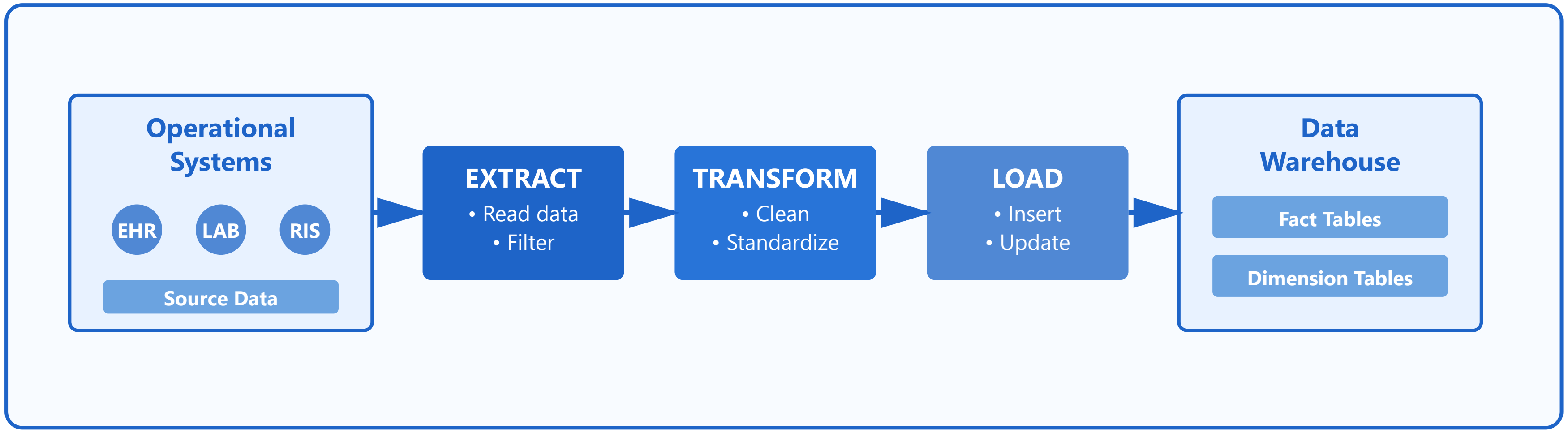
Data Warehousing for EHR



📥 ETL Processes

- Extract from operational systems
- Transform & clean data
- Load into warehouse
- Incremental updates

📊 Data Marts

- Disease-specific repositories
- Quality improvement data
- Research cohorts
- Departmental analytics

⭐ Star Schema

- Fact tables (encounters, labs)

⏱️ Real-time vs Batch

- Batch: overnight processing

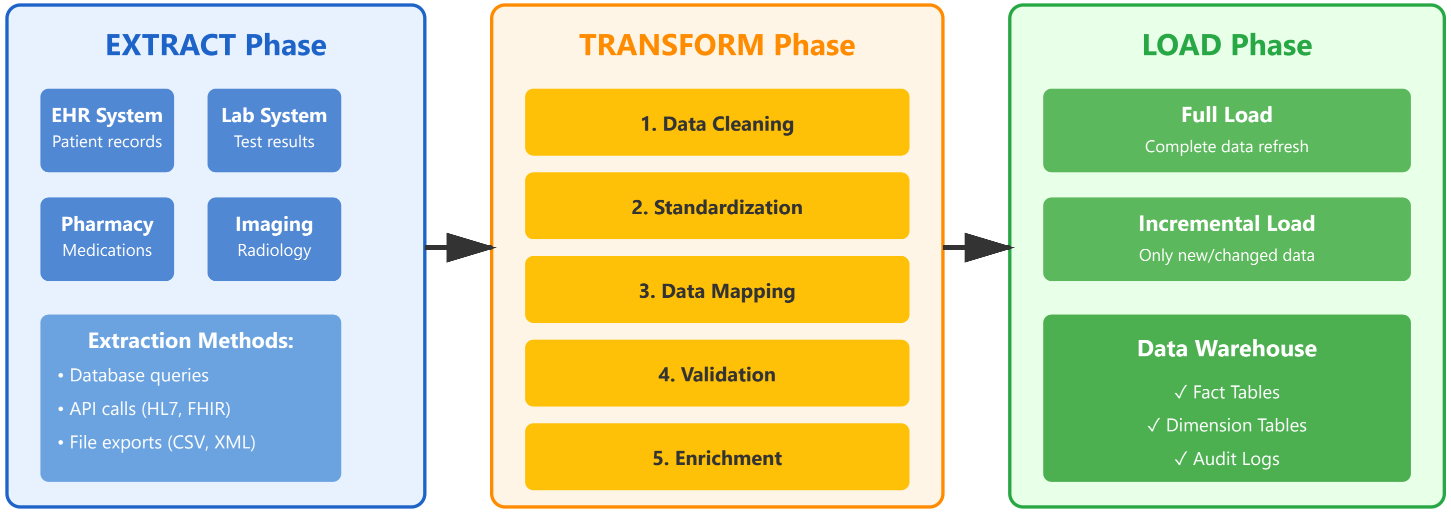
- Dimension tables (patient, time)
- Optimized for queries
- Aggregate calculations

- Real-time: streaming analytics
- Near real-time: micro-batching
- Trade-offs in complexity

📥 ETL Processes: Detailed Explanation

Overview

ETL (Extract, Transform, Load) is the backbone of any healthcare data warehouse. This process enables healthcare organizations to consolidate data from multiple disparate systems into a unified repository for analysis, reporting, and decision-making. In the EHR context, ETL processes must handle complex medical data structures, maintain data integrity, and ensure compliance with healthcare regulations like HIPAA.



1. Extract Phase

The extraction phase involves reading data from various operational healthcare systems. These systems often use different data formats, structures, and standards, making extraction challenging.

Example: Extracting Patient Lab Results

A hospital needs to extract lab results from their Laboratory Information System (LIS). The ETL process connects to the LIS database every night at 2 AM, queries for all lab results from the past 24 hours, and extracts records including patient ID, test type, result value, units, reference ranges, and timestamps. This data is temporarily stored in a staging area before transformation.

2. Transform Phase

The transformation phase is where raw data is cleaned, standardized, and prepared for analysis. This is the most complex phase in healthcare ETL due to the variety of medical terminologies and data quality issues.

Example: Standardizing Diagnosis Codes

Different hospital departments may record diabetes using various codes: "DM Type 2", "Diabetes Mellitus", "E11.9", etc. The transformation process maps all these variations to the standard ICD-10 code "E11.9" (Type 2 diabetes mellitus without complications). Additionally, it validates that the code is current and handles deprecated codes by mapping them to their current equivalents.

3. Load Phase

The loading phase inserts the transformed data into the data warehouse. This phase must handle data conflicts, maintain referential integrity, and track data lineage for audit purposes.

Example: Incremental Load Strategy

Rather than reloading all patient data daily, the ETL process tracks the last update timestamp. At 3 AM, it loads only patients whose records have been modified since the last ETL run. If Patient ID 12345 had a new lab result at 4 PM yesterday, only that patient's updated information is loaded, improving efficiency and reducing warehouse processing time from 6 hours to 45 minutes.

🔑 Key Considerations for Healthcare ETL

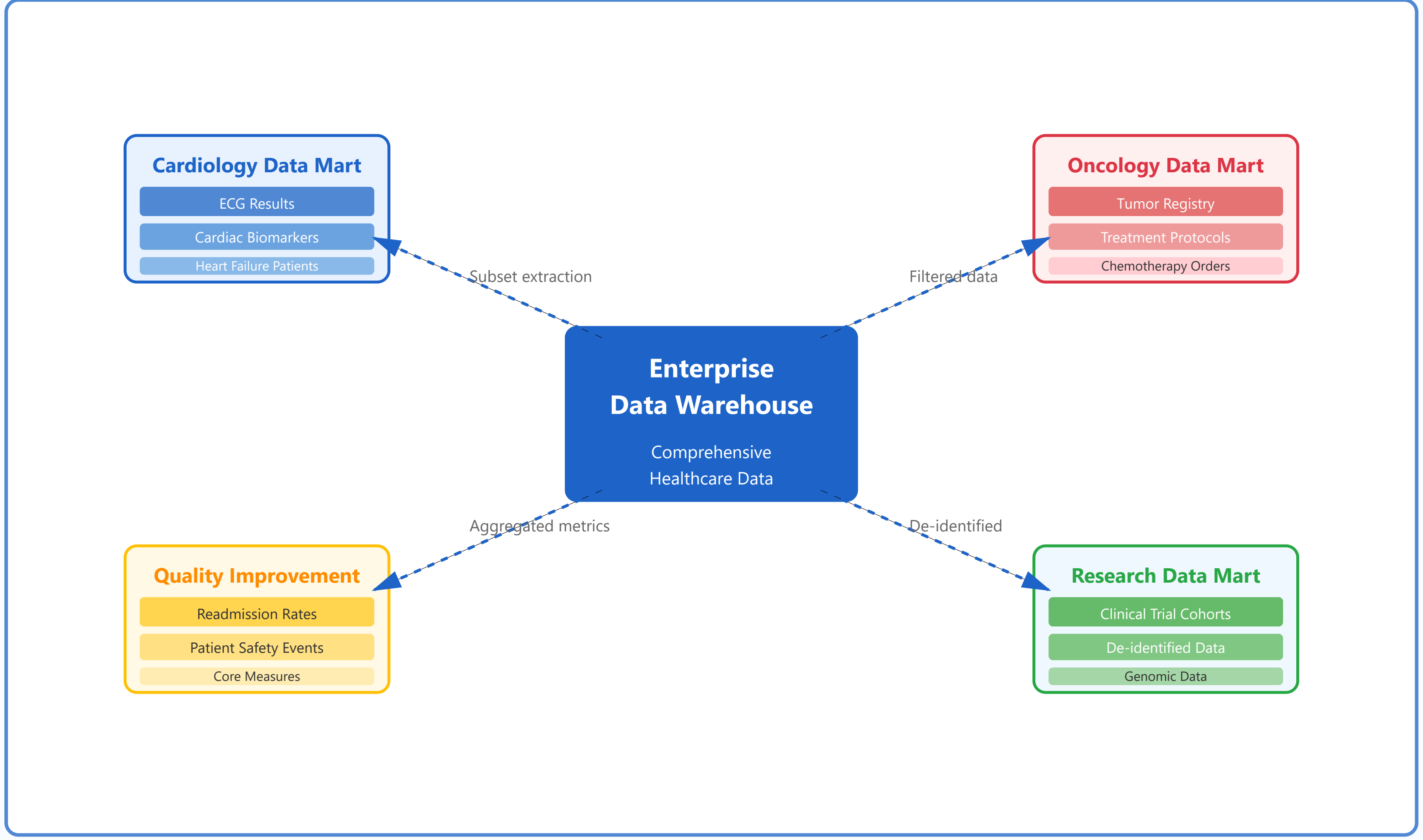
- HIPAA compliance requires encryption during extraction and loading
- Error handling must preserve data integrity without losing critical clinical information
- Audit trails track every transformation for regulatory compliance
- Performance optimization is crucial as healthcare data volumes grow exponentially

- Change Data Capture (CDC) techniques minimize impact on operational systems

📊 Data Marts: Detailed Explanation

Overview

A data mart is a subset of the data warehouse focused on a specific business line, department, or subject area. In healthcare, data marts enable specialized analysis for clinical departments, research teams, or quality improvement initiatives. They provide faster query performance and simplified data structures tailored to specific user needs.



Types of Healthcare Data Marts

1. Disease-Specific Data Marts

Diabetes Data Mart Example: Contains all patients with diabetes diagnoses (ICD-10: E08-E13), their HbA1c lab results over time, medication history (insulin, metformin, etc.), complication records (retinopathy, nephropathy), and outcomes data. This mart enables endocrinologists to track diabetic population health, identify patients needing intervention, and measure quality metrics like percentage of patients with HbA1c below 7%.

2. Departmental Data Marts

Emergency Department (ED) Analytics Mart: Focuses on ED-specific metrics including door-to-doctor time, length of stay, admission rates, left-without-being-seen rates, and resource utilization. The mart includes real-time dashboards showing current ED census, wait times by severity level, and staffing adequacy. This enables ED directors to optimize patient flow and resource allocation.

3. Quality Improvement Data Marts

Patient Safety Mart: Aggregates data on adverse events, medication errors, hospital-acquired infections, falls, and pressure ulcers. It tracks near-misses and includes root cause analysis data. Quality teams use this mart to identify trends, benchmark against national standards, and target improvement initiatives. For example, tracking central line-associated bloodstream infections (CLASI) rates across ICUs.

4. Research Data Marts

COVID-19 Research Mart: During the pandemic, many hospitals created specialized marts containing de-identified patient data including demographics, comorbidities, treatments, lab values (D-dimer, CRP, ferritin), ventilation duration, and outcomes. Researchers used this to identify risk factors, evaluate treatment effectiveness, and predict patient deterioration using machine learning models.

🔑 Benefits of Data Marts in Healthcare

- Improved query performance through focused, smaller datasets
- Simplified data models tailored to specific user expertise
- Enhanced data security through role-based access control
- Faster development and deployment of analytics solutions
- Better alignment with clinical workflows and decision-making processes
- Cost-effective scalability - add marts as needs emerge

Implementation Approaches

Top-Down Approach: Build the enterprise data warehouse first, then create data marts as subsets. This ensures consistency but requires more upfront investment.

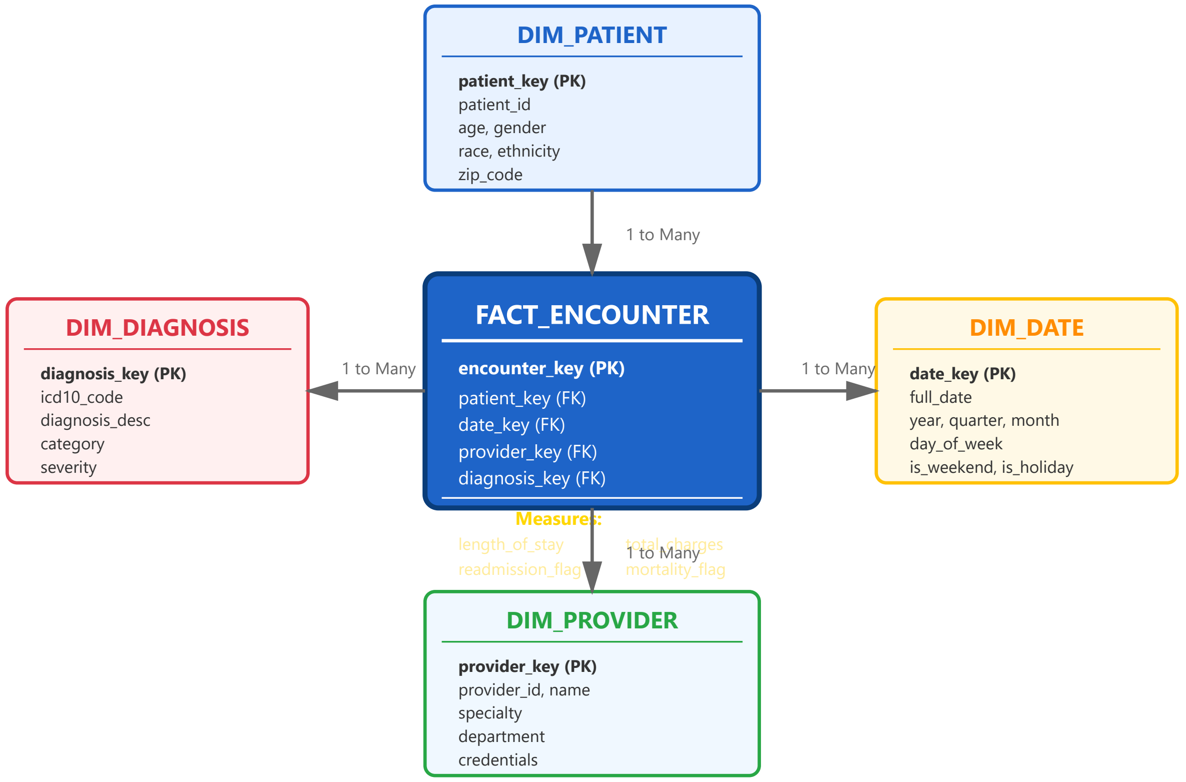
Bottom-Up Approach: Build individual data marts first based on urgent needs, then integrate them into an enterprise warehouse. This delivers faster ROI but may face integration challenges.

Hybrid Approach: Most healthcare organizations use a hybrid strategy, building a core data warehouse foundation while rapidly deploying specialized marts for high-priority clinical areas.

⭐ Star Schema: Detailed Explanation

Overview

The star schema is the most widely used dimensional model in data warehousing. Named for its star-like appearance, it consists of a central fact table surrounded by dimension tables. This design is intuitive for users, optimizes query performance, and provides flexibility for analysis. In healthcare, star schemas enable efficient analysis of clinical events, outcomes, and resource utilization.



Star Schema Components

Fact Tables: Store quantitative measures of business processes. In healthcare, these represent clinical events (encounters, lab tests, procedures) with numeric measurements (costs, counts, durations). Fact tables are typically large and grow continuously.

Dimension Tables: Provide descriptive context for facts. They answer "who, what, when, where, why" questions about the measures in fact tables. Dimension tables are relatively small and change slowly.

Example Query: Average Length of Stay by Diagnosis

Business Question: What is the average length of stay for heart failure patients by age group in Q1 2024?

SQL Query:

```
SELECT
  CASE WHEN p.age < 50 THEN 'Under 50'
    WHEN p.age BETWEEN 50 AND 70 THEN '50-70'
    ELSE 'Over 70' END as age_group,
  AVG(f.length_of_stay) as avg_los,
  COUNT(*) as encounter_count
FROM FACT_ENCOUNTER f
JOIN DIM_PATIENT p ON f.patient_key = p.patient_key
JOIN DIM_DIAGNOSIS d ON f.diagnosis_key = d.diagnosis_key
JOIN DIM_DATE dt ON f.date_key = dt.date_key
WHERE d.icd10_code LIKE 'I50%'
  AND dt.year = 2024 AND dt.quarter = 1
GROUP BY age_group;
```

Result: This simple query efficiently joins the star schema to reveal that patients over 70 with heart failure had an average LOS of 6.2 days compared to 4.1 days for those under 50.

Example: Laboratory Results Fact Table

FACT_LAB_RESULT: Each row represents one lab test result.
Foreign Keys: patient_key, date_key, provider_key, test_key, location_key
Measures: result_value (numeric), turnaround_time (minutes), cost
Degenerate Dimensions: order_number (doesn't warrant separate dimension table)

This structure enables queries like "What percentage of troponin tests ordered in the ED are elevated?" or "What's the average turnaround time for stat labs by shift?"

Slowly Changing Dimensions (SCD)

Healthcare data changes over time, and star schemas must handle these changes appropriately. For example, a patient's address changes, or a provider changes specialties.

Type 2 SCD Example: Provider Specialty Changes

Dr. Smith was a General Surgeon from 2020-2023, then became a Surgical Oncologist in 2024. Using Type 2 SCD:

DIM_PROVIDER Table:
provider_key: 1001, provider_id: DR_SMITH_001, specialty: General Surgery, effective_date: 2020-01-01, end_date: 2023-12-31, current_flag: N
provider_key: 1002, provider_id: DR_SMITH_001, specialty: Surgical Oncology, effective_date: 2024-01-01, end_date: 9999-12-31, current_flag: Y

This preserves history, enabling accurate analysis of surgical volumes by specialty over time while maintaining current information.

🔑 Advantages of Star Schema in Healthcare

- Simple, intuitive structure that clinical analysts can understand
- Fast query performance through denormalized dimensions
- Flexible - easily add new measures or dimensions without restructuring
- Optimized for BI tools and OLAP operations (slice, dice, drill-down)
- Efficient aggregations for dashboards and reports

- Consistent grain across facts enables complex cross-analysis

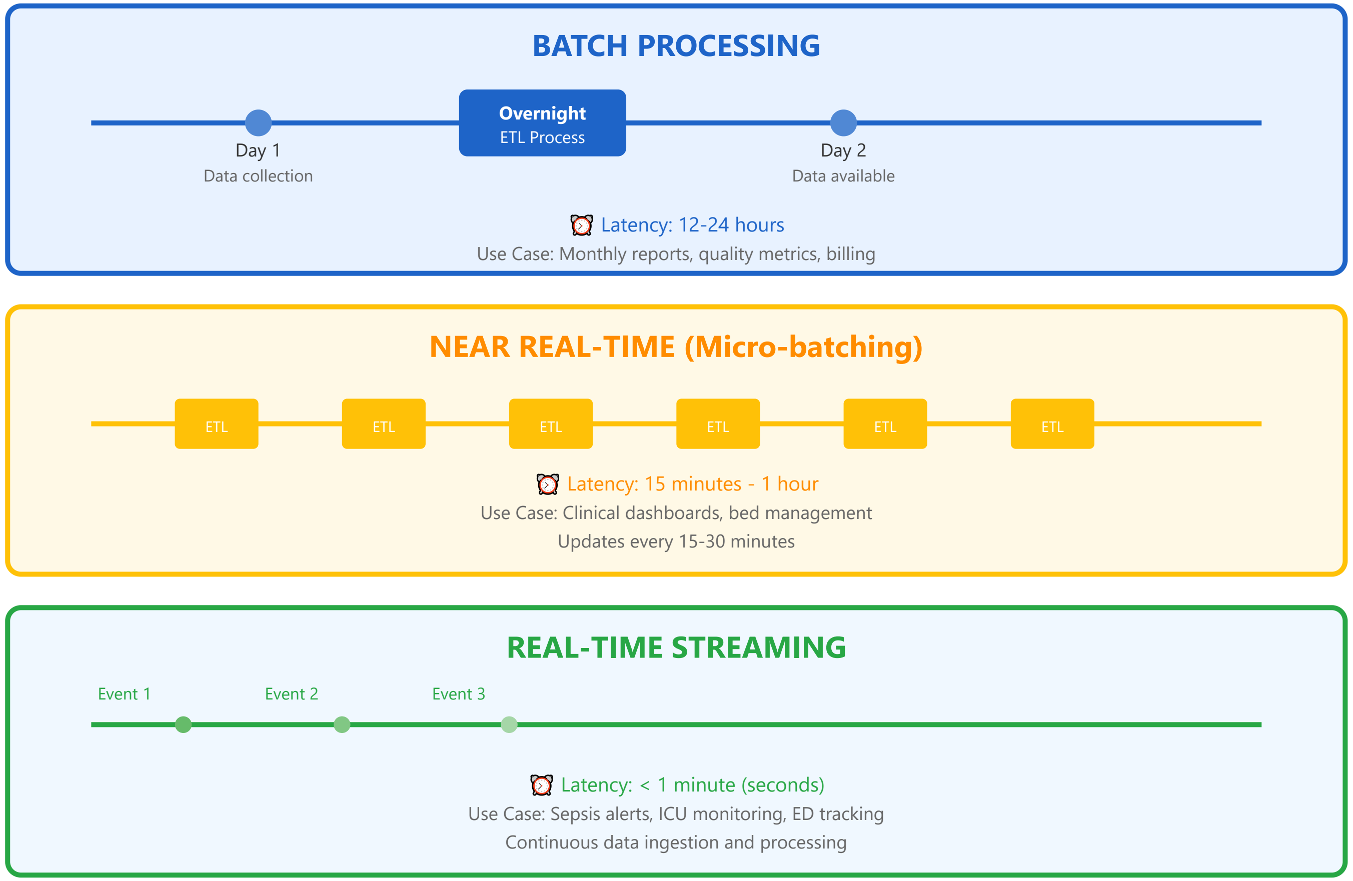
Star Schema vs. Snowflake Schema

While star schema uses denormalized dimensions, snowflake schema normalizes them into sub-dimensions. For example, in snowflake, DIM_PATIENT might link to separate DIM_GEOGRAPHY and DIM_DEMOGRAPHICS tables. Star schema is preferred in healthcare for its simplicity and query performance, though snowflake reduces storage redundancy.

⏱️ Real-time vs Batch Processing: Detailed Explanation

Overview

Healthcare data warehousing can employ different update strategies depending on requirements for data freshness, system complexity, and resource availability. The choice between real-time, near real-time, and batch processing significantly impacts system architecture, costs, and use cases.


BATCH PROCESSING
Overnight
ETL Process
Day 1
Data collection
Day 2
Data available
⏰ Latency: 12-24 hours
Use Case: Monthly reports, quality metrics, billing
NEAR REAL-TIME (Micro-batching)
ETL
ETL
ETL
ETL
ETL
ETL
⏰ Latency: 15 minutes - 1 hour
Use Case: Clinical dashboards, bed management
Updates every 15-30 minutes
REAL-TIME STREAMING
Event 1
Event 2
Event 3
⏰ Latency: < 1 minute (seconds)
Use Case: Sepsis alerts, ICU monitoring, ED tracking
Continuous data ingestion and processing

1. Batch Processing

Batch processing is the traditional approach where data is collected over a period and processed in large batches at scheduled intervals, typically overnight when system load is low.

Example: Overnight Claims Processing

Scenario: A hospital processes all billing and claims data at 11 PM daily.

Process:
• 11:00 PM - ETL job starts, extracts all encounters from past 24 hours
• 11:30 PM - Data transformation begins (coding, charge calculation)
• 2:00 AM - Load into data warehouse
• 3:00 AM - Aggregate tables and reports updated
• 6:00 AM - Finance team arrives to updated dashboards

Benefits: Minimal impact on operational systems, predictable processing windows, efficient resource utilization
Limitations: Data is 6-30 hours old when users access it

2. Near Real-Time Processing (Micro-batching)

Near real-time processing uses frequent small batches (every 15-60 minutes) to provide more current data without the complexity of true streaming. This balances freshness with system reliability.

Example: Emergency Department Bed Management

Scenario: ED dashboard shows current patient locations and wait times.

Implementation:
• Every 15 minutes, ETL process extracts ED patient tracking events
• Micro-batch processes: patient arrivals, triage completion, room assignments, discharges
• Dashboard updates showing: current ED census (47 patients), average wait time (32 min), beds available (3)
• Bed management team uses this to optimize patient flow

Technology: Apache Spark Structured Streaming, Azure Stream Analytics, or AWS Kinesis with windowing
Advantage: 90% of real-time benefits with 50% of the complexity

3. Real-Time Streaming

True real-time processing handles data as individual events occur, providing immediate insights. This is essential for time-critical clinical decisions but requires sophisticated infrastructure.

Example: Sepsis Early Warning System

Scenario: ICU deploys real-time sepsis detection to reduce mortality.

Real-time Data Streams:
• Vital signs from bedside monitors (heart rate, BP, temp) every 5 seconds
• Lab results (lactate, WBC) as soon as resulted
• Medication administration records
• Ventilator settings and parameters

Processing: Machine learning model analyzes streaming data, calculates sepsis risk score in real-time. When score exceeds threshold → immediate alert to nurse and physician within 30 seconds.

Outcome: Study showed 18% reduction in sepsis mortality with 45-minute faster intervention time.
Technology Stack: Apache Kafka for event streaming, Apache Flink for real-time computation, real-time ML inference

Example: Operating Room Utilization Tracking

Real-time Dashboard Components:
• Current case in each OR with elapsed time
• Next scheduled case and estimated start time
• Turnover time between cases

• Staff availability and equipment status

Data Sources: EHR procedure start/stop events, anesthesia system, RFID tracking

Business Value: Reduced OR idle time from 45 min to 20 min between cases, enabling 2 additional procedures per OR per week

Comparison and Decision Framework

Characteristic	Batch	Near Real-time	Real-time
Latency	12-24 hours	15-60 minutes	Seconds
Complexity	Low	Medium	High
Infrastructure Cost	$	$$	$$$$
Best Use Cases	Financial reports Quality metrics Research cohorts	Bed management Clinical dashboards Capacity planning	Sepsis alerts ICU monitoring Fraud detection

🔑 Choosing the Right Processing Strategy

- Start with batch processing for reporting and analytics - simplest and most reliable
- Add near real-time for operational dashboards with 15-30 minute refresh needs
- Implement true real-time only when immediate action is required for patient safety
- Hybrid architectures are common: batch for historical analysis, real-time for alerts
- Consider Lambda architecture: batch for accuracy, streaming for speed, both for comprehensive view

- Balance technical complexity against clinical value - not everything needs to be real-time

Implementation Challenges

Real-time Challenges: Data quality issues appear immediately without time for validation, requires 24/7 monitoring and support, higher infrastructure costs, complex failure handling and recovery.

Batch Challenges: Limited responsiveness to urgent situations, large processing windows can fail requiring full reruns, difficulty incorporating late-arriving data, users must wait for next cycle.

Best Practice: Most healthcare organizations use a tiered approach - batch processing for the bulk of data warehouse updates, near real-time for operational dashboards, and selective real-time streaming for critical clinical alerts where seconds matter.