Week05｜课时 1｜为什么 Week5 不是“做几个指标”：把业务口径变成工程接口

Table of contents

先把“指标 = SQL 结果”的直觉拆掉

Week05 的第一件事不是打开编辑器写 KPI SQL，而是先确认：

业务口径如果不能被测试、文档化、审计和工具调用，它就还不是工程资产。

进入课时 2 返回 Week05 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

Week04 已经让数据具备状态记忆。到了 Week05，真正的问题变成：

> **同一份数据被 BI、运营报表、Agent、评测和治理消费时，大家到底是不是在算同一个东西？**

如果答案是“不一定”，那问题通常不在 SQL 语法，而在口径没有成为工程接口。

参考学习时间

建议按一节标准课安排：先读完正文，再写出一个 OmniSupport 指标接口卡片。

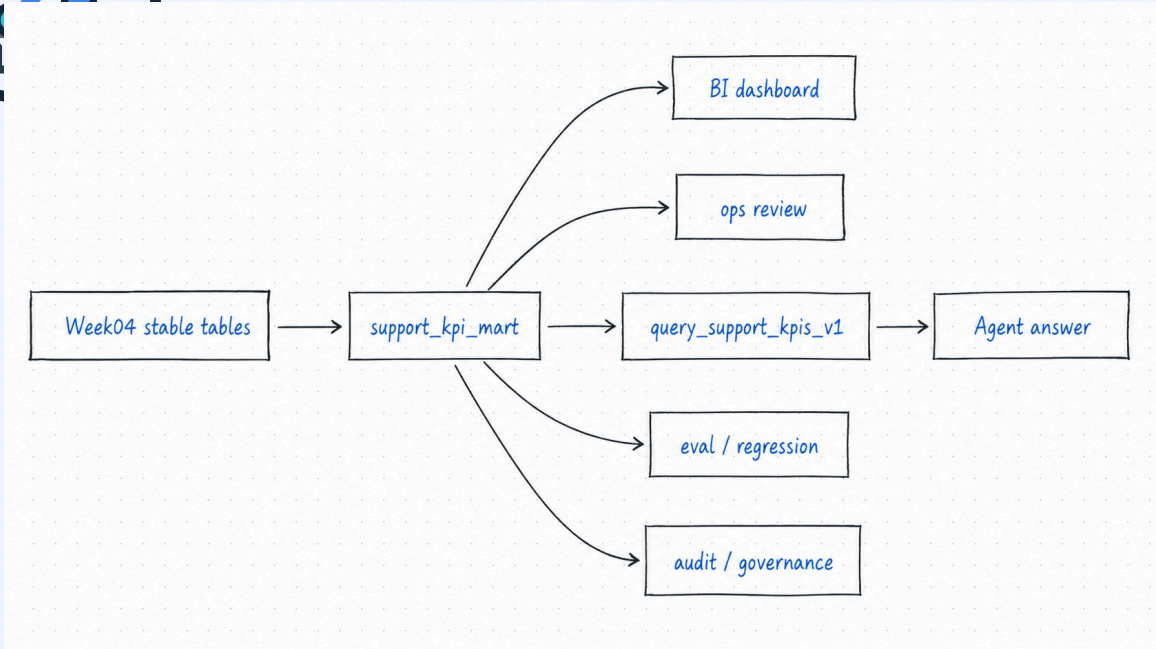
学完这一讲，你应该能做到什么

1. 解释什么是指标、业务口径和工程接口。
2. 说明 dashboard SQL 里的指标为什么不等于可复用指标。
3. 用真实客服运营 bad case 拆解口径冲突。
4. 写出一个最小指标接口卡片。
5. 解释为什么 BI、运营、Agent、评测和治理必须共用同一套口径。

本课产出

- docs/blueprints/week05/adr-week5-analytics-path.md
- docs/blueprints/week05/metric-interface-principles.md
- 一个 p1_ticket_count 或 sla_breach_count 的指标接口卡片草稿

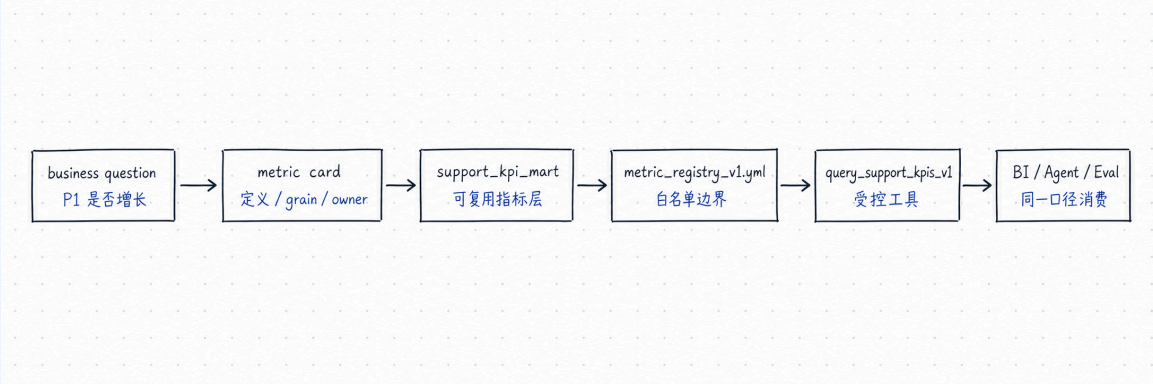
先看一张总图



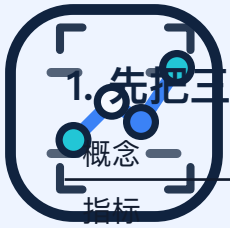
这张图的重点不是“多了一层 SQL”，而是：

- Week04 提供稳定表状态；
- Week05 把业务口径写进 support_kpi_mart 和 metric registry；
- BI 与 Agent 都从同一套受控口径出口取数。

本课路线：指标接口怎么进入系统



这条线把“业务问题”变成“工程接口”：先写清指标卡，再落到 mart，再进入 registry，最后由受控工具提供给 BI、Agent 和评测。



1. 先把三个词讲清楚

概念	小白版解释	OmniSupport 例子
指标	团队用来判断业务状态的数字	p1_ticket_count、sla_breach_count、avg_backlog_age_days
业务口径	这个数字到底怎么算、算谁、不算谁	P1 是 priority in ('p1', 'p1_critical')，时间用 created_date
工程接口	口径进入系统后的稳定消费边界	support_kpi_mart + metric_registry_v1.yml + query_support_kpis_v1

一句 SQL 可以算出数字，但接口还要回答：

- 来源模型是谁；
- 一行代表什么粒度；
- 允许哪些维度和过滤器；
- 谁负责口径；
- 哪些测试保护它；
- Agent 是否有权限查询；
- 查询结果如何审计和复盘。

> 第一次学指标，先问 grain，再问 SQL
>
> grain 是“一行到底代表什么”。如果 grain 没讲清，SQL 即使能跑，也可能把 ticket、event、comment 或 day 混在一起算。

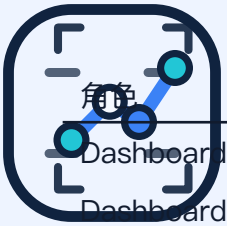
2. 一个真实风格的 bad case

运营问：

> “上周 P1 工单增长了吗？”

Dashboard A 用 created_at 统计新建工单；Dashboard B 用 updated_at 统计活跃工单；Agent 为了回答问题直接查 raw event，把 reopen 也算成新工单。运营不知道该相信谁，技术团队最后发现：三条 SQL 都没报错。

三个结果都能跑出数字，但没有一个能被团队放心使用。



角色	查了什么	得到什么
Dashboard A	按 created_at 算 P1 新建工单	新建 P1 数量
Dashboard B	按 updated_at 算 P1 活跃工单	最近被更新过的 P1 数量
Agent	直接查 raw event，把 re-open event 当新工单	数字更高，看起来更紧急
运营	对比三个数字	不知道该相信哪一个
技术团队	检查 SQL 运行状态	三条 SQL 都没报错

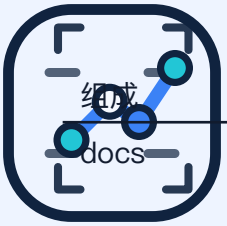
! 本课核心判断

SQL 能跑，只能证明语法和连接没坏；不能证明业务口径可负责。[1]

3. 指标的内涵与外延

组成	小白版解释	OmniSupport 例子	缺失会怎样
名称	机器和人都能稳定识别它	p1_ticket_count / P1 工单数	同名不同义，跨系统对不上
业务定义	这到底在回答什么问题	指定窗口内创建的 P1 工单数量	运营、BI、Agent 各解释各的
grain	一行代表什么	metric_date + product_line + priority + org_id + category	ticket、event、day 混算
source	从哪个模型来	support_kpi_mart	直接查 raw，复盘困难
filters	哪些过滤条件被允许	product_line、priority、org_id、category	过滤口径不一致，或越权查询
owner	谁对口径负责	support_ops	出事故没人能拍板
tests	哪些自动检查保护它	not null、accepted values、registry validation	SQL 改坏了也没人发现

[1] “指标接口”不是某个厂商的固定产品名，而是本课程为了教学把 metric definition、semantic contract、tool contract 和 audit boundary 合并讲清的表达。

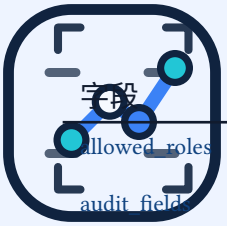


组成	小白版解释	OmniSupport 例子	缺失会怎样
docs	未来消费者怎么看懂它	dbt docs、metric interface notes	新同学和 Agent 都只剩猜
roles	谁能查、谁不能查	support_ops、instructor、admin	工具层可能越权
audit	查询证据怎么留	actor_role、release_id、row_count	无法追问答案来自哪次查询

指标的外延是它被谁消费：BI 看板、运营复盘、Agent 工具、评测、审计和治理。只要这些消费方共用同一张接口卡，后续分层、测试和工具契约才不会各自发散。

4. 指标接口卡片模板

字段	说明	示例
metric_name	稳定机器名	p1_ticket_count
business_label	人可读名称	P1 工单数
business_question	它回答的问题	最近 P1 工单是否增长
business_definition	业务定义	指定窗口内创建的 P1 工单数
source_model	来源模型	support_kpi_mart
grain	一行代表什么	metric_date, product_line, priority, org_id, category
measure_expression	计算逻辑	sum(p1_ticket_count)
time_field	时间字段	metric_date
allowed_dimensions	允许维度	product_line, priority, org_id, category
allowed_filters	允许过滤器	product_line, priority, org_id, category, data_release_id
owner	口径负责人	support_ops
tests	必要测试	not_null, accepted_values, registry validation
docs_link	文档入口	dbt docs 或 docs/blueprints/week05/metric-interface-principles.md



字段	说明	示例
allowed_roles	允许角色	support_ops，instructor，admin
audit_fields	审计字段	actor_role，actor_id，release_id，row_count
change_policy	改口径时怎么通知	先写 impact note，再更新 tests / docs / registry

对应到当前项目，analytics/metric_registry_v1.yml 已经把这些边界落成最小版本：

```
version: 1
registry_id: week05_support_metrics_v1
source_model: support_kpi_mart
safe_view: agent_tool_input_view
time_dimension: metric_date
max_window_days: 31
allowed_dimensions:
  - product_line
  - priority
  - org_id
  - category
allowed_roles:
  - support_ops
  - instructor
  - admin
```

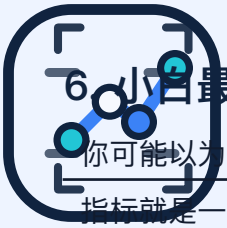
一个完整的 p1_ticket_count 草稿可以这样写：

```
metric_name: p1_ticket_count
business_label: P1 工单数
business_question: 最近 P1 工单是否增长，增长集中在哪些产品线或组织
business_definition: 指定时间窗口内创建的 P1 或 P1 critical 工单数量
source_model: support_kpi_mart
grain:
  - metric_date
  - product_line
  - priority
  - org_id
  - category
measure_expression: sum(p1_ticket_count)
time_field: metric_date
allowed_dimensions:
  - product_line
  - priority
```

```
  - org_id
  - category
allowed_filters:
  - product_line
  - priority
  - org_id
  - category
  - data_release_id
owner: support_ops
tests:
  - metric_date not null
  - p1_ticket_count >= 0
  - priority accepted values
docs_link: docs/blueprints/week05/metric-interface-principles.md
allowed_roles:
  - support_ops
  - instructor
  - admin
audit_fields:
  - actor_role
  - actor_id
  - data_release_id
  - row_count
change_policy: 口径变更必须更新 docs、tests、registry，并在交付摘要里说明影响范围
```

5. 企业最佳实践：指标上线前的 10 个问题

1. 这个指标回答的是哪个业务问题?
2. 一行代表什么 grain?
3. 时间窗口用哪个字段，为什么不是另一个时间字段?
4. reopen、升级、关闭后重开这类状态变化算不算?
5. 指标从哪个 source / mart 来，能不能追到上游?
6. 哪些维度和过滤器允许开放，哪些必须拒绝?
7. 谁是 owner，谁可以批准口径变更?
8. 有哪些 tests 能发现明显破坏?
9. Agent 能不能查，能查时必须留下哪些 audit 字段?
10. 改口径时，BI、运营、Agent、评测和后续周次谁会受影响?

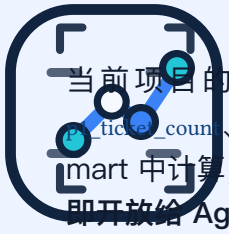


6. 小白最容易踩的误区

你可能以为	实际上
指标就是一条 SQL	指标是定义、来源、粒度、测试、文档、权限和接口的组合
dashboard 能显示就够了	Agent、评测和治理也需要同一口径
SQL 改了大家同步一下就行	没有 lineage 和 docs，影响范围不可控
Agent 会自动写对 SQL	Agent 很可能写出能运行但口径错误的 SQL
先让 Agent 自由查，后面再加权限	工具边界应该先于 Agent 查询自由度

7. OmniSupport 核心指标怎么讲

指标	小白解释	解决什么业务问题	常见误算
ticket_count	工单总数	看整体工作量基线	把 event 或 comment 当 ticket 计数
open_ticket_count	还没关闭的工单数	判断当前积压压力	把历史曾经 open 的工单也算进去
p1_ticket_count	高优先级工单数	观察严重问题是否增长	P1 定义不统一，或把 reopen event 当新工单
sla_breach_count	已经超过 SLA 的工单数	判断服务承诺是否被打破	只看 resolved ticket，漏掉仍 open 但已超时的 case
escalation_count	需要升级处理的工单数	观察一线支持是否扛不住	重复升级事件重复计数
avg_backlog_age_days	未解决工单平均积压天数	判断问题是否越拖越久	已 resolved 和 open ticket 混在一起算
avg_first_response_minutes	第一次客服响应用了多久	判断响应速度	把机器人、系统消息或客户回复当客服首响



当前项目的 Week05 registry 已落地 6 个核心指标：ticket_count、open_ticket_count、p1_ticket_count、sla_breach_count、escalation_count、avg_backlog_age_days。avg_first_response_minutes 已在 mart 中计算，但还没有进入 registry 白名单，这正好说明：**不是所有可计算字段都应该立即开放给 Agent**。[2]

8. 为什么 Week05 必须接在 Week04 后面

没有 Week04 的稳定表状态，Week05 口径很难负责。

- 你不知道指标输入对应哪版 source / mart；
- 你不知道某次指标变化是数据状态变了，还是 SQL 变了；
- 你无法把 data_release_id 绑定到 Agent 的查询审计里；
- 你很难把 Week08 的 retrieval 指标和 Week11 后的评测结果复盘到同一份数据状态。

所以 Week05 不是孤立的 analytics week。它是在 Week04 的 table state 上建立 **semantic and metric interface**。

9. 本课小练习

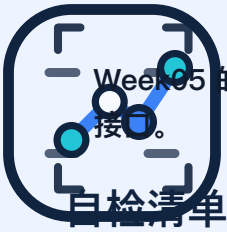
用 10 分钟选一个指标，写出接口卡片草稿。建议先选 p1_ticket_count 或 sla_breach_count，不要一开始就选平均首响这种容易涉及事件来源和角色判断的指标。

练习项	合格答案应该包含
业务问题	这个指标要帮谁做什么判断
grain	一行代表 ticket、day、metric 还是 event
source model	从哪个 mart 来，不直接查 raw
time field	用哪个时间字段，为什么
allowed dimensions	哪些维度可以给 BI / Agent 用
tests	至少 2 个能保护口径的检查
audit	Agent 查询时必须留下哪些字段

10. 本课最重要判断

如果只记一句话，就记这句：

[2] avg_first_response_minutes 牵涉“谁算客服首响”“机器人和系统消息算不算”等更细口径。先在 mart 里观察，暂不开放给 registry，是比仓促开放更稳的设计。



Week05 的交付物不是几个 SQL 文件，而是一套可以被人和 Agent 共同信任的指标接口。

自检清单

- ☐ 我能用一句话解释为什么 KPI 不是 SQL 结果。
- ☐ 我能说清 BI 和 Agent 为什么要复用同一套口径。
- ☐ 我能指出 Dashboard A / Dashboard B / Agent 三个 P1 数字为什么会不同。
- ☐ 我能列出 Agent 裸写 SQL 的至少三类风险。[3]
- ☐ 我能为 `p1_ticket_count` 写出 grain、source、time field、allowed dimensions。
- ☐ 我能解释为什么 `avg_first_response_minutes` 可以先在 mart 里观察，但暂不进 registry。
- ☐ 我知道本周最终要交付哪些口径资产。

课后最小行动

在你的项目笔记里写出一个指标接口卡片。先不写 SQL 也可以，但必须说清：

```
## p1_ticket_count metric interface

- business_definition:
- source_model:
- grain:
- time_dimension:
- allowed_dimensions:
- allowed_filters:
- owner:
- required_tests:
- allowed_roles:
- audit_fields:
```

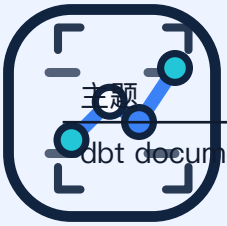
可参考附录里的 Metric Interface Card 模板。

延伸阅读

主题	推荐资料	为什么读
dbt Semantic Layer	dbt Docs: Semantic Layer	理解指标集中定义与下游一致性的工程目标。[4]

[3] Agent 查询指标必须先走 metric、dimension、filter、role 和 time window 白名单，并留下 audit 字段；否则它可能生成能运行但越权、昂贵或口径错误的 SQL。

[4] dbt Semantic Layer 的理念可以帮助理解集中指标定义，但 Week05 Student Core 先以本地 dbt Core、metric registry 和 tool contract 跑通最小闭环，不把托管平台作为硬依赖。



主题	推荐资料	为什么读
dbt documentation	dbt Docs: Documentation	理解模型和字段描述为什么服务未来消费者
OpenAI Structured Outputs	OpenAI Docs: Structured Outputs	理解工具接口为什么需要严格结构化输出
OpenAI Function Calling	OpenAI Docs: Function Calling	理解 Agent 为什么应通过工具契约调用指标，而不是裸写 SQL

Footnotes