C1. DataLayer.OHLCV — постановка задачи (детализация)

Версия: 1.0
Дата: 2025-10-15
Назначение: надёжная **загрузка, ресемплинг и хранение** OHLCV-рядов. Базовый ТФ **1m**, производные ТФ **5m/15m/1h**. Хранение в колоночном формате **Parquet** с метаданными, кэшем и идемпотентным догрузом истории. Все времена — **UTC**.

> **Входы:** API Bybit (кастодиальные ключи *read-only*), список тикеров.
> **Выходы:** `/data/{symbol}/{tf}.parquet` со схемой: `ts, o, h, l, c, v` *(обязательные)*; опционально `t` *(turnover)* и служебные флаги.
> **NFR:** пропуски `< 0.01%`, валидность таймстампов, задержка обновления `≤ 1 бар`.
> **DoD:** юнит-тесты на ресемплинг/валидацию; интеграция на 5 тикерах ≥1 год; отчёт о пропусках.
> **Риски:** лимиты API, редкие делисты, смещение времени.

---

1. Результат

- Консолидированные Parquet-файлы per `(symbol, tf)` с корректным календарём баров без дубликатов.
- Кэш «хвоста» (последние N дней) для быстрых чтений и ресемплинга.
- Сервис **DataReader** с API для выборок по времени.
- Отчёт **missing-report** по каждому `(symbol, tf)` и итоговая сводка по пропускам.

---

2. Схемы данных

2.1 Паркет-схема (минимальная)

```
ts: int64         # UNIX ms, начало бара (UTC)
o: float64
h: float64
l: float64
c: float64
v: float64        # объём в базовой единице инструмента
```

2.2 Расширения (опционально)

```
t: float64        # turnover (денежный оборот), если доступен из API
ver: int32        # версия записи (ревизии источника)
```

```
is_gap: bool      # вставленный «пустой» бар вместо пропущенного (см. §6)
source: string    # bybit/…
```

Инварианты: `l ≤ min(o,c) ≤ max(o,c) ≤ h`, `v ≥ 0`, монотонность `ts` с шагом=`tf_ms`.

---

3. Архитектура и компоненты

- **Fetcher**: идемпотентная загрузка 1m-баров из Bybit с пейджингом и rate-limit/backoff.
- **Resampler**: конвейер агрегации 1m → 5m/15m/1h (OHLCV-правила).
- **Deduper & Calendarizer**: дедуп по `(ts)` и выравнивание календаря (вставка `is_gap` баров).
- **Writer**: запись Parquet с `ZSTD` и стрим-слиянием чанков.
- **Tail Cache**: быстрый кэш последнего сегмента для чтения/ресемплинга.
- **DataReader**: API/CLI для выборок и сверок.

Хранилище каталогов:

```
/data/{symbol}/
  1m.parquet
  5m.parquet
  15m.parquet
  1h.parquet
  tail/
    1m.tail.parquet    # последние ~14 дней
```

---

4. Инкрементальная загрузка (идемпотентный бэкфилл)

1) Определить `ts_from` = (последний `ts` в локальном 1m + `tf_ms`) либо `epoch_start` при отсутствии.
2) Запрашивать у API диапазонами `[ts_from, ts_to)` страницами (по N баров) с **end-exclusive**.
3) На каждый батч: нормализовать в UTC, проверить шаг минут, отбросить будущие бары.
4) Смерджить с локальным хвостом (см. §7), выполнить **дедуп** по ключу `ts` с правилом *последняя запись выигрывает* и инкрементом `ver`.
5) Записать чанки в `1m.parquet` (append) и обновить `tail`.
6) Запустить **Resampler** для производных ТФ только на новых минутах.

Rate-limit/backoff: экспоненциальный с джиттером; ограничение параллелизма по тикерам (напр., `max_concurrent=2`).

---

5. Ресемплинг (1m → 5m/15m/1h)

Выравнивание на границы ТФ (UTC). Для окна минут `W = {t0,…,t_{k-1}}`: - `o = open(t0)`
- `h = max(h_i)`
- `l = min(l_i)`

- `c = close(t_{k-1})`
- `v = sum(v_i)`
- `(t)` если присутствует: `sum(t_i)`

Правила: - Если внутри окна есть `is_gap` минуты — бар помечается `is_gap=true`.
- Если весь бар пропущен (нет минут) — в календарь пишется `is_gap=true` с `o=h=l=c=prev_close`, `v=0`.
- Запрещены частично «незаполненные» окна без флага `is_gap`.

---

6. Выравнивание календаря и пропуски

- **Календарь**: равномерная сетка по `tf` в UTC от первого полного бара до «вчера/последнего закрытого».
- **Пропуск минуты**: вставить бар с `is_gap=true`, `o=h=l=c=prev_close`, `v=0`; попытаться дозагрузить при следующем запуске (до окна `gap_recovery_days`).
- **Норма пропусков**: итоговая доля `is_gap` по `(symbol, tf)` $\leq$ `0.01%`; иначе WARNING в отчёте и флаг для оркестрации.

---

7. Запись/слияние Parquet

- **Формат**: Parquet, компрессия `ZSTD level=7`, размер row-group `~256K строк`.
- **Слияние**: новые чанки мёрджатся с хвостом (последние `M` минут) в памяти и пишутся **атомарно** (временный файл → rename).
- **Мини-метаданные**: `data_hash` (SHA256 файла), `build_signature` (C15), `source=bybit`, `generated_at`.
- **Индексы**: min/max `ts` в footer для быстрых выборок.

---

8. Кэш и латентность

- `tail/*.parquet`: последние `N_days_tail` (напр., 14 дней для 1m).
- Обновление кэша синхронно при догрузе.
- Цель: задержка `≤ 1 бар` между закрытием минуты и появлением в локальном 1m/5m/15m/1h.

---

9. Валидация качества

- **Синтаксис:** типы колонок, отсутствие NaN/Inf.
- **Хронометрия:** монотонность `ts`, шаг = `tf_ms`, отсутствие будущих дат.
- **OHLC-инварианты:** `l≤min(o,c)≤max(o,c)≤h`.
- **Объём:** `v≥0`, при ресемплинге `v_sum` = сумма минут.
- **Календарь:** доля `is_gap` и список непрерывных «проблемных» интервалов.
- **Сверка источника:** случайная выборка `k` минут сверяется напрямую с API.

---

10. Интерфейсы

10.1 Python API

```
from datalayer import DataReader

# Вернёт DataFrame за [start, end) UTC, с колонками по схеме ТФ
DataReader(symbol: str, tf: str).read(start: str|int, end: str|int) ->
pd.DataFrame

# Функции обслуживания
backfill(symbols: list[str], since: str|None = None) -> None
resample(symbols: list[str], tfs: list[str] = ["5m","15m","1h"]) -> None
validate(symbols: list[str], tfs: list[str]) -> ValidationReport
missing_report(symbols: list[str], tfs: list[str]) -> pd.DataFrame
```

10.2 CLI

```
datalayer backfill --symbols BTCUSDT,ETHUSDT --since 2023-01-01
datalayer resample --symbols BTCUSDT,ETHUSDT --tfs 5m,15m,1h
datalayer validate --symbols BTCUSDT,ETHUSDT --tfs 1m,5m,15m,1h --out
reports/validate.json
datalayer missing-report --symbols ALL --tfs 1m,5m,15m,1h --out reports/
missing.csv
```

Коды: `0` — успех; `E_API`, `E_RATE_LIMIT`, `E_SCHEMA`, `E_TIME_DRIFT`, `E_WRITE`.

---

11. Конфигурация (пример)

```
api:
  adapter: bybit
  base_url: "https://api.bybit.com"
  read_only_keys_ref: "secret://bybit/{{ env }}/read_key"
  timeout_s: 10
  max_retries: 5
  max_concurrent: 2
  page_size: 1000
storage:
  base_dir: "/data"
  compression: "zstd"
  tail_days: 14
resample:
  tfs: ["5m","15m","1h"]
  strict_gaps: true
quality:
  max_gap_pct: 0.0001    # 0.01%
```

```
  gap_recovery_days: 7
  sanity_check_samples: 200
logging:
  level: INFO
```

---

12. Производительность и ресурсы

- Цель: загрузка **1 года 1m** по 1 тикеру ≤ 15 мин при `page_size=1000` и `max_concurrent=2`.
- Ресемплинг 1 год → `5m/15m/1h` ≤ 2 мин/ТФ.
- Память: ≤ 1.5× размера хвоста; запись чанков батчами.

---

13. Отчёт о пропусках

Файл CSV/Parquet:

```
symbol, tf, ts_from, ts_to, gaps_pct, gaps_count, longest_gap_bars
```

В сводке: топ-N проблемных интервалов, рекомендации (повторная догрузка, смена окна, проверка делиста/переименования).

---

14. Тесты и DoD

Юнит: - Ресемплинг: корректность OHLCV-агрегации на синтетике и референтных фикстурах.
- Валидация: инварианты OHLC, календарь, NaN-гарды.
- Дедуп: при повторных загрузках выигрывает последняя версия, `ver` инкрементируется.

Интеграция: - Догруз истории на ≥ 5 тикерах (например, BTCUSDT, ETHUSDT, SOLUSDT, XRPUSDT, LINKUSDT) за ≥ 365 дней.
- Сверка случайных 200 минут с API; отчёт о пропусках `gaps_pct < 0.01%`.
- Латентность обновления ≤ 1 бар в ходе «хвостовой» догрузки.

DoD: - Все тесты зелёные; отчёты `validate.json` и `missing.csv` присутствуют.
- Файлы `/data/{symbol}/{tf}.parquet` корректны, содержат `data_hash` и `build_signature`.
- Оркестратор (C18) добавлен job `datalayer backfill` (ежечасно хвост + ночной бэкаповый прогон).

---

15. Риски и антипаттерны

- **API-лимиты:** использовать backoff, детерминировать порядок тикеров; при `429` — экспоненциальная пауза.

- **Делисты/переименования:** маппинг символов; при делисте — freeze файл и завершить календарь.
- **Смещение времени:** сверять серверное время и UTC; алерт при `|drift| > 500 мс`.
- **Нестабильные цены/неконсистентность:** отбрасывать бары с `h<max(o,c)` или `l>min(o,c)`, логировать как `E_SCHEMA`.

---

16. Псевдокод конвейера

```
for symbol in symbols:
  ts_from = last_ts_local(symbol, '1m') + 60_000 or epoch_start
  while ts_from < now_utc_minus_1m:
    batch = fetch_api(symbol, '1m', ts_from, limit=page_size)
    batch = normalize_utc(batch)
    batch = filter_future_minutes(batch)
    merged = merge_dedup_tail(symbol, batch)
    write_parquet_append(symbol, '1m', merged)
    update_tail_cache(symbol, merged)
    resample_and_write(symbol, merged, tfs=['5m','15m','1h'])
    ts_from = merged.last_ts + 60_000

validate_all(); emit_missing_report()
```