日志展开解释

添加系统时间维度（`SysTimeDimension`）时所执行的步骤：

1. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.u.SysTimeDimensionBuilder 19 - addSysTimeDimension before:[Dim(name=部门, type=categorical, expr=null, dateFormat=yyyy-MM-dd, typeParams=null, isCreateDimension=1, bizName=department, description=null, isTag=0)], engineAdaptor.tencent.supersonic.headless.core.adaptor.db.MysqlAdaptor@486b5958**
   - **解释**: 这是在 `SysTimeDimensionBuilder` 类中执行 `addSysTimeDimension` 方法之前的状态。
   - `before`：当前已有一个维度，名称为“部门”（`name=部门`），类型为分类型（`type=categorical`），业务名称为 `department`（`bizName=department`）。该维度未设置表达式（`expr=null`），使用的日期格式为 `yyyy-MM-dd`，且标记为需要创建（`isCreateDimension=1`）。
   - `engineAdaptor`：指向 `MysqlAdaptor` 的一个实例，这意味着正在使用 MySQL 数据库适配器。
2. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.u.SysTimeDimensionBuilder 19 - addSysTimeDimension before:[Dim(name=, type=time, expr=null, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=imp_date, description=null, isTag=0), Dim(name=, type=categorical, expr=page, dateFormat=yyyy-MM-dd, typeParams=null, isCreateDimension=0, bizName=page, description=null, isTag=0)], engineAdaptor.tencent.supersonic.headless.core.adaptor.db.MysqlAdaptor@486b5958**
   - **解释**: 这里是另一个 `addSysTimeDimension` 方法调用之前的状态。
   - `before`：该列表包含两个维度：
     - 第一个维度是时间类型（`type=time`），名称为空（`name=`），表示日期为 `imp_date`，粒度为天（`timeGranularity=day`），且不需要创建新维度（`isCreateDimension=0`）。
     - 第二个维度是分类类型（`type=categorical`），与页面相关（`expr=page`），使用 `page` 作为业务名称（`bizName=page`），且也不需要创建新维度。
3. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.u.SysTimeDimensionBuilder 29 - addSysTimeDimension after:[Dim(name=, type=time, expr=null, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=imp_date, description=null, isTag=0), Dim(name=, type=categorical, expr=page, dateFormat=yyyy-MM-dd, typeParams=null, isCreateDimension=0, bizName=page, description=null, isTag=0), Dim(name=null, type=time, expr=imp_date, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=sys_imp_date, description=null, isTag=0), Dim(name=null, type=time, expr=DATE_FORMAT(DATE_SUB(imp_date, INTERVAL (DAYOFWEEK(imp_date) - 2) DAY), '%Y-%m-%d'), dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=false, timeGranularity=week), isCreateDimension=0, bizName=sys_imp_week, description=null, isTag=0),**

Dim(name=null, type=time, expr=DATE_FORMAT(imp_date, '%Y-%m') , dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=false, timeGranularity=month), isCreateDimension=0, bizName=sys_imp_month, description=null, isTag=0)], engineAdaptor.tencent.supersonic.headless.core.adaptor.db.MysqlAdaptor@486b5958

- **解释**: 在 `addSysTimeDimension` 方法调用后的状态。
- `after`: 此列表显示了在处理后的系统时间维度：
  - 第一个和第二个维度保持不变。
  - 第三个维度新增了 `sys_imp_date` 维度，基于 `imp_date`，是主维度，粒度为天。
  - 第四个维度是基于 `imp_date` 的星期维度（`sys_imp_week`），通过使用 `DATE_SUB` 和 `DAYOFWEEK` 函数创建，粒度为星期。
  - 第五个维度是基于 `imp_date` 的月份维度（`sys_imp_month`），通过使用 `DATE_FORMAT` 函数创建，粒度为月份。

4. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.u.SysTimeDimensionBuilder 19 - addSysTimeDimension before:[Dim(name=, type=time, expr=null, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=imp_date, description=null, isTag=0), Dim(name=页面, type=categorical, expr=page, dateFormat=yyyy-MM-dd, typeParams=null, isCreateDimension=1, bizName=page, description=null, isTag=0)], engineAdaptor.tencent.supersonic.headless.core.adaptor.db.MysqlAdaptor@486b5958**
   - **解释**: 又一次调用 `addSysTimeDimension` 方法之前的状态。
   - `before`: 这里同样显示了两个维度：
     - 第一个维度与之前的时间维度相同，基于 `imp_date`，粒度为天。
     - 第二个维度是分类维度，与页面相关，业务名称为 `page`，标记为需要创建（`isCreateDimension=1`）。
5. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.u.SysTimeDimensionBuilder 29 - addSysTimeDimension after:[Dim(name=, type=time, expr=null, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=imp_date, description=null, isTag=0), Dim(name=页面, type=categorical, expr=page, dateFormat=yyyy-MM-dd, typeParams=null, isCreateDimension=1, bizName=page, description=null, isTag=0), Dim(name=null, type=time, expr=imp_date, dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=true, timeGranularity=day), isCreateDimension=0, bizName=sys_imp_date, description=null, isTag=0), Dim(name=null, type=time, expr=DATE_FORMAT(DATE_SUB(imp_date, INTERVAL (DAYOFWEEK(imp_date) - 2) DAY), '%Y-%m-%d'), dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=false, timeGranularity=week), isCreateDimension=0, bizName=sys_imp_week, description=null, isTag=0), Dim(name=null, type=time, expr=DATE_FORMAT(imp_date, '%Y-%m') , dateFormat=yyyy-MM-dd, typeParams=DimensionTimeTypeParams(isPrimary=false, timeGranularity=month), isCreateDimension=0, bizName=sys_imp_month,**

description=null, isTag=0)], engineAdaptor.tencent.supersonic.headless.core.adaptor.db.MysqlAdaptor@486b5958

- **解释**: 调用 `addSysTimeDimension` 方法后的状态。
- `after`: 系统添加了与之前日志行中类似的时间维度（`sys_imp_date`，`sys_imp_week`，`sys_imp_month`），但保留了最初的两个维度（即时间维度 `imp_date` 和页面维度 `page`）。

总结：

日志详细描述了系统在 `SysTimeDimensionBuilder` 类中通过 `addSysTimeDimension` 方法添加时间维度的过程。系统检查现有的维度，并基于需要创建多个时间相关的维度（如 `sys_imp_date`，`sys_imp_week`，和 `sys_imp_month`），这些维度都是从原始时间字段 `imp_date` 派生而来，并根据需要确定是否需要生成新的维度。

`DefaultSemanticTranslator` 类在进行语义转换时的过程：

1. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.t.DefaultSemanticTranslator 51 - SemanticConverter before [QueryParam(groups=[], aggregators=[{"column":"pv","func","nameCh":"null","args","alias"}], orders=[], dimensionFilters=[], metricFilters=[], dateInfo={"dateMode","startDate":"2024-08-06","endDate":"2024-07-31","dateList":[],"unit":1,"period":"DAY","text":"null"}, limit=2000, queryType=METRIC, s2SQL=null, correctS2SQL=null, dataSetId=1, dataSetName=null, modelIds=[], params=[], metrics=[pv], dimensions=null, where=null, order=null, nativeQuery=false)]**
   - **解释**: 在执行语义转换之前的初始 `QueryParam` 对象状态。
   - `groups=[]`: 没有分组操作。
   - `aggregators=[...]`: 包含一个聚合操作，列为 `pv`，但聚合函数类型未知（`func=UNKNOWN`）。
   - `dimensionFilters=[]` **和** `metricFilters=[]`: 没有维度或指标的过滤条件。
   - `dateInfo={"dateMode":BETWEEN,"startDate":"2024-08-06","endDate":"2024-07-31",...}`: 这个查询的时间范围在 2024 年 7 月 31 日到 8 月 6 日之间。
   - `limit=2000`: 查询结果的行数限制为 2000 行。
   - `queryType=METRIC`: 查询类型是指标查询。
   - `dataSetId=1`: 查询的数据集 ID 为 1。
2. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.t.DefaultSemanticTranslator 54 - SemanticConverter accept [com.tencent.supersonic.headless.core.translator.converter.DefaultDimValueConverter]**
   - **解释**: `DefaultSemanticTranslator` 接受了一个名为 `DefaultDimValueConverter` 的转换器，这个转换器主要用于处理维度值的转换。

   DefaultDimValueConverter

3. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.t.DefaultSemanticTranslator 54 - SemanticConverter accept [com.tencent.supersonic.headless.core.translator.converter.SqlVariableParseConverter]**
   - **解释**: `DefaultSemanticTranslator` 接受了另一个名为 `SqlVariableParseConverter` 的转换器，这个转换器主要用于解析 SQL 变量。

   SqlVariableParseConverter

4. **14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.t.DefaultSemanticTranslator 58 - SemanticConverter after QueryParam(groups=[], aggregators=[{"column":"pv","func","nameCh":"null","args","alias"}], orders=[], dimensionFilters=[], metricFilters=[], dateInfo={"dateMode","startDate":"2024-08-06","endDate":"2024-07-31","dateList":[],"unit":1,"period":"DAY","text":"null"}, limit=2000, queryType=METRIC, s2SQL=null, correctS2SQL=null, dataSetId=1, dataSetName=null, modelIds=[], params=[], metrics=[pv], dimensions=null, where=null, order=null, nativeQuery=false) DataSetQueryParam(sql=SELECT SUM(pv) FROM t_1 WHERE (sys_imp_date >= '2024-07-31' AND sys_imp_date <= '2024-08-06'), tables=[MetricTable(alias=t_1, metrics=[pv], dimensions=[sys_imp_date], where=null, aggOption=DEFAULT)], supportWith=true, withAlias=true) MetricQueryParam(metrics=null, dimensions=null, where=null, limit=null, order=null, nativeQuery=false)**
   - **解释**: 经过语义转换之后，查询参数被进一步加工并生成了两个更具体的参数对象：
     - `DataSetQueryParam`: 这是一个数据集查询参数，其中包含了生成的 SQL 语句 `SELECT SUM(pv) FROM t_1 WHERE (sys_imp_date >= '2024-07-31' AND sys_imp_date <= '2024-08-06')`。这个 SQL 语句将时间维度 `sys_imp_date` 和指标 `pv` 进行聚合计算。`tables` 参数定义了数据表 `MetricTable`，其中包含别名 `t_1`，用于指定查询的表。
     - `MetricQueryParam`: 这是一个指标查询参数，但日志中显示这个参数还没有具体的指标、维度或其他条件。
5. **14:57:20 [http-nio-9080-exec-2] INFO c.t.s.h.c.t.DefaultSemanticTranslator 78 - parse dataSetQuery [DataSetQueryParam(sql=SELECT SUM(pv) FROM t_1 WHERE (sys_imp_date >= '2024-07-31' AND sys_imp_date <= '2024-08-06'), tables=[MetricTable(alias=t_1, metrics=[pv], dimensions=[sys_imp_date], where=null, aggOption=DEFAULT)], supportWith=true, withAlias=true)]**
   - **解释**: 这行日志表示 `DefaultSemanticTranslator` 正在解析数据集查询（`dataSetQuery`）。
   - `DataSetQueryParam`: 它包含了生成的 SQL 语句 `SELECT SUM(pv) FROM t_1 WHERE (sys_imp_date >= '2024-07-31' AND sys_imp_date <= '2024-08-06')`，该语句聚合了指标 `pv`，并根据时间维度 `sys_imp_date` 进行过滤。`tables` 参数指示了要查询的表 `MetricTable`，表的别名为 `t_1`，并启用了 `with` 子句和别名支持。
6. **14:57:20 [http-nio-9080-exec-2] INFO c.t.s.h.c.t.DefaultSemanticTranslator 125 - parse metricQuery [MetricQueryParam(metrics=[pv], dimensions=[sys_imp_date], where=null, limit=null, order=null, nativeQuery=false)] isAgg [DEFAULT]**
   - **解释**: 这行日志表示 `DefaultSemanticTranslator` 正在解析指标查询（`metricQuery`）。

- `MetricQueryParam`：该参数对象包含了一个指标 `pv` 和一个时间维度 `sys_imp_date`。这个查询不包括 `where` 条件、行数限制（`limit`）、排序（`order`）或本地查询（`nativeQuery=false`）。`isAgg [DEFAULT]` 表示使用了默认的聚合方式。

7. **14:57:20 [http-nio-9080-exec-2] INFO c.t.s.h.c.t.c.s.node.DataSourceNode 204 - dataSourceMeasures [{user_department=0, s2_pv_uv_statis=1, s2_stay_time_statis=0}]**
   - **解释**: 这行日志记录了数据源节点的度量信息。
   - `dataSourceMeasures`：这个信息展示了与数据源相关的度量，其中包括：
     - `user_department=0`：`user_department` 表没有使用。
     - `s2_pv_uv_statis=1`：`s2_pv_uv_statis` 表被使用（值为 1 表示启用）。
     - `s2_stay_time_statis=0`：`s2_stay_time_statis` 表没有使用。
8. 14:57:20 [http-nio-9080-exec-2] DEBUG c.t.s.h.c.t.c.s.node.DataSourceNode 229 - baseDataSource match all
   - `baseDataSource match all`: 这条消息意味着当前的数据源节点（`DataSourceNode`）已经匹配了所有相关的数据源条件。换句话说，系统确定了一个基础数据源，它能够匹配查询中所有的条件或需求。

总结：

展示了 `DefaultSemanticTranslator` 类如何将原始的查询参数（`QueryParam`）通过语义转换器进行处理，解析了数据集查询和指标查询，接着记录了数据源节点中的度量使用情况，最终生成更具体的查询参数（`DataSetQueryParam` 和 `MetricQueryParam`）用于执行实际的 SQL 查询。