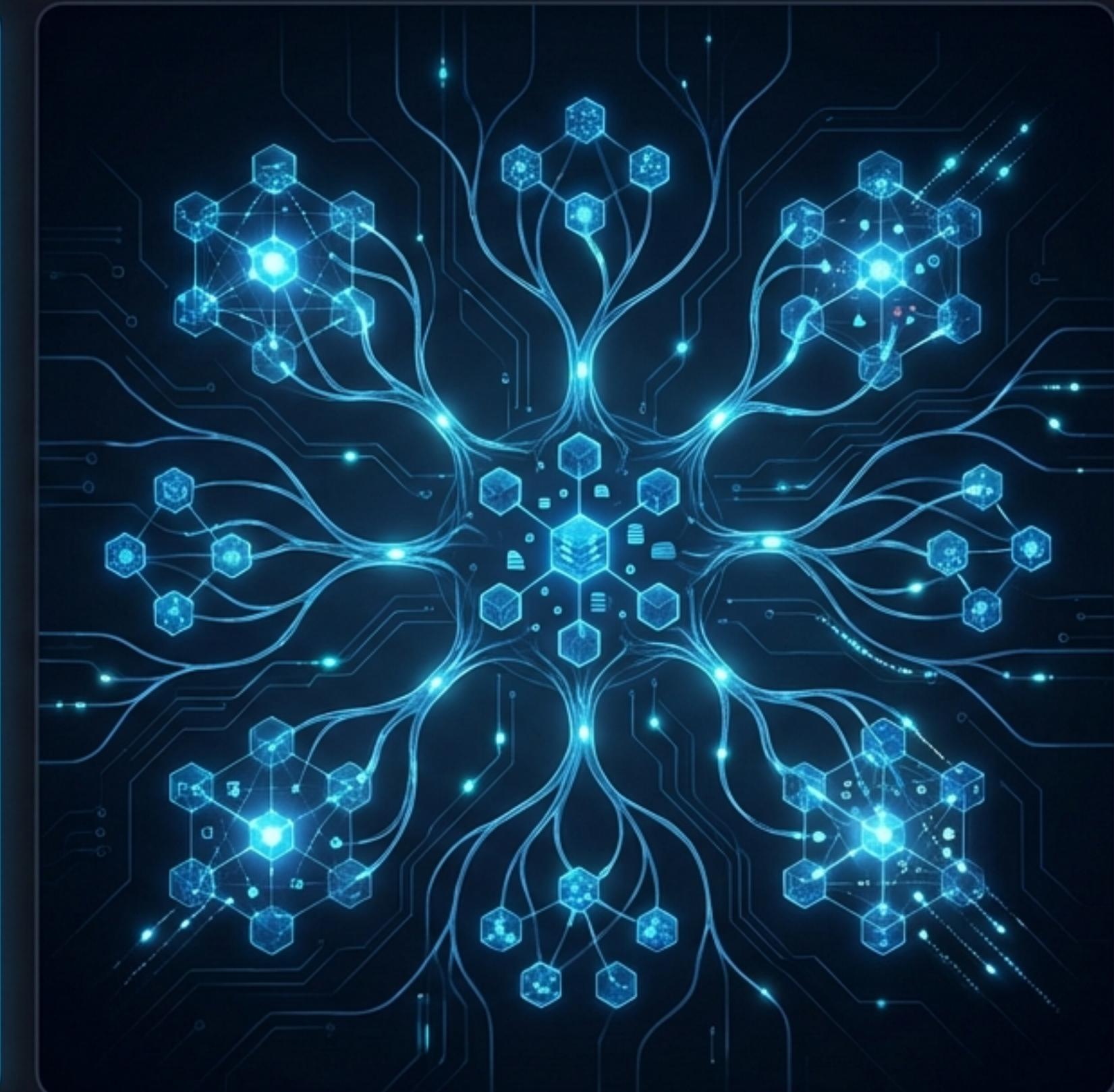




```
user@mycodb:~/ $ ./micro_swarm.exe --mode db_shell  
Micro-Swaem Shell v1.0 initialized.
```

# MycoDB & Die Micro-Swarm Shell

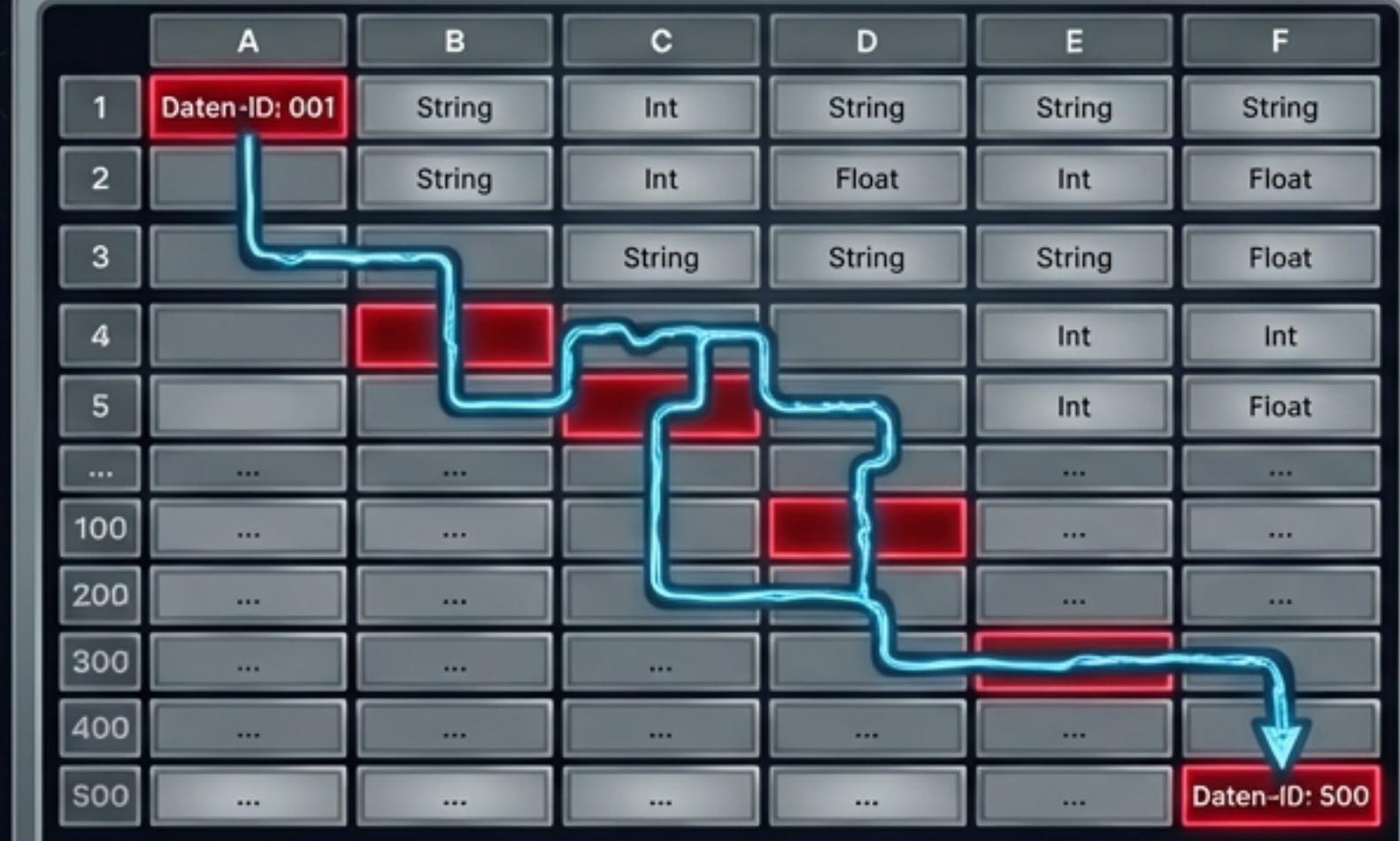
Von Null auf Datenbank-Profi:  
Das biologische Datenmodell  
verstehen und beherrschen.



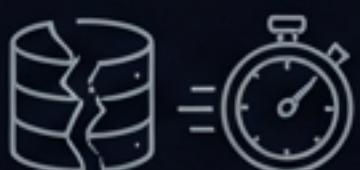
Eine Einführung in die MycoDB Architektur. Basierend auf dem 'Zero to Hero' Lernpfad.

# Was ist MycoDB? Biologisches Indexing statt starrer Tabellen.

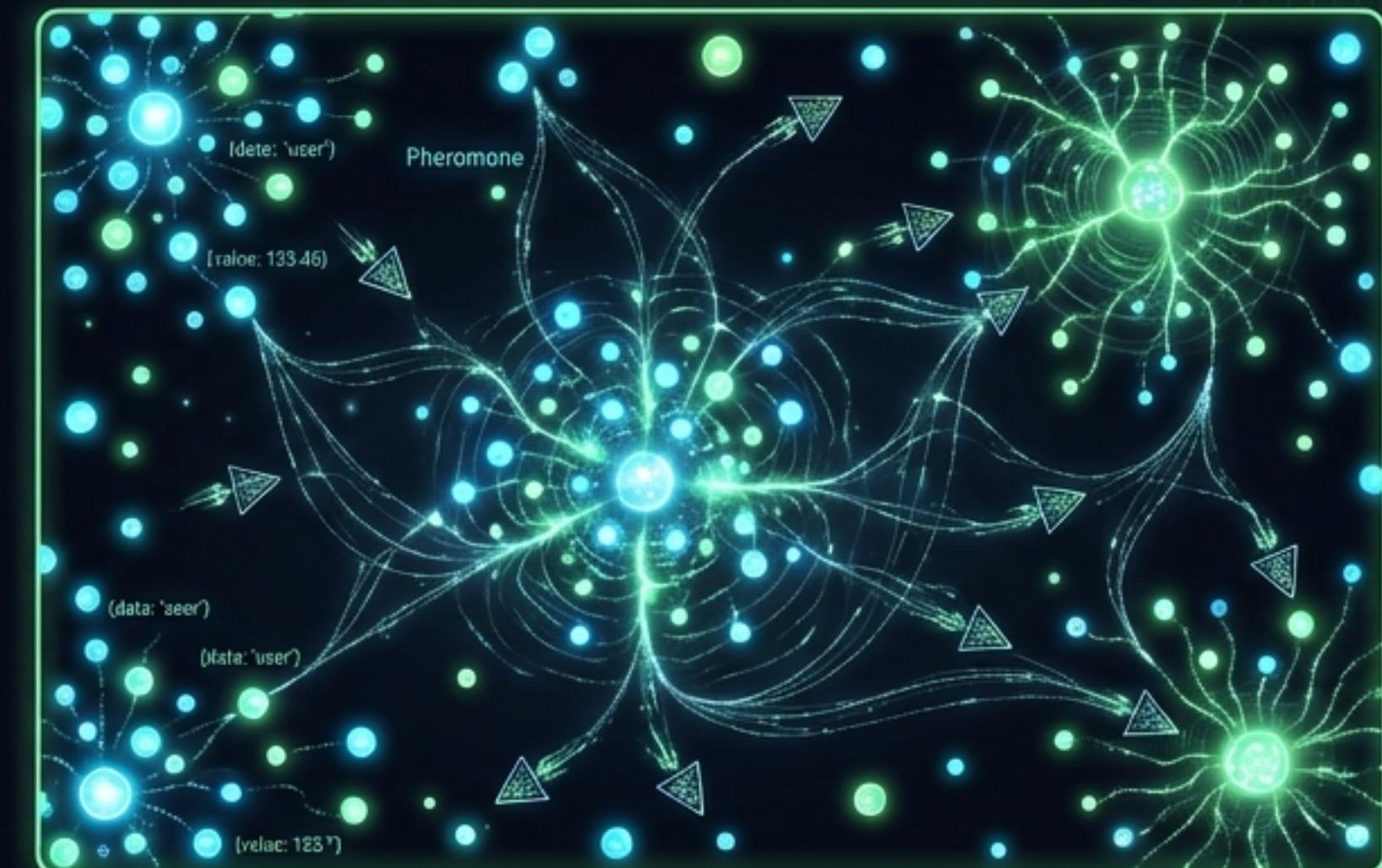
Traditionelles RDBMS



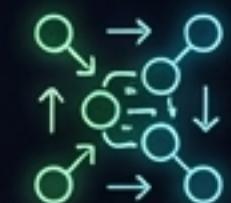
**Das Problem:** Logisch verwandte Daten liegen physisch getrennt. JOINS sind teuer und langsam.



MycoDB (Glass-Box AI)



**Die Lösung: Emergenz.** Daten sind 'Payloads', die von Agenten getragen werden. Sie folgen Pheromon-Spuren und clustern sich selbstständig.



*"Es ist die erste Datenbank, die man nicht nur abfragen, sondern der man beim 'Denken' (Sortieren) zusehen kann."*

# Der Morningstar-Report: Experimentelles Setup

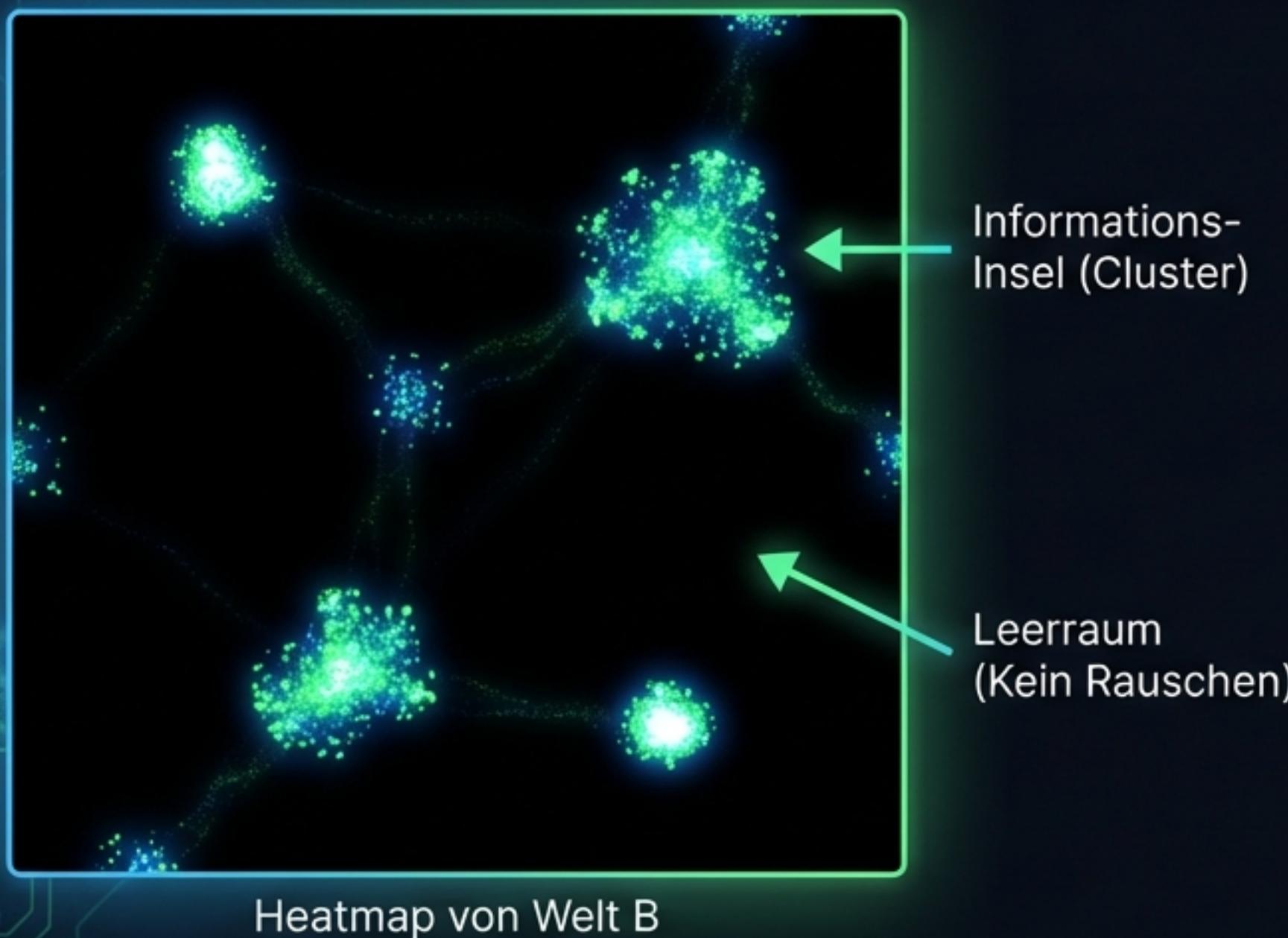
Testlauf vom 11. Jan 2026: 500.000 Datensätze im Vergleich.

Parameter	Welt A: "Der Maßanzug"	Welt B: "Die weite Welt"
Grid-Größe	900 x 900 (Kompakt)	2048 x 2048 (Weitläufig)
Datendichte	~62% (Gequetscht)	~12% (Sparse / Viel Freiraum)
Hypothese	Kurze Wege = Schnelle Suche?	Viel Leerraum = Ineffizient?

## Context

**Das Ziel:** Vergleich von 'Daten-Dichte' (Kompattheit) vs. 'Daten-Freiraum' (Emergenz) zur Optimierung von SQL-Abfragen.

# Das Ergebnis: Emergenz schlägt Dichte



Lokal-Scan (Radius 100):

**Welt B (Weit): 356 ms**

Welt A (Kompakt): 1.544 ms

- In der "weiten Welt" bildeten sich saubere Informations-Inseln.
- 97% der relevanten Daten lagen auf nur 0,06% der Fläche.
- In Welt A behinderten sich die Agenten gegenseitig ("Stau").

Fazit: Ein **Ingest-Prozess** mit genügend **Raum** transformiert Minuten-Abfragen in **Millisekunden**-Antworten.

# Die Shell: Ihr Cockpit

Der Einstieg (Lernwebseite: Erste 60 Sekunden)

## Zustandsbehaftung (Stateful)

Anders als SQL merkt sich die Shell den Kontext.

## Radius & Fokus

Ein gesetzter Radius oder Fokus bleibt für alle folgenden Befehle aktiv.

```
> .\micro_swarm.exe --mode db_shell --db chinook_optimized.myco
> Shell ready. Radius: GLOBAL. Focus: NONE.
> |
```

## Settings

Einstellungen wie 'limit' oder 'format' persistieren.

# Kapitel 1: Ingest & Struktur (Schwarm-Sort)

Wie die Daten in das System kommen



## Ingest

`ms_db_load_sql()`  
lädt SQL-Dump.

## Verteilung

Zufällige  
Startpositionen  
im Grid.

## Transport

Carrier-Agenten  
nehmen Payloads  
auf.

## Pheromone

Agenten folgen  
Foreign Keys  
(Beziehungen).

## Clustering

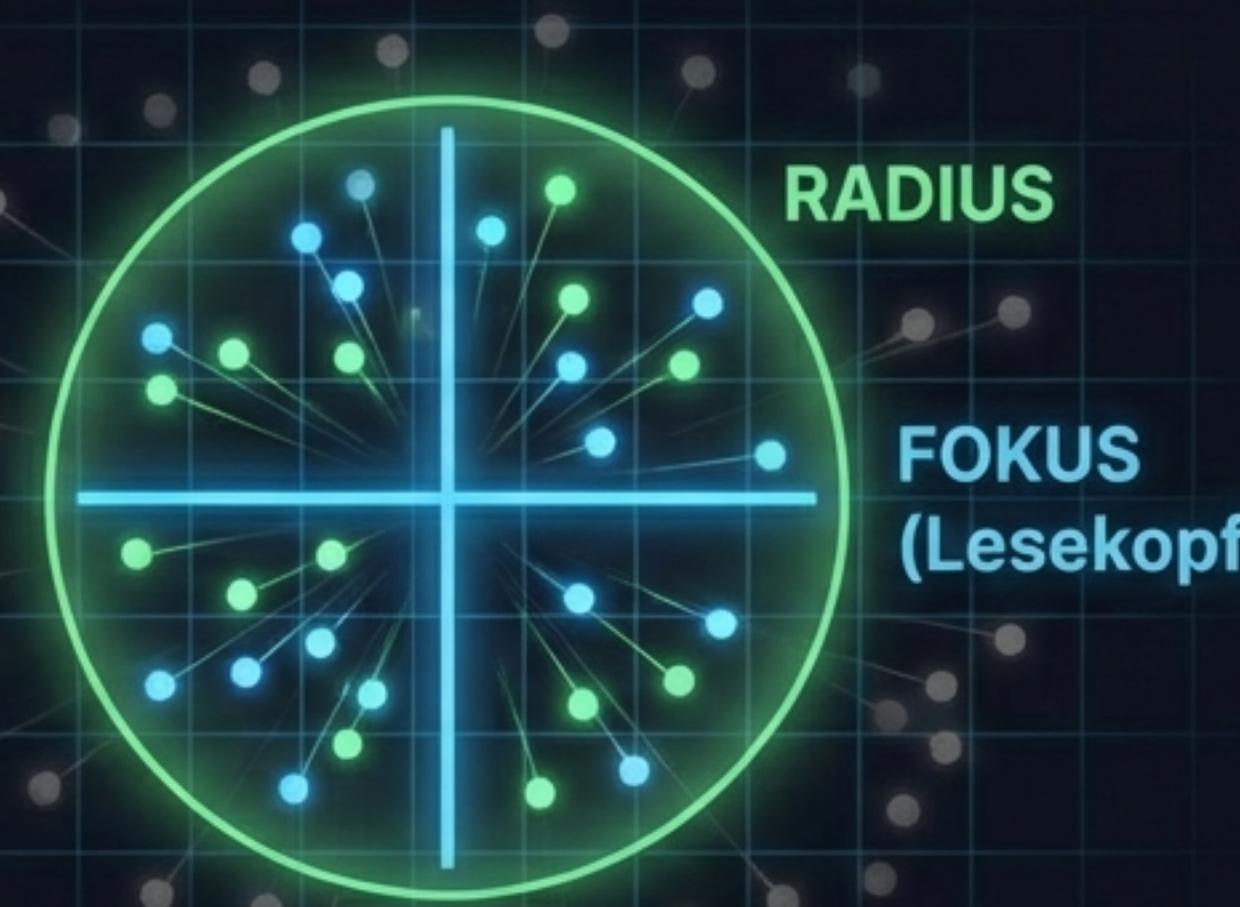
Alben wandern  
physisch zu  
Künstlern.

**Shell Commands**

```
Inspection Commands:  
- `tables`: "Listet alle Tabellen (Pheromon-Quellen)"  
- `schema <table>`: "Zeigt Spaltennamen"  
- `stats`: "Zählt Payloads im Grid"
```

# Kapitel 5: Fokus & Radius

Das Herzstück der MycoDB Navigation



## Command Explanation

- **goto <ID>**  
Setzt den Lesekopf physisch auf die X,Y-Position einer Payload (z.B. Künstler 'AC/DC').
- **radius <N>**  
Definiert den Suchkreis. Alle folgenden Abfragen sehen NUR Daten in diesem Kreis.
- **focus / unfocus**  
Zeigt aktuellen Status oder kehrt zum globalen Scan zurück.

**Performance Regel:** Kleiner Radius (< 100) + Guter Fokus = Maximale Geschwindigkeit (Nutzung der Informations-Inseln).

# Kapitel 2: Die Kurzschreibweise

Daten finden ohne SQL-Ballast

## Primary Key Lookup

Syntax: `Tabelle ID`

`>_Album 1`

Findet sofort das Album mit ID 1.

## Foreign Key / Spalten-Suche

Syntax: `Tabelle Spalte=Wert`

`>_Track AlbumId=1`

Findet alle Tracks, die zum Album 1 gehören.

## Global Search

Syntax: `Spalte=Wert`

`>_Name='Accept'`

Durchsucht ALLE Tabellen nach diesem Namen.

Ideal für schnelle Checks und Navigation im 'Myco-Shorthand'.

# Kapitel 3: SQL in MycoDB (SQL-Light)

## Die universelle Sprache – Kontext-Sensitiv

- Der Befehl `sql <Statement>` ermöglicht Standard-SQL.

Features: SELECT, WHERE, ORDER BY, LIMIT.

Wichtig: Das SQL wird **innerhalb** des aktuellen Radius ausgeführt.

Sucht nur lange Tracks  
in physischer Nähe von  
ID 5 (Künstler).



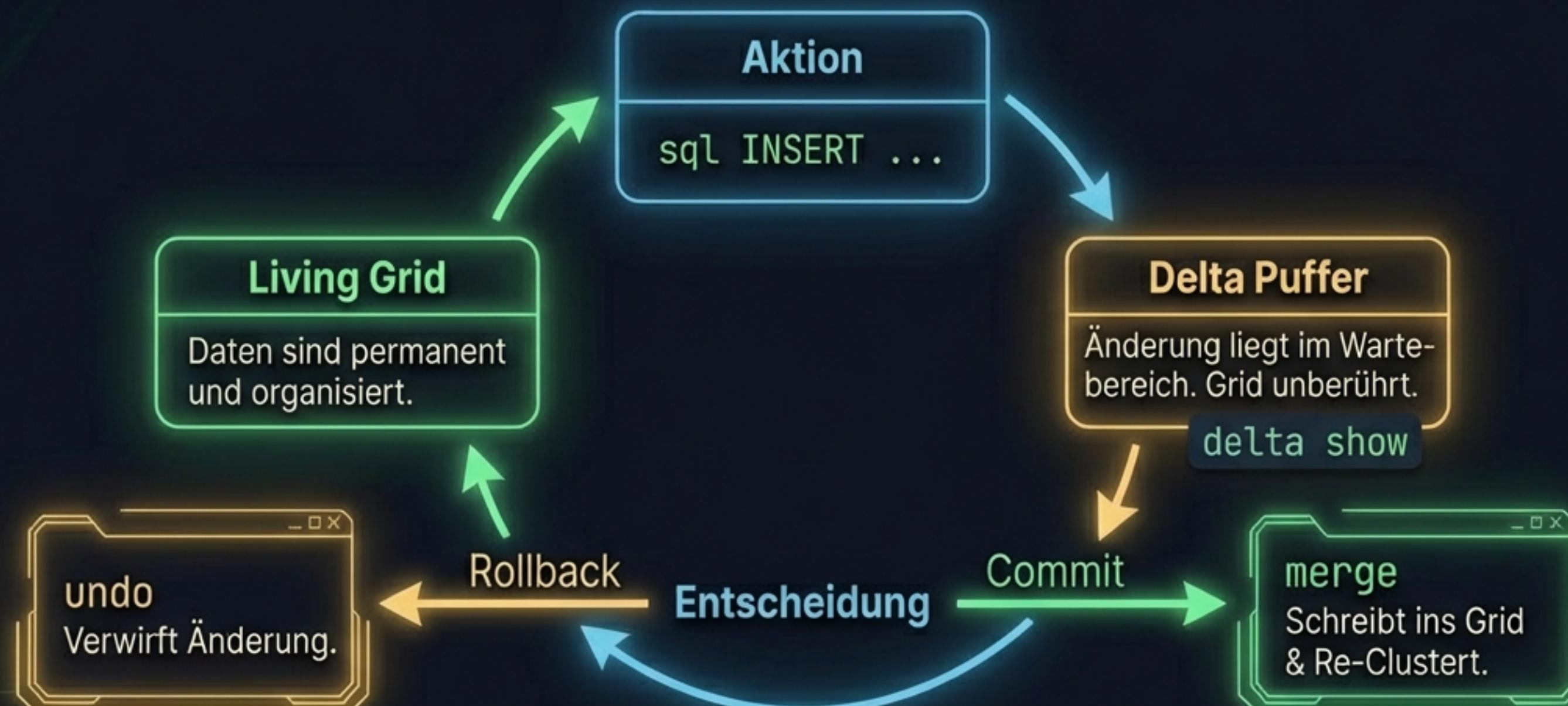
```
> goto 5
[System] Focus set to Payload 5 (X:120, Y:45)

> radius 50
[System] Radius set to 50.

> sql SELECT Name, Milliseconds FROM Track
WHERE Milliseconds > 300000
[Result] 3 rows found within radius 50.
| Name           | Milliseconds |
| ---            | ---           |
| Let There Be Rock | 366653      |
| Overdose       | 369319      |
| Problem Child  | 325041      |
```

# Kapitel 4: Das Delta-Prinzip

Daten sicher ändern in einem lebenden System



MycoDB hat keine ACID-Transaktionen, aber das Delta schützt vor Korruption.

# Kapitel 6: Produktivität & Tools

## Workflow-Automatisierung für Profis

### Makros (Automatisierung)

`save my_query`  
: Speichert letzten Befehl.

`run my_query`  
: Führt Makro aus.

`macros save <path>`  
: Exportiert alle Makros als JSON.

### Export (Reporting)

`export csv <path>`  
: Speichert Ergebnis als CSV.

`export json <path>`  
: Speichert Ergebnis als JSON.

`limit off`  
: Deaktiviert Zeilenlimit für vollen Export.

### History & Navigation

`history`  
: Zeigt Befehlsliste.

`last`  
: Wiederholt letzten Befehl.

`!n`  
: Wiederholt Befehl Nummer n.

# Zusammenfassung: Der 'Organic Data Journey'

**Ingest** (Chaos)



Zufällige  
Verteilung

**Schwarm-Sort**



Emergenz &  
Cluster-Bildung

**Shell** (Cockpit)



Navigation mit  
Focus & Radius

**Access** (Speed)



Zugriff auf 97%  
Relevanz in 356ms

## MycoDB nutzt Raum als Index: Platz schafft Geschwindigkeit.

- › Starten Sie die Shell und nutzen Sie 'ingest', um Ihre eigenen Daten laufen zu sehen.