Logical Replication is infrastructure

Gary Evans

Center of Excellence Manager

FUJITSU

Logical Replication is Infrastructure

FUJITSU

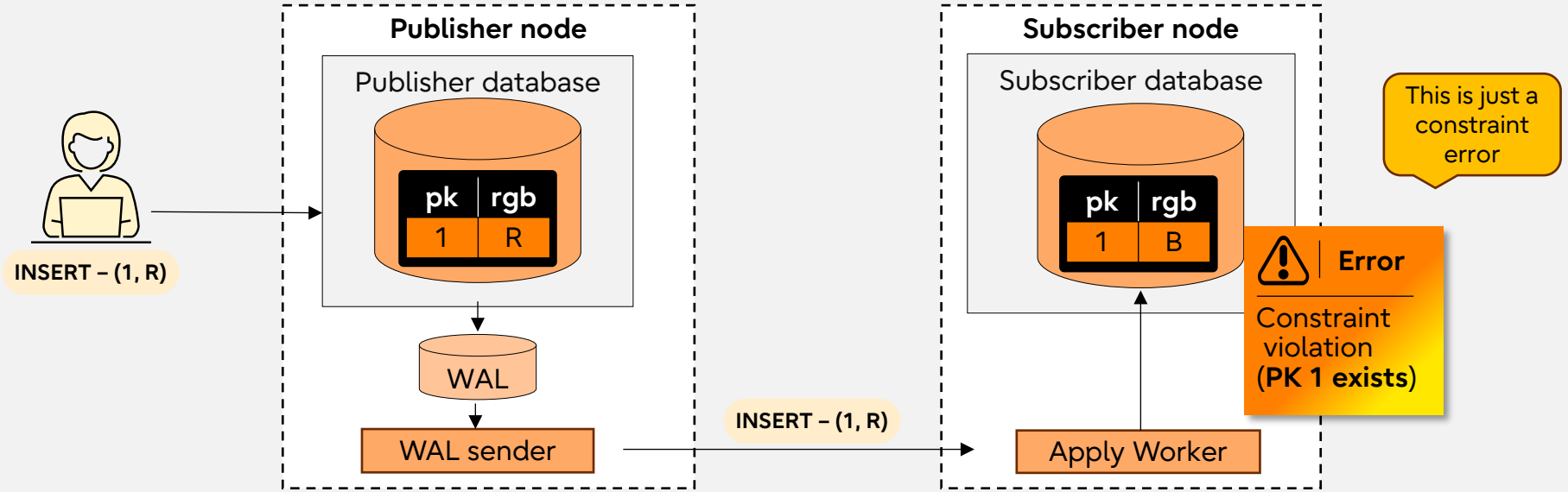
- **Blue/Green + upgrades**
  Cutover + rollback
- **Safe migrations**
  Sequencing + guardrails
- **Validation & reconciliation**
  Compare + prove correctness
- **Streaming / CDC**
  Logical decoding consumers: Debezium, Airbyte, Fivetran
- **Active/active replication**
  Origin tracking + conflict policy: pgactive / BDR-style)
- **App-level conflict resolution**
  Domain rules
- **Orchestration platforms**
  Automation + audit: DMS, pipeline engines

What we mean by *conflicts*

FUJITSU

A conflict conceptually occurs when incoming changes from the publisher disagree with the current state of data on the subscriber

PostgreSQL does not model this as a first-class object



Conflict handling before PostgreSQL 18

FUJITSU

- Insert conflicts surface as errors and **apply worker exits and is restarted**
- Apply worker becomes **stuck in error loop**
- Resolution requires **manual** intervention
- `pg_replication_origin_advance()` could be used to advance past conflicting transaction, but it is **difficult to figure out the LSN of conflicting transaction**
- Update and delete mismatches are not treated as conflicts and are **skipped by design**
- Error logs provide **minimal context about the underlying condition**
- No structured conflict classification or statistics

PostgreSQL 18: Making conflicts explicit

PostgreSQL 18 doesn’t change conflict behaviour, it changes visibility

FUJITSU

PostgreSQL 18 enhancements

PostgreSQL 18 does not add pgactive-style conflict handling

It does add classified logical-replication conflict events

These are named conflict classes used for:

- Logging
- Statistics
- Observability

They do not introduce conflict resolution

Most do not stop apply

Conceptual conflict names *vs* core PostgreSQL behaviour

FUJITSU

insert_exists	Duplicate key / Unique Key Violation on apply
update_exists	Silent overwrite, or constraint error if a uniqueness rule is violated
update_origin_differs	Last writer wins, no error, origin not tracked
update_missing	Could not find row error on apply
update_missing Row deleted locally	Could not find row error on apply
delete_origin_differs	Delete applies normally, origin not tracked(**check this**)
multiple_unique_conflicts	Multiple UNIQUE constraint violations during apply

PostgreSQL 18 – Conflict visibility without policy

FUJITSU

What PostgreSQL 18 adds

- Explicit conflict classification
  - Named conflict classes (*e.g.*, insert_exists, update_origin_differs)
- Subscription-level statistics
  - Aggregated counters for hard (apply stopping) and soft (apply continuing) conflicts
- Structured logging
  - Conflict type, relation, subscription context

What PostgreSQL 18 does not change

- No conflict resolution
- No origin enforcement
- No change to apply semantics
- No automatic retries or rollbacks

PostgreSQL 18 – Where conflict information lives

FUJITSU

Statistics views (telemetry)

- Subscription-level counters
- Aggregated totals
  (since last reset / restart)
- Good for monitoring and alerting
- Not an audit trail

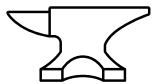
Server logs (forensics)

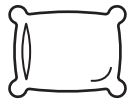
- One log entry per conflict event
- Includes:
  - Conflict class
  - Subscription
  - Relation
  - Action (INSERT / UPDATE / DELETE)
  - Whether apply continued or stopped
- This is the history

Hard versus Soft Conflicts

FUJITSU

Conflict class	Recorded	Logged	Stops apply	
insert_exists	✓	✓	✓	Hard
update_missing	✓	✓	✓	
delete_missing	✓	✓	✓	
update_exists	✓	✓	✗	Soft
update_origin_differs	✓	✓	✗	
delete_origin_differs	✓	✓	✗	





Logical Replication demo

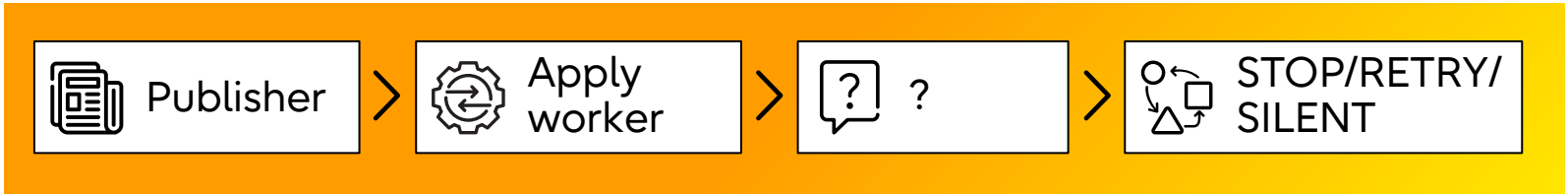
FUJITSU

What's missing today

Conflict detection exists

Conflict classification exists

What's missing is a structured escalation point



From PostgreSQL 18 visibility to PostgreSQL 19+ capability

PostgreSQL 18: Conflicts are detected, classified, and observable with no change to apply behaviour or policy

What's missing: A structured way to respond without crashing or looping

Next step: Make conflict handling mechanically safe without making it policy-driven

FUJITSU

Conflict handling boundaries

FUJITSU

Does do

- Detects conflicts
- Classifies conflict type
- Enumerates valid resolution actions
- Guarantees correctness of apply mechanics

Doesn't do

- Does not choose a resolution
- Does not encode business rules
- Does not enforce *last writer wins* or similar policies
- Does not automate outcomes

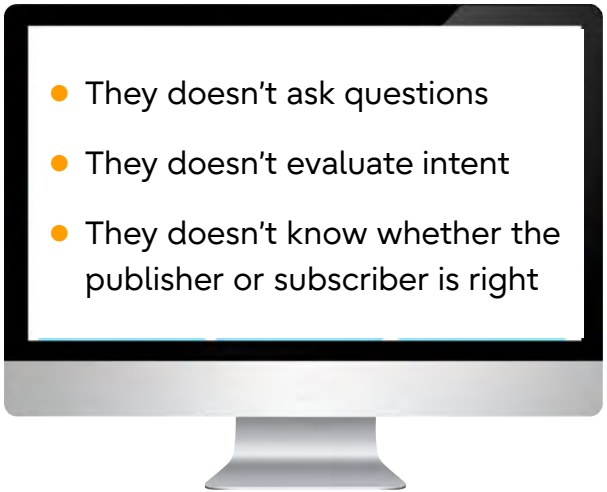
Conflict	Possible actions exposed
insert_exists	skip, retry, error
update_origin_differs	apply, ignore, defer
delete_missing	ignore, error
multiple_unique_conflicts	retry with ordering, error

Conflict resolvers belong below the decision layer

FUJITSU

Conflict resolvers are execution strategies, not decision logic

They define how a conflict is applied once a higher-level system has already decided what should happen



From a presentation describing future enhancements in Logical Replication

Resolver	Description	Applicable conflicts
apply_remote	The remote change is applied	• insert_exists • update_exists • multiple_unique_conflicts • update_origin_differs • delete_origin_differs
apply_or_skip	Remote change is converted to INSERT and is applied. If the complete row cannot be constructed from the info provided by the publisher, then the change is skipped	• update_missing • update_deleted
apply_or_error	Similar to above but raise an ERROR if cannot convert the remote change to INSERT	• update_missing • update_deleted
keep_local	The local version of row is retained	• insert_exists • update_exists • multiple_unique_conflicts • update_origin_differs • delete_origin_differs
last_update_wins	The change with later commit timestamp wins.	• insert_exists • update_exists • update_origin_differs • delete_origin_differs • update_deleted
error	Replication is stopped; manual action is needed.	Can be used for any conflict type.

Why this matters for the PostgreSQL ecosystem

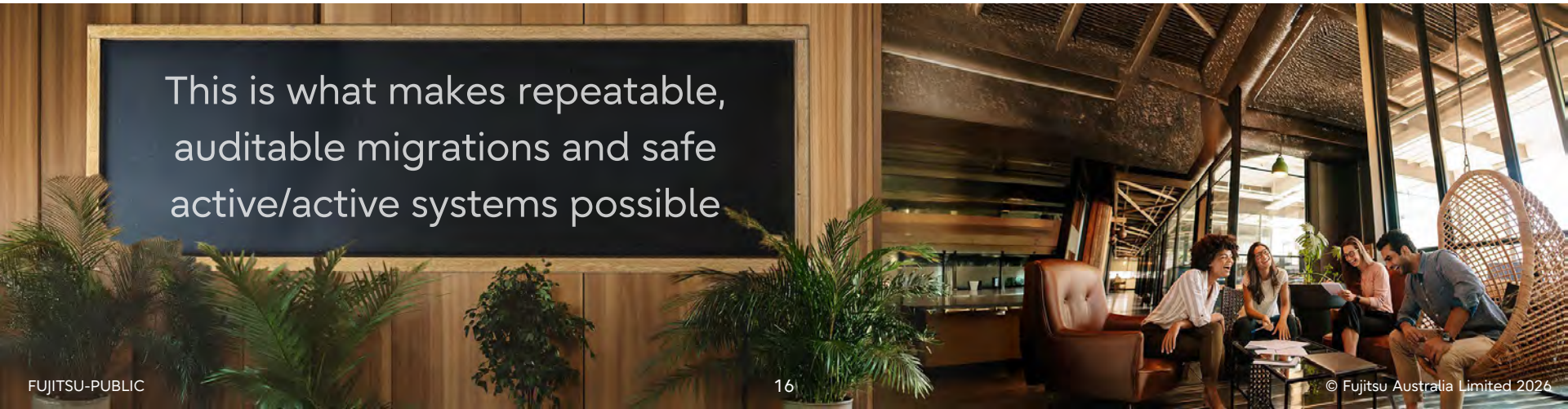
FUJITSU

Tooling stability

- Tool authors stop re-parsing logs
- No more brittle error-string matching

System design

- Deterministic, testable resolution flows
- One apply worker with many policies



Logical Replication is a platform, not a product

FUJITSU

What Logical Replication provides

- General-purpose change transport
- Ordered, row-level delivery
- Selective publication, asynchronous apply

What PostgreSQL adds over time

- General-purpose change transport
- PG18: Conflict detection, classification, observability
- PG19/20: Mechanisms for safer conflict handling (planned)

What remains external (by design)

- Policy and intent
- Business rules
- Automation and orchestration

What this enables

- Safe, repeatable migrations
- Blue/green and version coexistence
- Active/active architectures
- CDC and downstream integration

Thank you

Logical Replication is infrastructure

Gary Evans

Center of Excellence Manager

FUJITSU