2018

Dokumentation

IB (INTERNETBANKEN)

RASMUS STEEN

Innehåll

ER-modell 2
Konceptuell modell 2
Förarbete 2
Beskrivning 3
Entiteter 3
Diagram 4
Logisk modell 6
Diagram 6
Beskrivning 6
Fysisk modell 7
SQL 7
Stödjer 9
Applikationer 11
Kassörsapplikation 11
Management applikation 12
Kundgränssnitt 13
Swish 14

ER-modell

Konceptuell modell

Förarbete

Jag tog texten Pre study och undersökte den då den beskrev vad databasen behövde klara av.

Jag har här identifierat möjliga tabeller, relationer, funktioner, kolumner och automatiska funktioner. Detta var något som jag gjorde för att få en överblick även på detaljerna.

Förklaring av färger:

- Gul Tabeller
- Grön Relationer
- Lila Funktioner
- Blå Kolumner
- Röd Automatisk funktion

Account and account holders

The IB shall provide bank accounts for customers (account holders). An account holder can have several accounts. Each account can be shared between two or more account holders.

The bank cashier creates new accounts and new account holders. An account can be shared between account holder(s). The cashier can enable for several account holders to share one account.

Information on an account holder is name, birth date, street and city of residence. A userid is created for each account holder. This is to be used for access verification. A 4-digit pincode shall also be created for each userid.

Information related to an account is the balance and the account number.

Account holder access

The account holder can access their accounts by a web interface. They shall be able to list their accounts with the balance. It shall be visible if the account is shared with another account holder.

The account holder can move money between their own accounts. Each time they move money they need to pay 3% of the moved amount to your own special and secret account. You wish to get money for a house in a sunny foreign country.

Swishing the money

The account holders can use their accounts through a swish-application where they can swish money to another account. Swishing is done by, using the swish application, inserting the userid and a 4-digit pincode followed by the account number, the amount to swish and the receiving account number. The pincode is private for each account holder.

Each time a swish is done, a fee of 2% of the swished amount should be taken and stored in your special and secret account. See this as your insurance for days to come; perhaps a nice lodge in sunny Spain can come out of this.

Calculating interest

The calculation of interest is done on a daily basis. It is performed manually by executing a stored procedure that calculates the interest for each account. The procedure takes the following arguments; interestRate, dateOfCalculation. The interest is calculated by interestRate * accountBalance / 365.

The result shall be stored in a separate table with the values of the calculated interest, date for calculation, and account number.

The yearly accumulated interest for a specific account can be calculated by summing all entries, for the specific account.

Logging

A log entry shall be written to a log table each time the balance is updated for an account. The entry should consist of the account number, current balance, amount changed and time.

Secret account

The secret account is an ordinary account. The fees stored in the secret account is taken from the existing money, you can't produce more money than the bank has.

Beskrivning

Databasen skall kunna hantera kunder, konton, ränta och loggning. Det skall gå att knyta ihop konton och kunder. En kund skall kunna flytta pengar från ett konto till ett annat. När detta sker skall en transaktionskostnad tas ut av banken och transaktionen ska loggas automatiskt. En kund skall också kunna se alla sina konton. En kassör kan skapa kunder och konton. Det ska även vara möjligt att få ut rapporter för cheferna om hur det går för banken.

Entiteter

De entiteter som kommer att användas är:

- Customer (Kunder)
- Account (Konton)
- Interest (Ränta)
- Log (Loggning)

Chefer och kassörer har endast access till databasen från sina arbetsdatorer och behöver ingen egen tabell för att kunna logga in sig själva.

Relationer

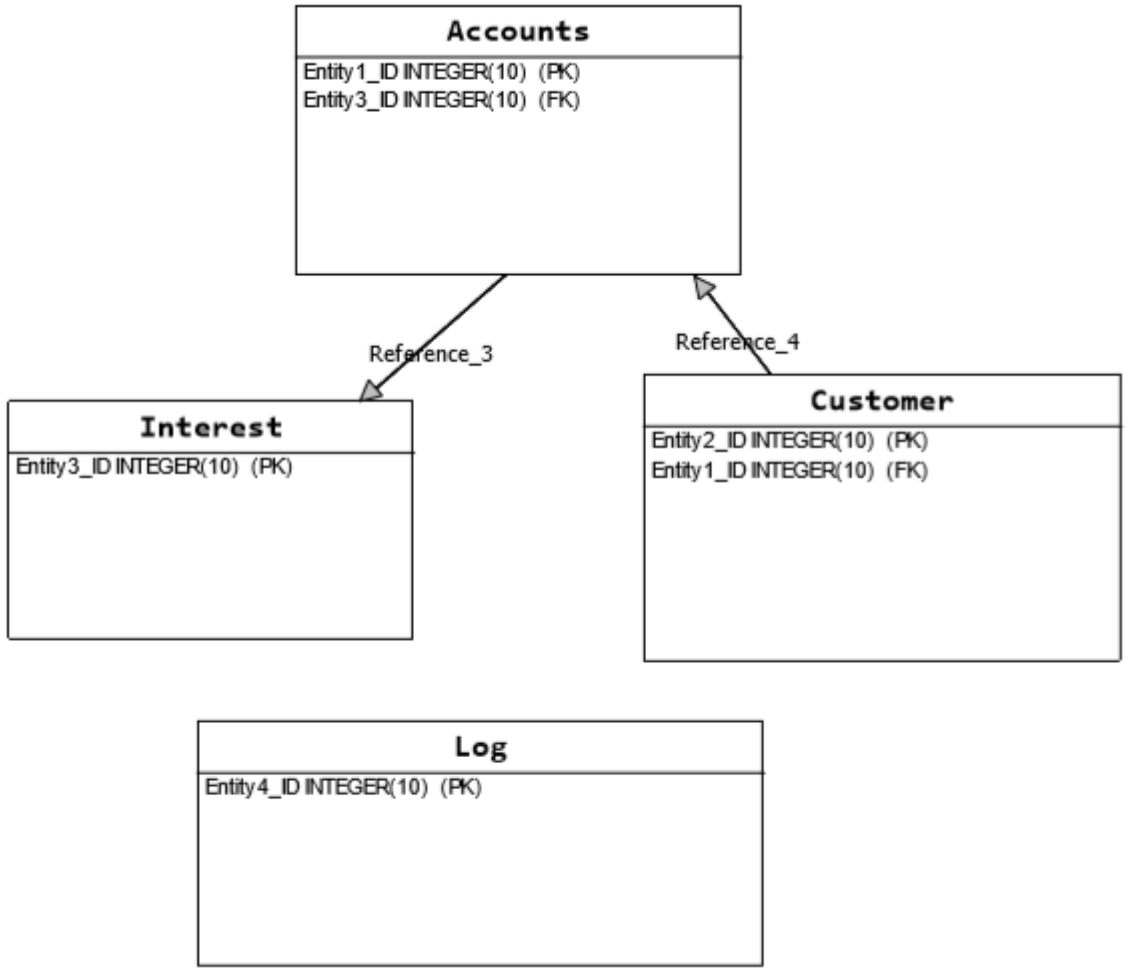
- En kund kan ha ett eller flera konton
- Ett konto kan ägas av en eller flera kunder och kan ha inga eller många räntor
- Ränta kan vara applicerat på ett konto

Tabell

Entiteter	**Kund**	**Konto**	**Ränta**
Kund		Har	
Konto	Ägs av		Har
Ränta			

Diagram

En första version



Relationer

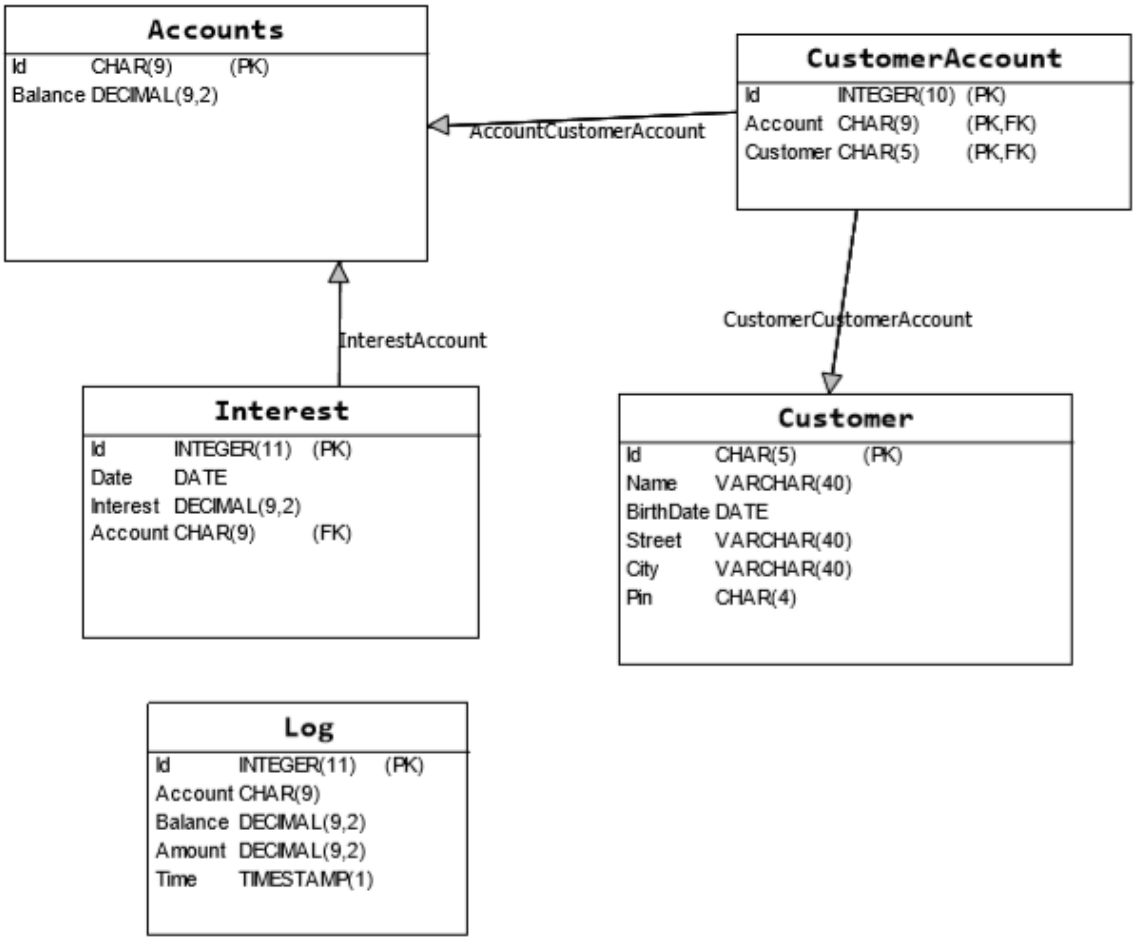
Relation	Typ
Customer – Account	N:M
Account - Interest	1:N

Kolumner

Tabeller	**Kolumner**
Account	Id, Balance
Customer	Id, Name, BirthDate, Street, City, Pin
CustomerAccount	Id, Account, Customer
Interest	Id, Date, Interest, Account
Log	Id, Account, Balance, Amount, Time

Logisk modell

Diagram



Beskrivning

Tabeller	**Kolumner**	**PK**	**FK**
Account	Id, Balance	Id	
Customer	Id, Name, BirthDate, Street, City, Pin	Id	
CustomerAccount	Id, Account, Customer	Id	Account, Customer
Interest	Id, Date, Interest, Account	Id	Account
Log	Id, Account, Balance, Amount, Time	Id	

Fysisk modell

SQL

Account

```
CREATE TABLE `account`
(
  `id` CHAR(9) NOT NULL,
  `balance` DECIMAL(9, 2),
  PRIMARY KEY (`id`)
) ENGINE = INNODB;
```

Customer

```
CREATE TABLE `customer`
(
  `id` CHAR(5) NOT NULL,
  `name` VARCHAR(40) NOT NULL,
  `birthDate` DATE,
  `street` VARCHAR(40),
  `city` VARCHAR(40),
  `pin` CHAR(4) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE = INNODB;
```

Interest

```
CREATE TABLE `interest`
(
  `id` INTEGER(11) AUTO_INCREMENT,
  `date` DATE,
  `interest` DECIMAL(9, 2),
  `account` CHAR(9) NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`account`) REFERENCES `account`(`id`)
) ENGINE = INNODB;
```

Log

```
CREATE TABLE `log`
(
  `id` INTEGER(11) AUTO_INCREMENT,
  `account` CHAR(9) NOT NULL,
  `balance` DECIMAL(9, 2),
  `amount` DECIMAL(9, 2),
  `time` TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE = INNODB;
```

CustomerAccount (många till många relation)

```
CREATE TABLE `customeraccount`
(
  `id` CHAR(15),
  `account` CHAR(9),
  `customer` CHAR(5),
  PRIMARY KEY (`id`, `account`, `customer`),
  FOREIGN KEY (`account`) REFERENCES `account`(`id`),
  FOREIGN KEY (`customer`) REFERENCES `customer`(`id`),
) ENGINE = INNODB;
```

Index

```
CREATE INDEX `index_customerName` ON `customer`(`name`);

CREATE FULLTEXT INDEX `index_logAccount` ON `log`(`account`);
```

Stödjer

Kassörs applikation

1. Lägga till en kund
2. Lägga till ett konto och knyta det till en kund
3. Knyta ett konto till ytterligare en kund
4. Visa alla kunder med deras konton och saldo
5. Visa alla konton och deras ackumulerade ränta

Funktion	**SQL**	**Returns**
Add customer	CALL addCustomer(name, date, street, city);	id and pin
Add account	CALL addAccount(balance, customerId);	id
Connect account	CALL connectAccount(customerId, accountId);	
Show customers	SELECT * FROM v_customeraccount;	
Show account interest	SELECT a.id, a.balance, SUM(i.interest) AS interest FROM account AS a LEFT JOIN interest AS i ON a.id = i.account GROUP BY a.id;	

Den lagrade proceduren addCustomer tar datum som CHAR(10) formaterad enligt yyyy-mm-dd.

Swish applikation

1. Flytta pengar

Funktion	**SQL**
Swish money	CALL swishMoney(customerId, pin, from, to, amount);

Kund gränssnitt

1. Visa alla konton för kunden och om dessa är delade
2. Flytta pengar mellan deras egna konton
3. Visa ackumulerad ränta för varje konto

Funktion	**SQL**
Show accounts	SELECT account, balance, getAccountCustomers(account) AS sharedBetween FROM v_customeraccount WHERE customer = customerId;
Move money	CALL moveMoney(customerId, from, to, amount);
Show interest	SELECT SUM(i.interest) AS interest FROM interest AS i JOIN v_customeraccount AS vca ON i.account = vca.account WHERE customer = customerId;

Rapporter

1. Visa summan för alla konton
2. Visa alla loggar
3. Visa det superhemliga kontot

Funktion	**SQL**
Show sum	SELECT SUM(balance) FROM account;
Show logs	SELECT * FROM log ORDER BY time DESC;
Show secret	SELECT * FROM account WHERE id = ‘secretacc’;

Fler rapporter

1. Räkna ut räntan för alla konton på en specifik dag
2. Visa all ackumulerad ränta för alla konton på en specifik dag
3. Visa all ackumulerad ränta för alla konton för ett helt år

Funktion	**SQL**
Perform calculation	CALL calculateInterest(interestRateInPercent, date);
Show on day	SELECT SUM(interest) AS interest FROM interest WHERE date = DATE(inDate);
Show for year	SELECT SUM(interest) AS interest FROM interest WHERE YEAR(date) = inYear;

Den lagrade proceduren calculateInterest tar datum som CHAR(10) formaterad enligt yyyy-mm-dd.

Alla datum är CHAR(10) formaterad enligt yyyy-mm-dd.

Applikationer

Kassörsapplikation

Denna applikation startas via ’node cli.js’ eller ’MODE=cas node cli.js’.

Du kommer då åt följande kommandon:

- menu
- exit
- setmode <mode>
- add customer <params[]>
- add account <params[]>
- connect <customet> <account>
- show customers
- show accounts

setmode

Tar en parameter som beskriver i vilket läge programmet kör.

Läge	**Värde**
Kassörsläge	cas
Management läge	man
Webläge	web

add customer

Tar fyra parametrar enligt: namn, födelsedatum, adress och stad

Returnerar kundId och pinkod.

```
CommandMe> add customer rasmus 1995-07-11 fårticksgatan134 enköping
+-------+------+
| id    | pin  |
|-------|------|
| ras02 | 3994 |
+-------+------+

CommandMe>
```

add account

Tar två parametrar enligt: saldo och kundId

Returnerar kontonummer.

```
CommandMe> add account 1234.56 ras02
+-----------+
| id        |
|-----------|
| 224410000 |
+-----------+

CommandMe>
```

connect account

Tar två parametrar enligt: kundId och kontonummer

Detta knyter ett konto till en kund och anropas när man vill lägga till en extra kund på ett konto.

```
CommandMe> connect ras01 224410000
Success!
CommandMe>
```

Management applikation

Denna applikation startas via ’MODE=man node cli.js’ eller ’node cli.js’ => setmode man.

Du kommer då åt följande kommandon:

- menu
- exit
- setmode <mode>
- interest <rate> <date>
- show sum
- show logs
- show secret
- show interest d <date>
- show interest y <year>

setmode

Tar en parameter som beskriver i vilket läge programmet kör.

Läge	**Värde**
Kassörsläge	cas
Management läge	man
Webläge	web

Interest

Tar två parametrar enligt: ränta datum

Denna funktion beräknar räntan för varje konto i banken.

```
CommandMe> interest 3 2018-03-16
Success!
CommandMe>
```

show interest d

Tar en parameter enligt: datum

Visar den totala räntan för en specifik dag.

```
CommandMe> show interest d 2018-03-16
+----------+
| interest |
|----------|
|   124.99 |
+----------+
```

show interest y

Tar en parameter enligt: år

Visar den totala räntan för ett helt år.

```
CommandMe> show interest y 2018
+----------+
| interest |
|----------|
|   612.25 |
+----------+
```

Kundgränssnitt

Kundgränssnittet startas via ’node index.js’ och gå till localhost:1337 i webläsaren.

PS: Om du angivit en annan port byter du ut den till det du angivit, samma sak gäller om du inte startar på den lokala datorn så byter du ut localhost till hostens ip eller domännamn.

Här behöver kunden logga in och hålls inloggad av en session i 30 minuter innan kunden ombedes logga in igen.

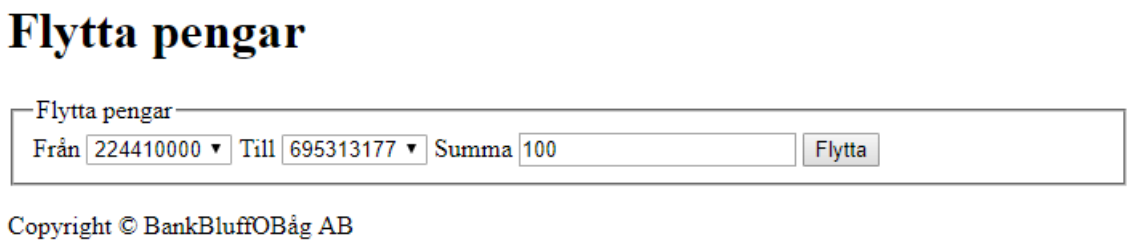
När kunden loggat in ser den sina konton.



Kunden kan även kolla räntan hen har samlat på sig.



Kunden kan flytta pengar mellan sina konton.



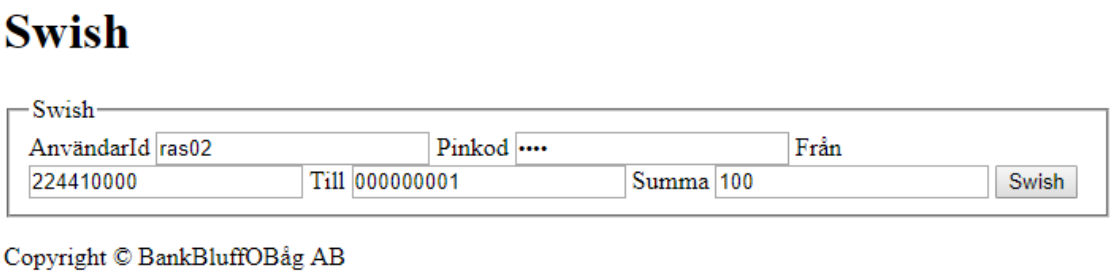
Vid en transaktion på 100 pengar dras det 3 kr extra från det skickande kontot då banken tar en transaktionsavgift på 3% av det skickade beloppet.

Alla dessa funktioner går att nå via navigationsfältet.

Swish

Startas via ’node index.js’ och nås via localhost:1337/swish

Här behövs ingen inloggning men kunden skriver in kundId och pinkod vid en transaktion.



Vid en swishning på 100 pengar dras det 2 kr extra från det skickande kontot då banken tar en transaktionsavgift på 2% av det skickade beloppet.