A5: Relational Schema, validation and schema refinement

1. Relational Schema

> The Relational Schema includes the relation schemas, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK. Relation schemas are specified in the compact notation:

Relation reference	**Relation Compact Notation**
R01	User(username, password NN, email UK NN, joinDate NN, picture)
R02	Customer(username → User, name NN, address, loyaltyPoints NN CK > 0, newsletter NN, inactive NN)
R03	Moderator(username → User)
R04	Administrator(username → User)
R05	Banned(username → Customer, bannedDate NN CK bannedDate > User.joinDate, moderator → Moderator)
R06	Commentary(id, username → Customer NN, date, text NN, flagsNo NN)
R07	Answer(idParent → Commentary, idChild → Commentary)
R08	Flagged(idComment → Commentary, hidden NN)
	Product(sku , title NN, cat → Category NN, price NN CK

R09	price > 0, discountPrice, rating NN CK rating > 0, stock NN)
R10	Attribute(id, name NN)
R11	AttributeProduct(idAttribute → Attribute, refProduct → Product, value NN)
R12	CatAtt(att → Attribute, cat → Category)
R13	Category(id, name)
R14	Favorite(username → Customer, refProduct → Product)
R15	Purchase(id, username → Customer NN, date NN, value NN, method NN)
R16	PurchaseProduct(id → Purchase, refProduct → Product, price NN, quantity NN CK quantity>0)
R17	Rating(username → Customer, refProduct → Product, value NN CK value > 0 AND CK value <= 5)

Where UK means UNIQUE KEY, NN means NOT NULL and CK means CHECK.

2. Domains

The specification of additional domains can also be made in a compact form, using the notation:

Domain Name	Domain Specification
Today	DATE DEFAULT CURRENT_DATE
Rating	INTEGER > 0 AND <= 5
PaymentMethod	ENUM ('Paypal', 'Debit', 'Credit')

3. Functional Dependencies and Schema Validation

> To validate the Relational Schema obtained from the Conceptual Model, all functional dependencies are identified and the normalization of all relation schemas is accomplished. Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using normalization.

Table R01 (User)

Keys: { username } , { email }	
Functional Dependencies	
FD0101	{username} → {password, email, joinDate, picture}
FD0102	{email} → {password, username, joinDate, picture}
Normal Form	BCNF

Table R02 (Customer)

Keys: { username }	
Functional Dependencies	
FD0201	{username} → {name, address, loyaltyPoints, newsletter, inactive}
Normal	

Form	BCNF

Table R03 (Moderator)	
Keys: { username }	
Functional Dependencies	
FD0301	none
Normal Form	BCNF

Table R04 (Administrator)	
Keys: { username }	
Functional Dependencies	
FD0401	none
Normal Form	BCNF

Table R05 (Banned)	
Keys: { username }	
Functional Dependencies	
FD0501	{username} → {bannedDate}
Normal Form	BCNF

Table R06 (Commentary)

Keys: { username, date }

Functional Dependencies

FD0601	{(username, date)} → {text, flagsNo}
Normal Form	BCNF

Table R07 (Answer)

Keys: {idParent, idChild}

Functional Dependencies

FD0701	none
Normal Form	BCNF

Table R08 (Flagged)

Keys: {idComment}

Functional Dependencies

FD0801	{idComment} → {hidden}
Normal Form	BCNF

Table R09 (Attribute)

Keys: {name}

Functional Dependencies

FD0901	none
Normal Form	BCNF

Table R10 (AttributeProduct)

Keys: {name, refProduct}

Functional Dependencies

FD1001	{(name, refProduct)} → {value}
Normal Form	BCNF

Table R11 (CatAtt)

Keys: {att, cat}

Functional Dependencies

FD1101	none
Normal Form	BCNF

Table R12 (Category)

Keys: {name}

Functional Dependencies

FD1201	none
Normal Form	BCNF

Table R13 (Favorite)

Keys: {username, refProduct}

Functional Dependencies

FD1301	none
Normal Form	BCNF

Table R14(Purchase)

Keys: {id}

Functional Dependencies

FD1401	{id} → {username, date, value}
Normal Form	BCNF

Table R15(Purchase)

Keys: {id}

Functional Dependencies

FD1401	{id} → {username, date, value}
Normal Form	BCNF

Table R16(PurchaseProduct)

Keys: {id, refProduct}

Functional Dependencies

FD1601	{(username, refProduct)} → {price, quantity}
Normal Form	BCNF

Table R17(Rating)

Keys: {username, refProduct}

Functional Dependencies

FD1701	{(username, refProduct)} → {value}
Normal Form	BCNF

As all relational schemas are in the Boyce–Codd Normal Form (BCNF), therefore there is no need to refine using normalisation. They are in the BCNF because in every case the left side of the functional dependency is a superkey.

4. SQL Code

> SQL code necessary to build (and rebuild) the database. This code should also be included in the group's github repository as an SQL script, and a link include here.

```
CREATE TABLE "user" (
    username text PRIMARY KEY,
    "password" text NOT NULL,
    email text UNIQUE NOT NULL,
    joinDate TIMESTAMP DEFAULT now() NOT NULL,
    picture text
);
```

```
CREATE TABLE customer (
    username_user text NOT NULL
        REFERENCES "user" ON DELETE CASCADE,
    "name" text NOT NULL,
    "address" text,
    loyaltyPoints INTEGER NOT NULL DEFAULT 0,
    newsletter INTEGER NOT NULL,
    inactive INTEGER NOT NULL,

    CONSTRAINT lp_positive CHECK ((loyaltyPoints >=
0)),
);

CREATE TABLE moderator (
    username_user text NOT NULL REFERENCES "user" ON
DELETE CASCADE
);

CREATE TABLE administrator (
    username_user text NOT NULL REFERENCES "user" ON
DELETE CASCADE
);

CREATE TABLE banned (
    username_customer text PRIMARY KEY REFERENCES
customer ON DELETE CASCADE,
    bannedDate DATE NOT NULL, --this has to be with
triggers
    username_moderator text NOT NULL REFERENCES
moderator ON DELETE CASCADE,

    -- CONSTRAINT banned_date CHECK (banned_date >
username_customer.joinDate)
);

CREATE TABLE comment (
    id SERIAL PRIMARY KEY,
    username_customer text NOT NULL REFERENCES
moderator ON DELETE CASCADE,
```

```
    "date" DATE NOT NULL,
    commentary text NOT NULL,
    flagsNo INTEGER NOT NULL
);

CREATE TABLE answer (
    idParent INTEGER NOT NULL REFERENCES comment ON
DELETE CASCADE,
    idChild INTEGER NOT NULL REFERENCES comment ON
DELETE CASCADE,

    UNIQUE(idParent, idChild)
);

CREATE TABLE flagged (
    idComment INTEGER NOT NULL REFERENCES comment ON
DELETE CASCADE,
    "hidden" BOOLEAN NOT NULL,
);

CREATE TABLE category (
    id SERIAL PRIMARY KEY,
    "name" text NOT NULL,
);

CREATE TABLE product (
    sku INTEGER PRIMARY KEY,
    title text NOT NULL,
    idCat INTEGER NOT NULL REFERENCES category ON
DELETE CASCADE,
    price REAL NOT NULL,
    discountPrice REAL,
    rating REAL NOT NULL,
    stock INTEGER NOT NULL,

    CONSTRAINT price_positive CHECK (price > 0),
    CONSTRAINT discount_positive CHECK (discountPrice
is NULL or discountPrice > 0),
    CONSTRAINT stock_positive CHECK(stock >= 0),
```

```
    CONSTRAINT rating_positive CHECK(rating >= 0)
);

CREATE TABLE attribute (
    id SERIAL PRIMARY KEY,
    "name" text NOT NULL,
);

CREATE TABLE attribute_product (
    idAttribute INTEGER NOT NULL REFERENCES attribute
ON DELETE CASCADE,
    refProduct INTEGER NOT NULL REFERENCES product ON
DELETE CASCADE,
    "value" text NOT NULL,
);

CREATE TABLE category_attribute (
    idAttribute INTEGER NOT NULL REFERENCES attribute
ON DELETE CASCADE,
    idCategory INTEGER NOT NULL REFERENCES category
ON DELETE CASCADE,

    UNIQUE(idAttribute, idCategory)
);

CREATE TABLE favorite (
    username_customer text NOT NULL REFERENCES
customer ON DELETE CASCADE,
    refProduct INTEGER NOT NULL REFERENCES product ON
DELETE CASCADE,

    UNIQUE(username_customer, refProduct)
);

CREATE TABLE purchase (
    id SERIAL PRIMARY KEY,
    username text NOT NULL REFERENCES customer ON
DELETE CASCADE,
    "date" DATE NOT NULL,
```

```
    "value" REAL NOT NULL,
    method text NOT NULL,

    CONSTRAINT value_positive CHECK ("value" > 0),
    CONSTRAINT method_check CHECK (method in
('Credit', 'Debit' , 'Paypal' ))
);

CREATE TABLE purchase_product (
    idPurchase INTEGER NOT NULL REFERENCES purchase
ON DELETE CASCADE,
    idProduct INTEGER NOT NULL REFERENCES product ON
DELETE CASCADE,
    price REAL NOT NULL,
    quantity INTEGER NOT NULL,

    CONSTRAINT quantity_positive CHECK (quantity >
0),
    CONSTRAINT price_positive CHECK (price > 0),

    UNIQUE(idPurchase, idProduct)
);

CREATE TABLE rating (
    username_customer text NOT NULL,
    refProduct INTEGER NOT NULL,
    "value" INTEGER NOT NULL CHECK (("value" > 0 )
AND ("value" <= 5)),
    PRIMARY KEY(username_customer, refProduct)
);
```

Link to the file here.

Revision history

Ana Cláudia Fonseca Santos, up200700742@fe.up.pt

Eduardo de Mendonça Rodrigues Salgado Ramos, up201505779@fe.up.pt

Mariana Lopes da Silva, up201506197@fe.up.pt

Xavier Reis Fontes, up201503145@fe.up.pt