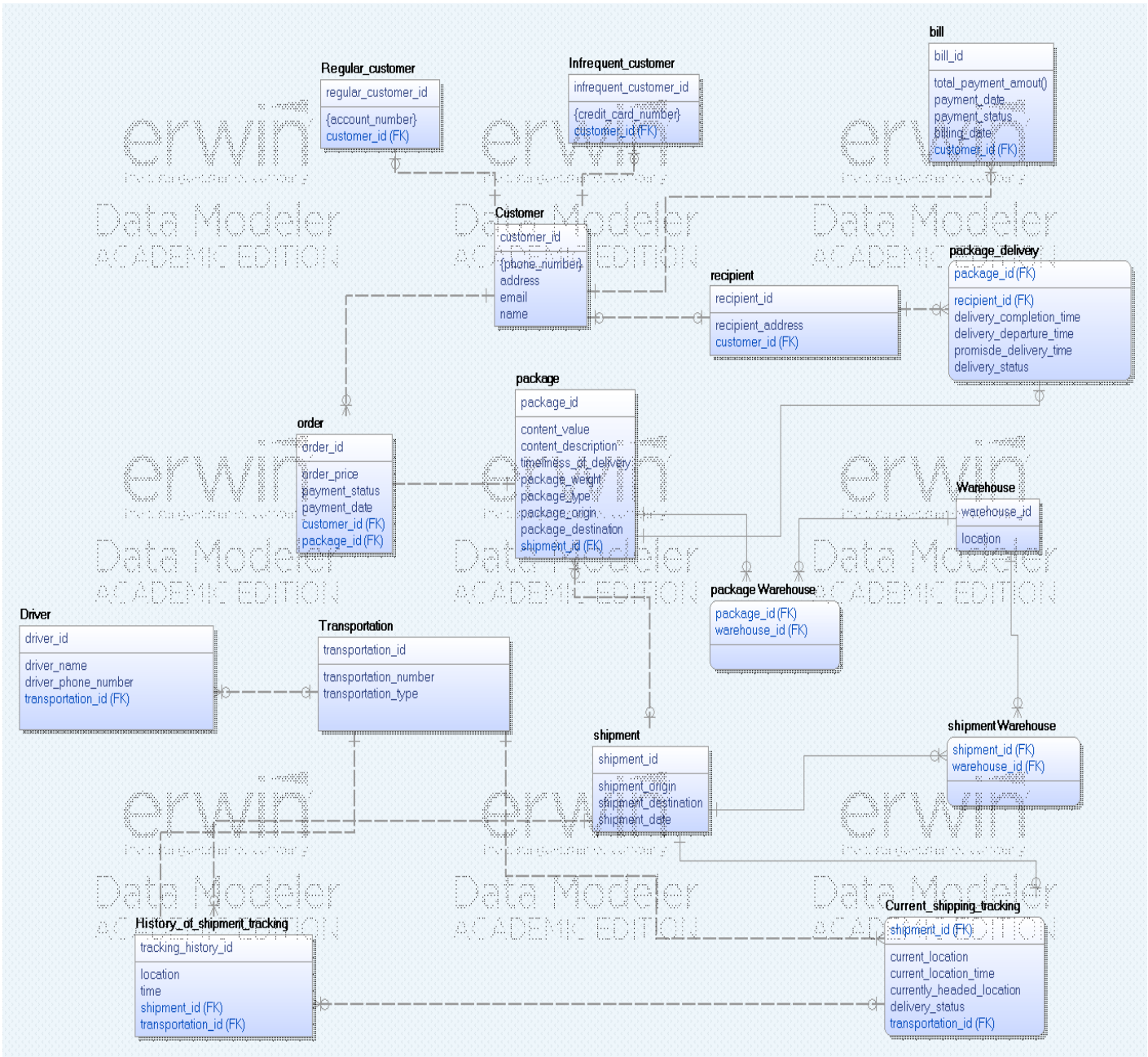
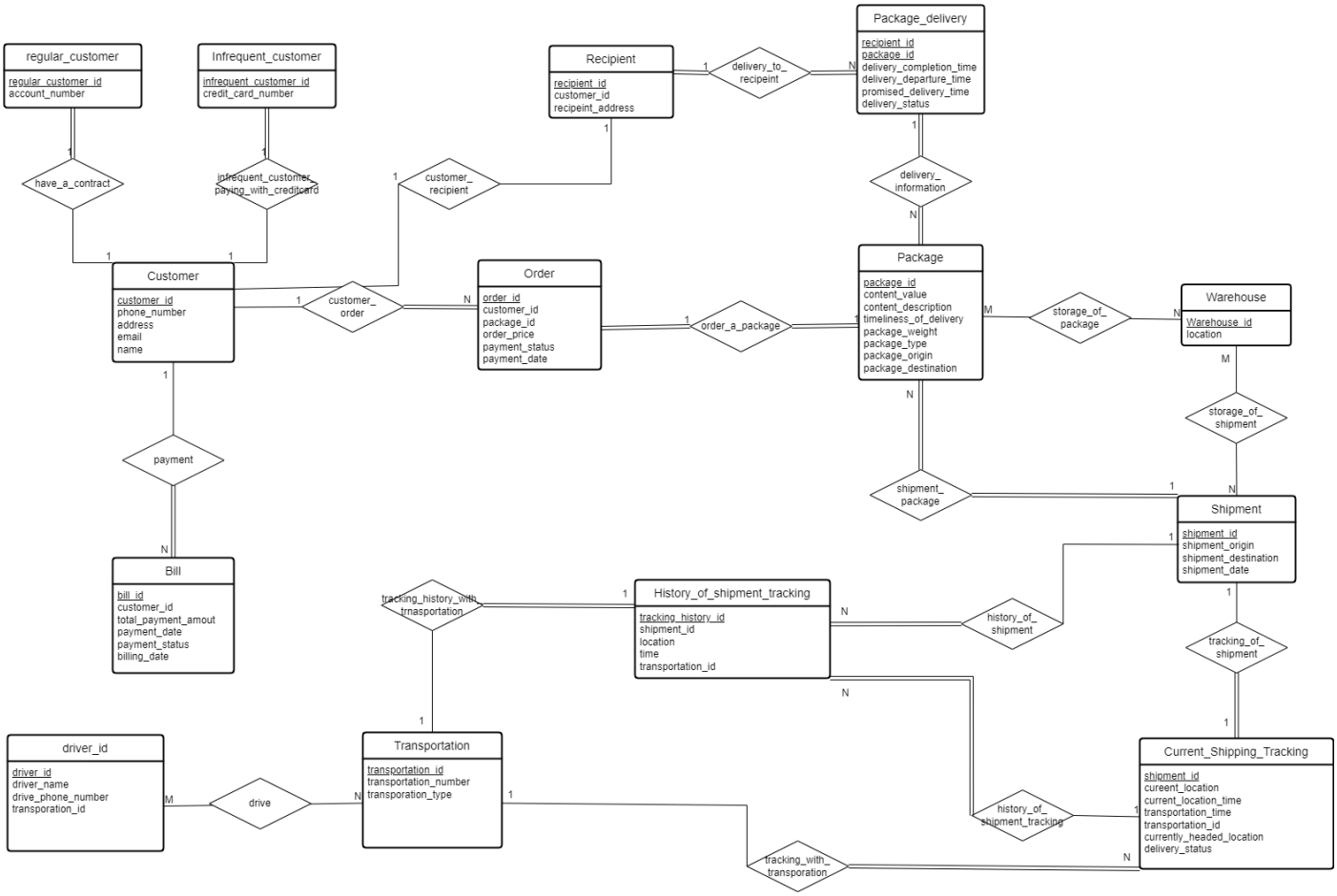
DB PROJECT1 REPORT

20200901 이효주

<1> Decomposed Logical Schema Diagram





위는 PROJECT1에서 구성한 Relation을 BCNF에 따라 decompose하였고, logical schema diagram과 ER diagram을 작성하였다.

Project1의 Relation들을 BCNF에 따라 어떻게 분해하였는지 살펴보면, 먼저 BCNF의 RULE은 아래와 같다.

A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F+ of the form

$\alpha \rightarrow \beta$

where α, β in R, at least one of the following holds:

- **<1> $\alpha \rightarrow \beta$ is trivial**
- **<2> α is a superkey for R**

<Functional Dependency>

1) Customer: <2> α is a superkey for R

customer_id(PK) -> phone_number, address, email, name

이 Relation에서는 고객 ID인 customer_id가 주어졌을 때, 해당 고객의 전화번호(phone_number), 주소(address), 이메일(email), 이름(name)이 결정된다.

2) Infrequent customer: <2> α is a superkey for R

infrequent_customer_id(PK) -> account_number, customer_id(FK)

가끔 이용하는 고객 ID인 infrequent_customer_id가 주어졌을 때, 해당 고객의 계좌 번호(account_number)와 customer_id가 결정된다.

3) Regular customer: <2> α is a superkey for R

regular_customer_id(PK) -> credit_card_number, customer_id(FK)

자주 이용하는 고객 ID인 regular_customer_id가 주어졌을 때, 해당 고객의 신용카드 번호(credit_card_number)와 customer_id가 결정된다.

4) Bill: <2> α is a superkey for R

bill_id(PK) -> customer_id(FK), total_payment_amount, payment_date, payment_status, billing_date

Bill Relation에서는 청구서 ID인 bill_id가 주어졌을 때, 해당 청구서의 고객 ID(customer_id), 총 지불 금액(total_payment_amount), 지불 날짜(payment_date), 지불 상태(payment_status), 청구 날짜(billing_date)가 결정된다.

5) Order: <2> α is a superkey for R

order_id(PK) -> customer_id(FK), package_id(FK), order_price, payment_status, payment_date

Order Relation에서는 주문 ID인 order_id가 주어졌을 때, 해당 주문의 고객 ID(customer_id), 소포 ID(package_id), 주문 가격(order_price), 지불 상태(payment_status), 지불 날짜(payment_date)가 결정된다.

6) Warehouse: <2> α is a superkey for R

warehouse_id(PK) -> location

Warehouse Relation에서는 창고 ID인 warehouse_id가 주어졌을 때, 해당 창고의 위치(location)가 결정된다.

7) Package: <2> α is a superkey for R

package_id(PK) -> content_value, content_description, timeliness_of_delivery, package_weight, package_type, package_origin, package_destination, shipment_id

Package Relation에서는 소포 ID인 package_id가 주어졌을 때, 해당 소포의 내용물 가치(content_value), 내용물 설명(content_description), 배송 시간(timeliness_of_delivery), 소포 무게(package_weight), 소포 유형(package_type), 소포 출발지(package_origin), 소포 목적지(package_destination)와 소포가 속한 운송 ID(shipment_id)가 결정된다.

8) Shipment: <2> α is a superkey for R

shipment_id(PK) -> shipment_origin, shipment_destination, shipment_date

Shipment Relation에서는 운송 ID인 shipment_id가 주어졌을 때, 해당 운송의 출발지(shipment_origin), 목적지(shipment_destination), 운송 날짜(shipment_date)가 결정된다.

9) package_Warehouse: <1> $\alpha \rightarrow \beta$ is trivial

warehouse_id(PK, FK), package_id(PK,FK) -> warehouse_id(PK, FK), package_id(PK,FK)

Package_Warehouse Relation에서는 창고 ID(warehouse_id)와 소포 ID(package_id)를 기반으로 해당 창고에서 보관되는 소포의 관계를 나타낸다.

10) shipment_Warehouse: <1> $\alpha \rightarrow \beta$ is trivial

shipment_id(PK, FK), package_id(PK,FK) -> shipment_id(PK, FK), package_id(PK,FK)

Shipment_Warehouse Relation에서는 운송 ID(shipment_id)와 창고 ID(warehouse_id)를 기반으로 해당 운송에서 사용되는 창고의 관계를 나타낸다.

11) Recipient: <2> α is a superkey for R

recipient_id(PK) -> customer_id(FK), recipient_address

Recipient Relation에서는 수령인 ID(recipient_id)가 주어졌을 때, 해당 수령인의 고객 ID(customer_id)와 수령인 주소(recipient_address)가 결정된다.

12) Package_delivery: <2> α is a superkey for R

recipient_id(PK, FK), package_id(PK) -> delivery_completion_time, delivery_departure_time, promised _delivery_time, , delivery_status

Package_Delivery Relation에서는 수령인 ID(recipient_id)와 소포 ID(package_id)를 기반으로 해당 소포의 배송 완료 시간(delivery_completion_time), 배송 출발 시간(delivery_departure_time), 약속된 배송 시간(promised_delivery_time), 배송 상태(delivery_status)가 결정된다.

13) Current_Shipping_Tracking: <2> α is a superkey for R

shipment_id(PK, FK) -> current_location, current_location_time, transportation_id(FK), currently_ headed_location, delivery_status

Current_Shipping_Tracking Relation에서는 운송 ID(shipment_id)를 기반으로 현재 위치(current_location), 현재 위치 시간(current_location_time), 운송수단 ID(transportation_id), 현재 향하는 위치(currently_headed_location), 배송 상태(delivery_status)가 결정된다.

14) History_of_Shipment_Tracking: <2> α is a superkey for R

tracking_history_id(PK) -> shipment_id(FK), location, time, transportation_id(FK)

History_of_Shipment_Tracking Relation에서는 추적 이력 ID(tracking_history_id)가 주어졌을 때, 해당 추적 이력의 운송 ID(shipment_id), 위치(location), 시각(time), 그리고 운송수단 ID(transportation_id)가 결정된다.

15) Transportation: <2> α is a superkey for R

transportation_id(PK) -> transportation_number, transportation_type

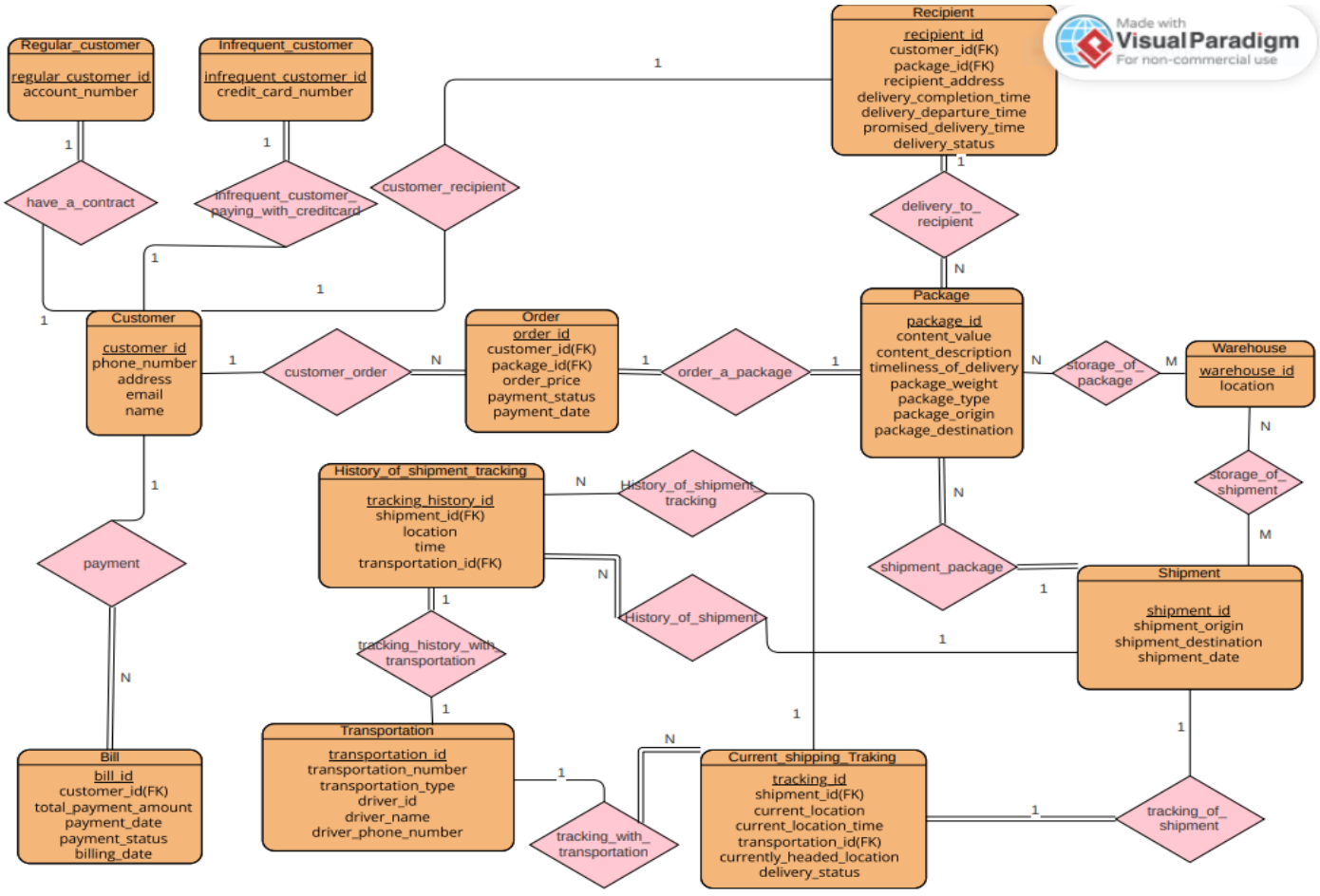
Transportation Relation에서는 운송수단 ID(transportation_id)가 주어졌을 때, 해당 운송수단의 운송수단 번호(transportation_number)와 운송수단 유형(transportation_type)이 결정된다.

16) Driver: <2> α is a superkey for R

driver_id(PK) -> driver_name, driver_phone_number, transportation_id(FK)

Driver Relation에서는 운전자 ID(driver_id)가 주어졌을 때, 해당 운전자의 이름(driver_name), 전화번호(driver_phone_number), 그리고 운송수단 ID(transportation_id)가 결정된다.

<ER Model in PROJECT1>



위는 Project1의 E-R Model이고, Project1과 크게 달라진 점은 2가지가 존재한다.

[1] Recipient Relation

Recipient_id(PK), customer_id, package_id, recipient_address, delivery_completion_time, delivery_departure_time, promised_delivery_time, delivery_status

이전의 Recipient relation은 위와 같다. (in PROJECT1)

이전의 경우 문제점이, 한 명의 Recipient가 여러 번 delivery를 받을 경우, customer_id(만약 이 회사에 customer로 order을 했을 경우 저장되는 정보)와 recipient_address가 여러 번 중복이 되어 나타나고, 여러 번 저장되는 문제점이 발생한다.

여기서는 2 가지의 functional dependency가 존재하는데,

1. recipient_id(PK)-> customer_id(FK), recipient_address, delivery_completion_time, delivery_departure_time, promised_delivery_time, delivery_status
2. package_id(FK)
   -> delivery_completion_time, delivery_departure_time, promised_delivery_time, delivery_status

BCNF를 만족하려면,

- **<1> $\alpha \rightarrow \beta$ is trivial**
- **<2> α is a superkey for R**

위 2가지 조건 중 하나를 만족해야하는데, 여기서 superkey는 {recipient_id}로, 주어진 Relation에서 식별 가능한 primary key이다.

package_id(FK) -> delivery_completion_time, delivery_departure_time, promised_delivery_time, delivery_status

이 functional dependency에서 {package_id}가 super key에 속하지 않으므로 BCNF를 만족하지 않는 것을 알 수 있기에, 따라서 위의 relation을 decompose하였다.

- **Package_delivery:**

package_id(PK, FK), delivery_completion_time, delivery_departure_time, promised_delivery_time, delivery_status, recipient_id(FK)

- **Recipient:**

recipient_id(PK), customer_id(FK), recipient_address

위와 같이 Relation을 분해하면 BCNF를 만족하게 된다. Recipient는 수령인과 관련된 정보를 포함하고 있으며, Package_delivery는 수령인과 소포에 대한 정보를 포함하고 있다.

[2] Transportation

transportation_id, transportation_number, transportation_type, driver_id, driver_name, driver_phone_name

이전의 Recipient relation은 위와 같다. (in PROJECT1)

이전의 경우 문제점이, 한 transportation에 여러 명의 driver가 존재할 수 있기에 driver의 정보가 중복으로 나타날 수 있다. (transporation_id는 임의로 설정한 indexing용 ID이고, transporation_number이 차 번호, driver_id가 운전사 용 ID이다.)

여기서는 2 가지의 functional dependency가 존재하는데,

1. transporation_id(PK)-> transportation_number, transportation_type, driver_id, driver_name, driver_phone_name
2. driver_id -> driver_name, driver_phone_name

BCNF를 만족하려면,

• **<1>** $\alpha \rightarrow \beta$ **is trivial**

• **<2>** α **is a superkey for R**

위 2가지 조건 중 하나를 만족해야하는데, 여기서 superkey는 {transporation_id}로, 주어진 Relation에서 식별 가능한primary key이다.

driver_id -> driver_name, driver_phone_name

이 functional dependency에서 {driver_id }가 super key에 속하지 않으므로 BCNF를 만족하지 않는 것을 알 수 있기에, 따라서 위의 relation을 decompose하였다.

- **Transportation:**

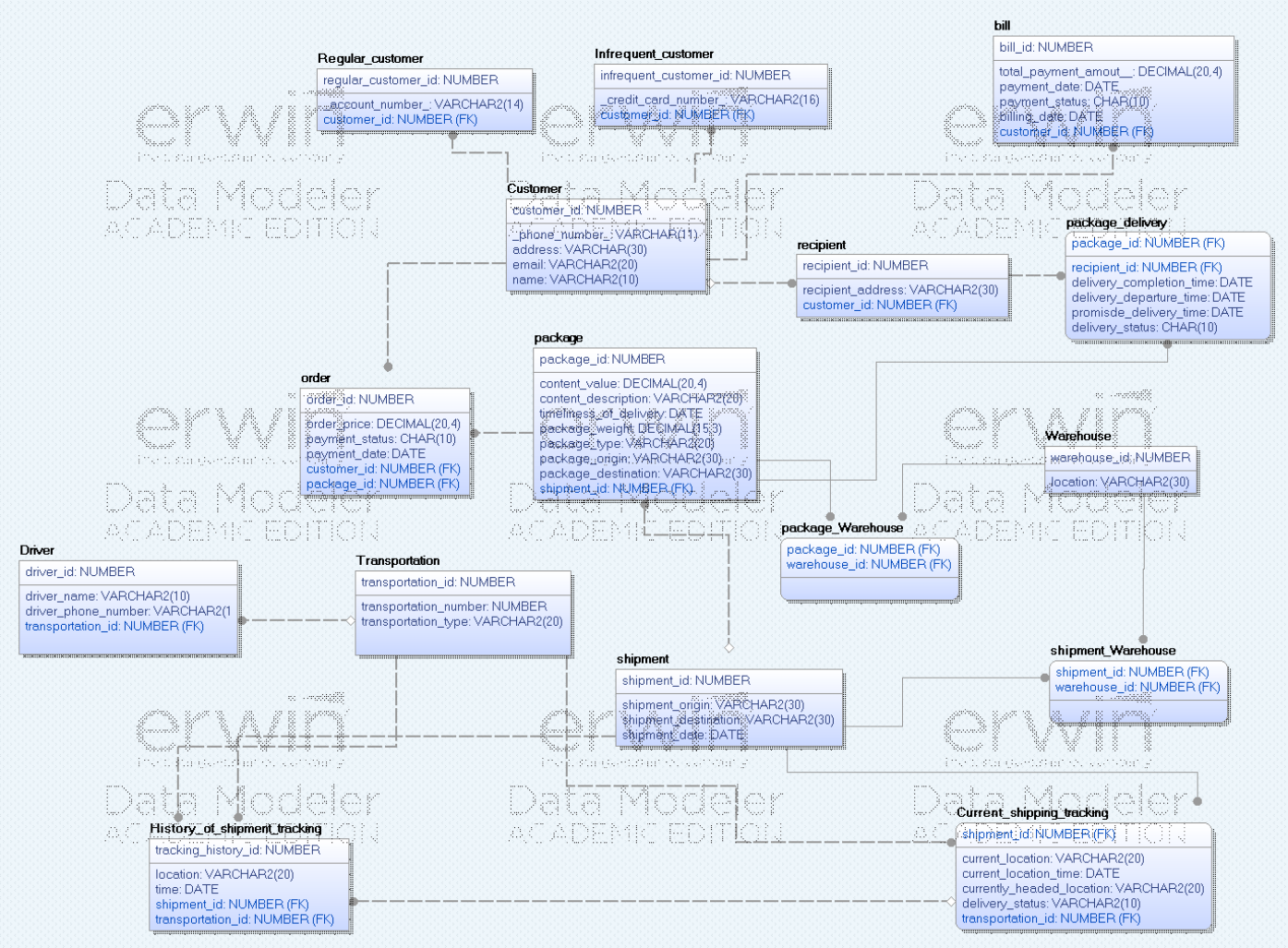
transportation_id(PK), transportation_number, transportation_type

- **Driver:**

driver_id(PK), driver_name, driver_phone_name, transportation_id(FK)

위와 같이 Relation을 분해하면 BCNF를 만족하게 된다. Transportation은 운송 수단 정보를 포함하고 있으며, Driver는 운송인과 운송인이 운송하는 수단에 대한 정보를 포함하고 있다.

<2> Physical Schema Diagram



1) Customer

customer_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

phone_number

- data type은 VARCHAR(11)로 설정하였고(휴대전화는 대개 10~11자리 이므로), 고객 관리 상 휴대전화 정보는 꼭 필요하다고 생각하여 Null값을 가질 수 없도록 하였다.

address

- data type은 VARCHAR(30)로 설정하였고, 주소도 package의 픽업 위치가 되거나 recipient의 주소가 될 수 있으므로 정보를 저장해 두는 것이 좋다고 생각하여 Null값을 가질 수 없도록 하였다.

email

- data type은 VARCHAR2(20)로 설정하였고, 이메일은 휴대전화 이외 추가적인 선택 정보라 생각하여 Null값을 가질 수 있도록 하였다.

name

- data type은 VARCHAR2(10)로 설정하였고, 이름은 Null값을 가질 수 없도록 설정하였다. 여기서 이름은 개인 고객의 이름이 될 수 있고, 한 회사의 이름이 될 수도 있을 것이다.

2) Infrequent customer

infrequent_customer_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

credit_card_number

- data type은 VARCHAR2(16)로 설정하였고, infrequent customer은 인터넷이나 전화 등을 통하여 신용 카드로 결제하기 때문에 신용카드 번호에 Null값을 가질 수 없도록 하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고, data type은 NUMBER이다. 관계는 zero or one의 one to one relationship으로 생성하였다. 종종 사용하는 고객에 대한 정보는 customer table에 존재해야 하기 때문에 Null 값은 허용하지 않았다.

3) Regular customer

regular_customer_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

account_number

- data type은 VARCHAR2(14)로 설정하였고, regular customer은 한 달 간의 주문 내역을 계좌 번호로 청구 받기 때문에 Null값을 가질 수 없도록 하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고, data type은 NUMBER이다. 관계는 customer와 zero or one의 one to one relationship으로 생성하였다. 정기 고객에 대한 정보는 customer table에 존재해야 하기 때문에 Null 값은 허용하지 않았다.

4) Bill

bill_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

total_payment_amount

- data type은 DECIMAL(20, 4)로 설정하였고, 한 달 간의 모든 주문에 대한 금액이 측정될 것이므로 Null값을 가질 수 없도록 하였다. 만약 한달 동안 주문을 하지 않은 경우에는 0원이라는 금액이 책정될 것이다.

payment_date

- data type은 DATE로 설정하였고, 청구 금액에 대해 이 날에 지불을 하였다 라는 정보를 확인할 수 있으므로 아직 지불을 하지 않은 경우에는 지불일이 없을 수 있으므로 Null값을 허용하였다.

payment_status

- data type은 CHAR(10)으로 설정하였고, 고객의 유형에 따라 지불을 이미 완료했을 수도 있고, 아직 지불 이전 등의 상태를 지닐 수 있을 것이다. 따라서 Null값을 가질 수 없도록 하였다.

billing_date

- data type은 DATE로 설정하였고, 청구일을 의미한다. 예를 들어, 정기 고객이라면 매달 청구일이 똑같을 것이고, 종종 사용하는 고객이라면 청구서를 요청한 날이 될 것이다. 따라서 Null값을 가질 수 없도록 하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고 data type은 NUMBER이다. 하나의 customer에 여러 bill을 가질 수 있는 zero, one or more의 one to many의 관계이고, bill에 있는 customer_id는 Null값을 가질 수 없도록 하였다.

5) Order

order_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

order_price

- datya type은 DECIMAL(20, 4)로 설정하였고, 고객이 주문을 할 때, 한 주문에 대한 금액이 바로 책정될 것이므로

Null값을 가질 수 없도록 하였다.

payment_status

- data type은 CHAR(10)로 설정하였고, 고객의 유형에 따라 한 주문에 대한 지불을 이미 완료했을 수도 있고, 아직 지불 이전 등의 상태를 지닐 수 있을 것이다. 따라서 Null값을 가질 수 없도록 하였다.

payment_date

- data type은 DATE로 설정하였고, 하나의 주문 금액에 대해 이 날에 지불을 하였다 라는 정보를 확인할 수 있으므로 아직 지불을 하지 않은 경우에는 지불일이 없을 수 있으므로 Null값을 허용하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고, data type은 NUMBER이다. 하나의 customer가 여러 주문을 할 수 있으므로 zero, one or more의 one to many 관계로 설정하였고, 주문이 들어간 고객에 대한 정보는 반드시 customer table에 존재해야 하므로 Null값을 허용하지 않았다.

package_id(FK)

- package table의 primary key인 package_id를 참조하고 있고, data type은 NUMBER이다. 하나의 주문 당 하나의 package가 발생한다고 가정하였기 때문에 cardinality value가 1인 one to one의 관계를 설정하였다. 또한 order table에 존재하는 어떤 package를 주문하였는지에 관한 정보는 package table에 반드시 존재해야 하므로 Null값을 허용하지 않았다.

6) Warehouse

warehouse_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

Location

- data type은 VARCHAR2(30)로 설정하였고, 창고에 관한 위치는 Null값을 가질 수 없도록 하였다.

7) Package

package_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

content_value

- data type은 DECIMAL(20, 4)로 설정하였고, 세관 신고가 필요하지 않더라도 이 package가 1~5만원 혹은 5~10만원의 가격대이다 라는 정보를 저장하고 있는 것이 운송회사의 측면에서 좋다고 생각하여 Null값을 허용하지 않았다.

content_description

- 어떤 package인지 설명할 수 있는 속성으로, data type은 VARCHAR2(20)로 설정하였고, 세관 신고가 필요할 경우 혹은 특수한 경우 작성할 수 있도록 Null값을 허용하였다.

timeliness_of_delivery

- Data type은 DATE로 설정하였고, Null 값은 허용하지 않았다. Delivery의 우선순위를 날짜로 제한하였고, 약속된 배송시간(day)을 통해 우선순위를 설정할 수 있기에 Null값을 허용하지 않았다.

package_weight

- Data type은 DECIMAL(15, 3)으로 설정하였고, package의 무게도 주문 금액이나 운송수단 등에 영향을 미치는 요소로, Null값을 허용하지 않았다.

package_type

- Data type은 VARCHAR2(20)로 설정하였고, package의 type도 주문 금액이나 운송수단 등에 영향을 미치는 요소이고, service type으로 package를 구분할 수 있도록 Null값을 허용하지 않았다.

package_origin

- Data type은 VARCHAR2(30)로 설정하였고, package의 픽업 위치로 반드시 존재해야 하는 정보이기 때문에 Null값을 허용하지 않았다.

package_destination

- Data type은 VARCHAR2(30)로 설정하였고, package의 배달 위치로 반드시 존재해야 하는 정보이기 때문에 Null값을

허용하지 않았다.

Shipment_id(FK)

- Shipment table의 primary key인 shipment_id를 참조하고 있고, data type은 NUMBER이다. 하나의 shipment에는 여러 package가 포함될 수 있으므로 one to many의 관계로 설정하였다. 그리고 아직 픽업을 하지 않은 경우 shipment에 포함되지 않을 수 있으므로 Null값을 허용하였다.

8) Shipment

shipment_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

shipment_origin

- data type은 VARCHAR2(30)로 설정하였고, 하나의 shipment가 어디에서 출발하였는지(예를 들자면 창고, 혹은 다량의 주문이 들어온 한 회사 등)를 알 수 있도록 Null값을 허용하지 않았다.

shipment_destination

- data type은 VARCHAR2(30)로 설정하였고 하나의 shipment가 어디로 향하는지(예를 들자면 창고, 혹은 회사, 특정 구역 등)를 알 수 있도록 Null값을 허용하지 않았다.

shipment_date

- data type은 DATE로 설정하였고, 언제 배송이 이루어졌는지 알 수 있는 속성이다. 다만 여러 package들로 shipment가 구성은 되었으나 아직 출발하지 않은 경우를 고려해 Null값을 허용하였다.

9) package_Warehouse

Package_id(PK, FK)

- primary key이며, package table의 primary key인 package_id를 참조하고 있고, data type은 NUMBER이다. 따라서 Null 값을 허용하지 않았다.

Warehouse_id(PK, FK)

- primary key이며, warehouse table의 primary key인 warehouse_id를 참조하고 있고, data type은 NUMBER이다. 따라서 Null값을 허용하지 않았다, Package_warehouse table의 경우 원래 many to many 관계였던 package와 warehouse의 관계를 table을 생성하여 many to many 관계를 바꿔주었다.

10) shipment_Warehouse

shipment_id(PK, FK)

- primary key이며, shipment table의 primary key인 shipment_id를 참조하고 있고, data type은 NUMBER이다. 따라서 Null 값을 허용하지 않았다.

Warehouse_id(PK, FK)

- primary key이며, warehouse table의 primary key인 warehouse_id를 참조하고 있고, data type은 NUMBER이다. 따라서 Null값을 허용하지 않았다. shipment_warehouse table의 경우 원래 many to many 관계였던 shipment와 warehouse의 관계를 table을 생성하여 many to many 관계를 바꿔주었다.

11) Recipient

recipient_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값은 가질 수 없도록 하였다.

recipient_address

- data type은 VARCHAR2(30)로 설정하였고, Null값은 가질 수 없도록 하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고, data type은 NUMBER이다. recipient가 customer일 수도 있지만, 수령만 하고, 주문을 한 고객이 아닐 수도 있는 경우를 고려하여 Null값을 허용하였다. 그리고 한 명의 고객은 한 명의 recipient를 의미할 수 있으므로 zero to one의 one to one 관계를 설정하였다.

12) Package_delivery

package_id(PK, FK)

- primary key이며, package table의 primary key인 package_id를 참조하고 있고, data type은 NUMBER이다. 따라서 Null 값을 허용하지 않았다.

recipient_id(FK)

- Recipient table의 primary key인 recipient_id를 참조하고 있고, data type은 NUMBER이다.

delivery_completion_time

- data type은 DATE으로 설정하였고, 아직 배송이 완료되지 않은 경우를 고려해 Null 값을 허용하였다.

delivery_departure_time

- data type은 DATE으로 설정하였고, 아직 배송이 출발되지 않은 경우를 고려해 Null 값을 허용하였다.

promised_delivery_time

- data type은 DATE으로 설정하였고, 약속된 배송시간은 Null값을 허용하지 않았다.

delivery_status

- data type은 CHAR(10)로 설정하였고, 배송 전/중/후에 관한 정보를 알 수 있도록 Null값을 허용하지 않았다.

13) Current_Shipping_Tracking

shipment_id(PK, FK)

- shipment table의 primary key인 shipment_id를 참조하고 있고, 하나의 shipment 당 운송 중이라면 하나의 현재 운송 중인 shipment에 대한 추적이 발생할 수 있으므로 zero or one의 one to one의 관계로 설정하였다. 그리고 현재 운송중인 shipment를 추적하고 있다면, shipment_id가 이 table에 반드시 존재해야 하고, 이 것으로 식별이 가능하므로 primary key로 설정하여 Null값을 허용하지 않았다.

current_location

- data type은 VARCHAR2(20)로 설정하였고, 현재 운송 중인 shipment에 대하여 위치를 알 수 있도록 Null값을 허용하지 않았다.

current_location_time

- data type은 DATE으로 설정하였고, 현재 운송 중인 시간 정보를 알 수 있도록 Null값을 허용하지 않았다.

currently_headed_location

- data type은 VARCHAR2(20)로 설정하였고, 현재 향하고 있는 위치를 알 수 있도록 Null값을 허용하지 않았다.

delivery_status

- data type은 VARCHAR2(10)로 설정하였고, 현재 운송 중인 정보에 대해서만 이 table에 저장 할 것이므로 Null값을 허용하지 않았고, 운송 중인 상태가 운송 완료로 바뀐다면 그러한 tuple은 이 table에서 삭제될 것이다.

transportation_id(FK)

- transportation table의 primary key인 transportation_id를 참조하고 있고, 하나의 transporation 당 여러 shipment의 운송이 발생할 수 있으므로, one or more의 one to many 관계를 설정하였다. 현재 운송 중에 있다면 반드시 어떠한 운송 수단을 이용해 운송 중인 것이므로, Null값을 허용하지 않았다.

14) History_of_Shipment_Tracking

tracking_history_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

Location

- data type은 VARCHAR2(20)로 설정하였고, 과거 운송할 당시의 위치를 저장하고 있으므로 Null값을 허용하지 않았다.

Time

- data type은 DATE로 설정하였고, 과거 운송할 당시의 시간을 저장하고 있으므로 Null값을 허용하지 않았다.

shipment_id(FK)

- shipment table의 primary key인 shipment_id를 참조하고 있고, 한 shipment의 과거 운송에 대한 tracking이므로 Null값을 허용하지 않았다. 그리고 하나의 shipment에 여러 개의 운송 이력이 남아있을 수 있으므로, zero, one or more의 one to many 관계를 설정하였다. 아직 shipment가 운송되지 않았을 경우나 막 배송을 시작한 경우에는 과거 이력이 없을 것을 고려하였다.

transportation_id(FK)

- transportation table의 primary key인 transportation_id를 참조하고 있고, 하나의 shipment에 대한 하나의 과거 추적은 하나의 운송 수단과 연결되기 때문에 one to one의 관계를 설정하였다. 그리고 과거 shipment tracking에는 반드시 운송 수단에 대한 정보가 존재해야 하므로 Null값은 허용하지 않았고, Cardinality value를 1으로 설정하였다.

15) Transportation

transportation_id(PK)

- data type은 NUMBER로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

transportation_number

- data type은 NUMBER로 설정하였고, 차 번호, 비행기 고유 번호의 정보를 저장하는 속성이기에 Null값을 허용하지 않았다.

transportation_method_type

- data type은 VARCHAR2(20)로 설정하였고, 운송 수단에 대한 설명으로 Null값을 허용하지 않았다.

16) Driver

driver_id(PK)

- data type은 NUMBER로 설정하였고, driver에 대한 id를 저장할 수 있으므로 Null값을 허용하지 않았다.

driver_name

- data type은 VARCHAR2(10)로 설정하였고, driver에 대한 이름을 저장할 수 있으므로 Null값을 허용하지 않았다.

driver_phone_number

- data type은 VARCHAR2(11)로 저장하였고, 회사와 driver의 연락 수단이 전화 번호라고 생각하여 Null값을 허용하지 않았다.

transportation_id(FK)

- transportation table의 primary key인 transportation_id를 참조하고 있고, data type은 NUMBER이고 Null값을 가질 수 없도록 하였다.