Building your own DuckDB Extension

Pedro Holanda
pedro@duckdblabs.com
pedroholanda.com

DuckDB Labs

Outline

- **Why Have Extensions?**
- **How Extensions Work?**
- **What Can You Extend?**
- **Examples.**
- **Building our extension.**

Why Have Extensions?

Motivation

- **Start from a light version of the DBMS**
  - **CSV Files** 🏃🏃🏃🏃
  - **Postgres Files** 🏃
- **External Dependencies;**
- **Binary Size;**
- **Compilation Time;**
- **Binary Distribution;**
- **Private/Closed-Source Additions.**

How Extensions Work

How it works - In a nutshell

- **Install Extensions (i.e., Downloads it from S3)**

```
INSTALL('${extension_name_or_local_path}')
```

- **Load Extensions (i.e., DLLOpens)**

```
LOAD('${extension_name_or_local_path}')
```

- **Official S3 Bucket**
  - **Just provide Extension Name**
  - **DuckDB CI automatically builds and distributes extensions for each platform.**
  - **Maintained and approved by the DuckDB Team.**

How it works - In a nutshell - Signed vs Unsigned

- **What about me? - Unofficial S3 Bucket**
  - **Binaries must be distributed by producer.**
  - **Specify different end-point (S3 Bucket) for Install**

```
SET custom_extension_repository='some.domain.com/some/path/vX.Y.Z';
```

  - **No checks on source-code - From DuckDB Team.**
  - **However:**
    - **Boiler-Plate CI does all the distribution for you!**

What can you extend?

What can you extend

- **Functions**
  - **Table Scanners**
  - **Copy Functions**
  - **Aggregation Functions**
  - **Scalar Functions**
  - **Table Producing Functions**
- **File System**
- **And Much More***
  - **Optimizer Rules**
  - **Catalog**
  - **Parser**

Examples

DuckDB Extensions

- **Parquet Scanner/Writer**
- **HTTPFS**
- **SQLite/Postgres Scanners**
- **Substrait**

Takeaways & Upcoming Features.

Takeaways and Upcoming Features

- **Easy to Install and Load.**
- **Can help users cover many use cases; While:**
  - **Keeping DuckDB main codebase small;**
  - **Clean;**
  - **And Without External Dependencies.**
- **Boiler-Plate Repo:**
  - **Easy to have users started on their own Repo**
  - **Does automatic platform Building/Distribution**

Upcoming Features

- **The future is extensible/pluggable**
- **Storage**
- **Physical Operators**
  - **Joins**
  - **Sorting**
- **Data Representations**
  - **Index Structures**
- **Buffer Manager**
- **And Much More***

Super-duper main takeaway

- **You can develop your own Extensions!**
- **Boilerplate Repository:**
  - **Smooth Development**
  - **Easy Binary Distribution**
- **Many examples already available:**
  - **https://github.com/duckdb/duckdb/tree/master/extension**
  - **https://github.com/duckdblabs/substrait**
  - **https://github.com/duckdblabs/postgres_scanner**

Build our own extension

Hands-on

- **Can I write DuckDB extensions?**
  - **Yes!**
  - **Boiler-plate Extension TEMPLATE**
  - **https://github.com/duckdb/extension-template**
- **Is it hard to get started?**
- **No, I'll show you!**

Hands-on

- **Goal**
  - **Build our First DuckDB Extension**
  - **Scalar Function**
    - **Stateless function that returns 1-1 with input**
    - **Returns one column.**
  - **Anonymize Phone Numbers (Scalar)**
- **Extend Generation Function to generate:**
  - **Phone Numbers**
  - **Emails**

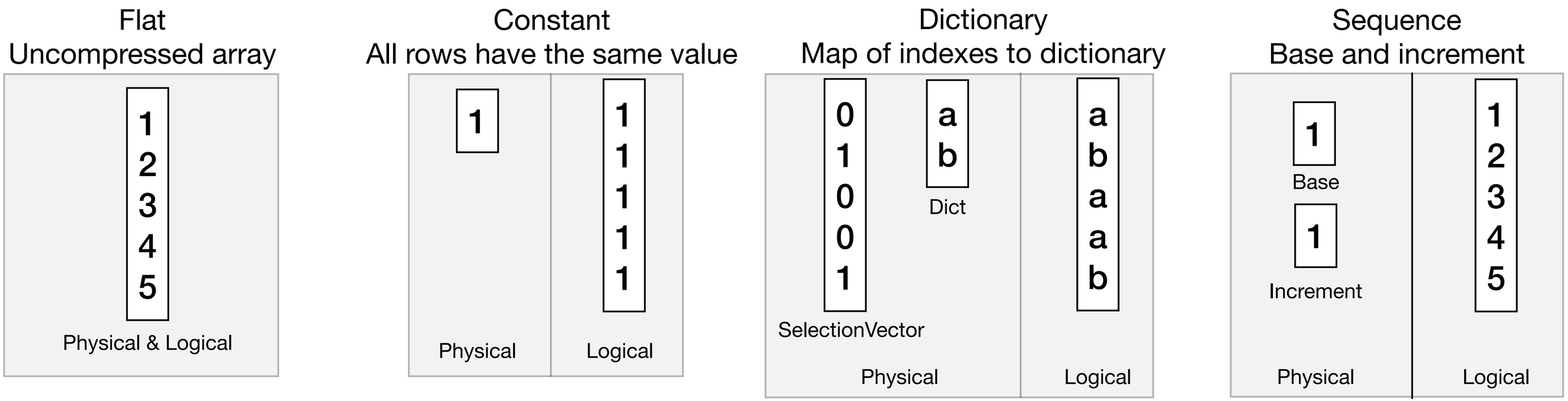
DuckDB - Vectors

- Vectors hold data of a single type
- For scalar types vectors are **logically** arrays
- VectorType determines **physical representation**
  - Allows us to push **compressed data** into the engine!

Vector
Integer

1
2
3
4
5

DuckDB - Vectors

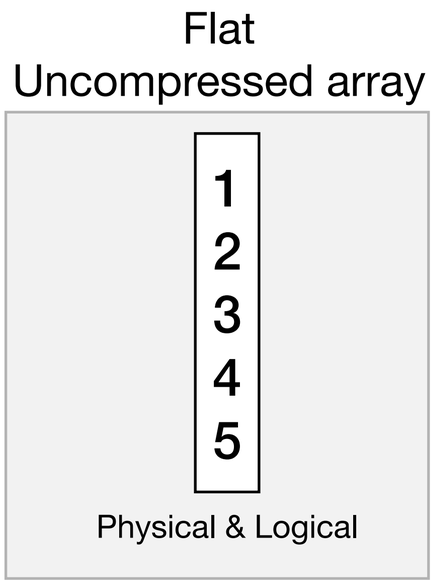


DuckDB - Vectors

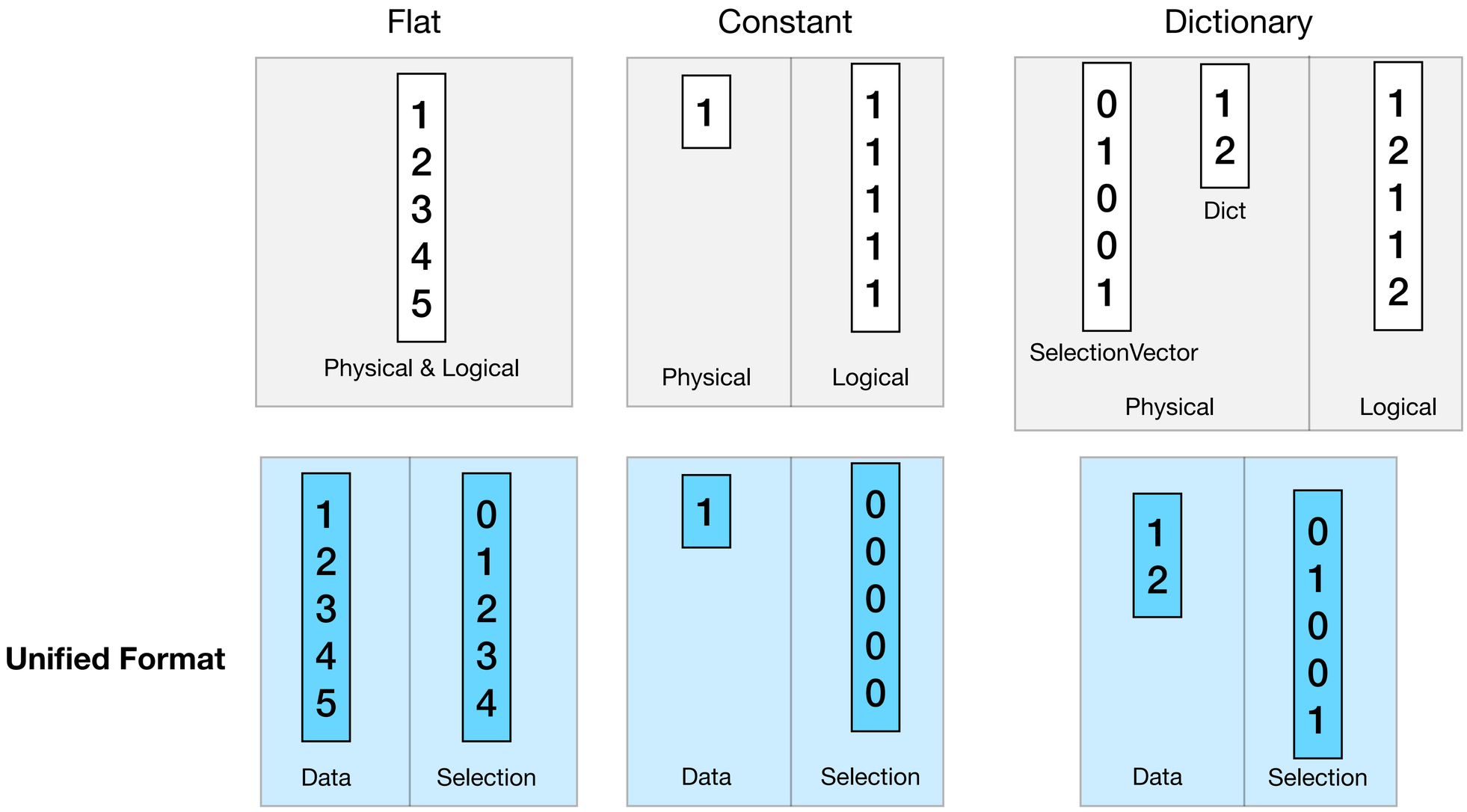
- Vectors can be processed as-is (compressed execution)
  - **Problem:** combinatorial explosion!
  - Giant code footprint

- **Flatten** - Convert vector into Flat Vector (i.e. decompress)
  - Need to move/copy data around!

- **ToUnified** - Convert vector to **unified format**

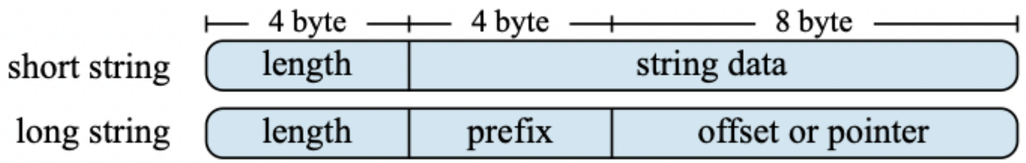


DuckDB - Vectors



No data copy or data movement required!

DuckDB - Vectors



- Strings are stored in the same format as Umbra
  - 16 bytes
  - Short strings are inlined (≤ 12 bytes)
  - Long strings have a prefix + pointer
- **Fast early-out in comparison**


Umbra: A Disk-Based System with In-Memory Performance

[1] Umbra: A Disk-Based System with In-Memory Performance