Overview

- 2 PM  Welcome to DuckCon!
- 2:05  State of the Duck v4
- 2:35  Hugging a Duck (Polina)
- 3:00  Building a Data Lake using DuckDB (Subash)
- 3:25  Break
- 3:40  The Duck(DB) Feather in Your Parquet Cap (Niger)
- 4:05  Lightning Talks (Rik, Arjen, Remco, Kshitij, Mike, Ian)
- 4:40  Drinks and Snacks (sponsored by Rill Data)
- ~6:40  End
State of the Duck
• **Where we are**
  • Adoption
• **Where we’re going**
  • Support Policy
  • Sneak Peek 0.10.0
• Q&A
DuckDB Adoption - January 2024

> 2 M Downloads / Month
> 5 M Extension Installs / Month
> 500K Unique Web Visitors / Month
> 14 K GitHub Stars
> 11 K Twitter Followers
> 6 K LinkedIn Followers
> 4 K Discord Members

^ DB-Engines Ranking
  Overall: 82, Relational: 45

DuckDB Adoption

- Python: ~1.3M Installations / Month
- Java: ? Data issue in Maven ;/
- Node.js ~ 120 K
- + Direct Downloads, WASM usage, ...
- > 10 K GitHub Stars
- > 8 K Twitter Followers
- ~ 2 K Discord Members
- ~ 1M Extension Installs / Month
- > 100K Unique Web Visitors / Month

June 2023
DuckDB Adoption - January 2024

Interest over time

Google Trends

Awesome DuckDB

A curated list of awesome DuckDB libraries, tools and resources.

DuckDB is an in-process SQL OLAP database management system.

Contents

- Resources
DuckDB is a fast in-process analytical database

DuckDB supports a feature-rich SQL dialect complemented with deep integrations into client APIs.
Where We Are
Where We Are
Topics

• Where we are
  • Adoption
• Where we’re going
  • Support Policy
  • Sneak Peek 0.10.0
• Q&A
Support Policy

- Many users, many questions
- DuckDB [Labs] not VC-Funded, no huge DevRel team
- Need to prioritise support effort
  - No guarantees for response time
  - No corporate freeloaders
  - No obscure APIs (e.g. Julia, Swift)
  - No internals
  - No weird platforms

Commercial Support Contracts

- **DuckDB Labs now offers commercial support contracts**
  - Privileged access to DuckDB core team
  - Higher priority issues
  - Guaranteed response time
  - Private issue tracker
  - Private datasets

https://duckdblabs.com/support/
Where we’re going

• Release 0.10.0 “Fusca” coming soon (early Feb)
• Release 1.0.0 planned for first half of 2024
  • No new features between 0.10.0 and 1.0.0
    • “Snow Leopard Duck”
  • Series of bugfix-only releases 0.10.1, 0.10.2, …
• Focus on stability and robustness
• Backwards-compatibility of database file format
v0.10.0 - Backwards Compatibility!

• 1.0: **Storage format stabilization**

• **Goals**
  • Stability
  • Able to *extend and improve* the format

• Backwards compatibility *guaranteed*
  • *Always* able to read older DuckDB files

• Forwards compatibility *best effort*
  • Able to read newer DuckDB files if no new features are used
v0.10.0 - Backwards Compatibility!

• **v0.10.0** -> backwards compatible with v0.9.2
  • Can read and write to files created with v0.9.2

```bash
$ > duckdb v092.db
v0.9.2
D CREATE TABLE lineitem AS FROM lineitem.csv;
$ > build/debug/duckdb v092.db
v0.10.0
D SELECT l_orderkey, l_partkey, l_comment FROM lineitem LIMIT 1;
```

<table>
<thead>
<tr>
<th>l_orderkey</th>
<th>l_partkey</th>
<th>l_comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>int64</td>
<td>int64</td>
<td>varchar</td>
</tr>
<tr>
<td>1</td>
<td>155190</td>
<td>to beans x-ray carefull</td>
</tr>
</tbody>
</table>

• Beta storage stabilization
  • **v1.0 onwards**: always support backwards compatibility
v0.10.0 - Partial Forwards Compatibility

- **v0.10.0** *partially* forwards compatible with v0.9.2
  - v0.9.2 can read **some** files created by v0.10.0

```bash
$ > build/debug/duckdb v010.db
v0.10.0
CREATE TABLE lineitem AS FROM lineitem.csv;

$ > duckdb v010.db
v0.9.2
SELECT l_orderkey, l_partkey, l_comment FROM lineitem LIMIT 1;
```

<table>
<thead>
<tr>
<th>l_orderkey</th>
<th>l_partkey</th>
<th>l_comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>155190</td>
<td>to beans x-ray carefull</td>
</tr>
</tbody>
</table>

- Files cannot be read if they contain:
  - New compression method (ALP)
  - New types (ARRAY/UHUGEINT)
Sneak Peek

February 2024
(Scheduled)

• ATTACH for Postgres/MySQL
  • Connect to a running database
  • Query as-if they are DuckDB tables

```
ATTACH 'dbname=postgresscanner' AS pgdb (TYPE postgres);
SELECT c_name, c_address FROM pgdb.customer LIMIT 3;
```

```
c_name    | c_address
-----------|------------
Customer#000000001 | j5JsirBM9PsCy0O1m
Customer#000000002 | 487LW1dovn6Q4dMVymKwwLE90Kf3QG
Customer#000000003 | fkRGN8nY4pKE
```

• Support many operations:
  • INSERT INTO, COPY
  • DELETE, UPDATE
  • Transactions, ALTER TABLE, …
v0.10.0 - CSV Reader Rework

• Major CSV reader overhaul
  • Large performance improvement!

• Many small improvements
  • Better error reporting, bug fixes
v0.10.0 - Fixed-Length Arrays

• New type: fixed-length arrays

```sql
CREATE TABLE vectors(v DOUBLE[3]);
INSERT INTO vectors VALUES ([1, 2, 3]);
```

• Similar to lists, but every entry **must** be the same size
  • Useful for operating on vectors
• New functions:
  • array_cross_product
  • array_dot_product
  • array_cosine_similarity
  • … and more!

https://duckdb.org/docs/sql/data_types/array
DuckDB supports out-of-core operators
- Aggregates, joins, sorting, ...

Previously operators did not communicate
- Problem when running complex queries!

v0.10 -> Temporary Memory Manager
- Communicate memory usage
- Improved memory utilization for complex queries
- Larger-than-memory execution for complex queries
v0.10.0 - Secret Manager

- DuckDB previously managed secrets (S3, …) through settings
- v0.10 introduces **Secrets** and the **Secret Manager**

```
CREATE SECRET (  
    TYPE S3,  
    KEY_ID 'mykey',  
    SECRET 'mysecret',  
    REGION 'myregion',
);
```

- Multi-secret management/scoping
- More secure
- Persistent
  - Secrets can be persisted to the DuckDB folder (~/.duckdb/stored_secrets)
  - Automatically available when using DuckDB
ALP floating point compression was added in v0.10
- Contributed by Azim, Leonardo and Peter from the CWI

- Replaces Chimp/Patas compression
- ALP is faster at both compression and decompression
- ALP achieves a higher compression ratio

<table>
<thead>
<tr>
<th>Compression</th>
<th>Load</th>
<th>Query</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALP</td>
<td>0.434s</td>
<td>0.02s</td>
<td>184MB</td>
</tr>
<tr>
<td>Patas</td>
<td>0.603s</td>
<td>0.08s</td>
<td>275MB</td>
</tr>
<tr>
<td>Uncompressed</td>
<td>0.316s</td>
<td>0.012s</td>
<td>489MB</td>
</tr>
</tbody>
</table>

60 million 8-byte floats
• **Storage improvements**
  • During checkpoint, deleted rows are now automatically vacuumed
    • Free up space in on-disk file
    • Speed up queries on tables (defragmentation)

• Checkpoint is now multi-threaded
  • Parallel vacuuming
v0.10.0 - CLI Improvements

- Improved multiline editing
- Multiline history
- Multiline becomes the default mode

```sql
SELECT
  _returnflag,
  _linestatus,
  sum(_quantity) AS sum_qty,
  sum(_extendedprice) AS sum_base_price,
  sum(_extendedprice * (1 - _discount)) AS sum_disc_price,
  sum(_extendedprice * (1 - _discount) * (1 + _tax)) AS sum_charge,
  avg(_quantity) AS avg_qty,
  avg(_extendedprice) AS avg_price,
  avg(_discount) AS avg_disc,
  count(*) AS count_order
FROM
  lineitem
WHERE
  _shipdate <= CAST('1998-09-02' AS date)
GROUP BY
  _returnflag,
  _linestatus
ORDER BY
  _returnflag,
  _linestatus;
```

- Faster text rendering
Next Year(s) in DuckDB

- Extension Ecosystem
- Optimiser Improvements
  - Partition/Sorting Awareness
  - Cardinality Estimation
- Lakehouse Data Formats
Topics

- Where we are
  - Adoption
- Where we’re going
  - Support Policy
  - Sneak Peek 0.10.0
- Q&A
Drinks and snacks sponsored by www.rilldata.com