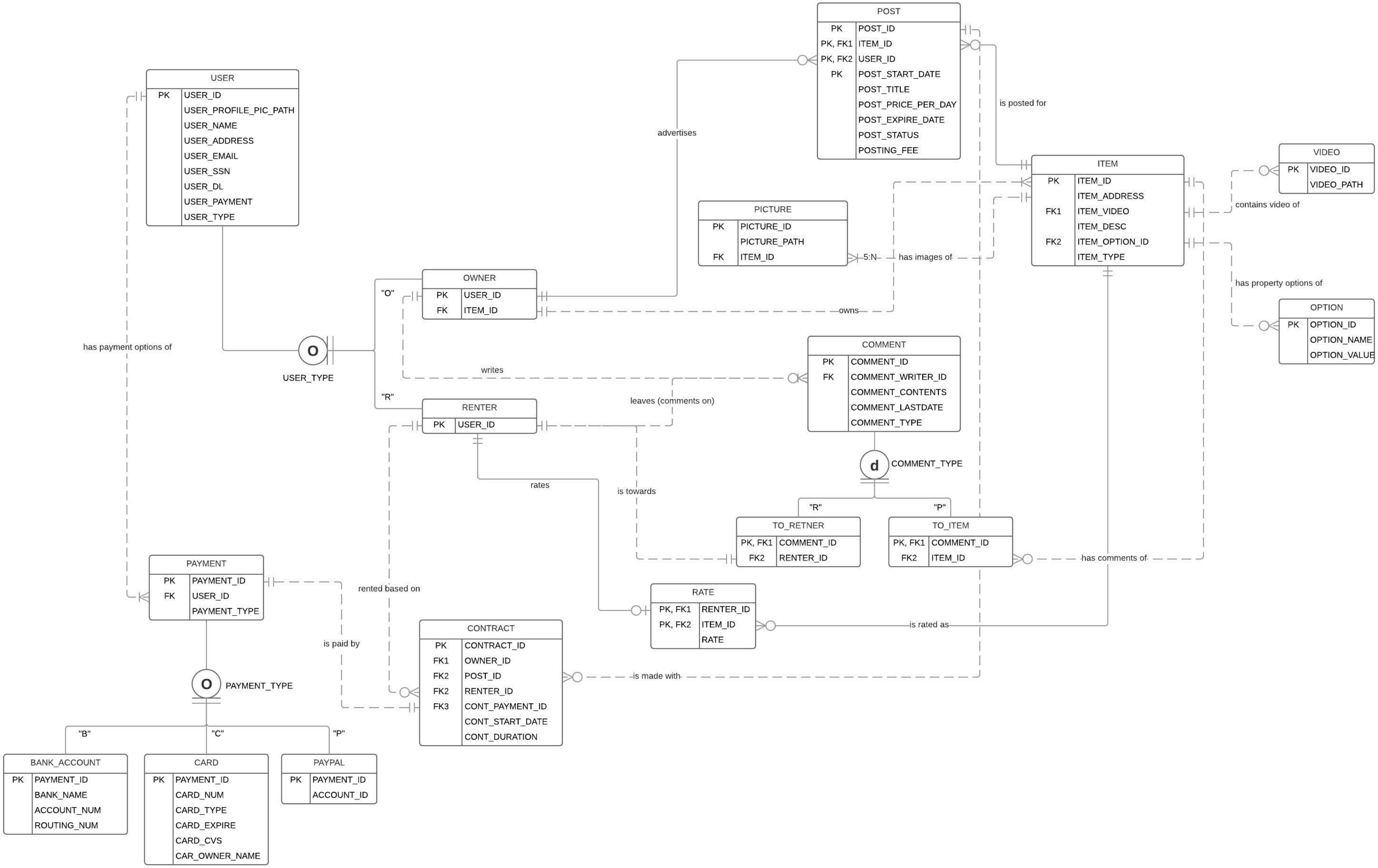
Chanshin Park – HW1

<ER Diagram>



USER	
PK	USER_ID
	USER_PROFILE_PIC_PATH
	USER_NAME
	USER_ADDRESS
	USER_EMAIL
	USER_SSN
	USER_DL
	USER_PAYMENT
	USER_TYPE

POST	
PK	POST_ID
PK, FK1	ITEM_ID
PK, FK2	USER_ID
PK	POST_START_DATE
	POST_TITLE
	POST_PRICE_PER_DAY
	POST_EXPIRE_DATE
	POST_STATUS
	POSTING_FEE

ITEM	
PK	ITEM_ID
	ITEM_ADDRESS
FK1	ITEM_VIDEO
	ITEM_DESC
FK2	ITEM_OPTION_ID
	ITEM_TYPE

VIDEO	
PK	VIDEO_ID
	VIDEO_PATH

OPTION	
PK	OPTION_ID
	OPTION_NAME
	OPTION_VALUE

PICTURE	
PK	PICTURE_ID
	PICTURE_PATH
FK	ITEM_ID

OWNER	
PK	USER_ID
FK	ITEM_ID

RENTER	
PK	USER_ID

COMMENT	
PK	COMMENT_ID
FK	COMMENT_WRITER_ID
	COMMENT_CONTENTS
	COMMENT_LASTDATE
	COMMENT_TYPE

TO_RETNER	
PK, FK1	COMMENT_ID
FK2	RENTER_ID

TO_ITEM	
PK, FK1	COMMENT_ID
FK2	ITEM_ID

RATE	
PK, FK1	RENTER_ID
PK, FK2	ITEM_ID
	RATE

PAYMENT	
PK	PAYMENT_ID
FK	USER_ID
	PAYMENT_TYPE

CONTRACT	
PK	CONTRACT_ID
FK1	OWNER_ID
FK2	POST_ID
FK2	RENTER_ID
FK3	CONT_PAYMENT_ID
	CONT_START_DATE
	CONT_DURATION

BANK_ACCOUNT	
PK	PAYMENT_ID
	BANK_NAME
	ACCOUNT_NUM
	ROUTING_NUM

CARD	
PK	PAYMENT_ID
	CARD_NUM
	CARD_TYPE
	CARD_EXPIRE
	CARD_CVS
	CAR_OWNER_NAME

PAYPAL	
PK	PAYMENT_ID
	ACCOUNT_ID

<Report>

- **Black** : HW statement / **Blue** : my description

A. Assumption

1. A single item(property) can have many posts based on its start_date(+duration)
   ex) POST #1 for APT A(5/1/2018 ~ 5/31/2018), POST #2 APT A(6/1/2018 ~ 7/1/2018)
2. A post can have multiple sublease contracts based on its contract period. (Also, a single post can have many renters based on multiple contracts)
3. The owner and renter cannot be 1:1, which means the owner can be renter at specific time and the renter can be owner when they buy item(property).

B. Entity Set

1. Entity Set List : USER, OWNER, RENTER, PAYMENT, BANK_ACCOUNT, CARD, PAYPAL, CONTRACT, COMMENT, TO_RENTER, TO_ITEM, RATE, ITEM, VIDEO, PICTURE, OPTION, POST

2. USER : fullname, address, email, profile photo, SSN, DL, payment options(at least one)
-> USER entity has USER_ID(UID), USER_NAME, USER_ADDRESS, USER_EMAIL, USER_PROFILE_PHOTO_PATH, USER_SSN, USER_DL, USER_PAYMENT attributes
-> There is no natural key that surrogate key(USER_ID) is required to be the primary key.
-> A user can be either an APT owner or a renter -> SUBTYPE : OWNER OR RENTER
-> USER has two subtypes, OWNER and RENTER, which has its own attribute (ITEM_ID for OWNER and CONTRACT_ID for RENTER).
The owner can be recognized by its own property(item) and the renter can be recognized by its contract with the post.
* **Trade-off** : OWNER-RENTER relationship can be expressed in RELATIONSHIP(OWNER ID, RENTER_ID) entity, but it cannot represent multiple tenants(different timeline) with the same owner and multiple renters.

3. ITEM : each item provides detail specific to a property: address, description -> ITEM entity includes ITEM_ADDRESS, ITEM_DESC
-> ITEM_ID is primary key, because there is no natural key for it

4. POST : provide the title, the available date to sublease, the price for rent per day
-> POST entity includes POST_TITLE, POST_START_DATE, POST_PRICE_PER_DAY, POST_EXPIRE_DATE, and POSTING_FEE
-> POST_ID+USER_ID+ITEM_ID+POST_START_DATE is the primary key, because there can be with multiple posts for one item(property) and a post with an item can have different period of posts(different start_date). And certain item(property) can have multiple owners in the same timeline or different time lines.
-> When a post expires(expire date), it becomes inactive(status)
: POST_STATUS attribute is the derived attribute calculated from POST_EXPIRE_DATE attribute. (difference between POST_EXPIRE_DATE and sysdate)
-> It can be postponed to application, but it will need updates in anytime POST occurrences are changed. (It can be huge burden if there are many posts)
-> When someone is interested in the post and makes payment, the available date of the property on the post is adjusted to reflect the remaining available date
:The application has to handle this based on POST(POST_EXPIRE_DATE - POST_START_DATE)

5. ITEM(property) : should be categorized based on its size : small(2-3 persons, medium (4-6 persons), large(7-10 persons)
-> ITEM_TYPE attribute is multivalued attribute. (Domain can be SMALL, MEDIUM, LARGE)
* **Trade-off** : SUBTYPE : SMALL, MEDIUM, LARGE -> It doesn't have to be SUBTYPE, because there isn't another unique attribute in each type, even though it is multivalued attribute
** The rest of entities are described in following relationship table.*

C. Relationship

NUM	ENTITY	RELATIONSHIP	CONNECTIVITY	ENTITY	Description
1	**USER**	has payment options of	**1:N**	**PAYMENT**	user may provide multiple payment options : some options are bank account, debit/credit, paypal (at least one) -> USER : PAYMENT = 1:N, because there has to be at least one payment option -> SUBTYPE by PAYMENT_TYPE : Each payment type has its own unique attributes. In order to remove NULLs in such unique attributes, make it as Overlapping(multiple payment options) and Total Completeness(every payment has to be one of PAYMENT_TYPE) subtype * **Trade-off** : it can be consisted of one entity(PAYMENT, without subtypes), but, then, it could have a lot of NULLs in each kind of payment type.
2	**ITEM**	has comments of	**0:N**	**TO_ITEM**	After staying at the apartment, he/she may leave a comment and rate the property. Only the ones who ever rented the property may make a comment on it. * **Trade-off** : COMMENT entity can have renter_id attribute, but, then, the comments for item can have NULL in it. In order to get rid of such case, specify COMMENT entity into two subtype entities(TO_RENTER, TO_ITEM). They have to be disjoint and total completeness. -> COMMENT entity and its subtype TO_ITEM entity -> The comments for items are only related to TO_ITEM subtype COMMENT occurrences
3	**ITEM**	is posted for	**0:N**	**POST**	The purpose of having items is to prevent describing the property every time the post for it is made. Each post is for an item. At a specific moment, there is at most one post for a property -> Multiple posts can be existed, because its advertising period can be different. However at specific moment, it should be one (Each POST_START_DATE should be within PK in the POST entity so that it could guarantee at least no duplicated start_date post can be existed) Before posting an item, ITEM occurrence can have no post for it.(0:N)
4	**ITEM**	has images of	**5:N**	**PICTURE**	Each item provides pictures(at least 5) -> cardinality is 5:N
5	**ITEM**	contains video of	**0:N**	**VIDEO**	Each item provides videos(optional) -> optional is 0:N
6	**ITEM**	has property options of	**0:N**	**OPTION**	each item provides various property options such as # of bedrooms, bathrooms, Wi-fi available, kitchen, pets allowed, etc -> options can be vary and unable to refine its number of occurrence so that it should be option_title and option_content **(MULTIVALUED ATTRIBUTES->A NEW ENTITY SET case)**
7	**ITEM**	is rated as	**0:N**	**RATE**	After staying at the apartment, he/she may leave a comment and rate the property -> Each item can have many rates from each renter and there exist a renter who doesn't want to rate it. * **Trade-off** : ITEM entity can have RATE attribute instead of a new entity(RATE), but RENTER and ITEM(RENT attribute) are M:N relationship. Therefore, separate RATE entity between ITEM entity and RENTER entity -> RENTER : ITEM = M:N => RENTER : RATE(1:M) + RATE : ITEM(N:1)

8	**OWNER**	owns	**1:N**	**ITEM**	Before making a post, a user must create items to describe his/her properties -> Only owner can owns items(properties), one or many
9	**OWNER**	advertises	**0:N**	**POST**	User may act as an APT owner by making posts to advertise his/her properties for sublease The user must pay a small fee to make a post and the fee depends on how long the post will be shown(active) in the system -> Each post has to have fee column which is paid by owner, posts can be published by one owner who has multiple properties or multiple posts with different advertising periods. -> There is no restrictions of how many posts or payments a user may make -> The user has the option to re-post it. -> Owner can choose the previous post based on POST_ID+START_DATE+ITEM_ID (PK of POST) * **Trade-off** : OWNER : ITEM(POSTING) = M:N, because the owner can be renter and owner and owner can post multiple posts with a certain item. Therefore, a new composite entity(POST) is necessary for removing M:N -> 1:M. And those relationships are strong relationship.
10	**OWNER**	writes	**0:N**	**COMMENT**	On the other hand, a property owner can also make comments on a user who ever rented his properties -> Only OWNER(specified/subtype of USER) can make comments to user
11	**TO_RENTER**	is towards	**1:1**	**RENTER**	On the other hand, a property owner can also make comments on a user who ever rented his properties -> Only in TO_RENTER comment occurrence case, it can have RENTER(renter_id) information.
12	**RENTER**	leaves (comment on)	**0:N**	**COMMENT**	After staying at the apartment, he/she may leave a comment and rate the property. Only the ones who ever rented the property may make a comment on it. -> RENTER can leave comments for the item which they had been contracted before
13	**RENTER**	rates	**0:1**	**RATE**	After staying at the apartment, he/she may leave a comment and rate the property -> Only RENTER(subtype of USER) can rate the item.
14	**RENTER**	rented based on	**0:N**	**CONTRACT**	A user may act as a renter by making a payment to a post * **Trade-off** : RENTER : POST = M:N => RENTER : CONTRACT (1:M) and CONTRACT : POST (N:1) * **Design Decision** : Therefore, it should be strong relationship. However, I make it remain weak relationship with defining a new primary key(CONTRACT_ID) instead, because there could be cancellation or some other kinds of tons of transactions can happen in future with this entity. So, RENTER-CONTRACT-POST are weak relationship even though CONTRACT is composite entity.
15	**CONTRACT**	is paid by	**1:1**	**PAYMENT**	A user may act as a renter by making a payment to a post He/she must pay the rent immediately using one of his payment options -> Every CONTRACT(with POST) occurrence should have a payment record. -> CONTRACT : PAYMENT = 1:1 (RENTER_ID, PAYMENT_ID)
16	**POST**	is made with	**0:N**	**CONTRACT**	A user may act as a renter by making a payment to a post -> POST can have many sublease CONTRACTs based on their different rental period that their relationship is 0:N