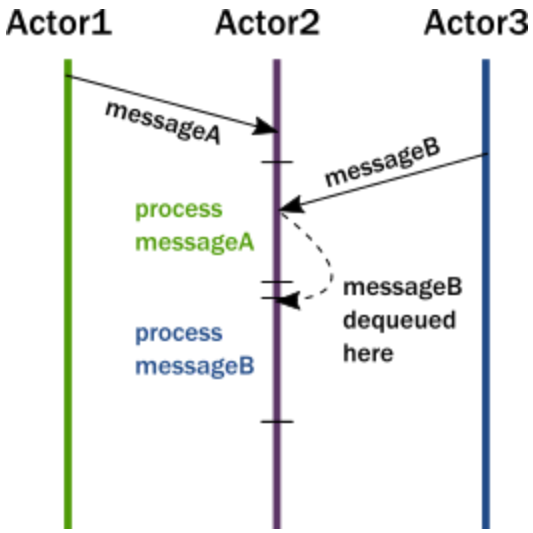
Ch4: Encoding & Evolution

Date: @March 10, 2023 8:58 PM

Topic: Message-Passing Dataflow

Recall	Notes
Message-passing	• asynchronous • message broker
advantages of message brokers compared to RPC	1. act as a buffer 2. redeliver messages, prevent messages from being lost 3. Flexibility for producers and consumers 4. one message to be sent to many recipients 5. decouple the sender from the recipient
Message brokers	queue or topic: provides only one-way dataflow. to be careful to preserve unknown fields
Actor model & distributed actor frameworks?	



📌 **SUMMARY:**
message brokers are used as follows: one process sends a message to a named *queue* or *topic*, and the broker ensures that the message is delivered to one or more *consumers* of or *subscribers* to that queue or topic.

The *actor model* is a programming model for concurrency in a single process.

Date: @March 9, 2023 9:56 PM

Topic: Dataflow through Services: REST and RPC (API)

Recall	Notes
	• Need to communicate over a network • Clients & Servers

How to evaluate compatibility? Who is the writer and who is the reader?	params + body ⇒ HTTP https://github.com/zalando/restful-api-guidelines/blob/main/chapters/compatibility.adoc https://easyitblog.info/posts/2022/backward-compatible-change-api-web-rest-http/

📌 **SUMMARY:** The API exposed by the server is known as a *service*.

Date: @March 9, 2023 9:10 PM

Topic: Dataflow through Databases

Recall	Notes
Database dataflow	• sending a message to your future self(process) • backward compatibility is clearly necessary. solution: do not delete the required column or do not set columns to be required. • forward compatibility is also often required for databases. solution: do not add required columns or need to fill in newly added required columns(migration) Issue: older reader, read → modified → write
update lost	

migrating(rewriting)

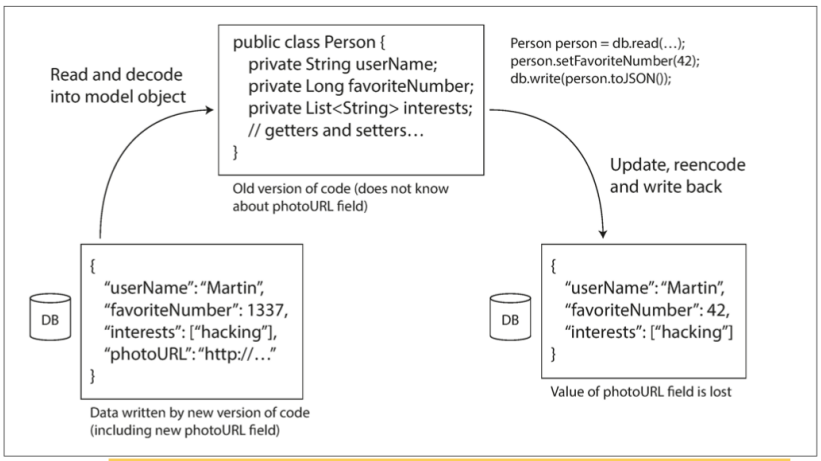


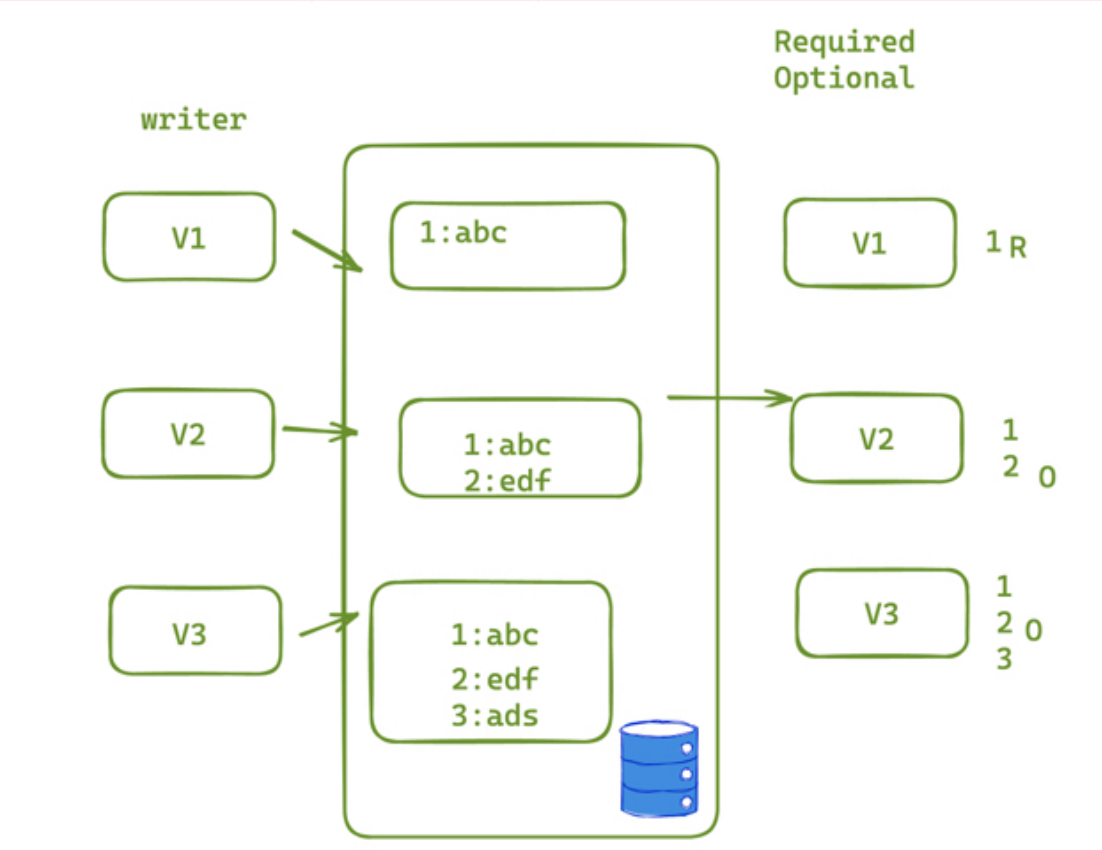
Figure 4-7. When an older version of the application updates data previously written by a newer version of the application, data may be lost if you're not careful.

Solution:

don't fully overwrite

- should avoid as much as possible as expensive

📌 **SUMMARY: Well-designed for compatibility**



Date: @March 7, 2023 8:35 PM

Topic: Modes of Dataflow

Recall

Three of the most common ways how data flows

Notes

1. Dataflow through database
2. Dataflow through services: REST & RPC
3. Message-Passing Dataflow

📌 **SUMMARY:** 1. Compatibility is a relationship between one process that encodes the data, and another process that decodes it.

Date: @March 11, 2023 4:29 PM

Topic: The Merits of Schemas

Recall	Notes
cons and pros of schema	• validation rules • compactable • readable • compatibility

📌 **SUMMARY:**
In summary, schema evolution allows the same kind of flexibility as schemaless/schema-on-read JSON databases provide, while also providing better guarantees about your data and better tooling.

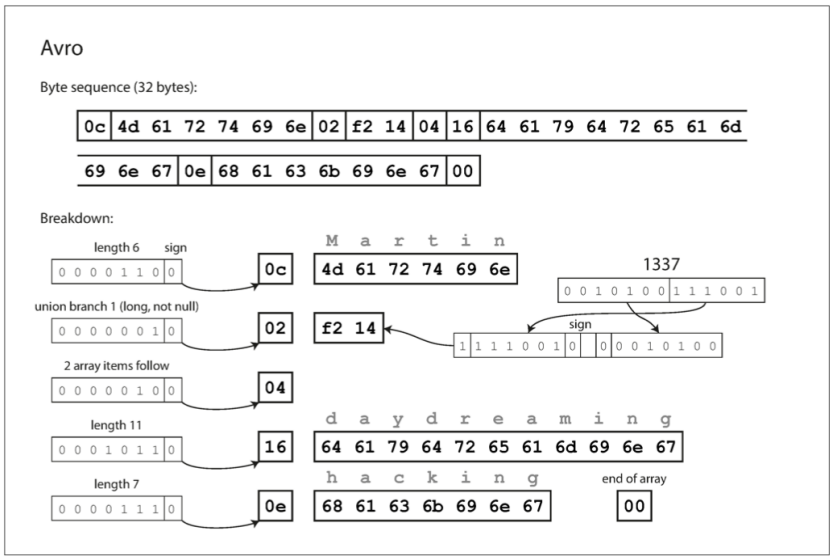
Date: @March 11, 2023 4:28 PM

Topic: Avro

Recall	Notes
Avro schema	

```
record Person {
  string userName;
  union { null, long } favoriteNumber = null;
  array<string> interests;
}
```

Avro encoding example



Reader schema and Writer schema

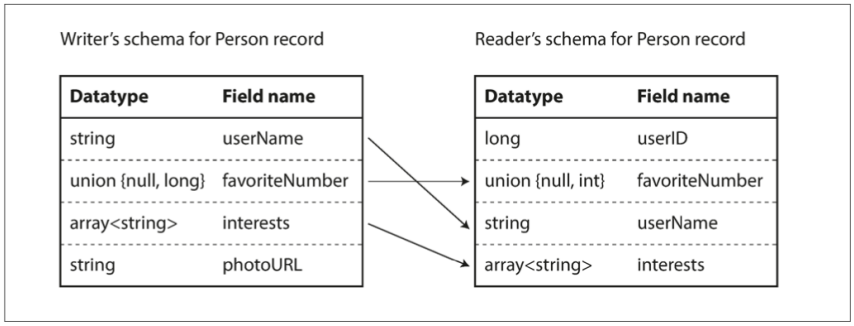


Figure 4-6. An Avro reader resolves differences between the writer's schema and the reader's schema.

How to determine writer's schema when decoding?

It depends on the scenarios in which Avro is being used.

- *Large file with lots of records: schema on top*
- *Database with individually written records: the schema version number is included in the record*
- *Sending records over a network connection: handshake & nigotiate*

📌 **SUMMARY:**
Like Protocol Buffers and Thrift, Avro needs a schema. The most difference compared to them is no tag number in the schema anymore. With Avro, forward compatibility means that you can have a new version of the schema as writer and an old version of the schema as reader. To maintain compatibility, you may only add or remove a field that has a default value.

Date: @March 6, 2023 9:07 PM

Topic: Thrift & Protocol Buffers

Recall	Notes

Schema of Thrift	`struct Person {`
	`  1: required string userName,`
	`  2: optional i64 favoriteNumber,`
	`  3: optional list<string> interests`
	`}`

Schema of Protocol Buffers	`message Person {` `  required string user_name       = 1;`

```
  optional int64  favorite_number = 2;
  repeated string interests       = 3;
}
```

Thrift has two encoding formats, ***BinaryProtocol***

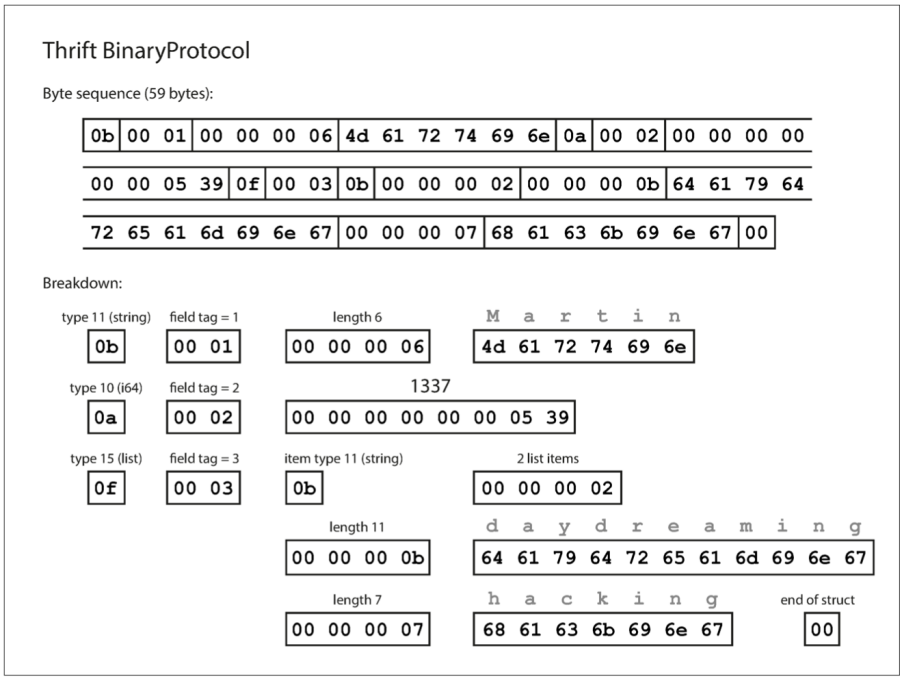


Figure 4-2. Example record encoded using Thrift's BinaryProtocol.

And ***CompactProtocol***

The major difference compared to MessagePack is there is not field names. Instead, the encoded data contains field tags.

Protocol Buffers

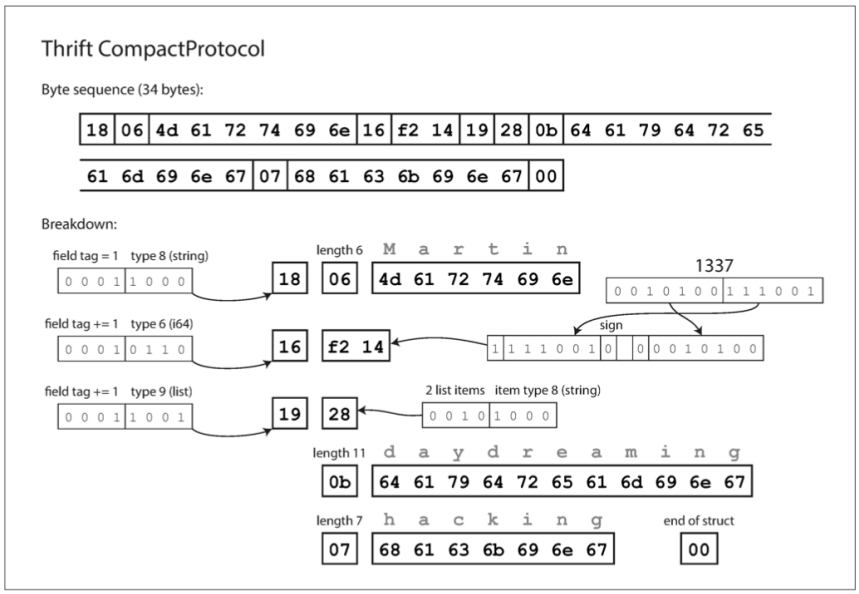


Figure 4-3. Example record encoded using Thrift's CompactProtocol.

How do we guarantee compatibility during schema evolution?

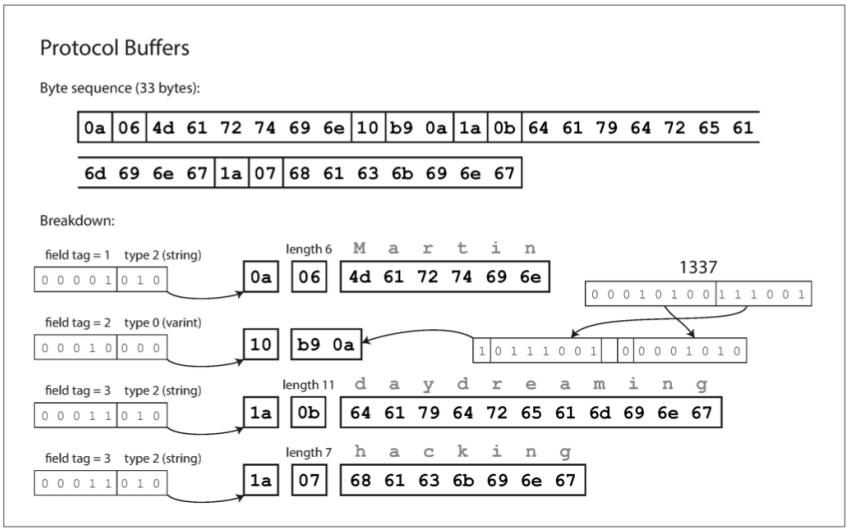


Figure 4-4. Example record encoded using Protocol Buffers.

1. Can change field name but never change field tag in the schema.
2. Do imcompatible change against fields by adding new ones and deprecating old ones.

📌 **SUMMARY:**
Both Thrift and Protocol Buffers require a schema for encoding data. A code generation tool takes the schemas to produce SDK for encoding and decoding data in various programming languages.

As schemas inevitably need to change over time, the most thing we should take care is compatibity during shcema evolution.

Date: @March 11, 2023 4:29 PM

Topic: JSON, XML, and Binary Variants

Recall	Notes
Subtle problems of JSON, XML, and CSV	• ambiguity around numbers. floating numbers not well-supported. • don't support binary strings. Get around this limitation by encoding the binary data as text using Base64. • Schemaless (schema-on-read).
Binary encoding	JSON is less verbose than XML, but both still use a lot of space compared to binary formats. This observation led to the development of a profusion of binary encodings for JSON

(MessagePack, BSON, BJSON, UBJSON, BISON, and Smile, to name a few).

some of these formats extend the set of datatypes.

e.g. Encoding json with MessagePack

```
{
    "userName": "Martin"
    "favoriteNumber": 13
    "interests": ["daydr
}
```

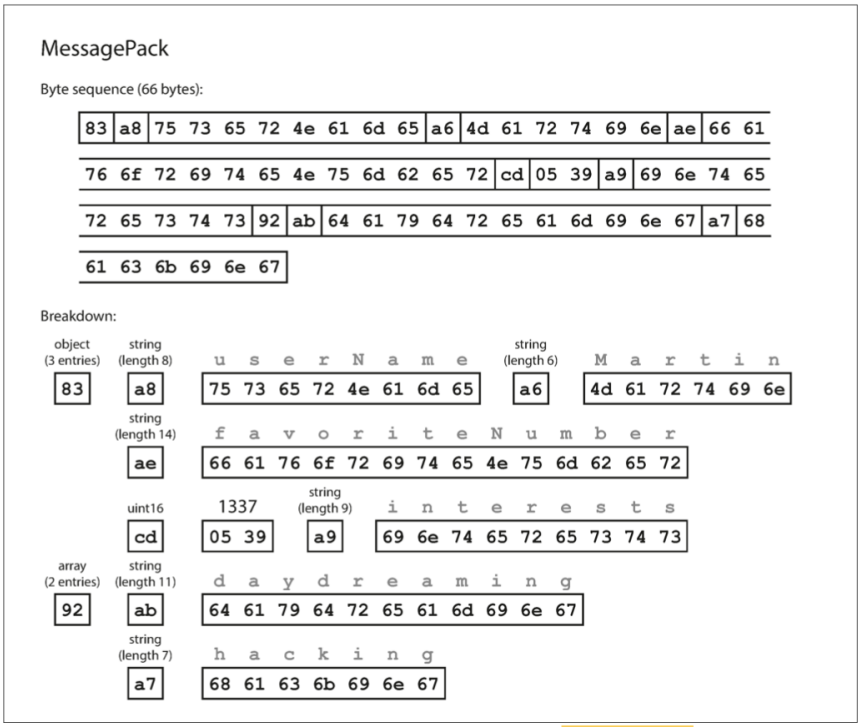


Figure 4-1. Example record (Example 4-1) encoded using MessagePack.

📌 **SUMMARY:** 1. Moving to standardized encodings that can be written and read by many programming languages, JSON and XML are the obvious contenders. They are widely known, widely supported, and almost as widely disliked.

The binary encoding with MeesagePack format is 66 bytes long, which is only a little less than the 81 bytes taken by the textual JSON encoding.

Date: @March 11, 2023 4:27 PM

Topic: Language-Specific Formats

Recall	Notes
Why need common encoding & decoding libraries? The problems of language-specific format?	• language specific • security issue. Attacker possibliy can remotely execute arbitrary code • Versioning management is afterthought • Efficiency is afterthought

📌 **SUMMARY:**
Different programming languages have their built-in support for encoding in-memory objects into byte sequences. Java has Serializable, Ruby has Marshal and so on. But they are not exchangable. So, we need to universal approaches for serialization.

Date: @March 3, 2023 9:15 PM

Topic: Encoding & Evolution

Recall	Notes
What's rolling upgrade and why?	Deploying a new version to a few nodes at a time, checking whether the new version is running smoothly, and gradually working your way through all the nodes.

With rolling upgrades, new and old versions of the code, and old and new data formats may potentially all coexist in the system at the same time. For a system to run smoothly, compatibility needs to be in both directions.

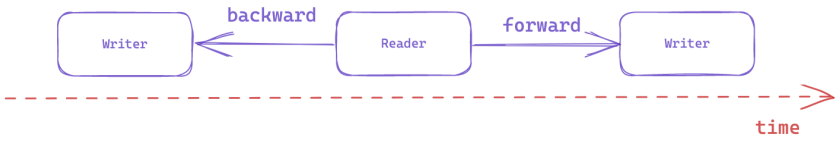
Zero downtime upgrading.

Backward compatibility

Newer code can read data that was written by older code.

Forward compatibility

Older code can read data that was written by newer code.



📌 **SUMMARY: Applications inevitably change over time. We should always take evolvability into account. That means we aim to build systems that make it easy to adapt to change.**