데이터 베이스 Project2

20191662 최찬우

<BCNF Decomposition>-Entity

1. Customer(ID, name, address, phone\_num,payment\_method)

Functional Dependency

1. ID->name,address,phone\_num, payment\_method

고객의 ID 속성의 값 각각에 대해 항상 고객의 이름, 주소, 전화번호, 지불 방법의 값이 오직 하나만 연관되어 있기 때문이다.

ID->name,address,phone\_num,payment\_method는

nontrivial한 functional dependency이므로

(ID)+가 Customer의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = ID,name,address,phone\_num,payment\_method (by 1)

이므로

Customer Entity는 BCNF이다.

2. Shipment(ID,date,status,cost)

Functional Dependency

1. ID->date,cost,status

배송 번호(ID) 속성의 값 각각에 대해 항상 배송 날짜(date), 배송 비용(cost), 배송 상태(status)의 값이 오직 하나만 연관되어 있기 때문이다.

ID->date,cost,status는 nontrivial한 functional dependency이므로

(ID)+가 Shipment의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = date,cost,status (by 1)

이므로

Shipment Entity는 BCNF이다.

3. Receipient(ID,name,address,phone\_num)

Functional Dependency

1. ID->name,address,phone\_num, payment\_method

수신자의 ID 속성의 값 각각에 대해 항상 수신자의 이름(name), 주소(address), 전화번호(phone\_num)의 값이 오직 하나만 연관되어 있기 때문이다.

ID->name,address,phone\_num는 nontrivial한 functional dependency이므로

(ID)+가 Receipient의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = ID,name,address,phone\_num (by 1)

이므로

Receipient Entity는 BCNF이다.

4. Vehicle(Vehicle\_number,type)

Functional Dependency

1. Vehicle\_number-> type

차량에 부여된 고유 번호(Vehicle\_number)속성의 값 각각에 대해 항상 차량 종류(type)의 값이 오직 하나만 연관되어 있기 때문이다.

Vehicle\_number-> type은 nontrivial한 functional dependency이므로

(Vehicle\_number)+가 Vehicle의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Vehicle\_number)+ = Vehicle\_number,type (by 1)

이므로

Vehicle Entity는 BCNF이다.

5. Service\_type(ID,Package Type,Weight,Timeliness)

Functional Dependency

1. ID-> Package Type,Weight,Timeliness

패키지의 여러 정보에 따라 이미 정해놓은 서비스 타입의 고유 번호(ID)속성의 값 각각에 대해 항상 포장지의 종류(Pacakge Type), 배송품의 무게(Weight), 배송 기한(Timeliness)의 값이 오직 하나만 연관되어 있기 때문이다

ID-> Package Type,Weight,Timeliness는 nontrivial한 functional dependency이므로

(ID)+가 Service\_type의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = Package Type,Weight,Timeliness (by 1)

이므로

Service\_type은 BCNF이다.

6. Package\_Info(Shipment ID, Package name, Description)

Functional Dependency

1. Shipment ID,Description -> Package name

Project1에서 작성한 ER-Model에 따르면, 각각의 배송에 대해 package의 정보를 저장해야하고 Shipment와 Package Info가 one to many이므로 다른 배송에 대해서는 다른 물건이 배송된다는 가정이 있었다. 이에 따르면 배송 번호(Shipment ID)와 배송 상품에 대한 설명(Description) 속성의 값 각각에 대해 항상 배송 물품의 이름(Package name)의 값이 오직 하나만 연관되어 있다.

이러한 가정하에 Shipment ID,Description -> Package name는 nontrivial한 functional dependency이므로 (Shipment ID,Description)+가 Package Info의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID,Description)+ = Package name,Description,Shipment ID

이므로

Package Info는 BCNF이다.

-----------------------------------------------------------------------------

<BCNF Decomposition>-Relationship

1. Cus\_Ship(Shipment ID, Customer ID)

Functional Dependency

1.Shipment ID->Customer ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 고객의 ID(Customer ID) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Customer ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Cus\_Ship의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Customer ID (by 1)

이므로

Cus\_ship은 BCNF이다.

2. Ship\_Service(Shipment ID, Service\_type ID)

Functional Dependency

1. Shipment ID->Service\_type ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 서비스 타입의 고유 번호(ID)속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Service\_type ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Ship\_Service의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Service\_type ID (by 1)

이므로

Ship\_Service는 BCNF이다.

3. Rec\_Ship(Shipment ID, Receipient ID)

Functional Dependency

1. Shipment ID->Receipient ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 수신자의 ID(Receipient ID) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Receipient ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Rec\_Ship의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Receipient ID (by 1)

이므로

Rec\_ship은 BCNF이다.

4. Delivery\_Vehicle(Shipment ID, Vehicle number)

Functional Dependency

1. Shipment ID-> Vehicle number

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 고유한 차량 번호(Vehicle number) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID-> Vehicle number는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Delivery\_Vehicle의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Vehicle number (by 1)

이므로

Delivery\_Vehicle은 BCNF이다.

5. Hazardous/Imternational(Shipment ID, Package name, Description)

Functional Dependency

1. Package name,Description -> Shipment ID

앞에서 Package Info에서 언급했던 것처럼 배송 번호(Shipment ID)와 배송 상품에 대한 설명(Description) 속성의 값 각각에 대해 항상 배송 물품의 이름(Package name)의 값이 오직 하나만 연관되어 있다.

이러한 가정하에 Shipment ID,Description -> Package name는 nontrivial한 functional dependency이므로 (Shipment ID,Description)+가 Package Info의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID,Description)+ = Package name,Description,Shipment ID

이므로

Hazardous/Imternational는 BCNF이다.

이를 바탕으로 logical Schema Diagram을 그리면 다음과 같으며 PRJ1에서와 동일하다.

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<Physical Diagram & OBDC Implementation>

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

1. Customer

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명ID: 고객이 회사의 서비스를 이용한 순간 부여되는 고유 번호(primary key)

-5자리로 고정된 ID이기 떄문에 char type으로 선언

name: 고객의 이름

-이름의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

address: 고객의 주소

-주소의 최대 글자만 20자로 정해주고 여러 주소의 길이를 수용할 수 있게 varchar type으로 선언

phone\_num: 고객의 핸드폰 번호

-배달의 민족처럼 고객 보호 번호를 위해 랜덤한 번호를 최대 10글자로 정의하기 위해 varchar type으로 선언

payment\_method: 지불 방식에 대한 속성(account number 또는 credit card가 들어감)

-지불 방법은 acccount number 또는 credit card라고 적히기 떄문에 varchar type으로 선언

2. Shipment

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명ID: 배송이 되는 순간 부여되는 고유한 배송 번호(primary key)

-5자리로 고정된 ID이기 떄문에 char type으로 선언

date: 배송 날짜

-23년 1월 1일이면 20230101의 형태로 저장하기 위해 char(8)로 선언

status: 배송 상태(배송 중, 배송 완료(제시간), 배송 완료(지연) 등의 정보가 들어감)

-배송 상태에 대한 여러 정보가 들어가기 떄문에 varchar type으로 선언

cost: 현재 배송에 지불되는 비용

- 비용에 대한 정보이므로 numeric type으로 선언

3. Receipient

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명ID: 고객이 특정 수신자에게 배송하겠다고 서비스를 이용한 순간 수신자에게 고유 번호가 부여됨(primary key)

-5자리로 고정된 ID이기 떄문에 char type으로 선언

name: 수신자의 이름

-이름의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

address: 수신자의 주소

-주소의 최대 글자만 20자로 정해주고 여러 주소의 길이를 수용할 수 있게 varchar type으로 선언

phone\_num: 수신자의 휴대폰 번호

-배달의 민족처럼 고객 보호 번호를 위해 랜덤한 번호를 최대 10글자로 정의하기 위해 varchar type으로 선언

4. Service\_type

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명ID: 패키지의 여러 정보에 따라 이미 정해놓은 서비스 타입의 고유 번호(primary key)

-5자리로 고정된 ID이기 떄문에 char type으로 선언

Package Type: 포장지의 종류

-포장 타입의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

Weight: 배송품의 무게

-무게의 tpye은 float type으로 선언하여 디테일한 측정이 가능하게 만듦

Timeliness: 배송 기한

-배송 기한은 overnight, second\_day 등의 정보가 들어가므로 varchar type으로 선언

5. Vehicle

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명Vehicle\_number: 각 차량의 차량 번호(primary key)

-차량의 번호는 회사에서 고유하게 지정해주며 5자리까지 가능하여 char type으로 선언

type: 차량 종류(truck, plane 등)

-차량 종류는 다양한 이름 길이를 가질 수 있으므로 varchar type으로 선언

6. Package\_Info

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명Package name: 배송 물품의 이름

-배송 물품의 이름은 다양할 수 있으므로 varchar type으로 선언

Description: 배송 물품의 구체적 정보

-구체적 정보 또한 국제배송, 위험 물질 등의 정보등이 적혀있어 그 길이가 다양할 수 있으므로 varchar type으로 선언

7. Cus\_ship

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명Cus\_ship은 Customer과 Shipment 간의 one to may relationship이므로 many 쪽의 Shipment의 primary key인 Shipment\_ID가 Cus\_ship의 primary key로 정의되고, Customer과 Shipment의 primary key는 Cus\_ship의 foreign key로 정의된다.

8. Rec\_ship

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명Rec\_ship은 Receipient와 hipment 간의 one to may relationship이므로 many 쪽의 Shipment의 primary key인 Shipment\_ID가 Rec\_ship의 primary key로 정의되고, Receipient와 Shipment의 primary key는 Cus\_ship의 foreign key로 정의된다.

.

9. Ship\_Service

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명Ship\_Service는 Shipment와 Service\_type의 many to one relationship이므로 many쪽의 Shipment의 primary key인 Shipment\_ID가 Ship\_Service의 primary key로 정의되고, Shipment와 Ship\_Service의 primary key는 Ship\_Service의 foreign key로 정의된다.

10. Delivery\_Vehicle

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명Delivery\_Vehicle은 Shipment와 Vehicle의 many to one 관계이므로 many 쪽의 Shipment의 primary key인 Shipment\_ID가 Delivery\_Vehicle의 priamary key로 정의되고, Shipment와 Vehicle의 primary key는 Delivery\_Vehicle의 foreign key로 정의된다.

<OBDC C language codes>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명1. MySQL 서버에 database file update

스크린샷이(가) 표시된 사진

자동 생성된 설명-> dbfiletxt.txt를 r 모드로 읽어와서 업로드한다.

스크린샷, 예술이(가) 표시된 사진

자동 생성된 설명

(dbfiletxt.txt 의 일부분 예시)

2. menu 창 구현

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명->code

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명->실행 결과

3. 각 type별 SQL 구문, 코드, 실행결과

(Type 1-1)

Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

“Delivery\_Vehicle set에서 Vehicle\_number(1721번)을 조회하면 해당 트럭의 배송 운행 이력으로도 볼 수 있는 Shipment ID들을 알 수 있고, 여기서 얻은 모든 Shipment ID들 중에 현재 배송 중인 status의 Shipment ID를 찾는다면, 고객의 정보를 찾을 수 있을 것이다”

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->실제 구현

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11001','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11002','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11003','10102');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11005','10103');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11007','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11009','10105');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11013','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11014','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment\_ID는 “11001, 11002, 11007, 11013, 11014”

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 운행중인 Shipment\_ID는 “11001, 11007”

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11001','00001');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11007','00004');

-> 이와 연관된 Customer\_ID는 “00001, 00004”

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00001', 'John Smith', '123 Main Street', 1111111111, 'Credit Card');

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00004', 'Emily Davis', '321 Elm Court', 4444444444, 'Credit Card');

->00001과 00004의 ID를 가지는 Customer의 이름은 “John Smith, Emily Davis”

따라서 실행 결과가 맞다.

(Type 1-2) -Type 1-1과 접근 방법 거의 동일

Assume truck 1721 is destroyed in a crash. Find all recipients who had a package on that truck at the time of the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

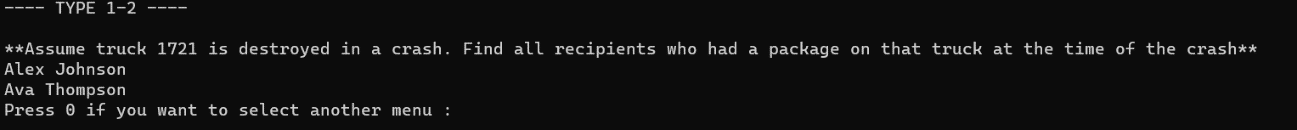
“Delivery\_Vehicle set에서 Vehicle\_number(1721번)을 조회하면 해당 트럭의 배송 운행 이력으로도 볼 수 있는 Shipment ID들을 알 수 있고, 여기서 얻은 모든 Shipment ID들 중에 현재 배송 중인 status의 Shipment ID를 찾는다면, 수신자의 정보를 찾을 수 있을 것이다”

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명->실제 구현

텍스트, 멀티미디어 소프트웨어, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11001','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11002','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11003','10102');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11005','10103');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11007','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11009','10105');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11013','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11014','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment\_ID는 “11001, 11002, 11007, 11013, 11014”

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 운행중인 Shipment\_ID는 “11001, 11007”

INSERT INTO Rec\_ship(Shipment\_ID, Receipient\_ID) VALUES('11001','22001');

INSERT INTO Rec\_ship(Shipment\_ID, Receipient\_ID) VALUES('11007','22004');

-> 이와 연관된 Receipient\_ID는 “22001, 22004”

INSERT INTO Receipient (Receipient\_ID, name, address, phone\_num) VALUES('22001', 'Alex Johnson', '567 Park Avenue', 1234567890);

INSERT INTO Receipient (Receipient\_ID, name, address, phone\_num) VALUES('22004', 'Ava Thompson', '1098 Vineyard Drive', 9998887777);

->22001과 22004의 ID를 가지는 Receipient의 이름은 “Alex Johnson, Ava Thompson”이다.

따라서 실행 결과가 맞다.

(Type 1-3)

Assume truck 1721 is destroyed in a crash. Find the last successful delivery by that truck prior to the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

Delivery\_Vehicle set에서 Vehicle\_number(1721번)을 조회하고 여기서 얻은 모든 Shipment ID들을 이용해 가장 최근 date의 Shipment ID를 찾는다면 관련 정보를 얻을 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명->실제 구현

텍스트, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11001','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11002','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11003','10102');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11005','10103');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11007','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11009','10105');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11013','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11014','1721');

INSERT INTO Delivery\_Vehicle(Shipment\_ID, Vehicle\_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment\_ID는 “11001, 11002, 11007, 11013, 11014”

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 Delivered된 Shipment\_ID는 “11002, 11013, 11014”

여기서 max(Shipment.date)는

23년6월1일인 20230601이므로 출력결과가 옳게 나오는 것을 알 수 있다.

(Type 2)

Find the customer who has shipped the most packages in the past year

prj1에서 설명하였던 방법(prj1 보고서 참고)

Cus\_ship set에는 Customer의 ID와 Shipment ID(primary key)가 저장되어 있다. 따라서 Cus\_ship set에서 각 Customer ID별 Shipment ID들을 조회하고, 해당 Shipment ID를 이용해 Shipment set에서 date가 지정된 과거 년도인 것만 찾은 다음 이를 모든 고객에 대해 반복하여 가장 많은 package를 배송한 고객의 ID를 찾을 수 있다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->실제 구현 내용

with 절에서 임시로 만든 relation F의 attribute는 FID와 NUM으로 이것이 의미하는 것은 특정 년도(위에서는 2022년)에 배송한 Customer의 ID와 몇 번 배송했는지를 나타낸다.

이를 통해 결국 알고 싶은 것은 고객의 이름이므로, Customer와 F를 join하여 Customer의 ID와 F의 ID가 같은 것을 찾으며, F의 NUM중 가장 큰 값을 찾으면 Customer의 이름을 알 수 있다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

프로그램에서는 연도를 입력받아야 특정 연도를 확인할 수 있으므로 with~ like ‘ 까지를 먼저 rquery에 집어넣고, 입력받은 연도의 타입을 sprintf을 이용해 문자열로 바꿔준 뒤 rquery에 strcat을 이용해 붙여주고, 다시 %’~끝까지 rquery에 붙여주어서 SQL 구문을 완성시키는 방법을 선택하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11016', '20220117', 'Delivered(on time)', 300.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11017', '20220118', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11018', '20220119', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11019', '20220120', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11020', '20220121', 'Delivered(on time)', 90.19);

->2022년도에 대한 Shipment\_ID

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11016','00003');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11017','00004');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11018','00004');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11019','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11020','00005');

->해당 Shipment\_ID와 연관된 Customer\_ID, 00004와 00005가 각 2번씩 가장 많이 배송을 한 것을 알 수 있다.

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00004', 'Emily Davis', '321 Elm Court', 4444444444, 'Credit Card');

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00005', 'David Wilson', '654 Pine Lane', 5555555555, 'Account');

->이와 연관된 Customer의 이름은 “Emily Davis, David Wilson”이며 2022년도에 대해서 맞다

마찬가지로

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11021', '20210122', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11022', '20210123', 'Delivered(on time)', 37.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11023', '20210124', 'Delivered(on time)', 67.19);

->2021년 Shipment\_ID를 찾고

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11021','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11022','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11023','00006');

->Customer의 ID를 찾으면 00005가 두 번으로 가장 많음을 알 수 있고

->00005의 이름은 앞에서 봤듯이 David Wilson이다.

따라서 실행 결과는 맞다

(Type 3)

Find the customer who has spent the most money on shipping in the past year

prj1에서 설명하였던 방법(prj1 보고서 참고)

Cus\_ship set에서 각 Customer ID별 Shipment ID들을 조회하고, 해당 Shipment ID를 이용해 Shipment set에서 date가 작년인것만 찾은 다음 그것들의 cost를 전부 더한다. 그리고 이를 모든 고객에 대해 반복하여 가장 비용이 큰 고객을 찾는다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->실제 구현 내용

2번과 거의 동일하게 with 절에서 임시로 만든 relation F의 attribute는 FID와 NUM으로 이것이 의미하는 것은 특정 년도(위에서는 2022년)에 배송한 Customer의 ID와 해당 Customer가 배송에 소비한 비용의 합을 나타낸다.

이를 통해 결국 알고 싶은 것은 고객의 이름이므로, Customer와 F를 join하여 Customer의 ID와 F의 ID가 같은 것을 찾으며, F의 NUM중 가장 큰 값을 찾으면 Customer의 이름을 알 수 있다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

sql query를 만드는 방식은 Type2와 완벽히 동일하여 설명을 생략함

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명->실행결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11016', '20220117', 'Delivered(on time)', 300.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11017', '20220118', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11018', '20220119', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11019', '20220120', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11020', '20220121', 'Delivered(on time)', 90.19);

->2022년도에 대한 Shipment\_ID

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11016','00003');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11017','00004');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11018','00004');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11019','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11020','00005');

->이를 통해 만들어지는 F는

|  |  |
| --- | --- |
| FID | NUM |
| 00003 | 300.19 |
| 00004 | 134.38 |
| 00005 | 157.38 |

따라서 가장 많은 비용을 낸 고객 ID는 00003

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00003', 'Michael Johnson', '789 Oak Road', 3333333333, 'Cash');

->그는 바로 Michael Johnson이며 실행 결과와 일치한다.

마찬가지로

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11021', '20210122', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11022', '20210123', 'Delivered(on time)', 37.19);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11023', '20210124', 'Delivered(on time)', 67.19);

->2021년 Shipment\_ID를 찾고

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11021','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11022','00005');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11023','00006');

->Customer의 ID를 찾으면 F는

|  |  |
| --- | --- |
| FID | NUM |
| 00005 | 104.38 |
| 00006 | 67.19 |

이므로 00005라는 ID를 가지는 고객이 가장 많이 소비하였으며

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00005', 'David Wilson', '654 Pine Lane', 5555555555, 'Account');

->이는 바로 David Wilson이다

따라서 실행 결과가 맞다

(Type 4)

Find those packages that were not delivered within the promised time

prj1에서 설명하였던 방법(prj1 보고서 참고)

Shipment Set에서 status가 배송 완료(지연)과 관련된 정보를 찾으면 된다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명->실제 구현 내용

Package\_Info natural join left outer join을 하면 Package\_Info의 Shipment\_ID을 기준으로 Shipment과 일치하는 행들만의 테이블이 형성된다. 그러한 테이블에서 배송완료(지연)이라는 조건을 만족하는 행을 찾으면 그것에 해당하는 Package의 이름을 알 수 있다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 1', 'International', '11001');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 1', 'Hazardous', '11001');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 3', 'Normal', '11003');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 4', 'International', '11004');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 5', 'Hazardous', '11005');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 5', 'Normal', '11005');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 7', 'International', '11007');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 8', 'Hazardous', '11008');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 9', 'Normal', '11009');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 10', 'International', '11010');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 11', 'Normal', '11014');

INSERT INTO Package\_Info (Package\_name, Description, Shipment\_ID) VALUES('Package 12', 'Normal', '11002');

-> 배송 상품이 실린 Shipment\_ID는 11001,11002,11003,11004,11005,11007,11008,11010,11014이다.

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11003', '20230603', 'In Transit', 60.25);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11004', '20230604', 'Delivered(on time)', 85.75);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11005', '20230605', 'In Transit', 70.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11008', '20230608', 'Delivered(on time)', 55.50);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11010', '20230410', 'Delivered(delayed)', 95.75);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->이와 관련된 Shpment들만 찾아보았을 때 Delivered(delayed)라고 적혀있는 Shipment\_ID는 11002, 11010, 11014 이며 이와 연관된 Package name은 Package10, Package11, Package12이다.

따라서 실행결과가 맞다

(Type 5)

Generate the bill for each customer for the past month. Consider creating several types of bills

prj1에서 설명하였던 방법(prj1 보고서 참고)

Simple bill(고객 별 소비 비용)

-Cus\_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Shipment set에서 cost를 전부 조회하고 이를 더하여 bill을 청구할 수 있음

service type 별 비용

-Cus\_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Ship\_Service set에서 각 Shipment ID에 대한 Service\_type를 알아내면 Service type별 비용을 청구할 수 있음

각 배송 별 비용

-Cus\_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Shipment set에서 각 Shipment ID 별 cost를 bill로 청구할 수 있음

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명->Simple bill 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table을 이용하여 고객의 ID 기준으로 cost를 합치며 이를 group by를 이용해 묶어주고 F라는 임시 relation을 만든다.(고객의 ID, 고객이 배송한 배송에 대한 비용의 총 합)

Simple bill에서 알고 싶은 내용은 고객의 이름, 고객의 주소, 고객의 주문 비용이므로 Customer과 F를 join하여 두 relation의 ID가 같으면 값을 출력해준다.

\*20230 -> 202305 (코드에서는 20230 이후 입력받은 month 데이터를 붙여주어서 20230으로 적어놓았지만, MySQL에서 실행할떄는 202305, 202304처럼 특정 month를 붙여주어야함)

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->A bill listing charges by type of service 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table과 Ship\_Service를 이용하여 고객의 배송 번호와 Ship\_Service의 배송번호(즉 특정 서비스를 이용하는 배송번호)가 일치하면 F라는 임시 relation을 만든다.(고객 ID, 고객이 주문한 배송이 이용한 서비스 타입 ID, 해당 배송에 대한 COST)

A bill listing charges by type of service에서 알고 싶은 내용은 고객이 배송에서 이용한 서비스 별로 비용을 청구하는 것이므로, Customer와 F를 이용하여 두 개의 ID가 같은 것에 대해 Customer의 이름,서비스 type ID에 대해 group by를 시켜준다.

\*20230 -> 202305



텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->An itemize billing listing each individual shipment and the charges for it 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table을 이용하여 F라는 임시 relation을 만든다.(고객 ID, 고객이 주문한 배송 ID, 해당 배송에 대한 COST)

An itemize billing listing each individual shipment and the charges for it에서 알고 싶은 내용은 고객이 배송한 배송번호 별로 비용을 청구하는 것이므로, Customer와 F를 이용하여 두 개의 ID가 같은 것에 대해 Customer의 이름과 Shipment\_ID(F.FTID)에 대해 group by를 시켜준다.

\*20230 -> 202305

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명->code

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명sql query를 만드는 방법은 Type2와 완전히 동일하여 생략

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

<5월의 Shipment ID>

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11024', '20230510', 'Delivered(delayed)', 10.75);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11025', '20230510', 'Delivered(delayed)', 11.75);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11026', '20230510', 'Delivered(delayed)', 12.75);

INSERT INTO Shipment (Shipment\_ID, date, status, cost) VALUES('11027', '20230510', 'Delivered(delayed)', 12.75);

<해당 Shipment ID에 대한 Customer ID>

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11024','00001');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11025','00001');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11026','00002');

INSERT INTO Cus\_ship(Shipment\_ID, Customer\_ID) VALUES('11027','00003');

<해당 Customer ID에 대한 Customer name>

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00001', 'John Smith', '123 Main Street', 1111111111, 'Credit Card');

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00002', 'Jane Doe', '456 Maple Avenue', 2222222222, 'Account');

INSERT INTO Customer (Customer\_ID, name, address, phone\_num, payment\_method) VALUES('00003', 'Michael Johnson', '789 Oak Road', 3333333333, 'Cash');

<해당 Shipment ID에 대한 Service\_type\_ID>

INSERT INTO Ship\_Service(Shipment\_ID, Service\_type\_ID) VALUES('11024','20001');

INSERT INTO Ship\_Service(Shipment\_ID, Service\_type\_ID) VALUES('11025','20002');

INSERT INTO Ship\_Service(Shipment\_ID, Service\_type\_ID) VALUES('11026','20003');

INSERT INTO Ship\_Service(Shipment\_ID, Service\_type\_ID) VALUES('11027','20003');

이걸 바탕으로

1. Simple bill

<해당 Shipment ID에 대한 Customer ID>참고

-John Smith: Shipment\_ID 11024, 11025이므로 해당 Shipment tuple의 cost 더하면 22.5

-Jane Doe: Shipmnet\_ID 11026이므로 해당 Shipment tuple의 cost 더하면 12.75

-Michael Johnson: Shipment\_ID 11027이므로 해당 Shipment tuple의 cost 더하면 12.75

\*주소는 확인해보면 잘 나오고 있음을 자명하게 알 수 있음

2. A bill listing charges by type of service

<해당 Shipment ID에 대한 Service\_type\_ID>참고

-John Smith: Shipment\_ID 11024, 11025이므로 Service\_type\_ID는 20001, 20002이며

각 service\_type과 연관된 Shipment cost별로 출력이 나오고 있음

-Jane Doe: Shipmnet\_ID 11026이므로 Service\_type\_ID는 20003이며

각 service\_type과 연관된 Shipment cost별로 출력이 나오고 있음

-Michael Johnson: Shipment\_ID 11027이므로 Service\_type\_ID는 20003이며

각 service\_type과 연관된 Shipment cost별로 출력이 나오고 있음

3. An itemize billing listing each individual shipment and the charges for it

<해당 Shipment ID에 대한 Customer ID> 참고

-John Smith: Shipment\_ID 11024, 11025이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

-Jane Doe: Shipmnet\_ID 11026이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

-Michael Johnson: Shipment\_ID 11027이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

+Type 5에 대한 추가 실행 결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명(4월)

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명(3월)

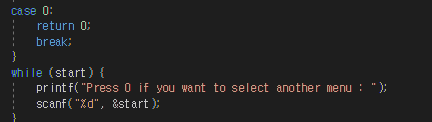
텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명(2월)

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명(1월)

(Type 0 and 결과값 출력 후 0입력 구현)





->code

Select Query Type에서 0번을 누르면 프로그램이 종료되게 구현하였다. (빨간색 원)

Query 결과가 나오면 0을 누를 때 까지 다시 Select Query Type으로 돌아가지 못하게 while(start){~~}을 통해 구현하였다.(노란색 원)

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명