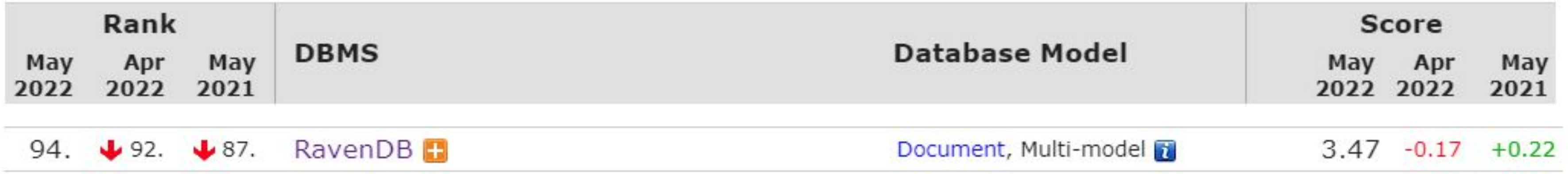
RavenDB

Vojtěch Lengál

Intro

- document-oriented DB
- supports ACID transactions
- multi-platform, written in C#

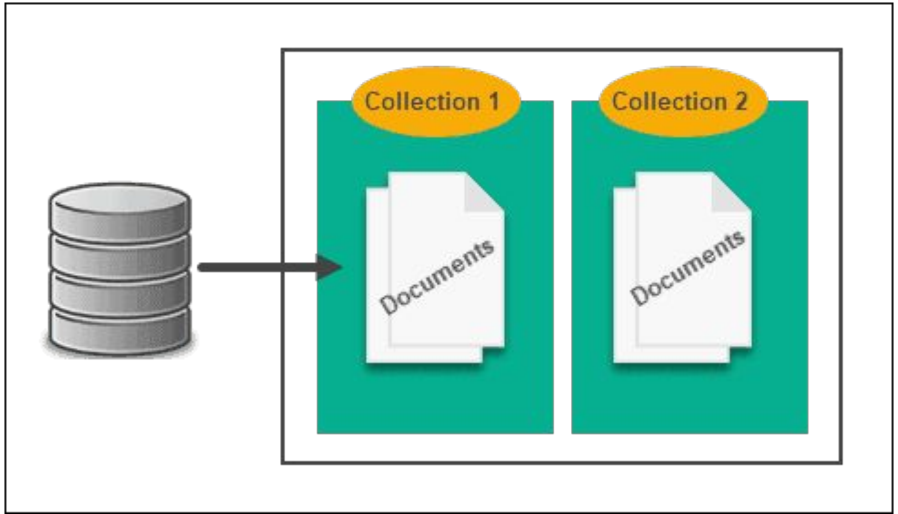
- https://db-engines.com/en/ranking



Rank			DBMS	Database Model	Score		
May 2022	Apr 2022	May 2021			May 2022	Apr 2022	May 2021
94.	↓ 92.	↓ 87.	RavenDB	Document, Multi-model	3.47	-0.17	+0.22

Document databases

- store data as structured documents (RavenDB: JSON)
- **Document**: record in a document database, typically stores information about one object
- **Collection**: group of documents, typically contains documents with similar contents.



ACID Transactions

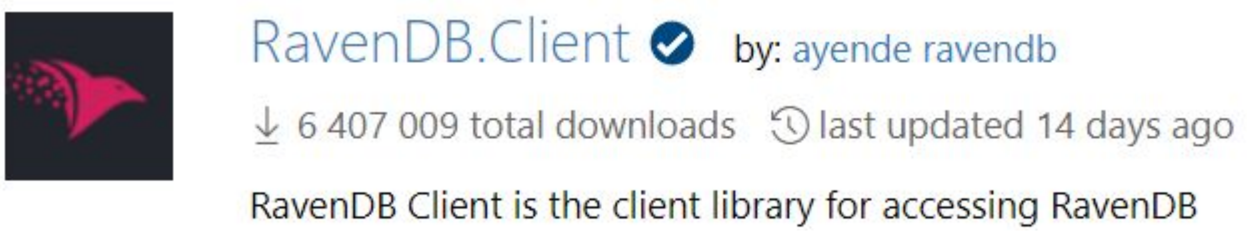
- **Atomicity**: transaction is a “single unit”, which either succeeds completely, or fails
- **Consistency**: DB is in a consistent state when a transaction starts and when it ends
- **Isolation**: intermediate state of a transaction is invisible to other transactions. (concurrent transactions appear to be serialized)
- **Durability**: After a transaction successfully completes, changes to data persist and are not undone, even in a case of system failure.

- supported by relational DBs

Client API

- multiple languages supported, most popular: C# (LINQ supported)

Microsoft .NET
C#
Java
Node.js
Python
Ruby
Go

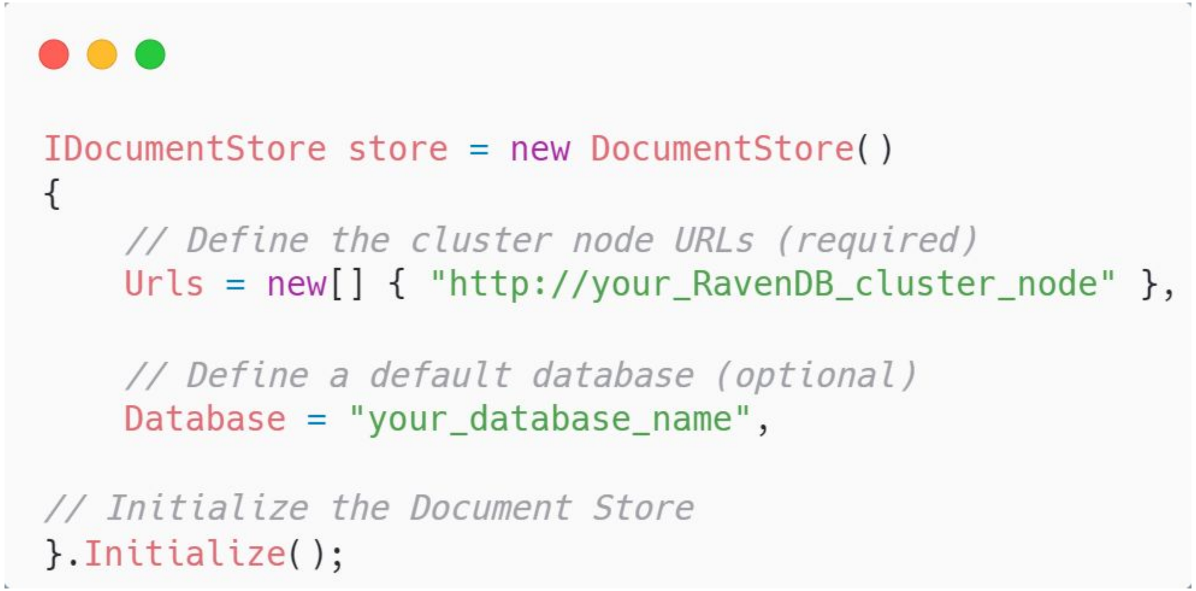

RavenDB.Client by: ayende ravendb

6 407 009 total downloads last updated 14 days ago

RavenDB Client is the client library for accessing RavenDB

Client API - terminology

- **Document store**: Client API object which establishes and manages the communication between client application and DB
- **Session**: represents a single transaction on a DB

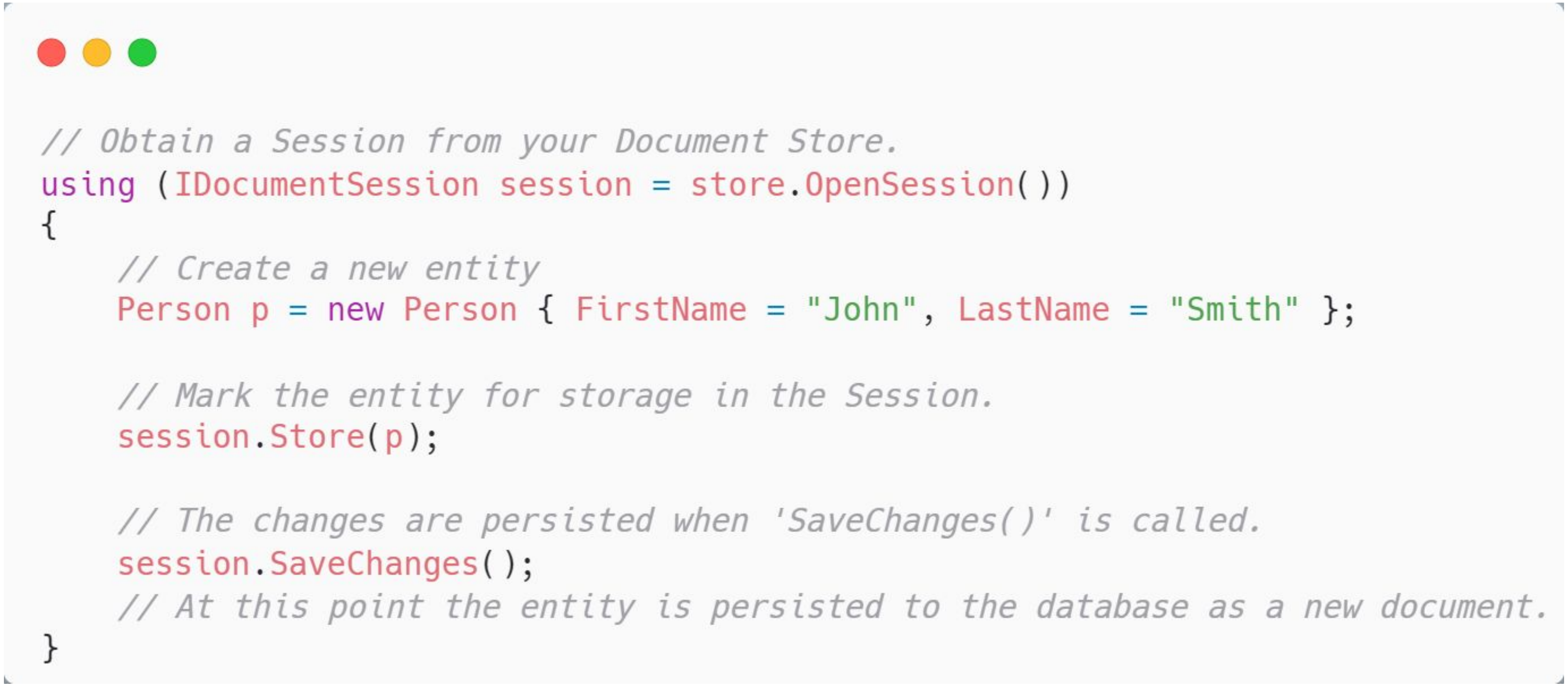


```
IDocumentStore store = new DocumentStore()
{
    // Define the cluster node URLs (required)
    Urls = new[] { "http://your_RavenDB_cluster_node" },

    // Define a default database (optional)
    Database = "your_database_name",

// Initialize the Document Store
}.Initialize();
```

Client API (C#)

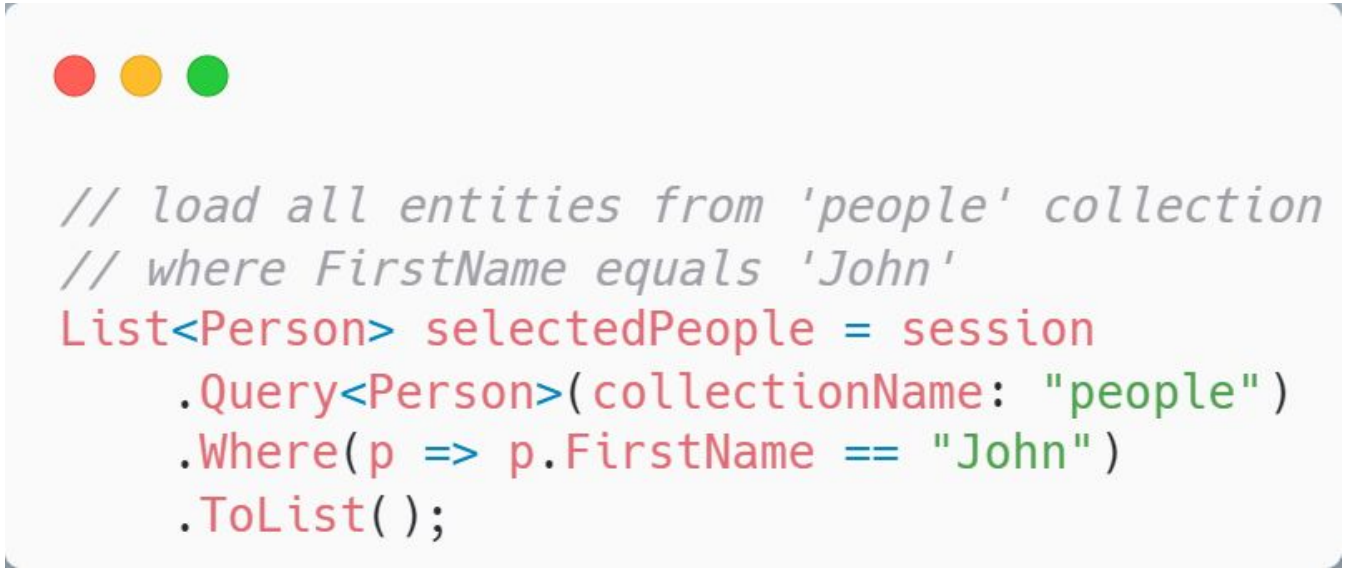


```
// Obtain a Session from your Document Store.
using (IDocumentSession session = store.OpenSession())
{
    // Create a new entity
    Person p = new Person { FirstName = "John", LastName = "Smith" };

    // Mark the entity for storage in the Session.
    session.Store(p);

    // The changes are persisted when 'SaveChanges()' is called.
    session.SaveChanges();
    // At this point the entity is persisted to the database as a new document.
}
```

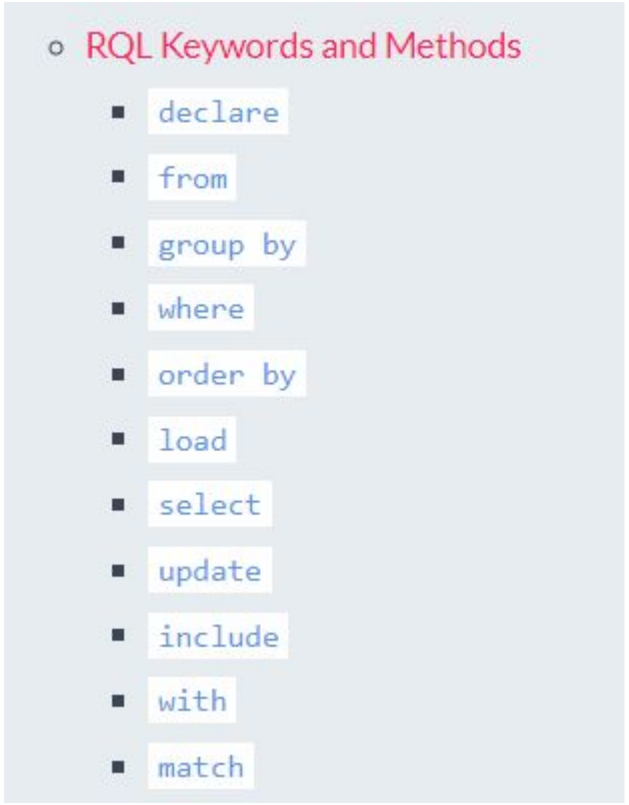
Client API (C# - LINQ)



```
// load all entities from 'people' collection
// where FirstName equals 'John'
List<Person> selectedPeople = session
    .Query<Person>(collectionName: "people")
    .Where(p => p.FirstName == "John")
    .ToList();
```

RQL

- SQL-like language for RavenDB
- limited functionality
  (mostly for querying data)

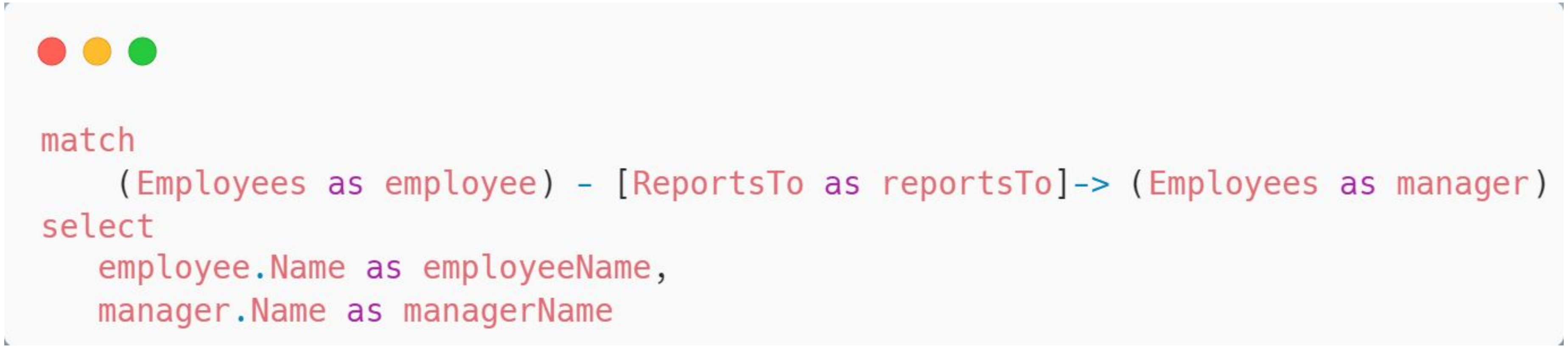


- RQL Keywords and Methods
  - `declare`
  - `from`
  - `group by`
  - `where`
  - `order by`
  - `load`
  - `select`
  - `update`
  - `include`
  - `with`
  - `match`

RQL - demo

Multi-model architecture

- support for Graph queries (experimental)
- nodes: documents & collections



```
match
    (Employees as employee) - [ReportsTo as reportsTo]-> (Employees as manager)
select
   employee.Name as employeeName,
   manager.Name as managerName
```

Other features

- supports querying via REST API
  `GET <server_URL>/databases/<database_name>/docs`

- Cloud service (Azure, AWS, …)