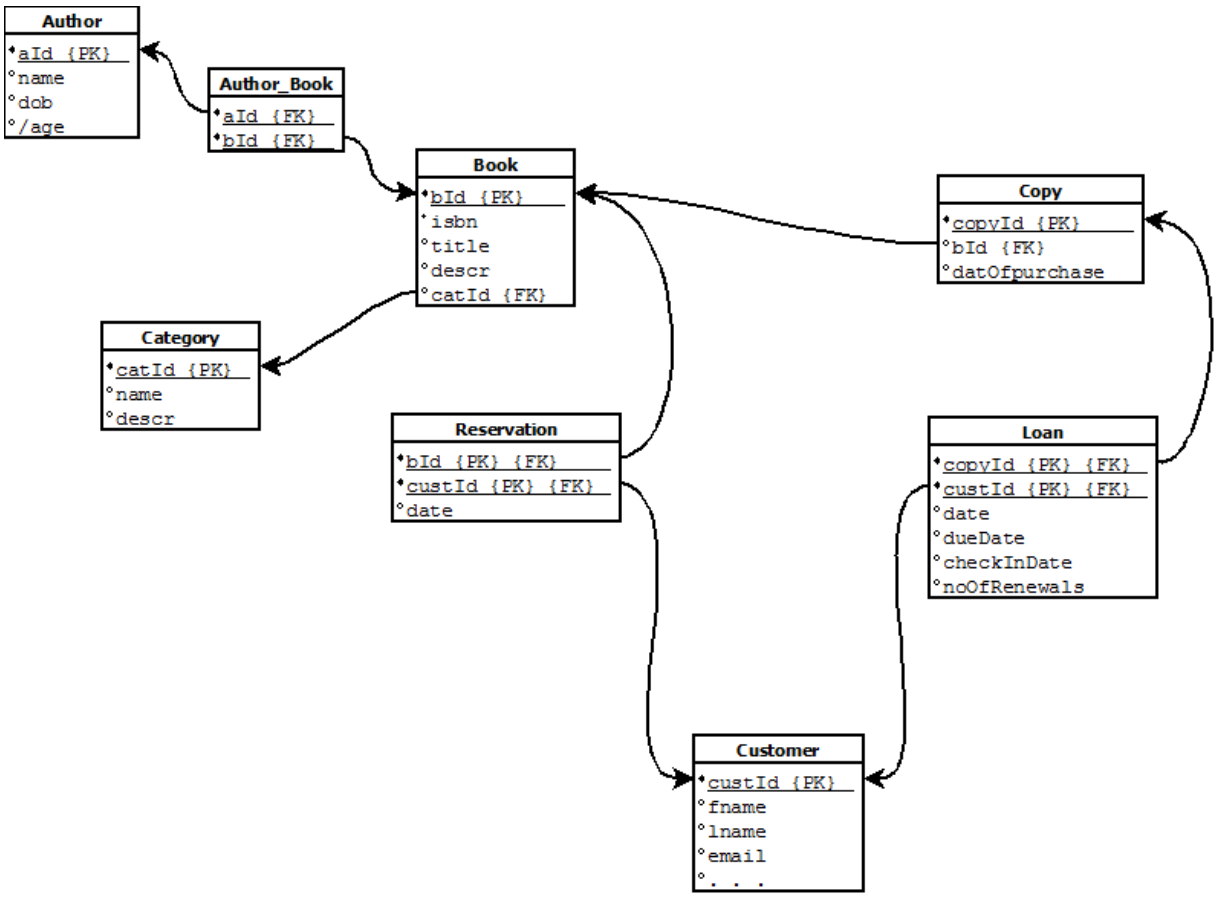
(Ofullständiga) kommentarer till övning 3

”Visa på alla fullständiga funktionella beroenden i er biblioteksmodell (använd gärna notationen på s 22 eller s 221 i kursboken).
Vilken normalform är ni i?”

Jag utgår från tabellerna i modellen från övning 2.



Author

En kandidatnyckel, aId.
FFB:
aId -> name
aId -> dob
(age är ett härlett attribut, kommer ej att lagras i tabellen)
Inga partiella eller transitiva beroenden samt alla determinanter är kandidatnycklar => relationen är i BCNF

Book

2 separata kandidatnycklar, bId och isbn.
FFB:
bId -> isbn, bId -> title, bId -> descr, bId -> catId
Motsvarande gäller för isbn!

Dessutom bId -> isbn (strider dock varken mot 2NF, 3NF eller BCNF)
Inga partiella eller transitiva beroenden och alla determinanter är kandidatnycklar => relationen är i BCNF.
Dock kan man överväga att ta bort BID och använda isbn som PK.

Category

En kandidatnyckel, catId.
FFB:
catId -> name
catId -> descry
Inga partiella eller transitiva beroenden och alla determinanter är kandidatnycklar => relationen är i BCNF.

Copy

En kandidatnyckel, copyId.
FFB:
copyId -> bId
copyId -> dateOfPurchase
Inga partiella eller transitiva beroenden och alla determinanter är kandidatnycklar => relationen är i BCNF.

Customer

En kandidatnyckel, custId.
FFB:
custId -> fname
custId -> lname
custId -> email
Om e-mail är att betrakta som unikt för varje person har vi också motsvarande som i Book.
Inga partiella eller transitiva beroenden och alla determinanter är kandidatnycklar => relationen är i BCNF.

Reservation

En kandidatnyckel: {bId, custId}
FFB:
{bId, custId} -> date
Inga partiella eller transitiva beroenden och alla determinanter är kandidatnycklar => relationen är i BCNF.

Author_Book, Loan

Tja, p.s.s. som ovan inser vi att dessa är i BCNF.

Blev inget bra exempel alltså – modellen var för genomtänkt från början för att meningsfullt kunna demonstrera normaliseringsprocessen. Tänkte inte på det…

”Uppdatera er modell så att ni åtminstone är i 3NF, helst BCNF.”
Som sagt, inget att göra här. I annat fall – bryt ut nya tabeller (se boken).

”Skapa följande lagrade procedurer
a) En procedur som visar på genomsnittligt antal lån per månad under 2009.”

Proceduren kanske kan se ut så här, skriven i PLPGSQL (i PostgreSQL är nyckelordet för procedurer och funktioner “FUNCTION”):

```
CREATE FUNCTION avg_loans_per_month(INT)
RETURNS DOUBLE PRECISION AS $$
DECLARE
    count_loans BIGINT;
BEGIN

    SELECT COUNT(date) FROM Loan
    WHERE EXTRACT(YEAR FROM date) = $1
    INTO count_loans;

    RETURN count_loans/12.0;

END;
$$ LANGUAGE PLPGSQL;
```

”b) En procedur som tar fram alla kunder som lånat en bok av en författare som ska skickas som parameter.”
I SQL blir det ungefär så här:

```
CREATE OR REPLACE FUNCTION cust_loan_author(VARCHAR)
RETURNS SETOF Customer AS $$

    SELECT cu.*
    FROM Customer cu, Loan l, Copy cp, Author_Book ab,
         Author a
    WHERE cu.custId = l.custId AND l.copyId = cp.copyId AND
          cp.bId = ab.bId AND ab.aId = a.aId
          and a.name LIKE $1;

$$ LANGUAGE SQL;
```

Vill du istället skriva funktionen I PLPGSQL, se t.ex.
http://wiki.postgresql.org/wiki/Return_more_than_one_row_of_data_from_PL/pgSQL_functions

”c) En procedur som har tre parametrar, kund-id, exemplar-id och dagens datum. Proceduren ska förlänga lånet som kunden har på detta exemplar och tredje parametern ska innehålla lånets nya utgångsdatum som svar.” - - - . . .