Databases and SQL for Social Data

Jared Edgerton

Why Databases Matter

As data scale increases:

- Flat files become fragile
- Joins become expensive
- Reproducibility suffers

Databases provide structure, integrity, and performance.

Thinking Relationally

Databases organize data as:

- Tables
- Rows (records)
- Columns (attributes)

Meaning emerges from relationships across tables.

Tables Are Not Datasets

A table represents:

- One entity type
- One level of analysis

Complex questions require *multiple* tables.

Keys and Identifiers

Relational databases rely on:

- Primary keys
- Foreign keys
- Stable identifiers

Keys encode assumptions about identity.

Schemas as Measurement

A schema specifies:

- Table structure
- Variable types
- Allowed values

Schema design is a measurement decision.

SQL as a Query Language

SQL expresses:

- What data you want
- How tables relate
- How results are filtered and aggregated

Optimization is handled by the engine.

Basic SQL Pattern

```
SELECT *
FROM events
WHERE year ⩾ 2020;
```

Declarative, not procedural.

Joins Encode Theory

Joins combine tables via keys.

```
SELECT a.*, b.country_name
FROM events a
JOIN countries b
  ON a.country_id = b.id;
```

Which join you choose changes results.

Indexes and Performance

Indexes:

- Speed up queries
- Increase write cost
- Reflect expected access patterns

Indexing choices imply usage expectations.

Embedded Databases

For research workflows:

- SQLite
- DuckDB

They require no server and integrate cleanly with scripts.

Querying with DuckDB (Python)

```python
import duckdb

con = duckdb.connect('data.duckdb')
df = con.execute(
    'SELECT year, COUNT(*) FROM events GROUP BY year'
).fetchdf()
```

Querying with DuckDB (R)

```r
library(DBI)
library(duckdb)

con ← dbConnect(duckdb(), 'data.duckdb')
df ← dbGetQuery(
  con,
  'SELECT year, COUNT(*) FROM events GROUP BY year'
)
```

SQLite as an Alternative

SQLite:

- Is file-based
- Is widely supported
- Works well for moderate-sized data

Often sufficient for social-science projects.

Querying from Analysis Code

Best practice:

- Keep SQL in strings or files
- Call from Python or R
- Avoid manual data exports

Queries are part of the pipeline.

Databases and Reproducibility

Databases support:

- Explicit schemas
- Versioned queries
- Consistent joins
- Auditable transformations

They reduce hidden data manipulation.

Common Failure Modes

Common problems include:

- Implicit joins
- Duplicated rows
- Missing keys
- Silent many-to-many merges

Always inspect join results.

What We Emphasize in Practice

- Design schemas deliberately
- Use SQL for data construction
- Query from scripts, not GUIs
- Treat joins as theoretical choices

Discussion

- What entities matter in your data?
- Where do joins encode assumptions?
- When does SQL outperform flat files?