IMEN891K / IDSC501 / AXGS-501 · DATA ENGINEERING · FALL 2026

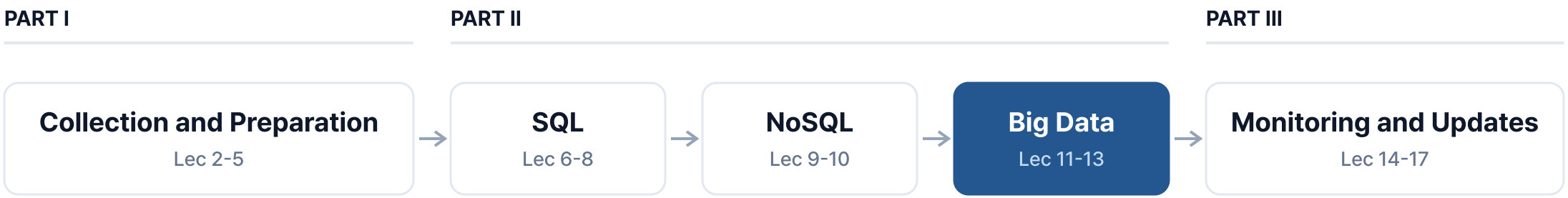
12.

Big Data II: Data Warehousing

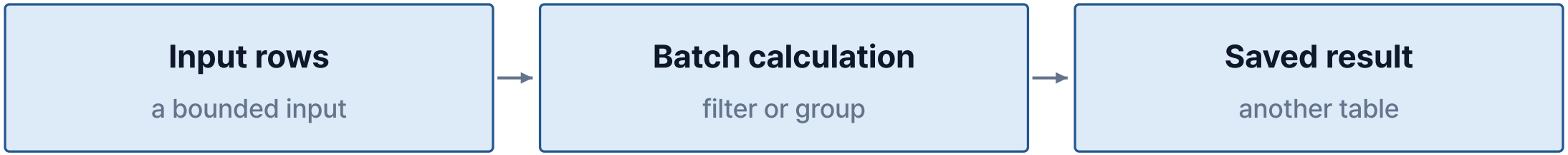
Sangwoo Mo · POSTECH

Industrial & Management Engineering

Where We Are


PART I
PART II
PART III
Collection and Preparation
Lec 2-5
SQL
Lec 6-8
NoSQL
Lec 9-10
Big Data
Lec 11-13
Monitoring and Updates
Lec 14-17

Recap: Batch Results



A saved result can serve repeated analysis

Outline: Big Data II

01	**Analytical Copies**	keep reporting work separate from sales transactions
02	**Rows for Analysis**	connect each sale to its product and daily total
03	**Refreshing Reports**	refresh the copy in the right order without counting twice

01 · Analytical Copies

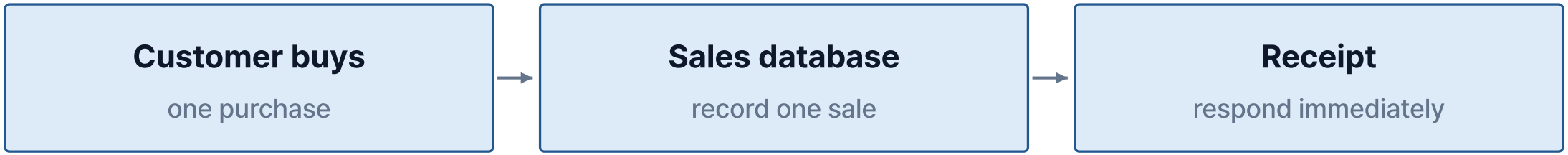
01	**Analytical Copies**	keep reporting work separate from sales transactions
02	**Rows for Analysis**	connect each sale to its product and daily total
03	**Refreshing Reports**	refresh the copy in the right order without counting twice

The Daily Sales Report



A report needs a clear date range and one correct total for each date

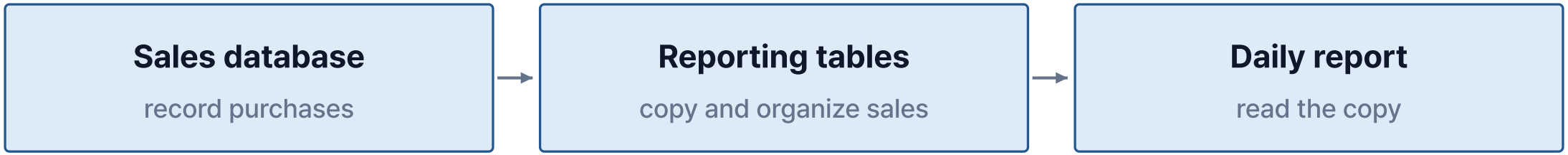
Sales Transactions



The operational database records each sale as it happens

An Analytical Copy

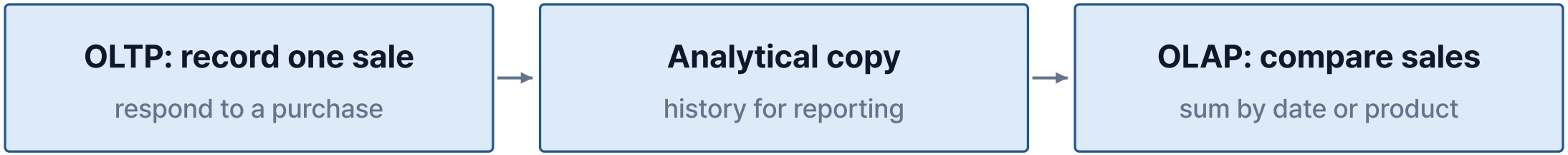
A data warehouse stores data for repeated analytical queries



A data warehouse keeps an organized copy for analysis

Transactions and Analysis

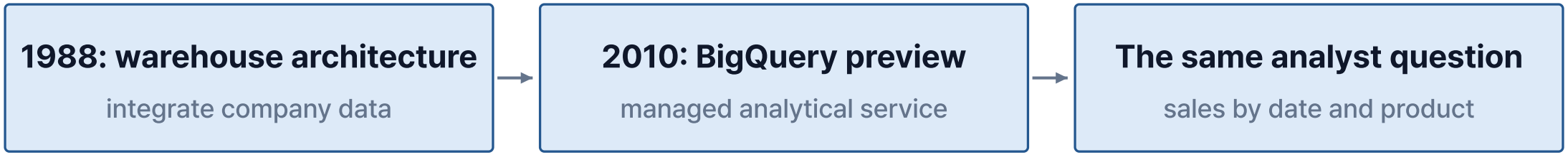
The same sales data supports different kinds of work



OLTP records individual purchases; OLAP summarizes and compares many purchases

Warehousing Becomes a Service

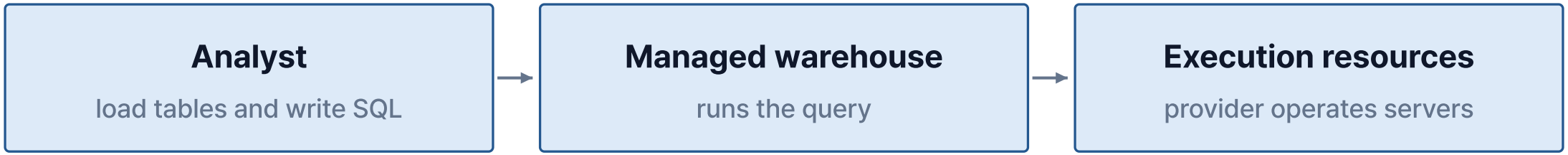
Warehouses organized company data before cloud services managed the infrastructure



Managed services change who operates the system; analysts still define the query

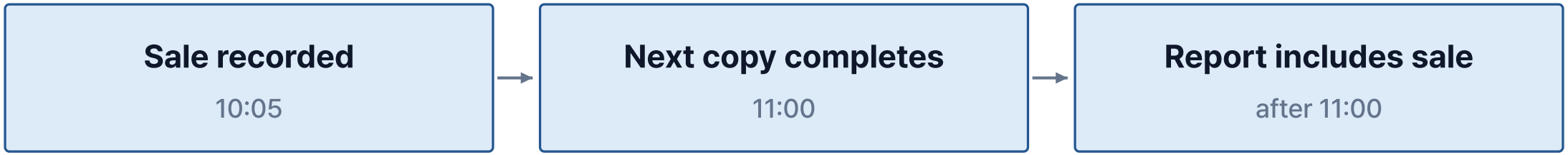
A Managed Warehouse

Serverless means the service manages the servers for the user



The service runs the infrastructure; the analyst defines the tables and query

Copy Freshness



The report includes only the sales already copied

02 · Rows for Analysis

01	Analytical Copies	keep reporting work separate from sales transactions
02	**Rows for Analysis**	connect each sale to its product and daily total
03	Refreshing Reports	refresh the copy in the right order without counting twice

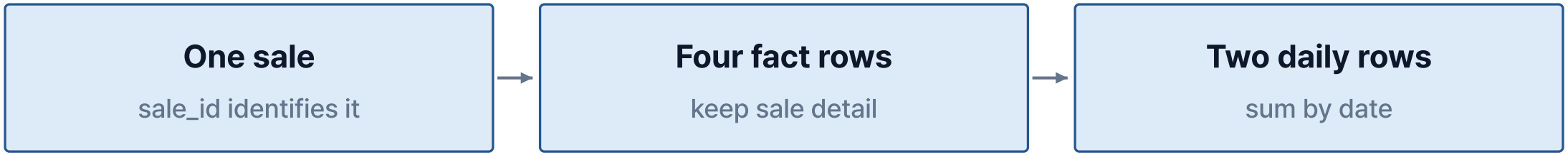
Four Sales

sale_id	sale_date	product_id	amount
1	2025-10-02	P1	300
2	2025-10-02	P2	200
3	2025-10-03	P1	400
4	2025-10-03	P2	100

October 2 has 300 + 200; October 3 has 400 + 100

A Row Has a Meaning

The grain of fact_sales is one sale



Grain means what one row represents

A Sales Fact

A fact row records one sale and its amount

sale_id	sale_date	product_id	amount
3	2025-10-03	P1	400

one row = one sale

Sale 3 records 400 for product P1 on October 3

Product Descriptions

dim_product has one row per product

product_id	name
P1	Chocolate
P2	Ice-cream

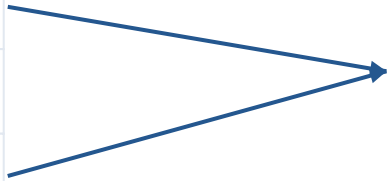
A dimension gives a description for each product code

Facts and a Dimension

product_id connects each sale to one product description

fact_sales: one row per sale

sale_id	product_id	amount
1	P1	300
2	P2	200
3	P1	400
4	P2	100



dim_product: one row per product

product_id	name
P1	Chocolate
P2	Ice-cream

product_id links a sale to its product name

Two P1 sales share the same Chocolate description

Names on Sales

```
SELECT sale_id,name,amount
FROM fact_sales JOIN dim_product USING(product_id)
WHERE sale_date='2025-10-03'
ORDER BY sale_id;
```

sale_id	name	amount
3	Chocolate	400
4	Ice-cream	100

The join adds names while keeping one row per sale

A Daily Total

Grouping changes the grain from one sale to one date

```sql
SELECT sale_date,SUM(amount) AS total
FROM fact_sales
GROUP BY sale_date ORDER BY sale_date;
```

sale_date	total
2025-10-02	500
2025-10-03	500

Four sale rows become two daily totals

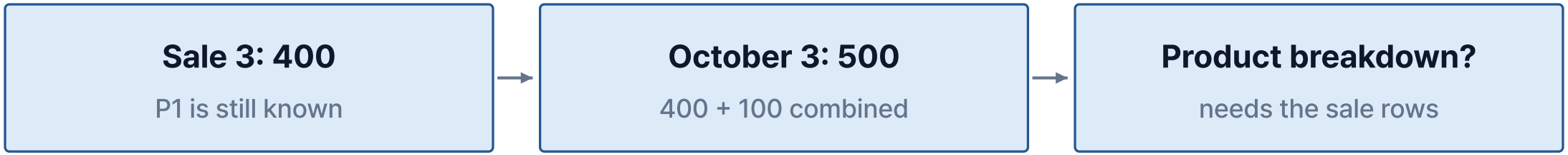
Two Views of the Same Sales

Changing the grouping answers a different analytical question

product	October 2	October 3	total
Chocolate	300	400	700
Ice-cream	200	100	300
All products	500	500	1000

The same four sales support daily totals and product totals

Detail and Summary



The daily total alone cannot recover the product amounts

Demo: Daily Sales

1. Sum amount for each sale_date
2. Keep only product P1 and compare the daily totals

```
SELECT sale_date,amount FROM fact_sales;
```

QUERY RESULT

sale_date	amount
2025-10-02	300
2025-10-02	200
2025-10-03	400
2025-10-03	100

Answer: Daily Sales

```sql
SELECT sale_date,SUM(amount) AS total
FROM fact_sales
GROUP BY sale_date ORDER BY sale_date;
```

sale_date	total
2025-10-02	500
2025-10-03	500

Each result row is one date, not one sale

Answer: Chocolate Sales

```sql
SELECT sale_date,SUM(amount) AS total
FROM fact_sales WHERE product_id='P1'
GROUP BY sale_date ORDER BY sale_date;
```

sale_date	total
2025-10-02	300
2025-10-03	400

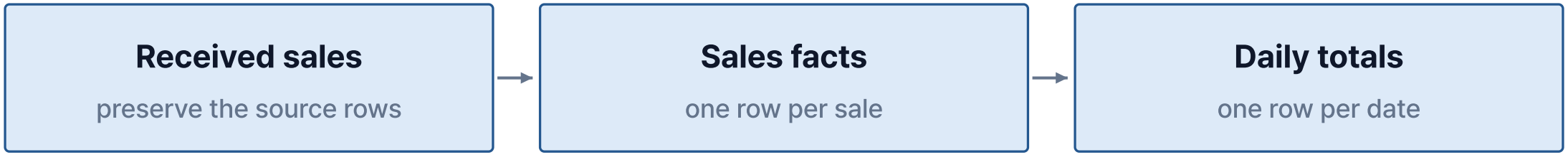
The filter changes which sales contribute to each total

03 · Refreshing Reports

01	Analytical Copies	keep reporting work separate from sales transactions
02	Rows for Analysis	connect each sale to its product and daily total
03	**Refreshing Reports**	refresh the copy in the right order without counting twice

From Received Data to a Report

Extract, transform and load connect the source to analytical tables



A pipeline turns received data into the table the report reads

Adding the New Date

date	already in report	new batch
October 2	500	—
October 3	—	500

October 3 contributes one new total of 500

Caveat: Counting a Batch Twice

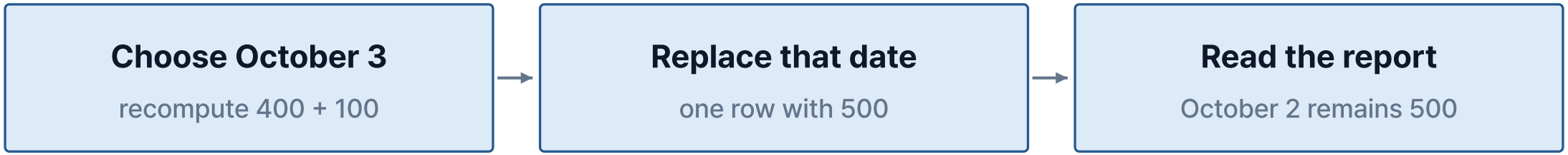
Successful SQL can still produce a wrong report



Repeating an append can double the same sales

Replace the Affected Date

A repeatable load preserves one correct row per date



Replacing the same date twice still leaves one total for that date

Demo: Repeat the Load

1. Run the supplied date replacement twice in one script
2. Check that both dates still have one total of 500

```
BEGIN;
DELETE FROM daily_sales
WHERE sale_date='2025-10-03';
INSERT INTO daily_sales
SELECT sale_date,SUM(amount)
FROM fact_sales
WHERE sale_date='2025-10-03'
GROUP BY sale_date;
COMMIT;
SELECT * FROM daily_sales ORDER BY sale_date;
```

QUERY RESULT

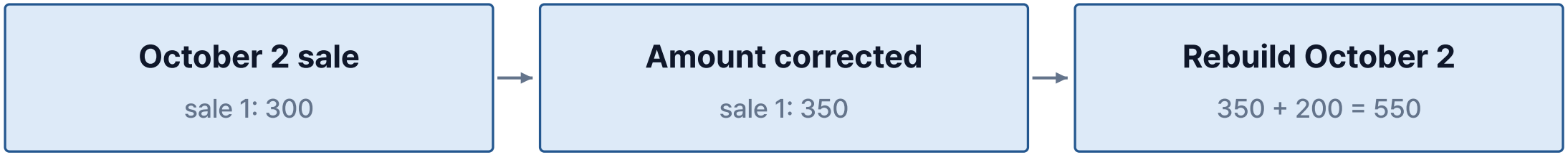
sale_date	total
2025-10-02	500
2025-10-03	500

Answer: One Total per Date

date	after one replacement	after two replacements
October 2	500	500
October 3	500	500

Both executions leave the same correct report

A Corrected Sale



A correction changes the affected date, not every date

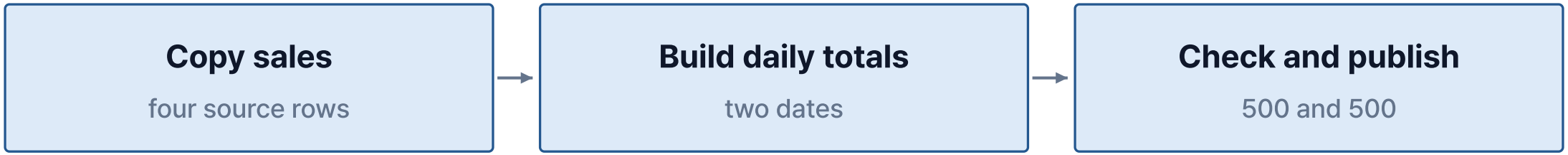
The Corrected Report

date	before correction	after refreshing October 2
October 2	500	550
October 3	500	500

Refreshing October 2 changes 500 to 550 and leaves October 3 unchanged

Run the Steps in Order

Orchestration manages dependencies between pipeline steps



The totals depend on copied sales; the report depends on checked totals

Takeaway: Data Warehousing

1. An **analytical copy** gives repeated reports a clear input
2. The **meaning of one row** determines how facts can be joined and summed
3. **Replace the affected date** and check the result so reruns do not double sales