**Technical Architecture Whitepaper**

**Schema Browser**

**1. Introduction**

Modern enterprises face the challenge of managing metadata across a fragmented landscape of databases and data platforms. Without a consistent approach to discover, catalog, and govern metadata, teams struggle with incomplete knowledge, compliance risks, and inefficient development cycles.

The **Schema Browser** addresses these challenges by providing an integrated, centralized solution for metadata discovery, enrichment, governance, and visualization across multiple data sources, including relational databases and big data systems.

This whitepaper describes the technical architecture, components, data flows, and operational principles of the Schema Browser platform.

**2. Objectives**

The Schema Browser has been designed to meet the following objectives:

* **Automated Metadata Discovery:** Continuously ingest schema information from multiple heterogeneous data sources.
* **Real-time Change Detection:** Respond instantly to schema changes using triggers and hooks.
* **Standardization and Enrichment:** Normalize metadata and enrich it with business context and data classifications.
* **Versioning and Auditability:** Maintain historical snapshots and logs for compliance and traceability.
* **Governed Access:** Ensure role-based permissions and secure access control.
* **Visualization and Consumption:** Provide interactive diagrams, lineage graphs, and profiling statistics to data consumers.

**3. Architectural Overview**

At its core, the Schema Browser architecture is composed of the following functional layers:

1. **Data Sources:** PostgreSQL, Oracle, SQL Server, MySQL, and Hive databases.
2. **Event Ingestion:** Triggers and hooks capture real-time schema events.
3. **Change Event Processing:** Deduplication, normalization, and event aggregation.
4. **Metadata Extraction:** Parsing, type mapping, semantic tagging, profiling, and change detection.
5. **Metadata Repository:** Centralized storage for canonical metadata, business annotations, lineage, and operational history.
6. **Security and Governance:** Access control and audit logging.
7. **Visualization Services:** ER diagram generation, lineage graph rendering, and statistics profiling.
8. **API Layer:** RESTful endpoints for programmatic access.

Each layer is modular, allowing scaling and customization for different data environments.

**4. Component Details**

Below, each major component is described in detail.

**4.1 Data Sources**

The Schema Browser supports any combination of relational and big data sources. It connects to:

* PostgreSQL via JDBC/ODBC
* Oracle via JDBC and system views
* SQL Server via system catalogs
* MySQL via information\_schema
* Hive via the Hive Metastore API

These connections are configured securely with dedicated credentials, limited to metadata queries.

**4.2 Triggers and Hooks**

Triggers and hooks provide **real-time event notifications** of schema changes:

* **Database Triggers:** Defined on DDL events (CREATE, ALTER, DROP) to capture schema modifications.
* **Hive Hooks:** Integrated into the Hive Metastore to detect table and column updates.

These mechanisms publish events into a message queue or webhook endpoint, where they are picked up by the Change Event Processor.

**4.3 Change Event Processor**

The Change Event Processor is responsible for:

* **Deduplication:** Collapsing multiple notifications of the same change.
* **Debouncing:** Grouping rapid, successive changes.
* **Event Normalization:** Producing a clean, structured event payload.

This ensures downstream components receive precise information without redundancy.

**4.4 Metadata Extractor**

This is the **core processing engine**, performing:

* **Parsing and Standardization:** Converts raw metadata into a normalized format across all systems.
* **Type Mapping:** Harmonizes data types into a canonical schema.
* **Semantic Tagging:** Identifies PII, sensitive fields, and business-specific labels.
* **Data Profiling:** Calculates statistics on null counts, distinct values, and distributions.
* **LoB/Application Mapping:** Associates metadata with lines of business and systems.
* **Change Detection:** Compares new snapshots against previous versions to detect deltas.

These operations produce enriched, versioned metadata ready for storage.

**4.5 Metadata Repository**

Metadata is stored in a central repository consisting of:

* **Schema Metadata:** Canonical representation of structures.
* **Business Metadata:** Glossary terms, descriptions, and sensitivity classifications.
* **Lineage Metadata:** Relationships and transformations across systems.
* **Operational Metadata:** Version history and change logs.

This repository is implemented in a relational database (e.g., PostgreSQL) with optional graph extensions for lineage storage.

**4.6 Security and Governance**

Access control and governance are embedded:

* **Role-Based Access Control:** Permissions by role (Viewer, Editor, Administrator).
* **Audit Logging:** Full record of access and modifications for compliance.

**4.7 Visualization Services**

Three visualization modules deliver insights to users:

* **ER Diagram Generator:** Creates relational diagrams of tables and columns.
* **Lineage Graph Builder:** Displays end-to-end data flows.
* **Statistics Profiler:** Shows data quality metrics.

Visualizations can be filtered and exported in standard formats (PNG, SVG).

**4.8 API Layer**

A RESTful API provides access to all metadata. Key endpoints include:

* List Schemas
* Retrieve Business Metadata
* Retrieve Lineage Graphs
* Retrieve Profiling Statistics
* Manage Access Controls
* Query Audit Logs

This enables integration with data catalogs, governance tools, and other enterprise systems.

**5. Data Flow**

The operational flow can be summarized as follows:

1. **Acquisition:**
   * The scheduler initiates metadata extraction, or an event triggers real-time ingestion.
2. **Processing:**
   * The Change Event Processor deduplicates and normalizes events.
   * The Metadata Extractor processes and enriches metadata.
   * The Change Detector identifies deltas.
3. **Storage:**
   * Metadata is stored in the repository and versioned.
4. **Governance:**
   * Access controls are enforced.
   * Audit logs are updated.
5. **Consumption:**
   * Visualization services refresh diagrams and lineage graphs.
   * APIs make metadata accessible to authorized consumers.

**6. Security Considerations**

The Schema Browser architecture enforces:

* **Least Privilege Access:** All database connections are read-only.
* **Encryption:** All data in transit uses TLS/SSL.
* **Audit Trails:** All actions are logged for compliance.
* **Access Policies:** Role-based controls govern permissions.

**7. Scalability and Extensibility**

The system is designed to scale horizontally:

* Multiple extractor instances can run in parallel.
* The metadata repository can be sharded by database or line of business.
* Event processing can be decoupled with message queues.
* New connectors can be added for additional data sources.

**8. Integration Points**

The Schema Browser can be integrated with:

* Enterprise data catalogs (e.g., Collibra, Alation)
* Data governance platforms
* BI tools and data preparation pipelines
* Custom dashboards via REST API

**9. Technology Stack**

Suggested stack includes:

* **Backend:** Python or Java (Spring Boot)
* **Storage:** PostgreSQL or Oracle DB
* **Queueing:** Kafka or RabbitMQ
* **UI:** React or Angular
* **Visualization:** D3.js, Mermaid, or Graphviz
* **Authentication:** OAuth 2.0 or LDAP

**10. Roadmap**

Planned enhancements include:

* Business glossary integration.
* Impact simulation tools.
* Self-service metadata editing.
* Enhanced lineage discovery from ETL logs.

**11. Conclusion**

The Schema Browser provides a modern, scalable solution for metadata management in complex data environments. Its architecture ensures:

* Timely and accurate metadata ingestion.
* Unified business and technical context.
* Robust governance and audit capabilities.
* Rich visualizations to build data trust.

With these capabilities, organizations can improve compliance, reduce data-related risk, and accelerate data-driven innovation.

**12. Appendix**

If you’d like, I can prepare:

* ER diagrams of the metadata repository.
* OpenAPI specifications.
* Deployment guides (Docker Compose, Kubernetes).
* Data steward training materials.

✅ **Next Steps:**  
If you need this whitepaper in a formatted document (Word or PDF), I can prepare it for you.  
Just say the word—I’m here to help!

**You said:**

Format the document in word .. need more detail 2. architecture overview, component details, data flow, technology stack

**ChatGPT said:**

Absolutely—let’s create a **detailed and well-formatted Word document**-style version of your **Technical Architecture Whitepaper**, expanding particularly on:  
✅ Architecture Overview  
✅ Component Details  
✅ Data Flow  
✅ Technology Stack

Below is the **expanded text** you can paste into Word, with clear section headings, subsections, and detailed descriptions.

**📄 Schema Browser – Technical Architecture Whitepaper**

**1. Introduction**

Modern enterprises operate in data ecosystems that span multiple platforms, technologies, and business lines. Maintaining a consistent, accurate, and governed understanding of metadata across such an environment is a complex challenge.

The **Schema Browser** is designed to solve this challenge by automating metadata discovery, enrichment, governance, and visualization across heterogeneous databases.

This document provides a technical architecture reference for the Schema Browser platform, including its goals, components, data flows, and operational principles.

**2. Objectives**

The primary objectives of the Schema Browser are to:

* Automate the discovery of metadata across diverse relational and big data platforms.
* Standardize and enrich metadata with business context and quality profiles.
* Maintain versioned histories and enable detailed lineage tracking.
* Enforce robust governance through access controls and audit logs.
* Deliver intuitive visualization and API access for consumers across the enterprise.

**3. Expanded Architecture Overview**

The Schema Browser architecture is logically divided into **eight functional layers**, each with specialized responsibilities.

These layers are:

1. **Data Source Layer**  
   Contains all connected databases—PostgreSQL, Oracle, SQL Server, MySQL, and Hive—where data schemas are defined.
2. **Event Detection Layer**  
   Responsible for capturing schema change events in real-time through triggers and hooks.
3. **Change Event Processing Layer**  
   A normalization and aggregation hub that ensures only relevant and deduplicated events trigger metadata updates.
4. **Metadata Extraction and Processing Layer**  
   A pipeline of components that parses, standardizes, enriches, and profiles metadata, ensuring it is consistent across systems.
5. **Metadata Repository Layer**  
   The canonical store for all metadata, versioned and categorized into schema, business, lineage, and operational facets.
6. **Governance Layer**  
   Enforces role-based access control and audit logging for compliance and security.
7. **Visualization Layer**  
   Provides interfaces to view, search, and analyze metadata through ER diagrams, lineage graphs, and statistics dashboards.
8. **API Layer**  
   Exposes RESTful endpoints for integration with other tools and programmatic access to metadata.

Each layer is modular and can be scaled independently, enabling flexible deployments in small teams or large enterprises.

**4. Expanded Component Details**

**4.1 Data Sources**

The Schema Browser supports any combination of relational databases and Hive. Each source is configured with secure credentials limited to metadata read access.

The system uses native query interfaces such as:

* PostgreSQL: information\_schema and system catalogs
* Oracle: DBA\_TABLES, DBA\_TAB\_COLUMNS
* MySQL: information\_schema
* SQL Server: sys.tables, sys.columns
* Hive: Hive Metastore API

This guarantees compatibility with platform-specific features and optimizes performance.

**4.2 Triggers and Hooks**

Schema change detection operates in real-time by deploying **triggers and hooks**:

* **Database Triggers** are installed to capture DDL operations such as CREATE, ALTER, and DROP.
* **Hive Hooks** integrate with the Hive Metastore to detect schema changes.
* When a change occurs, the trigger or hook generates a structured event record containing:
  + Database and schema name
  + Table or column name
  + Type of operation
  + Timestamp

These records are sent to the **Change Event Processor** via a message queue or webhook endpoint.

**4.3 Change Event Processor**

The Change Event Processor performs several critical functions:

* **Deduplication:** Eliminates duplicate notifications arising from repeated or cascading schema changes.
* **Debouncing:** Groups rapid sequences of related changes into a single event batch.
* **Event Normalization:** Converts platform-specific event formats into a consistent internal model.

This ensures that only actionable, clean events trigger downstream metadata extraction.

**4.4 Metadata Extractor**

The extractor is the heart of the platform, consisting of several submodules:

* **Parsing and Standardization:** Converts raw metadata into a normalized internal schema representation.
* **Type Mapping:** Harmonizes data types into canonical categories.
* **Semantic Tagging:** Automatically flags fields containing sensitive data, such as PII.
* **Data Profiling:** Optionally samples data to compute quality metrics like null rates and distinct counts.
* **LoB and Application Mapping:** Enriches metadata with business context, mapping tables to lines of business and applications.
* **Change Detection:** Compares incoming metadata against the repository to detect deltas.
* **Metadata Loader:** Prepares versioned metadata objects for storage.

**4.5 Metadata Repository**

The Metadata Repository is a relational database storing metadata across four categories:

* **Schema Metadata:** Canonical records of tables, columns, constraints, and data types.
* **Business Metadata:** Descriptions, glossary terms, and classifications.
* **Lineage Metadata:** Directed graphs describing how data flows and transforms across systems.
* **Operational Metadata:** Logs of changes and version history.

Each category is stored in normalized tables with clear referential integrity and timestamping.

**4.6 Security and Governance**

Security and governance are critical to enterprise deployments:

* **Access Control:** Defines granular role-based permissions, controlling who can view or modify metadata.
* **Audit Logging:** Tracks every action taken by users or APIs, including viewing metadata, exporting diagrams, or modifying business labels.

This design ensures compliance with regulatory requirements such as GDPR and SOX.

**4.7 Visualization Layer**

Visualization components consume metadata and provide:

* **ER Diagrams:** Automatically generated entity-relationship diagrams depicting tables and relationships.
* **Lineage Graphs:** Interactive views of data flows from sources through transformations to targets.
* **Statistics Dashboards:** Quality metrics computed during profiling.

These tools help users understand their data ecosystem with clarity.

**4.8 API Layer**

The API Layer provides RESTful endpoints for programmatic access to:

* Schema metadata
* Business metadata
* Lineage graphs
* Statistics
* Access control management
* Audit logs

All APIs are secured via token-based authentication and support pagination and filtering.

**5. Expanded Data Flow Description**

The operational flow begins when the system either:

* Executes a **scheduled extraction job**, querying all configured data sources for the latest schema definitions, or
* Receives a **real-time event notification** from a trigger or hook.

Upon receiving input, the Change Event Processor cleans and normalizes events, ensuring only unique, actionable changes are passed forward.

The Metadata Extractor then connects to the relevant data source and performs metadata parsing, standardization, enrichment, profiling, and change detection. If differences from previous versions are identified, these deltas are prepared for storage.

The Metadata Loader writes new or updated records into the Metadata Repository, versioning them and updating operational logs.

Once stored, the Security Layer enforces access permissions and logs all activity. Visualization components are notified to refresh their views, ensuring the latest metadata is always available.

Finally, users and tools can query the REST API to access or export metadata.

**6. Security Considerations**

The system enforces several security principles:

* All connections to databases are configured for read-only access.
* Metadata in transit is encrypted via TLS.
* Role-based access controls govern all operations.
* Every action is audited and stored indefinitely.

**7. Expanded Technology Stack**

The following stack is recommended:

* **Backend Framework:**
  + Java Spring Boot or Python FastAPI
  + Supports asynchronous processing and dependency injection.
* **Metadata Storage:**
  + PostgreSQL for schema and business metadata.
  + Optional Neo4j or JanusGraph for lineage storage.
* **Event Queue:**
  + Apache Kafka or RabbitMQ for scalable event ingestion.
* **Visualization:**
  + React.js front end with D3.js for ER and lineage rendering.
* **API Layer:**
  + RESTful APIs documented in OpenAPI.
* **Authentication:**
  + OAuth 2.0 with role mapping.
* **Deployment:**
  + Docker containers orchestrated via Kubernetes.

This stack is chosen for scalability, maintainability, and open-source friendliness.

**8. Integration Points**

The Schema Browser integrates with:

* Data catalogs like Collibra or Alation.
* BI tools such as Tableau and Power BI.
* ETL platforms including Apache NiFi.
* Governance frameworks for audit and compliance.

**9. Roadmap**

Future enhancements include:

* Automated lineage extraction from ETL logs.
* Self-service metadata editing.
* Business glossary integration.
* Impact simulation tools for schema changes.

**10. Conclusion**

The Schema Browser provides a robust, modular, and secure foundation for metadata management in complex data environments. By automating discovery, enrichment, governance, and visualization, it delivers trusted, actionable metadata that supports compliance and accelerates data-driven innovation.