IndexedDB

Transactional

What is IndexedDB?

IndexedDB API / blobs API [1]

	Cookie	Cookie	sessionStorage	localStorage	IndexedDB	WebSQL
	4kb	4kb	2.5 10M	2.5 10M	>100M	

IndexedDB

- 📝 **(NoSql)** - MySQL IndexedDB
- 🎨 - cookie localStorage sessionStorage IndexedDB
- 🧑‍💻 - IndexedDB localStorage
- 🎥 - IndexedDB
- 📤 - IndexedDB localStorage
- 🤹 - IndexedDB (transaction) MySQL

IndexedDB

- 🛢️ **IDBDatabase** ObjectStore
- 📊 **IDBObjectStore**
- 👣 **IDBRequest** indexedDB
- 👆 **IDBCursor**
- 📠 **IDBIndex**
- 🗃️ **IDBKeyRange**
- 💼 **IDBTransaction** indexedDB ACID

DataBase And Request

open close delete

```
// IDBDatabase
const database = await openDB('admin', 1)

function openDB (name, version = 1) {
  return new Promise((resolve, reject) => {
    // IDBRequest
    const request = indexedDB.open(name, version)
    request.onsuccess = event => resolve(event.target.result)
    request.onerror = event => reject(event.target.errorCode)
    // IDBDatabaseException.UNKNOWN_ERR(1):
    // IDBDatabaseException.NON_TRANSIENT_ERR(2):
    // IDBDatabaseException.NOT_FOUND_ERR(3):
    // IDBDatabaseException.CONSTRAINT_ERR(4):
    // IDBDatabaseException.DATA_ERR(5):
    // IDBDatabaseException.NOT_ALLOWED_ERR(6):
    // IDBDatabaseException.TRANSACTION_INACTIVE_ERR(7):
    // IDBDatabaseException.ABORT_ERR(8):
    // IDBDatabaseException.READ_ONLY_ERR(9):   READ_ONLY
    // IDBDatabaseException.TIMEOUT_ERR(10):
    // IDBDatabaseException.QUOTA_ERR(11):
    // IDBDatabaseException.VER_ERR(12):

  })
```

```
console.log('start')

const button = document.getElementById('button')
button.addEventListener('click', () => {
  console.log('click')
})
button.click()

openIndexedDb()

setTimeout(() => console.log('setTimeout'), 0)

Promise.resolve('Promise').then(msg => {
  console.log(msg)
})

//         button
sleep(2000)

setTimeout(() => this.log("setTimeout2"), 0)

button.dispatchEvent(new Event('click'))

console.log('end')
```

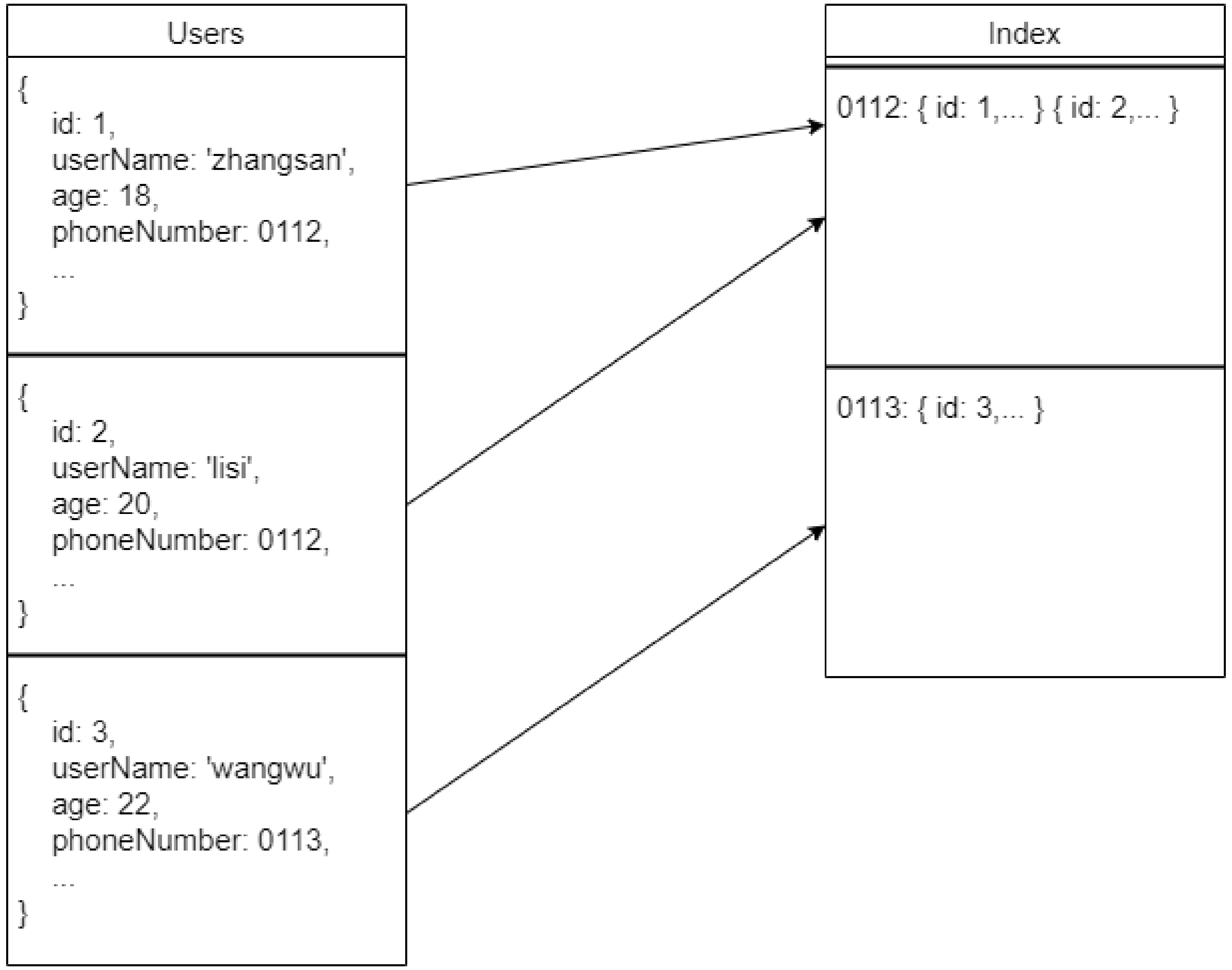
IDBObjectStore

```
{
  id: 1,
  name: '   ',
  age: 18,
  avatar: 'http://source.unsplash.com/collection/94734566/32×32',
  phoneNumber: '156****0112',
  userName: 'zhangsan',
  password: '123456',
  ...
}
```

IDBObjectStore & IDBIndex

open close delete insert get getAll

```
// IDBDatabase
const database = await openDB('admin', 1)

function openDB (name, version = 1) {
  return new Promise((resolve, reject) => {
    // IDBRequest
    const request = indexedDB.open(name, version)
    //
    request.onupgradeneeded = event => {
      const db = event.target.result
      // IDBObjectStore
      const objectStore = db.createObjectStore('user', {
        keyPath: 'id', //
        autoIncrement: true //
      })
      // IDBIndex
      objectStore.createIndex("userName", "userName", { unique: false });
      objectStore.createIndex("phoneNumber", "phoneNumber", { unique: false });
      objectStore.createIndex("age", "age", { unique: false });

      resolve(db)
    }

  })
```



IDBCursor

open close delete insert cursor queryByCursorOnIndex

```javascript
const list = [];
const store = this.db
  .transaction(this.storeName, "readwrite") // IDBTransaction
  .objectStore(this.storeName); // IDBObjectStore
const request = store.openCursor(); // IDBCursor
// IDBRequest
request.onsuccess = function (e) {
  const cursor = e.target.result;
  if (cursor) {
    list.push(cursor.value);
    cursor.continue();
  } else {
    console.log(list);
  }
}
```

```javascript
for (let i = 0; i < length; i++) {}
```

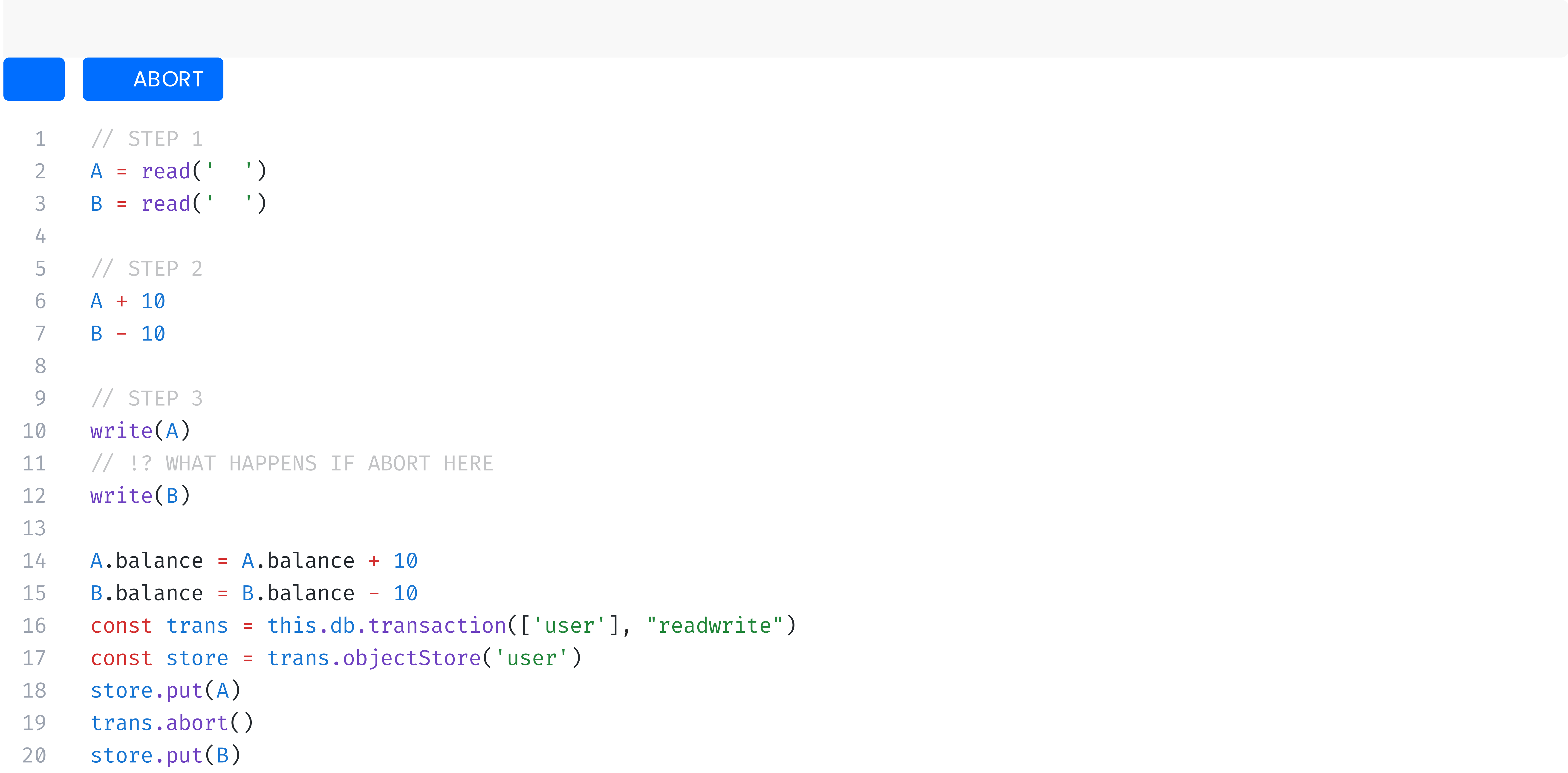
IDBCursor   IDBIndex

```
const list = []
const store = this.db
  .transaction(this.storeName, "readwrite")
  .objectStore(this.storeName)

const request = store
  .index(indexName)
  .openCursor(IDBKeyRange.only(indexValue))

request.onsuccess = function (e) {
  const cursor = e.target.result
  if (cursor) {
    list.push(cursor.value) // next object store object (book object)
    cursor.continue()
  } else {
    console.log("         ", list)
  }
}

request.onerror = function (e) {
  console.log(e)
}
```

IDBKeyRange

- Atomicity
- Isolation
- Consistency
- Durability

Atomicity



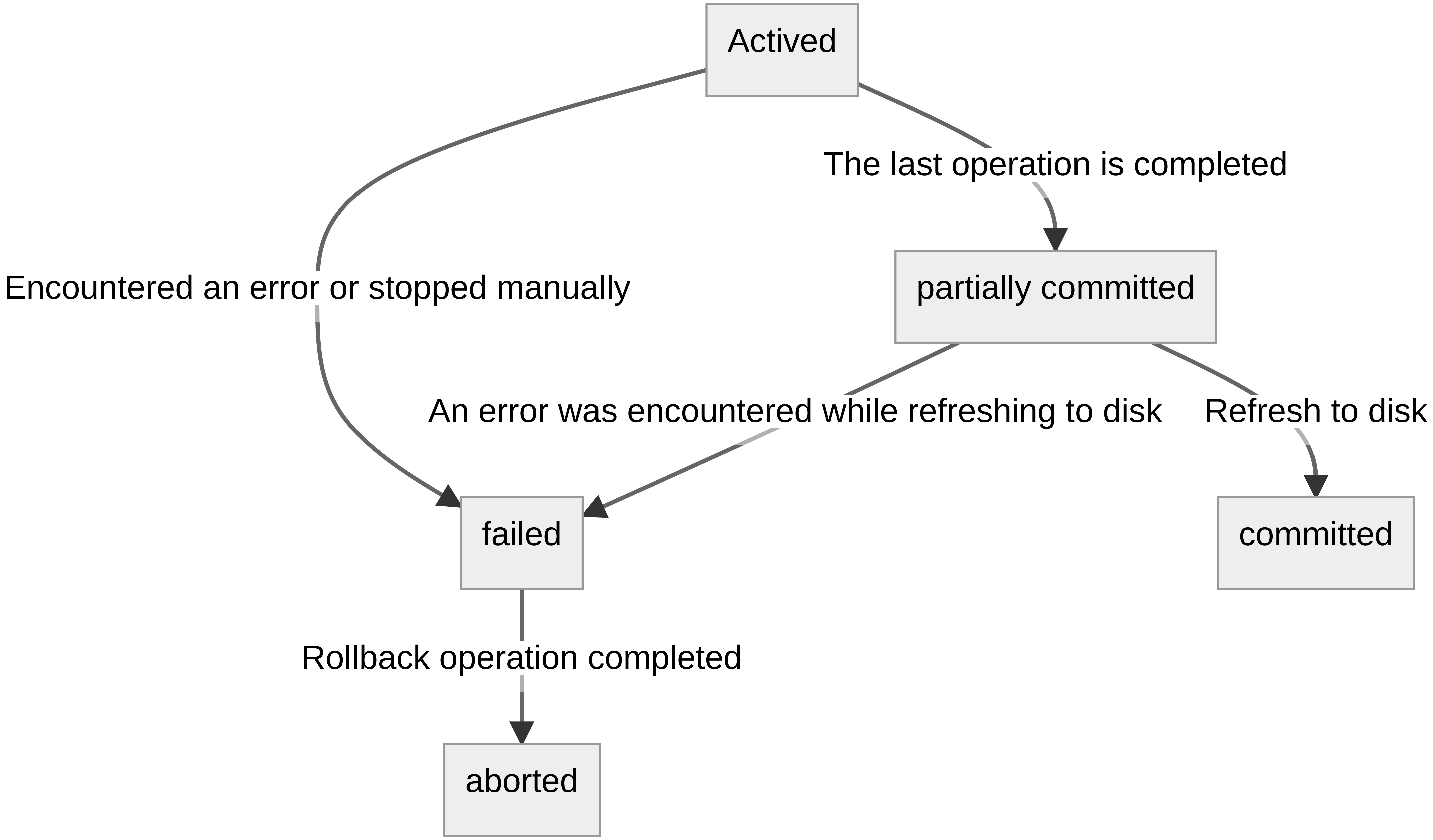
```
// STEP 1
A = read('   ')
B = read('   ')

// STEP 2
A + 10
B - 10

// STEP 3
write(A)
// !? WHAT HAPPENS IF ABORT HERE
write(B)

A.balance = A.balance + 10
B.balance = B.balance - 10
const trans = this.db.transaction(['user'], "readwrite")
const store = trans.objectStore('user')
store.put(A)
trans.abort()
store.put(B)
```

Isolation

A = read('   ')
A + 5
write(A)
B = read('   ')
B - 5
write(B)
	B2 = read(' ')
	B2 - 5
	write(B2)
	A2 = read(' ')
	A2 + 5
	write(A2)

A = read('   ') // 0
A2 = read('   ') // 0
A + 5 // 5
write(A) // 5
B = read('   ') // 10
B - 5 // 5
write(B) // 5
A2 + 5 // 5
write(A2) // 5
B2 = read('   ') // 5
B2 - 5 // 0
write(B2) // 0	

Consistency & Durability


Actived
The last operation is completed
Encountered an error or stopped manually
partially committed
An error was encountered while refreshing to disk
Refresh to disk
failed
committed
Rollback operation completed
aborted

IndexedDB

	symbols
Boolean	
String	
Date	
RegExp	lastIndex
Blob	
File	
FileList	
ArrayBuffer	

IndexedDB

Chrome Chrome Persistent storage

	Temporary storage	Persistent storage	Unlimited storage
...	...	...	...
Maximum storage space	Up to 20% of the shared pool.	As large as the available space on the hard drive. It has no fixed pool of storage.	As large as the available space on the hard drive.
...	...	...	...

chromium

```
// Pool size calculated by ratio.
int64_t pool_size_by_ratio = total * kTemporaryPoolSizeRatio;
  int64_t pool_size =
      kTemporaryPoolSizeFixed > 0
```

- ChromeCanvas
- spreadsheets

Learn More

Documentations · GitHub · Showcases