DMA P#1

Conceptual DB design & DB implementation

2024-1 데이터 관리와 분석
9조 강명훈 고우성 김원재 김영헌

INDEX

GOAL

- ✓ 주어진 요구사항들을 만족 하는 데이터베이스를 설계 및 구현.
- ✓ 사이트 A 가 사용하는 DB 의 ER diagram 도식화와 DB 구현을 목적으로 한다.

PART 1

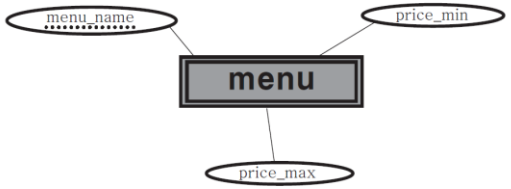
Requirement를 만족하는 ER diagram을 도식화한다.

PART 2

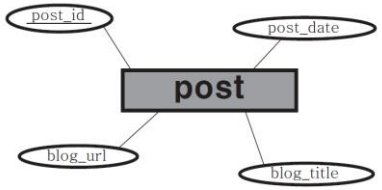
Part1에서 제시된 조건을 바탕으로 DB schema를 설계하여 데이터베이스 테이블을 실제로 생성한 후 데이터 입력까지를 목표로 한다.

PART 1

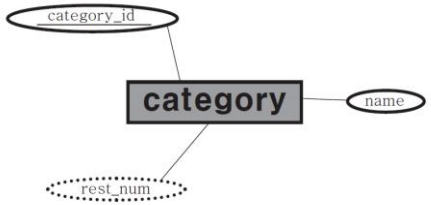
Initial conceptual design



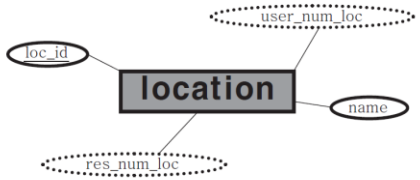
1. "menu" 초기 design



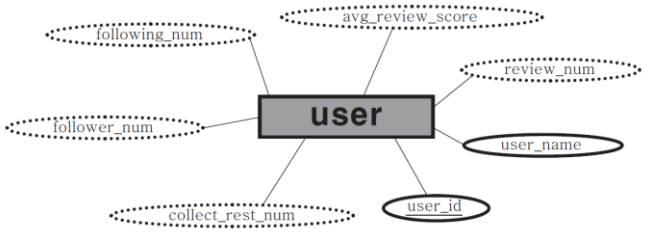
2. "post" 초기 design



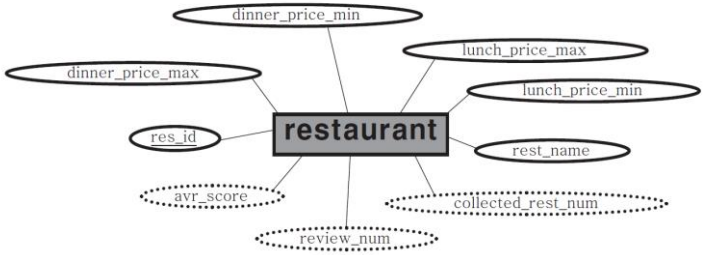
3. "category" 초기 design



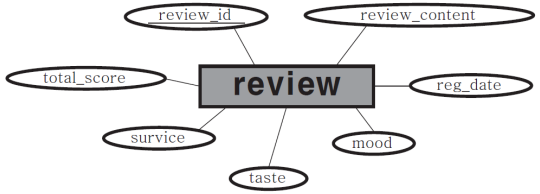
4. "location" 초기 design



5. "user" 초기 design

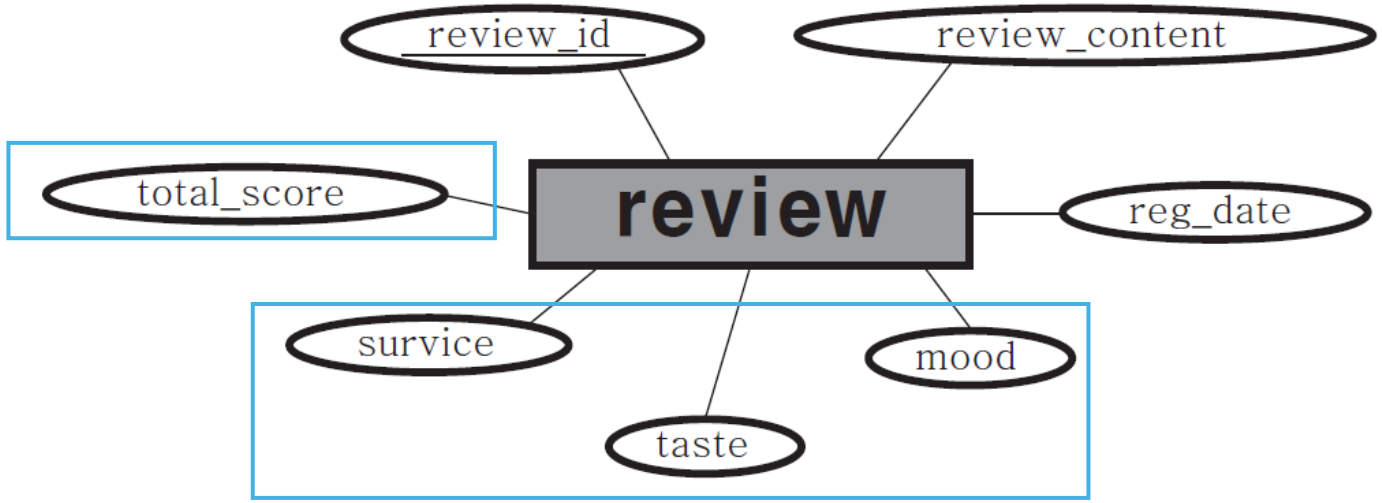


6. "restaurant" 초기 design



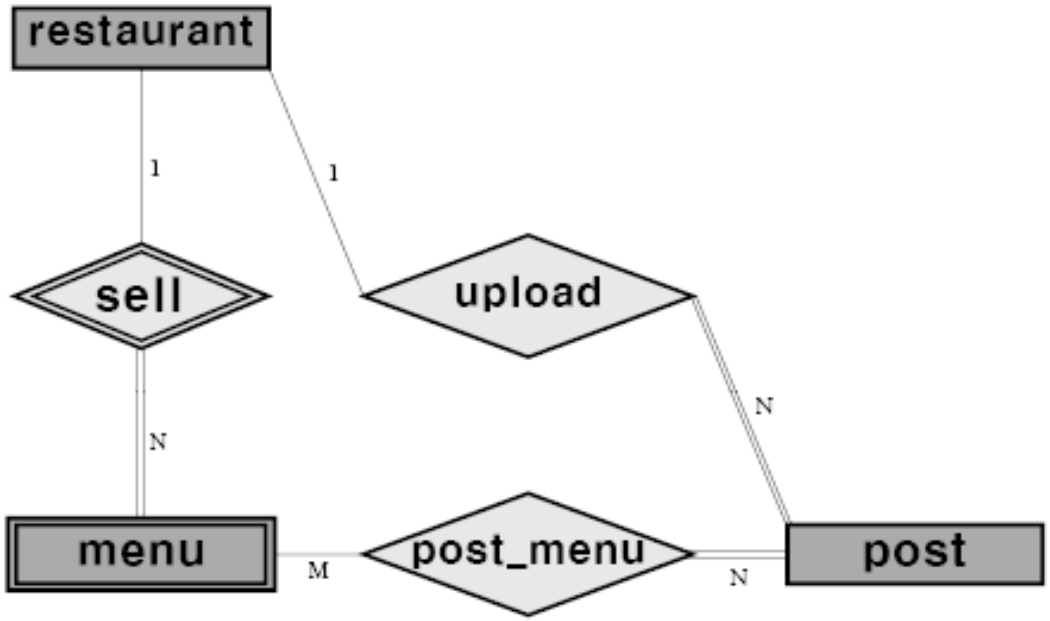
7. "review" 초기 design

Type of attribute

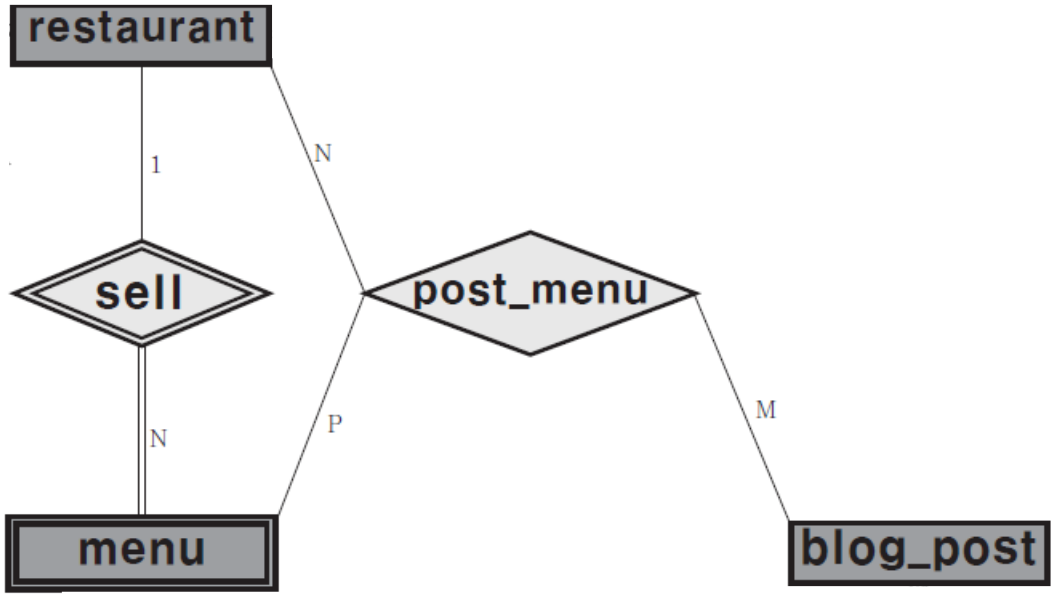


"composite attribute" *"derived attribute"*

Relationship type of degree

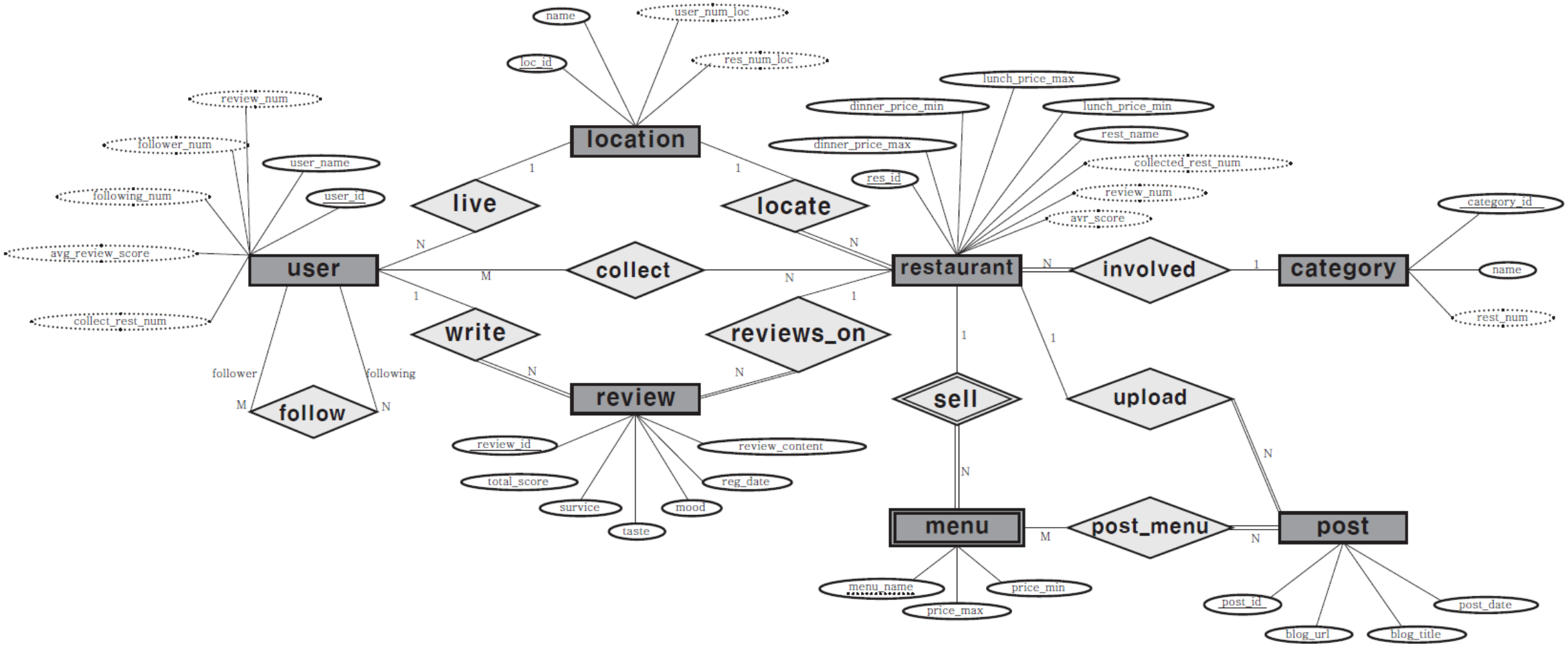


"binary relationship"



"ternary relationship"

Final ER diagram



PART 2

Data table

✓ 릴레이션 이름 별로, 데이터 삽입 시 column명, datatype, NOT NULL 여부, Primary key 표시(노란색 음영, 각 릴레이션에 2개 이상 칠해질 경우 각 column이 combination으로 엮여 PK로 설정됨.)

User	Restaurant		Review		Menu	Post_Menu	Location	Post		Follow	Collection	Category
user_id VARCHAR(255) NOT NULL	**restaurant_id** VARCHAR(255) NOT NULL	**dinner_price_min** INT(11)	**review_id** INT(11) NOT NULL	**taste_score** DECIMAL(11)	**menu_name** VARCHAR(255) NOT NULL	**post_id** INT(11) NOT NULL	**name** VARCHAR(255) NOT NULL	**blog_title** VARCHAR(255)	**restaurant** VARCHAR(255) NOT NULL	**followee_id** VARCHAR(255) NOT NULL	**user_id** VARCHAR(255) NOT NULL	**name** VARCHAR(255) NOT NULL
user_name VARCHAR(255) NOT NULL	**restaurant_name** VARCHAR(255) NOT NULL	**dinner_price_max** INT(11)	**review_content** VARCHAR(255)	**service_score** DECIMAL(11)	**price_min** INT(11)	**menu_name** VARCHAR(255) NOT NULL	**location_id** INT(11) NOT NULL	**blog_URL** VARCHAR(255) NOT NULL	**post_id** INT(11) NOT NULL	**follower_id** VARCHAR(255) NOT NULL	**restaurant_id** VARCHAR(255) NOT NULL	**category_id** INT(11) NOT NULL
region INT(11)	**lunch_price_min** INT(11)	**location** INT(11) NOT NULL	**reg_date** DATETIME	**mood_score** DECIMAL(11)	**price_max** INT(11)	**restaurant** VARCHAR(255) NOT NULL		**post_date** DATETIME NOT NULL				
	lunch_price_max INT(11)	**category** INT(11) NOT NULL	**user_id** VARCHAR(255) NOT NULL	**restaurant** VARCHAR(255) NOT NULL	**restaurant** VARCHAR(255) NOT NULL							
			total_score DECIMAL(11) NOT NULL									

Foreign key constraint

✓ Requirement4에선 foreign key constraint에 대한 내용을 코드로 작성하였다. 각 코드에 대한 설명을 가독성을 위해 다음과 같이 통일한다.

Table a(attribute A) -> Table b(attribute B) == Relationship C

→ 테이블 a의 attribute A(foreign key)가 테이블 b의 attribute B를 refer하고 이는 ER diagram에서의 Relationship C에 해당됨.

```
    # 1. User's foreign key
    cursor.execute("ALTER TABLE User ADD CONSTRAINT FOREIGN KEY (region)
REFERENCES Location(location_id)")
    print("1-1 done")
```

User(region) -> Location(location_id) == live

```
    # 2. Restaurant's foreign key
     cursor.execute("ALTER TABLE Restaurant ADD CONSTRAINT FOREIGN KEY
(location) REFERENCES Location(location_id)")
    print("2-1 done")
     cursor.execute("ALTER TABLE Restaurant ADD CONSTRAINT FOREIGN KEY
(category) REFERENCES Category(category_id)")
    print("2-2 done")
```

Restaurant(location) -> Location(location_id) == locate

Restaurant(category) -> Category(category_id) == involved

Foreign key constraint

```
    # 3. Review's foreign key
    cursor.execute("ALTER TABLE Review ADD CONSTRAINT FOREIGN KEY (user_id)
REFERENCES User(user_id)")
    print("3-1 done")
     cursor.execute("ALTER TABLE Review ADD CONSTRAINT FOREIGN KEY (res-
taurant) REFERENCES Restaurant(restaurant_id)")
    print("3-2 done")
```

Review(user_id) -> User(user_id) == write

Review(restaurant) -> Restaurant(restaurant_id) == reviews_on

```
    # 4. Post's foreign key
    cursor.execute("ALTER TABLE Post ADD CONSTRAINT FOREIGN KEY (restau-
rant) REFERENCES Restaurant(restaurant_id)")
    print("4-1 done")
```

Post(restaurant) -> Restaurant(restaurant_id) == upload

```
    # 5. Menu's foreign key
    cursor.execute("ALTER TABLE Menu ADD CONSTRAINT FOREIGN KEY (restau-
rant) REFERENCES Restaurant(restaurant_id)")
    print("5-1 done")
```

Menu(restaurant) -> Restaurant(restaurant_id) == sell

```
    # 6. Collection's foreign key
     cursor.execute("ALTER TABLE Collection ADD CONSTRAINT FOREIGN KEY
(restaurant_id) REFERENCES Restaurant(restaurant_id)")
    print("6-1 done")
     cursor.execute("ALTER TABLE Collection ADD CONSTRAINT FOREIGN KEY
(user_id) REFERENCES User(user_id)")
    print("6-2 done")
```

Collection(restaurant_id) -> Restaurant(restaurant_id) == collect

Collection(user_id) -> User(user_id) == collect

Foreign key constraint

```
    # 7. Post_Menu's foreign key
    cursor.execute("ALTER TABLE Post_Menu ADD CONSTRAINT FOREIGN KEY
(post_id) REFERENCES Post(post_id)")
    print("7-1 done")
    cursor.execute("ALTER TABLE Post_Menu ADD CONSTRAINT FOREIGN KEY (res-
taurant, menu_name) REFERENCES Menu(restaurant, menu_name)")
    print("7-2 done")
```

Post_Menu(post_id) -> Post(post_id) == post_menu

Post_Menu(restaurant, menu_name) -> Menu(restaurant, menu_name) == post_menu

```
    # 8. Follow's foreign key
    cursor.execute("ALTER TABLE Follow ADD CONSTRAINT FOREIGN KEY (fol-
lowee_id) REFERENCES User(user_id)")
    print("8-1 done")
    cursor.execute("ALTER TABLE Follow ADD CONSTRAINT FOREIGN KEY (fol-
lower_id) REFERENCES User(user_id)")

    print("8-2 done")
```

Follow(followee_id) -> User(user_id) == follow

Follow(follower_id) -> User(user_id) == follow

- ✓ Menu (restaurant,menu_name)으로 한 이유는, 참조 무결성 제약 조건을 지키기 위함이다. 만약restaurant_id를 Restaurant relation에서 refer할 경우 참조 무결성 제약이 어겨지고, 실제로도 코드 오류를 발생시키므로 위와 같이 설정하였다.

THANK YOU

2024-1 데이터 관리와 분석
9조 강명훈 고우성 김원재 김영헌