2019년 2학기 데이터 관리와 분석
박종헌 교수님

Project #1

Conceptual DB Design

7조
2018-11785 권수민
2018-12530 박소정
2015-18381 송상목
2014-12615 조현수

목차

1. 문제 정의

사이트 A는 가입자들에 대한 정보 관심사, 그리고 특정 관심사를 공유하는 모임들에 대한 정보를 저장하고 이를 가입자들에게 제공한다. 따라서 사이트가 이용하는 DB에 대한 ER diagram과 관계 스키마를 Redundancy를 최소화하는 방향으로 도식화하려고 한다. 사이트 A에 대한 Requirement는 (R1)-(R12)로 제시되어 있으며, 이는 8개의 entity와 이들 간의 Relationship을 통해 설명할 수 있다.

첫 번째로, 도시가 있다. 사이트 A에는 각 도시의 정보가 저장되어 있어야 하므로 entity로 표현하였다. 도시는 또한 해당 도시에 활동 중인 사용자들의 정보를 저장하고 있다.

두 번째로는 사용자가 있다. 사용자는 본인이 활동 중인 도시를 하나 설정한다. 이들은 특정 주제에 관심 있는 그룹에 속하게 되며, 이들 중 한 명이 해당 그룹의 그룹장이 된다. 또한, 사용자들은 자기 자신들끼리 '좋아요', '팔로우', '친구'라는 세 가지 방식으로 서로 교류한다. 사용자는 이용자뿐만 아니라 그룹에 대해서도 '좋아요'를 수행한다.

리더는 사용자 중 한 명이 특정 그룹의 리더로 임명된다. 이때, 리더 고유의 id가 attribute로 필요하기에 entity로 표현하였다.

그룹은 특정한 주제 하나에 관심을 갖는다. 그룹은 특정 장소에서 모임을 가지며, 해당 그룹에 속해있는 사용자들의 정보를 저장한다. 따라서 사용자, 주제, 모임, 장소와 Relationship을 갖는다. 또한, 가입자는 그룹에 대해 '좋아요'를 남기며, 그룹은 해당 가입자의 id를 저장한다.

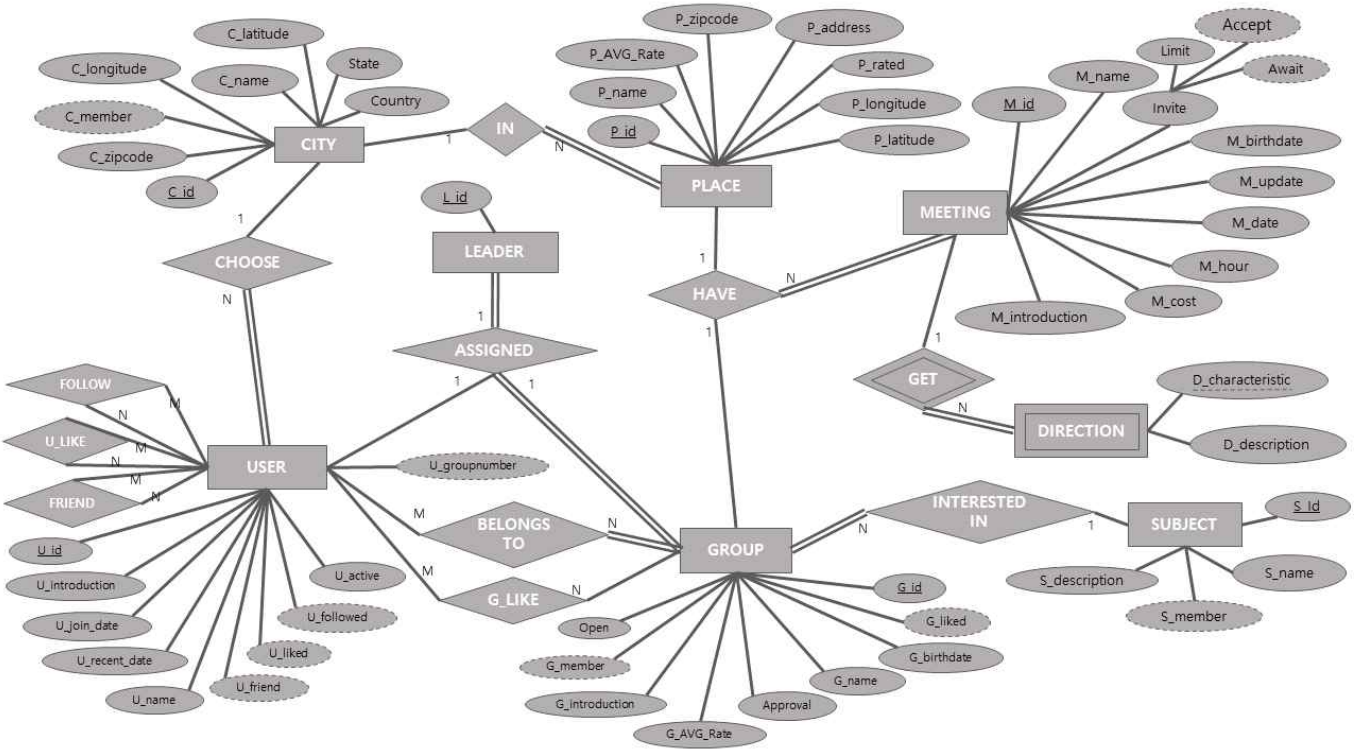
주제는 그룹이 관심을 갖는 하나의 주제이다. 주제에 대한 이름, 설명 등을 저장하고, 추가적으로 해당 그룹에 속한 멤버에 대한 정보를 저장한다.

다음으로 모임이 있다. 모임은 그룹이 개최하는 이벤트로써, 행사와 관련된 시간 정보를 포함한다. 또한, 모임은 개최되는 '장소'와 '찾아오는 방법'에 대한 정보를 추가적으로 필요로 한다. 따라서 장소와 찾아오는 법의 entity와 Relationship을 갖는다.

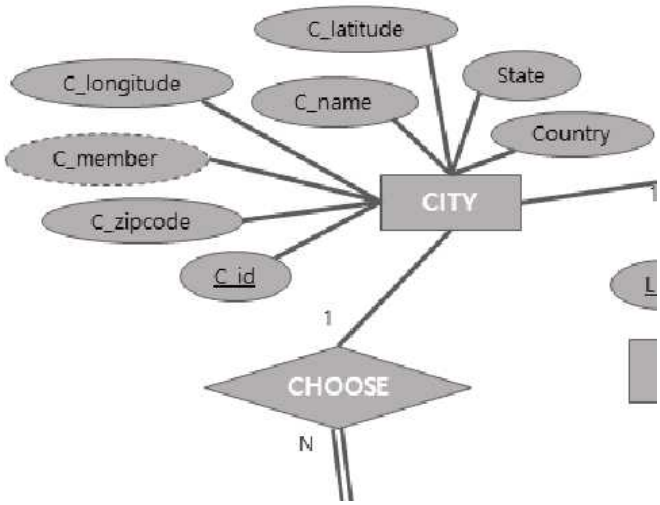
장소는 해당 장소가 속해있는 도시의 정보를 필요로 하기에 도시와 Relationship을 갖는다. 또한 장소는 그룹과 모임, 이 두 가지 entity와 동시에 Relationship에 참여한다.

마지막으로 찾아오는 방법이 있다. 이는 고유값이 존재하는 primary key가 별도로 존재하지 않기에 weak entity로 표현한다. 찾아오는 방법은 모임에 대한 구체적 정보이므로 모임의 id를 저장한다.

2. ER 다이어그램



(R1) 도시 정보

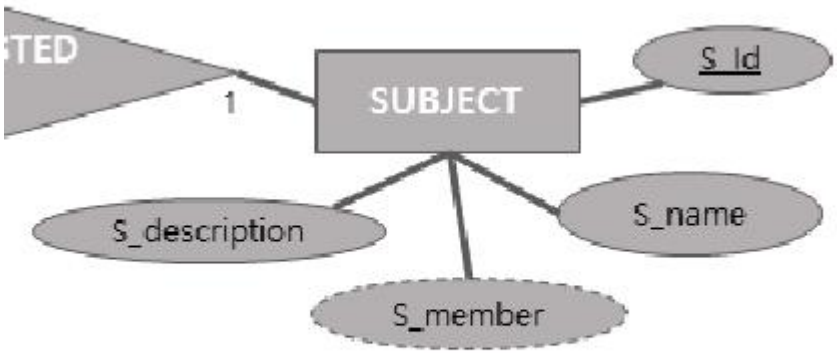


사이트 A는 도시의 정보를 가지고 있다. 이 정보는 **CITY**라는 entity를 만들어 저장된다. **CITY**의 attribute는 Requirement의 조건과 동일하게 고유의 id(C_id), 도시 이름(C_name), 주(State), 나라(Country), 우편번호(C_zipcode), 위도(latitude), 경도(longitude), 멤버 수(C_member)가 있다.

이때 C_id는 고유값이므로 primary key이다.

C_member는 해당 도시에서 활동 중인 사용자의 수이다. 이는 (R4)에서 언급되는 CHOOSE Relationship을 통해, 해당 USER의 U-id를 count하여 구할 수 있다.

(R2) 주제 정보

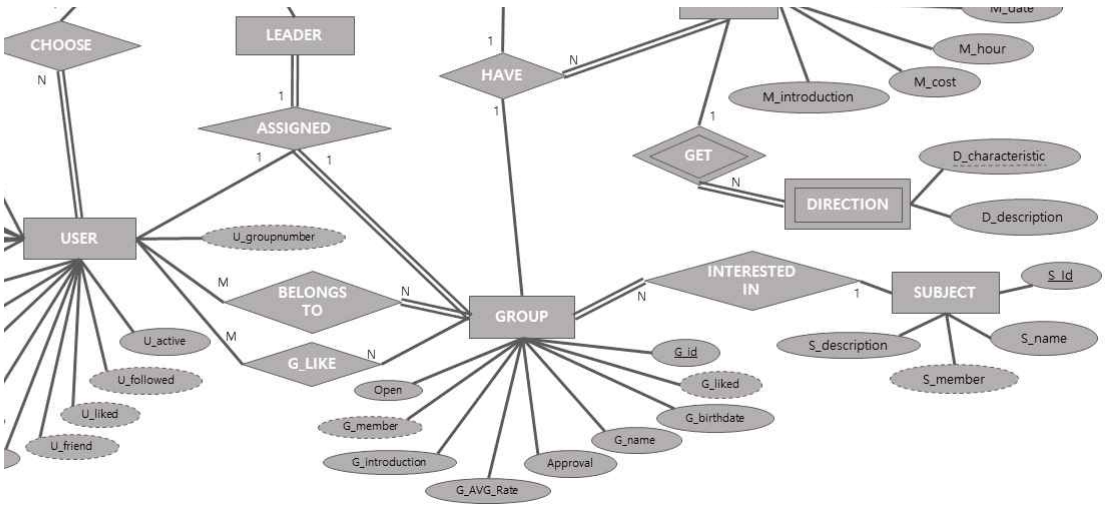


사이트 A는 주제에 대한 정보를 가지고 있다. 이 정보는 **SUBJECT**라는 entity를 만들어 저장된다. **SUBJECT**의 attribute는 Requirement의 조건과 동일하게 고유의 id(S_id), 주제 이름(S_name), 주제에 대한 설명(S_description), 멤버 수(S_member)가 저장되어 있다.

이때 S_id는 고유값이므로 primary key이다.

S_member는 (R3)에서 언급되는 **INTERESTED IN** Relationship을 통해, 해당 Group에서 G-member를 count하여 구할 수 있다.

(R3) 그룹 정보



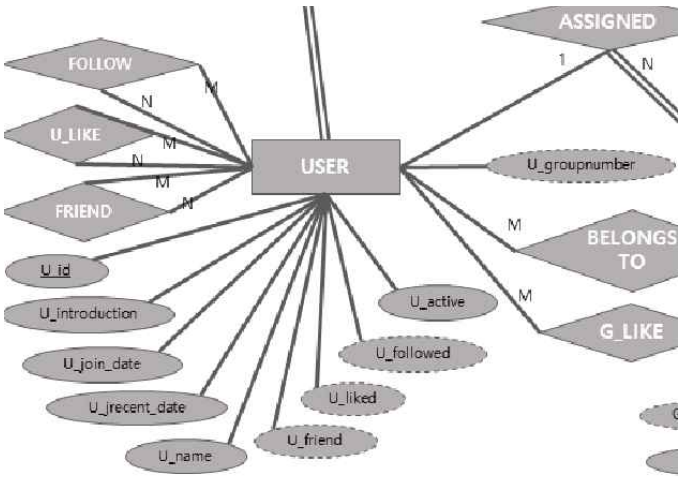
사이트 A는 그룹에 대한 정보를 가지고 있다. 이 정보는 **GROUP**이라는 entity를 만들어 저장된다. **GROUP**의 attribute는 고유의 id(G_id), 그룹 이름(G_name), 그룹 생성일(G_birthdate), 그룹 소개글(G_introduction), 평균 평점(G_AVG_Rate), 공개 여부(Open), 승인 필요 여부(Approval), 다른 가입자들에게서 받은 '좋아요'의 수(G_liked), 멤버 수(G_member)가 있다. 주제의 id()는 **INTERESTED IN** Relationship을 통해 **SUBJECT** entity의 S_id attribute를 조회할 수 있다.

이때 G_id는 고유의 값을 가지므로, **GROUP** entity의 primary key이다.

모든 **GROUP**은 각각 하나의 **SUBJECT**에 대한 정보를 갖기 때문에, **GROUP**과 **SUBJECT** entity는 **INTERESTED IN** Relationship을 통해 구현된다. 이때 하나의 **GROUP**에 대해서 하나의 **SUBJECT**만 선택할 수 있지만, 하나의 **SUBJECT**에 대해서는 여러 **GROUP**이 포함될 수 있다. 따라서 **SUBJECT**와 **GROUP**의 cardinality는 각각 1:N이다. 또한, 모든 **GROUP**은 특정 **SUBJECT**에 대한 정보를 가지고 있어야하며, **SUBJECT**는 반드시 **GROUP**에 속할 필요가 없으므로 **SUBJECT**와 **GROUP** entity는 partial total로 참여한다.

G_member는 **BELONGS TO** Relationship을 통해 해당 **USER** entity의 U_id를 count하여 구할 수 있다.

(R4) 가입자 정보



사이트 A는 가입자들에 대한 정보를 가지고 있다. 이 정보는 **USER**라는 entity를 만들어 저장된다. **USER**의 attribute는 Requirement의 조건과 동일하게 고유의 id(U_id), 가입자 이름(U_name), 가입 날짜(U_join_date), 최근 접속일(U_jrecent_date), 자기 소개글(U_introduction), 활동 여부(U_active), 가입자에게 '좋아요'를 남긴 가입자의 수(U_liked), 해당 가입자를 팔로우하는 가입자의 수(U_followed), 가입자의 친구의 수(U_friend), 가입자가 속한 그룹의 수(U_groupnumber)가 저장되어 있다. 모든 그룹의 id(G_id)는 **BELONGS TO** Relationship을 통해, **GROUP** entity의 G_id를 조회할 수 있다.

U_id는 고유의 값을 가지므로, **USER** entity의 primary key이다.

U_liked, U_followed, U_friend는 추후 (R11)에서 논의한다.
가입자가 속한 그룹의 수(U_groupnumber)는 **BELONGS TO** Relationship을 통해, 해당 **GROUP** entity의 G_id를 count하여 구할 수 있다.
활동 중인 도시에 대한 id는 **CHOOSE** Relationship을 통해, 해당 **CITY**의 id를 조회할 수 있다.

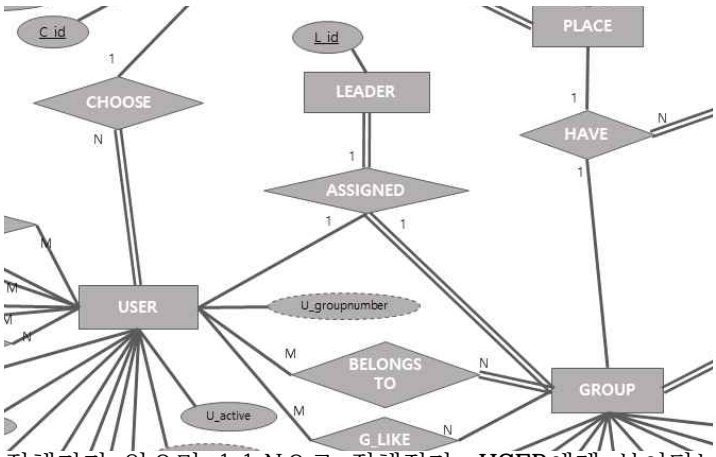
한 명의 **USER**는 활동 중인 **CITY**를 하나만 설정할 수 있으므로, **USER**와 **CITY** 사이의 관계를 **CHOOSE** Relationship을 구현하였다. 이때 한 명의 **USER**에 대하여 활동 중인 **CITY**를 하나만 설정할 수 있지만, 하나의 **CITY**는 여러 **USER**를 포함할 수 있다. 따라서 **CITY**와 **USER** entity의 cardinality는 1:N이다. 또한, **USER**가 도시를 설정해야만 사이트의 시스템이 구현가능하므로 **USER** entity는 total로 참여한다.

(R5) 그룹장 정보

그룹장(**LEADER**)은 가입자 중에서 임명되며, 그룹을 대표하는 역할을 맡는다. 그룹장의 attribute로는 사이트에서 부여받은 고유의 그룹장 id(L_id)가 있다. **USER**와 **GROUP** entity에 대한 설명은 각각 (R3)과 (R4)에서 언급하였다.

L_id는 **LEADER**의 primary key이다.

LEADER은 **USER**, **GROUP** entity와 함께 ternary relationship을 이룬다. ternary relationship은 두 entity pair가 결정되었을 때 나머지 entity가 하나로 결정이 되면 1:1:1, 정해지지 않으면 1:1:N으로 정해진다. **USER**에게 부여되는 고유의 id인 U_id와 별개로 그룹의 그룹장이 정해질 경우 그룹장의 id인 G_id가 부여된다.



L_id	U_id	G_id
Leader1	user2	group1
Leader2	user2	group2
Leader3	user2	group3
Leader4	user3	group4

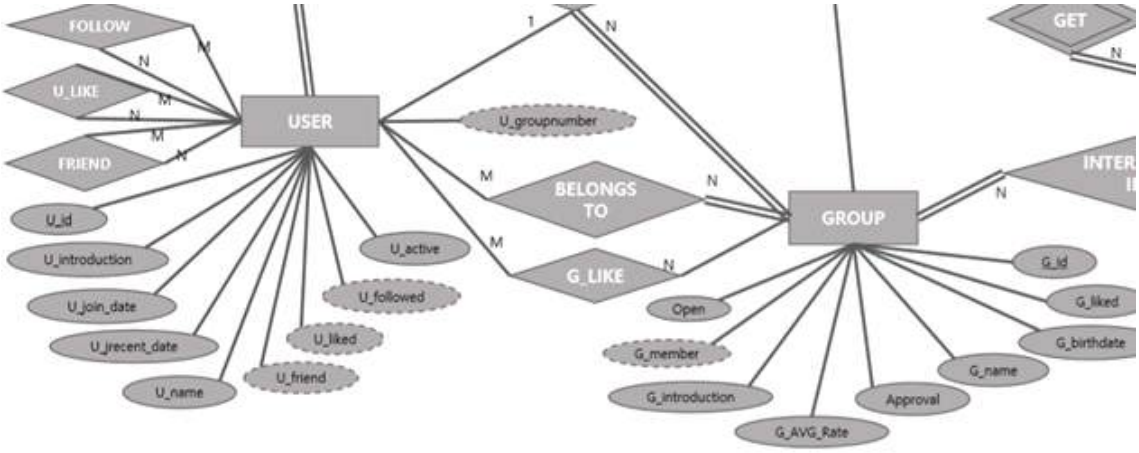
Leader5	user3	group5
Leader6	user4	group6

위의 표가 세 id가 부여되는 방식의 예시이다.

따라서 **LEADER**과 **USER**의 pair가 생성되었을 때 그에 대응되는 **GROUP**은 하나밖에 존재하지 않으므로 이 ternary relationship의 cardinality는 1:1:1이 된다.

그리고 **LEADER**가 아닌 **USER**는 존재할 수 있지만 **USER**가 아닌 **LEADER**는 존재할 수 없으므로 **LEADER**는 total, **USER**는 partial로 relationship에 참여한다. **GROUP**은 반드시 **LEADER**가 존재해야 하며 **LEADER** 역시 적어도 하나 이상의 **GROUP** 이상의 그룹장을 맡아야 하므로 **LEADER**과 **GROUP**은 모두 total로 참여한다.

(R6) 그룹 좋아요 정보



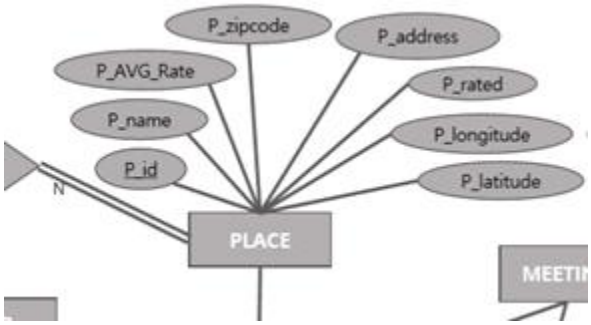
가입자들은 그룹에 '좋아요'를 남길 수 있다. 이를 **USER**와 **GROUP** 사이의 **G_LIKE** relationship으로 구현하였다.

한 그룹에 대해 좋아요를 남긴 사람이 여러 명일 수 있고, 한 사람이 여러 그룹에 대해 좋아요를 남길 수 있으므로 **G_LIKE** relationship에서 **USER**와 **GROUP**의 cardinality는 M:N이고, 좋아요를 남기지 않는 가입자와 좋아요를 받지 못한 그룹이 존재할 수 있으므로 둘 다 partial로 참여한다.

한 명의 가입자가 하나의 그룹에 대해 단 한 번만 '좋아요'를 누를 수 있다는 조건은 추후에 constraint를 통해 구현한다.

(R7) 만남 장소 정보

만남 장소(**PLACE**) entity는 고유 id(P_id), 장소 이름(P_name), 주소(P_address), 우편번호(P_zipcode), 위도(P_latitude), 경도(P_longitude), 평균 평점(P_AVG_Rate), 받은 평가의



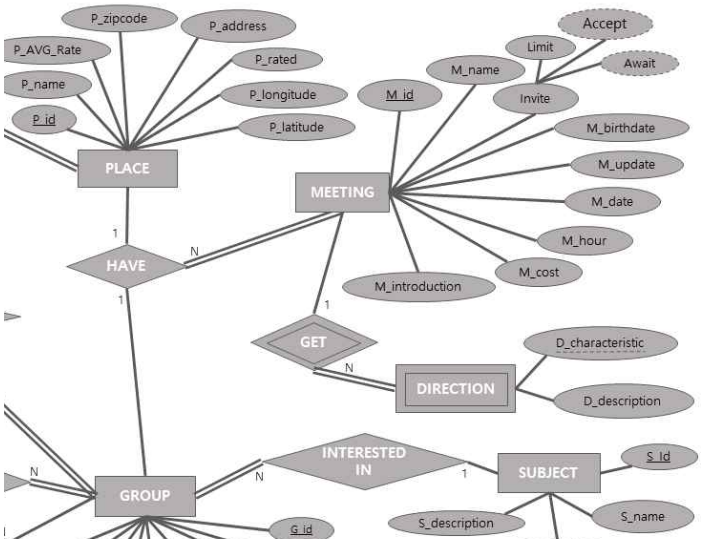
수(P_rated)를 attribute로 가지고 있다.

P_id는 **PLACE**의 primary key이다.

(R8) 오프라인 모임 정보

오프라인 모임(**MEETING**)은 고유 id(M_id), 모임 이름(M_name), 모임 생성 날짜(M_birthdate), 업데이트 날짜(M_update), 실제 모임 날짜(M_date), 예상 모임 시간(M_hour), 비용(M_cost), 모임 소개글(M_introduction)이 저장되어 있다. M_id는 **MEETING**의 primary key이다.

M_id는 **MEETING**의 primary key이다.

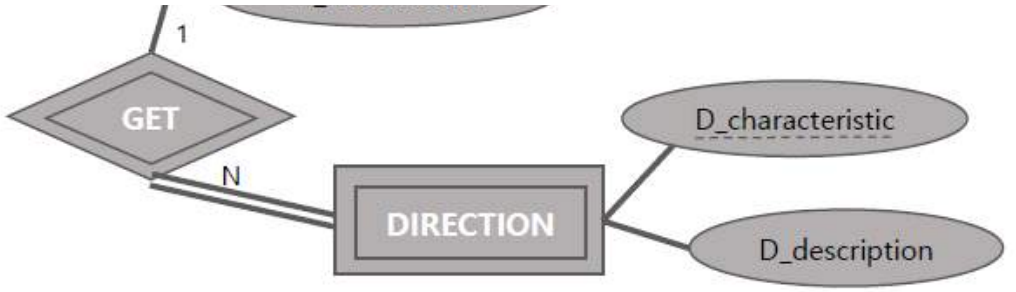


Invite는 추후에 (R10)에서 언급할 것이다.

그리고, **GROUP**, **PLACE**, **MEETING**는 ternary relationship을 이룬다. **GROUP**과 **PLACE**의 pair가 결정되었을 때 특정 그룹이 특정 장소에서 갖는 오프라인 모임은 여러 개일 수 있으므로 **GROUP**, **PLACE**, **MEETING** 세 entity의 cardinality는 1:1:N이 된다.

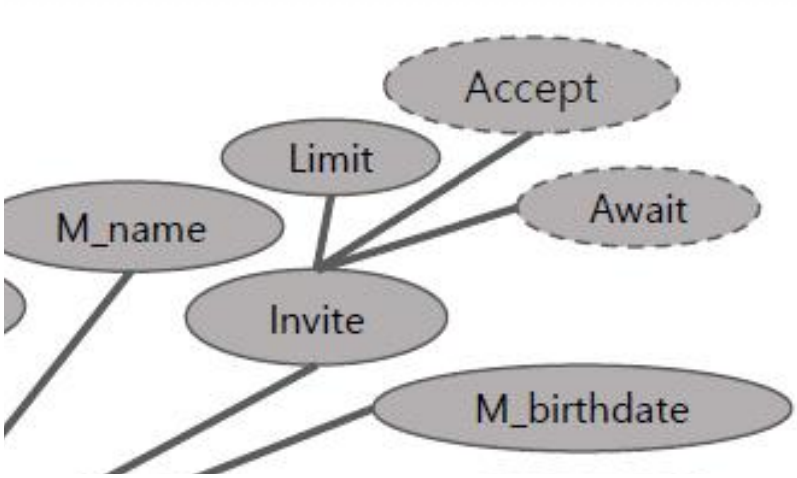
모임을 갖지 않는 그룹이 있을 수 있고, 모임이 개최되지 않는 장소는 있을 수 있으나 장소와 그룹을 갖지 않는 모임은 존재할 수 없으므로 **GROUP**과 **PLACE**는 partial로, **MEETING**은 total로 참여한다.

(R9) 찾아오는 법



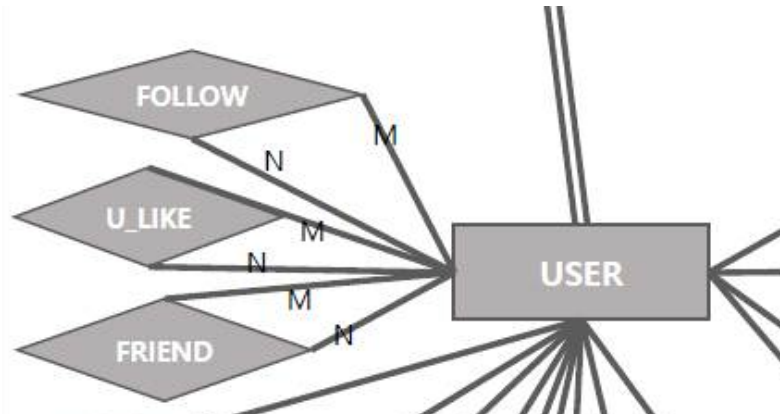
찾아오는 법은 특징과 세부설명으로 구성된다. 이를 D_characteristic, D_description으로 구분했다. 두 가지 모두 여러 개의 값을 가질 수 있다는 공통점이 있지만 description은 null 값이 있을 수 있어 구별된다. **DIRECTION**은 weak entity이며 D_characteristic은 partial key가 된다.
DIRECTION의 identifying relationship **GET**은 한 **MEETING**이 여러 개의 **DIRECTION**을 가질 수 있기 때문에 **MEETING**과 **DIRECTION**의 cardinality는 1:N이고 각각 partial, total로 참여한다.

(R10) 초대 정보



사이트 A는 모임에 초대된 사람의 수에 대한 정보를 저장하고 있기 때문에 entity type **MEETING**의 attribute invite에서 retrieve된다. 여기서 **GROUP**에 있는 사람들이 모두 **MEETING**의 대상이라고 보면 모두에게 초대가 간다. 이 때 초대를 했을 때 초대에 응한 사람, 응하지 않은 사람, 대답이 없는 사람이 나온다. 이를 통해서 초대에 응한 사람의 수 Accept와 응답 대기 중인 사람 Await의 수를 도출하는 것이 가능하다. 사람 수의 제한 limit은 모임을 하기 전에 미리 설정한다.

(R11) 교류



사용자들 사이의 교류를 기록하는 것으로 '좋아요', '팔로우' '친구' 세 가지로 나누어진다. 이를 'U_LIKE', 'FOLLOW', 'FRIEND'로 나타내었다. FOLLOW와 U_LIKE는 일방적으로 가능하지만 FRIEND는 쌍방 합의가 필요하다. 이는 추후에 constraint에서 작성한다.

세 relationship 모두 한 사용자가 여러 명과 교류를 맺을 수 있기 때문에 cardinality는 M:N이다. 또한, 모든 가입자가 참여할 필요는 없기 때문에 partial로 참여한다. 또한, 그룹에 대한 좋아요와 사용자에 대한 좋아요는 별개의 기능이기 때문에 이름을 구분하였다.

(R12)

해당 내용은 뒤에 constraint에서 자세히 기술한다.

3. Relational schema

앞의 2번에서 디자인한 ER Diagram을 기반으로 relational schema를 작성한다. 교재에 소개된 ER-to-relational mapping algorithm의 step 7가지를 따라서 순차적으로 작성한다. 그 결과 ERD에서의 entity와 relationship이 schema에 15개의 relation으로 mapping되었다.

● Step 1: Mapping of regular entity types

ER diagram에서 regular entity에 해당하는 entity들은 **USER, GROUP, CITY, SUBJECT, PLACE, MEETING, LEADER**로 총 7개이다. 이 각각의 entity에 해당하는 relation을 만들고, 각 entity에 연결된 attribute를 relation에 추가하였다. **MEETING** entity의 경우에는 composite attribute를 가지고 있으므로 simple component attribute로 분해하여 포함시킨다. derived attribute는 relational schema에서 표시하지 않는다.

그 결과로 mapping된 schema는 다음과 같다.

USER						
U_id	U_introduction	U_join_date	U_jrecent_date	U_active	U_name	C_User_id

GROUP							
Open	G_introduction	G_AVG_Rate	Approval	G_name	G_birthdate	G_id	S_Group_id

CITY						
C_id	C_zipcode	C_longitude	C_name	C_latitude	State	Country

LEADER
L_id

PLACE								
P_id	P_name	P_AVG_Rate	P_zipcode	P_address	P_rated	P_longitude	P_latitude	C_Place_id

MEETING								
M_id	M_name	M_birthdate	M_update	M_date	M_hour	M_cost	M_introduction	Invite_Limit

SUBJECT		
S_description	S_name	S_id

● Step 2: Mapping of weak entity types

ER diagram에는 weak entity로 **DIRECTION**이 존재한다. **DIRECTION** entity는 D_description과, partial key인 D_characteristic을 attribute로 가졌다. relational schema를 작성할 때 **DIRECTION**의 owner entity인 **MEETING**의 primary key인 M_id를 **DIRECTION**의 attribute로 추가해 주어야 한다. Owner entity의 primary key인 M_id와 weak entity의 partial key인 D_characteristic의 조합인 {M_id, D_characteristic}가 DIRECTION relation의 primary key가 된다.

step2의 결과로 생성된 relation은 다음과 같다.

DIRECTION		
M_id	D_characteristic	D_description

● Step 3: Mapping of Binary 1:1 Relationship Types

ER diagram에 binary 1:1 relationship이 존재하지 않으므로 step3는 건너뛴다.

● Step 4: Mapping of Binary 1:N Relationship Types

Bimary 1:N relationship type을 mapping하는 방법에는 두 가지가 있다. N-side의 entity에 foreign key로서 1-side의 entity의 primary key를 포함시킬 수 있고, 별도의 relation을 생성할 수 있다. 별도의 relation을 생성하는 경우는 relationship에 참여하지 않는 tuple이 많아 null value를 생성하는 것을 방지하기 위해 사용하는 방법이다.

ER에서 binary 1:N relationship type은 **CHOOSE, INTERESTED IN, IN, GET** 네 가지이다. GET은 weak entity의 identifying relationship이고, N-side인 weak entity들이 relationship에 total participation하므로 null value가 생기지 않는다. 따라서 별도의 relationship을 만들 필요가 없다. **DIRECTION**의 owner entity인 **MEETING**의 M_id를 Step 2에서 이미 **DIRECTION** relation에 포함시켰으므로 Step 4에서는 따로 attribute를 추가하지 않아도 된다.

CHOOSE, INTERESTED IN, IN 세 relationship은 모두 N-side가 total로 연결되어 있기 때문에 역시 null value가 생기지 않는다. 따라서 새로운 relation을 만들지 않고 1-side의 primary key를 foreigh key로서 포함시키는 방법을 사용한다. **CHOOSE**의 경우에는 **CITY**의 C_id가, **INTERESTED IN**의 경우에는 **SUBJECT**의 S_id가, **IN**의 경우에는 C_id가 각각 차례대로 **USER, GROUP, PLACE** relation의 attribute로 추가된다. 이름은 의미를 파악할 수 있도록 C_User_id, S_Group_id, C_Place_id로 한다.

결과로 **USER, GROUP, PLACE**에 foreign key가 하나씩 추가된다.

USER						
U_id	U_introduction	U_join_date	U_jrecent_date	U_active	U_name	C_User_id

GROUP							
Open	G_introduction	G_AVG_Rate	Approval	G_name	G_birthdate	G_id	S_Group_id

PLACE								
P_id	P_name	P_AVG_Rate	P_zipcode	P_address	P_rated	P_longitude	P_latitude	C_Place_id

● Step 5: Mapping of Binary M:N Relationship Types

Binary M:N relationship type을 mapping하기 위해서는 새로운 relation을 생성해야 한다. 이 경우에 relation은 relationship에 참여하는 두 개의 entity 각각의 primary key를 foreign key로써 포함하고, 이렇게 포함된 foreign key들의 조합이 relation의 primary key가 된다.

Binary M:N relationship에 해당하는 relationship은 **BELONGS TO, G_LIKE, FOLLOW, U_LIKE, FRIEND** 5개이다. 만들어진 새로운 **BELONGS TO** relation은 **USER**과 **GROUP**의 primary key인 U_id와 G_id를 foreign key로써 포함하며, 이 둘의 조합인 {U_Belong_id, G_Belong_id}가 **BELONGS TO**의 primary key가 된다. 마찬가지로 **G_LIKE** 역시 U_id와 G_id를 foreign key로 포함하며 {U_Glike_id, G_Glike_id}가 **G_LIKE**의 primary key가 된다.

FOLLOW, U_LIKE, FRIEND relationship은 **USER** entity의 recursive relationship이다. 그러므로 relationship에 참여하는 entity는 **USER** entity 하나뿐이며, relationship에 참여하는 두 참여자를 모두 **USER** entity의 primary key로 나타낸다. **FOLLOW, U_LIKE, FRIEND** 세 relation을 생성하고 각각 USER를 reference하는 foreign key를 두 개씩 추가한다. **FOLLOW**의 경우에는 U_FOLLOWER_id, U_FOLLOWEE_id, **U_LIKE**의 경우에는 U_LIKER_id, U_LIKED_id, **FRIEND**의 경우에는 동등한 관계이므로 U_1_id, U_2_id로 추가한다. 각 relation의 primary key는 두 foreign key의 집합이며 {U_FOLLOWER_id, U_FOLLOWEE_id}, {U_LIKER_id, U_LIKED_id}, {U_1_id, U_2_id}이다.

다음과 같은 relation들이 생성된다.

BELONGS TO		G_LIKE	
U_Belong_id	G_Belong_id	U_Glike_id	G_Glike_id

FOLLOW		U_LIKE		FRIEND	
U_FOLLOWER_id	U_FOLLOWEE_id	U_LIKER_id	U_LIKED_id	U_1_id	U_2_id

● Step 6: Mapping of Multivalued Attributes

ER diagram에 multivalued attributes가 없으므로 step 6는 생략한다.

● Step 7: Mapping of N-ary Relationship types

ER diagram에 **ASSIGNED**와 **HAVE** 두 개의 ternary relationship이 존재한다. ternary relationship 각각 relation을 만들어야 하며, relationship에 참여하는 각각의 entity들의 primary key들을 relation의 foreign key로 포함시킨다.

ASSIGNED의 경우에는 **LEADER, USER, GROUP**으로, 각각의 primary key인 L_Assigned_id, U_Assigned_id, G_Assigned_id가 relation에 추가되며 이 세 foreign key의 조합인 {L_Assigned_id, U_Assigned_id, G_Assigned_id}가 ASSIGNED의 primary key가 된다. 마찬가지로 **HAVE**에는 **PLACE, MEETING, GROUP**이 참여하며 각각의 primary key인 P_Have_id, M_Have_id, G_Have_id가 foreign key로 **HAVE** relation에 추가된다. 이 세 foreign key의 조합인 {P_Have_id, M_Have_id, G_Have_id}가 **HAVE**의 primary key가 된다.

HAVE		
G_Have_id	P_Have_id	M_Have_id

ASSIGNED		
U_Assigned_id	M_Assigned_id	G_Assigned_id

4. Constraints

데이터베이스에서 constraints는 크게 세 가지, inherent model-based constraints, schema-based constraints, 그리고 application-based constraints로 나뉘어진다.

우선 inherent model-based constraints부터 살펴보면, 데이터모델 자체에 내재되어 있는 relation의 특징에 의해 그 어떤 relation도 중복되는 tuple을 가질 수 없다.

Application program 등의 방법으로 구현해야 하는 application-based constraints에는 세 가지가 있다. 첫 번째는 **USER**이 하나의 **GROUP**에 대해 두 번 '**LIKE**'할 수 없다는 것이다. 두 번째는 **FRIEND/FOLLOW/LIKE**의 두 사용자가 같을 수 없다는 것이다. 이 중 특히 **FRIEND**의 경우 상호 동등한 관계이기 때문에 항상 u1_id가 u2_id보다 작다는 조건이 추가된다. 세 번째는 **FRIEND**의 경우 상호 수락을 해야만 tuple이 형성된다는 것이다.

마지막으로 schema-based constraints를 살펴보자. Schema-based constraints에는 domain constraints, key constraints, constraints on NULLs, entitty integrity constraints, referential integrity constriants가 있다.

우선 각 attribute의 domain constraints와 primary key는 부록2에 표시해두었다. 이때 lonitude는 -180부터 180까지의 실수값을 가질 수 있으며, 이때 -부호가 붙은 것은 서경, +값을 가지는 것은 동경 경도로 간주한다. latitude의 경우 -90부터 90까지의 실수값을 가질 수 있으며, -값을 가지는 것은 남위, +값을 가지는 것은 북위 위도로 간주한다.

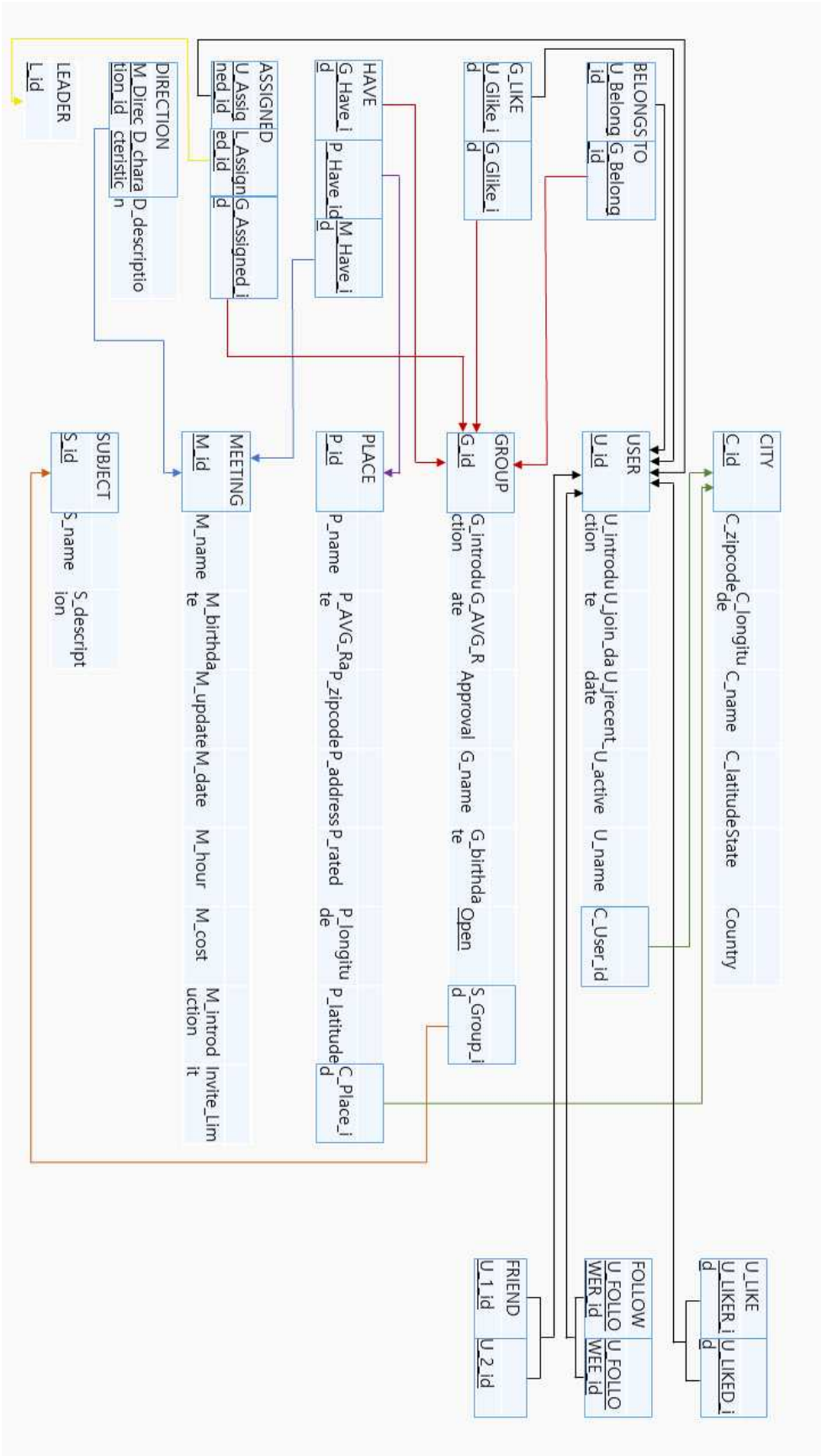
Constraints of NULLs는 NULL 값이 허용되는 attribute와 허용되지 않는 attribute를 구분하는 것이다. 다음 attribute에는 NULL 값이 허용될 수 있으며, 이 외에는 NULL 값이 허용되지 않는다.
CITY : C_zipcode, C_latitude, C_longitude
SUBJECT : S_description
GROUP : G_introduction
USER : U_introduction
PLACE : P_address, P_zipcode, P_longitude, P_latitude
MEETING : M_introduction

이때 **MEETING**의 찾아오는 법(**DIRECTION**)은 없어도 되는 항목이지만 해당 값이 존재할 때에만 relation에 tuple을 만드는 것으로 이해하여 NULL 값이 허용되는 attribute에 포함하지 않았다.

Entity integrity constraints에 의해 어떤 relation의 primary key도 NULL 값을 가질 수 없다.
Referential integrity constraints에 의해 relation의 foreign key가 refer하는 값은 NULL일 수 없고, 반드시 존재하는 tuple을 refer해야 한다.

5. 부록

(부록1) Relational Schema



(부록2) 각 attribute의 domain constraints와 primary key (밑줄 그은 attribute)

USER								
DC	U_id INTEGER	U_introduction VARCHAR(255)	U_join_date DATETIME(yyyy-mm-dd hh:mm:ss)	U_recent_date DATETIME(yyyy-mm-dd hh:mm:ss)	U_name VARCHAR(30)	U_active BOOLEAN	C_User_id INTEGER	
GROUP								
DC	G_id INTEGER	Approval BOOLEAN	G_birthdate DATETIME(yyyy-mm-dd hh:mm:ss)	G_name VARCHAR(30)	G_AVG_Rate FLOAT	G_introduction VARCHAR(255)	Open BOOLEAN	S_Group_id INTEGER
LEADER								
DC	L_id INTEGER							
CITY								
DC	C_id INTEGER	C_zipcode INTEGER	C_longitude FLOAT(-90,90)	C_latitude FLOAT(-180,180)	C_name VARCHAR(30)	State VARCHAR(30)	Country VARCHAR(30)	
PLACE								
DC	P_id INTEGER	P_name VARCHAR(30)	P_AVG_Rate FLOAT	P_zipcode INTEGER	P_address VARCHAR(255)	P_rated INTEGER	P_longitude FLOAT(-90,90)	P_latitude FLOAT(-180,180)	C_Place_id INTEGER
MEETING								
DC	M_id INTEGER	M_name VARCHAR(30)	M_birthdate DATETIME(yyyy-mm-dd hh:mm:ss)	M_update DATETIME(yyyy-mm-dd hh:mm:ss)	M_date DATETIME(yyyy-mm-dd hh:mm:ss)	M_hour INTEGER	M_cost VARCHAR(255)	M_introduction VARCHAR(255)	Invite_Limit INTEGER
SUBJECT								
DC	S_id INTEGER	S_name VARCHAR(30)	S_description VARCHAR(255)					
DIRECTION								
DC	M_Direction_id INTEGER	D_characteristic VARCHAR(255)	D_description VARCHAR(255)					
FOLLOW								
DC	U_Follower_id INTEGER	U_Followee_id INTEGER						
U_LIKE								
DC	U_Liker_id INTEGER	U_Liked_id INTEGER						
FRIEND								
DC	U_1_id INTEGER	U_2_id INTEGER						
BELONGS TO								
DC	U_Belong_id INTEGER	G_Belong_id INTEGER						
G_LIKE								
DC	U_Glike_id INTEGER	G_Glike_id INTEGER						
ASSIGNED								
DC	L_Assigned_id INTEGER	U_Assigned_id INTEGER	G_Assigned_id INTEGER					
HAVE								
DC	P_Have_id INTEGER	G_Have_id INTEGER	M_Have_id INTEGER					