SCORE A WHOLE COHORT WITHOUT EXPOSING HALF A RESULT

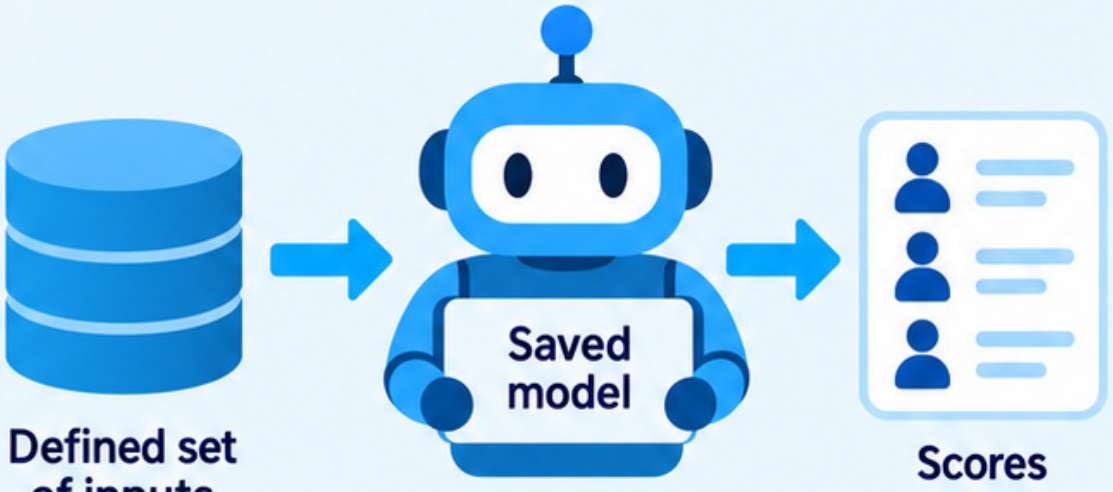
Batch inference

Some predictions are needed every morning, not for every click.

Today: design a nightly purchase-inactivity scoring job with pinned inputs, safe retries and a clear publication boundary.

BATCH INFERENCE SCORES A BOUNDED COLLECTION

A batch job applies a saved model to a defined set of inputs.
It does not fit model parameters.



Our example scores eligible customers once per night for a review queue.

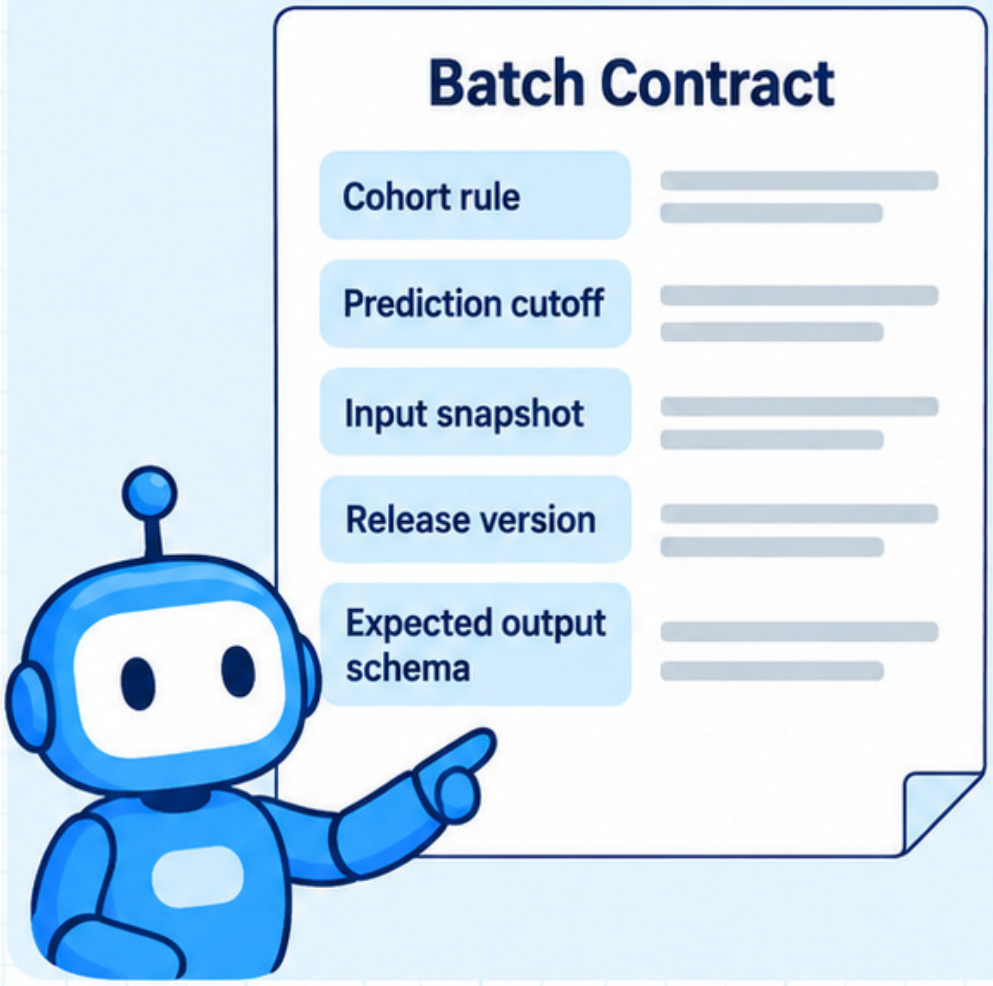
Define when results must be ready and how old they may be when used.

When results must be ready

How old they may be when used

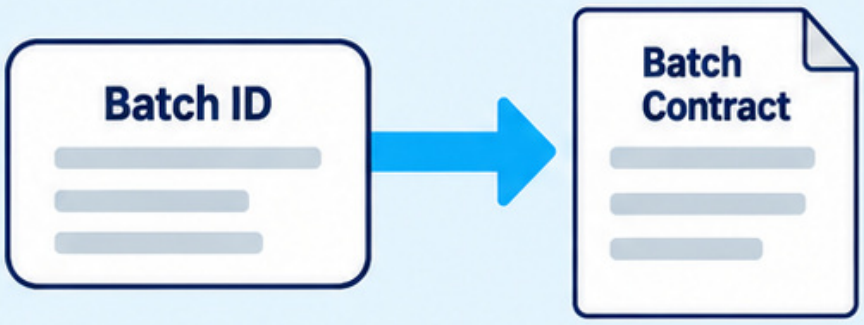
START WITH THE BATCH CONTRACT

Record the cohort rule, prediction cutoff, input snapshot, release version and expected output schema.



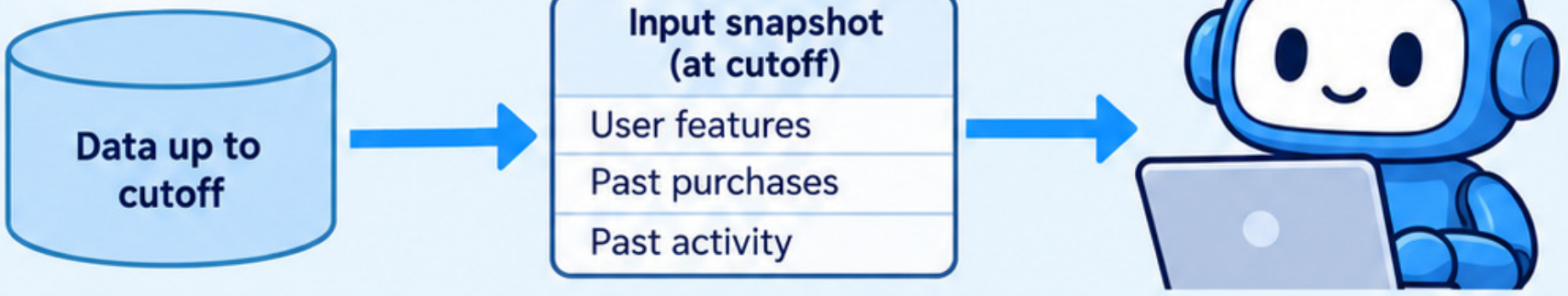
Add completion deadline, freshness limit, invalid-row policy and publication owner.

A batch ID must identify this fixed contract, not just a date string.

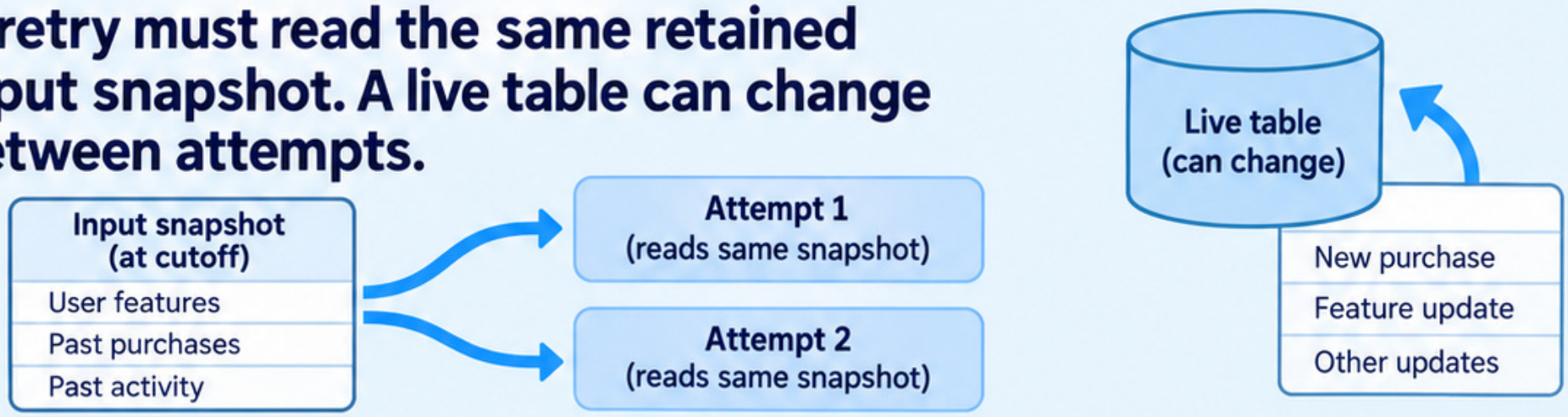


FREEZE WHAT WAS KNOWABLE AT THE CUTOFF

Use features available by the prediction cutoff, following Day 10 rules.



A retry must read the same retained input snapshot. A live table can change between attempts.



Do not include purchases or feature updates that became available after the cutoff.

RESOLVE THE RELEASE ONCE PER BATCH

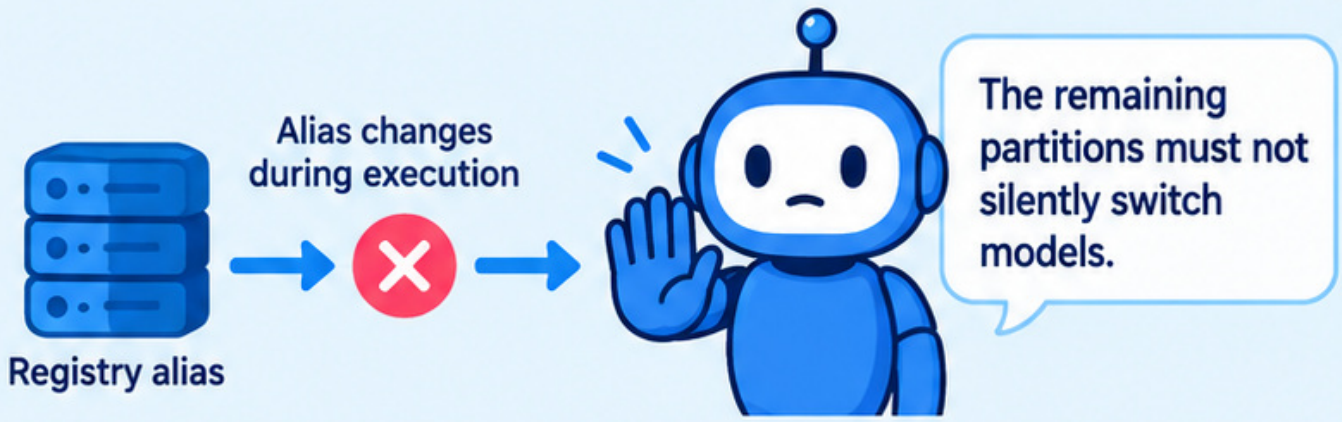
The batch manifest pins the fitted pipeline, feature contract and decision-policy version.

Batch Manifest

- Fitted pipeline version
- Feature contract version
- Decision-policy version

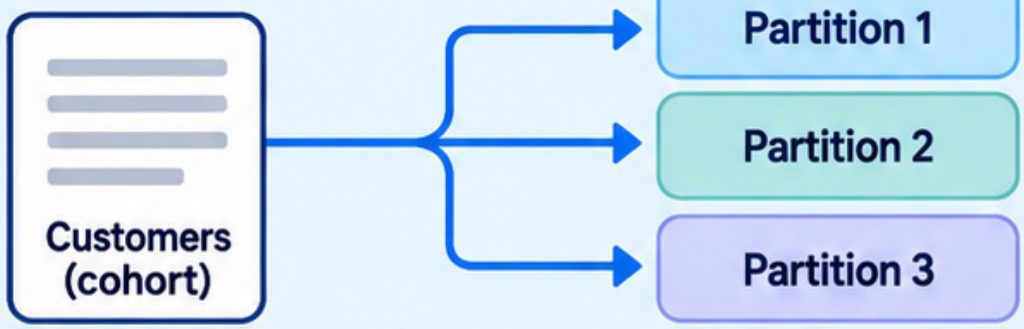
All partitions use that release.

If a registry alias changes during execution, the remaining partitions must not silently switch models.

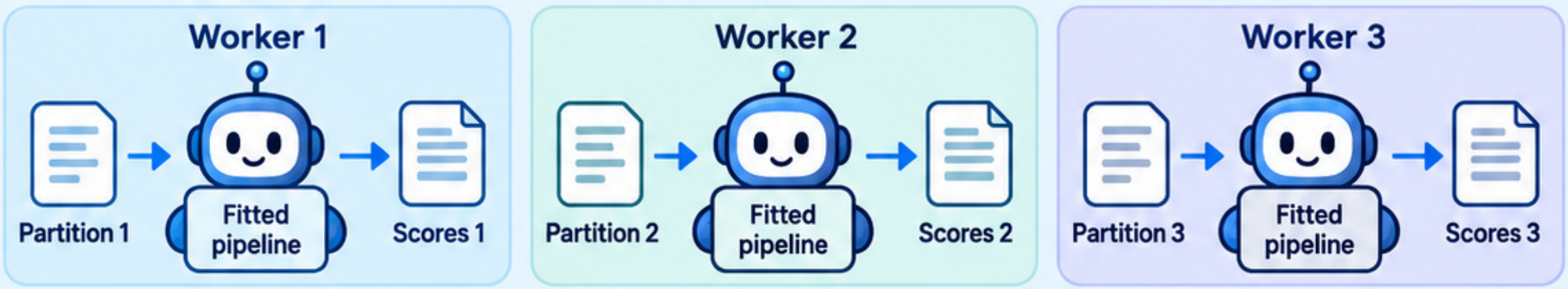


PARTITION WORK TO BOUND MEMORY AND RECOVERY

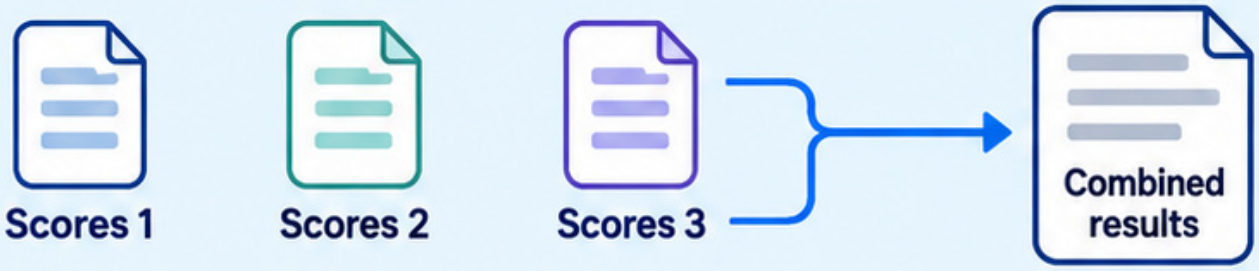
Split the fixed cohort into non-overlapping partitions with stable membership.



Load the fitted pipeline once per worker and score bounded chunks.



Record which partition owns each customer.
Check for missing or duplicate IDs when combining results.

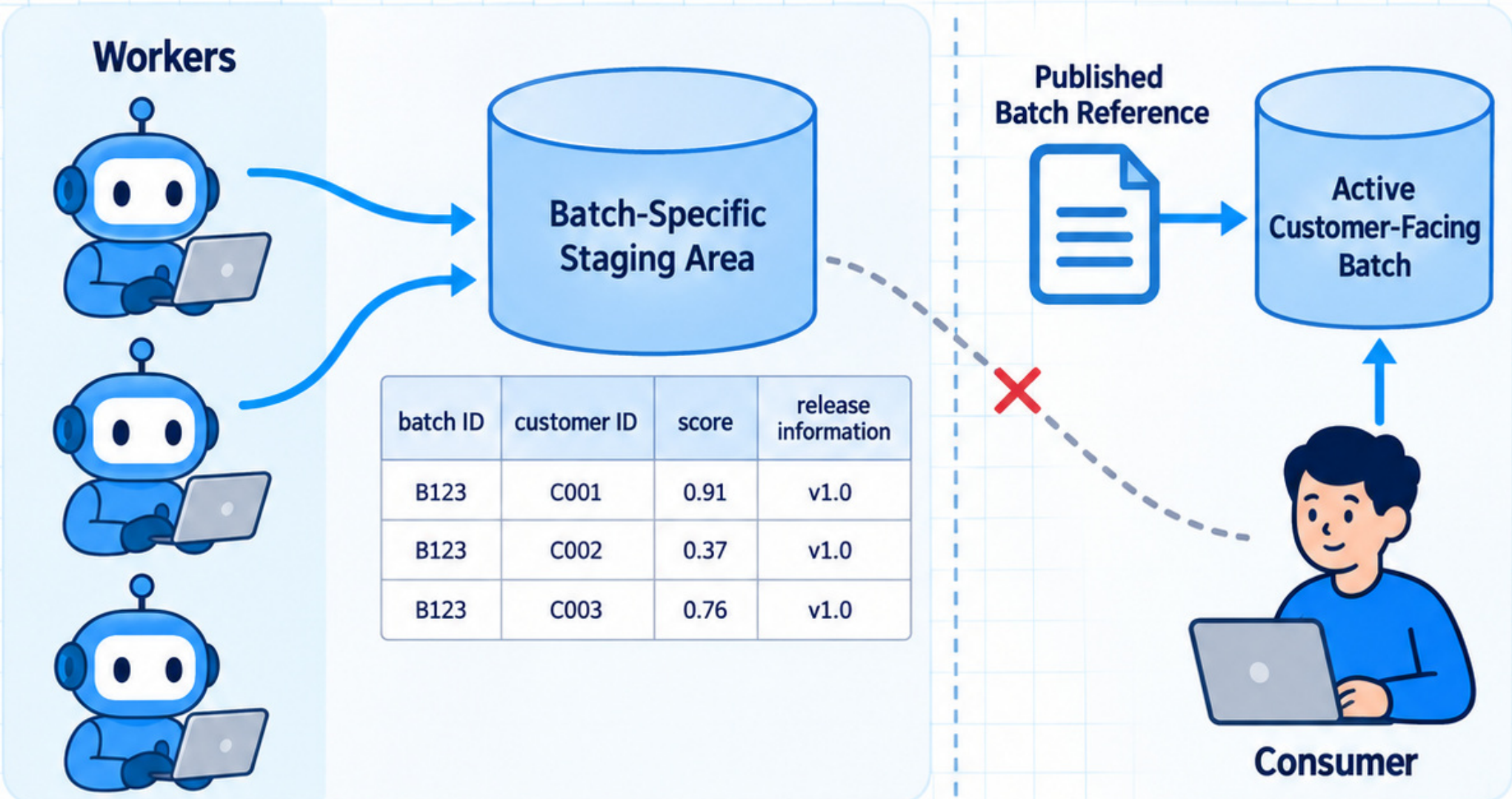


WRITE RESULTS TO STAGING FIRST

Workers write to a batch-specific staging area.

Each row carries batch ID, customer ID, score and relevant release information.

Staged rows are not the active customer-facing batch. Consumers read through the published batch reference.

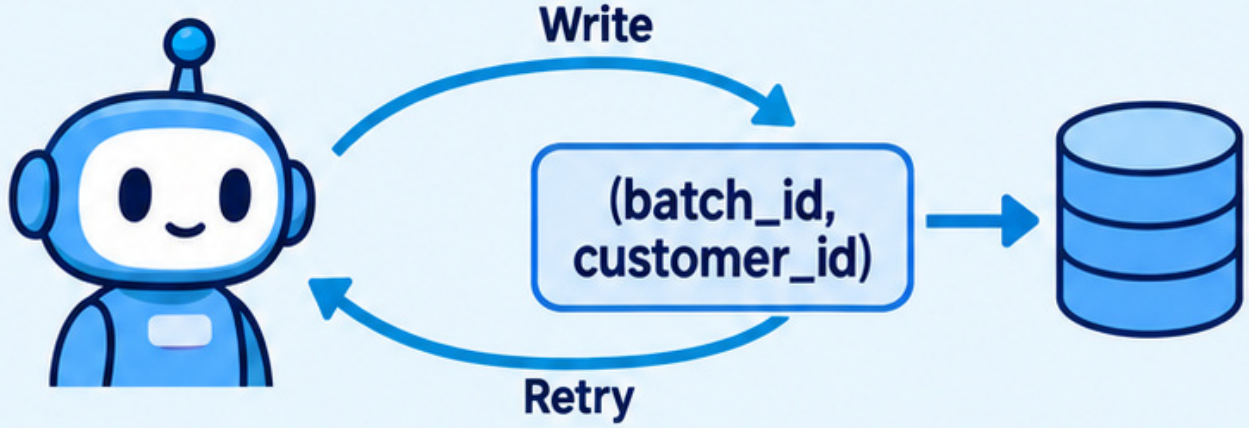


batch ID	customer ID	score	release information
B123	C001	0.91	v1.0
B123	C002	0.37	v1.0
B123	C003	0.76	v1.0

A RETRY MUST REPEAT THE SAME LOGICAL WRITE

Use a unique result key such as (batch_id, customer_id).

For the same immutable inputs and release, a repeated write must preserve the same logical result.



A conflicting result needs investigation or a new version. Do not silently replace it under the same identity.

HANDLE A CRASH AFTER THE WRITE SUCCEEDS

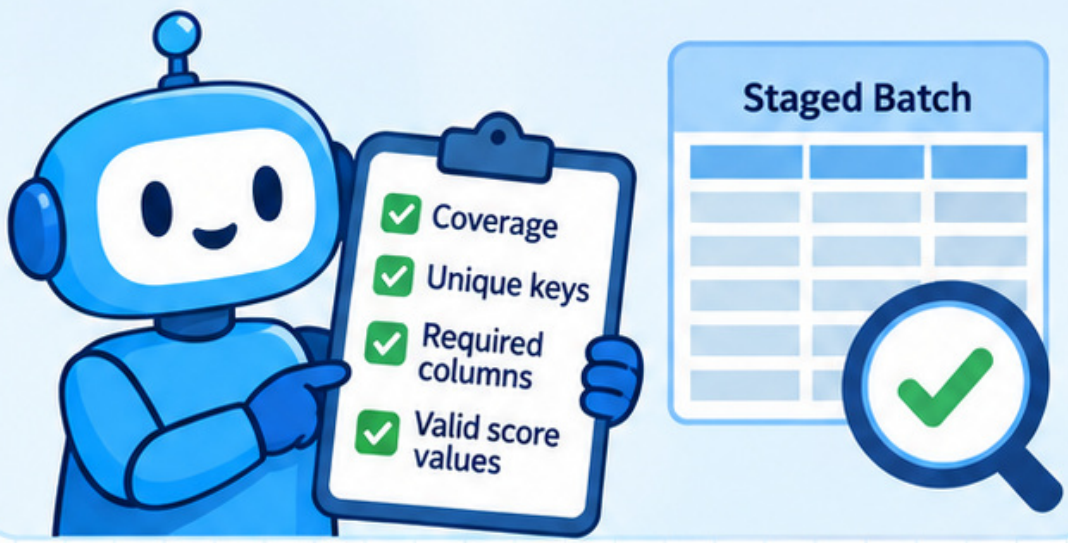
Toy sequence:

1. Worker writes a partition successfully.
2. It crashes before recording completion.
3. The scheduler retries the partition.
4. The retry reconciles existing rows using the same identity, then records completion.

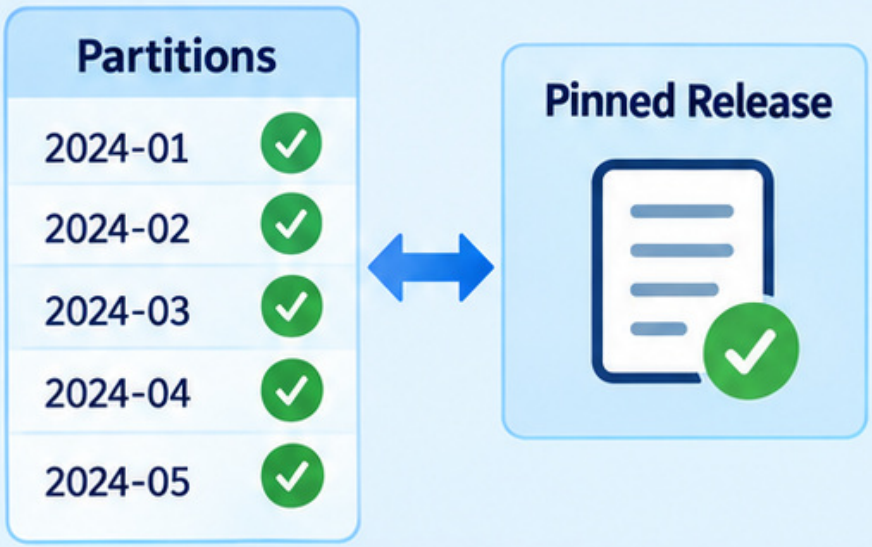
The retry reconciles existing rows using the same identity, then records completion. Acknowledgement loss must not create duplicate results.

PROVE THAT THE STAGED BATCH IS USABLE

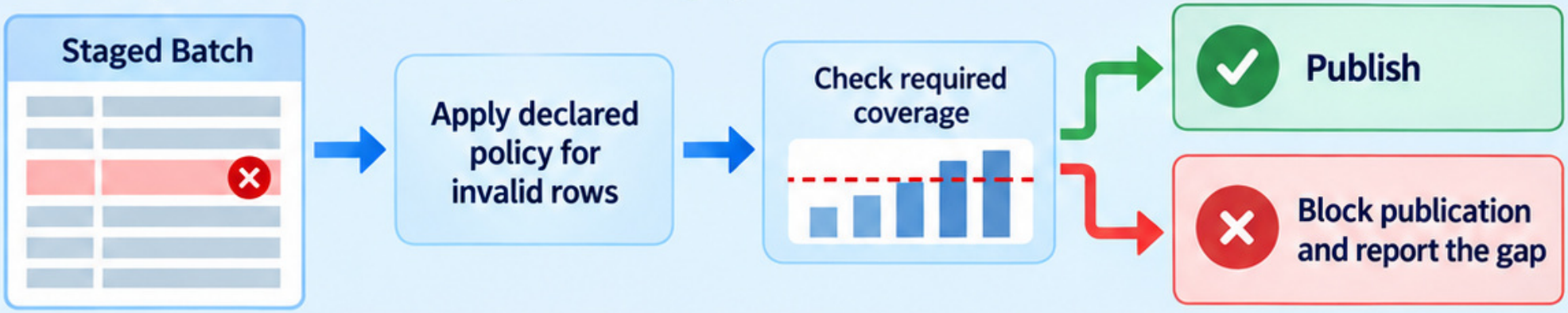
Check expected customer coverage, unique keys, required columns and valid score values.



Verify every required partition completed with the pinned release.

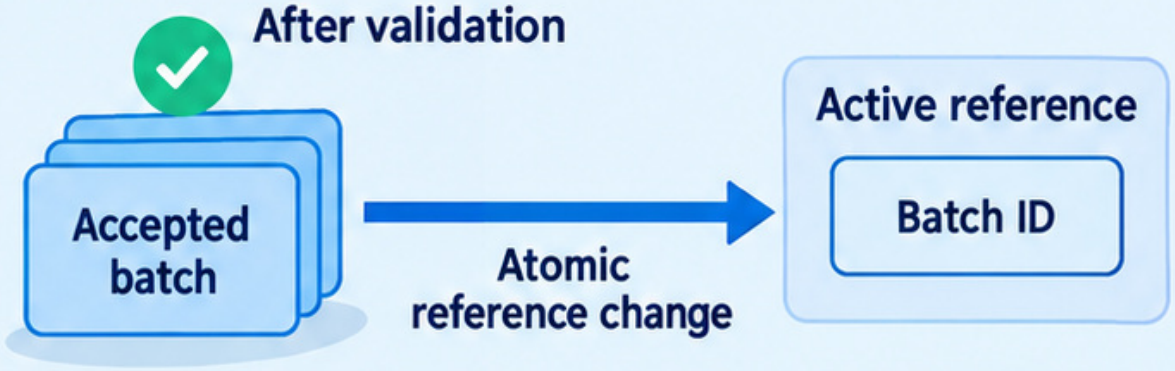


Apply the declared policy for invalid rows. If required coverage fails, block publication and report the gap.

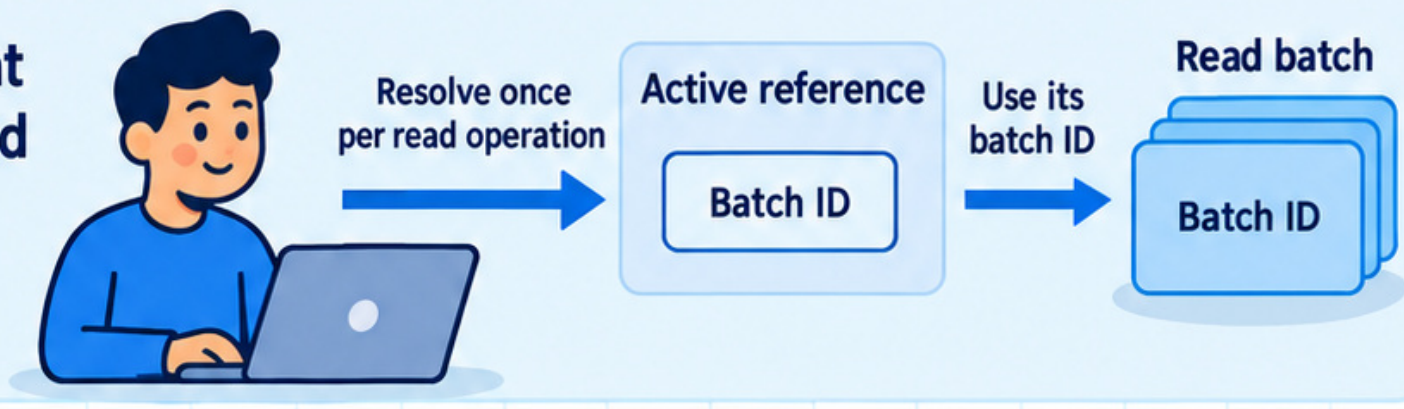


PUBLISH WITH ONE ATOMIC REFERENCE CHANGE

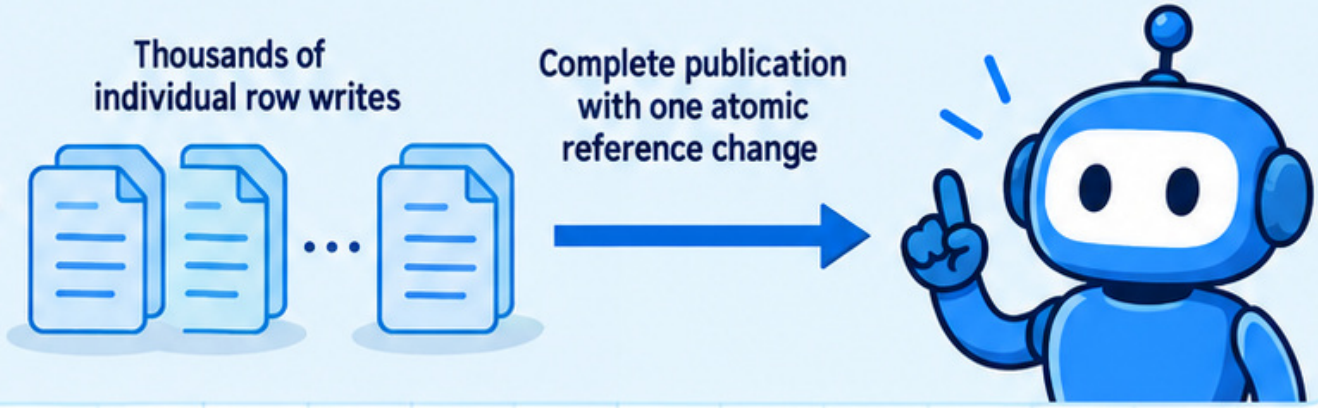
After validation, atomically move the active-batch reference to the accepted batch.



Consumers resolve that reference once per read operation and use its batch ID.

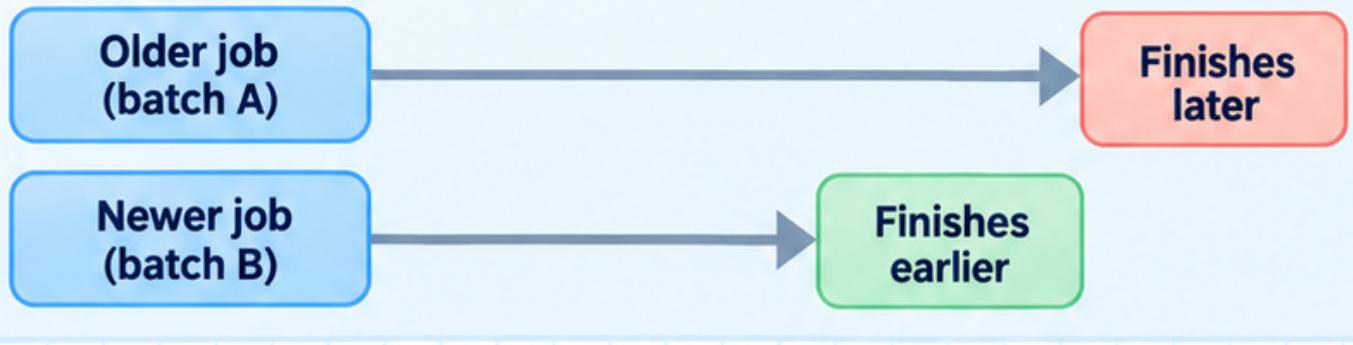


This separates complete publication from thousands of individual row writes.

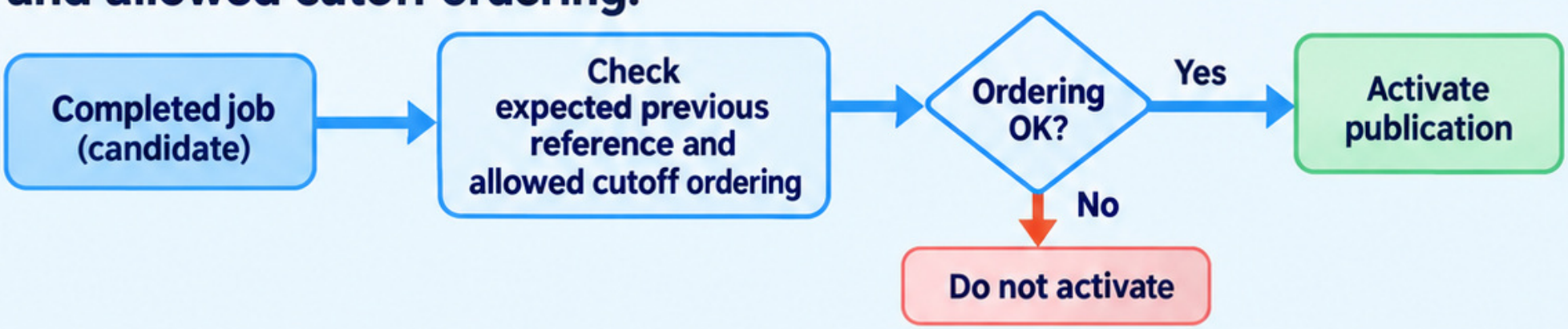


PROTECT PUBLICATION FROM AN OLDER JOB

Two jobs may finish out of order.



Before activation, check the expected previous reference and allowed cutoff ordering.



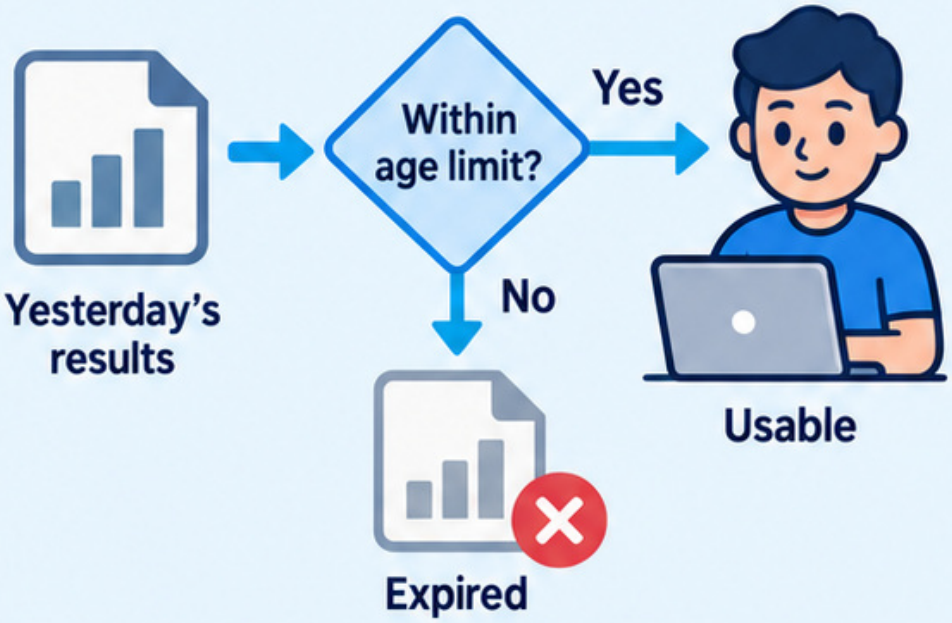
A late-finishing older batch must not silently replace a newer one. An intentional rollback needs its own explicit decision.

A SUCCESSFUL OLD BATCH CAN STILL BE UNUSABLE

Record the prediction cutoff, completion time and expiry policy.

- Prediction cutoff
- Completion time
- Expiry policy

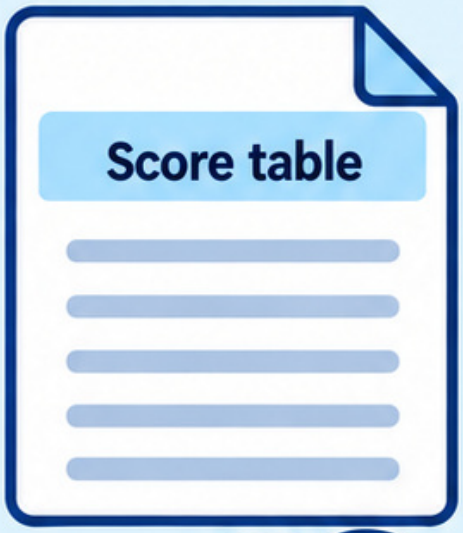
If tonight's job fails, yesterday's results are usable only while the declared age limit permits them.



After expiry, show unavailable or use an explicitly approved fallback.
Do not relabel old scores as fresh.

SELECT THE REVIEW QUEUE CONSISTENTLY

A score table is not yet the final queue.

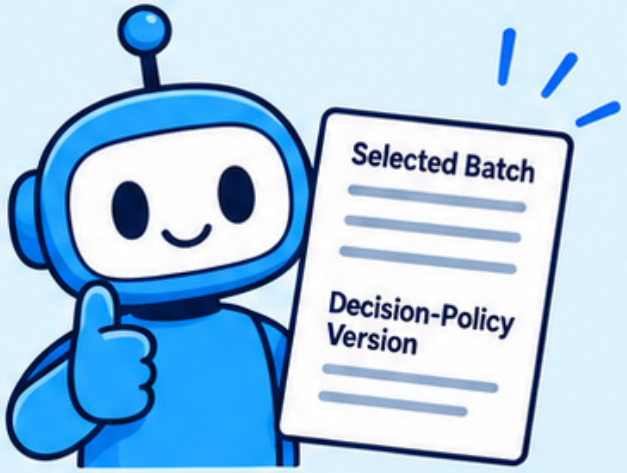


Apply the pinned threshold, capacity and tie policy to the accepted eligible cohort.

Selection Rule (versioned)

- Pinned threshold
- Capacity
- Tie policy

Record the selected batch and decision-policy version so the morning queue can be explained later.

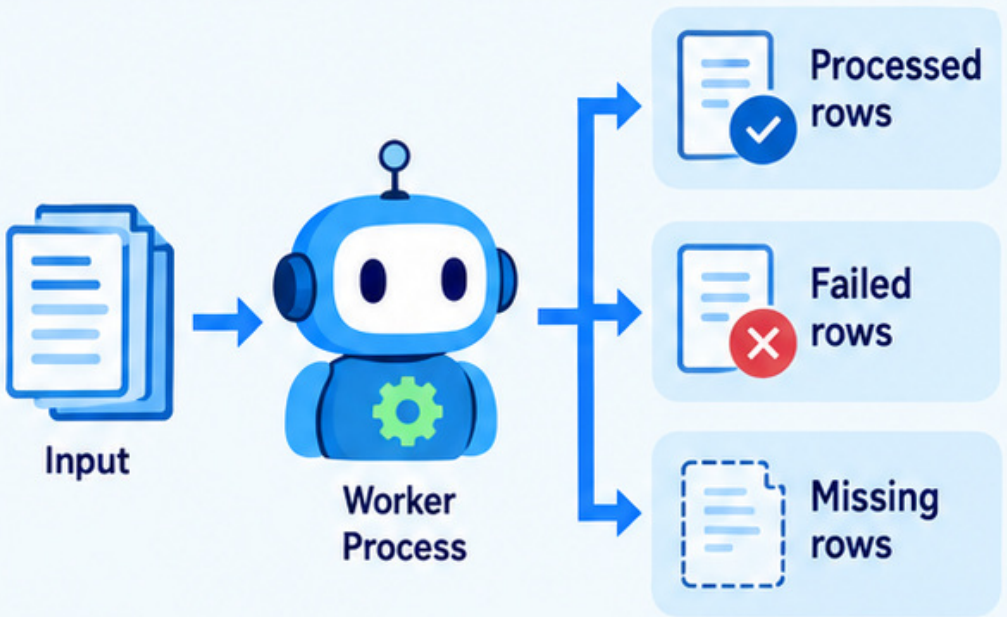


MONITOR COMPLETENESS AND TIMELINESS SEPARATELY

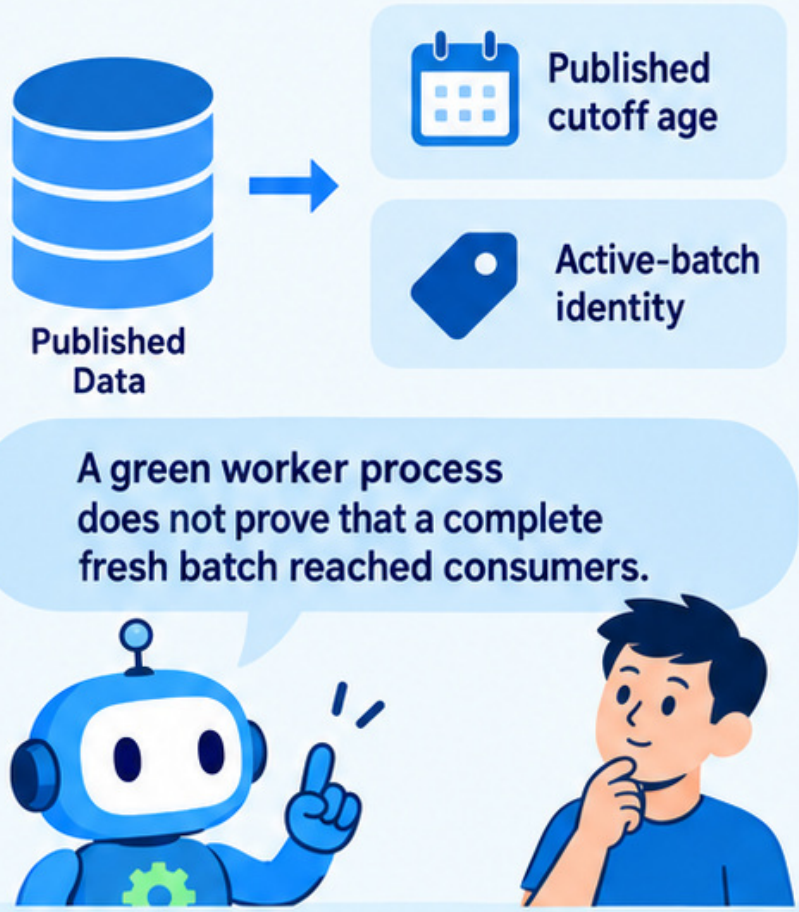
Track processed, failed and missing rows; partition retries; and total job duration.

Also track published cutoff age and the active-batch identity.

Job Execution

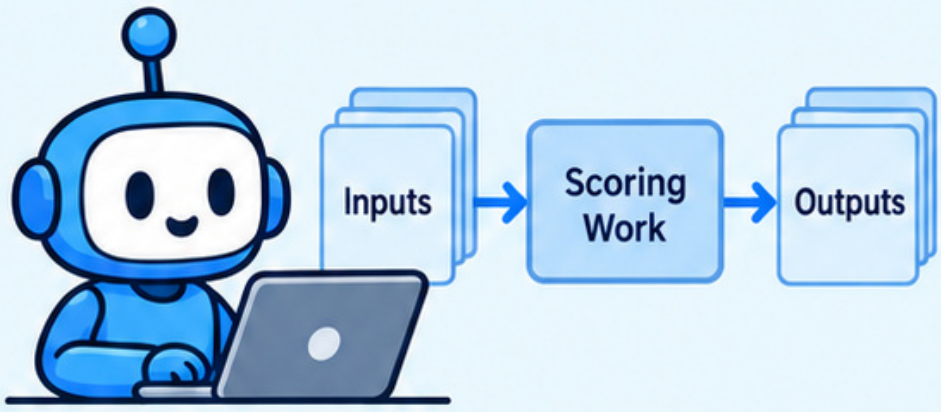


Published Result

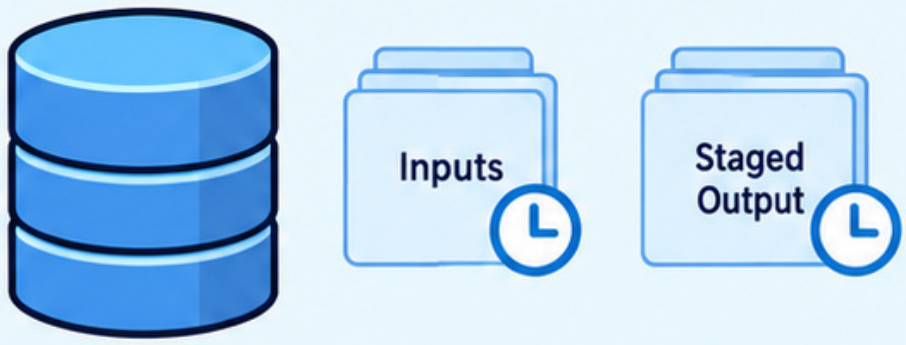


MATCH TOOLS TO THE BOUNDARIES

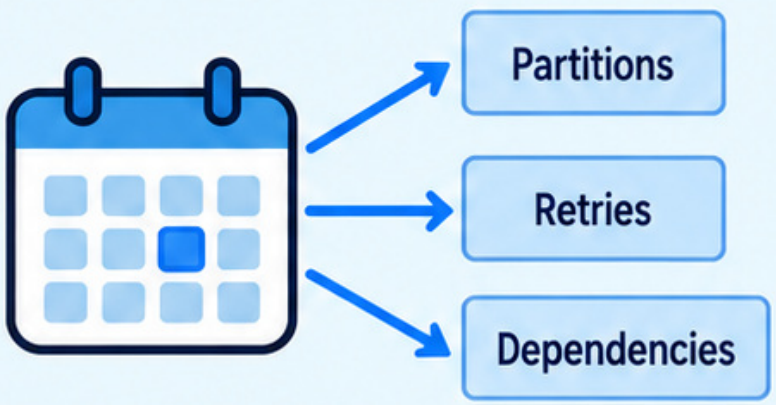
Python or a data engine: bounded scoring work.



Versioned storage: retained inputs and staged output.



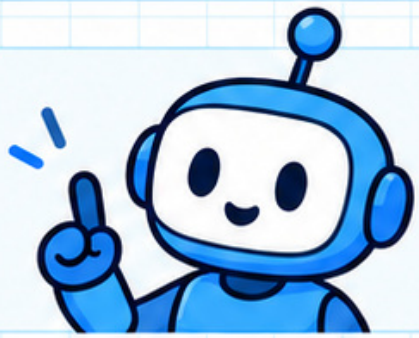
A scheduler: partitions, retries and dependencies.



PostgreSQL constraints and transactions: unique identities and conditional publication.

- Unique identities
- Conditional publication

An upsert alone does not define safe batch semantics.



YOUR TURN: DECIDE WHETHER TO PUBLISH

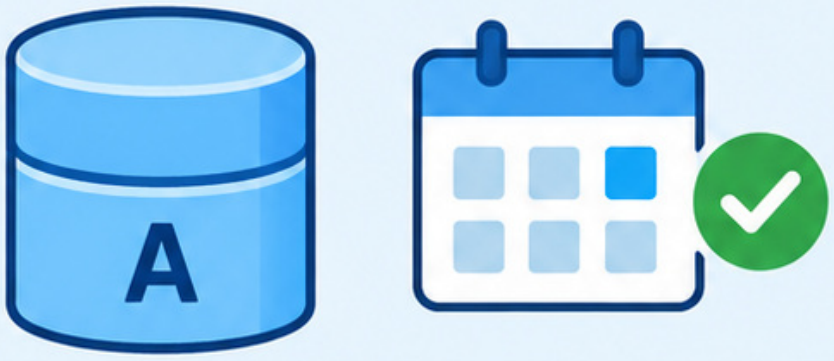
Toy batch B has three required partitions.

P1: complete.

P2: complete.

P3: failed.

The currently active batch A is still within its allowed age.



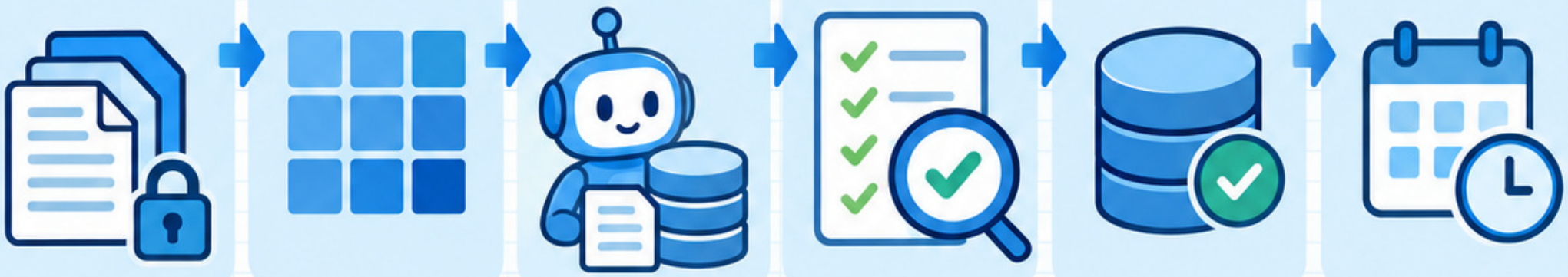
Should B become active?

What should the retry pin?

What changes if A expires before recovery?

A RELIABLE BATCH HAS A VISIBLE COMPLETION BOUNDARY

1. Pin inputs and release.
2. Partition consistently.
3. Write repeatably.
4. Validate the whole cohort.
5. Publish once.
6. Enforce freshness.



That is the contract behind a dependable morning score table.

DAY 14: Online inference.

Design the prediction path when a user is waiting.