Exercise Sheet: Denormalization and Aggregates

Data, Data Storage, Data Collection

Exercise 1 – A Library Network

The public libraries of a city share one database, built from the following rules.

- Each member is a registered person with an identifier, a name and an address.
- A book has an ISBN, a title, and one or more authors. An author may write several books.
- The network has ten branches. Each branch has a name, unique in the city, and an address.
- The network holds several copies of a book. Each copy has an identifier and belongs to one branch.
- A loan is one copy lent to one member on one date. It records the date the copy is returned, left empty while the copy is on loan. A copy is lent at most once a day.

(**Q1**) Design a schema from these rules: give its tables, underline their keys, and name the foreign keys.

Every September, the city's schools send the library a flat file with schema Pupils(PupilName, Address, School, Class), with one row per pupil to give a library card, and no member identifier. Some of these pupils are already members.

(**Q2**) Where would you store this file, and how would you connect its rows to the members already stored?

(**Q3**) Which data of the school file, once stored, is redundant with the rest of the database? Why is this denormalization not a choice?

The library stores the copies and the loans in the tables

- Copies(<u>CopyID</u>, ISBN, Branch) and
- Loans(<u>CopyID</u>, <u>LoanDate</u>, MemberID, ReturnDate).

At the moment, each copy stays in its branch: a copy borrowed at one branch is brought back to it. The network plans to let copies float: a copy returned at another branch would stay there, and Copies would then record its new branch. The statistics of each branch must count the loans made there, so the designers already add a column Branch to Loans, filled at each loan with the branch of the copy.

(**Q4**) Under today's rule, show that this column is redundant: which dependency holds in the new Loans table, and which normal form does it violate?

(**Q5**) The plan is adopted. Copy C7 belongs to the North branch, is lent twice there, and is then returned at the South branch, where it is lent once more:

<u>CopyID</u>	<u>LoanDate</u>	Branch
C7	2 Sep	North
C7	20 Sep	North
C7	9 Oct	South

(Columns MemberID and ReturnDate not shown.) Does {CopyID} → {Branch} still hold in Loans? Which normal form is the Loans table in?

The library stores the books and the branches in the tables

- Books(ISBN, Title) and
- Branches(Branch, BranchAddress).

The network holds 200 000 copies and has recorded about two million loans over the years, with about 2 000 loans and returns a day. Its catalogue search sends about 100 000 queries a day. Each result shows the title of a book and, for each of its copies that is not on loan, the branch and its address. In the normalized schema, each search joins Books, Copies, Branches and Loans.

(**Q6**) To avoid these joins, a colleague proposes to store Copies(CopyID, ISBN, Title, Branch, BranchAddress). Which dependencies of this table break BCNF, and what can go wrong? What should be tried before changing the tables?

(**Q7**) The library's software is a hosted product: it creates an index on every key and every foreign key, and allows no other index and no materialized view. For each copy, a search still reads the loans of the copy to find whether one has an empty ReturnDate. Propose a change of the tables so that the search no longer reads Loans, and give two costs of keeping it.

Exercise 2 – Selling Concert Tickets

A small platform sells concert tickets. In a normalized schema, it stores its users, its events, and the seats of each event with the user who holds each one (empty when the seat is free):

Users

UserID	Name
U1	Nina
U2	Omar
U3	Lea

Events

EventID	Artist	Venue	Date
E1	The Pines	Arena Nd	5 Dec
E2	Mira Sol	Club 9	12 Dec

Seats

EventID	Seat	UserID
E1	A1	U1
E1	A2	U2
E1	A3	
E2	1	U1
E2	2	

The platform's workload consists of the following five operations.

W1: Show an event page: its artist, venue and date, and the seats still free.
W2: Show a user's "My tickets" page: for each ticket, the artist, the date and the seat.
W3: Book a free seat for a user.
W4: Change the date of an event.
W5: Once a month, count the tickets sold per venue.

Recall that an aggregate is a group of related data that has its own key, is stored as a single value that may contain lists and nested records, and is the unit in which the application reads and writes data.

(**Q1**) Propose aggregates for this workload, so that W1 and W2 each read a single aggregate. For each kind of aggregate, give its key and its content as a nested record, and write it out for E1 and for U1.

(**Q2**) Which data does your design copy? For each operation W1 to W5, does your design then read or write one aggregate, or several?

(**Q3**) Nina and Lea click on seat A3 of E1 at the same moment. Where must the state of the seat live so that only one of them gets it?

(**Q4**) A new event E3 opens its sale: 20 000 seats sold in ten minutes. What happens with one aggregate per event? With one aggregate per seat? Propose a boundary in between.

Exercise 3 – An ATM Cut Off From Its Bank

A bank stores each account as an aggregate, with two copies: one in the bank's data center, and one in the branch of a small town, which the town's ATM uses. The bank has one rule: the balance of an account is never below zero. Paul's account has a balance of 300. One evening, the link between the town and the data center is cut: a network partition. The ATM keeps working, but it can only reach the branch's copy; card payments go through the data center and its copy. At the ATM, Paul can check his balance, withdraw money, or deposit money.

(**Q1**) Replay the proof sketch of the CAP theorem on this bank: which write lands on which side of the cut, which read lands on the other side, and what are the read's two options?
(**Q2**) Which operations at the ATM can break the bank's rule if the ATM answers during the partition?
(**Q3**) During the partition, Paul withdraws 200 at the ATM, while a card payment of 200 goes through the data center. What is the balance once the two copies are brought together? Why did neither side refuse its operation?
(**Q4**) To stay useful during a partition, ATMs keep accepting withdrawals up to a total of about 200, and refuse beyond it. What does a higher limit buy, and what does it cost? How can the bank deal with a balance below zero after the partition?
(**Q5**) Is this ATM consistent or available during a partition?
(**Q6**) The bank's rule "a balance is never below zero" is also called a consistency rule. Is it the consistency of the CAP theorem, of ACID transactions, or of data quality?

Exercise 4 – Placing the Conversations on Nodes

A messaging app stores its data as aggregates: one per conversation, holding its members and its messages, and one per user, holding the user's profile, the list of their conversations and their count of unread messages. It has four conversations, C1 to C4, and four users, U1 to U4. Its data is sharded over three nodes, numbered 0, 1 and 2: each aggregate is placed on node $hash(key)$ mod 3. The hash values of the eight keys are:

key	C1	C2	C3	C4	U1	U2	U3	U4
$hash(key)$	17	22	9	32	26	13	30	10

Conversations C2, C3 and C4 were all created on 12 Dec.

(**Q1**) Place each aggregate on its node. How many nodes does showing conversation C2 ask? And listing all the conversations created on 12 Dec?

To survive a failure, the app also keeps a copy of each aggregate on the next node, $(hash(key) + 1)$ mod 3.

(**Q2**) Node 1 fails. Can every aggregate still be read? And if nodes 1 and 2 fail together?
(**Q3**) In a conversation between two users, sending a message writes the conversation aggregate and the recipient's aggregate, whose count of unread messages goes up. How many writes does one message now make?
(**Q4**) Conversation C3 is the chat of a live-streamed concert: 10 000 messages are sent in ten minutes, and node 0 is busy. Would adding nodes help?