Serialisation, single-node databases, Structured Query Language (SQL) and Extract-Transform-Load/Extract-Load-Transform

Adam Boult (www.bou.lt)

April 30, 2025

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Using grep and awk with Comma-Separated Variables (CSVs)

Chapter 1

Comma Separated Values (CSV)

1.1 Introduction

1.1.1 Introduction

Part II

Databases

Chapter 2

Databases

2.1 Database operations

2.1.1 Joins

Have index common to two tables

Cross join

Keep all columns. can name [table name].[column name] to preserve any differences. including for index (could be missing for some)

Not really matched if original n length, and other m, then new is n*m length. so all combinations of matches are kept

Inner join

Drops those where no match. means data dropped in index missing

Any predicate, equi join, equality predicate

Outer join

Keep data if none matches. left outer join for one table, right outer join for other, or full outer join

2.1.2 Insert

2.1.3 Select

2.1.4 Update

2.1.5 Delete

2.1.6 Call

2.1.7 Create, read, update and delete (CRUD)

Chapter 3

Structured Query Language (SQL)

3.1 Working with databases

3.1.1 Working with databases

¿ATTACH DATABASE "/home/adam/Downloads/chinook.db" AS chinook;

SQLite3 show databases ¿.database

3.2 Working with tables within a database

3.2.1 Working with tables within a database

Use a database

Show tables within a database

SELECT * FROM Y

NULL as value

3.3 Working with a specific table

3.3.1 Working with a specific table

¿SELECT * FROM Table;

Doing operations

¿SELECT DISTINCT * FROM Table;

¿SELECT COUNT(DISTINCT *) FROM Table;

Filters:

¿SELECT * FROM Table WHERE Name=”Adam”;

More on filters:

+ LIKE for regex? + BETWEEN for ranges? + IN for vector + Can use multiple using AND, OR, NOT

Chapter 4

SQLite

4.1 Introduction

4.1.1 Introduction

SQL injection.

```
$sqlite3 test.db; .exit
```

Part III

Other serialisation types

Chapter 5

Extensible Markup Language (XML)

5.1 Introduction

5.1.1 Introduction

billion laughs attack. prevent with limited parsers

Chapter 6

JavaScript Object Notation (JSON)

6.1 Introduction

6.1.1 Introduction

Chapter 7

Yet Another Markup Language (YAML)

7.1 Introduction

7.1.1 Introduction

Chapter 8

Tom's Obvious Minimal Language (TOML)

8.1 Introduction

8.1.1 Introduction

Chapter 9

MongoDB and NoSQL

Part IV

NoSQL

Part V

Improving knowledge

Chapter 10

Knowledge bases

10.1 Storing knowledge

10.1.1 Resource Description Framework (RDF)

Introduction

How can we store information like "Joe Blogs was born in the city London"?

Information is described as an RDF triple:

- Subject
- Predicate
- Object

Examples

"Joe Blogs was born in the city London" can be written as:

(Joe Blogs, BornCity, London)

Confidence

We can associate a confidence with each triple.

10.1.2 Knowledge bases as graphs

Introduction

We can consider each fact to be a mini graph.

For "Joe Blogs was born in the city London" we have:

Joe Blogs $\rightarrow^{wasborninthecity}$ London

10.2 Using knowledge

10.2.1 Inferring facts

Introduction

Now we add another fact:

London $\rightarrow^{isacityinthecountry}$ UK

We can use these to define a new predicate: "was born in the country" and generate the fact:

Joe Blogs $\rightarrow^{wasborninthecountry}$ UK

10.2.2 Relational learning

Introduction

Consider two facts:

Alice $\rightarrow^{IsA}$ Doctor

Bob $\rightarrow^{HasMother}$ Alice

We can consider another fact:

Bob $\rightarrow^{IsA}$ Doctor

Confidence of inferred facts

How confident should we be of this?

In practice the graph between Bob and Doctor will have many paths (qualifications)

10.2.3 Prior and posterior confidence

Introduction

If we have a new fact, and a prior, we can create a posterior condfidence on the fact.

10.3 Collecting knowledge

Chapter 11

Expert systems

11.1 Logic-based agents

11.1.1 Forward and backward chaining

Introduction

Forward and backward chaining

Has a knowledge base (domain specific content)

Has an inference mechanism (domain independent algorithms)

If we want to find out whether the knowledge base implies a statement, we can check using semantics or sytax.

Inference is similar to search. We start with the knowledge base. We can apply the inference rule to all statements which match the top line of modus ponens (or another rule).

The actions are inference rules. We add new sentences to the b

Agents can also interact with the world in order to add to the knowledge base

Convert each item in knowledge base to CNF for resolution rule.