СБЕР

Как написать Lo-Fi приложение

А главное, зачем

Дмитрий Шведов
Блок «Сеть продаж»

Шведов Дмитрий

Frontend-разработчик в СБОЛ.Про «Подбор продукта»

6 лет в Сбере

11 лет опыт коммерческой разработки

Начинал программировать в качестве хобби с ZX-Spectrum


node
JS
PostgreSQL

Cloud-first

1. Проблемы с сетью
2. Проблемы с работоспособностью серверов
3. Физически данными владеет провайдер хостинга

WorkBox

Модули фреймворка

workbox-background-sync

Use background sync to reliably make a network request even if the user is offline.

workbox-broadcast-update

Send messages to pages when a cache is updated with a new response.

workbox-cacheable-response

Restrict which requests are cached based on a response's status code or headers.

workbox-core

Alter log levels and change cache names. Contains shared code used by all Workbox libraries

workbox-expiration

Removed cached requests based on the number of items in a cache or the age of the cached request.

workbox-google-analytics

Support for replaying offline Google Analytics interactions.

workbox-navigation-preload

Enable navigation preload, to get a network response for navigation requests faster.

workbox-precaching

Easily precache a set of files and efficiently manage updates to files.

workbox-range-requests

This modules provides support for responding to a `Range:` request using a slice of previously cached data.

workbox-recipes

Easily use common workbox patterns without needing to set them up yourself from individual packages.

workbox-routing

Routes requests in your service worker to specific caching strategies or callback functions.

workbox-strategies

A set of runtime caching strategies that will handle responding to a request, normally used with `workbox-routing`.

Workbox Vite plugin

```
export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      registerType: 'autoUpdate',
      manifest: {…
      },
      workbox: {
        globPatterns:['**/*.{js,csshtml,ico,png,svg,wasm}']
      }
    }),
```

IndexedDB?

WebSQL?

WASM + SQLite!

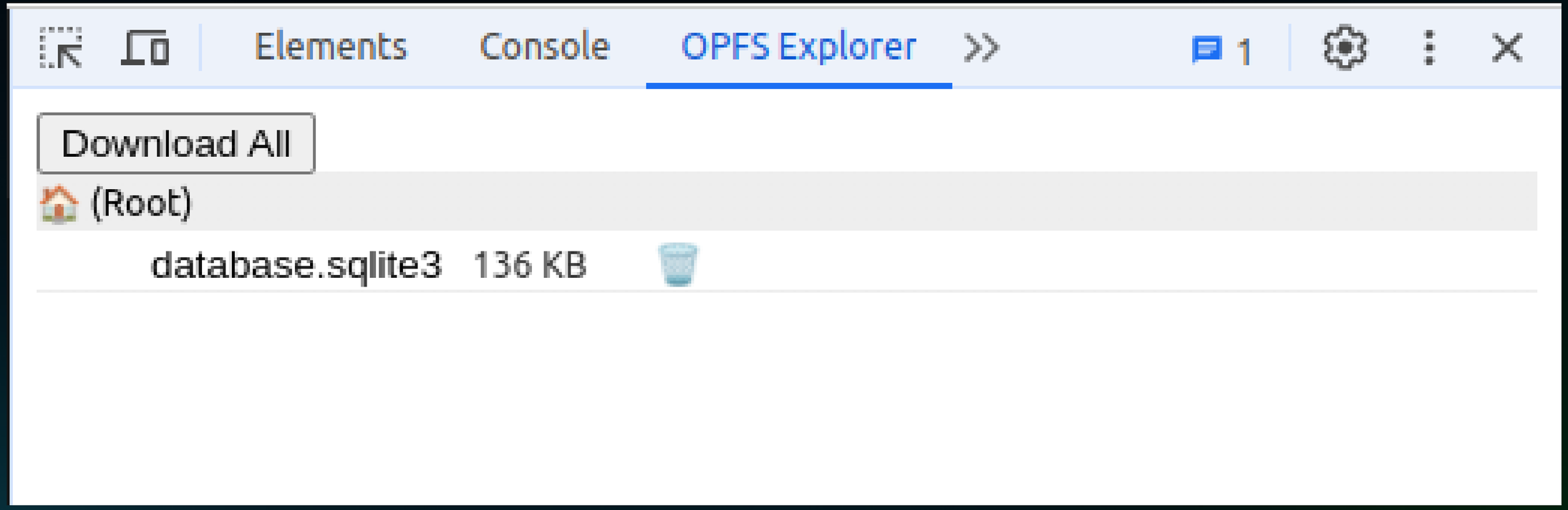
OPFS compatibility

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
`getDirectory`	✓ 86	✓ 86	✓ 111	✓ 72	✓ 15.2	✓ 109	✓ 111	✓ 74	✓ 15.2	✓ 21.0	✓ 109	✓ 15.2

OPFS Explorer



Программный доступ к OPFS

```
import { saveAs } from 'file-saver';

export const downloadDatabaseFile = async () => {
  const fsDirHandle = await navigator.storage.getDirectory()
  const fileHandle = await fsDirHandle.getFileHandle(DB_FILE_NAME)
  const file: File = await fileHandle.getFile()
  saveAs(file, "foo.sqlite");
}
```

Программный доступ к OPFS

```
import { saveAs } from 'file-saver';

export const downloadDatabaseFile = async () => {
  const fsDirHandle = await navigator.storage.getDirectory()
  const fileHandle = await fsDirHandle.getFileHandle(DB_FILE_NAME)
  const file: File = await fileHandle.getFile()
  saveAs(file, "foo.sqlite");
}
```

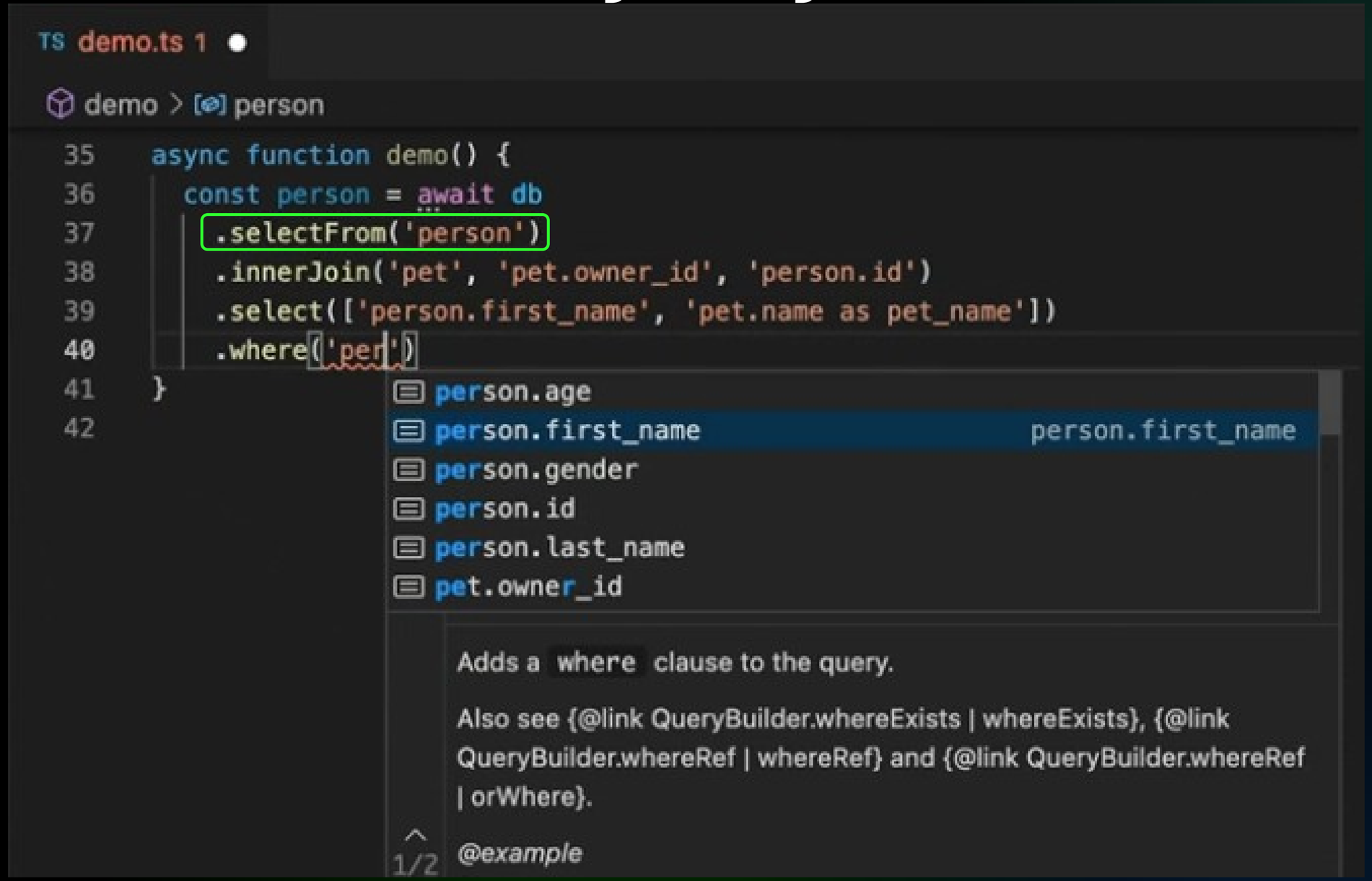
Программный доступ к OPFS

```
import { saveAs } from 'file-saver';

export const downloadDatabaseFile = async () => {
  const fsDirHandle = await navigator.storage.getDirectory()
  const fileHandle = await fsDirHandle.getFileHandle(DB_FILE_NAME)
  const file: File = await fileHandle.getFile()
  saveAs(file, "foo.sqlite");
}
```

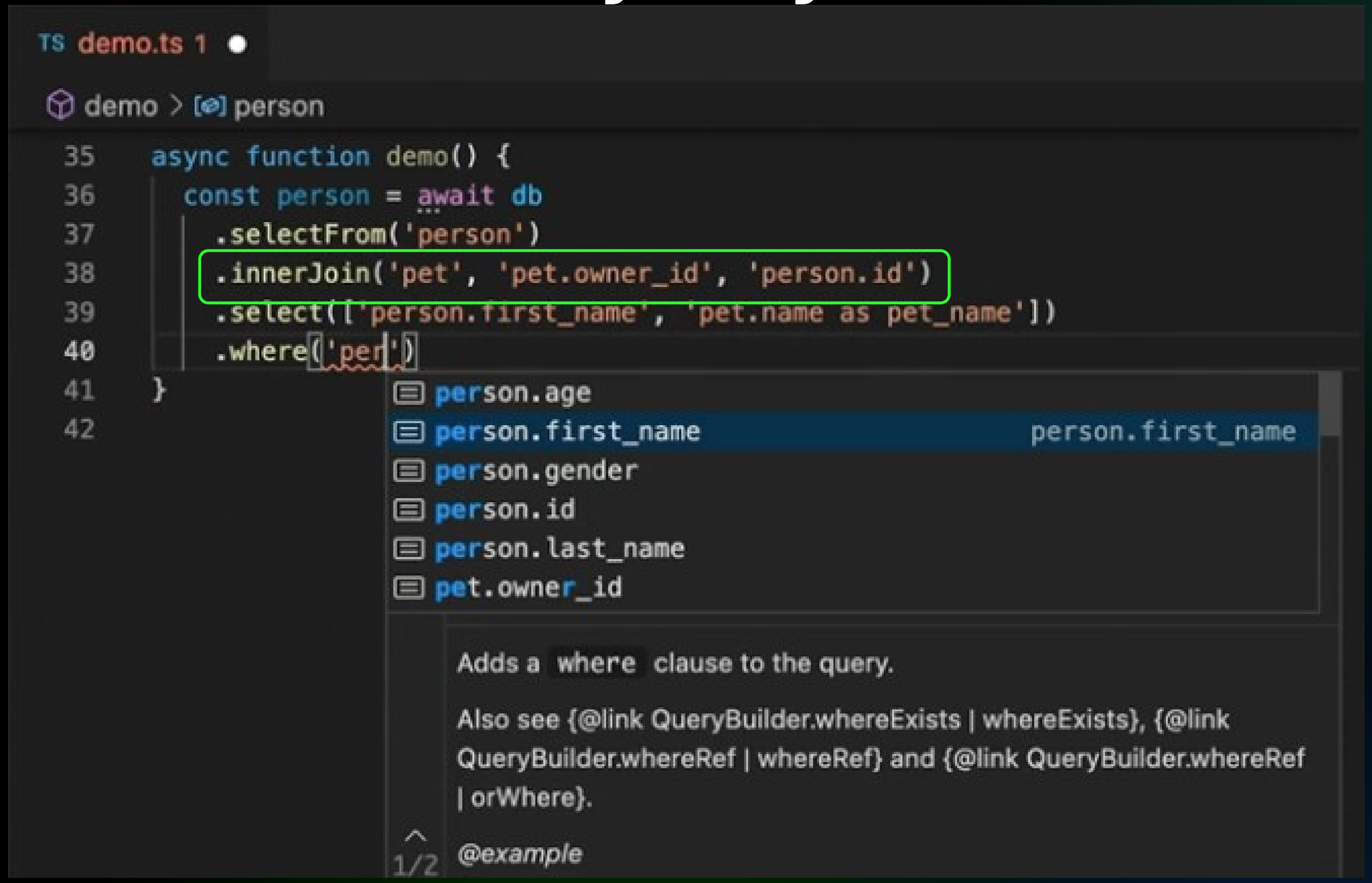
Программный доступ к OPFS

```
import { saveAs } from 'file-saver';

export const downloadDatabaseFile = async () => {
  const fsDirHandle = await navigator.storage.getDirectory()
  const fileHandle = await fsDirHandle.getFileHandle(DB_FILE_NAME)
  const file: File = await fileHandle.getFile()
  saveAs(file, "foo.sqlite");
}
```

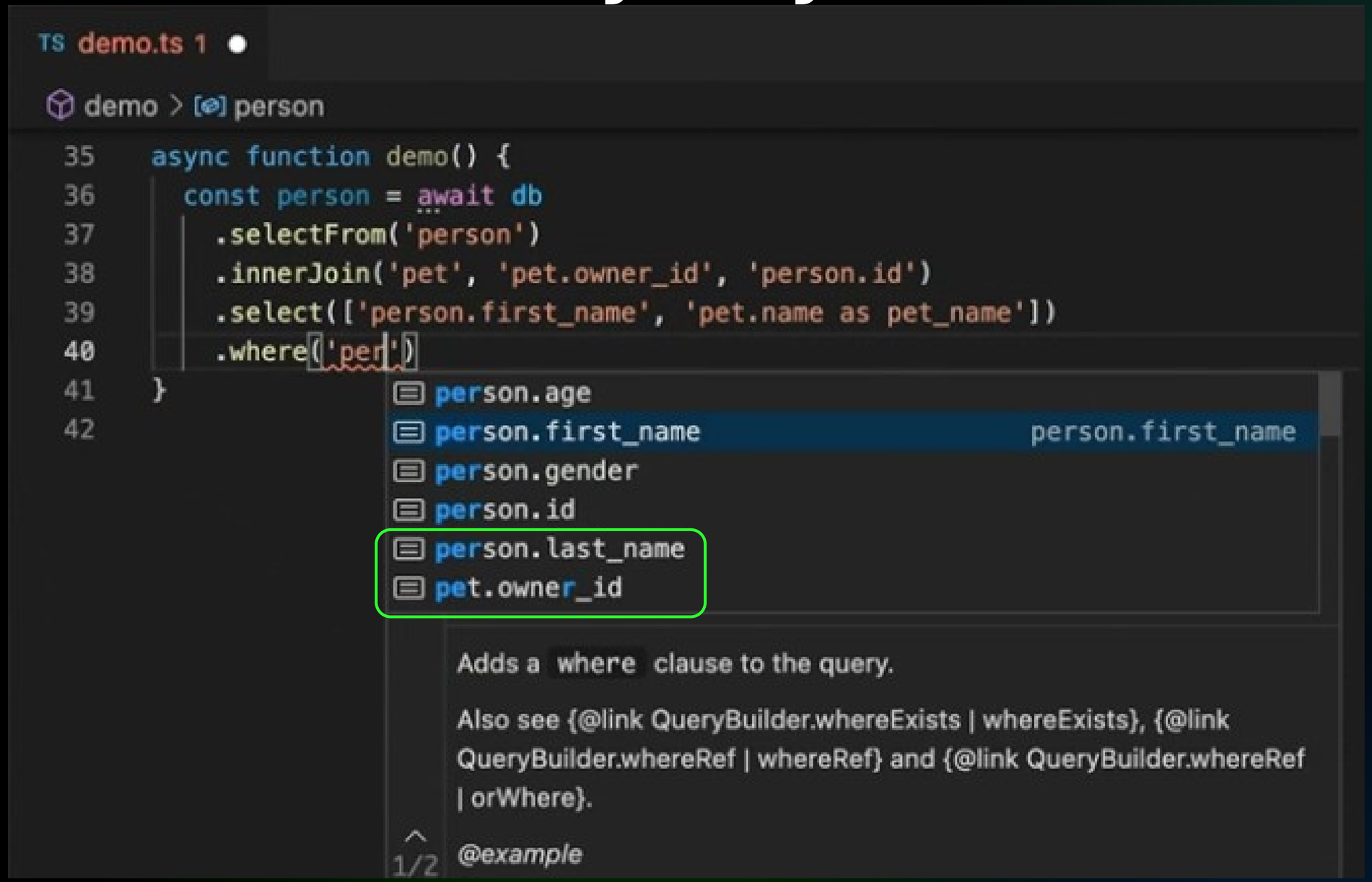
Kysely


TS demo.ts 1
demo > person
35 async function demo() {
36 const person = await db
37 .selectFrom('person')
38 .innerJoin('pet', 'pet.owner_id', 'person.id')
39 .select(['person.first_name', 'pet.name as pet_name'])
40 .where('per')
41 }
42
person.age
person.first_name person.first_name
person.gender
person.id
person.last_name
pet.owner_id
Adds a where clause to the query.
Also see {@link QueryBuilder.whereExists | whereExists}, {@link QueryBuilder.whereRef | whereRef} and {@link QueryBuilder.whereRef | orWhere}.
1/2
@example

Kysely


TS demo.ts 1
demo > person
35 async function demo() {
36 const person = await db
37 .selectFrom('person')
38 .innerJoin('pet', 'pet.owner_id', 'person.id')
39 .select(['person.first_name', 'pet.name as pet_name'])
40 .where('per')
41 }
42
person.age
person.first_name person.first_name
person.gender
person.id
person.last_name
pet.owner_id
Adds a where clause to the query.
Also see {@link QueryBuilder.whereExists | whereExists}, {@link QueryBuilder.whereRef | whereRef} and {@link QueryBuilder.whereRef | orWhere}.
1/2
@example

Kysely


TS demo.ts 1
demo > person
35 async function demo() {
36 const person = await db
37 .selectFrom('person')
38 .innerJoin('pet', 'pet.owner_id', 'person.id')
39 .select(['person.first_name', 'pet.name as pet_name'])
40 .where('per')
41 }
42
person.age
person.first_name
person.first_name
person.gender
person.id
person.last_name
pet.owner_id
Adds a where clause to the query.
Also see {@link QueryBuilder.whereExists | whereExists}, {@link QueryBuilder.whereRef | whereRef} and {@link QueryBuilder.whereRef | orWhere}.
1/2
@example

Инициализация Kysely

```
const { dialect } = new SQLocalKysely(DB_FILE_NAME);
dbRef.db = new Kysely<DB>({dialect});

const migrator = new Migrator({
  db: dbRef.db,
  provider: {
    async getMigrations() {
      const { migrations } = await import('./migrations');
      return migrations;
    },
  },
});

await migrator.migrateToLatest();
```

Миграция

```ts
export async function up(db: Kysely<any>): Promise<void> {
  await db.schema.createTable('note').ifNotExists()
    .addColumn('id', 'integer', (col) => col.primaryKey().autoIncrement())
    .addColumn('summary', 'varchar', (col) => col.notNull())
    .addColumn('parentId', 'integer')
    .addColumn('isOpened', 'boolean', (col) => col.notNull().defaultTo(true))
    .addColumn('dateCreated', 'timetz', (col) =>
      col.defaultTo(sql`(datetime(CURRENT_TIMESTAMP, 'localtime'))`).notNull()
    )
    .execute()
}

export async function down(db: Kysely<any>): Promise<void> {
  await db.schema.dropTable('note')
}
```

Миграция

```ts
export async function up(db: Kysely<any>): Promise<void> {
  await db.schema.createTable('note').ifNotExists()
    .addColumn('id', 'integer', (col) => col.primaryKey().autoIncrement())
    .addColumn('summary', 'varchar', (col) => col.notNull())
    .addColumn('parentId', 'integer')
    .addColumn('isOpened', 'boolean', (col) => col.notNull().defaultTo(true))
    .addColumn('dateCreated', 'timetz', (col) =>
      col.defaultTo(sql`(datetime(CURRENT_TIMESTAMP, 'localtime'))`).notNull()
    )
    .execute()
}

export async function down(db: Kysely<any>): Promise<void> {
  await db.schema.dropTable('note')
}
```

Схема БД

```ts
export type DB = {
  note: {
    id: Generated<number>;
    summary: string;
    parentId?: number | null;
    isOpened: Generated<boolean>;
    dateCreated: Generated<Date>;
  },
}
```

Инициализация Kysely

```
const { dialect } = new SQLocalKysely(DB_FILE_NAME);
dbRef.db = new Kysely<DB>({dialect});

const migrator = new Migrator({
  db: dbRef.db,
  provider: {
    async getMigrations() {
      const { migrations } = await import('./migrations');
      return migrations;
    },
  },
});

await migrator.migrateToLatest();
```

RTK Query

```
const apiRTK = createApi({
  endpoints: (build) => ({
    getAllNotes: build.query({
      queryFn: async (limit: number) => {
        const data = await dbRef.db!
          .selectFrom('note').selectAll().limit(limit)
          .execute();
        return {data};
      }
    }),
    ...getPersisterEndpoints(build),
```

RTK Query

```
const apiRTK = createApi({
  endpoints: (build) => ({
    getAllNotes: build.query({
      queryFn: async (limit: number) => {
        const data = await dbRef.db!
          .selectFrom('note').selectAll().limit(limit)
          .execute();
        return {data};
      }
    }),
    ...getPersisterEndpoints(build),
```

RTK Query

```
const apiRTK = createApi({
  endpoints: (build) => ({
    getAllNotes: build.query({
      queryFn: async (limit: number) => {
        const data = await dbRef.db!
          .selectFrom('note').selectAll().limit(limit)
          .execute();
        return {data};
      }
    }),
    ...getPersisterEndpoints(build),
```

RTK Query

```
const apiRTK = createApi({
  endpoints: (build) => ({
    getAllNotes: build.query({
      queryFn: async (limit: number) => {
        const data = await dbRef.db!
          .selectFrom('note').selectAll().limit(limit)
          .execute();
        return {data};
      }
    }),
    ...getPersisterEndpoints(build),
```

RTK Query

```
const {data: allNotes} = apiRTK.useGetAllNotesQuery(50);

const [trigge
const [trigge

const onSelec
  navigate(PA
}
```

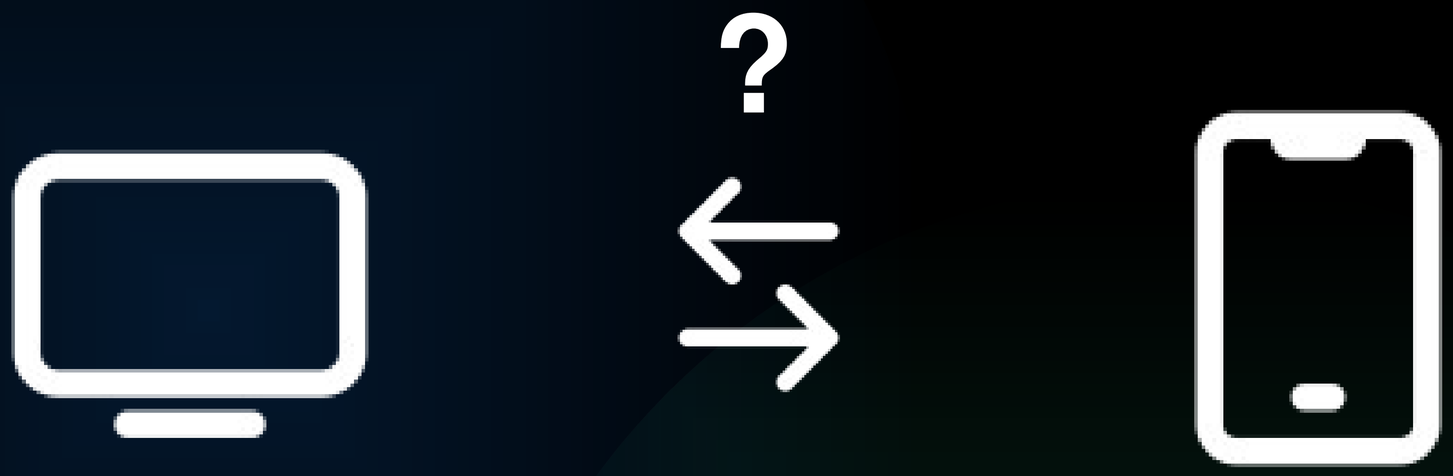
```
const allNotes: {
    id: number;
    summary: string;
    parentId: number | null | undefined;
    isOpened: boolean;
    dateCreated: Date;
}[] | undefined
```

RTK Query

```
const {data: allNotes} = apiRTK.useGetAllNotesQuery(50);

const [trigge
const [trigge

const onSelec
  navigate(PA
}
```

```
const allNotes: {
    id: number;
    summary: string;
    parentId: number | null | undefined;
    isOpened: boolean;
    dateCreated: Date;
}[] | undefined
```

https://minikent.ru



Использует Kysely

Есть react-hooks

Готовый сервер для бэкапа/синхронизации, end-to-end шифрование

Отсутствует авторизация, есть только bitcoin-подобная мнемоника из 12 слов

Подключается только к SQLite

Работает с готовой Postgres, через «Электрификацию»

Реактивная

От основателей local-first

Прекратили поддержку старой версии, а новая версия пока в alpha-release

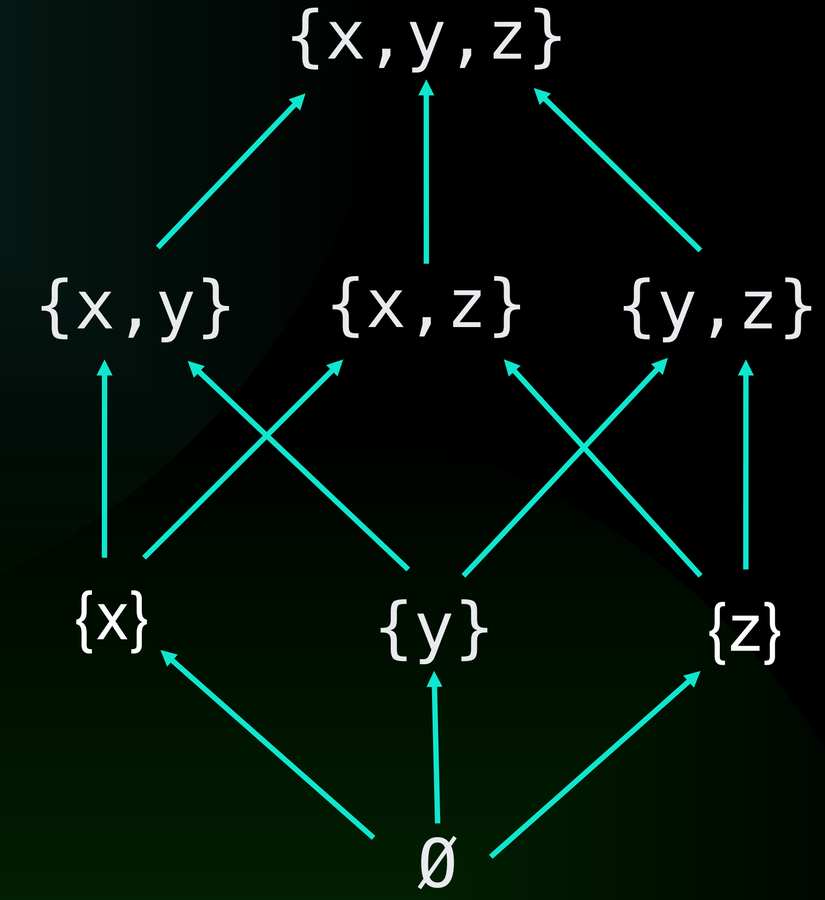
Используется во многих продуктах, например, RocketChat

Имеет свой протокол для синхронизации, но требует имплементации бекенда

Нет встроенного CRDT

CRDT

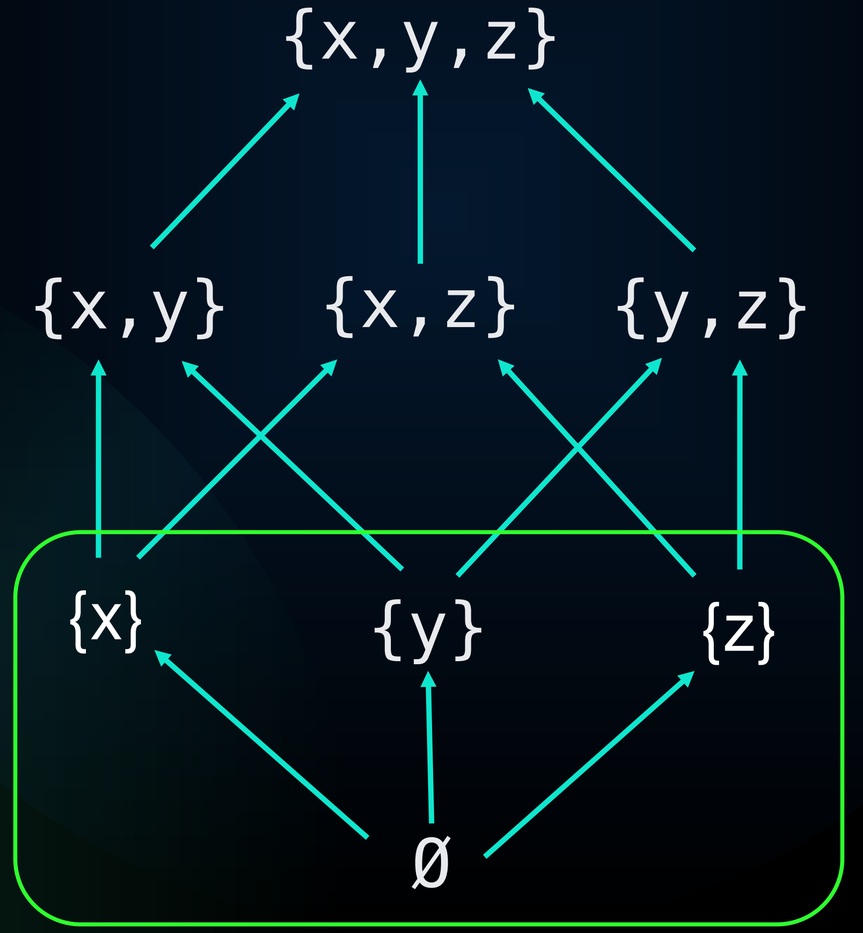
Conflict-free Replicated Data Types



CRDT

Conflict-free Replicated Data Types

Приложение может обновить любую реплику независимо, конкурентно и без координации с другими репликами

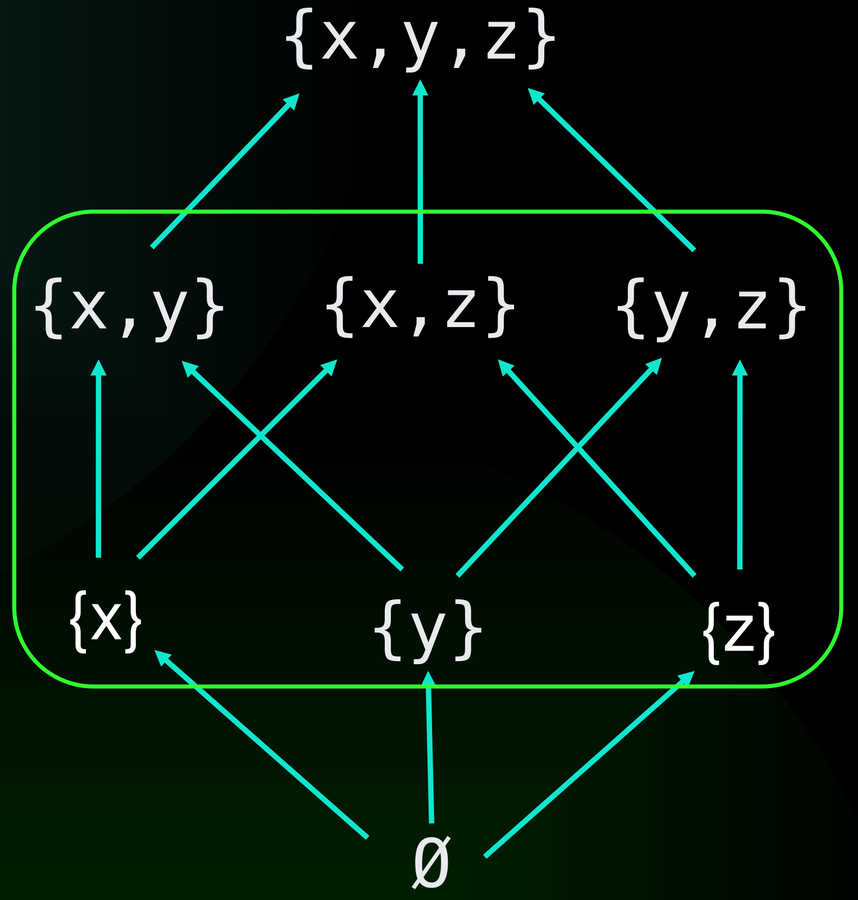


CRDT

Conflict-free Replicated Data Types

Приложение может обновить любую реплику независимо, конкурентно и без координации с другими репликами

Алгоритм, являющийся частью типа данных, автоматически разрешает любые неконсистентности, которые могут возникнуть



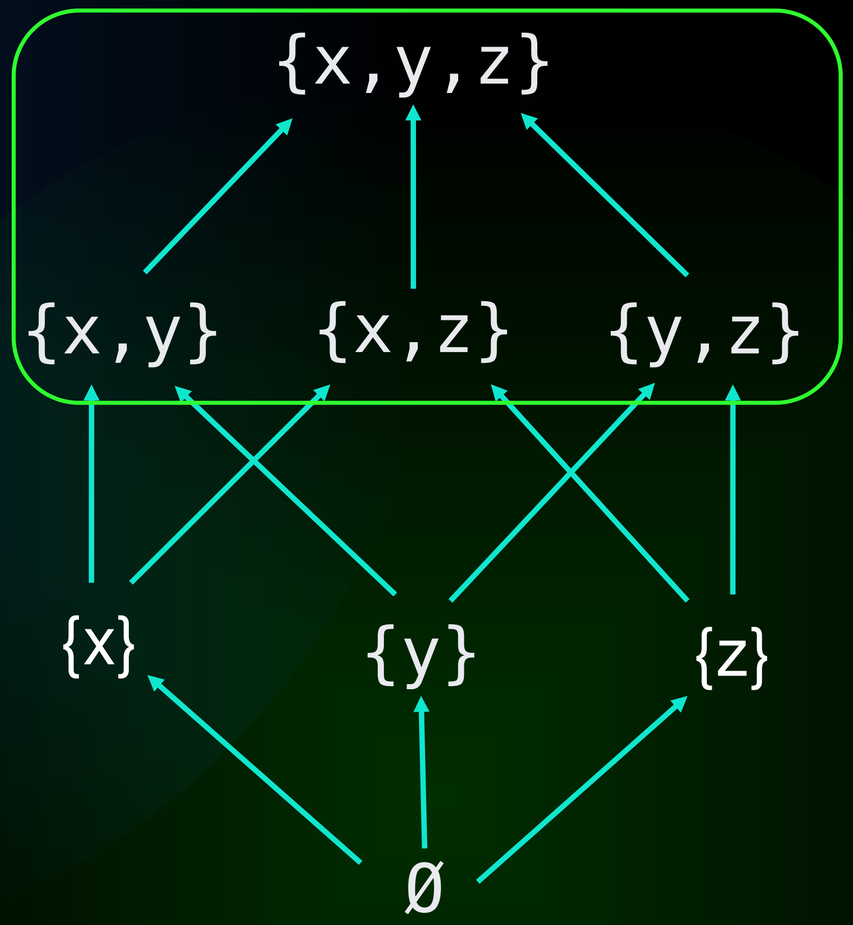
CRDT

Conflict-free Replicated Data Types

Приложение может обновить любую реплику независимо, конкурентно и без координации с другими репликами

Алгоритм, являющийся частью типа данных, автоматически разрешает любые неконсистентности, которые могут возникнуть

Хотя реплики могут иметь различное состояние в определенные моменты времени, они в конечном итоге гарантированно сойдутся




Yjs

Automerge

Martin Kleppmann

Идеалы Local-First

1. No Spinners

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device
3. The Network is Optional

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device
3. The Network is Optional
4. Seamless Collaboration with Your Colleagues

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device
3. The Network is Optional
4. Seamless Collaboration with Your Colleagues
5. The Long Now

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device
3. The Network is Optional
4. Seamless Collaboration with Your Colleagues
5. The Long Now
6. Security and Privacy by Default

Идеалы Local-First

1. No Spinners
2. Your Work Is Not Trapped On One Device
3. The Network is Optional
4. Seamless Collaboration with Your Colleagues
5. The Long Now
6. Security and Privacy by Default
7. You Retain Ultimate Ownership and Control

Lo-Fi

Dat Stack
hyper- swarm/dht & autobase (by dat-ecosystem)

TinyBase
James Pearce

Y-Sweet
Jamsocket team

hocuspocus
ueberdosis

Dexie Cloud
Awarica AB

Logux
Andrey Sitnik

VLCN
Matt Wonlaw

Automerge
Ink & Switch and contributors

Loro
Loro

Yjs
Kevin Jahns & contributors

Amplify DataStore
Amazon Web Services

Verdant
Grant Forrest

Trystero
Dan Motzenbecker / Dmotz

PowerSync
JourneyApps

Legend State
Jay Meistrich

Collabs
Composable Systems Lab @ CMU

Replicache
Rocicorp

DXOS
dxos.org

Jamsocket
Jamsocket team

Liveblocks Storage
Liveblocks team

m-ld
George Svarovsky et al.

Sockethub
Nick Jennings

```
/************** Begin file sqliteInt.h ****************************************/
/*
** 2001 September 15
**
** The author disclaims copyright to this source code.  In place of
** a legal notice, here is a blessing:
**
**    May you do good and not evil.
**    May you find forgiveness for yourself and forgive others.
**    May you share freely, never taking more than you give.
**
*************************************************************************
```

Спасибо!

Вопросы?