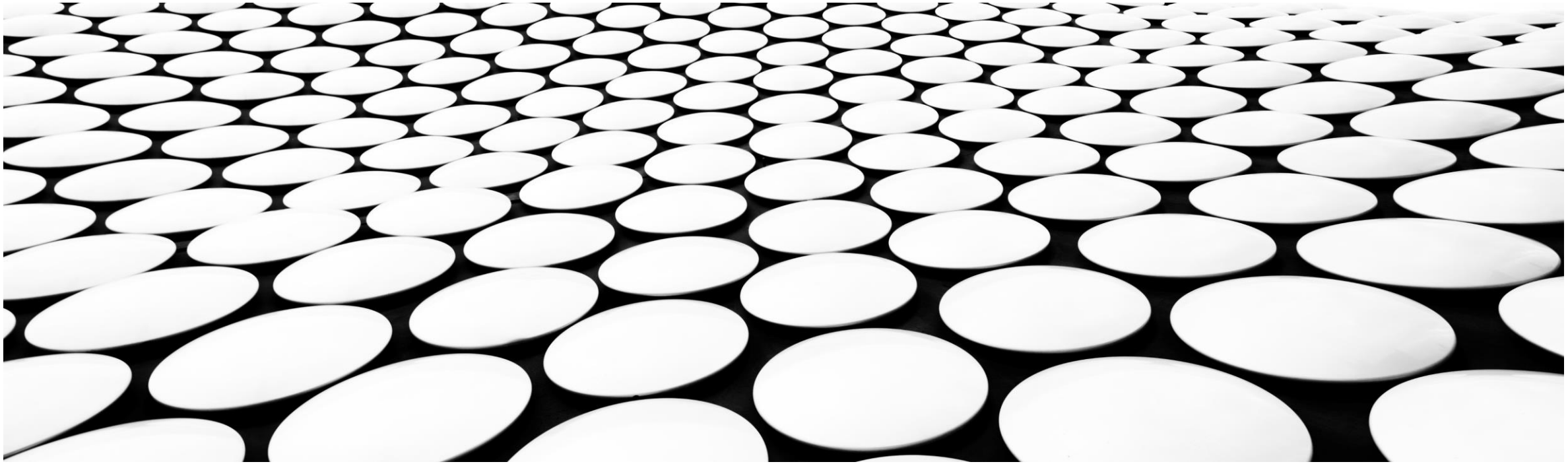
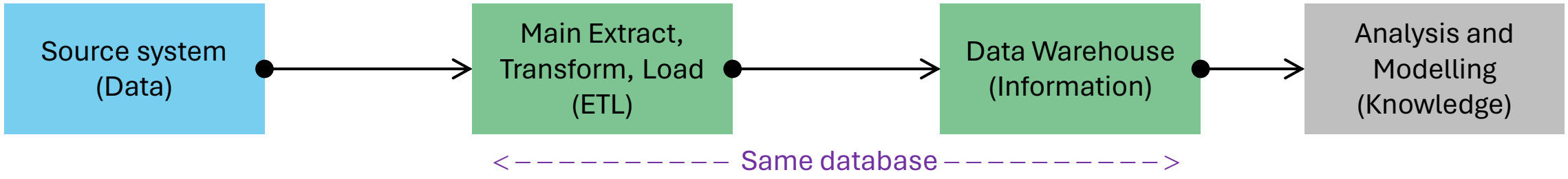
IMPACT OF SQL QUERY DESIGN AND INDEX USAGE ON DATA EXTRACTION DURATION

JOHN WONG, 2 MAY 2024

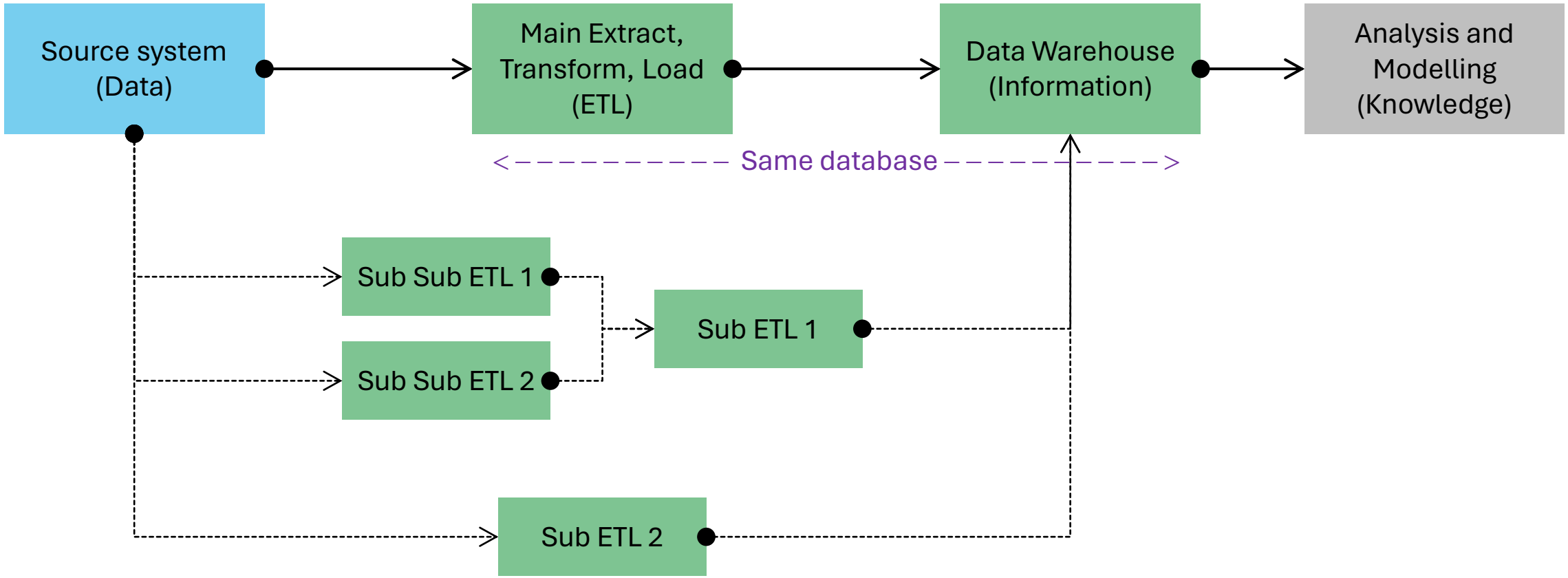


BACKGROUND

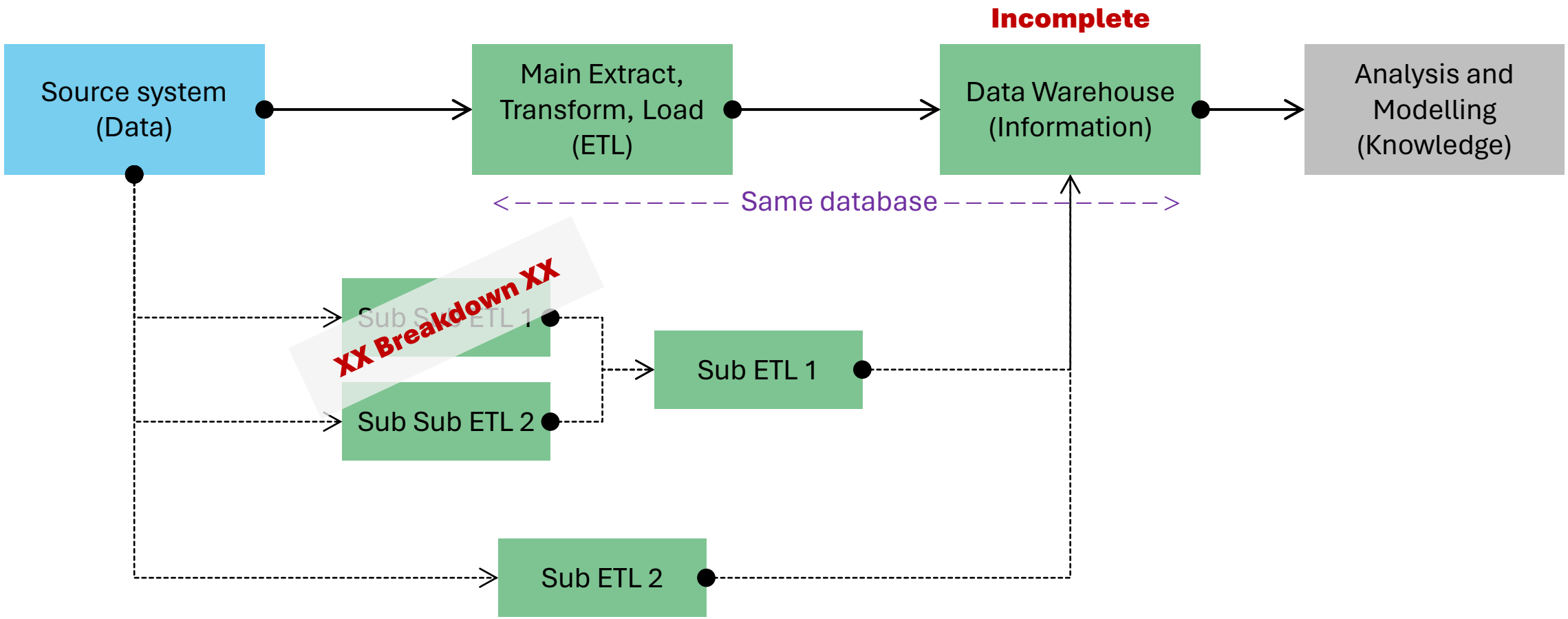
ETL PROCESS – DATA > INFORMATION > KNOWLEDGE



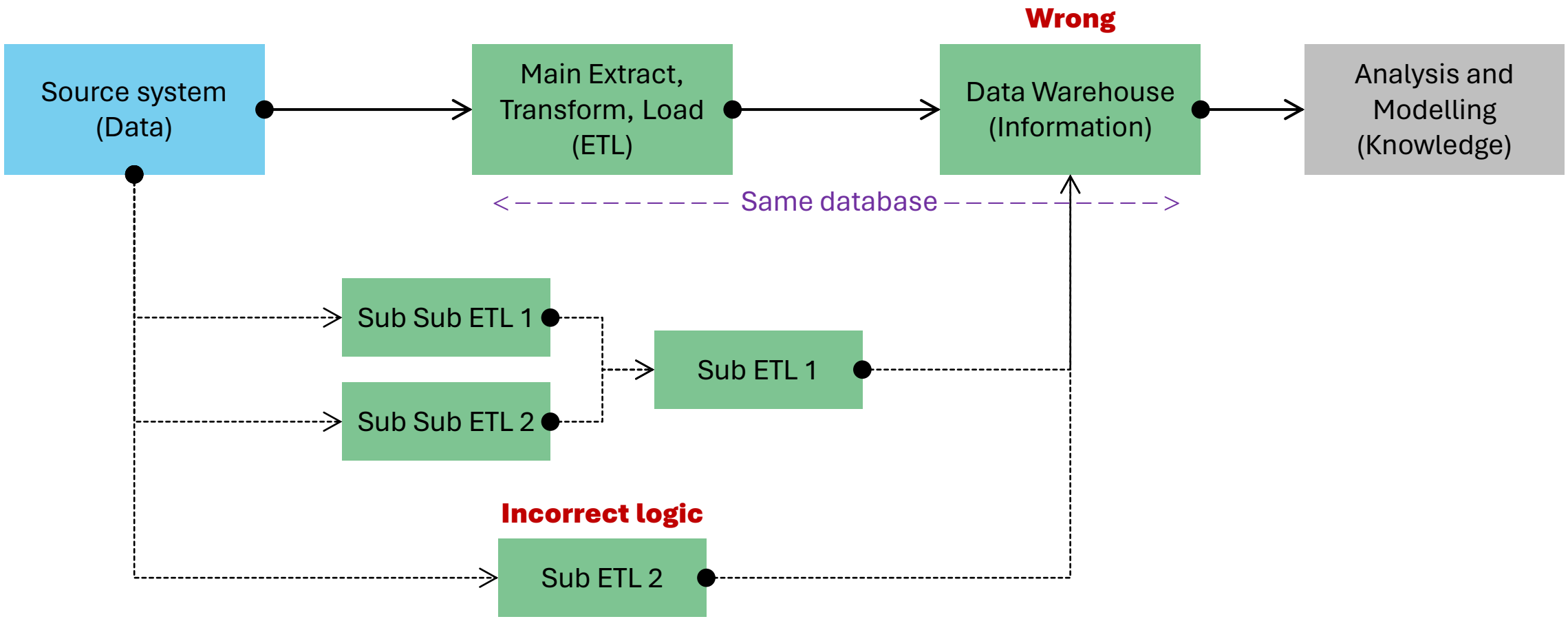
ETL PROCESS – “MAIN” CALLS WORKFLOWS “SUB” AND “SUB SUB”



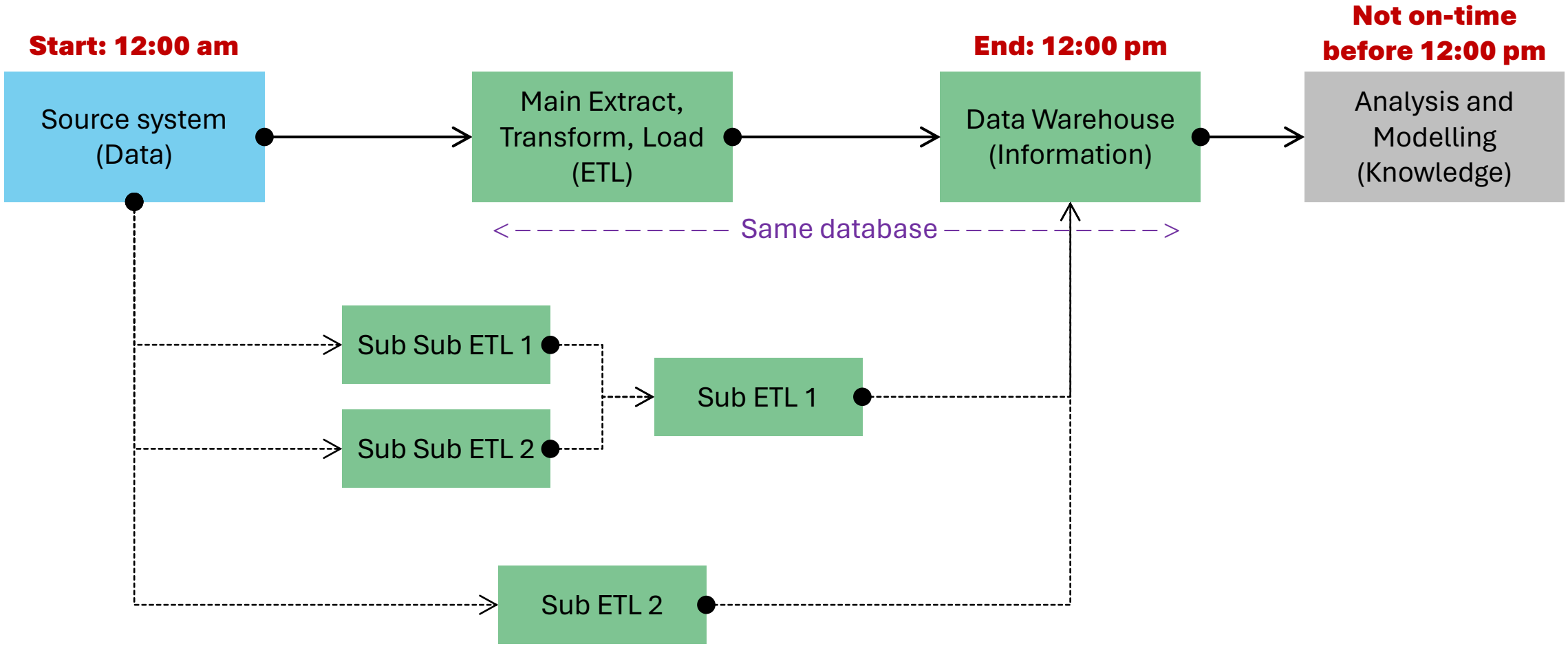
ETL PROCESS – WHAT CAN GO WRONG



ETL PROCESS – WHAT CAN GO WRONG



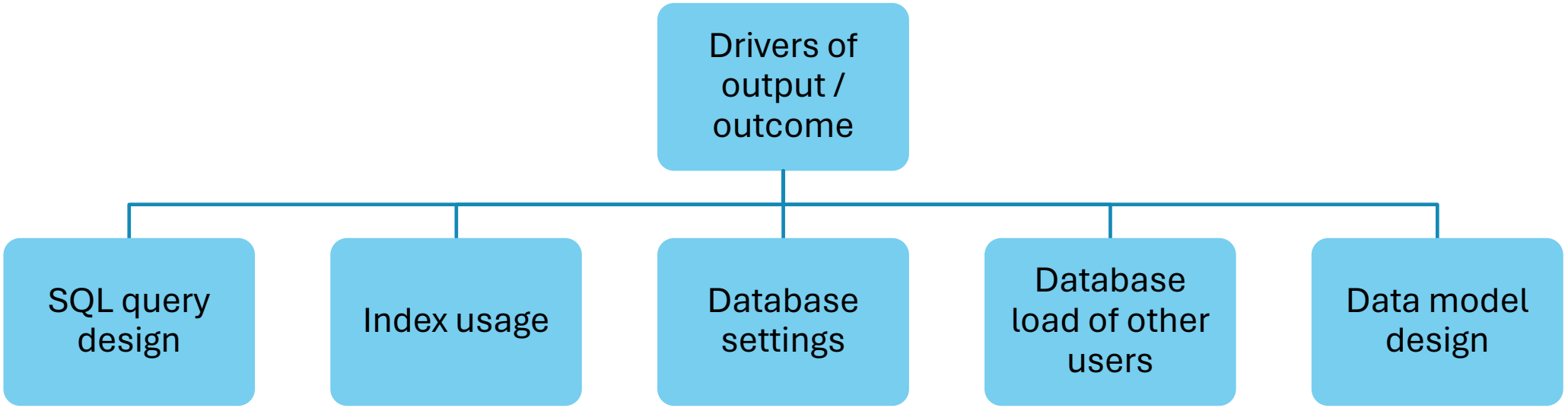
ETL PROCESS – WHAT CAN GO WRONG



ETL PROCESS – DESIRED OUTPUT / OUTCOME

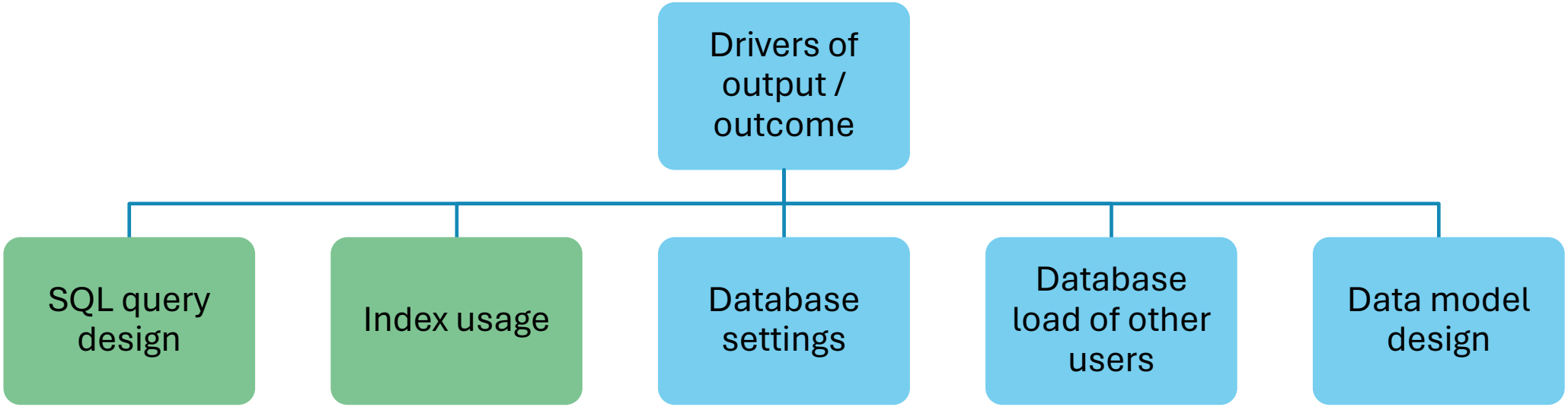
- Information is complete
- Information is correct
- Information is on-time
- Information acquisition is low-cost (move, compute, store)
- Information acquisition is low-load on database for other users

ETL PROCESS – DRIVERS OF DESIRED OUTPUT / OUTCOME



OBJECTIVE

EXPLORE IMPACT OF SQL QUERY DESIGN AND INDEX USAGE ON DATA EXTRACTION DURATION



<u>Design of experiment (DOE) to explore</u>

LEFT JOIN versus INNER JOIN

WHERE versus ON

Common Table Expression (CTE) versus Sub-Query

Automatic index versus manual index versus no index

Create Table versus memory

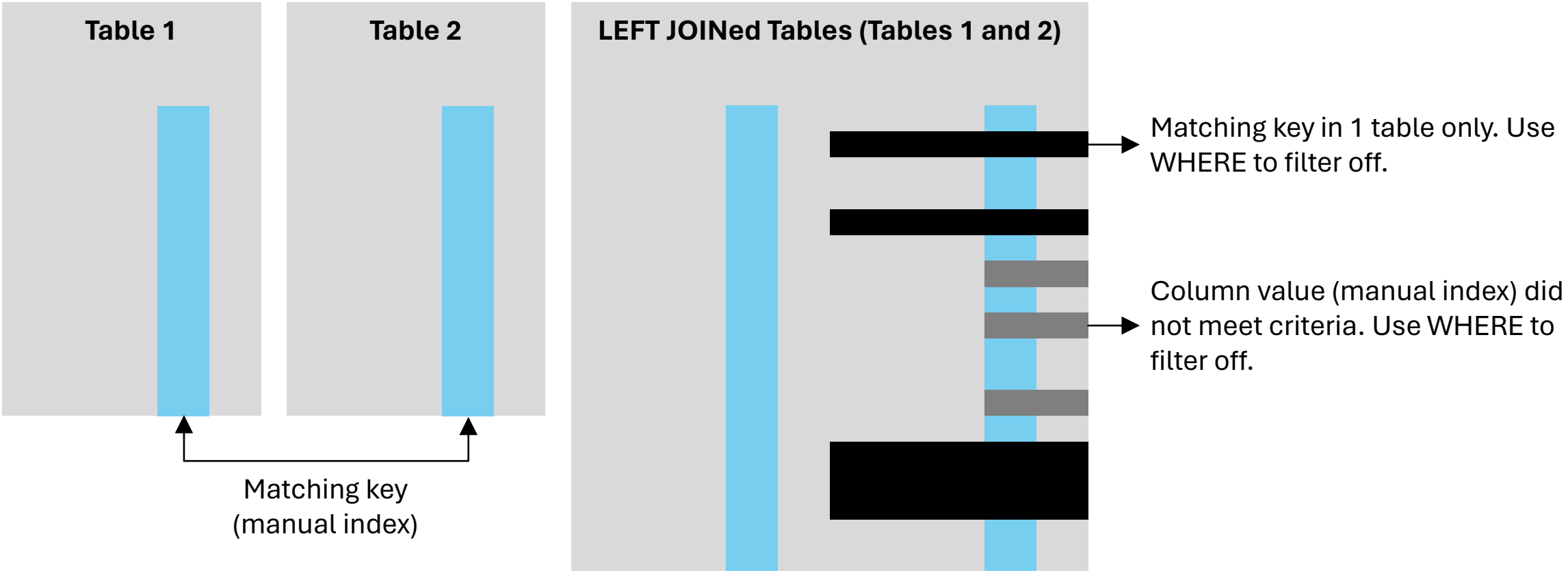
METHOD

DOE 1 METHOD – INDEX USAGE

- Join 2 tables using SQL query design with LEFT JOIN and WHERE
- Retain rows of LEFT JOINed tables where 2 tables have same matching key
- Retain rows of LEFT JOINed tables where column value meets criteria
- Create manual index on matching keys of 2 tables and column value of 1 table

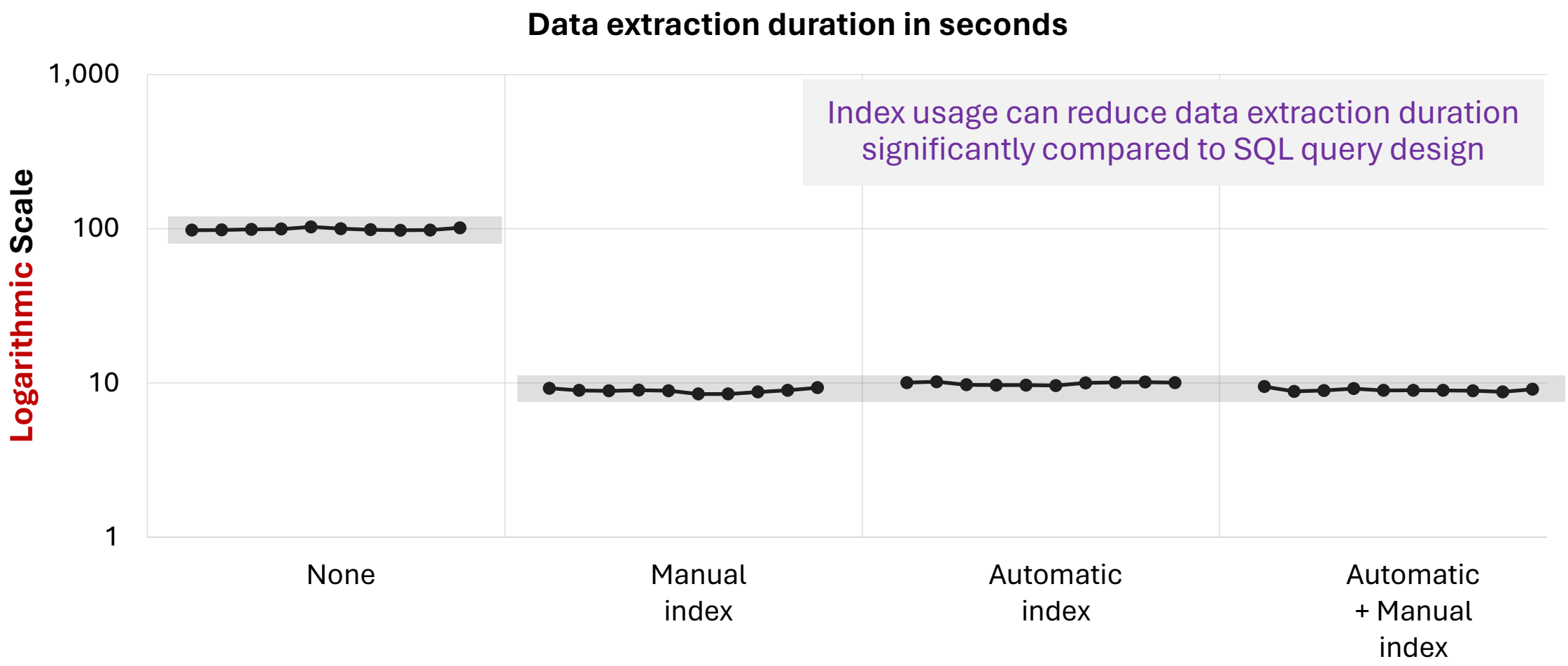
Run number	Automatic index	Manual index	Data extraction duration
Open database > 1 to 10 > Close database	No	No	
Open database > 1 to 10 > Close database	No	Yes	
Open database > 1 to 10 > Close database	Yes	No	
Open database > 1 to 10 > Close database	Yes	Yes	

DOE 1 METHOD – INDEX USAGE

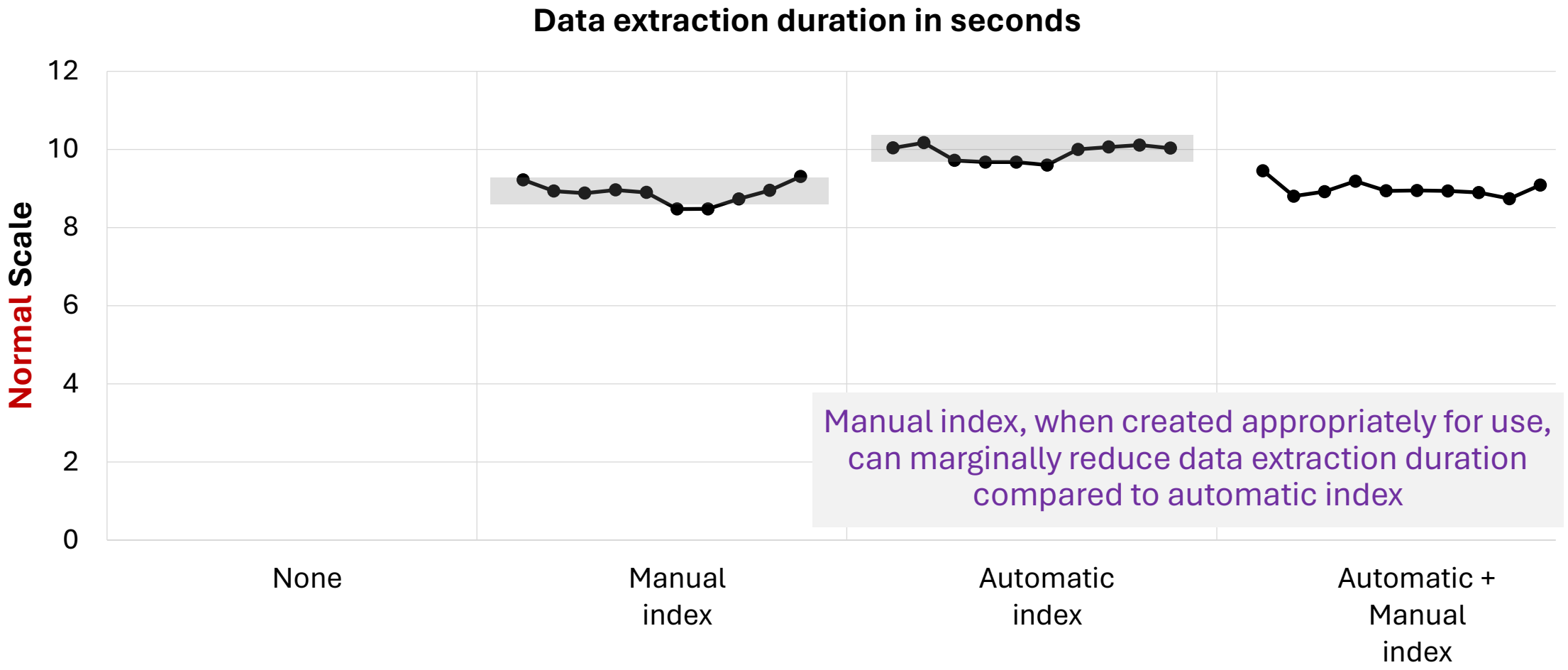


RESULT

DOE 1 RESULT – INDEX USAGE



DOE 1 RESULT – INDEX USAGE



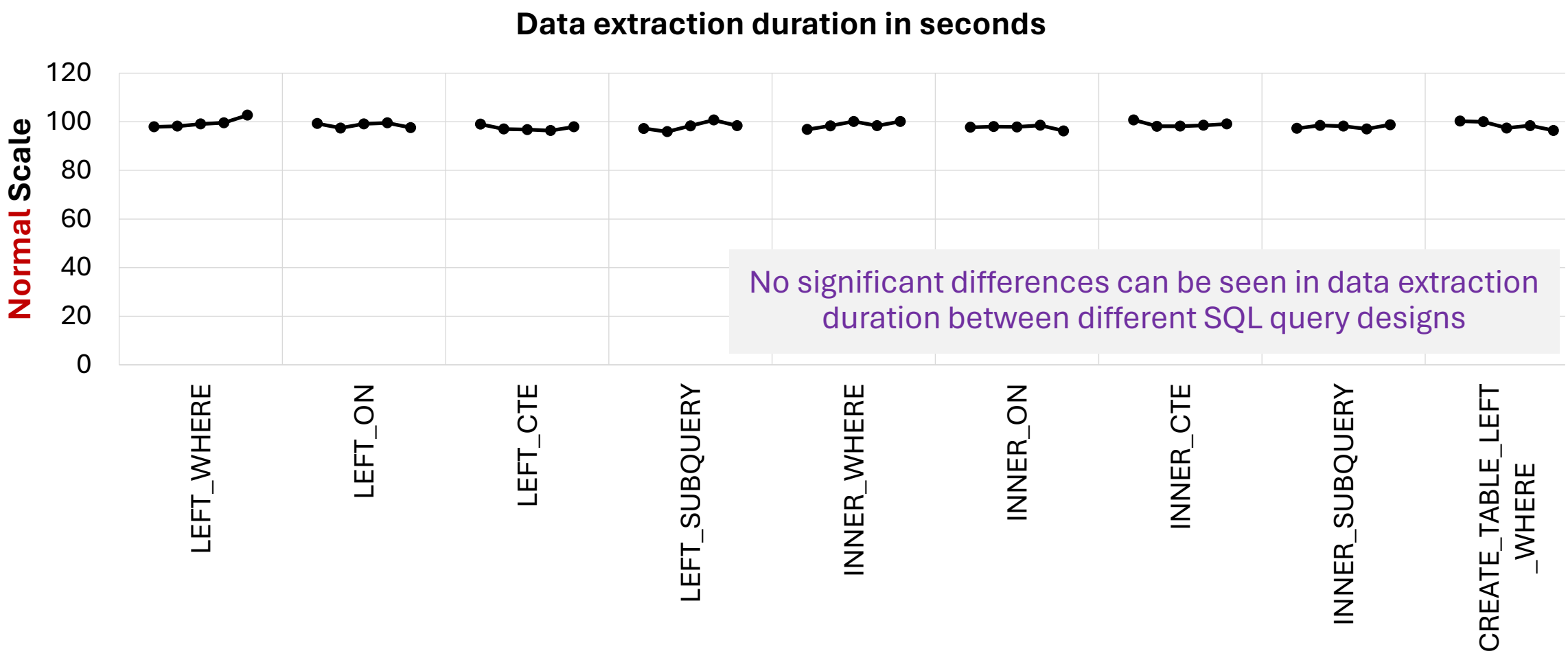
METHOD

DOE 2 METHOD – SQL QUERY DESIGN

SQL query design	Run number	Automatic index	Manual index	Data extraction duration
LEFT JOIN, WHERE	Open database > 1 to 5 > Close database	No	No	
LEFT JOIN, ON	Open database > 1 to 5 > Close database	No	No	
LEFT JOIN, CTE	Open database > 1 to 5 > Close database	No	No	
LEFT JOIN, SUBQUERY	Open database > 1 to 5 > Close database	No	No	
INNER JOIN, WHERE	Open database > 1 to 5 > Close database	No	No	
INNER JOIN, ON	Open database > 1 to 5 > Close database	No	No	
INNER JOIN, CTE	Open database > 1 to 5 > Close database	No	No	
INNER JOIN, SUBQUERY	Open database > 1 to 5 > Close database	No	No	
CREATE TABLE, LEFT JOIN, WHERE	Open database > 1 to 5 > Close database	No	No	

RESULT

DOE 2 RESULT – SQL QUERY DESIGN



No significant differences can be seen in data extraction duration between different SQL query designs

CONCLUSION

CONCLUSION

- Index usage can reduce data extraction duration significantly compared to SQL query design
- When manual index created appropriately for use, it can reduce data extraction duration marginally compared to automatic index
- When data extraction duration is short, it can also reduce load on database for other users

DB BROWSER FOR SQLITE

MANUAL INDEX

File Edit View Tools Help

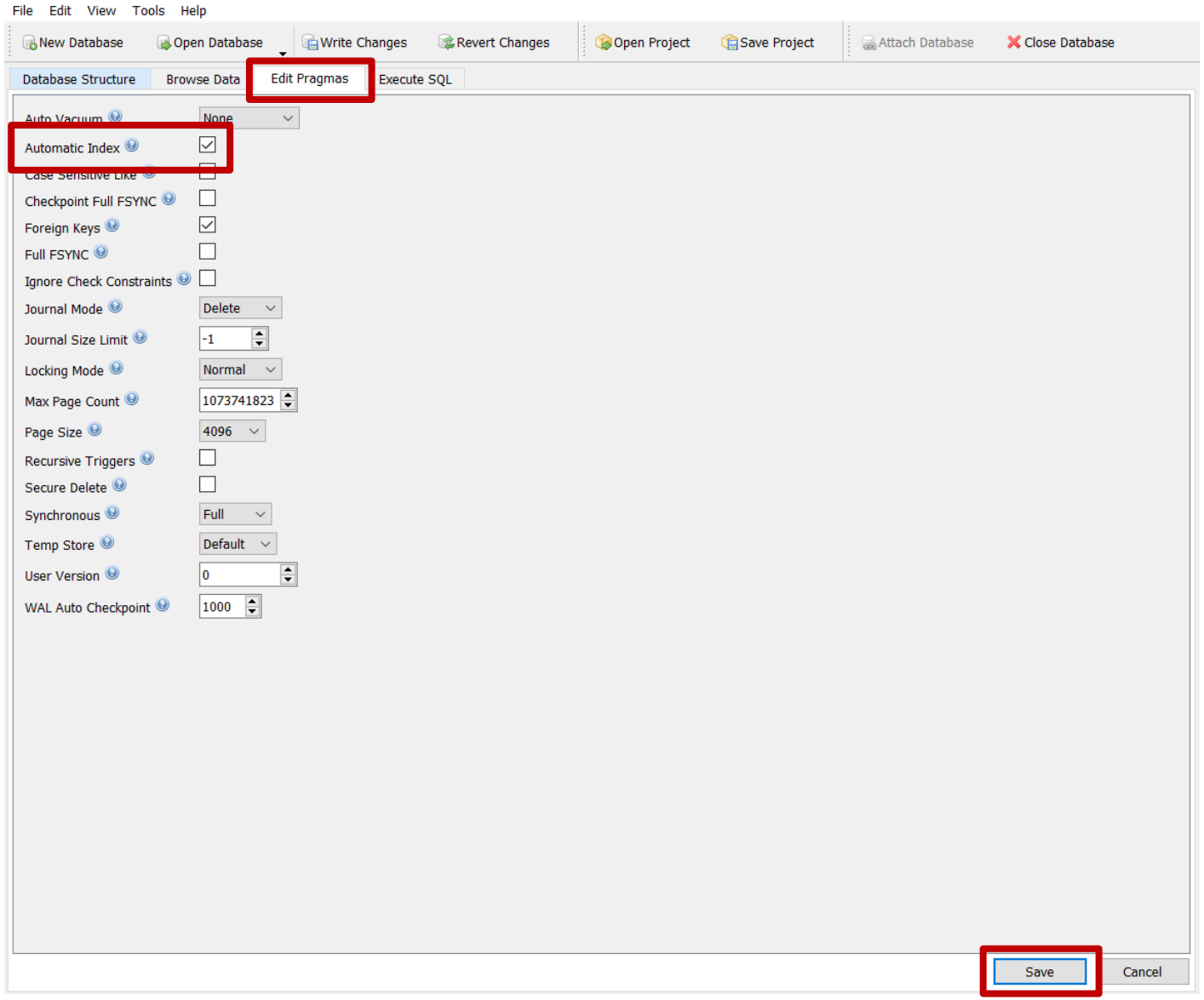
New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Print

Name	Type	Schema
Tables (4)		
durations_table		CREATE TABLE durations_table (source TEXT, time_start_stop TEXT)
table_1		CREATE TABLE "table_1" ("id" INTEGER, "value_integer" INTEGER, "value_string" TEXT, PRIMARY KEY("id"))
table_2		CREATE TABLE "table_2" ("id" INTEGER, "value_integer" INTEGER, "value_string" TEXT, "value_float" REAL, PRIMARY KEY("id"))
table_3		CREATE TABLE table_3(id INT, value_integer INT, value_string TEXT, value_float REAL)
Indices (3)		
idx_t1_string		CREATE INDEX "idx_t1_string" ON "table_1" ("value_string" ASC)
idx_t2_float		CREATE INDEX "idx_t2_float" ON "table_2" ("value_float" ASC)
idx_t2_string		CREATE INDEX "idx_t2_string" ON "table_2" ("value_string" ASC)
Views (0)		
Triggers (0)		

AUTOMATIC INDEX



SQL QUERY WITH DATA EXTRACTION DURATION

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

exercise_2_design_1_LEFT_WHERE.sql

```sql
-- Step 1: Create time to store start and end time
CREATE TABLE IF NOT EXISTS durations_table (source TEXT, time_start_stop TEXT);

-- Step 2: Delete previous data
DELETE FROM durations_table;

-- Step 3: This is the start time
INSERT INTO durations_table (source, time_start_stop) VALUES ('start', julianday('now'));

-- Step 4: Your query
-- ################### Design 1 ##########################################
SELECT t1.*, t2.*
FROM table_1 t1
LEFT JOIN table_2 t2
ON t1.value_string = t2.value_string
WHERE t2.id IS NOT NULL AND t2.value_float >= 1500;

-- Step 5: This is the end time
INSERT INTO durations_table (source, time_start_stop) VALUES ('end', julianday('now'));

-- Step 6: Display start and end time in human understandable format
SELECT source, strftime('%Y-%m-%d %H:%M:%f', time_start_stop) AS date_time FROM durations_table;

-- Step 7: ONLY IF NECESSARY
-- DROP TABLE durations_table;
```

Get data extraction start time

SQL QUERY WITH DATA EXTRACTION DURATION

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

exercise_2_design_1_LEFT_WHERE.sql

```sql
-- Step 1: Create time to store start and end time
CREATE TABLE IF NOT EXISTS durations_table (source TEXT, time_start_stop TEXT);

-- Step 2: Delete previous data
DELETE FROM durations_table;

-- Step 3: This is the start time
INSERT INTO durations_table (source, time_start_stop) VALUES ('start', julianday('now'));

-- Step 4: Your query
-- ################### Design 1 ##########################################
SELECT t1.*, t2.*
FROM table_1 t1
LEFT JOIN table_2 t2
ON t1.value_string = t2.value_string
WHERE t2.id IS NOT NULL AND t2.value_float >= 1500;

-- Step 5: This is the end time
INSERT INTO durations_table (source, time_start_stop) VALUES ('end', julianday('now'));

-- Step 6: Display start and end time in human understandable format
SELECT source, strftime('%Y-%m-%d %H:%M:%f', time_start_stop) AS date_time FROM durations_table;

-- Step 7: ONLY IF NECESSARY
-- DROP TABLE durations_table;
```

Insert your SQL query here

SQL QUERY WITH DATA EXTRACTION DURATION

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

exercise_2_design_1_LEFT_WHERE.sql

```sql
-- Step 1: Create time to store start and end time
CREATE TABLE IF NOT EXISTS durations_table (source TEXT, time_start_stop TEXT);

-- Step 2: Delete previous data
DELETE FROM durations_table;

-- Step 3: This is the start time
INSERT INTO durations_table (source, time_start_stop) VALUES ('start', julianday('now'));

-- Step 4: Your query
-- ################### Design 1 ##########################################
SELECT t1.*, t2.*
FROM table_1 t1
LEFT JOIN table_2 t2
ON t1.value_string = t2.value_string
WHERE t2.id IS NOT NULL AND t2.value_float >= 1500;

-- Step 5: This is the end time
INSERT INTO durations_table (source, time_start_stop) VALUES ('end', julianday('now'));

-- Step 6: Display start and end time in human understandable format
SELECT source, strftime('%Y-%m-%d %H:%M:%f', time_start_stop) AS date_time FROM durations_table;

-- Step 7: ONLY IF NECESSARY
-- DROP TABLE durations_table;
```

Get data extraction end time and display both times

Hope you find this useful!

CONNECT WITH ME

CONNECT WITH ME

- LinkedIn
  - https://www.linkedin.com/in/wongchikeongjohn/

- GitHub
  - https://github.com/johnwck/my_da_ds_work/tree/master