**Schema Browser**

**The Schema Browser is an enterprise-grade platform that discovers, catalogs, visualizes, and governs metadata across multiple data sources such as relational databases, data warehouses, and big data platforms.**

**It acts as a central metadata hub, automatically extracting technical, business, and lineage information from systems like PostgreSQL, Oracle, MySQL, SQL Server, and Hive, and presenting it in an accessible, governed way.**

**he Schema Browser begins its operation by establishing secure connections to all configured data sources. These data sources can include a diverse array of technologies, such as PostgreSQL, Oracle, SQL Server, MySQL, and Hive. For each of these systems, the Schema Browser has pre-configured credentials and connection strings, allowing it to connect using either JDBC drivers, ODBC, or native APIs, depending on the platform.**

**When the system starts, it checks whether it is scheduled to run an extraction job or if it has been notified by an event trigger. The extraction process has two modes: scheduled extraction, which happens at defined intervals (for example, every 4 hours), and event-driven extraction, which happens whenever a schema change is detected in real time.**

**In the scheduled mode, the system launches connections to all target databases, reads their system catalogs—such as information\_schema for relational databases or Hive Metastore tables—and collects a snapshot of all structural metadata. This includes the names of tables, their columns, data types, constraints, indexes, and any relevant properties.**

**Simultaneously, the Schema Browser listens for real-time events generated by triggers or hooks installed on the databases. For example, when a table is altered in Oracle, or a new column is added in PostgreSQL, database triggers fire and push a notification to a message queue or webhook endpoint. In Hive, hooks integrated into the metastore emit events whenever DDL operations occur. These events are then sent to a central component called the Change Event Processor.**

**The Change Event Processor’s job is to act as a filter and aggregator for these events. In real operational environments, databases can emit dozens of similar events in a short timeframe, especially when complex schema changes happen. The processor therefore debounces the events so they don’t overwhelm downstream systems and deduplicates them so that multiple notifications of the same change are collapsed into a single clean event. It also groups related changes together, such as a table rename immediately followed by column modifications, producing a coherent description of what changed and why.**

**Once the event has been processed, it is passed to the Metadata Extractor, which forms the heart of the Schema Browser. This extractor begins by parsing the raw metadata fetched from the database or supplied in the event. The parsing module reads the technical definitions—such as column names and data types—and transforms them from the vendor’s proprietary representations into a standardized, canonical model. For instance, Oracle’s VARCHAR2 or PostgreSQL’s TEXT types might both be mapped internally to a common “String” type in the Schema Browser’s data model.**

**After standardization, the metadata moves into a type mapping phase where additional normalization occurs to ensure consistency across systems. This is followed by semantic tagging, in which the extractor inspects the metadata for patterns that identify sensitive information. If a column name contains keywords like “email,” “SSN,” or “credit card,” the system automatically tags it as containing personally identifiable information (PII) or sensitive data. Business labels or glossary terms can also be attached at this stage.**

**Next, the extractor performs data profiling by sampling the contents of tables or columns—if permitted—and calculating descriptive statistics. This profiling captures metrics such as the percentage of nulls in each column, the number of unique values, and, where relevant, min/max ranges. These statistics help data stewards understand data quality without requiring separate profiling tools.**

**Once the metadata has been fully parsed, standardized, tagged, and profiled, the extractor invokes the Line of Business and Application Mapper, which links the metadata to specific business domains, applications, and customer systems. For example, a table might be mapped to the Mortgage Line of Business and the Core Banking application.**

**Before saving any updates, the Change Detector module compares the newly processed metadata snapshot against the most recent version stored in the repository. This comparison identifies deltas: whether columns were added, data types changed, or tables were dropped. Any differences are marked for versioning and audit purposes.**

**The Metadata Loader then prepares the final metadata objects and saves them into the Metadata Repository, which is the system’s central catalog. The repository is organized into several logical sections: Schema Metadata holds the canonical representations of tables and columns; Business Metadata holds descriptions, tags, and classifications; Lineage Metadata stores records of data flows and transformations; and Operational Metadata maintains version histories and change logs.**

**Once stored, the metadata is protected by a Security and Governance layer. This includes Access Control, which defines role-based permissions, specifying who can view, edit, or delete metadata. It also includes Audit Logging, which records every interaction with the metadata—who accessed it, what action they took, and when. This ensures that the platform not only centralizes metadata but also meets compliance and governance standards.**

**On the user-facing side, the Schema Browser provides a set of visualization tools. The ER Diagram Generator queries the Schema Metadata and automatically constructs entity-relationship diagrams that depict how tables and columns relate to each other. These diagrams can be filtered by database, schema, or business context and exported as images or vector graphics.**

**The Lineage Graph Builder uses the Lineage Metadata to render data flow diagrams that illustrate where data originates, how it transforms through ETL processes, and where it ends up. This helps data analysts and stewards perform impact analysis when planning schema changes.**

**Finally, the Statistics Profiler surfaces the data profiling results, giving stewards insights into data completeness, uniqueness, and other quality attributes.**

**Whenever new metadata arrives—whether through scheduled extractions or real-time events—the Schema Browser automatically re-runs this pipeline: extracting, enriching, versioning, storing, governing, and refreshing visualizations. This ensures that all stakeholders—from data stewards to compliance teams—have an accurate, trustworthy, and up-to-date understanding of their data landscape.**

**🟢 1️⃣ Databases**

**These are your data sources.  
They can be any combination of:**

* **PostgreSQL**
* **Oracle**
* **SQL Server**
* **MySQL**
* **Hive**

**Purpose:**

* **Host your transactional and analytical data.**
* **Provide schemas (tables, columns, constraints).**
* **Emit events when something changes.**

**Working Principle:**

* **The system connects to each DB via JDBC or native drivers.**
* **Periodically extracts metadata (structure, types).**
* **Also listens to triggers/hooks for real-time change detection.**

**🟢 2️⃣ Triggers & Hooks**

**These are change notification mechanisms that detect when your schema or data changes.**

**Components:**

* **DB Triggers:**
  + **DDL (CREATE, ALTER, DROP) triggers fire when schema changes.**
  + **DML triggers can also capture row-level changes.**
  + **Typically implemented via native trigger mechanisms in Oracle, PostgreSQL, MySQL.**
* **Hive Hooks:**
  + **Hive supports hooks in the metastore.**
  + **Fires when a table is created, altered, or dropped.**

**Working Principle:**

* **When an event occurs (e.g., someone adds a column), a trigger fires.**
* **The trigger writes to an event queue or log or calls a webhook.**
* **These events are picked up by the Change Event Processor.**

**🟢 3️⃣ Change Event Processor**

**Purpose:**

* **Central hub for ingesting all change events.**

**Working Principle:**

* **Listens to event streams from Triggers/Hooks.**
* **Debounces rapid-fire events (e.g., multiple table alters).**
* **Deduplicates similar events.**
* **Groups related events (e.g., ALTER TABLE + COMMENT ON TABLE).**
* **Prepares a normalized event describing what changed.**
* **Sends it downstream to the Metadata Extractor.**

**Why you need it:**

* **Without it, you would get redundant or conflicting updates.**
* **Helps keep the metadata repository consistent.**

**🟢 4️⃣ Metadata Extractor**

**This is the core engine that:**

* **Connects to databases.**
* **Pulls metadata.**
* **Cleans, enriches, and standardizes it.**

**Subcomponents & Working Principle:**

**🔹 Parsing & Standardization**

* **Reads raw metadata from information\_schema, system catalogs, or Hive metastore.**
* **Converts to a common model (e.g., unified JSON structure).**

**🔹 Type Mapper**

* **Maps DB-specific data types (e.g., NUMBER, INT, VARCHAR) to canonical types (e.g., Integer, String).**

**🔹 Semantic Tagger**

* **Tags columns with business semantics:**
  + **PII detection (e.g., columns containing “email” or “SSN”).**
  + **Sensitivity classification (e.g., confidential, internal).**

**🔹 Quality Profiler**

* **Gathers statistics:**
  + **Null percentages**
  + **Distinct counts**
  + **Min/max values (where possible)**

**🔹 LoB / CSI / DB Mapper**

* **Associates the metadata to:**
  + **Line of Business (Retail, Cards, Mortgage)**
  + **CSI (Customer System Identifier)**
  + **Application name**

**🔹 Scheduler & Dependency Manager**

* **Coordinates when and how frequently extraction runs.**
* **Ensures dependencies are respected (e.g., waits for Hive metastore to refresh).**

**🔹 Metadata Loader**

* **Prepares metadata objects for repository storage.**

**🔹 Change Detector**

* **Compares the new metadata snapshot to the previous version.**
* **Marks deltas (e.g., “new column added”, “table dropped”).**

**🟢 5️⃣ Metadata Repository**

**This is your central data catalog—a structured database to store everything about your metadata.**

**Subcomponents:**

**🔹 Schema Metadata**

* **Canonical representation of tables, columns, constraints, indexes.**

**🔹 Business Metadata**

* **Tags, labels, business glossary associations.**

**🔹 Lineage Metadata**

* **Graph of:**
  + **Where data comes from (source)**
  + **How it’s transformed (ETL logic)**
  + **Where it goes (targets)**
* **Useful for impact analysis and data governance.**

**🔹 Operational Metadata**

* **Version history:**
  + **When metadata changed.**
  + **Who changed it.**
* **Logs of all past states.**

**🟢 6️⃣ Security & Governance**

**Provides control and accountability over your metadata.**

**Components:**

**🔹 Access Control**

* **Who can read, edit, or delete metadata.**
* **Role-based permissions:**
  + **Viewer**
  + **Editor**
  + **Administrator**

**🔹 Audit Logging**

* **Records every action:**
  + **Who accessed metadata.**
  + **When and what was changed.**
  + **Before/after snapshots.**

**🟢 7️⃣ Visualization Components**

**These are your front-end tools for exploring metadata.**

**🔹 ER Diagram Generator**

* **Reads schema metadata.**
* **Auto-generates entity-relationship diagrams.**
* **Allows:**
  + **Filtering by database or LoB.**
  + **Exporting diagrams (PNG, SVG).**

**🔹 Lineage Graph Builder**

* **Reads lineage metadata.**
* **Creates directed graphs:**
  + **Shows data flow from source to target.**
  + **Supports impact analysis (e.g., “What happens if I change this column?”).**

**🔹 Statistics Profiler**

* **Displays profiling stats:**
  + **Null counts**
  + **Cardinality**
  + **Data distributions**
* **Helps users assess data quality.**

**🟢 🎯 End-to-End Working Example**

**Imagine this scenario:**

1. **A developer adds a column to customer\_account table in Oracle.**
2. **The DB Trigger fires, writing an event to the event stream.**
3. **The Change Event Processor detects it, deduplicates, and forwards the event.**
4. **The Metadata Extractor connects to Oracle:**
   * **Parses the updated schema.**
   * **Standardizes the types.**
   * **Tags the new column as PII.**
   * **Profiles initial null counts.**
   * **Maps it to the Retail LoB.**
5. **The Change Detector compares with the old version:**
   * **Notes the new column.**
   * **Prepares a delta.**
6. **The Metadata Loader saves it into the Metadata Repository:**
   * **Schema Metadata updated.**
   * **Operational Metadata logs the change.**
   * **Audit Logging records who did it.**
7. **The Visualization Components:**
   * **ER Diagram Generator shows the updated table.**
   * **Lineage Graph reflects the new column.**
   * **Statistics Profiler shows the column stats.**
8. **Access Control ensures only authorized users can view or edit the metadata.**

**✅ This end-to-end pipeline guarantees:**

* **Near real-time metadata updates.**
* **Governance and auditability.**
* **Clear, consistent presentation to users.**

**Database Schemas for Metadata Repository**

Below is a **modular design** aligned with your architecture components:

**✅ 1️⃣ schema\_metadata**

Stores canonical representation of tables, columns, data types.

sql

CopyEdit

CREATE TABLE schema\_metadata (

schema\_id SERIAL PRIMARY KEY,

database\_name VARCHAR(100) NOT NULL,

schema\_name VARCHAR(100) NOT NULL,

table\_name VARCHAR(100) NOT NULL,

table\_type VARCHAR(50), -- e.g., BASE TABLE, VIEW

column\_name VARCHAR(100) NOT NULL,

ordinal\_position INT,

data\_type VARCHAR(100),

is\_nullable BOOLEAN,

is\_primary\_key BOOLEAN DEFAULT FALSE,

default\_value VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**✅ 2️⃣ business\_metadata**

Stores business tags, descriptions, LoB mappings.

sql

CopyEdit

CREATE TABLE business\_metadata (

business\_id SERIAL PRIMARY KEY,

schema\_id INT REFERENCES schema\_metadata(schema\_id),

lob\_name VARCHAR(100),

application\_name VARCHAR(100),

business\_description TEXT,

data\_classification VARCHAR(50), -- e.g., Confidential, Public

glossary\_term VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**✅ 3️⃣ lineage\_metadata**

Stores data lineage relationships.

sql

CopyEdit

CREATE TABLE lineage\_metadata (

lineage\_id SERIAL PRIMARY KEY,

source\_schema\_id INT REFERENCES schema\_metadata(schema\_id),

target\_schema\_id INT REFERENCES schema\_metadata(schema\_id),

transformation\_description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**✅ 4️⃣ operational\_metadata**

Stores versioning and change history.

sql

CopyEdit

CREATE TABLE operational\_metadata (

operation\_id SERIAL PRIMARY KEY,

schema\_id INT REFERENCES schema\_metadata(schema\_id),

operation\_type VARCHAR(50), -- e.g., ADD COLUMN, DROP TABLE

previous\_state JSONB,

new\_state JSONB,

performed\_by VARCHAR(100),

performed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**✅ 5️⃣ access\_control**

Role-based permissions.

sql

CopyEdit

CREATE TABLE access\_control (

access\_id SERIAL PRIMARY KEY,

schema\_id INT REFERENCES schema\_metadata(schema\_id),

role\_name VARCHAR(50), -- e.g., Viewer, Editor, Admin

can\_read BOOLEAN DEFAULT TRUE,

can\_write BOOLEAN DEFAULT FALSE,

can\_delete BOOLEAN DEFAULT FALSE,

granted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**✅ 6️⃣ audit\_log**

Who accessed or modified metadata.

sql

CopyEdit

CREATE TABLE audit\_log (

audit\_id SERIAL PRIMARY KEY,

schema\_id INT REFERENCES schema\_metadata(schema\_id),

user\_name VARCHAR(100),

action VARCHAR(50), -- e.g., VIEWED, UPDATED

action\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**🟢 🌐 API Specifications for Metadata Access**

Below is a **REST API design** you can expose to your consumers:

**✅ 1️⃣ Get All Schemas**

**GET** /api/v1/schemas

**Response:**

json

CopyEdit

[

{

"schema\_id": 123,

"database\_name": "PostgreSQL",

"schema\_name": "public",

"table\_name": "customer",

"columns": [

{

"column\_name": "customer\_id",

"data\_type": "Integer",

"is\_nullable": false,

"is\_primary\_key": true

}

]

}

]

**✅ 2️⃣ Get Schema by ID**

**GET** /api/v1/schemas/{schema\_id}

**✅ 3️⃣ Search Schemas by Table Name**

**GET** /api/v1/schemas/search?table\_name=customer

**✅ 4️⃣ Get Business Metadata**

**GET** /api/v1/schemas/{schema\_id}/business-metadata

**✅ 5️⃣ Get Lineage**

**GET** /api/v1/lineage/{schema\_id}

**Response:**

json

CopyEdit

{

"source\_table": "customer",

"target\_table": "customer\_dim",

"transformation": "ETL process: filter and aggregate"

}

**✅ 6️⃣ Create / Update Business Metadata**

**POST** /api/v1/business-metadata

**Body:**

json

CopyEdit

{

"schema\_id": 123,

"lob\_name": "Retail",

"application\_name": "Core Banking",

"business\_description": "Stores customer data",

"data\_classification": "Confidential",

"glossary\_term": "Customer"

}

**✅ 7️⃣ Get Audit Logs**

**GET** /api/v1/audit/{schema\_id}

**✅ 8️⃣ Get Access Control**

**GET** /api/v1/access/{schema\_id}

**✅ 9️⃣ Update Access Control**

**PUT** /api/v1/access/{schema\_id}

**Body:**

json

CopyEdit

{

"role\_name": "Editor",

"can\_read": true,

"can\_write": true,

"can\_delete": false

}

**🟢 ✨ Additional Recommendations**

✅ Use **OpenAPI / Swagger** to auto-document the API.  
✅ Implement **pagination and filtering** in GET endpoints.  
✅ Consider **GraphQL** if you need flexible queries across schemas and lineage.  
✅ Add **OAuth 2.0** for authentication.  
✅ Include **rate limiting and logging** for security.

**✨ Key Features**

Here are the major capabilities that make the Schema Browser powerful:

**✅ 1. Multi-Database Metadata Extraction**

* Connects to diverse data sources.
* Extracts table structures, columns, data types, constraints, and indexes.
* Converts vendor-specific metadata into a **canonical, unified format**.

**✅ 2. Real-Time Change Detection**

* Leverages **database triggers and Hive hooks** to detect schema changes as they happen.
* Processes change events to keep the metadata repository **synchronized** in near real time.

**✅ 3. Metadata Standardization & Enrichment**

* **Standardizes data types** across systems (e.g., mapping VARCHAR2 and TEXT to String).
* **Classifies columns** (e.g., PII, sensitive data).
* **Associates metadata** with Lines of Business, applications, and glossary terms.

**✅ 4. Data Lineage Visualization**

* Captures and displays **end-to-end lineage**:
  + Where data comes from
  + How it transforms
  + Where it goes
* Helps teams understand data dependencies and impact analysis.

**✅ 5. ER Diagram Generation**

* Automatically renders **Entity Relationship Diagrams (ERDs)**.
* Supports filtering by database, schema, or business domain.
* Enables exporting diagrams for documentation or analysis.

**✅ 6. Data Profiling**

* Computes **column-level statistics**:
  + Null percentages
  + Unique value counts
  + Basic distributions
* Provides insights into data quality.

**✅ 7. Security and Governance**

* Role-based **access control** ensures only authorized users can view or modify metadata.
* **Audit logging** tracks who accessed metadata and what actions were performed.

**✅ 8. Centralized Metadata Repository**

* Stores:
  + **Schema metadata** (technical definitions)
  + **Business metadata** (descriptions, tags)
  + **Lineage metadata** (data flow relationships)
  + **Operational metadata** (versioning and history)
* Supports versioned snapshots for auditing and rollback.

**🟢 ⚙️ Working Principle**

Below is a step-by-step look at how the Schema Browser works internally:

**🔹 1. Metadata Acquisition**

* **Scheduled Extractors** connect to all configured databases on a periodic basis.
* **Triggers and Hooks** emit events for real-time change detection.

**🔹 2. Event Processing**

* The **Change Event Processor**:
  + Debounces rapid-fire events.
  + Deduplicates redundant notifications.
  + Groups related changes.
* Sends clean, actionable events to the extractor pipeline.

**🔹 3. Parsing and Standardization**

* The **Metadata Extractor**:
  + Reads raw metadata (e.g., system catalogs).
  + Maps proprietary types to a **common schema**.
  + Enriches metadata with semantic tags and Line of Business associations.
  + Profiles data quality metrics.

**🔹 4. Change Detection and Versioning**

* The **Change Detector** compares new metadata to previously stored versions.
* Identifies:
  + Added or dropped tables.
  + Modified columns.
  + Changed data types.
* Logs all changes in **Operational Metadata**.

**🔹 5. Repository Storage**

* All processed metadata is saved in the **Metadata Repository**, organized into:
  + Schema Metadata
  + Business Metadata
  + Lineage Metadata
  + Operational Metadata

**🔹 6. Governance and Access**

* **Access Control** enforces permissions on who can see or edit each metadata asset.
* **Audit Logging** records all interactions for compliance.

**🔹 7. Visualization**

* Users access the **Schema Browser UI** or APIs to:
  + Generate ER diagrams.
  + Explore lineage graphs.
  + Analyze data profiles.
* Metadata can be exported in multiple formats (PNG, SVG, JSON).

**🟢 💡 Why Use a Schema Browser?**

Organizations today operate in **complex, hybrid data environments**.  
The Schema Browser provides:

✅ **Visibility:** A unified view of all data structures and dependencies.  
✅ **Governance:** Auditability, access control, and versioning.  
✅ **Productivity:** Faster onboarding and impact analysis for developers and analysts.  
✅ **Quality:** Clear insights into data health and completeness.

**Pitch Deck – Schema Browser Business Value**

Here’s a **structured outline** you can convert into slides:

**🎨 Slide 1 – Title Slide**

**Schema Browser**  
*Unifying Metadata Discovery, Lineage, and Governance Across Your Data Estate*

**🎨 Slide 2 – The Challenge**

* **Heterogeneous Data Landscape**
  + Multiple databases: PostgreSQL, Oracle, MySQL, Hive
  + Lack of standardization
* **Poor Visibility**
  + No clear lineage
  + Hard to find where data comes from
* **Governance Gaps**
  + Limited access control
  + Weak audit trails
* **Impact on Business**
  + Higher compliance risks
  + Longer development cycles
  + Lower trust in data

**🎨 Slide 3 – The Solution**

**Schema Browser**

* A centralized metadata platform
* Real-time schema discovery
* Standardized catalog
* Lineage visualization
* Role-based governance

**🎨 Slide 4 – Key Features**

✅ **Automated Metadata Extraction**  
✅ **Real-Time Change Detection**  
✅ **Lineage and Impact Analysis**  
✅ **ER Diagram Generation**  
✅ **Data Profiling**  
✅ **Access Control & Audit Logging**

**🎨 Slide 5 – Business Benefits**

**For Compliance**

* Transparent audit logs
* Controlled data access

**For Developers**

* Faster onboarding
* Less time searching for schemas

**For Data Stewards**

* Single source of truth
* Easier impact assessment

**For Executives**

* Reduced operational risk
* Improved decision-making confidence

**🎨 Slide 6 – ROI and Value**

* **Time saved**: Up to 40% reduction in manual metadata tracking.
* **Risk reduction**: Lower chance of regulatory penalties.
* **Efficiency gains**: Faster delivery of data products.

**🎨 Slide 7 – How It Works**

1️⃣ Connects to all databases  
2️⃣ Extracts and standardizes metadata  
3️⃣ Detects schema changes  
4️⃣ Updates lineage and catalogs  
5️⃣ Provides governed access to metadata consumers

**🎨 Slide 8 – Roadmap**

* **Phase 1**: Multi-DB ingestion
* **Phase 2**: Business glossary integration
* **Phase 3**: Self-service metadata APIs
* **Phase 4**: Advanced lineage and impact simulation

**🎨 Slide 9 – Next Steps**

* Pilot deployment
* User training
* Integration with existing data governance tools

✅ **Tip:** If you want, I can help create **slide visuals and icons** for this deck.

**🟢 📘 User Guide for Data Stewards**

Below is **step-by-step guidance** you can publish in your wiki or internal documentation.

**🎯 Overview**

The **Schema Browser** is your central hub to explore, document, and govern data assets.

As a **Data Steward**, you are responsible for:

* Validating metadata accuracy
* Assigning business descriptions
* Managing access permissions
* Reviewing audit logs

**🟢 1️⃣ Logging In**

* Use your enterprise credentials.
* Role: **Data Steward**
* Permissions:
  + Read all metadata
  + Edit business descriptions
  + Assign classifications
  + View audit logs

**🟢 2️⃣ Browsing Metadata**

Navigate to **Schemas** to see:

* List of databases
* Tables and columns
* Data types
* Last modified timestamps

✅ Use **search filters** to quickly locate assets.

**🟢 3️⃣ Reviewing Business Metadata**

* Select a table.
* Open the **Business Metadata** tab.
* You can:
  + Add or edit descriptions.
  + Assign glossary terms.
  + Classify sensitivity level (e.g., Confidential).

**🟢 4️⃣ Exploring Lineage**

* Navigate to **Lineage Graph**.
* Review upstream and downstream data flows.
* Identify dependencies for impact analysis.

**🟢 5️⃣ Generating ER Diagrams**

* Open **ER Diagram Generator**.
* Select a schema or table.
* Export diagrams in PNG or SVG.

**🟢 6️⃣ Data Profiling**

* Access **Statistics Profiler**.
* Review:
  + Null counts
  + Unique value counts
  + Distribution samples

**🟢 7️⃣ Managing Access Control**

* Go to **Access Control**.
* Assign roles (Viewer, Editor) to users or groups.

**🟢 8️⃣ Reviewing Audit Logs**

* Navigate to **Audit Logs**.
* Filter by:
  + User
  + Action
  + Date range

✅ **Tip:** Regularly review logs and access controls to ensure compliance.

**🟢 🌐 API Reference Documentation**

Below is a **clean, concise API spec**.  
You can adapt it to OpenAPI or Swagger.

**🟢 Base URL**

arduino

CopyEdit

https://yourdomain.com/api/v1

**🟢 Authentication**

**Header:**

makefile

CopyEdit

Authorization: Bearer <token>

**🟢 Endpoints**

**1️⃣ List All Schemas**

**GET** /schemas

**Query Parameters:**

* database\_name (optional)
* table\_name (optional)
* page (optional)
* size (optional)

**Response:**

json

CopyEdit

[

{

"schema\_id": 1,

"database\_name": "PostgreSQL",

"schema\_name": "public",

"table\_name": "customer",

"columns": [

{"column\_name": "id", "data\_type": "Integer"}

]

}

]

**2️⃣ Get Schema Details**

**GET** /schemas/{schema\_id}

**3️⃣ Search Schemas**

**GET** /schemas/search?query=customer

**4️⃣ Get Business Metadata**

**GET** /schemas/{schema\_id}/business-metadata

**5️⃣ Update Business Metadata**

**POST** /business-metadata

**Body:**

json

CopyEdit

{

"schema\_id": 1,

"lob\_name": "Retail",

"application\_name": "Core Banking",

"description": "Customer master data",

"classification": "Confidential",

"glossary\_term": "Customer"

}

**6️⃣ Get Lineage**

**GET** /lineage/{schema\_id}

**7️⃣ Get Audit Logs**

**GET** /audit/{schema\_id}

**8️⃣ Get Access Control**

**GET** /access/{schema\_id}

**9️⃣ Update Access Control**

**PUT** /access/{schema\_id}

**Body:**

json

CopyEdit

{

"role\_name": "Editor",

"can\_read": true,

"can\_write": true,

"can\_delete": false

}

**🟢 Response Codes**

| **Code** | **Description** |
| --- | --- |
| 200 | Success |
| 201 | Created |
| 400 | Bad Request |
| 401 | Unauthorized |
| 404 | Not Found |
| 500 | Server Error |

✅ **Tip:** Enable pagination for large responses (page and size).

✅ **Security:** Use HTTPS and JWT tokens.