【最佳实践】实时计算Flink在在线教育行业的实时数仓建设实践

行业背景

- 行业现状：
    - 在线教育是运用互联网、人工智能等现代信息技术进行教与学互动的新型教育方式，是教育服务的重要组成部分。发展在线教育，有利于构建网络化、数字化、个性化、终身化的教育体系，有利于建设“人人皆学、处处能学、时时可学”的学习型社会。
- 大数据在其行业中的作用：
    - 对未来客户的画像更加精准，营销推广时可以对接更好的服务并提升成交转化率（提升ROI不一定，这涉及到外部竞争）；
    - 更全面的评估老师、学生、机构、行业等在线教育行业的各个参与者；
    - 大数据帮助在线教育行业更快发展

业务场景

某公司开发了个在线教育类APP，培训机构可以在APP中会发布一些直播课程，离线课程，习题，学习文章等内容。用户可在线学习新知识，离线巩固已学知识，并对学过的内容进行课后练习/测试。
业务的构建涉及到几部分：

1. APP：应用程序，用户访问入口
2. 后台系统：
    1. 教学老师：通过分析学生课堂参与情况，提供不同的授课方案，因材施教。
    2. 运维人员：通过运维监控指标，实时监控在线教育直播网络质量。
    3. 运营人员：根据学生注册、学习质量、平台成单量等统计信息针对性开展平台运营工作：
        1. 学生办理注册、增删课程等操作；
        2. 学生学习质量审核；
        3. 平台指标查看，如平台日成单量统计。

技术架构

架构解析：

数据采集：该场景中，数仓的数据来源有两部分：app的埋点至消息队列 Kafka 以及 hbase 等业务数据库的增量日志。值得注意的一点是，实时数仓往往和离线数仓配合使用，共享一套管控系统，如权限/元数据管理/调度等系统。

实时数仓架构：该场景中，整个实时数仓的ETL和BI部分的构建，全部通过 Flink + Kafka 完成，原始日志app_log_origin是从客户端直接收集上来的。然后数据处理，加维等操作后，最终输入到业务系统。

业务指标

- 实时数据中间层
  - 学生操作日志 ETL 清洗（分析学生操作在线信令日志）
    - 获取学生移动图片操作
    - 获取学生 hover 图片操作
    - 获取学生画线操作
    - 音频播放
    - 音频暂停
    - 图文匹配错误
    - 图文匹配正确
  - 学生注册考试等级日志 ETL 清洗
- 学生行为分析
  - 学生在线（直播）课程课堂表现统计
  - 学生离线（录播）课程学习时长统计
- 运维/网络监控
  - 直播课程（音频）网络监控
  - 直播课程（视频）网络监控
- 运营分析
  - 每小时不同 level 的学生注册人数统计
  - 每日课程顾问追踪统计

说明：该案例中仅包含以上场景及指标，在实际的应用场景下还包括日uv/pv，topN热门授课教师，教师授课质量、数量审核等其他指标。

业务代码

场景一：对原始日志进行实时数据清洗

学生操作日志 ETL 清洗（分析学生操作在线信令日志）

> 学生在直播课程中，会做一些随堂练习/测试，通过页面点击等操作形成原始埋点日志，为了很快的感知学生的学习表现（课堂表现），业务方针对不同的操作进行计分处理。为了下游有效的对数据进行处理，针对学生不同的操作，将原始数据（多层 JSON 数据）进行清洗（单层 JSON 数据），写入 kafka 中。

- 埋点数据样例

```
--输入
{
    "createTime":"",
    "data":{
        "userid":"",
        "roomid":"",
        "timestamp":"",
        "role":"",
        "msgid":"",
        "msg":{
            "msgtype":"",
            "msg_data":{
                "target_id":"",
                "target_type":"",
                "action":"",
                "sub_action":"",
                "page_index":""
            }
        }
    }
}
--输出
{
    "messageCreateTime":"",
    "timeStamp":"",
    "messageTimeStamp":"",
    "userId":"",
    "roomId":"",
    "role":"",
    "msgId":"",
    "msgType":"",
    "targetId":"",
    "targetType":"",
    "action":"",
    "subAction":"",
    "pageIndex":"",
    "event":""
}
```

输入表

```
create table timeline_analysis_student_stream (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
    -- 事件时间
    `createTime` as cast(JSON_VALUE(`message`, '$.createTime')as VARCHAR),
    -- 用户 ID
    `userid` as cast(JSON_VALUE (cast(JSON_VALUE (`message`, '$.data') as VARCHAR
    -- 教室 ID
      `roomid` as cast(JSON_VALUE (cast(JSON_VALUE (`message`, '$.data') as VARCH
    -- 操作时间
      `time_stamp` as cast(JSON_VALUE (cast(JSON_VALUE (`message`, '$.data') as V
      -- 角色
    `role` as cast(JSON_VALUE (cast(JSON_VALUE (`message`, '$.data') as VARCHAR),
    -- 消息 ID
    `msgid` as cast(JSON_VALUE (cast(JSON_VALUE (`message`, '$.data') as VARCHAR)
    -- 消息类型
    `msg_msgType` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VALUE (`message`
    -- 消息目标 ID
    `msg_msgData_targetId` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VALUE (
    -- 消息目标类型
      `msg_msgData_targetType` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VAL
    -- 学生操作
      `msg_msgData_action` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VALUE (
    -- 学生次操作
      `msg_msgData_subAction` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VALU
    -- PPT 页码
      `msg_msgData_pageIndex` as cast(JSON_VALUE (cast(JSON_VALUE (cast(JSON_VALU
) with (
    type = 'kafka011',
    topic = 'timeline_client_topic',
    `group.id` = 'timeline_analysis_student_consumer',
       ...
);
```

输出表

```
create table signal_student_classroom_internation (
   messageKey VARBINARY,
   `message` VARBINARY,
   PRIMARY KEY (messageKey)
) with (
    type = 'kafka011',
    topic = 'timeline_analysis_student',
    ...
);
```

业务代码

- 获取学生移动图片操作
    - 当学生学习词性（形容词/副词），课堂小练习让学生将屏幕中出现的单词图片进行分类，学生需要移动图片进入不同的分类桶中。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetType = 'shape' AND
    msg_msgData_action = 'move';
```

- 获取学生 hover 图片操作
    - 当学生学习单词时，需要学习单词读音，当学生鼠标悬停到图片时进行发音教学。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetType = 'shape' AND
    msg_msgData_action = 'mouse' AND
    msg_msgData_subAction = 'over';
```

- 获取学生画线操作
    - 学生通过画线来进行随堂图文匹配练习。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetType = 'shape' AND
    msg_msgData_action = 'add';
```

- 获取学生音频播放操作
    - 学生播放课件中的音频。

```

INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetType = 'template' AND
    msg_msgData_action = 'audio' AND
    msg_msgData_subAction = 'start';
```

- 获取学生音频暂停操作
    - 学生暂停课件中的音频。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetType = 'template' AND
    msg_msgData_action = 'audio' AND
    msg_msgData_subAction = 'pause';
```

- 获取学生图文匹配错误操作
    - 连线操作后，返回给学生连线结果。会影响课堂表现分数。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
```

```
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetId = 'match' AND
    msg_msgData_targetType = 'template' AND
    msg_msgData_action = 'match' AND
    msg_msgData_subAction = 'drop:wrong';
```

- 获取学生图文匹配正确操作
  - 连线操作后，返回给学生连线结果。会影响课堂表现分数。

```
INSERT INTO signal_student_classroom_internation
SELECT
 cast(messageKey as VARBINARY) as messageKey,
 cast(CONCAT('{ "messageCreateTime": "',createTime,'","timeStamp": ',NOW()*1000,'
FROM timeline_analysis_student_stream
WHERE
    msgid = '305' AND
    msg_msgType = '116' AND
    role = '2' AND
    msg_msgData_targetId = 'match' AND
    msg_msgData_targetType = 'template' AND
    msg_msgData_action = 'match' AND
    msg_msgData_subAction = 'drop:correct';
```

学生注册考试等级日志 ETL 清洗

> 学生在 WEB/APP 页面注册时需要考试测评等级，以便后期学习对应 Level 的课程，通过 Flink 做数据清洗，将埋点到 kafka 上日志，输出到 Hbase。

- 埋点数据样例

```
{
    "id":"",
    "chinese_name":"",
    "english_name":"",
    "level":"",
    "pid":"",
    "create_time":"",
    "update_time":"",
    "dept_id":""
}
```

输入表

```
create table blink_stg_activity__channel_name_dictionary_da (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
      -- ID
    id as JSON_VALUE(`message`,'$.id'),
      -- 中文名称
    chinese_name as JSON_VALUE(`message`,'$.chinese_name'),
      -- 英文名称
    english_name as JSON_VALUE(`message`,'$.english_name'),
      -- 测试登记
    level as JSON_VALUE(`message`,'$.level'),
      -- 唯一标识 ID
    pid as JSON_VALUE(`message`,'$.pid'),
      -- 创建时间
    create_time as JSON_VALUE(`message`,'$.create_time'),
      -- 更新时间
    update_time as JSON_VALUE(`message`,'$.update_time'),
      -- 部门 ID
    dept_id as JSON_VALUE(`message`,'$.dept_id')
) with (
    type = 'kafka010',
    topic = 'blink_stg_activity__channel_name_dictionary_da',
    `group.id` = 'blink_stg_activity__channel_name_dictionary_da',
    ...
);
```

输出表

```
create table blink_stg_activity__channel_name_dictionary_da_sinkhbase (
    rowkey varchar,
    id varchar,
    chinese_name varchar,
    english_name varchar,
    level varchar,
    pid varchar,
    create_time varchar,
    update_time varchar,
    dept_id varchar,
    primary key (rowkey)
) with (
    type = 'cloudhbase',
    tableName = 'channel_name_dictionary',
    ...
);
```

业务代码

```
insert into
    blink_stg_activity__channel_name_dictionary_da_sinkhbase
```

```
SELECT
    MD5(id) as rowkey,
    id ,
    chinese_name ,
    english_name ,
    level ,
    pid ,
    create_time ,
    update_time ,
    dept_id
from
    blink_stg_activity__channel_name_dictionary_da;
```

场景二：学生行为分析

学生在线（直播）课程课堂表现统计

> 场景一中针对学生操作日志进行了清洗，该场景消费其清洗之后的数据，针对不同的用户 ID、Web 服务端 ID、角色、操作事件进行分组，开 1min 窗口，通过 count(event)聚合进行计分，求得每分钟学生在线（直播）课程的课堂表现。

- 该指标上游数据是在学生操作日志 ETL 清洗的基础上进行统计

```
{
    "userId":"",
    "roomId":"",
    "role":"",
    "event":"",
    "timeStamp":""
}
```

输入表

```
create table timeline_analysis_student_mashup_stream (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
      -- 用户 ID
    `userId` as cast(JSON_VALUE (`message`, '$.userId') as BIGINT),
       -- Web 服务器 ID
    `webserverId` as cast(JSON_VALUE (`message`, '$.roomId') as BIGINT),
      -- 角色
    `role` as cast(JSON_VALUE (`message`, '$.role') as TINYINT),
    -- 操作事件
    `event` as cast(JSON_VALUE (`message`, '$.event') as VARCHAR),
      -- 事件时间
    time_stamp as TO_TIMESTAMP(cast(JSON_VALUE (`message`, '$.timeStamp') as BIGI
    WATERMARK wk FOR time_stamp AS WITHOFFSET (time_stamp, 0)--为rowtime定义waterm
) with (
    type = 'kafka011',
    topic = 'timeline_analysis_student',
    `group.id` = 'timeline-analysis-student-mashup-consumer',
    ...
);
```

输出表

```
create table timeline_signal_analysis_mysql (
    start_time TIMESTAMP,
    end_time TIMESTAMP,
    webserver_id BIGINT,
    user_id BIGINT,
    role TINYINT,
    event VARCHAR,
    event_count BIGINT,
    create_time TIMESTAMP
) with (
    type='RDS',
    tableName='timeline_signal_analysis',
    ...
);
```

业务代码

- 学生课堂表现解析
    - 学生在课堂中举手回答问题等行为进行积分，以此衡量学生课堂表现。

```
insert into timeline_signal_analysis_mysql
select
    TUMBLE_START(time_stamp,INTERVAL '1' MINUTE) as start_time,
    TUMBLE_END(time_stamp,INTERVAL '1' MINUTE) as end_time,
    webserverId as webserver_id,
    userId as user_id,
    role as role,
    event as event,
    COUNT(event) as event_count,
    CURRENT_TIMESTAMP as create_time
FROM timeline_analysis_student_mashup_stream
GROUP BY TUMBLE (time_stamp,INTERVAL '1' MINUTE),
    userId,
    webserverId,
    role,
    event;
```

学生离线（录播）课程学习时长统计

> 通过 subEvent = 'PPT_SUCCESS' 将完成课程的事件整理出来，通过自关联的方式，和源表进行 JOIN 打宽，计算 'PPT_SUCCESS' 的时间点与最初播放 PPT 的时间差值。

- 埋点数据样例

```
{
    "classroom_id":"",
    "user_type":"",
    "user_id":"",
    "event_time":"",
    "sub_event":"",
    "extra":{
        "data_time":"",
        "msg":{
            "pptIndex":""
        }
    }
}
```

输入表

```
create table qos_log_kafka (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
      -- (录播) 教室 ID
    `classroomId` as cast(JSON_VALUE(`message`, '$.classroom_id')as VARCHAR),
      -- 用户类型
    `userType` as cast(JSON_VALUE(`message`, '$.user_type')as VARCHAR),
      -- 用户 ID
    `userId` as cast(JSON_VALUE(`message`, '$.user_id')as BIGINT),
      -- 事件时间
    `eventTime` as cast(JSON_VALUE(`message`, '$.event_time')as BIGINT),
      -- 次操作
      `subEvent` as cast(JSON_VALUE(`message`, '$.sub_event')as VARCHAR),
      -- 数据时间
      `extraDataTime` as cast(cast(JSON_VALUE(cast(JSON_VALUE(`message`, '$.extra
    -- PPT 页码
    `extraMsgIndex` as cast(JSON_VALUE(cast(JSON_VALUE(cast(JSON_VALUE(`message`,
) with (
    type = 'kafka011',
    topic = 'qos_log',
      ...
);
```

输出表

```
create table user_enter_classroom_take_time_mysql (
    user_id BIGINT,
    classroom_id VARCHAR,
    user_type VARCHAR,
    spend_time BIGINT,
    event_time TIMESTAMP,
    create_time TIMESTAMP
) with (
    type='rds',
    tableName='user_enter_classroom_take_time',
    ...
);
```

业务代码

- 学生进入教室时长
    - 离线录播课程，通过 PPT 的播放时间来计算学生进入教室的时长。

```
CREATE VIEW qos_log_kafka_view AS
SELECT
    `userId`,
    `classroomId`,
    `userType`,
    `eventTime`,
     subEvent,
    `extraDataTime`
FROM qos_log_kafka
WHERE subEvent = 'PPT_SUCCESS';

insert into user_enter_classroom_take_time_mysql
SELECT
  a.userId,
  a.classroomId,
  a.userType,
  b.extraDataTime-a.extraDataTime,--毫秒值
  TO_TIMESTAMP(a.eventTime),
  CURRENT_TIMESTAMP
FROM qos_log_kafka a
JOIN qos_log_kafka_view b ON a.userId=b.userId AND a.classroomId=b.classroomId
WHERE a.extraDataTime<b.extraDataTime;
```

场景三：运维/网络监控

> 通过学生直播课程中，视频/音频运维埋点信息计算，以 userId, agoraChannelId,classroomId, userType, event,agoraAudioStateUid/agoraVideoStateUid进行分组，开 30s 的滚动窗口，求最近 30s 直播课的视频/音频质量（丢包/异常平均值、总次数），供下游运维同学监控，实时调整音频/视频质量，给用户最佳的学习体验。

- 埋点数据样例

```
{
    "classroom_id":"",
    "user_type":"",
    "user_id":"",
    "agora_channel_id":"",
    "event":"",
    "agora_videoState":{
        "fr":"",
        "uid":""
    },
    "agora_audioState":{
        "lost":"",
        "uid":""
    },
    "messageCreateTime":""
}
```

输入表

```
create table qos_agora_record_kafka (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
      -- 直播教室 ID
    `classroomId` as cast(JSON_VALUE(`message`, '$.classroom_id')as VARCHAR),
      -- 用户类型
    `userType` as cast(JSON_VALUE(`message`, '$.user_type')as VARCHAR),
      -- 用户 ID
    `userId` as cast(JSON_VALUE(`message`, '$.user_id')as BIGINT),
      -- 渠道 ID
    `agoraChannelId` as cast(JSON_VALUE(`message`, '$.agora_channel_id')as BIGINT
      -- 事件
    `event` as cast(JSON_VALUE(`message`, '$.event')as VARCHAR),
      -- 视频故障记录
    `agoraVideoStateFr` as cast(JSON_VALUE(cast(JSON_VALUE(`message`, '$.agora_vi
      -- 视频故障唯一标识 ID
    `agoraVideoStateUid` as cast(JSON_VALUE(cast(JSON_VALUE(`message`, '$.agora_v
      -- 音频丢失记录
    `agoraAudioStateLost` as cast(JSON_VALUE(cast(JSON_VALUE(`message`, '$.agora_
      -- 音频丢失唯一标识 ID
    `agoraAudioStateUid` as cast(JSON_VALUE(cast(JSON_VALUE(`message`, '$.agora_a
      -- 事件时间
    `messageCreateTime` as cast(JSON_VALUE(`message`, '$.messageCreateTime')as BI
     WATERMARK wk FOR messageCreateTime AS WITHOFFSET (messageCreateTime, 60000)-
) with (
    type = 'kafka011',
    topic = 'agora_record',
    ...
);
```

输出表

```
create table user_av_mysql (
      -- 开窗时间
    start_time TIMESTAMP,
      -- 关窗时间
    end_time TIMESTAMP,
      --用户 ID
    user_id BIGINT,
    web_server_id BIGINT,
      -- 直播教室 ID
    classroom_id VARCHAR,
      -- 用户类型
    user_type VARCHAR,
    extra_uid BIGINT,
    event VARCHAR,
      -- 异常总和值
    event_sum BIGINT,
      -- 异常平均值
    event_avg DOUBLE,
      -- 异常次数
    event_count BIGINT,
    create_time TIMESTAMP
) with (
    type='rds',
    tableName='user_av_record',
    ...
);
```

直播课程（音频）网络监控

业务代码

```
insert into user_av_mysql
select
    TUMBLE_START(messageCreateTime, INTERVAL '30' SECOND) as start_time,
    TUMBLE_END(messageCreateTime, INTERVAL '30' SECOND) as end_time,
    CASE WHEN `userId` is NULL THEN -1 else userId END as user_id,
    CASE WHEN `agoraChannelId` is NULL THEN -1 else agoraChannelId END as web_ser
    CASE WHEN `classroomId` is NULL THEN -1 else classroomId END as classroom_id,
    userType as user_type,
    agoraAudioStateUid as extra_uid,
    CONCAT(event,'_AUDIO_STATE') as event,
    SUM(agoraAudioStateLost) as event_sum,
    AVG(agoraAudioStateLost) as event_avg,
    COUNT(event) as event_count,
    CURRENT_TIMESTAMP as create_time
```

```
FROM qos_agora_record_kafka
WHERE agoraAudioStateLost >= 0 AND userType = 'student'
GROUP BY TUMBLE (messageCreateTime, INTERVAL '30' SECOND),
    userId,
    agoraChannelId,
    classroomId,
    userType,
    event,
    agoraAudioStateUid;
```

直播课程（视频）网络监控

业务代码

```
insert into user_av_mysql
select
    TUMBLE_START(messageCreateTime,INTERVAL '30' SECOND) as start_time,
    TUMBLE_END(messageCreateTime,INTERVAL '30' SECOND) as end_time,
    CASE WHEN `userId` is NULL THEN -1 else userId END as user_id,
    CASE WHEN `agoraChannelId` is NULL THEN -1 else agoraChannelId END as web_ser
    CASE WHEN `classroomId` is NULL THEN -1 else classroomId END as classroom_id,
    userType as user_type,
    agoraVideoStateUid as extra_uid,
    CONCAT(event,'_VIDEO_STATE') as event,
    SUM(agoraVideoStateFr) as event_sum,
    AVG(agoraVideoStateFr) as event_avg,
    COUNT(event) as event_count,
    CURRENT_TIMESTAMP as create_time
FROM qos_agora_record_kafka
WHERE agoraVideoStateFr >= 0 AND userType = 'student'
GROUP BY TUMBLE (messageCreateTime, INTERVAL '30' SECOND),
    userId,
    agoraChannelId,
    classroomId,
    userType,
    event,
    agoraVideoStateUid;
```

场景四：运营分析

每小时不同 level 的学生注册人数统计

> 学生通过不同渠道（Web 广告输入、App 广告输入等）进行注册，本场景会读取注册端日志，并关联用户注册时的考试等级表（分为 A/B/C/D 四个 level），以此展现给运营人员，每小时不同 level&渠道 的学生注册人数，实时的调整运营推广策略。

- 埋点数据样例

```
--学生表
{
    "id":"",
    "channel_id":"",
    "update_time":""
}
--用户注册数据
{
    "id":"",
    "name":"",
    "register_date_time":"",
    "status":""
}

--学生测试等级表：使用场景—“学生注册考试等级日志ETL清洗”的结果表
```

输入表

```
create table student_da_src (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
    `id` as JSON_VALUE (`message`, '$.id'),--用户 ID
    `channel_id` as JSON_VALUE (`message`, '$.channel_id'),--渠道 ID
    `update_time` as JSON_VALUE (`message`, '$.update_time')--更新时间
) with (
    type = 'kafka010',
    topic = 'uc_account-student',
    ...
);
```

```
create table user_da_in (
    messageKey VARBINARY,
    `message` VARBINARY,
    topic VARCHAR,
    `partition` INT,
    `offset` BIGINT,
    `id` as JSON_VALUE (`message`, '$.id'),--用户 ID
    `name` as JSON_VALUE (`message`, '$.name'),--用户名称
    `register_date_time` as JSON_VALUE (`message`, '$.register_date_time'),--注册
    `status` as JSON_VALUE (`message`, '$.status')--状态
) with (
    type = 'kafka010',
    topic = 'uc_account-user',
    `group.id` = 'uc_account-user',
    ...
);
```

```
create table channel_da (
    rowkey varchar,
    id VARCHAR,
    `level`  VARCHAR,
    primary key (rowkey),
    PERIOD FOR SYSTEM_TIME
) with (
    type = 'cloudhbase',
    tableName = 'databus:activity.channel',
    ...
    );
```

输出表

```
create table sink_table (
    uk varchar,
    reg_date bigint,
    level varchar,
    leads bigint,
    primary key (uk)
) with (
    type = 'elasticsearch',
    index = 'vk_app_es_sign_csh',
    typeName = 'vk_app_es_sign_csh',
    ...
);
```

业务代码

```
create view student_da_src_view as
SELECT
   last_value(id) as id,
   last_value(update_time) as update_time,
   last_value(channel_id) as channel_id
from student_da_src
group by id;

create view user_da_in_view as
SELECT
   last_value(id) as id,
   last_value(name) as name,
   last_value(register_date_time) as register_date_time,
   last_value(status) as status
from user_da_in
group by id;

insert into
    sink_table
SELECT
      case when level in ('A','B','C','D') then level else 'other' end as uk
```

```
     ,cast(date_format(register_date_time,'yyyyMMddHH') as bigint) as reg_date
     ,case when level in ('A','B','C','D') then level else 'other' end as levels
     ,COUNT(distinct t.id) AS leads
FROM
     student_da_src_view t
LEFT JOIN user_da_in_view  u ON u.id = t.id
LEFT JOIN channel_da FOR SYSTEM_TIME AS OF PROCTIME() ch ON ch.rowkey = MD5(t.cha
where u.name not LIKE '%测试%'
and u.name not LIKE 'DM\\_%'
and u.name not LIKE '%test%'
and u.status='NORMAL'
group by date_format(register_date_time,'yyyyMMddHH')
        ,case when level in ('A','B','C','D') then level else 'other' end
        ,concat(date_format(register_date_time,'yyyyMMddHH'),case when level in (
;
```

每日课程顾问追踪统计

> 首先通过 ID 进行分组，求出相同 ID 的最新消息（达到去重效果），在最新消息的基础上使用全局Group 聚合，根据事件时间（天）、课程顾问 ID 统计每天每位课程顾问找学生确认“学习进度/约课”的次数。

- 埋点数据样例

```
{
    "id":"",
    "leads_flow_event_id":"",
    "group_id":"",
    "cc_id":"",
    "student_id":"",
    "order_id":"",
    "leads_id":"",
    "confirm_date_time":"",
    "create_time":"",
    "update_time":"",
    "order_create_time":"",
    "canceled_date_time":"",
    "apply_refund_date":"",
    "status":""
}
```

输入表

```
create table cc_data_pack_order_info_src (
    `messageKey` VARBINARY,
    `message` VARBINARY,
    `topic` VARCHAR,
    `partition` INT,
    `offset` BIGINT,
      -- ID
    `id` as JSON_VALUE (`message`, '$.id'),
        --  (Course Consultant) 课程顾问 ID
      `cc_id` as JSON_VALUE (`message`, '$.cc_id'),
        -- 学生 ID
      `student_id` as JSON_VALUE (`message`, '$.student_id'),
        -- 确认时间
      `confirm_date_time` as JSON_VALUE (`message`, '$.confirm_date_time'),
        -- 创建时间
      `create_time` as JSON_VALUE (`message`, '$.create_time'),
        -- 更新时间
      `update_time` as JSON_VALUE (`message`, '$.update_time'),
        -- 订单创建时间
      `order_create_time` as JSON_VALUE (`message`, '$.order_create_time'),
        -- 订单取消时间
      `canceled_date_time` as JSON_VALUE (`message`, '$.canceled_date_time'),
      -- 付款时间
      `apply_refund_date` as JSON_VALUE (`message`, '$.apply_refund_date'),
        -- 状态
      `status` as JSON_VALUE (`message`, '$.status')
  ) with (
      type = 'kafka010',
      topic = 'data_pack_order_info',
      `group.id` = 'data_pack_order_info',
      ...
  );
```

输出表

```
CREATE TABLE index_sink (
  `cc_id` bigint(20) NOT NULL,
  `cc_index` bigint(10) NOT NULL,
  `type` int(6) NOT NULL,
  `attribution_time` varchar NOT NULL,
  `update_time` timestamp NOT NULL,
  PRIMARY KEY (`cc_id`, `type`, `attribution_time`)
) WITH (
    type='rds',
    tableName='staff_index',
    ...
);
```

业务代码

```

CREATE VIEW cc_data_pack_order_info_view as
select
    last_value (cc_id) as cc_id,
    last_value (confirm_date_time) as confirm_date_time,
    last_value (`status`) as `status`
from
    cc_data_pack_order_info_src
```

```sql
group by
    id;


insert into index_sink
select
    cast(cc_id as bigint) as cc_id,
    count(*) as cc_index,
    cast(1 as int) as type,
    date_format(confirm_date_time,'yyyy-MM-dd') as attribution_time,
    current_timestamp as update_time
from
    cc_data_pack_order_info_view
where
    confirm_date_time is not null
    and `status` is not null
    and `status` = 3
group by
    date_format(confirm_date_time,'yyyy-MM-dd'), cc_id;
```