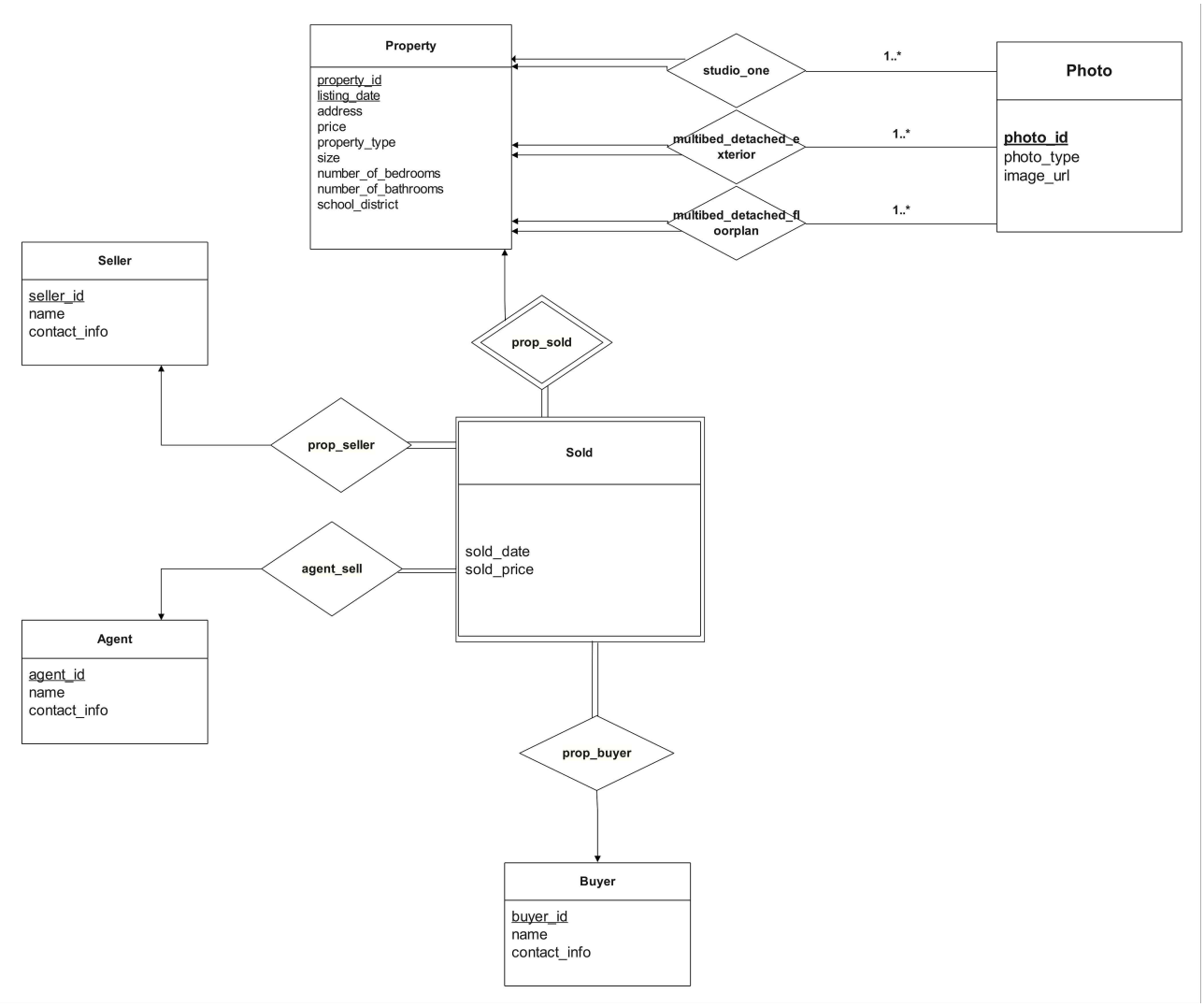
데이터베이스 시스템 프로젝트1 보고서

20190741 김지수

1. E-R diagram

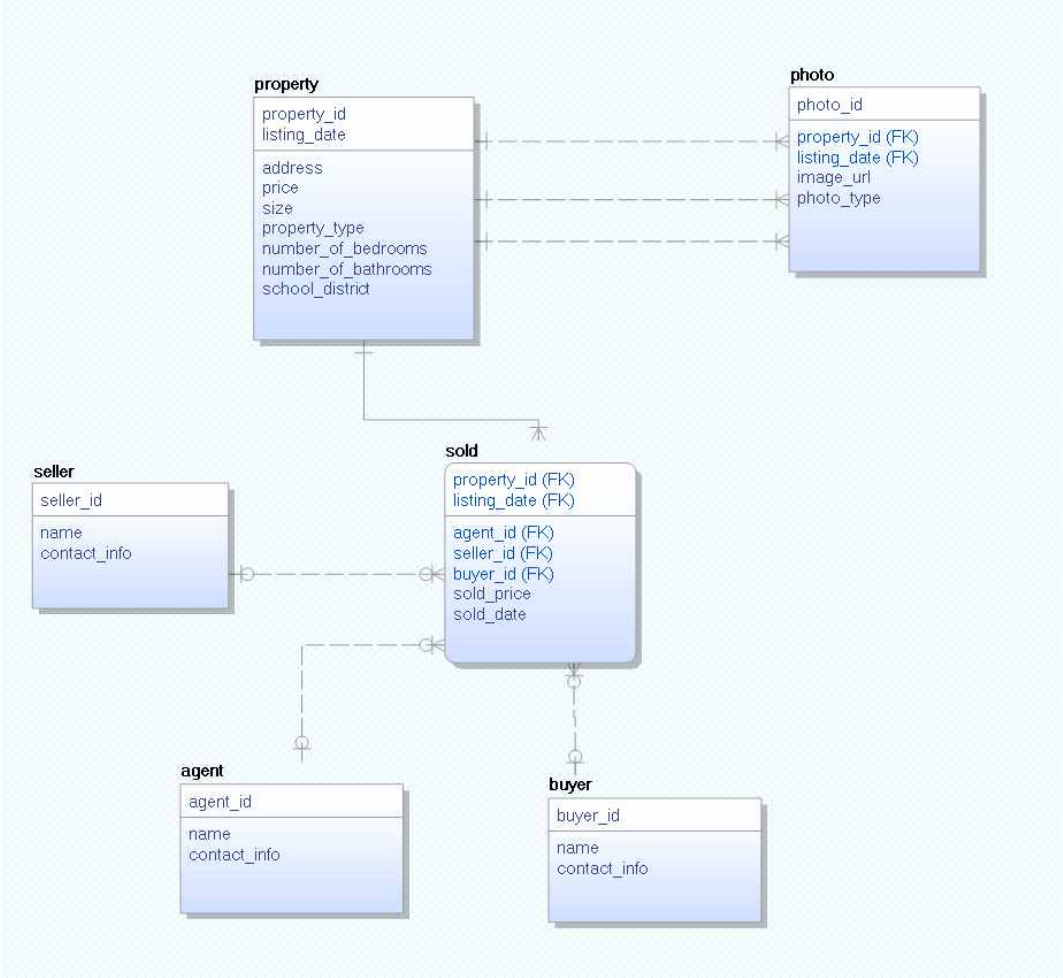


본 부동산 데이터베이스에서는 대한민국의 부동산 거래 시스템과 동일하게 부동산 중개인(공인중개사)이 판매자와 구매자를 연결하는 것을 가정하고 작성하였다. 다시 말해서, 부동산 판매자의 중개인과 구매자의 중개인이 동일한 것이다. 이러한 배경지식을 바탕으로 E-R 다이어그램을 구성하였다.

판매되는 부동산을 구별하는 것으로는 property_id와 listing_date를 Primary key로 지정하였다. 동일한 매물이 이후에 다시 판매되는 경우에는 이를 구분할 수 없기 때문에 판매가 등록된 날짜도 같이 지정한 것이다. 이러한 작업을 통해서 동일한 매물도 판매 기록이 저장되는 개체인 sold 개체와의 조인 연산을 통해서 요구되는 쿼리를 수행할 수 있도록 설계하였다.

2. Relational Schema

아래는 위의 E-R diagram을 바탕으로 erwin 프로그램을 사용하여 작성한 Relational Schema를 첨부하였다.



<Erwin relational model>

각 Entity에서 Attribute의 Null, Not Null은 Erwin 파일에 표시했습니다.

3. About Entities

3.1. seller

seller	부동산 판매인	Type
seller_id	seller의 id	INTEGER PRIMARY KEY
name	seller의 이름	varchar(20)
contact_info	seller의 연락처	varchar(20)

부동산 판매자에 대한 기본 정보를 생각하여 개체를 구성하였다. 요구되는 쿼리에는 필요하지 않으나 기본적으로 판매자 연락처 정보와 이름은 필요할 것이라고 생각하여 추가하였다.

3.2. buyer

buyer	부동산 구매인	Type
buyer_id	buyer의 id	INTEGER PRIMARY KEY
name	buyer의 이름	varchar(20)
contact_info	buyer의 연락처	varchar(20)

3.3. agent

agent	부동산 중개인	Type
agent_id	agent의 id	INTEGER PRIMARY KEY
name	agent의 이름	varchar(20)
contact_info	agent의 연락처	varchar(20)

3.4. property

property	부동산 매물	Type
property_id	property의 id	INTEGER PRIMARY KEY
listing_date	property의 등록날짜	DATE PRIMARY KEY
address	주소	varchar(50)
price	가격	INTEGER
size	크기(평수)	INTEGER
property_type	property의 종류	varchar(40)
number_of_bedrooms	침실의 개수	INTEGER
number_of_bathrooms	화장실의 개수	INTEGER
school_district	학군(1학군~11학군)	INTEGER

부동산 정보를 저장하는 property 개체에는 모든 부동산의 정보를 담고 있다. 만약 해당 부동산이 팔린 것인지 확인하기 위해서는 sold 개체에 있는지 여부를 확인하기 위해서 NOT IN 연산을 사용하면 간단하게 확인할 수 있다. 추가적으로 요구사항 a)에 있는 “Mapo”구를 찾아내기 위해서는 문자열로 되어있는 address에서 LIKE '%Mapo%’를 통해서 찾아낼 수 있다. 이를 통해서 주소를 입력받고 추가로 지역(district)까지 속성으로 추가하지 않아 중복성을 제거하였다. property_type에는 명세서에 나온 studio, one-bedroom apartment, multi-bedroom apartment, detached house가 varchar(40)으로 입력된다.

3.5. sold

sold 개체는 property 개체에 종속적이다. 즉 약한 개체 집합으로, sold는 판매되는 해당 property가 없으면 존재하지 않는다. 다시 말해서, 부동산이 기존에 이미 등록되어야지 판매될 수 있는 것이다. 따라서 primary key로는 강한 개체집합인 property의 primary key와 필요한 경우 sold의 속성을 primary key로 갖는다. 여기서 본 데이터베이스 시스템에서는 하루에 동일한 날짜에 등록된 부동산이 여러 번 판매될 수 없다고 고려하였다. 다시 말해서, 동일한 매물이 하루에 두 번 판매되는 일은 없는 것이다. 이는 부동산이라는 재화의 속성을 고려한 것이고, 해당 매물에 동일한 판매자와 구매자가 해당 날짜에 2번 거래하기보다는 부동산 수수료로 인해 계약이 없었던 것으로 하는 것이 합리적이기 때문에 이렇게 설정하였다. 따라서 sold 개체의 primary키는 property 개체의 primary key만을 갖는 것으로 정의하였다.

추가로 sold 엔티티는 agent_id, seller_id, buyer_id를 foreign key로 갖는다. 이는 join 연산을 통해서 구매자, 판매자, 중개인(agent)이 판매된 부동산 매물 중 어떤 것과 관련이 있는지 확인하기 위함이다. 이를 통해서 c)와 d) 항목에 존재하는 agent와 해당 agent가 판매한 부동산 매물정보를 처리할 수 있다. 구체적인 예시 쿼리는 마지막 파트에서 작성했다. 아래는 E-R 다이어그램에서 표시된 것을 기반으로 entity를 표로 작성하였다. (외래키는 생략하였다.)

sold	부동산 구매인	Type
property_id	property의 id	INTEGER PRIMARY KEY
listing_date	property의 등록날짜	DATE PRIMARY KEY
sold_price	해당 부동산의 최종 판매된 가격	INTEGER
sold_date	해당 부동산의 판매날짜	DATE

여기서 property의 price와 sold의 sold_price는 redundancy가 아니다. 왜냐하면 판매자가 부동산을 등록한 가격과 구매자와 판매자가 합의하여 최종적으로 거래된 가격은 다를 수 있기 때문이다. 실제로 agent의 실적으로 고려하는 것에는 최종 판매된 가격이 사용된다. 한편, 구입할 수 있는 부동산의 가격을 고려할 때는 property의 price가 고려되는 것이다. sold 개체에 있는 sold_date와 property에 있는 listing_date 속성을 활용해서 해당 부동산 매물이 판매되기까지 소요된 기간을 구할 수 있다.

3.6 photo

photo 개체는 부동산의 사진 정보를 담는 개체로, photo_type은 interior_photo, exterior_photo, floor_plan 등의 값을 갖는다. photo는 photo_id를 primary key로 갖도록 설정하였다. 해당 이유는 사진의 장수에는 제한이 없기 때문에, 동일 부동산의 동일 인테리어 사진에 대해서도 이미지가 여러 장이 존재할 수 있기 때문이다. URL의 경우에는 언제나 상황에 따라서 변경될 수 있기 때문에 primary key로 사용하는 것을 고려하지 않았다. 부동산의 종류에 따라서 photo와 서로 다른 카디널리티를 가지기 때문에 이를 E-R 다이어그램에서 별도로 표현하였다. 추가적으로 해당 사진이 어느 부동산과 관련이 있는 사진인지를 나타내기 위해서 property_id와 listing_date를 foreign key로 갖는다.

photo	부동산 사진	Type
photo_id	photo의 id	INTEGER PRIMARY KEY
image_url	사진정보	Image type으로 설정
photo_type	사진 타입	varchar(40)

photo가 property의 primary key를 foreign key로 가지기 때문에 해당 부동산의 사진은 property와 photo의 join을 통해서 구할 수 있다. 또한 “데이터 베이스에서 가장 비싼” 이라는 조건이므로 property에는 판매된 부동산이나 아직 미판매된 부동산 데이터가 모두 작성되어 있어 해당 쿼리를 쉽게 수행할 수 있다.

4. About Relationships

E-R 다이어그램에서 나타나는 relation set을 정리하였다.

relation name	관계 대상	설명
prop_sold	Property <-> Sold	Sold가 Weak entity set으로 property가 참조되는 개체이다. Sold라는 판매 장부 기록은 해당 부동산이 부동산 리스트에 등록되어 있어야 기록될 수 있기 때문이다. 이러한 이유로 Property에 listing_date이 primary key로 동일 부동산이라도 당일에 두 개의 거래가 없다고 전재하였으므로 이들을 구별할 수 있다. 한편 Sold는 약한 개체 집합이므로 강한 개체 집합인 Property의 primary key를 primary key로 갖는다. 여러 건의 거래가 단일 매물에 대해 이루어질 수 있기 때문에 관계를 일대다로 표현하였으며, sold에 있는 인스턴스는 모두 해당하는 Property가 존재해야 하므로 2개의 선으로 표현하였다. 당연히 등록된 부동산이 판매기록이 없을 수 있기 때문에 일대다에서 일은 단일선으로 표시되었다.
prop_seller	Seller <-> Sold	부동산을 판매하는 판매자와 판매된 부동산을 기록하는 장부 사이의 관계이다. 당연히 판매는 한 명의 판매자로부터 이루어지고, Seller는 등록이 되어 있어도 실제로 판매기록은 없을 수 있으므로 일대다에서 일은 단일선으로 표시되었다. 이러한 관계를 통해 Sold는 Seller의 seller_id를 foreign key로 갖는다.

agent_sell	Agent <-> Sold	부동산을 중개하는 대리인과와 판매된 부동산을 기록하는 장부 사이의 관계이다. 당연히 판매는 한 명의 대리인으로부터 이루어지고(위의 한국 부동산 가정), Agent는 등록이 되어 있어도 실제로 판매기록은 없을 수 있으므로 일대다에서 일은 단일선으로 표시되었다. 이러한 관계를 통해 Sold는 Agent의 agent_id를 foreign key로 갖는다.
prop_buyer	Buyer <-> Sold	부동산을 구매하는 대리인과와 판매된 부동산을 기록하는 장부 사이의 관계이다. 당연히 판매는 한 명의 구매자으로부터 이루어지고, 구매자는 등록이 되어 있어도 실제로 구매기록은 없을 수 있으므로(집 구경만 진행함) 일대다에서 일은 단일선으로 표시되었다. 이러한 관계를 통해 Sold는 Buyer의 buyer_id를 foreign key로 갖는다.
studio_one	Property(종류가 studio나 one-bed) <-> Photo	부동산 중 studio나 one-bedroom apartment가 property_type인 것들과 Photo 사이의 관계를 표현하였다. 위의 타입들의 경우 최소 한 장 이상의 interior photo가 필요하므로 위의 그림과 같이 표현하였다. 일대다의 관계인데 Property가 모두 studio_one 관계에 참여해야하므로 이중선에 화살표를 같이 표현하였다. (사이버 캠퍼스 질의 참조함)
multibed_detached_exterior	Property(종류가 multi-bed나 detached house) <-> Photo	부동산 중 multi-bedroom apartment나 detached house가 property_type인 것들과 Photo 사이의 관계를 표현하였다. 위의 타입들의 경우 최소 한 장 이상의 exterior photo가 필요하므로 위의 그림과 같이 표현하였다. 일대다의 관계인데 Property가 모두 해당 관계에 참여해야 하고 사진은 최소 1장 이상이므로 이중선에 화살표를 같이 표현하였다.
multibed_detached_floorplan	Property(종류가 multi-bed나 detached house) <-> Photo	부동산 중 multi-bedroom apartment나 detached house가 property_type인 것들과 Photo 사이의 관계를 표현하였다. 위의 타입들의 경우 위와 더불어 최소 한 장 이상의 floor plan 사진도 필요하기 때문에 이와 같이 추가하였다. 일대다의 관계인데 Property가 모두 해당 관계에 참여해야 하고 사진은 최소 1장 이상이므로 이중선에 화살표를 같이 표현하였다.

5. Query Example

아래는 구성한 데이터베이스 모델을 기반으로 수업시간에 배웠던 SQL코드를 생각하여 임시로 작성하였다. 추후 쿼리 수행을 목표로 하는 만큼 임시적으로 코드를 작성했다.

먼저 아래는 테이블 구성 코드이다.

```
CREATE TABLE Agent (
    agent_id INTEGER PRIMARY KEY,
    Name TEXT,
    Contact_Info TEXT
);
```

```
CREATE TABLE Buyer (
    buyer_id INTEGER PRIMARY KEY,
    Name TEXT,
    Contact_Info TEXT
);

CREATE TABLE Seller (
    seller_id INTEGER PRIMARY KEY,
    Name TEXT,
    Contact_Info TEXT
);

CREATE TABLE Property (
    property_id INTEGER PRIMARY KEY,
    listing_date DATE PRIMARY KEY,
    address TEXT,
    price INTEGER,
    size INTEGER,
    property_type TEXT,
    Number_of_Bedrooms INTEGER,
    Number_of_Bathrooms INTEGER,
    School_District INTEGER,
);

CREATE TABLE Photo (
    photo_id INTEGER PRIMARY KEY,
    property_ID INTEGER,
    listing_date DATE
    photo type TEXT,
    image_url TEXT,
    FOREIGN KEY (property_ID) REFERENCES Property(property_ID)
    FOREIGN KEY (listing_date) REFERENCES Property(listing_date)
);

CREATE TABLE Sold (
    property_id INTEGER,
    listing_date DATE,
    seller_id INTEGER,
    agent_id INTEGER,
    buyer_id INTEGER,
    sold_date DATE,
    sold_price INTEGER,
    PRIMARY KEY (property_id, listing_date),
```

```
    FOREIGN KEY (property_id, listing_date) REFERENCES Property(property_id, listing_date),
    FOREIGN KEY (seller_id) REFERENCES Seller(seller_id),
    FOREIGN KEY (agent_id) REFERENCES Agent(agent_id),
    FOREIGN KEY (buyer_id) REFERENCES Buyer(buyer_id)
);
```

이를 기반으로 a)~ g)의 쿼리를 작성해보았다.

a) Find address of homes for sale in the district “Mapo” costing between ₩1,000,000,000 and ₩1,500,000,000.

```
SELECT p.address
FROM Property p
WHERE p.address LIKE '%Mapo%' AND p.price BETWEEN 1000000000 AND 1500000000
AND (p.property_id, p.listing_date) NOT IN (SELECT property_id, listing_date FROM Sold);
```

b) Find the address of homes for sale in the 8th school district with 4 or more bedrooms and 2 bathrooms.

```
SELECT p.address
FROM Property p
WHERE p.School_District = 8 AND p.Number_of_Bedrooms >= 4 AND p.Number_of_Bathrooms
= 2 AND (p.property_id, p.listing_date) NOT IN (SELECT property_id, listing_date FROM Sold);
```

c) Find the name of the agent who has sold the most properties in the year 2022 by total won value.

```
SELECT a.Name
FROM Agent as a
INNER JOIN Sold as s ON a.agent_id = s.agent_id
WHERE strftime('%Y', s.sold_date) = '2022'
GROUP BY a.agent_id
ORDER BY SUM(s.sold_price) DESC
LIMIT 1;
```

d) For each agent, compute the average selling price of properties sold in 2022, and the average time the property was on the market. (Note that this suggests the use of date attributes in your design)

```
SELECT a.Name,
        AVG(s.sold_price) AS Average_Selling_Price,
        AVG(julianday(s.sold_date) - julianday(p.listing_date)) AS Average_Time_On_Market
FROM Agent AS a
INNER JOIN Sold AS s ON a.agent_id = s.agent_id
```

```
INNER JOIN Property p ON s.property_id = p.property_id AND s.listing_date = p.listing_date
WHERE strftime('%Y', s.sold_date) = '2022'
GROUP BY a.agent_id;
```

strftime은 교재의 sql 온라인 인터프리터 사이트가 SQLite database이므로 여기서 연도 추출을 위해 활용할 수 있는 방법을 찾아서 사용하였다.

e) Show photos of the most expensive studio, one-bedroom, multi-bedroom apartment(s), and detached house(s), respectively, from the database.

```
SELECT ph.image_url, p.property_type
FROM Photo ph
INNER JOIN Property p ON ph.property_ID = p.property_id
WHERE p.property_type IN ('Studio', 'One-Bedroom', 'Multi-Bedroom', 'Detached House')
AND p.price = (
    SELECT MAX(price)
    FROM Property
    WHERE property_type = p.property_type
)
ORDER BY p.property_type, p.price DESC;
```

from the database이므로 모든 부동산 정보가 작성되어 있는 property에서 작업을 수행하였다.
각각의 부동산 종류에 따라서 다음과 같이 코드를 작성할 수 있다.

```
SELECT ph.image_url, p.price
FROM Photo ph
INNER JOIN Property p ON ph.property_ID = p.property_id
WHERE p.property_type = 'Studio'
AND (p.property_type, p.price, p.listing_date) IN (
    SELECT property_type, MAX(price), listing_date
    FROM Property
    WHERE property_type = 'Studio'
)
ORDER BY p.price DESC
LIMIT 1;
```

f) Record the sale of a property that had been listed as being available. This entails storing the sales price, the buyer, the selling agent, the buyer's agent(if any), and the date.

:property_id 는 사용자가 입력한 property_id 값이다.
사용자 입력값이 해당 property에 기존에 등록된 것인지 확인하고 sold에 기록해야 한다.

```
INSERT INTO Sold (property_id, listing_date, seller_id, agent_id, buyer_id, sold_date,
sold_price)
VALUES (:property_id, :listing_date,
        (SELECT seller_id FROM seller WHERE seller_id = :seller_id),
        (SELECT agent_id FROM agent WHERE agent_id = :agent_id),
        (SELECT buyer_id FROM Buyer WHERE buyer_id = :buyer_id),
        DATE('now'), :sold_price)
WHERE EXISTS (
    SELECT 1
    FROM Property p
    WHERE p.property_id = :property_id
    AND p.listing_date = :listing_date
);
```

g) Add a new agent to the database.

```
INSERT INTO Agent (agent_id, name, contact_info)
VALUES (‘입력아이디', ’입력 이름', '입력 연락정보');
```