DATAΛAD

Типы данных, NULL и безопасный импорт

Платформенный контракт от источника до SQL и Runspace

Тип

Схема определяет смысл данных

NULL

Отсутствие не равно нулю

Импорт

Preview, проверка, материализация

Контракт

Один результат для всех контуров

Parquet • Grid • SQL • формулы • Runspace • пайплайны • экспорт

Для кого это руководство

Руководство предназначено для дата-инженеров, аналитиков и владельцев датасетов. Оно описывает, как DATAΛAD определяет, хранит и изменяет типы данных, как трактует пустые значения и какие решения необходимо принять во время импорта.

DATAΛAD — типизированная платформа данных. Табличный интерфейс является одним из способов работы с датасетом, но не отдельным источником истины. Физическая схема итогового Parquet-датасета задаёт единый контракт для:

- Свода данных (Grid);
- SQL-запросов и агрегатов;
- формул;
- Runspace и инструментов агента;
- пайплайнов и ETL;
- слияний, сверок и профилирования;
- экспорта и внешних потребителей.

Одинаковое значение должно иметь одинаковый смысл во всех этих контурах. Поэтому DATAΛAD не подменяет `NULL` нулём только при отображении и не пытается «угадать» неоднозначный тип уже после загрузки.

1. Тип — это контракт датасета

Тип столбца определяет не только его внешний вид. От него зависят:

- допустимые операции и формулы;
- правила сравнения, сортировки и фильтрации;
- семантика агрегатов;
- поведение соединений и сверок;
- планирование SQL-запросов;
- физическое представление и сжатие в Parquet;
- совместимость с BI-, ETL- и ML-инструментами.

Например, строки `"2"`, `"10"` и `"100"` можно показать как числа, но у текстового столбца сортировка и сравнение останутся строковыми. Для платформы важно не то, как ячейка выглядит на экране, а какой тип зафиксирован в схеме.

Основные типы в интерфейсе импорта

Тип в DATAΛAD	Физический смысл	Типичные данные
Текст	`Utf8` / `VARCHAR`	коды, названия, произвольные строки
Целое число	целочисленный Arrow/Parquet type	количество, номер, счётчик
Число	`Float` / `DOUBLE` или decimal-compatible input	сумма, коэффициент, измерение
Да / нет	`Boolean`	признак, флаг, результат проверки
Дата	`Date32` / `DATE`	календарная дата без времени
Дата / время	`Timestamp`	момент события

Сложные типы Parquet могут встречаться в платформе, но базовый интерфейс импорта намеренно предлагает небольшой и однозначный набор типов. Это снижает риск случайно создать схему, которую трудно использовать в SQL, Grid или других инструментах.

2. Как проходит импорт

Импорт — это не простое отображение исходного файла. Это контролируемое создание нового типизированного датасета.

1. DATAΛAD читает источник и определяет его структуру.
2. Для CSV или табличного листа определяется строка заголовка и набор столбцов.
1. Детерминированные правила предлагают тип каждого столбца.
2. Preview показывает результат до записи датасета.
3. Пользователь может переименовать столбцы и явно изменить предложенные типы.
1. При необходимости выбирается политика для пустых числовых и логических значений.
1. Итоговая схема и значения атомарно материализуются в Parquet.
2. Все инструменты платформы читают уже один и тот же физический результат.

Исходный CSV или XLSX при этом не изменяется. Повторный импорт можно выполнить с другой схемой или другой политикой пустых значений и получить отдельный датасет.

Что происходит с готовым Parquet

Если источником является уже сформированный Parquet, его физическая схема и значения являются входным контрактом. Политики заполнения пустых значений предназначены прежде всего для импорта источников, которые преобразуются в новый Parquet, а не для скрытого переписывания зарегистрированного типизированного файла.

3. Автоматическое определение типов

Автоматическое определение типов выполняется детерминированно, без LLM в контуре записи данных. Одинаковые входные байты и настройки дают одинаковую схему и одинаковый результат.

Основные правила

- Чистые целые значения становятся целыми числами.
- Числа с дробной частью становятся числами.
- Пустые значения сами по себе не мешают определить числовой или логический

тип.

- Строгие значения `true` и `false` распознаются как boolean.
- Однозначные ISO-даты распознаются как даты.
- RFC 3339 и поддерживаемые однозначные представления даты-времени

распознаются как timestamp.

- Числа с распространёнными разделителями, например `244 152,63` или

`1,234.56`, могут быть нормализованы и типизированы, только если весь непустой столбец проходит проверку.

- Смешанный столбец остаётся текстом, если хотя бы одно существенное значение

не удовлетворяет безопасному правилу преобразования.

Что DATAΛAD намеренно не угадывает

- `01/02/2026`, если нельзя однозначно определить день и месяц;
- `да/нет`, `yes/no` и доменные сокращения без явного выбора boolean;
- идентификаторы, похожие на числа, если их текстовая форма существенна;
- смешанные числа, даты и произвольный текст;
- значения, для которых преобразование потребовало бы недокументированной

эвристики.

Такое значение остаётся текстом. Пользователь может явно выбрать другой тип в Preview, но тогда DATAΛAD покажет последствия преобразования.

Почему правило консервативное

Ошибочно оставить столбец текстом можно исправить явным изменением схемы. Ошибочно превратить код, неоднозначную дату или специальное значение в число гораздо опаснее: исходная семантика может быть потеряна без заметной ошибки.

4. Ручной выбор типа при импорте

В Preview можно оставить автоматически предложенный тип или выбрать:

- текст;

- целое число;
- число;
- да / нет;
- дату;
- дату / время.

Ручной выбор является явным решением владельца данных и имеет приоритет над автоматическим предложением.

Безопасное преобразование

Преобразование выполняется через безопасный cast:

- совместимое значение записывается в выбранном типе;
- действительно пустое значение обрабатывается согласно выбранной политике;
- непустое несовместимое значение не превращается в произвольное значение;
- несовместимое значение становится `NULL`;
- количество таких потерь показывается в предупреждении импорта.

Пример:

Исходное значение	Целевой тип	Политика	Результат
`"125"`	целое число	любая	`125`
пусто	целое число	сохранить `NULL`	`NULL`
пусто	целое число	записать `0`	`0`
`"не указано"`	целое число	записать `0`	`NULL` + предупреждение
`true`	да / нет	любая	`true`
пусто	да / нет	записать `false`	`false`
`"неизвестно"`	да / нет	записать `false`	`NULL` + предупреждение

Ключевое правило: политика пустых значений применяется только к действительно пустому входу. Она не маскирует ошибки качества данных.

5. `NULL`, ноль, `false` и пустая строка — разные значения

Значение	Семантика
`NULL`	значение отсутствует, неизвестно или неприменимо
`0`	известно, что числовое значение равно нулю
`false`	известно, что логическое утверждение ложно
`""`	известная строка нулевой длины

Эти состояния нельзя безусловно взаимозаменять.

Влияние на вычисления

- `COUNT(*)` считает строки независимо от `NULL`.
- `COUNT(column)` не считает строки, где `column IS NULL`.
- `SUM` и `AVG` обычно пропускают `NULL`.
- Ноль участвует в `SUM` и `AVG`.
- `false` участвует в логических вычислениях как известное ложное значение.
- Сравнение с `NULL` требует `IS NULL` / `IS NOT NULL`.
- `NULL` в ключе соединения обычно не совпадает с другим `NULL`.

Поэтому визуально незаполненная ячейка не должна автоматически становиться нулём на уровне Grid. Иначе Grid, SQL, Runspace и экспорт могли бы показывать разную картину одного датасета.

Если для конкретного расчёта бизнес-правило трактует отсутствие как ноль, это можно выразить явно в SQL, например через `COALESCE`, не меняя исходную семантику всего столбца.

6. Политика пустых значений при импорте

В блоке **«Пустые типизированные ячейки»** доступны две независимые опции. Обе выключены по умолчанию.

Пустые числа

- **Сохранить как** `NULL` — поведение по умолчанию.
- **Записать** `0` **в итоговый датасет** — опциональная материализация нуля.

Опция применяется к автоматически распознанным числовым столбцам и к столбцам, которые пользователь вручную переводит в целое число или число.

Пустые «да / нет»

- **Сохранить как** `NULL` — поведение по умолчанию.
- **Записать** `false` **в итоговый датасет** — опциональная материализация

ложного значения.

Опция применяется к автоматически распознанным boolean-столбцам и к столбцам, которые пользователь вручную переводит в тип «да / нет».

Пустой текст

Для строковых столбцов автоматической замены `NULL` на пустую строку нет. `NULL` означает отсутствие значения, а `""` — известную строку нулевой длины. DATAΛAD сохраняет это различие для фильтров, профилирования, SQL и контроля качества.

Область действия

- Политика действует только на текущий импорт.
- Она не является настройкой отображения Grid.
- Она не меняет исходный файл.
- Она не переопределяет поведение уже созданных датасетов.

- Выбранное значение записывается физически в итоговый Parquet.
- После импорта Grid, SQL, формулы, Runspace, пайплайны и экспорт видят один

и тот же результат.

Почему по умолчанию сохраняется NULL

Отсутствие данных нельзя универсально трактовать как ноль или `false`. Безопасный платформенный default должен сохранять исходную неопределённость, а доменное решение о подстановке принимает владелец данных при импорте.

7. Как выбрать правильную политику

Сценарий	Рекомендуемая политика	Причина
Сумма платежа не заполнена	обычно `NULL`	неизвестно, был ли платёж нулевым
Счётчик событий, где отсутствие по контракту означает «не было»	`0` допустим	домен явно определяет отсутствие как ноль
Результат проверки не поступил	`NULL`	`false` ошибочно означал бы провал проверки
Флаг «отключён», где пустое поле по контракту означает «нет»	`false` допустим	это зафиксированное правило источника
Пропуск телеметрии	`NULL`	ноль является измерением, а не пропуском
Неуказанная дата	`NULL`	для даты нет корректного нулевого значения
Идентификатор с ведущими нулями	текст	числовой тип разрушит формат идентификатора

Практическое правило:

> **Включайте замену только тогда, когда контракт источника явно утверждает: «пусто» эквивалентно `0` или `false`.**

Если такого утверждения нет, сохраняйте `NULL`.

8. После импорта: изменение типа столбца

Если проблема обнаружена после создания датасета, тип можно изменить через контролируемый анализ преобразования в Grid. DATALAD предварительно показывает:

- исходный и целевой тип;
- количество значений, которые не удастся преобразовать;
- примеры теряемых значений;
- зависимые формулы;
- блокирующие зависимости, например формульный столбец или ключ активного

слияния.

Преобразование с потерями требует явного подтверждения. Для исправления систематической ошибки входного контракта предпочтительнее повторный импорт с правильной схемой: так решение остаётся воспроизводимым и не зависит от ручной операции после загрузки.

9. Типы в формулах, SQL и автоматизации

Формулы

Формула работает с физическими типами столбцов. `NULL` не должен неявно превращаться в ноль для всех формул: это изменило бы смысл исходных данных. Если конкретная формула требует нулевой подстановки, правило необходимо выразить в самой формуле или подготовить данные на этапе импорта/ETL.

SQL

Типизированная схема позволяет:

- использовать числовые и временные функции без повторного cast;
- строить корректные фильтры и соединения;
- явно контролировать `NULL`;
- получать предсказуемые планы запросов.

Runspace, агент и пайплайны

Автоматизация получает ту же схему, что и человек в Grid. Это позволяет агенту и детерминированным инструментам рассуждать о столбце как о числе, дате или boolean, не извлекая тип повторно из визуального представления.

LLM может рекомендовать преобразование, но не определяет физическую схему в горячем контуре импорта. Запись выполняется детерминированным кодом и проверяемыми правилами.

10. Почему платформенный подход лучше

Единая семантика

Нет отдельного «режима Grid», в котором `NULL` выглядит как ноль, пока SQL и экспорт видят другое значение.

Воспроизводимость

Один источник и один набор параметров импорта дают одинаковый датасет. Пайплайны можно повторять, кэшировать и проверять.

Контроль качества

Несовместимые непустые значения не маскируются нулями. Они становятся наблюдаемой проблемой импорта с количеством потерь.

Производительность

Корректные физические типы улучшают колоночное хранение, сжатие, predicate pushdown и выполнение агрегатов по сравнению с хранением всего как текста.

Интероперабельность

Parquet со стабильной схемой одинаково понимают SQL-движок, BI, ETL, Python, Spark-совместимые процессы и другие потребители.

Безопасная автоматизация

Инструментам не нужно каждый раз угадывать тип по образцу значений. Автоматическое действие можно валидировать относительно известной схемы.

11. Антипаттерны

Не рекомендуется:

- хранить все столбцы как текст «на всякий случай»;
- глобально считать любой `NULL` нулём;
- превращать несовместимый непустой текст в `0` или `false`;
- угадывать неоднозначные даты без заданного формата;
- смешивать идентификаторы и измерения в одном числовом типе;
- менять только визуальное представление, не меняя физическую схему;
- использовать LLM как недетерминированный cast в контуре записи.

12. Чек-лист дата-инженера

Перед импортом

- Определите, является ли столбец измерением, идентификатором, датой или признаком.
- Проверьте разделители чисел и формат дат.
- Уточните у владельца источника смысл пустых значений.
- Не включайте нулевую подстановку без доменного контракта.
- Просмотрите предложенные типы и предупреждения Preview.

После импорта

- Проверьте физические типы ключевых столбцов.
- Сверьте число строк.
- Проверьте число значений, ставших `NULL` при cast.
- Отдельно посчитайте `NULL`, нули и `false`.
- Проверьте агрегаты и ключи соединения.
- Зафиксируйте выбранную политику в документации пайплайна или источника.

13. Параметры API

Политика пустых значений передаётся как часть запроса импорта.

Сохранить безопасное поведение по умолчанию:

```
{
  "empty_values": {
    "numeric": "preserve_null",
    "boolean": "preserve_null"
  }
}
```

Материализовать ноль и `false`:

```
{
  "empty_values": {
    "numeric": "zero",
    "boolean": "false"
  }
}
```

Явное переопределение типа столбца в поддерживаемом import flow:

```
{
  "column_types": [
    { "column": "quantity", "type": "int" },
    { "column": "amount", "type": "float" },
    { "column": "approved", "type": "bool" },
    { "column": "event_date", "type": "date" }
  ]
}
```

Допустимые friendly type identifiers:

```
text, int, float, bool, date, timestamp
```

Если `empty_values` не передан, DATAΛAD сохраняет `NULL`. Неизвестные поля политики отклоняются, чтобы опечатка не изменила смысл импорта незаметно.

Краткое правило

DATAΛAD хранит тип и значение один раз — в датасете. Автоматическое определение консервативно, ручное преобразование проверяемо, `NULL` не подменяется молча, а доменная замена на `0` или `false` выполняется явно во время импорта.