# Indexed DB

Coders.Bay Linz

Rijad Bajrektarevic

19/02/2024

# Agenda

## Introduction

IndexedDB a robust storage API that empowers users to store almost anything right in their browser. This is not just about storing simple data types, but also complex JavaScript objects, files, and blobs, making it a versatile tool for web development.

IndexedDB employs a key-value storage model. This means that each piece of data, or ‘value’, is associated with a unique identifier, or ‘key’. This key is used to quickly locate and retrieve the value from the database.

The ‘key’ can be any data type, and the ‘value’ can be any object.

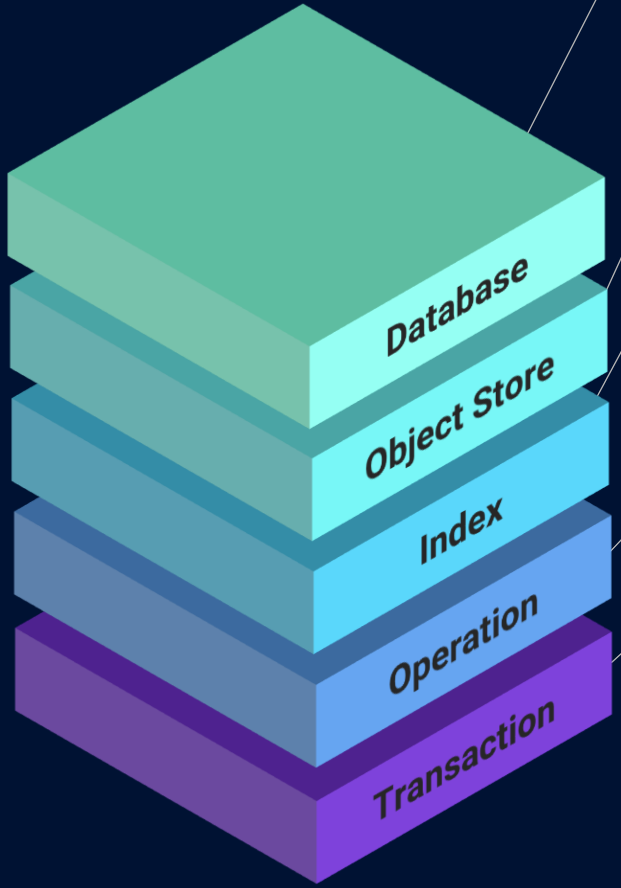
This flexible storage model is one of the reasons why IndexedDB is such a powerful tool for web development.”To obtain and update data in a sequence of transactions, you must first establish a connection to your database and specify its structure.

Source: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

In 2011, IndexedDB made its browser debut. In January 2015, the API was approved as a W3C standard; in January 2018, API 2.0 took its place. Work on API 3.0 is ongoing. As a result, IndexedDB is available in ordinary scripts and Web Workers, and it has good browser compatibility. Even with IE10, masochistic coders can give it a shot.

Source: https://www.sitepoint.com/indexeddb-store-unlimited-data/#:~:text=IndexedDB%20Introduction,API%202.0%20in%20January%202018.

# Basic overview and structure of IndexedDB



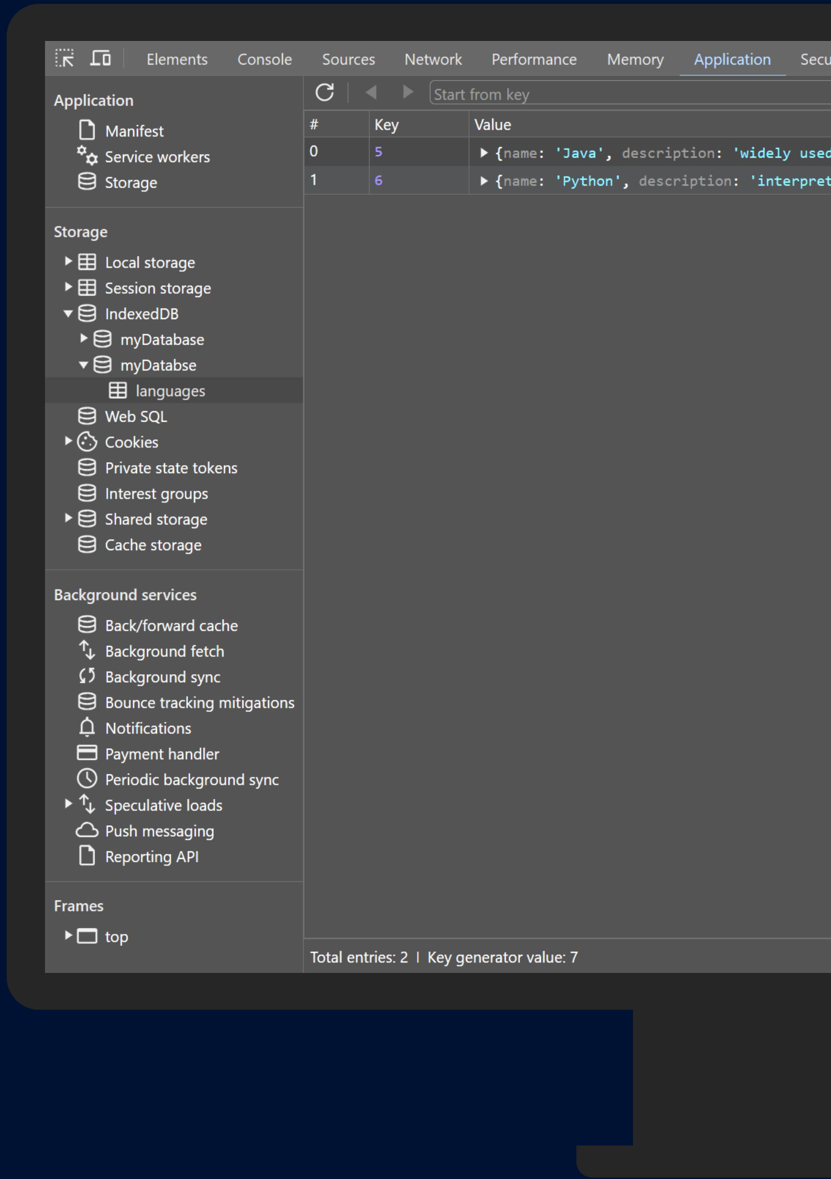
- the greatest IndexedDB level.
- multiple databases can be created within the IDB

- a single bucket for data storage
- are comparable to tables in RDMS

- data structure that improves the speed of data retrieval operations
- allowing you to quickly locate data without having to scan every record in the database.

- Operation = interaction with the database

- all read and write operations have to be a part of a transaction
- allows for atomic read-modify-write operations without the danger of other threads acting on the DB at the same time



Application
- Manifest
- Service workers
- Storage

Storage
- Local storage
- Session storage
- IndexedDB
  - myDatabase
  - myDatabse
    - languages
- Web SQL
- Cookies
- Private state tokens
- Interest groups
- Shared storage
- Cache storage

Background services
- Back/forward cache
- Background fetch
- Background sync
- Bounce tracking mitigations
- Notifications
- Payment handler
- Periodic background sync
- Speculative loads
- Push messaging
- Reporting API

Frames
- top

| # | Key | Value |
|---|---|---|
| 0 | 5 | ▸ {name: 'Java', description: 'widely usec |
| 1 | 6 | ▸ {name: 'Python', description: 'interpret |

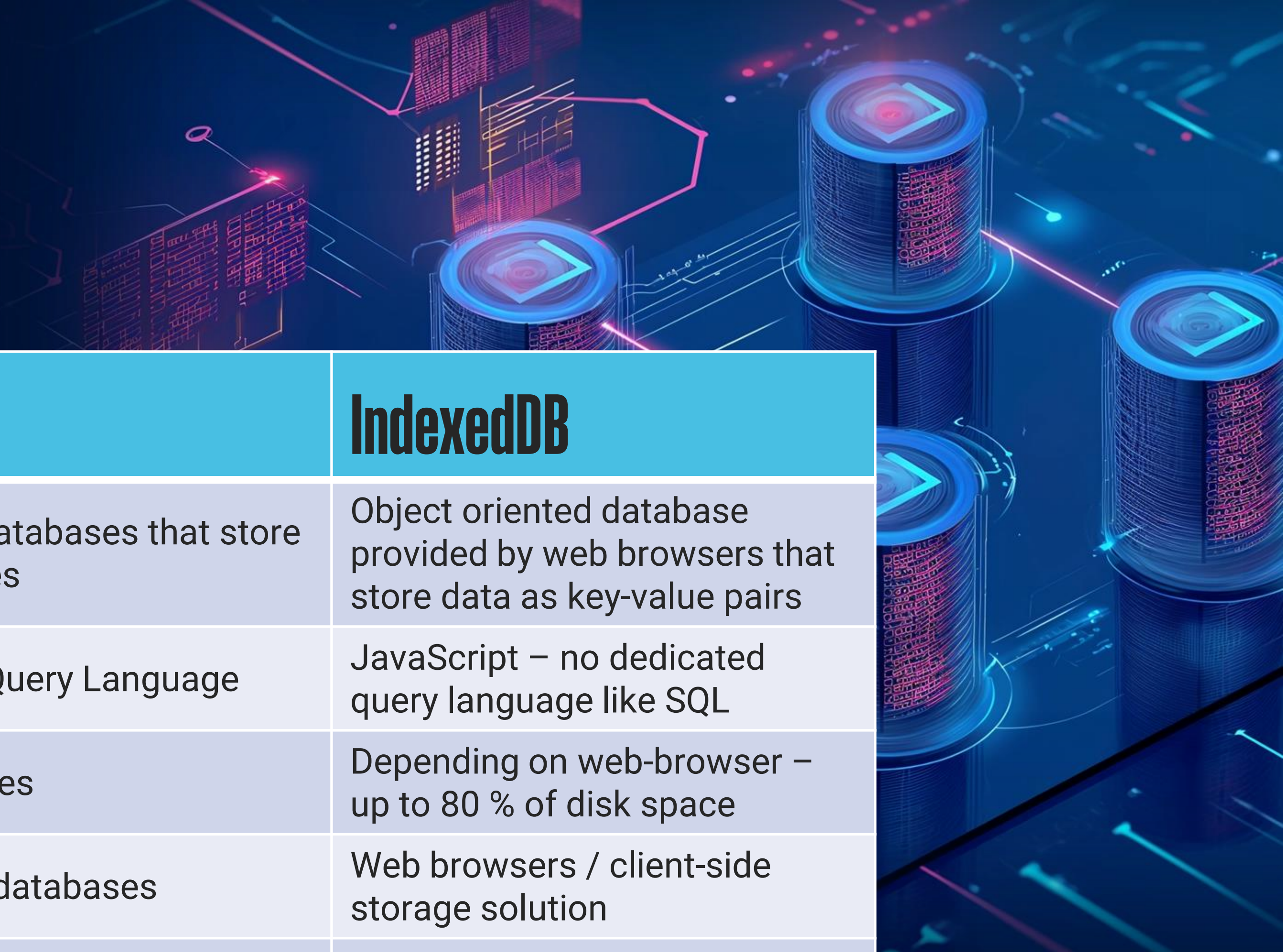
Total entries: 2 | Key generator value: 7

# LocalStorage vs IndexedDB

| | LS | IDB |
|---|---|---|
| Maximum Storage | 2.5 MB – 10 MB | Unlimited (depending on disk space & browser) |
| Data Types | String only | String, Date, Number |
| Allow Search | No | Yes |
| Performance | synchronous operations | asynchronous operations, usually slower than LS |
| Persistance | Data persistent, despite restart or system crash | |

Source: https://browsee.io/blog/unleashing-the-power-a-comparative-analysis-of-indexdb-local-storage-and-session-storage/

# RDBMS vs IndexedDB



| | RDBMS | IndexedDB |
|---|---|---|
| **Structure of data** | Relational databases that store data in tables | Object oriented database provided by web browsers that store data as key-value pairs |
| **Query language** | Structured Query Language | JavaScript – no dedicated query language like SQL |
| **Maximum capacity** | 524 Petabytes | Depending on web-browser – up to 80 % of disk space |
| **Storage location** | Server-side databases | Web browsers / client-side storage solution |
| **Concurrency** | Support concurrent transactions using control mechanisms | Support concurrent transactions by allowing only one transaction access to the object store in read-write mode |

Sources:
https://learn.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-ver16
https://stackshare.io/stackups/indexeddb-vs-sqlite

# Query language comparison

| | | |
|---|---|---|
| CREATE | INSERT INTO table_name<br>VALUES (value1, value2) | let transaction = db.transaction([„table“], „readwrite“);<br>let store = transaction.objectStore(„table“);<br>let request = store.add({column1: „value1“, column2: „value2}); |
| READ | SELECT * FROM table_name | transaction = db.transaction([„table“], „readonly“);<br>store = transaction.objectStore(„table“);<br>request = store.get(key); |
| UDPATE | UPDATE table_name SET column = value<br>WHERE condition | transaction = db.transaction([„table“], „readwrite“);<br>store = transaction.objectStore(„table“);<br>request = store.put({column1: „newValue1“, column2:<br>newValue2}, key); |
| DELETE | DELETE FROM table_name<br>WHERE condition | transaction = db.transaction([„table“], „readwrite“);<br>store = transaction.objectStore(„table“);<br>request = store.delete(key); |

Sources:
https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB
https://www.youtube.com/watch?v=yZ26CXny3iI&t=605s&ab_channel=AlexEagleson
https://www.youtube.com/watch?v=z-Lc2Og5AxQ&t=302s&ab_channel=Code180

## Showcase

# PROs & CONs

+

**Versatile Data Storage:** IndexedDB can store a variety of data types -> objects, arrays, blobs etc.

**Large Storage Capacity:** from a few megabytes to gigabytes

**Transactions Support:** data integrity through atomic read-update-write operations

**Fast Accessibility:** by storing your data in the browser, data can be accessed quickly

**Browser Support:** most modern browsers support IDB

–

**Steep Learning Curve:** difficult, if you‘re not familiar with JavaScript – SQL is easier

**Scalability Limitations:** while it is great, for client-side storage, it may not scale well for large datasets

**Synchronization Challanges:** data is only available in your browser

**Security:** while data is relatively secured in the browser, there are a few things to consider
- data is not encrypted automatically
- data is stored on your physical device
- plain text form using browser developer tools

## Usecases

**Offline data availabilityy:** data can be accessed even without being connected to the internet

**Performance optimization:** possibility to apeed up load times for repeat visits, enabling a good UX

Examples of data, that can be stored in the browser that‘s unrelated to any data on a server:

- To-Do list
- Calendar
- Games that are played locally

Google

Float

ORCA Service Technologies

SPACEX

mozilla

SIEMENS

JKU JOHANNES KEPLER UNIVERSITY LINZ

firefly SOFTWARE

mongoDB®

Source: https://app.theirstack.com/search/tech/new?query=JTdCJTIybmFtZSUyMiUzQSUyMkNvbXBhbmllcyUyMHVzaW5nJTIwSW5kZXhlZERCJTIyJTJD

## Q&A