3. Data Storage & Management

Summary

- Classify data by type—SQL for core business data, blob storage for static files, and specialized platforms for logs/analytics—balancing cost, performance, and isolation.
- Enforce tiered storage and lifecycle policies (hot→cold→archive) with clear backup, retention, and escrow processes to meet reliability, compliance (GDPR), and restore requirements.
- Partition tenant data (e.g., per federation/region or per club) to avoid noisy-neighbor issues, improve scalability, and enable differentiated SLAs or pricing.

Pending Decisions

- Decide tenant database partitioning strategy: per-club vs. per-federation/region vs. hybrid and the database services to use (e.g. Azure SQL Database or Managed Instance or Elastic Pools)
- Finalize retention periods and automated cleanup/archival rules for each data type.
- Specify escrow requirements and how they integrate into storage and export processes.

Open Items

- Develop and validate backup/restore procedures, documentation, and multi-region redundancy.
- Design static content lifecycle workflows (tier transitions and secure deletion).
- Define approach for meta-logs and audit trails (event sourcing vs. audit trailing).
- Plan offloading or migrating large historical tables (e.g., email logs) to cost-effective storage or analytics stores.

Overview

First and foremost thing to realize: what is the nature of our data?

Breakdown of the various types of data we and our users generate should reveal the variability in it. And in turn that needs to point to answers to the various storage types to use for various scenarios to have an ideal balance between ease and speed of writing and reading the data.

Bad practice: storing millions of log-like, email-like, event-like records in a SQL database is a bad idea. Especially if infrequently queried and thinking of long term storage.

Data types

Core business data → SQL Server

What the users will generate, what will fill our app up with life: clubs, members, labels, contacts, addresses, avatars, invoices, club events, etc...

How should that be stored? Is it relational? Or only parts of it? Apart from the generating core application, what else needs to access it? Do we need to make reports out of it?

- for the core, frequently accessed data: depending on the 16. Hosting. Use the well-known and well-reliable Microsoft SQL Server based relational database storage
  - CQRS Pattern - Azure Architecture Center | Microsoft Learn; and read-scaleout: Read Queries on Replicas - Azure SQL Database & Azure SQL Managed Instance | Microsoft Learn. Thus also reducing the impact of gathering analytics (e.g. making reporting queries)
  - from the beginning define the simplest of contracts/schema at least: e.g. internally using auto-incrementing integer IDs, while externally everything is exposed as UUID/GUIDs
- making reporting queries: really consider if this needs to happen on the same records (data) as business critical operations also take place on (e.g. if a dashboard tries to display overly complicated information about Members, make sure it doesn't take down various processes trying to insert or update members... they are not equally important). For this, consider again CQRS and read-scaleout when talking about SQL databases
  - for example consider the following: duplicate data into a separate store, like an ETL pipeline, transforming it into its optimal format for analytics (e.g. different schema), reporting, etc.. and let loose more dynamic, easily-changing queries on it without affecting business processes

Static content

.e.g. Images, Videos, PDFs, Email body

- store them in a file/blob storage
- important consideration of their access
  - certain files should be private and accessed only through authentication and authorization
    - e.g. Avatars, Invoices.
    - the main application (or through one of its microservices) should grant access to the file, something really should be the owner to reduce dangling and exposed data
  - there can be public content, which are ideally exposed through a CDN
- consider the various layers of data availability: hot, cold, archive
  - old, infrequently, archived data should really be moved into its proper tier and not live on an expensive layer for no reason
- try to be mindful of dangling data: really delete content when its not required anymore, to avoid trash piling up
  - can be done in a multi-stage process, e.g. tagging it, moving it down into a colder tier, then after a while of no access anymore then just trash it

Analytical data

Consider also 11. Analytics & Reporting. Most importantly: it does not need to live in the same place as the core business data

Log data

Not in the main store where the core business data resides. Especially if it is a SQL database.

Errors

- push to Sentry
- expose as OpenTelemetry, and will be pushed to somewhere based on 11. Analytics & Reporting

Performance data

Expose as OpenTelemetry, and will be pushed to somewhere based on 11. Analytics & Reporting

Current LISAX stats

- main SLQ database
  - over ***405 GB*** of SQL data
  - table statistics

Number of records	Number of tables
100M+	1
10M-100M	13
1M-10M	43
100K-1M	55
10K-100K	96
1K-10K	100
100-1K	64
<100	154

top tables

Table	Number of records	Candidate for bad choice of storage
EmailRecipients	120,820,127	✋ : too much history
Emails	91,788,163	✋ : too much history
BulkEmailRecipients	51,925,672	✋ : too much history
CourtBookingParticipants	51,749,292	
InvoiceLogs	50,772,444	✋ : not necessary to reside in the main database
AuditLogs	45,712,494	✋ not necessary to reside in the main database
CalendarEventDates	40,422,387	

PushMessageTokens	39,584,073	✋ : not necessary to reside in the main database
CalendarEventLocation	24,611,349	
InvoiceLines	24,514,924	
CalendarEvents	23,636,521	
CourtBookings	20,579,178	
Media	11,781,287	
Devices	11,492,017	✋ : not necessary to reside in the main database
Invoices	9,852,102	
TeamPersons	8,313,008	
PersonProducts	7,927,965	
CustomFieldValues	7,351,551	
Matches	6,315,290	
Presences	5,891,649	
Contacts	5,376,802	
HockeyExchange.Logs	5,113,520	✋ : why
UserLoginLogs	3,621,567	✋ : not necessary to reside in the main database

- static content (i.e blob) storage:
  - over **4.2TiB** data
  - over **127M** records
- cache, search index, logs etc:
  - 20 GB Elastic search + log
  - over 230 GB Azure logs

Tenants

- LISAX has > 1700 Tennis clubs and > 300 Hockey clubs.
- For optimal performance, reliability and security, it would be best to split up the storage.
  - ❓ split per club:
    - is it even possible (cost, maintenance, etc..) to support that many “databases”?
    - maybe a hybrid, balanced option?
  - ⭐ split per sport/federation/customer group/region
    - at least this much should be done
    - to avoid noisy neighbours, data leaks, closer physical data location, horizontal scaling, etc..
- if the system is split (data wise) in a good way then can consider imposing limitations and/or pricing per tenant/client. E.g. large clubs can opt in for more storage for an extra price

Backups

In combination with 16. Hosting | Local redundancy, backup in any case. Keep backups of crucial data (especially the core business data). Backups are only as good as their restore capabilities, so make sure it is known where those backups are and how to use them (i.e. documentation). And backups are in multiple places, ideally.

Data retention policies

For every data type, consider, even for the briefest moment, whether it should be with us forever or not. Core business data: probably yes. Logs: quite probably not. Static content... depends. Just have a think.

Meta logs, audit trails

- see also the logging and user behaviour tracking sections at 11. Analytics & Reporting | Data Collection
  - with every core data type, the question should be asked whether changes on or around this thing should be tracked and if so to what degree
- consider a form, a level of, a combination of event sourcing (e.g. Event Sourcing pattern - Azure Architecture Center) and audit trailing (e.g. Audit Trail - OutSystems 11 Documentation)

Escrow

From the beginning and all the way through certain data may need to confirm to an escrow process. So this to be kept in mind when designing and storing.