DATA QUALITY: WAP PATTERN

How is bad data generated?

1 LOGGING ERRORS

Occur during data capture & recording; FACT data-related

Missing event captures

App crash before logging completion, network interruptions, rate limiting causing event drops

Incorrect timestamps

Time zone handling errors, race conditions in event sequencing, batch processing delays causing temporal inconsistencies

Malformed event data

Character encoding issues, truncated/ corrupt log issues

Duplicate events

Redudant loggin config, failed deduplication issues

2 SNAPSHOOTING ERRORS

Occur when capturing the state of a system at a specific point in time. Dimensional data-related (rare)

Consistency issues

Partial captures during state changes, distributed systems coordination failures

Timing issues

Incorrect snapshot scheduling, missed windows, overlapping snapshot attempts

Resource constraints

Insufficient memory, storage limitations

State representation issues

3 PRODUCTION DQ ISSUES

Occur during active data processing production environments when real user data & production loads interact with your systems

Data source issues

Upstream systems sending bad data, inconsistent data formats from sources, missing required fields, uncommunicated business logic changes, ...

Environmental factors

Production load affecting data processing, resource constraints, performance optimization shortcuts leading to corrupted data, production-specific cases not caught during testing.

4 SCHEMA EVOLUTION ISSUES

Occur during Db/ Schema updates, migrations or changes on data models impacting existing data structures

Breaking changes

Adding required fields w/o default values,
Removing columns still in use by downstream processes
Type changes incompatible w/ existing data
Renaming fields w/o proper migration strategy

Version management

Multiple schema versions running simultaneously
Incomplete schema migration rollouts
Backward compatibility issues
Missing schema documentation
Inconsistent schema enforcement across systems

5 PIPELINE ISSUES INTO PROD

Occur during deployment of new/ updated data pipelines

Testing gaps

Insufficient testing of edge cases
Lack of integration testing
Missing validation of business rules
Incomplete data quality checks

Deployment issues

Configuration differences between environments
Insufficient rollback procedures
Incomplete dependency management
Missing monitoring and alerting

6 NON-IDEMPOTENCY / BACKFILLING

Occur during data reprocessing attempts, historical data backfills

Processing problems

Multiple runs producing different results
State dependency issues
Temporal coupling in processing logic
Incomplete clean-up between runs

Backfill challenges

Incorrect handling of historical data
Time-based processing assumptions
Missing data version control
Incomplete statement management during reprocessing

7 NON THOROUGH VALIDATION

Validation Scope issues

Surface-level checks missing deep data issues, insufficient cross-field validation, missing business rule validation, incomplete reference data checks

Process Gaps

Lack of automated validation; Missing data profiling; Insufficient anomaly detection; Incomplete data lineage tracking; Poor error handling & reporting

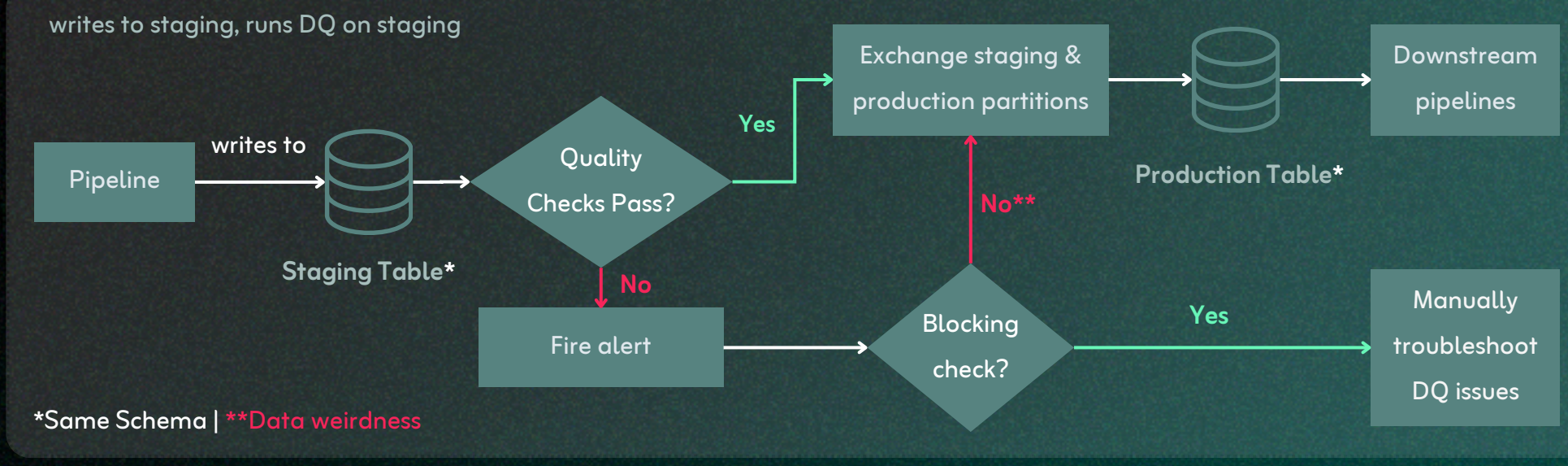
TYPES OF DATA QUALITY CONTRACTS

Writing in Production is a Contract!

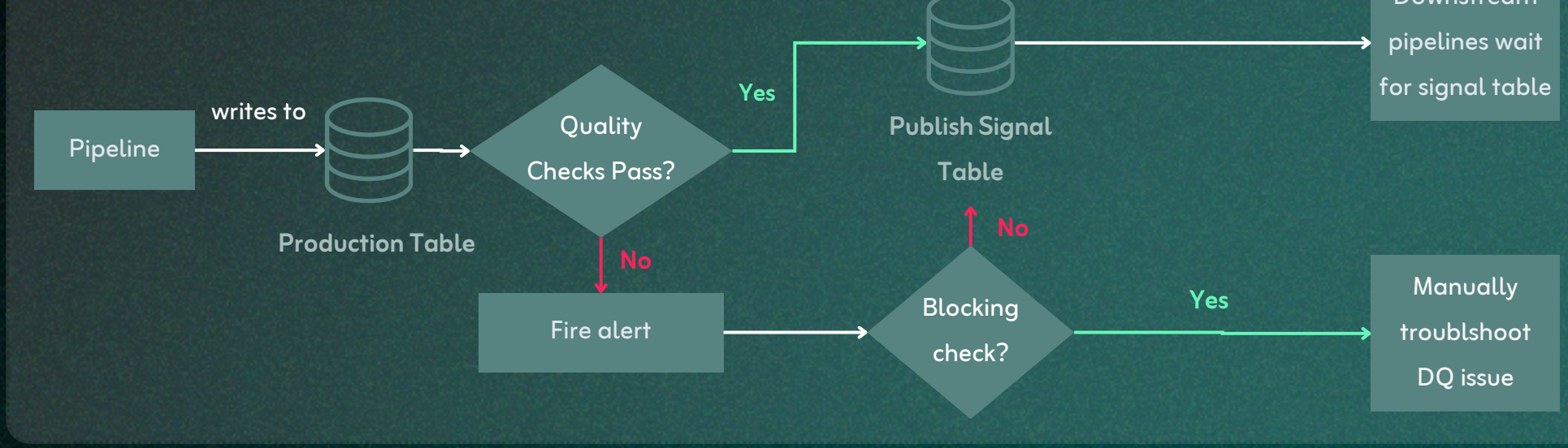
3 components of a contract: 1. Schema (columns, data types,...); 2. Quality checks; 3. Data path to production

1 WAP (Write-Audit-Publish) NETFLIX

Writes data into staging table, validates DQ checks upon staging prior to publishing data into production



2 SIGNAL Table



	PROS	CONS
WAP	• Downsteam pipelines can intuitively depend on the production table directly. • No chance of production data getting written prior to passing the DQ checks	• Partition exchange may cause pipeline delays by several minutes. More likely to miss SLA.
SIGNAL	• Simpler pattern • Data lands in 1 spot. Never has to be moved. • Greater chance of hitting SLA, data lands sooner. Uses less compute, less I/O	• Signal table is a Non-intuitive design for downstream users. • Likelihood of propagating DQ issues (bad data) • Does not cater for Ad-hoc queries well