

Diketahui skema basisdata relasional sebuah toko online sbb.

Customer = **(CustomerID, Name, Address, CreditCardNo, Email, Rating, Discount)**

Product = **(ProductID, Name, Brand, UnitPrice)**

ShoppingCart = **(CartID, CustomerID, Date, TotalPrice)**

DetailShoppingCart = **(CartID, ProductID, NbOfProduct)**

Yang **dicetak tebal** adalah primary key dari tabel yang bersangkutan.

Untuk aturan bisnis di bawah ini:

1. Sebutkan klasifikasi skema constraint: type, attribute, relation, atau database constraint.

2. Sebutkan teknik yang paling sesuai digunakan untuk menjaga integritas data dan berikan penjelasan. Penjelasan minimum yang harus diberikan ada di keterangan di dalam tanda kurung di sebelah pilihan teknik. Satu persoalan dapat membutuhkan lebih dari satu teknik.

Berikut beberapa pilihan teknik yang dapat digunakan:

- Mendefinisikan **tipe atribut** (sebutkan tipe yang digunakan pada atribut apa di tabel apa).
- **Not null constraint** (sebutkan atribut mana yang harus diberikan *not null constraint*).
- **Unique constraint** (sebutkan atribut dari tabel mana yang harus diberikan *unique constraint*).
- **Check constraint** (sebutkan ekspresi *constraint* yang harus dipenuhi).
- **Foreign key references** (sebutkan tabel dan atribut mana yang me-refer dan ke atribut dan tabel mana *reference-nya*).
- **Trigger** (tuliskan trigger yang harus dibuat, gunakan cara penulisan *trigger* seperti yang ada di slide kuliah).

Anda boleh menambahkan pembuatan *function/stored procedure* untuk mendukung teknik yang Anda pilih. Jika Anda pakai, tuliskan spesifikasi dan kode program *function/procedure* tersebut (menggunakan cara penulisan seperti pada slide kuliah).

Aturan bisnis:

Atribut UnitPrice dari tabel Product bertipe POSINTEGER. Type POSINTEGER terdiri atas nilai integer > 0.

Paragraph

v

≡

~

A

A

v

A

v

A

v

B

I

o

:=

!=

“ ”

≡

≡

≡

f(x)

grid

1. Klasifikasi skema Product: type constraint karena tipe atribut harus POSINTEGER

2. Teknik yang digunakan:

- Mendefinisikan tipe atribut (tipe POSINTEGER) pada atribut UnitPrice

LTYF-JGRA | 🔍 🔍 🔍 🔍 🔍 🔍 🔍

Atribut UnitPrice dari tabel Product bertipe POSINTEGER. Type POSINTEGER terdiri atas nilai integer > 0.

## Soal 1

Customer = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)

Product = (ProductID, Name, Brand,  
UnitPrice)

ShoppingCart = (CartID, CustomerID,  
Date, TotalPrice)

DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

Type POSINTEGER terdiri atas nilai integer > 0.

- Klasifikasi constraint: Type Constraint
- Teknik menjaga integritas data: mendefinisikan type POSINTEGER = integer > 0

Atribut UnitPrice dari tabel Product bertipe POSINTEGER.

- Klasifikasi constraint: Attribute Constraint
- Teknik menjaga integritas data: type POSINTEGER digunakan pada atribut UnitPrice di tabel Product



KNIGHTS & SOFTWARE ENGINEERING

## Aturan bisnis:

Atribut Name di tabel Product harus unik dan tidak boleh null.

Paragraph                          

1. Klasifikasi skema Product: *attribute constraint* karena pada atribut diminta syarat unik dan tidak boleh null

2. Teknik yang digunakan:

- Not null constraint (name)
  - Unique constraint (name)
  - Check constraint → CHECK (name NOT IN (SELECT name FROM Product))

## Soal 2

**Customer = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)**

Product = (ProductID, Name, Brand, UnitPrice)

ShoppingCart = (CartID, CustomerID,  
Date, TotalPrice)

DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

Atribut Name di tabel Product harus unik dan tidak boleh null.

Klasifikasi constraint: **relation constraint**

Teknik menjaga integritas data:

- Menerapkan *unique constraint* dan *not null constraint* pada atribut Name di tabel Product





Aturan bisnis:

CustomerID pada tabel ShoppingCart merupakan salah satu dari CustomerID di tabel Customer. Jika ada data Customer akan dihapus, sedangkan CustomerID-nya merupakan salah satu CustomerID di tabel ShoppingCart, maka data Customer tersebut tidak bisa dihapus.

Paragraph

v

|

≡

v

A

v

A

v

A

v

B

I

C

:=

:=

“

”

|

≡

≡

|

¶

v

f(x)

¶

v

¶

v

¶

v

¶

v

¶

v

¶

v

¶

v

¶

v

¶

1. Klasifikasi skema: *database constraint* karena syarat penghapusan atribut melibatkan 2 skema yaitu ShoppingCart dan Customer

2. Teknik yang digunakan:

- *Foreign key references* karena CustomerID pada ShoppingCart reference ke CustomerID pada Customer

## Soal 4

Customer = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)

Product = (ProductID, Name, Brand,  
UnitPrice)

ShoppingCart = (CartID, CustomerID,  
Date, TotalPrice)

DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

CustomerID pada tabel ShoppingCart merupakan salah satu dari CustomerID di tabel Customer. Jika ada data Customer akan dihapus, sedangkan CustomerID-nya merupakan salah satu CustomerID di tabel ShoppingCart, maka data Customer tersebut tidak bisa dihapus.

Klasifikasi constraint: *database constraint*

Teknik menjaga integritas data:

- Menerapkan *foreign key reference* dari atribut CustomerID di tabel ShoppingCart ke atribut CustomerID pada tabel Customer dan...
- Menerapkan *referential action restrict* untuk operasi delete pada foreign key tersebut (on delete restrict)



INFORMATION & SOFTWARE ENGINEERING

#### Aturan bisnis:

Nilai atribut TotalPrice pada tabel ShoppingCart harus sesuai dengan total penjumlahan harga semua Product untuk IDCart yang bersesuaian di tabel DetailShoppingCart, yaitu penjumlahan dari DetailShopping.NbOfProduct \* Product.UnitPrice untuk Product yang sesuai dengan DetailShoppingCart.ProductID.

Paragraph

1. Klasifikasi skema: database constraint karena aturan tersebut melibatkan lebih dari 1 skema

2. Teknik yang digunakan: *Trigger*

```
CREATE TRIGGER UpdatePrice AFTER INSERT OF DetailShoppingCart
```

begin atomic

## UPDATE ShoppingCart

SET TotalPrice = TotalPrice +

```
SELECT SUM(DetailShoppingCart.NbOfProduct * Product.UnitPrice)
```

FROM DetailShoppingCart NATURAL JOIN ShoppingCart NATURAL JOIN Product

## Soal 4

**Customer** = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)

Product = (ProductID, Name, Brand,  
UnitPrice)

**ShoppingCart** = (CartID, CustomerID,  
Date, TotalPrice)

DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

Nilai atribut TotalPrice pada tabel ShoppingCart harus sesuai dengan total penjumlahan harga semua Product untuk IDCart yang bersesuaian di tabel DetailShoppingCart, yaitu penjumlahan dari DetailShopping.NbOfProduct \* Product.UnitPrice untuk Product yang sesuai dengan DetailShoppingCart.ProductID.

#### Klasifikasi constraint: database constraint

Teknik menjaga integritas data:

- Menggunakan trigger after insert+delete dan after update NbOfProduct pada tabel DetilShoppingCart serta after update UnitPrice pada tabel Product



KNOWLEDGE & SOFTWARE ENGINEERING

**Alternatif-1: Menggunakan Trigger****Ide:**

- sesudah penambahan/penghapusan data di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut NbOfProduct di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut UnitPrice di tabel Product, secara otomatis dilakukan perubahan nilai TotalPrice di tabel ShoppingCart

-- Trigger after insert DetailShoppingCart

-- Asumsi: NbOfProduct pasti tidak null

create trigger DetailShoppingCart\_after\_insert\_trigger after insert of DetailShoppingCart

referencing new row as nrow

for each row

begin

declare harga integer;

set harga = (SELECT UnitPrice FROM Product

WHERE ProductID = nrow.ProductID);

UPDATE ShoppingCart

SET TotalPrice = TotalPrice + harga \* nrow.NbOfProduct

WHERE CartID = nrow.CartID;

↳

end;

**Soal 4**Customer = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)Product = (ProductID, Name, Brand,  
UnitPrice)ShoppingCart = (CartID, CustomerID,  
Date, TotalPrice)DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

KNOWLEDGE &amp; SOFTWARE ENGINEERING

**Alternatif-1: Menggunakan Trigger****Ide:**

- sesudah penambahan/penghapusan data di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut NbOfProduct di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut UnitPrice di tabel Product, secara otomatis dilakukan perubahan nilai TotalPrice di tabel ShoppingCart

-- Trigger after update NbOfProduct di DetailShoppingCart

-- Asumsi: NbOfProduct pasti tidak null

create trigger DetailShoppingCart\_after\_update\_trigger after update of DetailShoppingCart on (NbOfProduct)

referencing new row as nrow

referencing old row as orow

for each row

begin

declare harga integer;

set harga = (SELECT UnitPrice FROM Product

WHERE ProductID = nrow.ProductID);

UPDATE ShoppingCart

SET TotalPrice = TotalPrice - harga \* orow.NbOfProduct +  
harga \* nrow.NbOfProduct

WHERE CartID = nrow.CartID;

end;

**Soal 4**Customer = (CustomerID, Name,  
Address, CreditCardNo, Email, Rating,  
Discount)Product = (ProductID, Name, Brand,  
UnitPrice)ShoppingCart = (CartID, CustomerID,  
Date, TotalPrice)DetailShoppingCart = (CartID,  
ProductID, NbOfProduct)

KNOWLEDGE &amp; SOFTWARE ENGINEERING

## Soal 4

Customer = (CustomerID, Name, Address, CreditCardNo, Email, Rating, Discount)

Product = (ProductID, Name, Brand, UnitPrice)

ShoppingCart = (CartID, CustomerID, Date, TotalPrice)

DetailShoppingCart = (CartID, ProductID, NbOfProduct)

### Alternatif-1: Menggunakan Trigger

Ide:

- sesudah penambahan/penghapusan data di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut NbOfProduct di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut UnitPrice di tabel Product, secara otomatis dilakukan perubahan nilai TotalPrice di tabel ShoppingCart

-- Trigger after delete DetailShoppingCart

```
create trigger DetailShoppingCart_after_delete_trigger after delete of
DetailShoppingCart
```

referencing old row as orow

for each row

begin

```
declare oldprice integer; -- nilai yang dihapus
set oldprice = (SELECT UnitPrice * orow.NbOfProduct
```

FROM Product

WHERE ProductID = orow.ProductID);

UPDATE ShoppingCart

SET TotalPrice = TotalPrice - oldprice

WHERE CartID = orow.CartID;

end;



KNOWLEDGE & SOFTWARE ENGINEERING

## Soal 4

Customer = (CustomerID, Name, Address, CreditCardNo, Email, Rating, Discount)

Product = (ProductID, Name, Brand, UnitPrice)

ShoppingCart = (CartID, CustomerID, Date, TotalPrice)

DetailShoppingCart = (CartID, ProductID, NbOfProduct)

### Alternatif-1: Menggunakan Trigger

Ide:

- sesudah penambahan/penghapusan data di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut NbOfProduct di tabel DetailShoppingCart, atau sesudah perubahan nilai atribut UnitPrice di tabel Product, secara otomatis dilakukan perubahan nilai TotalPrice di tabel ShoppingCart

-- Trigger after update UnitPrice di Product

-- Asumsi: UnitPrice pasti not null

```
create trigger Product_after_update_trigger after update of Product
on (UnitPrice)
```

referencing new row as nrow

referencing old row as orow

for each row

begin

... -- next slide

end;



```
-- Trigger after update UnitPrice di Product
-- Asumsi: UnitPrice pasti not null
create trigger Product_after_update_trigger after update of Product on (UnitPrice)
referencing new row as nrow
referencing old row as orow
for each row
begin
    for r as SELECT * FROM DetailShoppingCart WHERE ProductID = nrow.ProductID do
        declare oldprice integer;
        declare newprice integer;
        set oldprice = r.NbOfProduct * orow.UnitPrice;
        set newprice = r.NbOfProduct * nrow.UnitPrice;
        UPDATE ShoppingCart
        SET TotalPrice = TotalPrice - oldprice + newprice
        WHERE CartID = r.CartID;
    end for;
end;
```