Week06 | 课时 3 | Partition / Backfill / Replay：把补数做成可控动作

Table of contents

补数不是“再跑一遍”，而是对明确边界的受控重算

这一讲把 Week03 的 replay/backfill 思维升级成 Week06 的 asset-level partition recovery。

进入课时 4 返回 Week06 总览

下载讲义

提供适合离线阅读的 PDF 版和适合批注整理的 Word 版。

PDF 版 · 打印 / 离线阅读 Word 版 · 批注 / 二次整理

这节课解决什么问题

如果没有 partition boundary，backfill 很容易变成“全量重跑一次试试”。这在 AI 数据链路里很危险：

- 可能重复写入 raw / silver
- 可能覆盖已经发布的 baseline

- 可能让下游指标和索引版本漂移
- 可能无法解释补数影响范围

参考学习时间

50–60 分钟

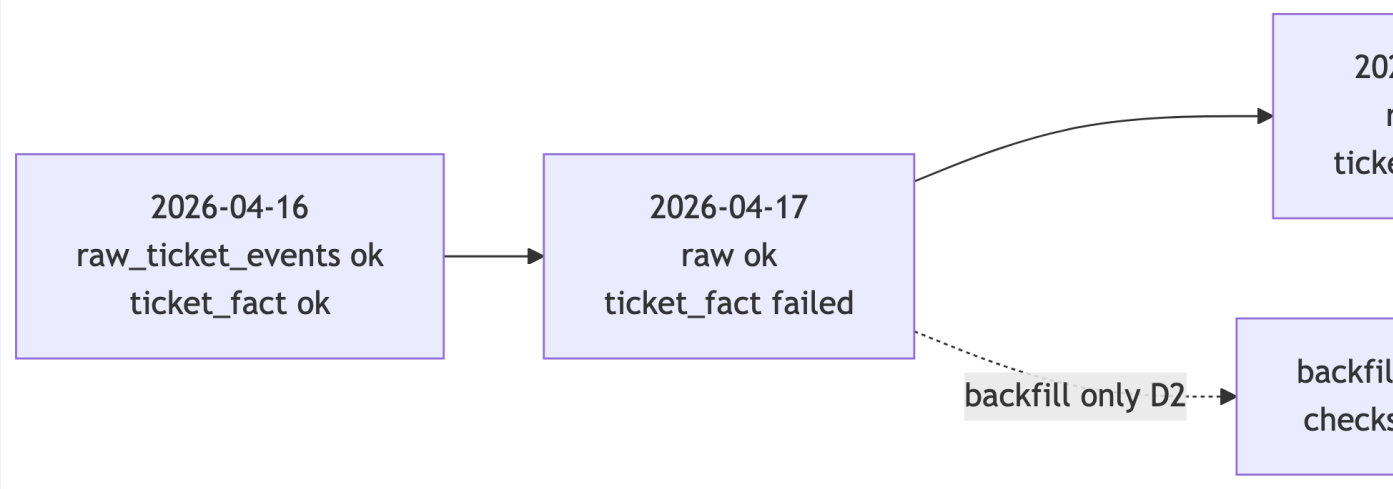
学完这一讲，你应该能做到什么

1. 说明 daily partition 为什么是 Student Core 默认。
2. 区分 retry、rerun、replay、restore、backfill。
3. 写出一个 partition-level backfill plan。
4. 解释 backfill 后为什么必须跑 checks 和 evidence。
5. 设计安全缺口模拟方式，而不是危险删除主表。

本课产出

- docs/blueprints/week06/week06-partition-backfill-strategy.md
- reports/week06/backfill/

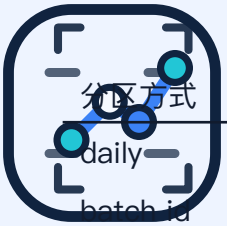
先看一张时间线



核心判断是：**只补失败边界，不盲目全量重跑。**

1. 为什么先用 daily partition

Student Core 默认 daily partition，是因为它最容易解释、最容易回放、最容易和业务时间对齐。



分区方式	适合场景	Week06 判断
daily	大多数批处理、指标和报告	默认选项
batch id	一次 manifest / batch 边界非常清晰	写进 metadata，必要时辅助排查
manifest id	需要严格重放某次输入声明	适合 replay report
product line	多业务线隔离	后续扩展，不做 Student Core 默认
dynamic / multi partitions	大规模生产编排	先讲边界，不作为本周必做

在当前 omnisupport-copilot Week6 项目实现里，课堂默认分区是 2026-04-17，环境变量落点是：

```
# 可执行：已与项目 runbook 对齐
WEEK06_PARTITION_START_DATE=2026-03-01
WEEK06_DEFAULT_PARTITION=2026-04-17
WEEK06_REPORT_DIR=reports/week06
```

怎么看输出：如果你的 dry-run plan 不是围绕 2026-04-17 生成，先检查 .env.local 是否从 infra/env/.env.example 同步过。

2. 五个恢复动作不要混用

动作	什么时候用	关键边界
retry	同一次执行里的瞬时失败	不改变输入和分区，只重试同一动作
rerun	同一作业定义重新跑	仍需保证幂等，不能绕过 checks
replay	重放同一批或同一来源输入	需要 manifest / batch 证据
backfill	补历史空洞或旧分区	需要明确 partition window
restore	当前状态不可信，先回到可用状态	需要 snapshot / checkpoint

⚠ 没有 partition boundary，补数只能靠经验

如果你说不清 window_start、window_end、upstream_manifest_id 和 idempotency_guard，那就还没有进入 backfill，只是在赌一次重跑。

3. Backfill plan 最少包含什么

```
{
  "partition_key": "2026-04-17",
  "window_start": "2026-04-17T00:00:00+00:00",
  "window_end": "2026-04-17T23:59:59.999999+00:00",
  "mode": "dry-run",
  "upstream_manifest_id": "week06-seed-manifest-001",
  "expected_input_count": 3,
  "current_output_count": 0,
  "gap_reason": "missing_partition_report",
  "proposed_action": "dry_run_ticket_ingest_for_partition",
  "idempotency_guard": "ticket_id + partition_key",
  "downstream_impact": "hold Week07/Week08 consumption until checks pass",
  "operator": "student-devbox",
  "target_assets": [
    "week06/ingestion/raw_ticket_events_partitioned",
    "week06/silver/ticket_fact_partitioned"
  ],
  "report_path": "reports/week06/backfill/backfill_plan_2026-04-17.json"
}
```

这段是示意结构，字段名以 contracts/run_evidence/week06_run_evidence.schema.json 和 pipelines/data_factory/backfill_plan.py 的真实实现为准。

4. 安全制造缺口

实验里不要危险删除主表。更稳的方式是：

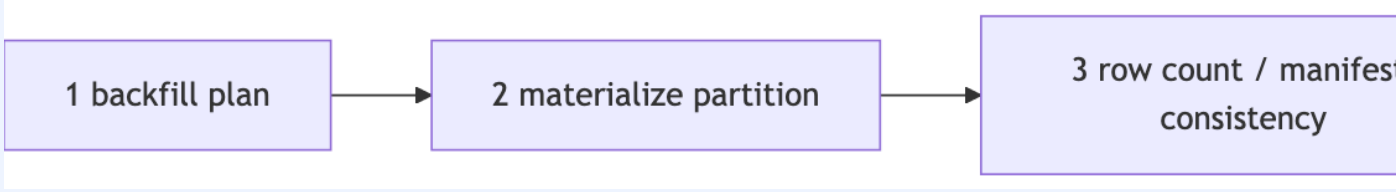
- 删除或移动某个 partition report
- 使用 dry-run 模式只生成计划不写下游
- 构造一份明显缺字段的 seed manifest
- 让某个 check 失败，但不破坏长期数据

项目侧 dry-run 命令已经落地：

```
# 可执行：已与项目 runbook 对齐
docker compose --profile tools --env-file infra/env/.env.local -f infra/docker-compose.yml run --rm devbox \
  python -m pipelines.data_factory.backfill_plan --partition 2026-04-17 --mode dry-run
```

常见错误：如果输出提示 `Unsupported Week06 backfill mode`，检查 `--mode` 是否写成了 `dry-run`；如果 expected input 为 0，先确认 seed 数据中是否真的有 `2026-04-17` 的记录。

5. 补数后的验收顺序



> ! 核心判断
>
> **没有 checks 和 evidence 的 backfill，只是又跑了一遍；有边界、有验证、有证据的 backfill，才是工程恢复动作。**

自检清单

- ☐ 我能解释为什么不默认全量重跑
- ☐ 我能区分 retry / rerun / replay / backfill / restore
- ☐ 我能写出一个 dry-run backfill plan
- ☐ 我知道安全缺口模拟不能破坏主表
- ☐ 我知道 backfill 后必须进入 checks 和 evidence

课后最小行动

补一份恢复策略：

```
## Week06 recovery strategy

- 默认 partition:
- 允许 backfill 的资产:
- 禁止直接删除的对象:
- dry-run report path:
- backfill 后必须跑的 checks:
```