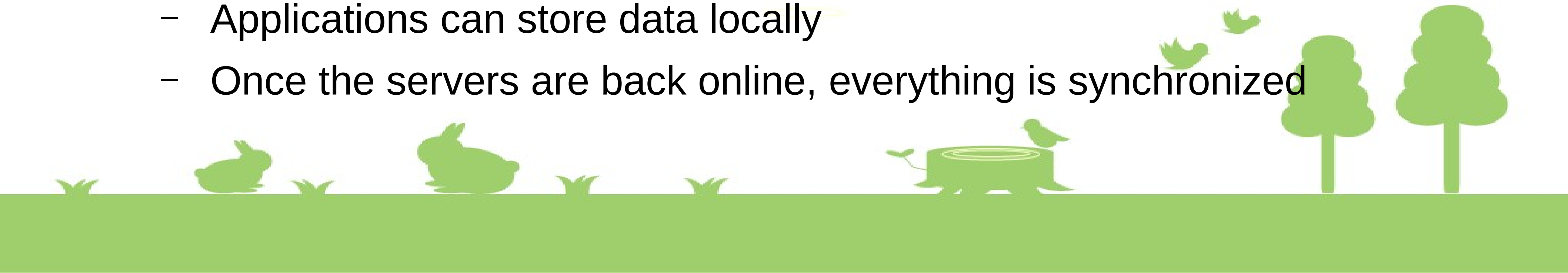
PouchDB

14507587

PouchDB - Basic Characteristics

- open-source JavaScript database
- In-browser database
  - Browser support: IndexedDB or WebSQL
- Inspired by CouchDB
- Designed for applications that should work offline as well as they work online
  - Applications can store data locally
  - Once the servers are back online, everything is synchronized

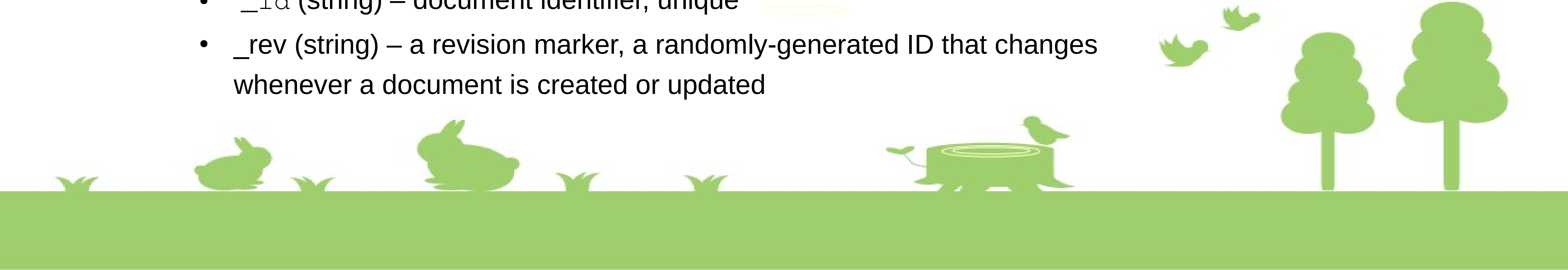


Download and installation

- `pouchdb` can be installed with `npm`
  - `npm install pouchdb`
- Also `pouchdb-server` is needed for inspecting databases
- `pouchdb-find` for more complex queries
- Web page with further information: *https://pouchdb.com/*

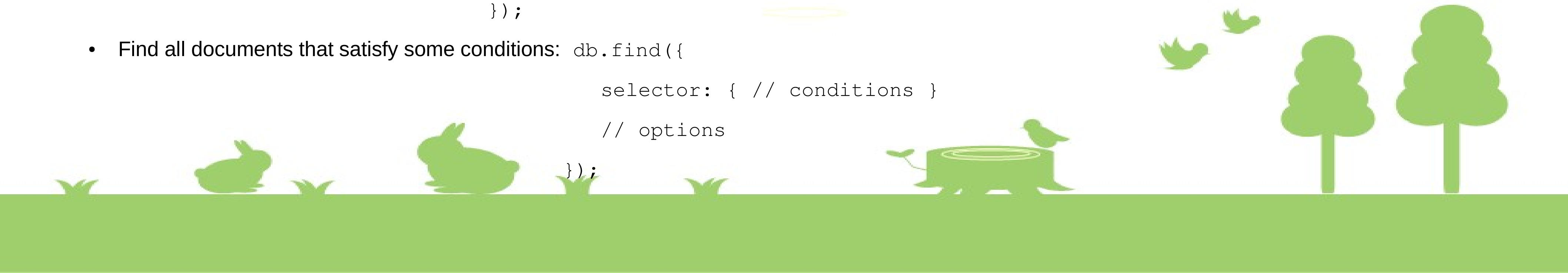
Supported Data Models

- PouchDB is a document store (document-oriented database system)
  - Records (documents) do not need to have a uniform structure
  - The types of the values of individual columns can be different for each record.
  - Columns can have more than one value (arrays).
  - Records can have a nested structure.
- **PouchDB document**:
  - In JSON format
  - Each document has following fields:
    - `_id` (string) – document identifier, unique
    - _rev (string) – a revision marker, a randomly-generated ID that changes whenever a document is created or updated



DML Operations

- Create a new local database: `var db = new PouchDB('name');`
- Create a new remote database (or connect to an existing one): `var db = new PouchDB('link');`
  - `link` should be path to a database in CouchDB
- Storing a document: `db.put(doc);`
- Get a document with _id: `db.get(id).then(function (doc) { // do somethinf with doc });`
- Updating document:
```
db.get(id).then( function (doc) {
                     // do update on doc
                     return db.put(doc);
                 });
```
- Remove a document:
```
db.get(id).then(function (doc) {
                     return db.remove(doc);
                 });
```
- Find all documents that satisfy some conditions:
```
db.find({
     selector: { // conditions }
     // options
});
```



Querying

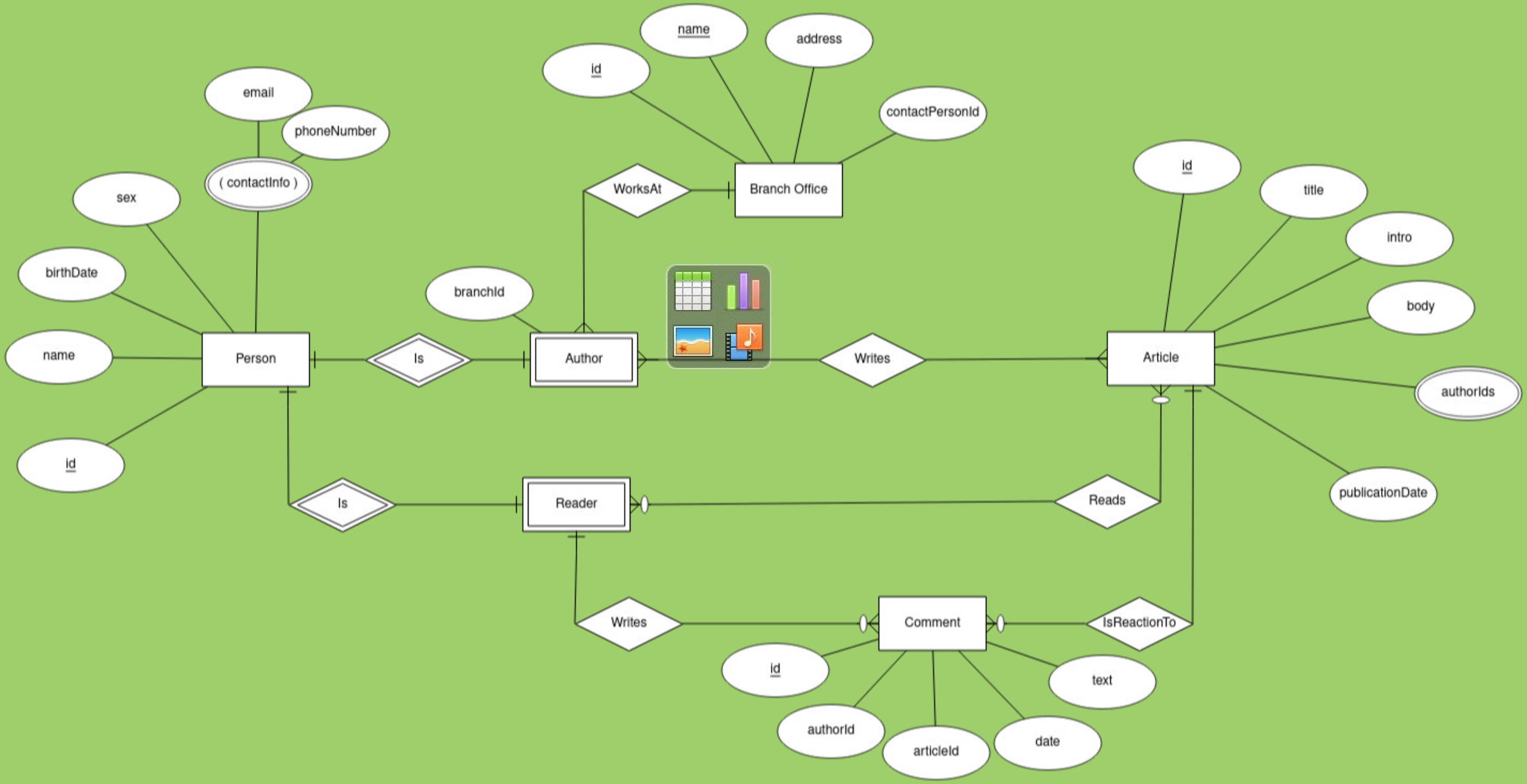
- Bulk operations:
  - `db.allDocs([options], [callback])`
    - Read all documents from database “db” (SQL: SELECT * FROM db)
  - `db.bulkDocs(docs, [options], [callback])`
    - Write all docs to database “db” (SQL: repeated INSERT)

- `db.get(docId, [options], [callback])`
  - Fetch a document with `_id` matching `docId` (SQL: SELECT * FROM db where _id = docId)
- `db.find(request [, callback])`
  - return the list of documents that match the request (SQL: SELECT * FROM db where *some conditions*)
  - (in separate plugin pouchdb-find)

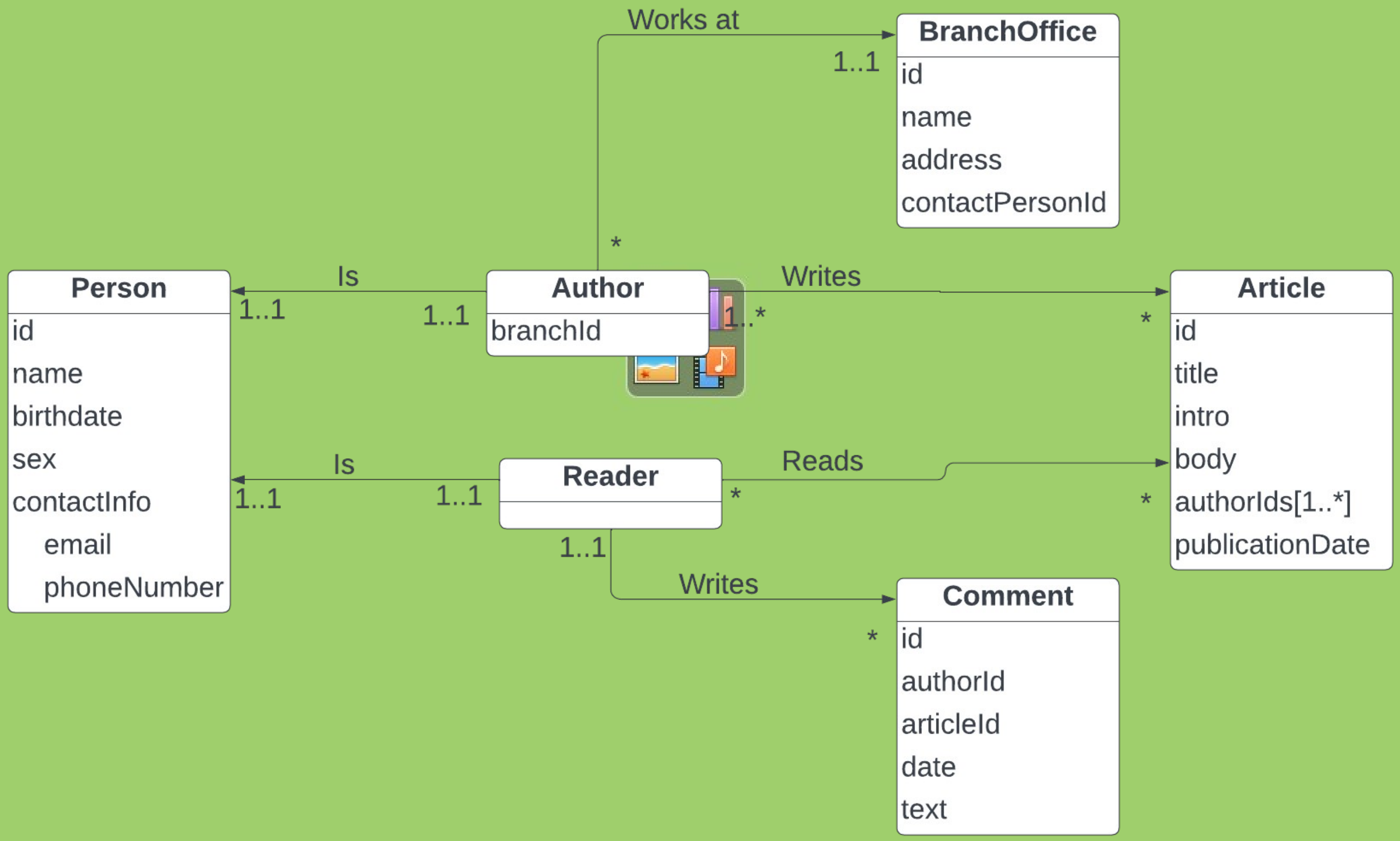
Chosen Domain

- Articles Database System
- Entities:
  - Article – database `articles`
  - Person – database `people`
    - Author, Reader – separate entities
    - Both are stored in people database, but Author has an additional attribute
  - Comment – database `comments`
  - BranchOffice – database `offices`

ER schema


name
address
id
email
contactPersonId
phoneNumber
(contactInfo)
WorksAt
Branch Office
sex
id
title
intro
birthDate
branchId
body
name
Person
Is
Author
Writes
Article
authorIds
id
publicationDate
Is
Reader
Reads
Writes
Comment
IsReactionTo
id
text
authorId
articleId
date

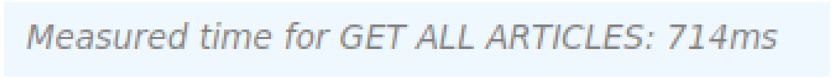
Logical (Database) Schema



Sample Data

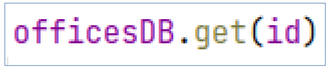
- I downloaded the list of people:
  - https://media.githubusercontent.com/media/datablist/sample-csv-files/main/files/people/people-1000.csv
  - I then assigned first 50 as authors and the rest as readers
- I generated articles and comments with the help of ChatGPT
  - I specified fields and presented some ideas and the AI generated some articles/comments in json format (which I had to correct, because there was a lot of mistakes)
- I wrote the json with branch offices myself

Queries + Description

- I created a simple application that uses my database
  - Article Database System
- Each page (except the main page) uses some query to get wanted data from the database
  - On the bottom of each page (except main and /all.html) you can see how long the query took
    - e.g.: 
      Measured time for GET ALL ARTICLES: 714ms
- Each query can be found in the corresponding HTML code as a JS script

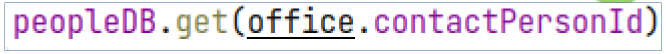
Query: get branch office details

- URL: `http://localhost:63342/pouchDbArticles/office.html?id=5`
- HTML file: `/office.html`
- Office ID is specified as a GET parameter
- Office is accessed using get()
  - Get office with specified id:

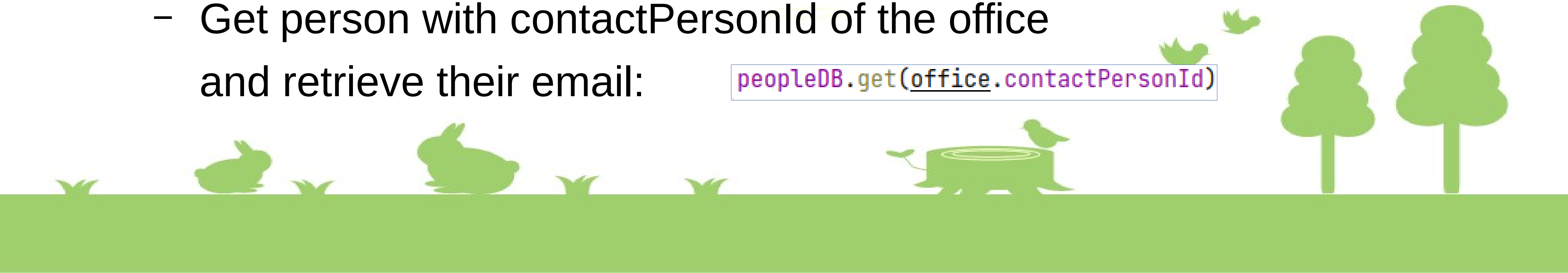


```
officesDB.get(id)
```

- Then contact email is needed - get()
  - Get person with contactPersonId of the office and retrieve their email:



```
peopleDB.get(office.contactPersonId)
```



Query: get an article and all related comments

- URL: `http://localhost:63342/pouchDbArticles/articleWithComments.html?id=1`
- HTML file: `/articleWithComments.html`
- Article ID is specified as a GET parameter
- The article is accessed using `get()`
  - Get article with specified id:

```
articleDB.get(id)
```

- Then the list of authors is needed – `find()`
  - Get all author names that have one of the given ids (author IDs):

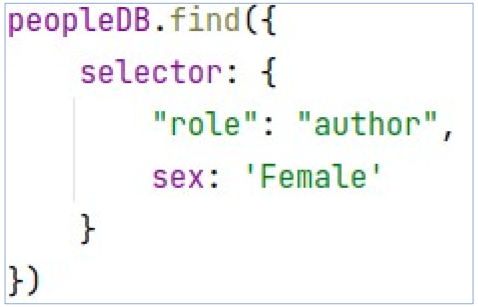
```
peopleDB.find({
    selector: {
        role: "author",
        _id: {$in: article.authorIds} }
})
```

- Finally, the comments are needed – `find()`
  - Get all comments that have articleId as the specified id (from URL):

```
commentsDB.find({ selector: { "articleId": id} })
```

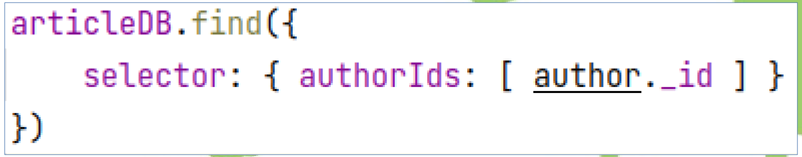
Query: Get all female authors + their articles

- URL: `http://localhost:63342/pouchDbArticles/femalesWithArticles.html`
- HTML file: `/femalesWithArticles.html`
- Get all female authors using find()
  - Get all authors that have sex = ‘Female’:



```
peopleDB.find({
    selector: {
        "role": "author",
        sex: 'Female'
    }
})
```

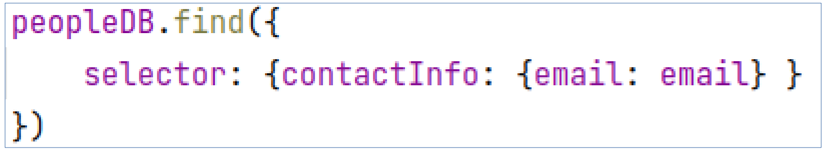
- Get all of their articles
  - Get all articles that have only one author and their ID matches current female author:



```
articleDB.find({
    selector: { authorIds: [ author._id ] }
})
```

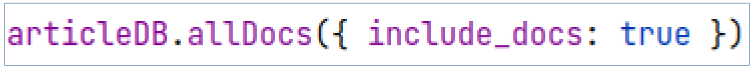
Other queries

- Get person based on their email:



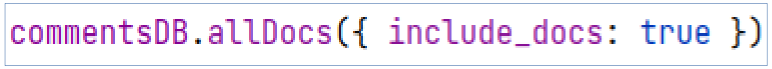
```
peopleDB.find({
    selector: {contactInfo: {email: email} }
})
```

- Get all article documents:



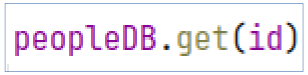
```
articleDB.allDocs({ include_docs: true })
```

- Get all comment documents:



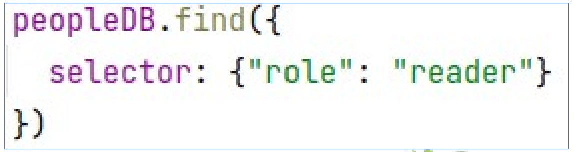
```
commentsDB.allDocs({ include_docs: true })
```

- Get person with specified id:

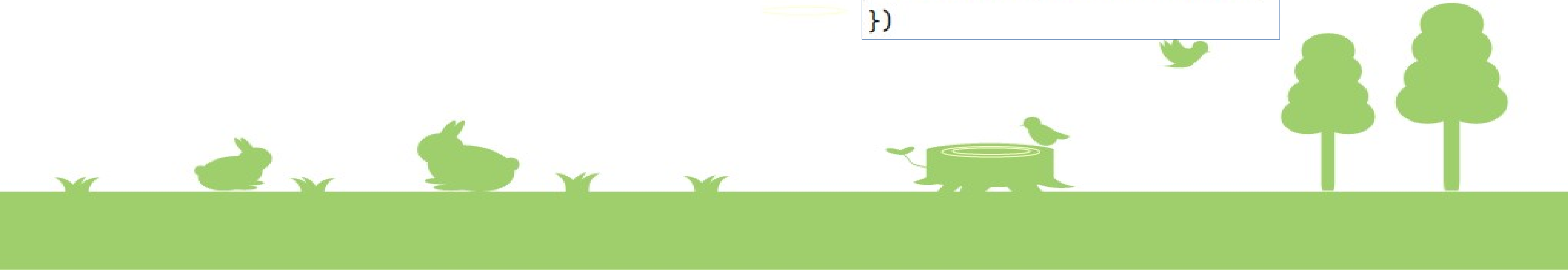


```
peopleDB.get(id)
```

- Get all readers:



```
peopleDB.find({
  selector: {"role": "reader"}
})
```

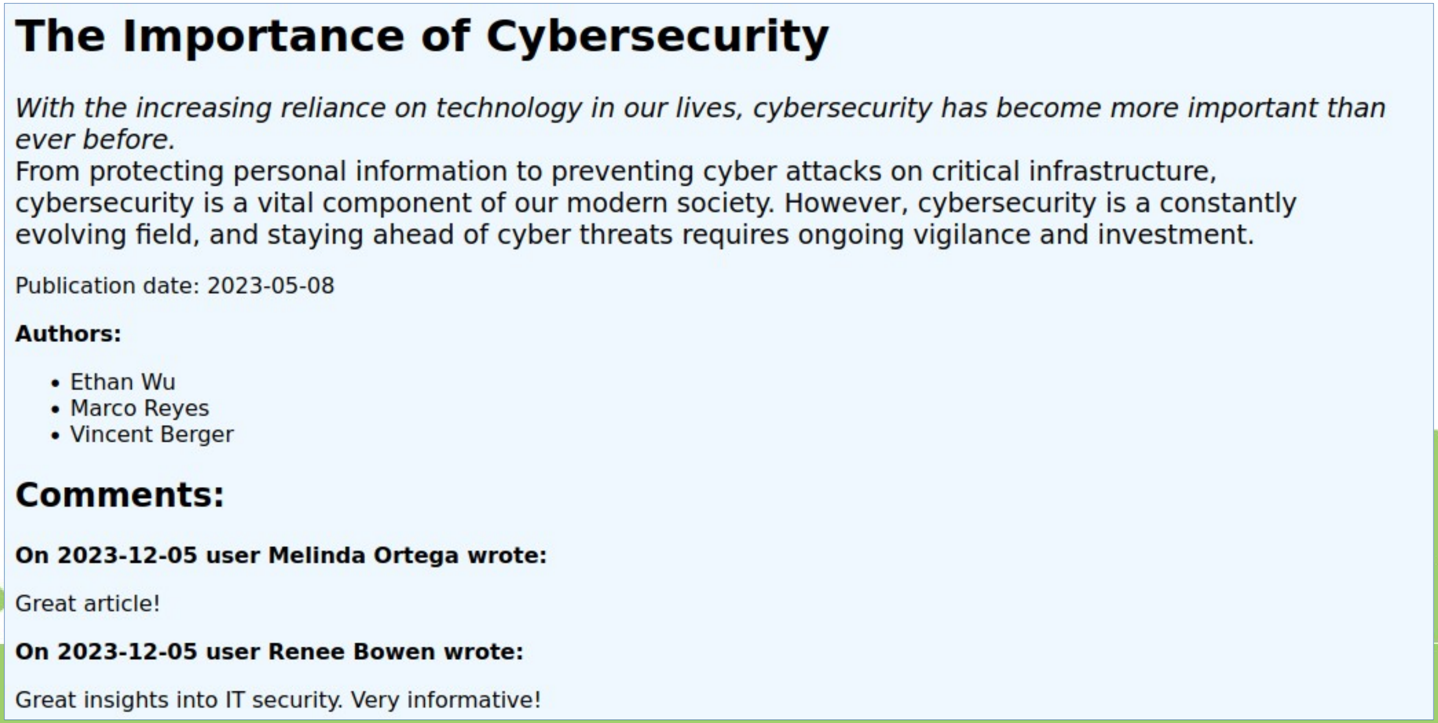


Demonstration of Work with the System

- **Articles Database System usage**
  - Start PouchDB server: `pouchdb-server start`
    - The response should look like: `pouchdb-server has started on http://127.0.0.1:5984/`
  - Open the main page of the app in a browser and move around with provided links

Example usage of the application

- Open main page (index.html) → List of all articles → The Importance of Cybersecurity → Go to article page with comments:



The Importance of Cybersecurity

With the increasing reliance on technology in our lives, cybersecurity has become more important than ever before.
From protecting personal information to preventing cyber attacks on critical infrastructure, cybersecurity is a vital component of our modern society. However, cybersecurity is a constantly evolving field, and staying ahead of cyber threats requires ongoing vigilance and investment.

Publication date: 2023-05-08

Authors:

- Ethan Wu
- Marco Reyes
- Vincent Berger

Comments:

On 2023-12-05 user Melinda Ortega wrote:

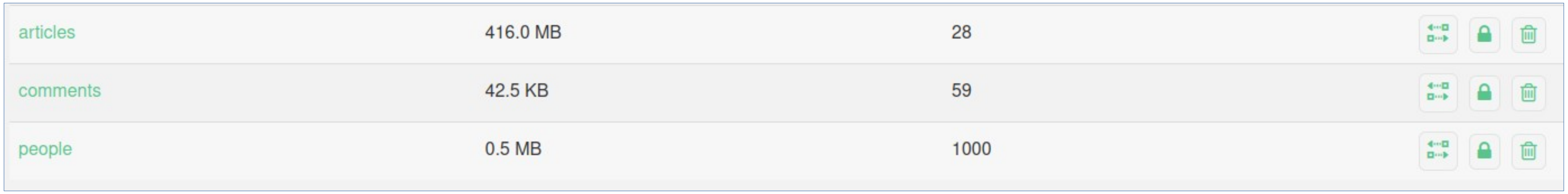
Great article!

On 2023-12-05 user Renee Bowen wrote:

Great insights into IT security. Very informative!

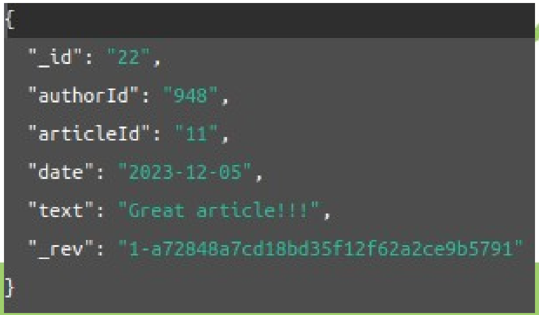
Demonstration of Work with the System

- **PouchDB usage**
  - Go to `http://localhost:5984/_utils/#/_all_dbs`
  - There is a list of all local databases – once you start the app and read from a database, you can see the database here and inspect it:



articles	416.0 MB	28
comments	42.5 KB	59
people	0.5 MB	1000

  - Detail of a comment with _id: “10”:



```
{
  "_id": "22",
  "authorId": "948",
  "articleId": "11",
  "date": "2023-12-05",
  "text": "Great article!!!",
  "_rev": "1-a72848a7cd18bd35f12f62a2ce9b5791"
}
```

Results of Experiments, Commentary

- PouchDB is really easy to use in JS code
- The documentation is really good (https://pouchdb.com/)
- The final application is a whole application, not just a set of random scripts (which is something that I expected at the beginning, so I am pleasantly surprised)
- It was quite complicated to start using pouchdb, especially pouchdb-find. There were a lot of different information about how the scripts should be imported

List of Additional Files

- video.mp4 – video presentation
- ER_diagram.png – ER diagram of the database
- ER_diagram.erdplus – script for the ER diagram
- Logical_Database_Schema.png – schema of the attributes and relationships
- README.md – contains instructions for installation and usage