Department of Information Technology
Uppsala University
December 13, 2012

Database Design

Assignments 1-3

Emanuel Berg
830622-1535
embe8573@student.uu.se
http://user.it.uu.se/~embe8573

1 Assignment 1: Normalization

1.1 Current state

In the below subsections, an arrow indicates a fully functional dependency.

1.1.1 Book table

`Title` $\rightarrow$ {`ISBN, Title, PublYear`}
`ISBN` $\rightarrow$ {`TitleNr, Title, PublYear`}
`Author` $\rightarrow$ `AuthorNat`

This table is not in $2NF$. For example, the attribute $Title$ is dependent on the primary key $\{TitleNr, CopyNr, Author\}$, but it is also dependent on $TitleNr$ and $ISBN$, so $Title$ is not *fully* functionally dependent on the primary key (or: "the determinant is not minimal").

The problem with the book table is that the same data appear more than once. If a book has more than one author, or if the library has more than one copy of a book, multiple rows containing the same data will be created.

In general, this is bad design, but a specific problem that arises is that, if data is to be changed, it must be changed at *every* occurrence. Apart from this being tedious, some places may be overlooked by mistake (that is, it is less secure).

Another flaw is that information about the author(s) should not appear in the book table. This design makes the database frustrating to navigate.

1.1.2 Customer

`CustomerNr` $\rightarrow$ {`PersonNr, Name, Address, Tel, NrBooks`}
`PersonNr` $\rightarrow$ {`CustomerNr, Name, Address, Tel, NrBooks`}
`Address` $\leftrightarrow$ `Tel`

The customer table is in $1NF$ only if addresses are considered atomic, otherwise a complex attribute, say, "Residence", could be used.

Moreover, the customer table is not in $3NF$ as the non-key attributes Address and Tel determine each other.

As for addresses not being $1NF$, this is a purely technical problem that has to do with the software that manages/implements the database. As for Address determining Tel and vice verse, I don't see why this should cause any problems.

1.1.3 Loan

{TitleNr, CopyNr} → {CustomerNr, Date, BorrowerName}
CustomerNr ↔ BorrowerName

If dates are considered atomic, this is in $1NF$. But, it is not in $3NF$ as the non-key attributes $CustomerNr$ and $BorrowerName$ determine each other. Apart from a confusing design (the name of the borrower is a property of the borrower, not the loan), the problem of redundancy arises as data appear repeatedly in the database.

1.2 New design

1.2.1 Tables

Keys are **primary** or *alternative*.

author(Name, Book)

authorNationality(Author, Nationality)
Author → Nationality

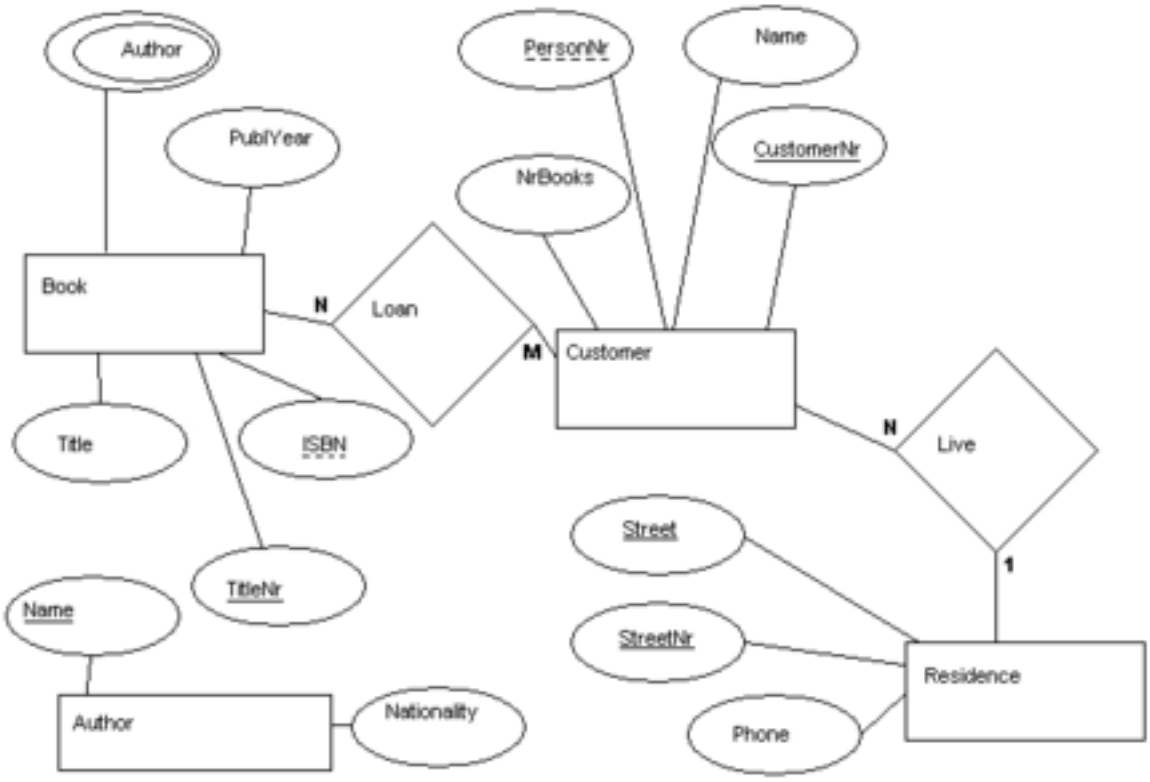
book(TitleNr, *ISBN*, Title, PublYear)
TitleNr → ISBN, Title, PublYear)
ISBN → TitleNr, Title, PublYear)

customer(CustomerNr, *PersonNr*, Name, NrBooks, Address)
customerNr → PersonNr, Name, NrBooks, Address
personNr → CustomerNr, Name, NrBooks, Address

loan(TitleNr, CopyNr, CustomerNr, Date)
TitleNr, CopyNr → Customer, Date

residence(ResidenceNo, *Street*, *StreetNr*, Phone)
ResidenceNo → Phone, Street, StreetNr
Street, StreetNr → Phone, ResidenceNo

1.2.2 ER diagram



2 Assignment 2

2.1 Setup

```
whenever error continue;
set output off;

create databank dataBank;

create unique sequence number
initial_value = 0
increment = 1;

drop table manager cascade;
drop table customer cascade;
drop table location cascade;
drop table account cascade;
drop table transact cascade;

set output on;
whenever error exit;
```

2.2 Part 1

```
create table manager
  (id integer
      constraint pk_manager primary key
      default next_value of number,
   bonus integer default 0);

create table location
   (id integer
      constraint pk_location primary key
      default next_value of number,
   street varchar(20) default 'unknown',
   city varchar(15) references city(name));

create table customer
  (id integer
      constraint pk_customer primary key
```

```
      default next_value of number,
   customer_name varchar(20) default 'unknown',
   location integer references location(id));

create table account
  (id integer
      constraint pk_account primary key
      references customer(id),
   balance real default 0.0,
   credit real default 0.0);

create table transact
  (id integer
      constraint pk_transact primary key
      default next_value of number,
   account integer references account(id),
   employee integer references employee(number),
   amount integer default 0,
   sdate date default current_date,
   stime time default current_time);
```

2.3 Part 2

```
insert into manager (id)
   select distinct manager
   from employee
   where manager is not null;

insert into manager (id)
   select distinct manager
   from dept
   where
      manager not in (select id from manager)
      and manager is not null;
```

2.4 Part 3

```
alter table employee
  add constraint cst_manages
  foreign key (manager) references manager(id);
```

```
alter table dept
  add constraint cst_head_of
  foreign key (manager) references manager(id);
```

2.5 Part 4

```
update manager
   set bonus = bonus + 10000
   where id in (select distinct manager from dept);
```

2.6 Part 5

```
insert into customer(id)
   select distinct account from debit
   where account is not null;

insert into account(id)
   select distinct account from debit
   where account is not null;

insert into transact (account, employee, amount, sdate)
   select debit.account,
          debit.employee,
          sale.quantity*item.price,
          debit.sdate
   from   debit join (sale join item on sale.item =
      item.number)
          on debit.number = sale.debit;
```

3 Assignment 3

Run this to set it all up:

```
whenever error continue;
set output off;

create databank dataBank;

create unique sequence number
```

```
initial_value = 0
increment = 1;

drop table manager cascade;
drop table customer cascade;
drop table account cascade;
drop table transact cascade;

set output on;
whenever error exit;

create table manager
  (number integer constraint pk_manager primary key,
   bonus integer default 0);

create table customer
  (number integer constraint pk_customer primary key
     default next_value of number,
   name varchar(20) default 'unknown',
   livesOnStreet varchar(20),
   city varchar(15) references city(name));

insert into customer (number) values (0);

create table account
  (number integer constraint pk_account primary key
     default next_value of number,
   customer integer default 0 references customer(
      number),
   balance real,
   credit real);

insert into account (number)
select distinct account from debit;

create table transact
  (number integer constraint pk_transact primary key,
   account integer references account(number),
   amount integer,
   sdate date default current_date,
   stime time default current_time,
```

```
    employee integer references employee(number));

insert into transact (number, account, sdate, employee)
select number, account, sdate, employee from debit;

insert into manager (number)
select distinct manager from employee
where manager is not null;

insert into manager (number)
select distinct manager from dept
where manager not in (select number from manager);

update manager set bonus = bonus + 10000
where number in (select distinct manager from dept);
```

3.1 Part 1

```
select * from employee;
```

3.2 Part 2

```
select number, name from dept;
```

3.3 Part 3

```
select number, name from parts
where qoh = 0;
```

3.4 Part 4

```
select number, name from employee
where salary between 9000 and 10000;
```

3.5 Part 5

```
select number, name, (startyear - birthyear) as
    startyear
from employee;
```

3.6 Part 6

```
select number, name from employee
where name like '%son,%';
```

3.7 Part 7

```
select number, name from item
where supplier in
   (select number from supplier
    where name = 'Playskool');
```

3.8 Part 8

```
select item.number, item.name
from item join supplier
on item.supplier = supplier.number
where supplier.name = 'Playskool';
```

3.9 Part 9

```
select number, name, color
from parts
where weight > (select weight
                from parts
                where name = 'tape drive');
```

3.10 Part 10

```
select p1.number, p1.name, p1.color
from parts as p1 join (parts as p2 join parts as p3
                       on  p2.name = 'tape drive'
                       and p2.name = p3.name)
on p1.weight > p2.weight;
```

3.11 Part 11

```
select avg(weight) as avarageWeightOfBlackParts
from parts
where color ='black';
```

3.12 Part 12

```
whenever error continue;
set output off;
drop view test_view;
set output on;
whenever error exit;

create view test_view as
   select supplier.number, supplier.name, quan*weight
      as totalweight
   from supplier, supply, parts
   where city in (select name from city
                  where state = 'Mass')
   and supplier.number = supply.supplier
   and supply.part = parts.number
   group by supplier.number, supplier.name, weight,
      quan;

select number, name, sum(totalweight) as totalweight
from test_view
group by number, name;
```

3.13 Part 13

```
whenever error continue;
set output off;
drop table cheap_item cascade;

CREATE TABLE cheap_item
   (number INTEGER CONSTRAINT pk_cheap_item PRIMARY KEY,
    name VARCHAR(20),
    dept INTEGER NOT NULL,
    price INTEGER,
    qoh INTEGER CONSTRAINT ck_cheap_item_qoh CHECK (qoh
       >= 0),
    supplier INTEGER NOT NULL);

set output on;
whenever error exit;
```

```
insert into cheap_item
select * from item
where price < (select avg(price)
               from item);

select * from cheap_item;
```

3.14 Part 14

```
whenever error continue;
set output off;
drop table cheap_item cascade;

create table cheap_item
  (number integer
      constraint pk_cheap_item primary key
      default next_value of number,
   name varchar(20),
   dept integer,
   price integer,
   qoh integer constraint ck_cheap_qoh check (qoh >= 0)
      ,
   supplier integer not null);

set output on;
whenever error exit;

insert into cheap_item (name, price, supplier)
select name, price, supplier from item
where price < (select avg(price)
               from item);

select * from cheap_item;
```

3.15 Part 15

```
select debit, price*quantity as totalprice from
sale join item
on sale.item = item.number;
```

3.16 Part 16

```
-- delete from supplier
-- where city = 'Los Angeles';

-- This happens:
-- MIMER/DB error -10106 in function EXECUTE
--        Referential constraint SYSADM.FK_ITEM_SUPPLIER
    violated
--        UPDATE/DELETE operation not valid for table
   SYSADM.SUPPLIER
-- That is, as there is a reference to the row in
   another table,
-- it cannot be deleted.
```

3.17 Part 17

```
whenever error continue;
set output off;
drop view sale_supply;

CREATE VIEW sale_supply(supplier, item, quantity) as
   SELECT supplier.name, item.name, sale.quantity
   FROM (supplier join item
         on supplier.number = item.supplier)
         left outer join sale on
         sale.item = item.number;
set output on;
whenever error exit;

select * from sale_supply;
```