

# Information Systems - Lab assignment 2

Siemen Boelkens - s2788845

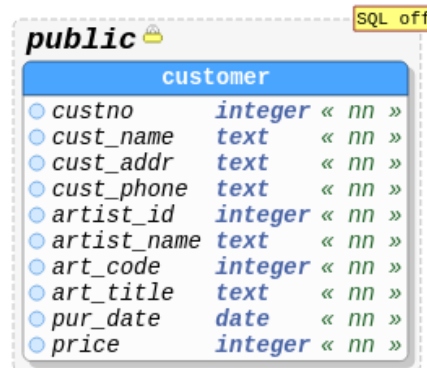
Hichem Bouakaz - s2525763

December 7, 2018

## 1 Database Normalization

*Normalize the schema up to the third normal form and create it in postgresql. For each table, indicate the primary and foreign keys. Provide the scripts to generate the schema.*

### 1.1 Unnormalized Database



public		
customer		
<input type="radio"/> custno	integer	« nn »
<input type="radio"/> cust_name	text	« nn »
<input type="radio"/> cust_addr	text	« nn »
<input type="radio"/> cust_phone	text	« nn »
<input type="radio"/> artist_id	integer	« nn »
<input type="radio"/> artist_name	text	« nn »
<input type="radio"/> art_code	integer	« nn »
<input type="radio"/> art_title	text	« nn »
<input type="radio"/> pur_date	date	« nn »
<input type="radio"/> price	integer	« nn »

Figure 1: Unnormalized Database Schema

### 1.2 First Normal Database

The first normal the database should only contain atomic values at each row and column, and should have no repeating groups.

In the given example we have ,  $(artist\_id, artist\_name, art\_code, art\_title, pur\_date, price)$  a repeating group to change that we have introduced a primary key from the  $(\"custno\", \"art\_code\", \"pur\_date\")$  column combination ,see [Figure 2](#) .

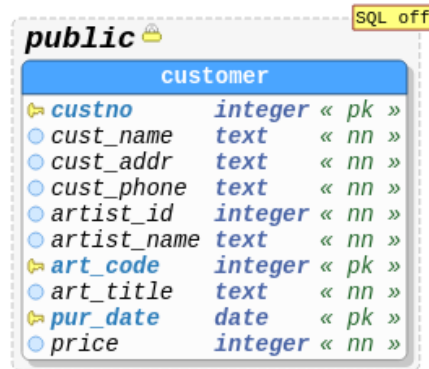


Figure 2: First normal Database Schema.

Code [subsection 4.1](#)

### 1.3 Second Normal Database

The second normal form requires that every nonkey attribute is fully functionally dependent on the primary key, therefore we split the table in [Figure 2](#) into three tables where every non key is dependent on the primary key:

- **customer:** non key elements cust\_name, cust\_addr, cust\_phone are dependent on the custno primary key.
- **purchase:** non key element price is dependent on the primary key combination (cust\_no, art\_code, pur\_date)
- **art:** non key elements art\_title, artist\_id, artist\_name are dependent on the primary key art\_code.

See [Figure 3](#), for the code check [subsection 4.3](#)

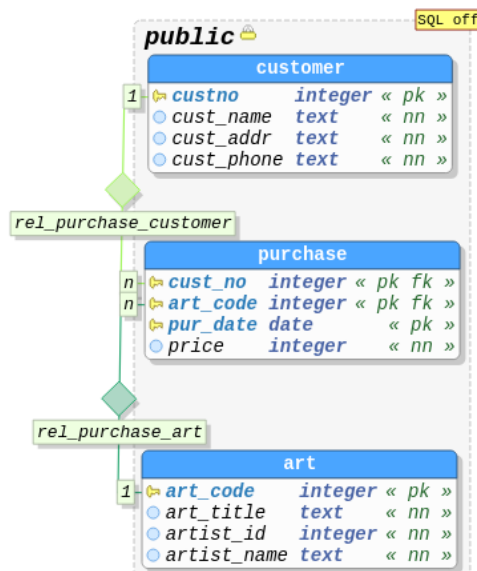


Figure 3: Second Normal Database Schema

## 1.4 Third Normal Database

Third Normal Form has no transitive functional dependency between nonkey attributes, to normalize the database to the third normal form we splitted the art table into art and artist since the artist\_name can be determined by the artist.id, See [Figure 4](#)

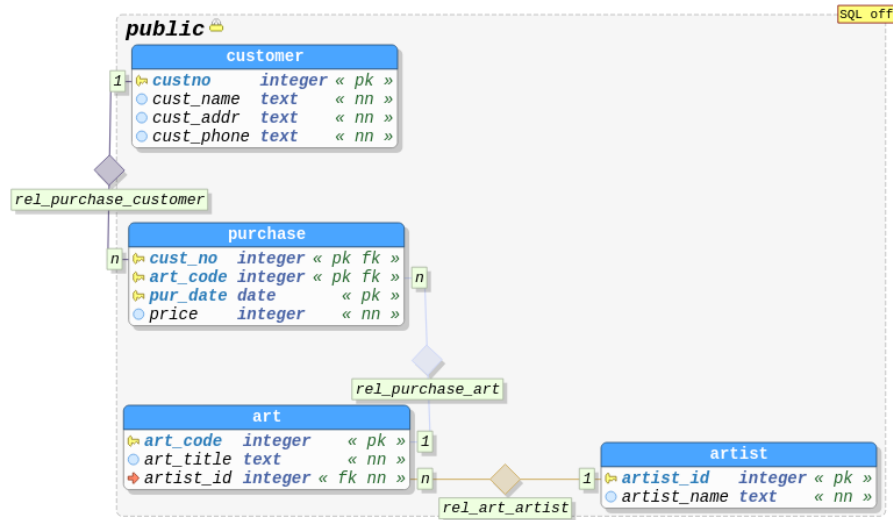


Figure 4: Third Normal Database Schema.

Related code: [subsection 4.4](#)

## 2 Uppercase triggers

Create triggers (and procedures/functions if necessary) in the appropriate tables to set to uppercase the customer and the artists' names with every insert or update.

```

1 CREATE FUNCTION public.uppercase_cust_name_on_insert ()
2     RETURNS trigger
3     LANGUAGE plpgsql
4     VOLATILE
5     CALLED ON NULL INPUT
6     SECURITY INVOKER
7     COST 100
8     AS $$
9
10    BEGIN
11        NEW.cust_name = UPPER(NEW.cust_name);
12        RETURN NEW;
13    END;
14
15    $$;
16
17 CREATE TRIGGER uppercase_cust_name_on_insert
18     BEFORE INSERT OR UPDATE
19     ON public.customer

```

```

20         FOR EACH ROW
21         EXECUTE PROCEDURE public.uppercase_cust_name_on_insert();

1  CREATE FUNCTION public.uppercase_artist_name_on_insert ()
2      RETURNS trigger
3      LANGUAGE plpgsql
4      VOLATILE
5      CALLED ON NULL INPUT
6      SECURITY INVOKER
7      COST 100
8      AS $$
9
10     BEGIN
11         NEW.artist_name = UPPER(NEW.artist_name);
12         RETURN NEW;
13     END;
14
15 $$;
16
17 CREATE TRIGGER uppercase_artist_name_on_insert_trigger
18     BEFORE INSERT OR UPDATE
19     ON public.artist
20     FOR EACH ROW
21     EXECUTE PROCEDURE public.uppercase_artist_name_on_insert();

```

### 3 Price trigger

*Create a trigger that checks that the selling price is greater than 0.*

```

1  CREATE FUNCTION public.validate_price_on_insert ()
2      RETURNS trigger
3      LANGUAGE plpgsql
4      VOLATILE
5      CALLED ON NULL INPUT
6      SECURITY INVOKER
7      COST 100
8      AS $$
9
10     BEGIN
11         IF NEW.price = 0 THEN
12             RAISE EXCEPTION 'The price should not be %', NEW.price;
13         END IF;
14         RETURN NEW;
15     END;
16
17 $$;
18
19 CREATE TRIGGER validate_price_on_insert_trigger
20     BEFORE INSERT OR UPDATE
21     ON public.purchase
22     FOR EACH ROW

```

```
EXECUTE PROCEDURE public.validate_price_on_insert();
```

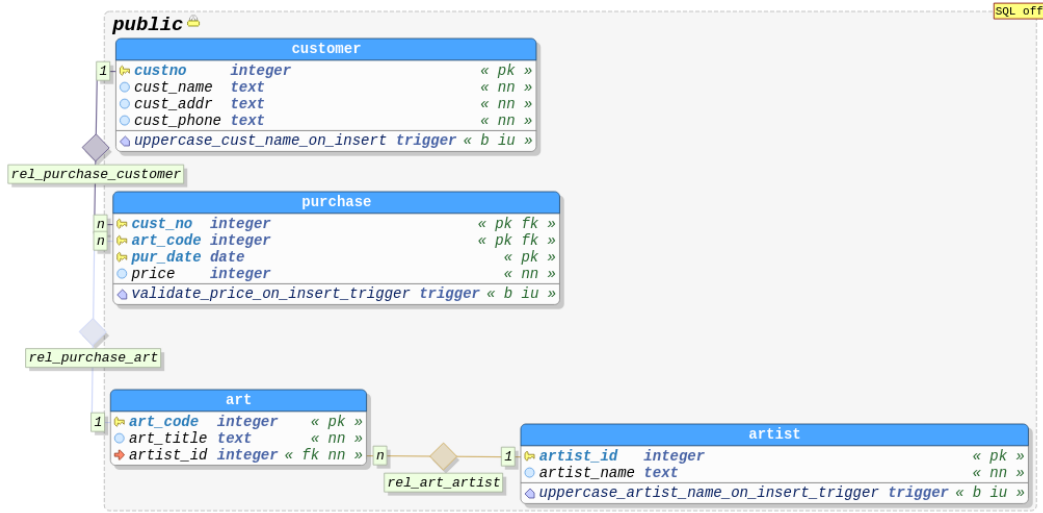


Figure 5: Third Normal Database with triggers.

## 4 Appendix: Code

### 4.1 Unnormalized Database

#### Unnormalized Database

```
1
2 DROP TABLE IF EXISTS "customer";
3 CREATE TABLE "public"."customer" (
4     "custno" integer NOT NULL,
5     "cust_name" text NOT NULL,
6     "cust_addr" text NOT NULL,
7     "cust_phone" text NOT NULL,
8     "artist_id" integer NOT NULL,
9     "artist_name" text NOT NULL,
10    "art_code" integer NOT NULL,
11    "art_title" text NOT NULL,
12    "pur_date" date NOT NULL,
13    "price" integer NOT NULL
14 ) WITH (oids = false);
```

## 4.2 First Normal Form Database

First Normal Form Database Database

```
1
2 DROP TABLE IF EXISTS "customer";
3 CREATE TABLE "public"."customer" (
4     "custno" integer NOT NULL,
5     "cust_name" text NOT NULL,
6     "cust_addr" text NOT NULL,
7     "cust_phone" text NOT NULL,
8     "artist_id" integer NOT NULL,
9     "artist_name" text NOT NULL,
10    "art_code" integer NOT NULL,
11    "art_title" text NOT NULL,
12    "pur_date" date NOT NULL,
13    "price" integer NOT NULL,
14    CONSTRAINT "customer_pkey" PRIMARY KEY ("custno", "art_code", "pur_date")
15 ) WITH (oids = false);
```

## 4.3 Second Normal Form Database

Second Normal Form Database Database

```
1
2 CREATE TABLE public.art(
3     art_code integer NOT NULL DEFAULT nextval('public.art_art_code_seq'::regclass),
4     art_title text NOT NULL,
5     artist_id integer NOT NULL,
6     artist_name text NOT NULL,
7     CONSTRAINT art_pk PRIMARY KEY (art_code)
8
9 );
10
11 CREATE TABLE public.customer(
12     custno integer NOT NULL,
13     cust_name text NOT NULL,
14     cust_addr text NOT NULL,
15     cust_phone text NOT NULL,
16     CONSTRAINT customer_custno PRIMARY KEY (custno)
17
18 );
19
20 CREATE TABLE public.purchase(
21     cust_no integer NOT NULL,
22     art_code integer NOT NULL,
23     pur_date date NOT NULL,
24     price integer NOT NULL,
25     CONSTRAINT purchase_pk PRIMARY KEY (cust_no,art_code,pur_date)
26
27 );
28
29 ALTER TABLE public.purchase ADD CONSTRAINT purchase_cust_no_fk FOREIGN KEY (cust_no)
30 REFERENCES public.customer (custno) MATCH FULL
31 ON DELETE NO ACTION ON UPDATE NO ACTION;
32
33 ALTER TABLE public.purchase ADD CONSTRAINT purchase_art_code_fk FOREIGN KEY (art_code)
34 REFERENCES public.art (art_code) MATCH FULL
35 ON DELETE NO ACTION ON UPDATE NO ACTION;
```



## 4.4 Third Normal Form Database

First Normal Form Database Database

```
1
2
3
4 DROP TABLE IF EXISTS "artist";
5 DROP SEQUENCE IF EXISTS artist_artist_id_seq;
6 CREATE SEQUENCE artist_artist_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807 START 1 CACHE 1;
7
8 CREATE TABLE "public"."artist" (
9     "artist_id" integer DEFAULT nextval('artist_artist_id_seq') NOT NULL,
10    "artist_name" text NOT NULL,
11    CONSTRAINT "artist_artist_id" PRIMARY KEY ("artist_id")
12 ) WITH (oids = false);
13
14 DROP TABLE IF EXISTS "art";
15 DROP SEQUENCE IF EXISTS art_art_code_seq;
16 CREATE SEQUENCE art_art_code_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807 START 1 CACHE 1;
17
18 CREATE TABLE "public"."art" (
19     "art_code" integer DEFAULT nextval('art_art_code_seq') NOT NULL,
20     "art_title" text NOT NULL,
21     "artist_id" integer NOT NULL,
22     CONSTRAINT "art_artist_id_fkey" FOREIGN KEY (artist_id) REFERENCES artist(artist_id) NOT DEFERRABLE
23 ) WITH (oids = false);
24
25
26
27 DROP TABLE IF EXISTS "customer";
28 CREATE TABLE "public"."customer" (
29     "custno" integer NOT NULL,
30     "cust_name" text NOT NULL,
31     "cust_addr" text NOT NULL,
32     "cust_phone" text NOT NULL,
33     CONSTRAINT "customer_custno" PRIMARY KEY ("custno")
34 ) WITH (oids = false);
35
36
37 DROP TABLE IF EXISTS "purchase";
38 CREATE TABLE "public"."purchase" (
39     "cust_no" integer NOT NULL,
40     "art_code" integer NOT NULL,
41     "pur_date" date NOT NULL,
42     "price" integer NOT NULL,
43     CONSTRAINT "purchase_pkey" PRIMARY KEY ("cust_no", "art_code", "pur_date"),
44     CONSTRAINT "purchase_cust_no_fkey" FOREIGN KEY (cust_no) REFERENCES customer(custno) NOT DEFERRABLE
45 ) WITH (oids = false);
```

## 4.5 Third Normal Form Database with Triggers

First Normal Form Database Database

```
1
2 CREATE SEQUENCE public.artist_artist_id_seq
3     INCREMENT BY 1
4     MINVALUE 1
5     MAXVALUE 9223372036854775807
6     START WITH 1
7     CACHE 1
8     NO CYCLE
9     OWNED BY NONE;
10
11 CREATE TABLE public.artist(
12     artist_id integer NOT NULL DEFAULT nextval('public.artist_artist_id_seq'::regclass),
13     artist_name text NOT NULL,
14     CONSTRAINT artist_artist_id PRIMARY KEY (artist_id)
15 );
16
17 CREATE SEQUENCE public.art_art_code_seq
18     INCREMENT BY 1
19     MINVALUE 1
20     MAXVALUE 9223372036854775807
21     START WITH 1
22     CACHE 1
23     NO CYCLE
24     OWNED BY NONE;
25
26 CREATE TABLE public.art(
27     art_code integer NOT NULL DEFAULT nextval('public.art_art_code_seq'::regclass),
28     art_title text NOT NULL,
29     artist_id integer NOT NULL,
30     CONSTRAINT art_pk PRIMARY KEY (art_code)
31 );
32
33 CREATE TABLE public.customer(
34     custno integer NOT NULL,
35     cust_name text NOT NULL,
36     cust_addr text NOT NULL,
37     cust_phone text NOT NULL,
38     CONSTRAINT customer_custno PRIMARY KEY (custno)
39 );
40
41
42 CREATE TABLE public.purchase(
43     cust_no integer NOT NULL,
44     art_code integer NOT NULL,
45     pur_date date NOT NULL,
46     price integer NOT NULL,
47     CONSTRAINT purchase_pkey PRIMARY KEY (cust_no,art_code,pur_date)
48 );
49
50
```

```

51 CREATE FUNCTION public.uppercase_cust_name_on_insert ()
52     RETURNS trigger
53     LANGUAGE plpgsql
54     VOLATILE
55     CALLED ON NULL INPUT
56     SECURITY INVOKER
57     COST 100
58     AS $$
59
60 BEGIN
61     NEW.cust_name = UPPER(NEW.cust_name);
62     RETURN NEW;
63 END;
64
65 $$;
66
67 CREATE TRIGGER uppercase_cust_name_on_insert
68     BEFORE INSERT OR UPDATE
69     ON public.customer
70     FOR EACH ROW
71     EXECUTE PROCEDURE public.uppercase_cust_name_on_insert();
72
73
74 CREATE FUNCTION public.uppercase_artist_name_on_insert ()
75     RETURNS trigger
76     LANGUAGE plpgsql
77     VOLATILE
78     CALLED ON NULL INPUT
79     SECURITY INVOKER
80     COST 100
81     AS $$
82
83 BEGIN
84     NEW.artist_name = UPPER(NEW.artist_name);
85     RETURN NEW;
86 END;
87
88 $$;
89
90 CREATE TRIGGER uppercase_artist_name_on_insert_trigger
91     BEFORE INSERT OR UPDATE
92     ON public.artist
93     FOR EACH ROW
94     EXECUTE PROCEDURE public.uppercase_artist_name_on_insert();
95
96
97 CREATE FUNCTION public.validate_price_on_insert ()
98     RETURNS trigger
99     LANGUAGE plpgsql
100    VOLATILE
101    CALLED ON NULL INPUT
102    SECURITY INVOKER
103    COST 100
104    AS $$

```

```

105
106 BEGIN
107     IF NEW.price = 0 THEN
108         RAISE EXCEPTION 'The price should not be %', NEW.price;
109     END IF;
110     RETURN NEW;
111 END;
112
113 $$;
114
115 CREATE TRIGGER validate_price_on_insert_trigger
116     BEFORE INSERT OR UPDATE
117     ON public.purchase
118     FOR EACH ROW
119     EXECUTE PROCEDURE public.validate_price_on_insert();
120
121 ALTER TABLE public.art ADD CONSTRAINT art_artist_id_fkey FOREIGN KEY (artist_id)
122 REFERENCES public.artist (artist_id) MATCH SIMPLE
123 ON DELETE NO ACTION ON UPDATE NO ACTION;
124
125
126 ALTER TABLE public.purchase ADD CONSTRAINT purchase_cust_no_fkey FOREIGN KEY (cust_no)
127 REFERENCES public.customer (custno) MATCH SIMPLE
128 ON DELETE NO ACTION ON UPDATE NO ACTION;
129
130
131 ALTER TABLE public.purchase ADD CONSTRAINT purchase_art_code_fk FOREIGN KEY (art_code)
132 REFERENCES public.art (art_code) MATCH FULL
133 ON DELETE NO ACTION ON UPDATE NO ACTION;

```