FLIP-60: Restructure the Table API & SQL documentation

Status

Discussion thread	https://lists.apache.org/thread/kly3zcm7p15lw4vwyl1jpx7bmnmyhtd6
Vote thread	
JIRA	FLINK-19524 - Google Season of Docs - FLIP 60 OPEN
Release	

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

The documentation for Table & SQL API has evolved over time. In the meantime, more streaming concepts, a SQL Client, connectors, catalogs etc. were added. Also the vision of a unified API for both batch and streaming is making progress.

It is time to rework the structure in order to:

- attract and target both pure SQL users and programming users
- find features more easily
- have a clear reading flow/order of topics

Public Interfaces

Not applicable.

Proposed Changes

This structure aims to get to the point quickly. It shows the features early to attract users and uses cross links e.g. to execution or concepts for interested readers.

The top-level pages are:

Overview *we start with a good overview*

Table API *get to the point, the features and operations*

SQL *get to the point, the features and operations*

Setup & Execution *explain the setup and how to execute features*

Connect to External Systems *details*

Built-in Functions *details*

Configuration *details*

User-defined Extensions *custom details*

Concepts *reference from other pages*

Detailed Structure

- **Table & SQL Ecosystem**
  Highest level. Next to Application Development because most of the current list siblings are not applicable to this section. The section is also rena med because SQL Client, connectors, and catalogs are not only an API.
  - **Overview**
    What is the Table & SQL Ecosystem?
    Might be a big page but is a nice executive summary and informative.
    - Main features like schema awareness, abstraction, connectors, catalogs etc.
    - How do we achieve unified data processing? Dynamic Tables
    - Quickly mention planners
    - Advantages/Disadvantages over DataStream API

        - E2E example for SQL
        - E2E example for Table API in Java/Scala/Python
        - E2E example for Table API in Java/Scala/Python with DataStream API
        - Short presentation of the SQL Client
    - **Table API**
      Goal: Show users the main table features early and link to concepts if necessary.
      How to use the API? Intended for users with programming knowledge.
        - **Overview**
            - Short getting started with link to more detailed execution section.
            - Explain the most important methods in unified Table Environment
            - Present sqlUpdate/sqlQuery etc.
            - Querying, Execution, Optimization internals behind the API
        - **Full Reference**
          Available operations in the API. This location allows to further split the page in the future if we think an operation needs more space without affecting the top-level structure.
            - Present the API operations
        - ***... more features in the future***
    - **SQL**
      Goal: Show users the main features early and link to concepts if necessary.
      How to use SQL? Intended for users with SQL knowledge.
        - **Overview**
          Getting started with link to more detailed execution section.
        - **Full Reference**
          Available operations in SQL as a table. This location allows to further split the page in the future if we think an operation needs more space without affecting the top-level structure.
        - **Data Definition**
          Explain special SQL syntax around DDL.
        - **Pattern Matching**
          Make pattern matching more visible.
        - ***... more features in the future***
    - **Setup & Execution**
        - **Programmatic**
          How to setup a project and submit a job?
            - Dependency structure for using the API
            - TableEnvironments
            - Features and limitations of table environments
            - How to setup Python projects?
        - **SQL Client**
          How to use the SQL Client? Docs for pure SQL users.
        - **… Notebooks, JDBC, and more executions in the future**
    - **Connect to External Systems**
      How to connect to other systems for data or metadata?
        - **Overview**
            - What are available sources and sinks?
            - What are catalogs? What can I manage with it?
        - **Available Connectors**
        - **Available Catalogs**
        - **Hive**
    - **Built-in Functions**
    - **Configuration**
    - **User-defined Extensions**
      Use API extension points.
        - **Sources & Sinks**
        - **Functions**
        - **Catalogs**
    - **Concepts**
      What are the general concepts (independent of API/SQL Client) any user should know about?
      We put this at the end and link from the main pages to pages here if necessary.
        - **Planners**
          What is a Planner? Temporary docs. Removed in the future.
            - Blink Planner Features and Limitations
            - Flink Planner Features and Limitations
        - **Data Types**
          Which data can we process?
        - **Unbounded Data Processing**
          Which operation needs special attention when working with unbounded data?
            - Dynamic Tables (with all update modes)
            - Time Attributes
            - Query Configuration
            - Joins in Continuous Queries
        - **Temporal Tables**
          Explain the concept of a temporal table.

Compatibility, Deprecation, and Migration Plan

We redirect old page links to the new structure.

Test Plan

Not applicable.

Rejected Alternatives

The first version was more like a walkthrough to guide the reader:

Getting Started
Overview
Planners
Concepts

Executing Programs
Table API

SQL Client

Data Types *we start with available data types*
Connect to External Systems *a program starts with connectors*
SQL *continue with operators*
Built-in Functions *and functions*

Configuration *finally we tweak the execution*

However, the problem here was that a user would read a lot of "boring" concept pages first and the actual list of operations would be nested in the Table API section and SQL section. The API sections should have highest priority and be presented as one of the first items in the list.