**package** org.zc.sched.plugins.spark

**import** java.sql.Connection

**import** java.sql.PreparedStatement

**import** java.sql.Statement

**import** java.sql.ResultSet

**import** java.sql.Timestamp

**import** org.apache.spark.SparkConf

**import** org.apache.spark.sql.SparkSession

**import** org.zc.sched.model.Task

**import** org.zc.sched.model.DBConn

**import** org.zc.sched.util.DBUtil

**import** org.zc.sched.util.ConfigUtil

**import** org.zc.sched.util.DateUtil

**import** org.zc.sched.util.Log

**abstract** **class** TaskExecutor(task: Task) **extends** Serializable **with** Log {

**val** sparkParallelism = ConfigUtil.getInt("spark.parallelism")

**val** parallelism = task.taskExt.getOrElse("parallelism", sparkParallelism).toString.toInt

**val** executorClasspath = task.taskExt.getOrElse(ConfigUtil.getString("spark.executor.classpath"), "")

**val** appName = task.taskName + "-" + task.runTime

**val** sparkConf = **new** SparkConf()

.setAppName(appName)

.setExecutorEnv("SPARK\_CLASSPATH", executorClasspath)

**lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.getOrCreate()

/\*\*

\* 执行任务

\*/

**def** execute

**def** run {

execute

spark.stop()

}

/\*\*

\* 获取数据库连接信息

\*/

**def** getDbConn(id: Int): Option[DBConn] = {

**val** conn = DBUtil.getConn

**val** sql = s"SELECT type\_id, conn\_type, hostname, port, username, password, db\_name, charset FROM t\_db\_conn WHERE id = ${id}"

**val** stmt: Statement = conn.createStatement()

**val** rs: ResultSet = stmt.executeQuery(sql)

**val** db = **if** (rs.next()) {

Some(DBConn(id, rs))

} **else** None

DBUtil.closeAll(rs, stmt, conn)

db

}

/\*\*

\* 记录数据库日志

\*/

**def** logTask(level: Int, content: String) {

**val** conn = DBUtil.getConn

**val** sql = "INSERT INTO t\_task\_log (task\_id, run\_time, seq\_no, level, content, create\_time) VALUES (?, ?, ?, ?, ?, ?)"

**val** ps: PreparedStatement = conn.prepareStatement(sql)

ps.setInt(1, task.taskId)

ps.setTimestamp(2, **new** Timestamp(DateUtil.getDatetime(task.runTime, Task.RUNTIME\_FORMAT).getTime))

ps.setLong(3, task.taskNo)

ps.setInt(4, level)

ps.setString(5, content)

ps.setTimestamp(6, **new** Timestamp(System.currentTimeMillis()))

ps.executeUpdate()

DBUtil.closeAll(**null**, ps, conn)

}

}

**object** TaskExecutor **extends** Log {

**private** **var** conn: Connection = **null**

**private** **var** ps: PreparedStatement = **null**

**private** **var** rs: ResultSet = **null**

**def** main(args: Array[String]): Unit = {

**if** (args.length < 3) {

log.error("Invalid arguments")

System.exit(1)

}

log.info(args.mkString("Argument list: { ", ", ", " }"))

**val** taskId = args(0).toInt

**val** runTime = args(1)

**val** appClass = args(2)

conn = DBUtil.getConn

// 获取任务

log.info("Get task")

**val** task = getTask(taskId, runTime)

**if** (task.isDefined) {

// 获取任务扩展属性

log.info("Get task extended attributes")

task.get.taskExt.++=(getTaskExt(task.get))

log.info("Got task extended attributes: " + task.get.taskExt.mkString("{ ", "\n ", " }"))

}

DBUtil.closeAll(rs, ps, conn)

**if** (task.isDefined) {

Class.forName(appClass).getConstructors.head.newInstance(task.get).asInstanceOf[TaskExecutor].run

} **else** {

log.error(s"Can not find valid task by id: ${taskId}")

System.exit(1)

}

}

/\*\*

\* 获取任务

\*/

**def** getTask(taskId: Int, runTime: String) = {

**val** sql = "SELECT a.name, a.task\_cycle, IF(b.run\_time = b.first\_time, 1, 0) is\_first, b.redo\_flag, b.extra\_param" +

" FROM t\_task a INNER JOIN t\_task\_pool b" +

" ON a.id = b.task\_id AND a.id = ? AND b.run\_time = ?"

ps = conn.prepareStatement(sql)

ps.setInt(1, taskId)

ps.setTimestamp(2, **new** Timestamp(DateUtil.getDatetime(runTime, Task.RUNTIME\_FORMAT).getTime))

rs = ps.executeQuery()

**if** (rs.next()) {

Some(Task(taskId, runTime, rs))

} **else** {

None

}

}

/\*\*

\* 获取任务扩展属性

\*/

**def** getTaskExt(task: Task) = {

**val** taskExt: collection.mutable.Map[String, String] = collection.mutable.Map()

**val** sql = "SELECT prop\_name, prop\_value FROM t\_task\_ext WHERE task\_id = ?"

ps = conn.prepareStatement(sql)

ps.setInt(1, task.taskId)

rs = ps.executeQuery()

**while** (rs.next()) {

taskExt(rs.getString(1)) = rs.getString(2)

}

taskExt

}

}

**package** org.zc.sched.model

**import** java.sql.ResultSet

**import** org.apache.commons.lang3.StringUtils

**import** org.json4s.\_

**import** org.json4s.jackson.JsonMethods.\_

**import** com.fasterxml.jackson.core.JsonParser.Feature

**import** org.zc.sched.util.DateUtil

**case** **class** Task(taskId: Int, runTime: String, taskName: String, taskCycle: String, isFirst: Boolean,

redoFlag: Boolean, extraParam: Map[String, Any]) {

**val** taskNo = DateUtil.formatDate(Task.RUNTIME\_FORMAT).toLong

**val** taskExt: collection.mutable.Map[String, String] = collection.mutable.Map()

**val** theTime = DateUtil.getDatetime(runTime, Task.RUNTIME\_FORMAT)

**val** prevTime = **if** (Task.TASK\_CYCLE\_HOUR.equalsIgnoreCase(taskCycle)) {

DateUtil.nextHour(-1, theTime)

} **else** DateUtil.nextDate(-1, theTime)

**val** theDate = DateUtil.formatDate(theTime)

**val** prevDate = DateUtil.formatDate(DateUtil.nextDate(-1, theTime))

**val** statDate = prevDate.replaceAll("-", "").toInt

}

**object** Task {

**val** RUNTIME\_FORMAT = "yyyyMMddHHmmss"

**val** TASK\_CYCLE\_DAY = "day"

**val** TASK\_CYCLE\_WEEK = "week"

**val** TASK\_CYCLE\_MONTH = "month"

**val** TASK\_CYCLE\_HOUR = "hour"

**val** TASK\_CYCLE\_INTERVAL = "interval"

**val** TASK\_CYCLE\_INSTANT = "instant"

**def** apply(taskId: Int, runTime: String, rs: ResultSet): Task = {

**val** extraParam = rs.getString("extra\_param")

**val** \_extraParam = **if** (StringUtils.isNotBlank(extraParam)) {

mapper.configure(Feature.ALLOW\_UNQUOTED\_FIELD\_NAMES, **true**)

parse(extraParam).values.asInstanceOf[Map[String, Any]]

} **else** Map[String, Any]()

Task(taskId, runTime, rs.getString("name"), rs.getString("task\_cycle"), rs.getBoolean("is\_first"), rs.getBoolean("redo\_flag"), \_extraParam)

}

}

**package** org.zc.sched.model

**import** java.util.Properties

**import** java.sql.ResultSet

**import** org.zc.sched.util.ConfigUtil

**case** **class** DBConn(id: Int, dbType: Int, connType: Int, hostname: String, port: Int,

username: String, password: String, dbName: String, charset: String) {

// 驱动名

**val** JDBC\_DRIVER\_MYSQL = ConfigUtil.getString("jdbc.driver.mysql")

**val** JDBC\_DRIVER\_HIVE = ConfigUtil.getString("jdbc.driver.hive")

**val** JDBC\_DRIVER\_PHOENIX = ConfigUtil.getString("jdbc.driver.phoenix")

**def** jdbcDriver = **if** (dbType > 0) {

dbType **match** {

**case** DBConn.DB\_TYPE\_MYSQL => JDBC\_DRIVER\_MYSQL

**case** DBConn.DB\_TYPE\_HIVE => JDBC\_DRIVER\_HIVE

**case** DBConn.DB\_TYPE\_PHOENIX => JDBC\_DRIVER\_PHOENIX

**case** \_ => **throw** **new** RuntimeException(s"Unsupported database type: ${dbType}")

}

} **else** { **null** }

**def** jdbcUrl = **if** (dbType > 0) {

dbType **match** {

**case** DBConn.DB\_TYPE\_MYSQL => s"jdbc:mysql://${hostname}:${port}/${dbName}?useUnicode=true&characterEncoding=${charset}"

**case** DBConn.DB\_TYPE\_HIVE =>

**if** (connType == DBConn.CONN\_TYPE\_ZOOKEEPER) s"jdbc:hive2://${hostname}/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2"

**else** s"jdbc:hive2://${hostname}:${port}/${dbName}"

**case** DBConn.DB\_TYPE\_PHOENIX => s"jdbc:phoenix:${hostname}"

**case** \_ => **throw** **new** RuntimeException(s"Unsupported database type: ${dbType}")

}

} **else** { **null** }

**def** connProps = {

**val** props = **new** Properties

props.put("user", username)

props.put("password", password)

props

}

}

**object** DBConn {

// 数据库类型

**val** DB\_TYPE\_MYSQL = 1

**val** DB\_TYPE\_ORACLE = 2

**val** DB\_TYPE\_MSSQLSERVER = 3

**val** DB\_TYPE\_SYBASE = 4

**val** DB\_TYPE\_POSTGRESQL = 5

**val** DB\_TYPE\_DB2 = 6

**val** DB\_TYPE\_HIVE = 7

**val** DB\_TYPE\_DERBY = 8

**val** DB\_TYPE\_FS = 9

**val** DB\_TYPE\_HDFS = 10

**val** DB\_TYPE\_SAIKU = 11

**val** DB\_TYPE\_PHOENIX = 12

// 数据库连接方式

**val** CONN\_TYPE\_CLI = 0

**val** CONN\_TYPE\_JDBC = 1

**val** CONN\_TYPE\_ODBC = 2

**val** CONN\_TYPE\_HTTP = 3

**val** CONN\_TYPE\_ZOOKEEPER = 4

**def** apply(id: Int, rs: ResultSet): DBConn = {

DBConn(id, rs.getInt("type\_id"), rs.getInt("conn\_type"), rs.getString("hostname"), rs.getInt("port"),

rs.getString("username"), rs.getString("password"), rs.getString("db\_name"), rs.getString("charset"))

}

}

**package** com.jiuzhi.etl.fad

**import** java.sql.SQLException

**import** com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.model.Task

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.util.DateUtil

**import** org.zc.sched.util.JdbcUtil

**import** com.jiuzhi.etl.fad.model.Device

/\*\*

\* 解析访问日志得到dim\_device

\*/

**class** DimDevice(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// visit log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** visitLogPath = rootPath + task.prevDate + "/" + topic

**val** newVisitLogPath = rootPath + task.theDate + "/" + topic

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**def** execute {

// 任务重做

**if** (task.redoFlag) {

JdbcUtil.executeUpdate(adDb, "TRUNCATE TABLE dim\_device")

JdbcUtil.executeUpdate(adDb, s"INSERT INTO dim\_device SELECT \* FROM dim\_device\_${task.statDate}")

}

// 读取hdfs json文件

**val** visitlog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_device\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "deviceid", "imsi", "imei", "vender",

"model", "osVersion", "platform", "androidid", "operator",

"CASE WHEN network = 'GPRS' THEN '1GPRS' WHEN network = 'Wifi' THEN '0WIFI' WHEN network = 'Unknown' THEN '-Unknown' ELSE network END",

"src", "mac", "apppkg", "clnt", "CAST(isRoot AS INT)",

"CAST(gp AS INT)", "gaid", "rom", "lang", "ua",

"cityId", "country", "createTime", "updateTime")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(visitLogPath, newVisitLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "deviceid", "imsi", "imei", "vender",

"model", "osVersion", "platform", "androidid", "operator",

"CASE WHEN network = 'GPRS' THEN '1GPRS' WHEN network = 'Wifi' THEN '0WIFI' WHEN network = 'Unknown' THEN '-Unknown' ELSE network END",

"src", "mac", "apppkg", "clnt", "CAST(isroot AS INT)",

"CAST(gp AS INT)", "gaid", "CAST(rom AS LONG)", "lang", "ua",

"CAST(cityid AS LONG)", "country", "CAST(createtime AS TIMESTAMP) create\_time", "CAST(updatetime AS TIMESTAMP) update\_time")

}

**if** (log.isDebugEnabled) {

visitlog.printSchema

visitlog.show(50, **false**)

}

// 读取dim\_device

**val** device = spark.read.jdbc(adDb.jdbcUrl, "dim\_device", adDb.connProps)

.selectExpr("udid", "deviceid", "imsi", "imei", "vender",

"model", "os\_version", "platform", "android\_id", "operator",

"CASE WHEN network = 'GPRS' THEN '1GPRS' WHEN network = 'Wifi' THEN '0WIFI' WHEN network = 'Unknown' THEN '-Unknown' ELSE network END",

"src", "mac", "app\_key", "clnt", "is\_root",

"has\_gplay", "gaid", "rom", "lang", "ua",

"city\_id", "country", "create\_time", "update\_time")

**if** (log.isDebugEnabled) {

device.printSchema

device.show(50, **false**)

}

**import** spark.implicits.\_

// 合并

**val** result = visitlog.union(device)

.map(Device(\_)).rdd

.groupBy(\_.udid)

.map { row => (row.\_1, row.\_2.toSeq.sortBy(\_.update\_time.getTime)) }

.map(\_.\_2.reduceLeft { (acc, curr) => Device.update(acc, curr) })

.map(Device.finalize(\_))

.toDF()

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 写入临时表

**val** tmpTable = "tmp\_dim\_device\_" + System.currentTimeMillis()

JdbcUtil.executeUpdate(adDb, s"CREATE TABLE ${tmpTable} LIKE dim\_device")

**try** {

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, tmpTable, adDb.connProps)

} **catch** {

**case** e: SQLException =>

JdbcUtil.executeUpdate(adDb, s"DROP TABLE IF EXISTS ${tmpTable}")

**throw** **new** RuntimeException(e)

}

// 备份dim\_device

**try** {

JdbcUtil.executeUpdate(adDb, s"RENAME TABLE dim\_device TO dim\_device\_${task.statDate}")

} **catch** {

**case** \_: MySQLSyntaxErrorException =>

log.info(s"Table dim\_device\_${task.statDate} already exists")

JdbcUtil.executeUpdate(adDb, "DROP TABLE dim\_device")

}

// 更新dim\_device

JdbcUtil.executeUpdate(adDb, s"RENAME TABLE ${tmpTable} TO dim\_device")

// 删除历史数据

**val** prevDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-3, task.theTime))

JdbcUtil.executeUpdate(adDb, s"DROP TABLE IF EXISTS dim\_device\_${prevDate}")

}

}

**package** com.jiuzhi.etl.fad.model

**import** java.sql.Timestamp

**import** org.apache.spark.sql.Row

**case** **class** Device(udid: String, deviceid: String, **var** imsi: String, imei: String, vender: String,

model: String, os\_version: String, platform: String, android\_id: String, operator: String,

**var** network: String, src: String, mac: String, app\_key: String, clnt: String,

is\_root: Int, has\_gplay: Int, gaid: String, rom: Long, lang: String,

ua: String, city\_id: Long, country: String, create\_time: Timestamp, **var** update\_time: Timestamp)

**object** Device {

**def** apply(row: Row): Device = {

Device(row.getString(0), row.getString(1), row.getString(2), row.getString(3), row.getString(4),

row.getString(5), row.getString(6), row.getString(7), row.getString(8), row.getString(9),

row.getString(10), row.getString(11), row.getString(12), row.getString(13), row.getString(14),

row.getInt(15), row.getInt(16), row.getString(17), row.getLong(18), row.getString(19),

row.getString(20), row.getLong(21), row.getString(22), row.getTimestamp(23), row.getTimestamp(24))

}

**def** update(acc: Device, curr: Device): Device = {

**if** ("null".equalsIgnoreCase(acc.imsi) || "".equals(acc.imsi)) acc.imsi = curr.imsi

**if** (acc.network < curr.network) acc.network = curr.network

acc.update\_time = curr.update\_time

acc

}

**def** finalize(device: Device): Device = {

**if** (device.network < "2") device.network = device.network.substring(1)

device

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.JdbcUtil

**import** com.jiuzhi.etl.fad.model.Active

/\*\*

\* 解析访问日志得到fact\_active

\*/

**class** FactActive(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// visit log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** visitLogPath = rootPath + task.prevDate + "/" + topic

**val** newVisitLogPath = rootPath + task.theDate + "/" + topic

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**def** execute {

// 读取hdfs json文件

**val** visitlog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_device\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "apppkg", "DATE\_FORMAT(createtime, 'yyyy-MM-dd') AS active\_date", "appVersion", "CAST(cityId AS LONG)", "country", "createtime")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(visitLogPath, newVisitLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg", "DATE\_FORMAT(createtime, 'yyyy-MM-dd') AS active\_date", "appversion", "CAST(cityid AS LONG)", "country", "CAST(createtime AS TIMESTAMP)")

}

**if** (log.isDebugEnabled) {

visitlog.printSchema

visitlog.show(50, **false**)

}

**import** spark.implicits.\_

// 分析

**val** active = visitlog.map(Active(\_)).rdd

.groupBy(row => row.udid + row.app\_key + row.active\_date)

.map {

\_.\_2.toSeq.sortBy(\_.create\_time.getTime)

.reduceLeft { (acc, curr) => Active.update(acc, curr) }

}

.toDF()

.drop("create\_time")

**if** (log.isDebugEnabled) {

active.printSchema

active.show(50, **false**)

}

// 读取fact\_client

**val** client = spark.read.jdbc(adDb.jdbcUrl, "fact\_client", adDb.connProps)

.selectExpr("udid", "app\_key", "clnt", "init\_version", "create\_date", "DATE\_FORMAT(create\_time, 'yyyy-MM-dd') AS \_create\_date")

**if** (log.isDebugEnabled) {

client.printSchema

client.show(50, **false**)

}

// 关联fact\_client获取init\_version clnt create\_date等信息

**val** result = active.join(client, Seq("udid", "app\_key"))

.selectExpr("udid", "app\_key", "CAST(DATE\_FORMAT(active\_date, 'yyyyMMdd') AS INT) AS active\_date", "version", "city\_id",

"country", "visit\_times", "clnt", "init\_version", "create\_date", "DATEDIFF(active\_date, \_create\_date) AS date\_diff")

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

JdbcUtil.executeUpdate(adDb, s"DELETE FROM fact\_active WHERE active\_date = ${task.statDate}")

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, "fact\_active", adDb.connProps)

}

}

**package** com.jiuzhi.etl.fad.model

**import** java.sql.Timestamp

**import** org.apache.spark.sql.Row

**case** **class** Active(udid: String, app\_key: String, active\_date: String, version: String, city\_id: Long,

country: String, create\_time: Timestamp, **var** visit\_times: Int = 1)

**object** Active {

**def** apply(row: Row): Active = {

Active(row.getString(0), row.getString(1), row.getString(2), row.getString(3), row.getLong(4),

row.getString(5), row.getTimestamp(6))

}

**def** update(acc: Active, curr: Active) = {

acc.visit\_times = acc.visit\_times + curr.visit\_times

acc

}

}

**package** com.jiuzhi.etl.fad

**import** java.sql.SQLException

**import** com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.DateUtil

**import** org.zc.sched.util.JdbcUtil

**import** com.jiuzhi.etl.fad.model.Client

/\*\*

\* 解析访问日志得到fact\_client

\*/

**class** FactClient(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// visit log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** visitLogPath = rootPath + task.prevDate + "/" + topic

**val** newVisitLogPath = rootPath + task.theDate + "/" + topic

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**def** execute {

// 任务重做

**if** (task.redoFlag) {

JdbcUtil.executeUpdate(adDb, "TRUNCATE TABLE fact\_client")

JdbcUtil.executeUpdate(adDb, s"INSERT INTO fact\_client SELECT \* FROM fact\_client\_${task.statDate}")

}

// 读取hdfs json文件

**val** visitlog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_device\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "apppkg", "clnt", "appVersion", "appVersion init\_version",

"CAST(path AS INT)", "createTime", "updateTime", "CAST(DATE\_FORMAT(createTime, 'yyyyMMdd') AS INT)")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(visitLogPath, newVisitLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg", "clnt", "appversion", "appversion init\_version",

"sdkver", "sdkver init\_sdkver", "CAST(path AS INT)", "CAST(createtime AS TIMESTAMP) create\_time", "CAST(updatetime AS TIMESTAMP) update\_time",

"CAST(DATE\_FORMAT(createtime, 'yyyyMMdd') AS INT)")

}

**if** (log.isDebugEnabled) {

visitlog.printSchema

visitlog.show(50, **false**)

}

// 读取fact\_client

**val** client = spark.read.jdbc(adDb.jdbcUrl, "fact\_client", adDb.connProps)

.select("udid", "app\_key", "clnt", "version", "init\_version", "sdkver", "init\_sdkver", "app\_path", "create\_time", "update\_time", "create\_date")

**if** (log.isDebugEnabled) {

client.printSchema

client.show(50, **false**)

}

**import** spark.implicits.\_

// 合并

**val** result = visitlog.union(client)

.map(Client(\_)).rdd

.groupBy(row => row.udid + row.app\_key)

.map { row => (row.\_1, row.\_2.toSeq.sortBy(\_.update\_time.getTime)) }

.map(\_.\_2.reduceLeft { (acc, curr) => Client.update(acc, curr) })

.toDF()

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 写入临时表

**val** tmpTable = "tmp\_fact\_client\_" + System.currentTimeMillis()

JdbcUtil.executeUpdate(adDb, s"CREATE TABLE ${tmpTable} LIKE fact\_client")

**try** {

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, tmpTable, adDb.connProps)

} **catch** {

**case** e: SQLException =>

JdbcUtil.executeUpdate(adDb, s"DROP TABLE IF EXISTS ${tmpTable}")

**throw** **new** RuntimeException(e)

}

// 备份fact\_client

**try** {

JdbcUtil.executeUpdate(adDb, s"RENAME TABLE fact\_client TO fact\_client\_${task.statDate}")

} **catch** {

**case** \_: MySQLSyntaxErrorException =>

log.info(s"Table fact\_client\_${task.statDate} already exists")

JdbcUtil.executeUpdate(adDb, "DROP TABLE fact\_client")

}

// 更新fact\_client

JdbcUtil.executeUpdate(adDb, s"RENAME TABLE ${tmpTable} TO fact\_client")

// 删除历史数据

**val** prevDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-3, task.theTime))

JdbcUtil.executeUpdate(adDb, s"DROP TABLE IF EXISTS fact\_client\_${prevDate}")

}

}

**package** com.jiuzhi.etl.fad.model

**import** java.sql.Timestamp

**import** org.apache.spark.sql.Row

**case** **class** Client(udid: String, app\_key: String, clnt: String, **var** version: String, init\_version: String,

**var** sdkver: String, init\_sdkver: String, app\_path: Int, create\_time: Timestamp, **var** update\_time: Timestamp,

create\_date: Int)

**object** Client {

**def** apply(row: Row): Client = {

Client(row.getString(0), row.getString(1), row.getString(2), row.getString(3), row.getString(4),

row.getString(5), row.getString(6), row.getInt(7), row.getTimestamp(8), row.getTimestamp(9),

row.getInt(10))

}

**def** update(acc: Client, curr: Client) = {

acc.version = curr.version

acc.sdkver = curr.sdkver

acc.update\_time = curr.update\_time

acc

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.model.Task

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.util.JdbcUtil

/\*\*

\* 黑名单统计fact\_blacklist

\*/

**class** FactBlacklist(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

// 广告数据源

**val** srcAdDb = getDbConn(task.taskExt.get("ad\_srcdb\_id").get.toInt).get

**def** execute {

// 读取fact\_client

**val** client = spark.read.jdbc(adDb.jdbcUrl, "fact\_client", adDb.connProps)

.selectExpr("app\_key", "clnt", s"DATEDIFF('${task.prevDate}', create\_time) AS date\_diff")

// 读取adv\_customer

**val** customer = spark.read.jdbc(srcAdDb.jdbcUrl, "adv\_customer", srcAdDb.connProps)

.selectExpr("customid AS clnt", "blackdate")

// 关联

**val** result = client.join(broadcast(customer), "clnt")

.groupBy("app\_key", "clnt")

.agg(count(when(col("date\_diff") < col("blackdate"), 1)).alias("black\_count"), count(when(col("date\_diff") + 1 === col("blackdate"), 1)).alias("release\_count"))

.withColumn("stat\_date", lit(s"${task.statDate}"))

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

JdbcUtil.executeUpdate(adDb, s"DELETE FROM fact\_blacklist WHERE stat\_date = ${task.statDate}")

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, "fact\_blacklist", adDb.connProps)

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.JdbcUtil

/\*\*

\* 解析访问日志得到运营级别信息fact\_runlevel

\*/

**class** FactRunlevel(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// visit log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** visitLogPath = rootPath + task.prevDate + "/" + topic

**val** newVisitLogPath = rootPath + task.theDate + "/" + topic

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**def** execute {

// 读取hdfs json文件

**val** visitLog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_device\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "apppkg AS app\_key", "clnt", "runlevel", "CAST(DATE\_FORMAT(createTime, 'yyyyMMdd') AS INT) AS create\_date")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(visitLogPath, newVisitLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg AS app\_key", "clnt", "runlevel", "CAST(DATE\_FORMAT(createtime, 'yyyyMMdd') AS INT) AS create\_date")

}

**if** (log.isDebugEnabled) {

visitLog.printSchema

visitLog.show(50, **false**)

}

**val** result = visitLog.groupBy("app\_key", "clnt", "runlevel", "create\_date")

.agg(countDistinct("udid").alias("user\_count"))

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

JdbcUtil.executeUpdate(adDb, s"DELETE FROM fact\_runlevel WHERE create\_date = ${task.statDate}")

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, "fact\_runlevel", adDb.connProps)

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SparkSession

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.DateUtil

**import** org.zc.sched.util.JdbcUtil

/\*\*

\* 取一段时间的mid\_upgrade得到fact\_upgrade

\*/

**class** FactUpgrade(task: Task) **extends** TaskExecutor(task) **with** Serializable {

**override** **lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.enableHiveSupport()

.getOrCreate()

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

// 天数

**val** days = task.taskExt.getOrElse("days", 60).toString.toInt

**val** startDate = DateUtil.formatDate(DateUtil.nextDate(-days))

**val** dbName = task.taskExt.get("hive\_db").get

**def** execute {

// 读取mid\_upgrade

**val** sql = s"SELECT udid, app\_key, version, up\_version, create\_date, upgrade\_date" +

" FROM mid\_upgrade" +

s" WHERE stat\_date = '${task.statDate}'" +

s" AND create\_date >= '${startDate}'"

spark.sql("USE " + dbName)

**val** upgrade = spark.sql(sql)

**if** (log.isDebugEnabled) {

upgrade.printSchema

upgrade.show(50, **false**)

}

// 读取fact\_client

**val** client = spark.read.jdbc(adDb.jdbcUrl, "fact\_client", adDb.connProps)

.select("udid", "app\_key", "clnt")

**if** (log.isDebugEnabled) {

client.printSchema

client.show(50, **false**)

}

// 关联得到clnt

**val** result = upgrade.join(client, Seq("udid", "app\_key"))

.selectExpr("udid", "app\_key", "version", "up\_version", "DATE\_FORMAT(create\_date, 'yyyyMMdd') AS create\_date", "DATE\_FORMAT(upgrade\_date, 'yyyyMMdd') AS upgrade\_date", "clnt")

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

JdbcUtil.executeUpdate(adDb, "TRUNCATE TABLE fact\_upgrade")

result.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, "fact\_upgrade", adDb.connProps)

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SparkSession

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

/\*\*

\* 解析升级下发日志得到log\_upgrade

\*/

**class** LogUpgrade(task: Task) **extends** TaskExecutor(task) **with** Serializable {

**override** **lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.enableHiveSupport()

.getOrCreate()

// upgrade log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** upgradeLogPath = rootPath + task.prevDate + "/" + topic

**val** newUpgradeLogPath = rootPath + task.theDate + "/" + topic

// 精确模式

**val** strict = task.taskExt.getOrElse("is\_strict", 1).toString.toInt

**val** dbName = task.taskExt.get("hive\_db").get

**def** execute {

// 读取hdfs json文件

**val** upgrade = **if** (task.isFirst) {

// 初始化从MySQL数据库读

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

spark.read.jdbc(adDb.jdbcUrl, "t\_upgrade\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "apppkg", "appVersion", "upVersion", "TO\_DATE(createtime)")

} **else** {

**if** (strict == 1) {

spark.read.option("allowUnquotedFieldNames", **true**).json(upgradeLogPath, newUpgradeLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg", "appversion", "upversion", "TO\_DATE(createtime)")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(upgradeLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg", "appversion", "upversion", "TO\_DATE(createtime)")

}

}

**if** (log.isErrorEnabled) {

upgrade.printSchema

upgrade.show(50, **false**)

}

// 入库

upgrade.distinct

.coalesce(parallelism)

.createOrReplaceTempView("tmp\_log\_upgrade")

spark.sql("USE " + dbName)

spark.sql(s"ALTER TABLE log\_upgrade DROP IF EXISTS PARTITION(stat\_date = '${task.statDate}') PURGE")

spark.sql(s"INSERT INTO log\_upgrade PARTITION(stat\_date = '${task.statDate}') SELECT \* FROM tmp\_log\_upgrade")

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SparkSession

**import** org.apache.spark.sql.SaveMode

**import** org.apache.spark.sql.AnalysisException

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.DateUtil

/\*\*

\* 解析广告下发反馈日志得到广告下发反馈信息mid\_ad

\*/

**class** MidAd(task: Task) **extends** TaskExecutor(task) **with** Serializable {

**override** **lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.enableHiveSupport()

.getOrCreate()

// ad request log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** requestTopic = task.taskExt.get("request\_topic").get

**val** requestLogPath = rootPath + task.prevDate + "/" + requestTopic

**val** newRequestLogPath = rootPath + task.theDate + "/" + requestTopic

// ad reply log文件目录

**val** replyTopic = task.taskExt.get("reply\_topic").get

**val** replyLogPath = rootPath + task.prevDate + "/" + replyTopic

**val** newReplyLogPath = rootPath + task.theDate + "/" + replyTopic

// 精确模式

**val** strict = task.taskExt.getOrElse("is\_strict", 1).toString.toInt

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**val** dbName = task.taskExt.get("hive\_db").get

**def** execute {

// 读取hdfs json下发日志

**val** requestLog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_req\_ad", Array(s"createtime < '${task.theDate}'"), adDb.connProps)

.selectExpr("adcode", "udid", "adkey", "0 adverid", "position", "apppkg", "clnt", "cityId", "createtime")

} **else** {

**if** (strict == 1) {

spark.read.option("allowUnquotedFieldNames", **true**).json(requestLogPath, newRequestLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.select("adcode", "udid", "adkey", "adverid", "position", "apppkg", "clnt", "cityid", "createtime")

} **else** {

**try** {

spark.read.option("allowUnquotedFieldNames", **true**).json(requestLogPath)

.select("adcode", "udid", "adkey", "adverid", "position", "apppkg", "clnt", "cityid", "createtime")

} **catch** {

**case** e: AnalysisException **if** (e.getMessage().contains("cannot resolve")) => spark.read.option("allowUnquotedFieldNames", **true**).json(requestLogPath)

.selectExpr("adcode", "udid", "adkey", "0 adverid", "position", "apppkg", "clnt", "cityid", "createtime")

**case** e: AnalysisException **if** (e.getMessage().contains("Path does not exist")) => spark.emptyDataFrame

}

}

}

**if** (log.isDebugEnabled) {

requestLog.printSchema

requestLog.show(50, **false**)

}

// 联合mid\_ad得到全量下发日志

**val** prevDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-2, task.theTime))

spark.sql("USE " + dbName)

**val** send = **if** (requestLog.columns.isEmpty) {

spark.sql(s"SELECT adcode, udid, adkey, adverid, position, app\_key, clnt, city\_id, send\_time FROM mid\_ad WHERE stat\_date = '${prevDate}'")

} **else** {

spark.sql(s"SELECT adcode, udid, adkey, adverid, position, app\_key, clnt, city\_id, send\_time FROM mid\_ad WHERE stat\_date = '${prevDate}'")

.union(requestLog)

}

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

send.printSchema

send.show(50, **false**)

}

// 读取hdfs json反馈日志

**val** replyLog = **if** (task.isFirst) {

spark.read.jdbc(adDb.jdbcUrl, "t\_channel\_stat", Array(s"createtime < '${task.theDate}'"), adDb.connProps)

.select("adcode", "udid", "createtime", "rtype")

} **else** {

**if** (strict == 1) {

spark.read.option("allowUnquotedFieldNames", **true**).json(replyLogPath, newReplyLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.select("adcode", "udid", "createtime", "rtype")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(replyLogPath)

.select("adcode", "udid", "createtime", "rtype")

}

}

**if** (log.isDebugEnabled) {

replyLog.printSchema

replyLog.show(50, **false**)

}

// 联合mid\_ad得到全量展现日志

**val** show = spark.sql(s"SELECT adcode, udid, show\_time FROM mid\_ad WHERE stat\_date = '${prevDate}' AND show\_time IS NOT NULL")

.union(replyLog.where("rtype = 1").drop("rtype"))

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

show.printSchema

show.show(50, **false**)

}

// 联合mid\_ad得到全量点击点击

**val** click = spark.sql(s"SELECT adcode, udid, click\_time FROM mid\_ad WHERE stat\_date = '${prevDate}' AND click\_time IS NOT NULL")

.union(replyLog.where("rtype = 2").drop("rtype"))

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

click.printSchema

click.show(50, **false**)

}

// 联合mid\_ad得到全量安装日志

**val** install = spark.sql(s"SELECT adcode, udid, install\_time FROM mid\_ad WHERE stat\_date = '${prevDate}' AND install\_time IS NOT NULL")

.union(replyLog.where("rtype = 3").drop("rtype"))

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

install.printSchema

install.show(50, **false**)

}

// 联合mid\_ad得到全量关闭日志

**val** close = spark.sql(s"SELECT adcode, udid, close\_time FROM mid\_ad WHERE stat\_date = '${prevDate}' AND close\_time IS NOT NULL")

.union(replyLog.where("rtype = 4").drop("rtype"))

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

close.printSchema

close.show(50, **false**)

}

// 联合mid\_ad得到全量卸载日志

**val** uninstall = spark.sql(s"SELECT adcode, udid, uninstall\_time FROM mid\_ad WHERE stat\_date = '${prevDate}' AND uninstall\_time IS NOT NULL")

.union(replyLog.where("rtype = 5").drop("rtype"))

.dropDuplicates(Seq("adcode", "udid"))

**if** (log.isDebugEnabled) {

uninstall.printSchema

uninstall.show(50, **false**)

}

// 关联

**val** result = send.join(show, Seq("adcode", "udid"), "left")

.join(click, Seq("adcode", "udid"), "left")

.join(install, Seq("adcode", "udid"), "left")

.join(close, Seq("adcode", "udid"), "left")

.join(uninstall, Seq("adcode", "udid"), "left")

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

result.createOrReplaceTempView("tmp\_mid\_ad")

spark.sql(s"ALTER TABLE mid\_ad DROP IF EXISTS PARTITION(stat\_date = '${task.statDate}') PURGE")

spark.sql(s"INSERT INTO mid\_ad PARTITION(stat\_date = '${task.statDate}') SELECT \* FROM tmp\_mid\_ad")

// 删除历史分区

**val** oldDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-4, task.theTime))

spark.sql(s"ALTER TABLE mid\_ad DROP IF EXISTS PARTITION(stat\_date = '${oldDate}') PURGE")

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SparkSession

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.DateUtil

/\*\*

\* 关联log\_upgrade和mid\_version得到设备App升级信息mid\_upgrade

\*/

**class** MidUpgrade(task: Task) **extends** TaskExecutor(task) **with** Serializable {

**override** **lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.enableHiveSupport()

.getOrCreate()

**val** dbName = task.taskExt.get("hive\_db").get

**def** execute {

// 读取升级下发日志log\_upgrade

spark.sql("USE " + dbName)

**val** upgradeLog = spark.sql("SELECT udid, app\_key, version, up\_version, MIN(create\_date) AS create\_date FROM log\_upgrade GROUP BY udid, app\_key, version, up\_version")

**import** spark.implicits.\_

// 读取用户使用App版本记录

**val** version = spark.sql(s"SELECT udid, app\_key, version, TO\_DATE(create\_time) FROM mid\_version WHERE stat\_date = '${task.statDate}' ORDER BY create\_time")

.map { row => (row.getString(0), row.getString(1), row.getString(2), row.getDate(3)) }.rdd

.groupBy(row => row.\_1 + row.\_2)

.flatMap { row =>

**val** left = row.\_2.dropRight(1)

**val** right = row.\_2.drop(1)

left.zip(right)

.map { row =>

(row.\_1.\_1, row.\_1.\_2, row.\_1.\_3, row.\_2.\_3, row.\_2.\_4)

}

}.toDF("udid", "app\_key", "version", "up\_version", "update\_date")

**if** (log.isDebugEnabled) {

version.printSchema

version.show(50, **false**)

}

// 关联版本

**val** upgrade = upgradeLog.join(version, Seq("udid", "app\_key", "version", "up\_version"), "left")

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

upgrade.printSchema

upgrade.show(50, **false**)

}

// 入库

upgrade.createOrReplaceTempView("tmp\_mid\_upgrade")

spark.sql(s"ALTER TABLE mid\_upgrade DROP IF EXISTS PARTITION(stat\_date = '${task.statDate}') PURGE")

spark.sql(s"INSERT INTO mid\_upgrade PARTITION(stat\_date = '${task.statDate}') SELECT \* FROM tmp\_mid\_upgrade")

// 删除历史分区

**val** preDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-2, task.theTime))

spark.sql(s"ALTER TABLE mid\_upgrade DROP IF EXISTS PARTITION(stat\_date = '${preDate}') PURGE")

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.SparkSession

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.DateUtil

**import** org.zc.sched.util.JdbcUtil

/\*\*

\* 解析访问日志得到用户使用App版本信息mid\_version

\*/

**class** MidVersion(task: Task) **extends** TaskExecutor(task) **with** Serializable {

**override** **lazy** **val** spark = SparkSession

.builder()

.config(sparkConf)

.enableHiveSupport()

.getOrCreate()

// visit log文件目录

**val** rootPath = task.taskExt.get("root\_path").get

**val** topic = task.taskExt.get("topic").get

**val** visitLogPath = rootPath + task.prevDate + "/" + topic

**val** newVisitLogPath = rootPath + task.theDate + "/" + topic

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

**val** dbName = task.taskExt.get("hive\_db").get

**def** execute {

// 读取hdfs json文件

**val** visitlog = **if** (task.isFirst) {

// 初始化从MySQL数据库读

spark.read.jdbc(adDb.jdbcUrl, "t\_device\_logs", Array(s"createTime < '${task.theDate}'"), adDb.connProps)

.selectExpr("udid", "apppkg", "appVersion", "UNIX\_TIMESTAMP(createTime)")

} **else** {

spark.read.option("allowUnquotedFieldNames", **true**).json(visitLogPath, newVisitLogPath)

.where(s"createtime >= '${task.prevDate}' AND createtime < '${task.theDate}' AND udid > ''")

.selectExpr("udid", "apppkg", "appversion", "UNIX\_TIMESTAMP(createtime)")

}

**if** (log.isDebugEnabled) {

visitlog.printSchema

visitlog.show(50, **false**)

}

// 读取mid\_version

**val** prevDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-2, task.theTime))

spark.sql("USE " + dbName)

**val** version = spark.sql(s"SELECT udid, app\_key, version, UNIX\_TIMESTAMP(create\_time) AS ctimestamp FROM mid\_version WHERE stat\_date = '${prevDate}'")

// 联合

**val** result = version.union(visitlog)

.groupBy("udid", "app\_key", "version")

.agg(min("ctimestamp").alias("mtimestamp"))

.selectExpr("udid", "app\_key", "version", "FROM\_UNIXTIME(mtimestamp)")

.coalesce(parallelism)

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

result.createOrReplaceTempView("tmp\_mid\_version")

spark.sql(s"ALTER TABLE mid\_version DROP IF EXISTS PARTITION(stat\_date = '${task.statDate}') PURGE")

spark.sql(s"INSERT INTO mid\_version PARTITION(stat\_date = '${task.statDate}') SELECT \* FROM tmp\_mid\_version")

// 删除历史分区

**val** oldDate = DateUtil.formatDate("yyyyMMdd", DateUtil.nextDate(-4, task.theTime))

spark.sql(s"ALTER TABLE mid\_version DROP IF EXISTS PARTITION(stat\_date = '${oldDate}') PURGE")

// 更新dim\_version

JdbcUtil.executeUpdate(adDb, "TRUNCATE TABLE dim\_version")

result.selectExpr("version AS id")

.distinct

.coalesce(parallelism)

.write.mode(SaveMode.Append).jdbc(adDb.jdbcUrl, "dim\_version", adDb.connProps)

}

}

**package** com.jiuzhi.etl.fad

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.sql.SaveMode

**import** org.zc.sched.plugins.spark.TaskExecutor

**import** org.zc.sched.model.Task

**import** org.zc.sched.util.JdbcUtil

/\*\*

\* 开放数据

\*/

**class** OpenData(task: Task) **extends** TaskExecutor(task) **with** Serializable {

// 广告数据库

**val** adDb = getDbConn(task.taskExt.get("ad\_db\_id").get.toInt).get

// 客户数据库

**val** cusDb = getDbConn(task.taskExt.getOrElse("cus\_db\_id", 0).toString.toInt)

// 其他目标数据库

**val** tarDbs = task.taskExt.getOrElse("tar\_db\_ids", "").split(",").map(x => getDbConn(x.toInt).get)

// 包含渠道

**val** includeChannels = task.taskExt.getOrElse("include\_channels", "")

// 排除渠道

**val** excludeChannels = task.taskExt.getOrElse("exclude\_channels", "")

// 测试

**val** debug = task.taskExt.getOrElse("debug", 0).toString.toInt

**def** execute {

**val** result = **if** (task.isFirst) {

spark.read.jdbc(adDb.jdbcUrl, "fact\_client", Array("app\_key > '' AND clnt > ''"), adDb.connProps)

.groupBy("create\_date", "clnt")

.agg(count("\*").alias("csum"))

.selectExpr("CONCAT(SUBSTR(create\_date, 0, 4), '-', SUBSTR(create\_date, 5, 2), '-', SUBSTR(create\_date, 7, 2)) AS cdate", "clnt AS chacode", "csum")

.coalesce(parallelism)

} **else** {

spark.read.jdbc(adDb.jdbcUrl, "fact\_client", Array(s"create\_date = '${task.statDate}' AND app\_key > '' AND clnt > ''"), adDb.connProps)

.groupBy("clnt")

.agg(count("\*").alias("csum"))

.selectExpr("clnt AS chacode", "csum")

.withColumn("cdate", lit(s"${task.prevDate}"))

.coalesce(parallelism)

}

**if** (log.isDebugEnabled) {

result.printSchema

result.show(50, **false**)

}

// 入库

**if** (debug == 0) {

**if** (cusDb.isDefined) {

**val** db = cusDb.get

JdbcUtil.executeUpdate(db, s"DELETE FROM t\_channeldaily WHERE cdate = '${task.prevDate}'")

result.write.mode(SaveMode.Append).jdbc(db.jdbcUrl, "t\_channeldaily", db.connProps)

}

} **else** {

result.show(10000, **false**)

}

// 过滤渠道

**val** result1 = **if** (includeChannels != "") {

**val** includes = includeChannels.split(",").mkString("'", "','", "'")

result.where(s"clnt IN (${includes})")

} **else** **if** (excludeChannels != "") {

**val** excludes = excludeChannels.split(",").mkString("'", "','", "'")

result.where(s"clnt NOT IN (${excludes})")

} **else** {

result

}

// 入库

**if** (debug == 0) {

result1.cache

tarDbs.foreach { db =>

**val** filter = **if** (includeChannels != "") {

**val** includes = includeChannels.split(",").mkString("'", "','", "'")

s"AND chacode IN (${includes})"

} **else** **if** (excludeChannels != "") {

**val** excludes = excludeChannels.split(",").mkString("'", "','", "'")

s"AND chacode NOT IN (${excludes})"

} **else** { "" }

JdbcUtil.executeUpdate(db, s"DELETE FROM t\_channeldaily WHERE cdate = '${task.prevDate}' ${filter}")

result1.write.mode(SaveMode.Append).jdbc(db.jdbcUrl, "t\_channeldaily", db.connProps)

}

} **else** {

result1.show(10000, **false**)

}

}

}

task\_manager.sh

#!/bin/bash

# 任务管理器

# 1、清理任务池一段时间的历史任务

# 2、根据任务周期实例化任务到任务池

# 3、检查状态为“等待”的任务的依赖关系，满足执行条件则更新状态为“就绪”

BASE\_DIR=`pwd`

REL\_DIR=`dirname $0`

cd $REL\_DIR

DIR=`pwd`

cd - > /dev/null

source /etc/profile

source ~/.bash\_profile

source $SHELL\_HOME/common/include.sh

source $SHELL\_HOME/common/date\_util.sh

source $SHELL\_HOME/common/db/config.sh

source $SHELL\_HOME/common/db/mysql/mysql\_util.sh

source $SCHED\_HOME/common/task\_util.sh

source $SCHED\_HOME/manager/manage\_util.sh

# 捕捉kill信号

trap 'warn "$0 is killed, pid: $$, script will exit soon";unset RUN\_MODE' TERM

# 清理任务管理器日志

function clean\_daemon\_log()

{

debug "Clean up the daemon log"

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_manager.log.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_manager.sql.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

}

# 拆分任务执行日志

# 每个月一次，成功后会生成一个空的flag文件

function split\_task\_log()

{

local flag\_file=$SCHED\_LOG\_DIR/task\_log\_split\_flag.$(date +%Y%m)

debug "Check if exists flag file: $flag\_file"

if [[ ! -f $flag\_file ]]; then

rm -f $SCHED\_LOG\_DIR/task\_log\_split\_flag.\*

# 清理历史任务日志，并生成flag文件

debug "Clean history task log from table: t\_task\_log and generate flag file: $flag\_file"

split\_log && touch $flag\_file

fi

}

# 清理任务池一段时间的历史任务

# 每天一次，成功后会生成一个空的flag文件

function clean\_task\_pool()

{

local flag\_file=$SCHED\_LOG\_DIR/task\_pool\_clean\_flag.$(date +%Y%m%d)

debug "Check if exists flag file: $flag\_file"

if [[ ! -f $flag\_file ]]; then

rm -f $SCHED\_LOG\_DIR/task\_pool\_clean\_flag.\*

# 清理历史任务实例，并生成flag文件

debug "Clean history task instance from table: t\_task\_pool and generate flag file: $flag\_file"

clean\_task\_instance && touch $flag\_file

fi

}

# 实例化任务

function init\_task()

{

debug "Get tasks and instantiate one by one"

get\_tasks | while read task\_id task\_cycle cycle\_value start\_time end\_time date\_serial priority max\_try\_times; do

debug "Begin instantiate task: (task\_id, task\_cycle, cycle\_value, start\_time, end\_time) ($task\_id, $task\_cycle, $cycle\_value, $start\_time, $end\_time)"

make\_task\_instance $task\_id $task\_cycle $cycle\_value $start\_time $end\_time | while read task\_id run\_time; do

debug "Insert task: (task\_id, run\_time, task\_state, priority, max\_try\_times) ($task\_id, $run\_time, $first\_cycle, $TASK\_STATE\_INITIAL, $priority, $max\_try\_times)"

insert\_task $task\_id $run\_time $TASK\_STATE\_INITIAL $priority $max\_try\_times

done

done

}

# 检查状态为“初始化”的任务的依赖关系

function check\_task\_deps()

{

debug "Get initial tasks and check one by one"

get\_initial\_tasks | while read task\_id run\_time task\_cycle first\_time date\_serial; do

debug "Begin check task: (task\_id, run\_time, task\_cycle, first\_time, date\_serial) ($task\_id, $run\_time, $task\_cycle, $first\_time, $date\_serial)"

task\_state=$(check\_dependence $task\_id $run\_time $task\_cycle $first\_time $date\_serial)

debug "Done check task: (task\_id, run\_time) ($task\_id, $run\_time) task\_state = $task\_state"

if [[ -z "$task\_state" ]]; then

debug "Update task: (task\_id, run\_time) ($task\_id, $run\_time) set task\_state = \$TASK\_STATE\_READY"

update\_task\_instance $task\_id $run\_time "task\_state = $TASK\_STATE\_READY" > /dev/null

fi

done

}

# 执行操作

function execute()

{

clean\_daemon\_log

log\_fn split\_task\_log

log\_fn clean\_task\_pool

log\_fn init\_task

log\_fn check\_task\_deps

}

# 滚动日志

function roll\_log()

{

local cur\_date=$(date +'%Y-%m-%d')

local prev\_date=$(date +'%Y-%m-%d' -d "$cur\_date 1 day ago")

if [[ -s $log\_file && ! -f $log\_file.$prev\_date ]]; then

sed "/${cur\_date}/Q" $log\_file > $log\_file.$prev\_date

sed -n "/${cur\_date}/,\$p" $log\_file > $log\_file.tmp

mv -f $log\_file.tmp $log\_file

fi

}

# 打印用法

function print\_usage()

{

echo "Usage: $0 [-l log level<0:debug/1:info/2:warn/3:error>] [-m run mode<once/loop>]"

}

function main()

{

info "Current working directory: $BASE\_DIR, invoke script: $0 $@"

while getopts ":l:m:" name; do

case "$name" in

l)

LOG\_LEVEL="$OPTARG";;

m)

RUN\_MODE="$OPTARG";;

?)

print\_usage

exit 1;;

esac

done

# 出错立即退出

set -e

# 创建日志文件目录

mkdir -p $SCHED\_LOG\_DIR

log\_file=$SCHED\_LOG\_DIR/task\_manager.log

if [[ "$RUN\_MODE" = "$RUN\_MODE\_LOOP" ]]; then

info "Script will execute periodically"

while [[ "$RUN\_MODE" = "$RUN\_MODE\_LOOP" ]]; do

execute >> $log\_file 2>&1

roll\_log

if [[ "$RUN\_MODE" != "$RUN\_MODE\_LOOP" ]]; then

break

fi

info "$0 sleep for $TASK\_CHECK\_INTERVAL"

sleep $TASK\_CHECK\_INTERVAL

info "$0 wake up"

done

else

info "Script will execute one time and exit"

execute

fi

}

main "$@"

task\_scheduler.sh

#!/bin/bash

# 任务调度器

# 1、调度任务周期为“时间间隔”的任务

# 2、调度任务状态为“就绪”或“失败”且尝试次数小于最大尝试次数的任务

BASE\_DIR=`pwd`

REL\_DIR=`dirname $0`

cd $REL\_DIR

DIR=`pwd`

cd - > /dev/null

source /etc/profile

source ~/.bash\_profile

source $SHELL\_HOME/common/include.sh

source $SHELL\_HOME/common/date\_util.sh

source $SHELL\_HOME/common/db/config.sh

source $SHELL\_HOME/common/db/mysql/mysql\_util.sh

source $SCHED\_HOME/common/task\_util.sh

source $SCHED\_HOME/scheduler/schedule\_util.sh

# 捕捉kill信号

trap 'warn "$0 is killed, pid: $$, script will exit soon";unset RUN\_MODE' TERM

# 清理日志文件

function clean\_log()

{

debug "Clean up the log"

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_scheduler.log.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_scheduler.sql.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_runner.log.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_runner.sql.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_proxy.log.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_proxy.sql.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_starter.log.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

find $SCHED\_LOG\_DIR -maxdepth 1 -type f -name "task\_starter.sql.\*" | xargs -r ls -c | sed "1,${LOG\_FILE\_KEEP\_NUM} d" | xargs -r rm -f

# 清理任务运行日志目录

if [[ -d $TASK\_LOG\_DIR ]]; then

ls -c $TASK\_LOG\_DIR | sed "1,${TASK\_LOG\_KEEP\_DAY} d" | xargs -r -I {} rm -rf $TASK\_LOG\_DIR/{}

fi

if [[ -d $TASK\_DATA\_DIR ]]; then

ls -c $TASK\_DATA\_DIR | sed "1,${TASK\_LOG\_KEEP\_DAY} d" | xargs -r -I {} rm -rf $TASK\_DATA\_DIR/{}

fi

}

# kill超时任务

function kill\_task()

{

get\_task\_timeout | while read task\_id run\_time; do

pname="$SCHED\_HOME/task\_runner.sh $task\_id $run\_time"

pid=`ps -ef | grep "$pname" | grep -v grep | awk '{print $2}'`

if [[ -n "$pid" ]]; then

info "Kill task $pid $pname"

kill $pid

else

info "Kill task $task\_id $run\_time"

update\_task\_instance $task\_id $run\_time "task\_state = $TASK\_STATE\_KILLED"

fi

done

}

# 调度周期为“时间间隔”的任务

# 每次只获取一个任务

function schedule\_interval\_task()

{

get\_task\_interval | while read task\_id run\_time cycle\_value timeout; do

# 启动任务代理

cur\_date=$(date +'%Y-%m-%d')

info "Invoke task proxy: nohup $SCHED\_HOME/task\_proxy.sh $task\_id $run\_time $cycle\_value $timeout >> $SCHED\_LOG\_DIR/task\_proxy.log.${cur\_date} 2>&1 &"

nohup $SCHED\_HOME/task\_proxy.sh $task\_id $run\_time $cycle\_value $timeout >> $SCHED\_LOG\_DIR/task\_proxy.log.${cur\_date} 2>&1 &

done

}

# 调度状态为“就绪”的任务

# 根据服务器的最大并发数和当前并发数

function schedule\_ready\_task()

{

get\_task\_ready | while read task\_id run\_time last\_try; do

# 启动任务

cur\_date=$(date +'%Y-%m-%d')

info "Invoke task runner: nohup $SCHED\_HOME/task\_runner.sh $task\_id $run\_time $last\_try >> $SCHED\_LOG\_DIR/task\_runner.log.${cur\_date} 2>&1 &"

nohup $SCHED\_HOME/task\_runner.sh $task\_id $run\_time $last\_try >> $SCHED\_LOG\_DIR/task\_runner.log.${cur\_date} 2>&1 &

done

}

# 执行操作

function execute()

{

# 发送心跳

send\_heartbeat

# 清理日志

clean\_log

# 杀死超时任务

kill\_task

# 调度时间间隔任务

log\_fn schedule\_interval\_task

# 调度就绪任务

log\_fn schedule\_ready\_task

}

# 滚动日志

function roll\_log()

{

local cur\_date=$(date +'%Y-%m-%d')

local prev\_date=$(date +'%Y-%m-%d' -d "$cur\_date 1 day ago")

if [[ -s $log\_file && ! -f $log\_file.$prev\_date ]]; then

sed "/${cur\_date}/Q" $log\_file > $log\_file.$prev\_date

sed -n "/${cur\_date}/,\$p" $log\_file > $log\_file.tmp

mv -f $log\_file.tmp $log\_file

fi

}

# 打印用法

function print\_usage()

{

echo "Usage: $0 [-l log level<0:debug/1:info/2:warn/3:error>] [-m run mode<once/loop>]"

}

function main()

{

info "Current working directory: $BASE\_DIR, invoke script: $0 $@"

while getopts ":l:m:" name; do

case "$name" in

l)

LOG\_LEVEL="$OPTARG";;

m)

RUN\_MODE="$OPTARG";;

?)

print\_usage

exit 1;;

esac

done

# 出错立即退出

set -e

# 创建日志文件目录

mkdir -p $SCHED\_LOG\_DIR

log\_file=$SCHED\_LOG\_DIR/task\_scheduler.log

if [[ "$RUN\_MODE" = "$RUN\_MODE\_LOOP" ]]; then

info "Script will execute periodically"

while [[ "$RUN\_MODE" = "$RUN\_MODE\_LOOP" ]]; do

execute >> $log\_file 2>&1

roll\_log

if [[ "$RUN\_MODE" != "$RUN\_MODE\_LOOP" ]]; then

break

fi

info "$0 sleep for $TASK\_SCHEDULE\_INTERVAL"

sleep $TASK\_SCHEDULE\_INTERVAL

info "$0 wake up"

done

else

info "Script will execute one time and exit"

execute

fi

}

main "$@"

task\_runner.sh

#!/bin/bash

#

# 任务运行器

# 1、调用任务执行器执行任务

# 2、获取任务执行状态并更新

# 3、发送电子邮件、手机短信告警

#

# 注意:

# 1、任务执行器必须放在$SCHED\_HOME/plugins目录下

BASE\_DIR=`pwd`

REL\_DIR=`dirname $0`

cd $REL\_DIR

DIR=`pwd`

cd - > /dev/null

source /etc/profile

source ~/.bash\_profile

source $SHELL\_HOME/common/include.sh

source $SHELL\_HOME/common/date\_util.sh

source $SHELL\_HOME/common/db/config.sh

source $SHELL\_HOME/common/db/mysql/mysql\_util.sh

source $SCHED\_HOME/common/task\_util.sh

source $SCHED\_HOME/scheduler/schedule\_util.sh

# 捕捉kill信号

trap 'kill\_task' TERM

# 发送告警

function send\_alarm()

{

# 发送电子邮件

if [[ -n "$sub\_emails" ]]; then

info "Send email notification to: $sub\_emails"

echo "$mail\_content" | $SHELL\_HOME/common/mail\_sender.py "$sub\_emails" "$mail\_subject"

fi

# 发送手机短信

if [[ -n "$sub\_mobiles" ]]; then

info "Send sms notification to: $sub\_mobiles"

echo "$sms\_content" | $SHELL\_HOME/common/sms\_sender.sh "$sub\_mobiles"

fi

}

# 任务成功

function succeed\_task()

{

# 告警、成功告警

if [[ $is\_alarm -eq 1 || $is\_alarm -eq 2 ]]; then

# 电子邮件

if [[ -s $log\_path/mail.tmp ]]; then

mail\_subject=`head -n 1 $log\_path/mail.tmp`

mail\_content=`sed '1 d' $log\_path/mail.tmp`

else

mail\_subject="任务: ($task\_id, $run\_time) 执行成功"

mail\_content=`awk '{printf("%s<br/>",$0)}' $log\_path/task.info`

fi

# 手机短信

if [[ -s $log\_path/sms.tmp ]]; then

sms\_content=`cat $log\_path/sms.tmp`

else

sms\_content="任务: ($task\_id, $run\_time) 执行成功"

fi

# 发送告警

send\_alarm

fi

}

# 任务失败

function fail\_task()

{

if [[ $last\_try -eq 1 ]]; then

# 告警、失败告警

if [[ $is\_alarm -eq 1 || $is\_alarm -eq 3 ]]; then

# 电子邮件

if [[ -s $log\_path/mail.tmp ]]; then

mail\_subject=`head -n 1 $log\_path/mail.tmp`

mail\_content=`sed '1 d' $log\_path/mail.tmp`

else

mail\_subject="任务: ($task\_id, $run\_time) 执行失败"

mail\_content=`awk '{printf("%s<br/>",$0)}' $log\_path/task.info $log\_path/task.error`

fi

# 手机短信

if [[ -s $log\_path/sms.tmp ]]; then

sms\_content=`cat $log\_path/sms.tmp`

elif [[ -s $log\_path/task.error ]]; then

sms\_content="任务: ($task\_id, $run\_time) 执行失败, "`tail -n 1 $log\_path/task.error | sed 's/^.\* \[ \(.\*\) \]$/\1/'`

else

sms\_content="$mail\_subject"

fi

# 发送告警

send\_alarm

fi

else

debug "The try does not reach the maximum"

fi

}

# 更新任务

function update\_task()

{

info "Update task: (task\_id, run\_time) ($task\_id, $run\_time) set task\_state = $task\_state"

result=$(update\_task\_instance $task\_id $run\_time "task\_state = $task\_state, end\_time = NOW()")

counter=1

while [[ $result -ne 1 && $counter -lt 10 ]]; do

sleep 30

counter=$((counter + 1))

result=$(update\_task\_instance $task\_id $run\_time "task\_state = $task\_state, end\_time = NOW()")

done

if [[ $result -eq 1 ]]; then

info "Update task: (task\_id, run\_time) ($task\_id, $run\_time) successfully"

else

error "Update task state failed (task\_id, run\_time) ($task\_id, $run\_time)"

fi

}

# 杀死任务

function kill\_task()

{

log\_task $LOG\_LEVEL\_INFO "Task: (task\_id, run\_time) ($task\_id, $run\_time) is killed"

task\_state=$TASK\_STATE\_KILLED

fail\_task

update\_task

}

function main()

{

info "Current working directory: $BASE\_DIR, invoke script: $0 $@"

# 参数判断

if [[ $# -lt 2 ]]; then

error "Invalid arguments: $@, usage: $0 <task id> <run time>"

exit 1

fi

task\_id="$1"

run\_time="$2"

last\_try="${3:-0}"

# 判断任务是否存在且正常

info "Check if exists valid task: $task\_id"

if [[ $(exists\_task $task\_id) -eq 0 ]]; then

error "Can not find valid task by id: $task\_id"

exit 1

fi

# 任务运行器

info "Get task executor"

task\_executor=$(get\_task\_executor $task\_id)

if [[ -z "$task\_executor" ]]; then

error "Can not find task executor for task: $task\_id"

exit 1

fi

# 告警

debug "Get alarm switch"

is\_alarm=$(get\_prop\_value $task\_id is\_alarm)

is\_alarm=${is\_alarm/NULL/0}

if [[ $is\_alarm -gt 0 ]]; then

debug "Alarm switch is on"

# 告警方式

alarm\_way=$(get\_prop\_value $task\_id alarm\_way)

if [[ $alarm\_way -eq 0 ]]; then

sub\_emails=$(get\_prop\_value $task\_id sub\_emails)

elif [[ $alarm\_way -eq 1 ]]; then

sub\_mobiles=$(get\_prop\_value $task\_id sub\_mobiles)

elif [[ $alarm\_way -eq 2 ]]; then

sub\_emails=$(get\_prop\_value $task\_id sub\_emails)

sub\_mobiles=$(get\_prop\_value $task\_id sub\_mobiles)

else

warn "Unsupported alarm way: $alarm\_way"

fi

else

debug "Alarm switch is off"

fi

# 更新任务状态为“正在运行”

result=$(update\_task\_instance $task\_id $run\_time "task\_state = $TASK\_STATE\_RUNNING, start\_time = NOW()")

if [[ $result -ne 1 ]]; then

error "Update task state failed (task\_id, run\_time) ($task\_id, $run\_time)"

exit 1

fi

# 流水号

seq\_no=$(date +'%s')

# 创建任务日志目录

cur\_date=$(date +%Y%m%d)

log\_path=$TASK\_LOG\_DIR/$cur\_date/${task\_id}-${run\_time}

mkdir -p $log\_path

# 启动任务

info "Invoke task executor: $SCHED\_HOME/plugins/$task\_executor $task\_id $run\_time $seq\_no > $log\_path/task.info 2> $log\_path/task.error"

nohup $SCHED\_HOME/plugins/$task\_executor $task\_id $run\_time $seq\_no > $log\_path/task.info 2> $log\_path/task.error

# 判断任务执行结果

if [[ $? -eq 0 ]]; then

log\_task $LOG\_LEVEL\_INFO "Task: (task\_id, run\_time) ($task\_id, $run\_time) done successfully"

task\_state=$TASK\_STATE\_SUCCESS

succeed\_task

else

log\_task $LOG\_LEVEL\_ERROR `sed 's/^.\* \[ \(.\*\) \]$/\1/' $log\_path/task.error | awk '{printf("%s\\\n",$0)}'`

task\_state=$TASK\_STATE\_FAILED

fail\_task

fi

# 更新任务，状态、结束时间

update\_task

}

main "$@"

exec\_mdx.sh

#!/bin/bash

#

# 模拟saiku执行mdx

# Dependencies: yum dos2unix

BASE\_DIR=`pwd`

REL\_DIR=`dirname $0`

cd $REL\_DIR

DIR=`pwd`

cd - > /dev/null

source /etc/profile

source ~/.bash\_profile

source $SHELL\_HOME/common/db/db\_util.sh

source $SCHED\_HOME/plugins/db\_util.sh

# 获取源数据库连接

function get\_src\_db()

{

if [[ -z "$src\_db\_id" ]]; then

error "Empty source database id"

exit 1

fi

debug "Get source database by id: $src\_db\_id"

src\_db=($(get\_db $src\_db\_id))

if [[ -z "${src\_db[@]}" ]]; then

error "Can not find source database by id: $src\_db\_id"

exit 1

fi

debug "Got source database: ${src\_db[@]}"

# saiku连接信息

saiku\_url="http://${src\_db[1]}:${src\_db[2]}/$saiku\_path"

saiku\_user="${src\_db[3]}"

saiku\_passwd="${src\_db[4]}"

conn\_name="${src\_db[5]}"

saiku\_charset="${src\_db[6]}"

timeout=60

}

# 替换mdx语句中的变量

function replace\_var()

{

sed "s/#the\_day#/${the\_day}/g;s/#prev\_day#/${prev\_day}/g;s/#next\_day#/${next\_day}/g;s/#run\_time#/${run\_time}/g;s/#prev\_day1#/${prev\_day1}/g"

}

# 获取mdx语句

# 替换mdx语句中的变量

function get\_mdx()

{

log\_task $LOG\_LEVEL\_INFO "Get mdx and replace variables"

echo -e `get\_prop\_value $task\_id src\_mdx` | replace\_var > $log\_path/src\_mdx.tmp

}

# 登录获取cookie

function login\_saiku()

{

login\_url=$saiku\_url/session

curl -s --connect-timeout $timeout -c $log\_path/cookie.tmp -d "username=$saiku\_user&password=$saiku\_passwd" $login\_url

# 判断cookie是否获取成功

if [[ ! -s $log\_path/cookie.tmp ]]; then

error "Log into saiku failed"

exit 1

fi

}

# 刷新Cube

function refresh\_cube()

{

if [[ $saiku\_version =~ 3 ]]; then

# saiku3

refresh\_url=$saiku\_url/admin/datasources/${conn\_name}/refresh?\_=`date +%s`

else

refresh\_url=$saiku\_url/$saiku\_user/discover/$conn\_name/refresh?\_=`date +%s`

fi

curl -s --connect-timeout $timeout -b $log\_path/cookie.tmp $refresh\_url > /dev/null

}

# 初始化查询

function init\_query()

{

# 生成查询名称

query\_name=`uuidgen`

# 创建查询请求地址

if [[ ! $saiku\_version =~ 3 ]]; then

init\_query\_url=$saiku\_url/$saiku\_user/query/$query\_name

# 创建查询

curl -s --connect-timeout $timeout -b $log\_path/cookie.tmp -o $log\_path/query\_result.tmp -X POST -d "connection=$conn\_name&catalog=$catalog\_name&schema=$schema\_name&cube=$cube\_name" $init\_query\_url

# 判断查询是否创建成功

if [[ ! -s $log\_path/query\_result.tmp ]]; then

error "Create query failed, please check the parameters: {connection=$conn\_name&catalog=$catalog\_name&schema=$schema\_name&cube=$cube\_name}"

exit 1

fi

else

# saiku3

init\_query\_url=$saiku\_url/api/query/$query\_name

fi

}

# 构建请求参数

function build\_param()

{

dos2unix -q $log\_path/src\_mdx.tmp

if [[ $saiku\_version =~ 3 ]]; then

echo "{"

echo "\"cube\" : {"

echo "\"name\" : \"${cube\_name}\","

echo "\"connection\" : \"${conn\_name}\","

echo "\"catalog\" : \"${catalog\_name}\","

echo "\"schema\" : \"${schema\_name}\""

echo "},"

echo "\"mdx\" : \"`cat $log\_path/src\_mdx.tmp`\","

echo "\"name\" : \"${query\_name}\","

echo "\"type\" : \"MDX\""

echo "}"

else

echo "mdx=`cat $log\_path/src\_mdx.tmp`"

fi | tr '\n' ' ' | tr '\t' ' ' | tee $log\_path/post\_data

}

# 错误检查

function check\_error()

{

egrep '\["error"\]' | awk -F '\t' '{

if($2 != "null"){

gsub("\\\\","",$2)

print substr($2,2,length($2)-2)

}

}'

}

# 过滤数据

function filter\_data(){

egrep '\["cellset",[0-9]+,[0-9]+,"value"\]' | awk -F '\t' '{

value=substr($2,2,length($2)-2)

if(value~"^[0-9,.]+$"){

gsub(",","",value)

}

split($1,aa,",")

if(aa[3]==0 && NR>1){

printf("\n")

col=0

row=row+1

}

if(col>0){

printf("\t")

}

printf(value)

col=col+1

}'

}

function execute()

{

# 获取待执行mdx语句

log\_task $LOG\_LEVEL\_INFO "Get mdx to be executed"

get\_mdx

# 获取源数据库连接信息

get\_src\_db

# 登录获取cookie

log\_task $LOG\_LEVEL\_INFO "Log into saiku and get cookie"

login\_saiku

# 是否刷新

if [[ $is\_refresh -eq 1 ]]; then

log\_task $LOG\_LEVEL\_INFO "Refresh cube"

refresh\_cube

fi

# 初始化查询

log\_task $LOG\_LEVEL\_INFO "Create a new query"

init\_query

# 查询请求地址

if [[ $saiku\_version =~ 3 ]]; then

# saiku3

query\_url=$saiku\_url/api/query/execute

else

query\_url=$saiku\_url/$saiku\_user/query/$query\_name/result/flat

fi

# 执行mdx语句

if [[ -n "$tar\_db\_id" ]]; then

# 获取目标数据库连接信息

get\_tar\_db

tar\_table=$tar\_table\_name

# 任务重做

if [[ $redo\_flag -eq 1 ]]; then

# 必须要有统计日期字段

if [[ -z "$stat\_column" ]]; then

error "Can not find statistical column, program do not know how to redo the task"

exit 1

fi

execute\_tar "DELETE FROM $tar\_table WHERE $stat\_column = DATE($prev\_day)" "-vvv" > $log\_path/redo.log

fi

# 构建请求参数

info "Build query parameters for post request"

post\_data=`build\_param`

# 设置请求头

if [[ $saiku\_version =~ 3 ]]; then

http\_header="-H Content-Type:application/json -H Accept:application/json"

fi

# 执行查询，下载数据

log\_task $LOG\_LEVEL\_INFO "Execute mdx query and download data to file: $data\_path/result.json"

http\_code=$(curl -s -w %{http\_code} --connect-timeout $timeout -b $log\_path/cookie.tmp -o $data\_path/result.json $http\_header -X POST -d "$post\_data" $query\_url)

if [[ ! "$http\_code" =~ ^200|30[0-9]$ ]]; then

error "Query failed, saiku server return http code: $http\_code"

exit 1

fi

# json数据转tab

log\_task $LOG\_LEVEL\_INFO "Convert json data to tab separated data"

echo -e "$data\_path/result.json\n" | awk -f $SHELL\_HOME/common/JSON.awk > $data\_path/result.tmp

# 执行mdx结果错误判断

local error\_msg=`check\_error < $data\_path/result.tmp`

if [[ -n "$error\_msg" ]]; then

error "$error\_msg"

exit 1

fi

debug "Filter data"

filter\_data < $data\_path/result.tmp > $data\_path/result.txt

# 判断文件内容是否为空

if [[ ! -s $data\_path/result.txt ]]; then

error "Can not fetch any data"

exit 1

fi

# 获取表头行数

header\_lines=`sed '/\t"DATA\_CELL"$/q' $data\_path/result.tmp | tail -n 1 | cut -d , -f 2`

# 预装载数据

debug "Preload data"

preload

# 导入目标表

local sql="LOAD DATA LOCAL INFILE '$data\_path/result.txt' $load\_mode INTO TABLE $tar\_table IGNORE $header\_lines LINES"

# 指定字段

if [[ -n "$tar\_columns" ]]; then

sql="$sql ( $tar\_columns )"

fi

# 统计日期

if [[ -n "$stat\_column" ]]; then

sql="$sql SET $stat\_column = DATE($prev\_day)"

fi

# 设置字段

if [[ -n "$tar\_set\_columns" ]]; then

if [[ -n "$stat\_column" ]]; then

sql="$sql, $tar\_set\_columns"

else

sql="$sql SET $tar\_set\_columns"

fi

fi

log\_task $LOG\_LEVEL\_INFO "Load data to target table: $tar\_table"

execute\_tar "$sql" "-vvv" > $log\_path/load\_data.log

else

curl -s --connect-timeout $timeout -b $log\_path/cookie.tmp -d "mdx=`cat $log\_path/src\_mdx.tmp`" $query\_url

fi

}

source $SCHED\_HOME/plugins/task\_executor.sh

task\_executor.sh

# 任务执行器

#

# 功能

# 1 初始化环境

# 2 获取任务配置信息

# 3 执行任务

source $SHELL\_HOME/common/include.sh

source $SHELL\_HOME/common/date\_util.sh

source $SHELL\_HOME/common/db/mysql/mysql\_util.sh

source $SCHED\_HOME/common/task\_util.sh

# 初始化文件目录

function init\_dir()

{

# 任务日志文件目录

log\_path=$TASK\_LOG\_DIR/$cur\_date/${task\_id}-${run\_time}

if [[ ! -d $log\_path ]]; then

log\_path=$TASK\_LOG\_DIR/$(date +'%Y%m%d' -d "1 day ago")/${task\_id}-${run\_time}

fi

# 创建任务数据文件目录

data\_path=`echo $log\_path | sed "s@${TASK\_LOG\_DIR}@${TASK\_DATA\_DIR}@"`

debug "Create task data file directory: $data\_path"

mkdir -p $data\_path

# sql日志

sql\_log\_file=$log\_path/task\_executor.sql

}

# 获取任务配置信息

function get\_task\_config()

{

# 获取任务实例

debug "Get task instance by: (task\_id, run\_time) ($task\_id, $run\_time)"

task\_instance=($(get\_task\_instance $task\_id $run\_time))

debug "Got task instance: (task\_cycle, is\_first, redo\_flag) (${task\_instance[@]})"

# 任务周期

task\_cycle=${task\_instance[0]}

# 任务是否第一次运行

is\_first=${task\_instance[1]}

# 任务重做标记

redo\_flag=${task\_instance[2]}

# 获取任务上次运行周期

the\_time=$(format\_time $run\_time)

local last\_cycle=$(get\_next\_cycle $run\_time $task\_cycle ago)

prev\_time=$(format\_time $last\_cycle)

# 获取任务扩展属性

debug "Get task extended attributes by task id: $task\_id, exclude: src\_sql, src\_mdx, src\_urls"

get\_task\_ext $task\_id "'src\_sql', 'src\_mdx', 'src\_urls'" > $log\_path/task.ext

# 设置任务扩展属性

debug "Set task extended attributes by source file: $log\_path/task.ext"

source $log\_path/task.ext

# 加载文件

if [[ -n "$source\_file" ]]; then

debug "Source file: $source\_file"

source $source\_file

fi

}

function main()

{

# 参数判断

if [[ $# -lt 2 ]]; then

error "Invalid arguments: $@, usage: $0 <task id> <run time> [task serial number]"

exit 1

fi

task\_id="$1"

run\_time="$2"

seq\_no="${3:-$(date +%s)}"

# 出错立即退出

set -e

set -o pipefail

info "Current working directory: $BASE\_DIR, invoke script: $0 $@"

# 初始化时间变量

init\_date ${run\_time:0:8}

# 初始化目录

init\_dir

# 获取任务配置信息

log\_task $LOG\_LEVEL\_INFO "Get task configuration information"

get\_task\_config

# 执行任务

log\_task $LOG\_LEVEL\_INFO "Execute task"

execute

}

main "$@"

mail\_sender.py

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

# 发送邮件

import sys

import os

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.application import MIMEApplication

# 外网

smtp\_host = "smtp.9zhitx.com"

# 内网

smtp\_host = "10.10.10.7"

smtp\_port = 25

mail\_user = "zhangchao@9zhitx.com"

mail\_passwd = "aq01230100245"

timeout = 60

def main(argv):

to\_list = sys.argv[1]

subject = sys.argv[2]

msg = MIMEMultipart()

msg['From'] = mail\_user

msg['To'] = to\_list

msg['Subject'] = subject

# 正文

content = ""

if (len(sys.argv) > 3 and len(sys.argv[3].strip()) > 0):

content = sys.argv[3]

else:

for line in sys.stdin:

content += line

msg\_text = MIMEText(content, \_subtype='html', \_charset='utf-8')

msg.attach(msg\_text)

# 附件

if (len(sys.argv) > 4 and len(sys.argv[4].strip()) > 0):

attachments = sys.argv[4].split(",")

for attachment in attachments:

msg\_file = MIMEApplication(open(attachment, 'rb').read())

msg\_file.add\_header('Content-Disposition', 'attachment', filename=os.path.basename(attachment))

msg.attach(msg\_file)

server = None

try:

server = smtplib.SMTP(smtp\_host, smtp\_port, timeout)

server.login(mail\_user, mail\_passwd)

server.sendmail(mail\_user, to\_list.split(";"), msg.as\_string())

print "邮件发送成功"

sys.exit(0)

except Exception, e:

sys.stderr.write("邮件发送失败: " + str(e) + "\n")

sys.exit(1)

finally:

if server: server.close()

if \_\_name\_\_ == "\_\_main\_\_":

if (len(sys.argv) < 3):

sys.stderr.write("Usage: " + sys.argv[0] + " <to list> <mail subject> <mail content> [attachment file]\n")

sys.exit(1)

main(sys.argv)

sms\_sender.sh

# 发送短信

# 用户名

ACCOUNT=jizitinxa1

# 密码

PASSWORD=1ad521233ssfsf539

# 短信网关

SMS\_URL=http://sms.chanzor.com:8001/sms.aspx

# 超时时间

TIMEOUT=5

# 短信内容最大长度

MAX\_LENGTH=280

# 临时文件目录

TMP\_DIR=/tmp/sms\_$USER

function main()

{

if [[ $# -lt 1 ]]; then

echo "Usage: $0 <mobile phone numbers> <message content>" >&2

exit 1

fi

phone\_nums="$1"

if [[ $# -gt 1 ]]; then

msg\_content="$2"

else

msg\_content=`cat`

fi

# 流水号

no=$(date +'%Y%m%d%H%M%S%N')

# 创建目录

mkdir -p $TMP\_DIR/$no

# 删除一段时间的历史文件（10天前）

find $TMP\_DIR -ctime +10 -type d | xargs -r rm -rf

# 超出单条短信内容最大长度，则拆分成多条

echo -e "$msg\_content" | awk '{

total=total + length($0)

if(total > max\_length){

if(msg > ""){

print msg > "'$TMP\_DIR/$no/message-'"++j

total=length($0)

msg=$0

}else{

size=int(length($0) / max\_length)

mod=length($0) % max\_length

if (mod > 0) size = size + 1

for(i=0;i<size;i++){

print substr($0,i \* max\_length + 1,max\_length) > "'$TMP\_DIR/$no/message-'"++j

}

}

}else{

if(msg > ""){

msg=msg"\n"$0

}else{

msg=$0

}

}

} END {

size=int(length(msg) / max\_length)

mod=length(msg) % max\_length

if (mod > 0) size = size + 1

for(i=0;i<size;i++){

print substr(msg,i \* max\_length + 1,max\_length) > "'$TMP\_DIR/$no/message-'"++j

}

}' max\_length=$MAX\_LENGTH

# 发送短信

ls -c $TMP\_DIR/$no/message-\* | while read file\_name; do

# 短信内容

post\_data="content=九指运维短信报警：`cat $file\_name`【牙牙关注】"

curl -s -w "\n%{http\_code} %{time\_connect} %{time\_starttransfer} %{time\_total}\n" --connect-timeout $TIMEOUT --data-urlencode "$post\_data" "${SMS\_URL}?action=send&account=${ACCOUNT}&password=${PASSWORD}&mobile=${phone\_nums}" >> $TMP\_DIR/$no/sms\_log 2>&1

done

}

main "$@"

sdk.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Schema name=*"%{sdk}"* metamodelVersion=*"4.0"* xmlns=*"http://mondrian.pentaho.com/schema"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://mondrian.pentaho.com/schema http://mondrian.pentaho.com/schema/mondrian.xsd"*>

<!-- 物理模型 -->

<PhysicalSchema>

<Query alias=*"v\_date"* keyColumn=*"id"*>

<ExpressionView>

<SQL dialect=*"mysql"*><![CDATA[SELECT \* FROM dim\_date WHERE id >= 20151219 AND id <= CURDATE()]]></SQL>

</ExpressionView>

</Query>

<InlineTable alias=*"v\_yorn"*>

<ColumnDefs>

<ColumnDef name=*"id"* type=*"Integer"* />

<ColumnDef name=*"name"* type=*"String"* />

</ColumnDefs>

<Key>

<Column name=*"id"* />

</Key>

<Rows>

<Row>

<Value column=*"id"*>0</Value>

<Value column=*"name"*>N</Value>

</Row>

<Row>

<Value column=*"id"*>1</Value>

<Value column=*"name"*>Y</Value>

</Row>

</Rows>

</InlineTable>

<InlineTable alias=*"v\_logtype"*>

<ColumnDefs>

<ColumnDef name=*"id"* type=*"Integer"* />

<ColumnDef name=*"name"* type=*"String"* />

</ColumnDefs>

<Key>

<Column name=*"id"* />

</Key>

<Rows>

<Row>

<Value column=*"id"*>1</Value>

<Value column=*"name"*>主动</Value>

</Row>

<Row>

<Value column=*"id"*>2</Value>

<Value column=*"name"*>定时</Value>

</Row>

</Rows>

</InlineTable>

<Table name=*"dim\_cohort"* keyColumn=*"id"* />

<Table name=*"dim\_channel"* keyColumn=*"id"* />

<Table name=*"dim\_product"* keyColumn=*"id"* />

<Table name=*"dim\_version"* keyColumn=*"id"* />

<Table name=*"dim\_isp"* keyColumn=*"id"* />

<Table name=*"dim\_network"* keyColumn=*"name"* />

<Table name=*"dim\_apppath"* keyColumn=*"id"* />

<Table name=*"dim\_device"* keyColumn=*"uuid"* />

<Link source=*"dim\_isp"* target=*"dim\_device"* foreignKeyColumn=*"isp\_code"* />

<Link source=*"dim\_network"* target=*"dim\_device"* foreignKeyColumn=*"network"* />

<Link source=*"v\_yorn"* target=*"dim\_device"* foreignKeyColumn=*"have\_vpn"* />

<Table name=*"fact\_client"*>

<ColumnDefs>

<CalculatedColumnDef name=*"is\_upgraded"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(version != init\_version, 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"fact\_active"* />

<Table name=*"agg\_lc\_01\_client"* />

<Table name=*"agg\_l\_02\_client"* />

<Table name=*"agg\_l\_03\_client"* />

<Table name=*"agg\_l\_04\_client"* />

<Table name=*"agg\_l\_05\_client"* />

<Table name=*"agg\_l\_06\_client"* />

<Table name=*"agg\_l\_07\_client"* />

</PhysicalSchema>

<!-- 逻辑模型 -->

<!-- 共享维度 -->

<!-- 日期维度 -->

<Dimension name=*"Date"* table=*"v\_date"* key=*"Date"* type=*"TIME"* caption=*"%{date}"*>

<Attributes>

<Attribute name=*"Year"* keyColumn=*"the\_year"* levelType=*"TimeYears"* hasHierarchy=*"false"* />

<Attribute name=*"HalfYear"* levelType=*"TimeHalfYear"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"half\_year"* />

</Key>

<Name>

<Column name=*"half\_year"* />

</Name>

</Attribute>

<Attribute name=*"Quarter"* levelType=*"TimeQuarters"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"quarter\_num"* />

</Key>

<Name>

<Column name=*"en\_quarter"* />

</Name>

</Attribute>

<Attribute name=*"Month"* levelType=*"TimeMonths"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"month\_num"* />

</Key>

<Name>

<Column name=*"month\_num"* />

</Name>

</Attribute>

<Attribute name=*"Week"* levelType=*"TimeWeeks"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"week\_num"* />

</Key>

<Name>

<Column name=*"week\_num"* />

</Name>

</Attribute>

<Attribute name=*"Day"* keyColumn=*"id"* nameColumn=*"day\_num"* levelType=*"TimeDays"* hasHierarchy=*"false"* />

<Attribute name=*"Date"* keyColumn=*"id"* nameColumn=*"the\_day"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Monthly"*>

<Level attribute=*"Year"* caption=*"%{year}"* />

<Level attribute=*"HalfYear"* caption=*"%{half\_year}"* />

<Level attribute=*"Quarter"* caption=*"%{quarter}"* />

<Level attribute=*"Month"* caption=*"%{month}"* />

<Level attribute=*"Day"* caption=*"%{day}"* />

</Hierarchy>

<Hierarchy name=*"Daily"*>

<Level attribute=*"Date"* caption=*"%{date}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 设备维度 -->

<Dimension name=*"Device"* table=*"dim\_device"* key=*"UUID"* caption=*"%{device}"*>

<Attributes>

<Attribute name=*"Platform"* keyColumn=*"platform"* hasHierarchy=*"false"* />

<Attribute name=*"OSVersion"* keyColumn=*"os\_version"* hasHierarchy=*"false"* />

<Attribute name=*"Network"* table=*"dim\_network"* keyColumn=*"name"* orderByColumn=*"priority"* hierarchyCaption=*"%{network}"* caption=*"%{type}"* />

<Attribute name=*"Model"* keyColumn=*"model\_name"* caption=*"%{model}"* />

<Attribute name=*"Resolution"* keyColumn=*"resolution"* caption=*"%{resolution}"* />

<Attribute name=*"Country"* keyColumn=*"country"* hasHierarchy=*"false"* />

<Attribute name=*"Province"* keyColumn=*"region"* hasHierarchy=*"false"* />

<Attribute name=*"City"* keyColumn=*"city"* hasHierarchy=*"false"* />

<Attribute name=*"ISPName"* table=*"dim\_isp"* keyColumn=*"isp\_name"* hasHierarchy=*"false"* />

<Attribute name=*"ISPCode"* table=*"dim\_isp"* keyColumn=*"id"* hasHierarchy=*"false"* />

<Attribute name=*"VPN"* table=*"v\_yorn"* keyColumn=*"id"* nameColumn=*"name"* hierarchyCaption=*"%{used\_vpn}"* caption=*"%{yesorno}"* />

<Attribute name=*"UUID"* keyColumn=*"uuid"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Platform"* caption=*"%{platform}"*>

<Level attribute=*"Platform"* caption=*"%{name}"* />

<Level attribute=*"OSVersion"* caption=*"%{version}"* />

</Hierarchy>

<Hierarchy name=*"Region"* caption=*"%{region}"*>

<Level attribute=*"Country"* caption=*"%{country}"* />

<Level attribute=*"Province"* caption=*"%{province}"* />

<Level attribute=*"City"* caption=*"%{city}"* />

</Hierarchy>

<Hierarchy name=*"ISP"* caption=*"%{isp}"*>

<Level attribute=*"ISPName"* caption=*"%{name}"* />

<Level attribute=*"ISPCode"* caption=*"%{code}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 日期间隔维度 -->

<Dimension name=*"DateDiff"* table=*"dim\_cohort"* key=*"Day"* caption=*"%{date\_diff}"*>

<Attributes>

<Attribute name=*"Day"* keyColumn=*"id"* caption=*"%{day\_diff}"* />

</Attributes>

</Dimension>

<!-- App版本维度 -->

<Dimension name=*"Version"* table=*"dim\_version"* key=*"Num"* caption=*"%{version}"*>

<Attributes>

<Attribute name=*"Num"* keyColumn=*"id"* caption=*"%{version\_num}"* />

</Attributes>

</Dimension>

<!-- 渠道维度 -->

<Dimension name=*"Channel"* table=*"dim\_channel"* key=*"Id"* caption=*"%{channel}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"channel\_name"* hasHierarchy=*"false"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Channel"*>

<Level attribute=*"Name"* caption=*"%{name}"* />

<Level attribute=*"Id"* caption=*"%{code}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 产品维度 -->

<Dimension name=*"Product"* table=*"dim\_product"* key=*"Id"* caption=*"%{product}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* hasHierarchy=*"false"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Product"*>

<Level attribute=*"Name"* caption=*"%{name}"* />

<Level attribute=*"Id"* caption=*"%{code}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 是否维度 -->

<Dimension name=*"Yorn"* table=*"v\_yorn"* key=*"Id"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{yesorno}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 网络类型维度 -->

<Dimension name=*"Network"* table=*"dim\_network"* key=*"Name"* caption=*"%{network}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"name"* orderByColumn=*"priority"* caption=*"%{type}"* />

</Attributes>

</Dimension>

<!-- 次数维度 -->

<Dimension name=*"Times"* table=*"dim\_cohort"* key=*"Id"*>

<Attributes>

<Attribute name=*"Range\_1"* keyColumn=*"times\_1"* nameColumn=*"times\_1\_desc"* caption=*"%{times\_range}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 新增用户Cube -->

<Cube name=*"Client"* caption=*"%{new\_user}"*>

<Dimensions>

<Dimension source=*"Date"* caption=*"%{create\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Version"* caption=*"%{current\_version}"* />

<Dimension source=*"Product"* />

<Dimension name=*"Upgrade"* source=*"Yorn"* caption=*"%{is\_upgraded}"* />

<Dimension name=*"Silent"* source=*"Yorn"* caption=*"%{is\_silent}"* />

<!-- App安装路径维度 -->

<Dimension name=*"AppPath"* table=*"dim\_apppath"* key=*"Id"* caption=*"%{app\_path}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"path"* caption=*"%{app\_path}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_client"* name=*"%{base\_group}"*>

<Measures>

<!-- 新增用户 -->

<Measure name=*"New User Count"* column=*"uuid"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{new\_user}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"uuid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"customer\_id"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"AppPath"* foreignKeyColumn=*"pkg\_path"* />

<ForeignKeyLink dimension=*"Silent"* foreignKeyColumn=*"is\_silent"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgraded"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_lc\_01\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<CopyLink dimension=*"Date"* attribute=*"Month"*>

<Column aggColumn=*"the\_year"* table=*"v\_date"* name=*"the\_year"* />

<Column aggColumn=*"half\_year"* table=*"v\_date"* name=*"half\_year"* />

<Column aggColumn=*"quarter\_num"* table=*"v\_date"* name=*"quarter\_num"* />

<Column aggColumn=*"month\_num"* table=*"v\_date"* name=*"month\_num"* />

</CopyLink>

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_02\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_03\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"customer\_id"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_04\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<ForeignKeyLink dimension=*"AppPath"* foreignKeyColumn=*"pkg\_path"* />

<NoLink dimension=*"Silent"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_05\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_06\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_07\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Silent"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgrade"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<NamedSets>

<NamedSet name=*"Online Channels"*>

<Formula>

{

[Channel].[Channel].[Id].[SX910032001], [Channel].[Channel].[Id].[SX360031001], [Channel].[Channel].[Id].[SXWD0030001],

[Channel].[Channel].[Id].[SXBD0029001], [Channel].[Channel].[Id].[SXAZ0028001], [Channel].[Channel].[Id].[SXAZ0027001],

[Channel].[Channel].[Id].[SXJF0026001], [Channel].[Channel].[Id].[SXLX0024001], [Channel].[Channel].[Id].[SX3G0022001],

[Channel].[Channel].[Id].[SXAR0021001], [Channel].[Channel].[Id].[SXVI0020001], [Channel].[Channel].[Id].[SXLQ0019001],

[Channel].[Channel].[Id].[SXYH0016001], [Channel].[Channel].[Id].[SXMY0014001], [Channel].[Channel].[Id].[SXHW0013001],

[Channel].[Channel].[Id].[SXXM0012001], [Channel].[Channel].[Id].[SXMZ0011001], [Channel].[Channel].[Id].[SXOP0010001],

[Channel].[Channel].[Id].[SXZL0009001], [Channel].[Channel].[Id].[SXSJ0007001], [Channel].[Channel].[Id].[SXND0005001],

[Channel].[Channel].[Id].[SXTX0001001], [Channel].[Channel].[Id].[SXSG0110009], [Channel].[Channel].[Id].[SXWN0001001],

[Channel].[Channel].[Id].[SXYY0004001]

}

</Formula>

</NamedSet>

</NamedSets>

</Cube>

<!-- 活跃用户Cube -->

<Cube name=*"Active"* caption=*"%{active\_user}"*>

<Dimensions>

<Dimension name=*"CreateDate"* source=*"Date"* caption=*"%{create\_date}"* />

<Dimension name=*"ActiveDate"* source=*"Date"* caption=*"%{active\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Version"* caption=*"%{current\_version}"* />

<Dimension source=*"Product"* />

<Dimension source=*"DateDiff"* />

<Dimension name=*"VisitTimes"* source=*"Times"* caption=*"%{visit\_times}"* />

<!-- 日志类型维度 -->

<Dimension name=*"LogType"* table=*"v\_logtype"* key=*"Id"* caption=*"%{log\_type}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{log\_type}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_active"* name=*"%{base\_group}"*>

<Measures>

<!-- 活跃用户 -->

<Measure name=*"Active User Count"* column=*"uuid"* aggregator=*"distinct-count"* caption=*"%{active\_user}"* />

<!-- 访问次数 -->

<Measure name=*"Visit Times"* column=*"visit\_times"* aggregator=*"sum"* caption=*"%{visit\_times}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"CreateDate"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"ActiveDate"* foreignKeyColumn=*"active\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"uuid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"customer\_id"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"DateDiff"* foreignKeyColumn=*"date\_diff"* />

<ForeignKeyLink dimension=*"VisitTimes"* foreignKeyColumn=*"visit\_times"* />

<ForeignKeyLink dimension=*"LogType"* foreignKeyColumn=*"log\_type"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<!-- 人均访问次数 -->

<CalculatedMember name=*"Avg Visit Times"* dimension=*"Measures"* formatString=*"#,###.0"* caption=*"%{user\_visit\_times}"*>

<Formula>[Measures].[Visit Times] / [Measures].[Active User Count]</Formula>

</CalculatedMember>

<!-- 新增用户 -->

<CalculatedMember name=*"New User Count"* dimension=*"Measures"* formatString=*"#,###"* caption=*"%{new\_user}"*>

<Formula>Aggregate(CrossJoin([DateDiff].[Day].[0], [LogType].[Name].[Name].AllMembers))</Formula>

</CalculatedMember>

<!-- 新增用户占比 -->

<CalculatedMember name=*"New User Percent"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{new\_user\_pct}"*>

<Formula>[Measures].[New User Count] / [Measures].[Active User Count]</Formula>

</CalculatedMember>

</CalculatedMembers>

<NamedSets>

<NamedSet name=*"Online Channels"*>

<Formula>

{

[Channel].[Channel].[Id].[SX910032001], [Channel].[Channel].[Id].[SX360031001], [Channel].[Channel].[Id].[SXWD0030001],

[Channel].[Channel].[Id].[SXBD0029001], [Channel].[Channel].[Id].[SXAZ0028001], [Channel].[Channel].[Id].[SXAZ0027001],

[Channel].[Channel].[Id].[SXJF0026001], [Channel].[Channel].[Id].[SXLX0024001], [Channel].[Channel].[Id].[SX3G0022001],

[Channel].[Channel].[Id].[SXAR0021001], [Channel].[Channel].[Id].[SXVI0020001], [Channel].[Channel].[Id].[SXLQ0019001],

[Channel].[Channel].[Id].[SXYH0016001], [Channel].[Channel].[Id].[SXMY0014001], [Channel].[Channel].[Id].[SXHW0013001],

[Channel].[Channel].[Id].[SXXM0012001], [Channel].[Channel].[Id].[SXMZ0011001], [Channel].[Channel].[Id].[SXOP0010001],

[Channel].[Channel].[Id].[SXZL0009001], [Channel].[Channel].[Id].[SXSJ0007001], [Channel].[Channel].[Id].[SXND0005001],

[Channel].[Channel].[Id].[SXTX0001001], [Channel].[Channel].[Id].[SXSG0110009], [Channel].[Channel].[Id].[SXWN0001001],

[Channel].[Channel].[Id].[SXYY0004001]

}

</Formula>

</NamedSet>

</NamedSets>

</Cube>

<!-- 留存用户Cube -->

<Cube name=*"Keep"* caption=*"%{keep\_user}"*>

<Dimensions>

<Dimension name=*"CreateDate"* source=*"Date"* caption=*"%{create\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Product"* />

<Dimension source=*"DateDiff"* visible=*"false"* />

<!-- 日志类型维度 -->

<Dimension name=*"LogType"* table=*"v\_logtype"* key=*"Id"* caption=*"%{log\_type}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{log\_type}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_active"* name=*"%{base\_group}"*>

<Measures>

<!-- 活跃用户 -->

<Measure name=*"Active User Count"* column=*"uuid"* aggregator=*"distinct-count"* visible=*"false"* caption=*"%{active\_user}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"CreateDate"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"uuid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"customer\_id"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"DateDiff"* foreignKeyColumn=*"date\_diff"* />

<ForeignKeyLink dimension=*"LogType"* foreignKeyColumn=*"log\_type"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<!-- 新增用户 -->

<CalculatedMember name=*"New User Count"* dimension=*"Measures"* formatString=*"#,###"* caption=*"%{new\_user}"*>

<Formula>Aggregate(CrossJoin([DateDiff].[Day].[0], [LogType].[Name].[Name].AllMembers))</Formula>

</CalculatedMember>

<!-- 次日留存率 -->

<CalculatedMember name=*"Keep User Percent 1"* dimension=*"Measures"* caption=*"%{keep\_pct\_1}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[1] / [Measures].[New User Count], NULL)</Formula>

<CalculatedMemberProperty name=*"FORMAT\_STRING"* expression=*"Iif([Measures].[Keep User Percent 1] &lt; 0.2, '|0.00%|style=green', '|0.00%|style=red')"* />

</CalculatedMember>

<!-- 2日留存率 -->

<CalculatedMember name=*"Keep User Percent 2"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_2}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[2] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 3日留存率 -->

<CalculatedMember name=*"Keep User Percent 3"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_3}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[3] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 4日留存率 -->

<CalculatedMember name=*"Keep User Percent 4"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_4}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[4] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 5日留存率 -->

<CalculatedMember name=*"Keep User Percent 5"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_5}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[5] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 6日留存率 -->

<CalculatedMember name=*"Keep User Percent 6"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_6}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[6] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 7日留存率 -->

<CalculatedMember name=*"Keep User Percent 7"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_7}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[7] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 30日留存率 -->

<CalculatedMember name=*"Keep User Percent 30"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_30}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[30] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 3日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 3"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in3}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[3]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 7日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 7"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in7}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[7]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 14日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 14"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in14}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[14]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 30日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 30"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in30}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[30]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 7日后留存率 -->

<CalculatedMember name=*"Keep User Percent Out 7"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_out7}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[8] : [DateDiff].[Day].[50]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 14日后留存率 -->

<CalculatedMember name=*"Keep User Percent Out 14"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_out14}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[15] : [DateDiff].[Day].[50]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 30日后留存率 -->

<CalculatedMember name=*"Keep User Percent Out 30"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_out30}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[31] : [DateDiff].[Day].[50]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

</CalculatedMembers>

<NamedSets>

<NamedSet name=*"Online Channels"*>

<Formula>

{

[Channel].[Channel].[Id].[SX910032001], [Channel].[Channel].[Id].[SX360031001], [Channel].[Channel].[Id].[SXWD0030001],

[Channel].[Channel].[Id].[SXBD0029001], [Channel].[Channel].[Id].[SXAZ0028001], [Channel].[Channel].[Id].[SXAZ0027001],

[Channel].[Channel].[Id].[SXJF0026001], [Channel].[Channel].[Id].[SXLX0024001], [Channel].[Channel].[Id].[SX3G0022001],

[Channel].[Channel].[Id].[SXAR0021001], [Channel].[Channel].[Id].[SXVI0020001], [Channel].[Channel].[Id].[SXLQ0019001],

[Channel].[Channel].[Id].[SXYH0016001], [Channel].[Channel].[Id].[SXMY0014001], [Channel].[Channel].[Id].[SXHW0013001],

[Channel].[Channel].[Id].[SXXM0012001], [Channel].[Channel].[Id].[SXMZ0011001], [Channel].[Channel].[Id].[SXOP0010001],

[Channel].[Channel].[Id].[SXZL0009001], [Channel].[Channel].[Id].[SXSJ0007001], [Channel].[Channel].[Id].[SXND0005001],

[Channel].[Channel].[Id].[SXTX0001001], [Channel].[Channel].[Id].[SXSG0110009], [Channel].[Channel].[Id].[SXWN0001001],

[Channel].[Channel].[Id].[SXYY0004001]

}

</Formula>

</NamedSet>

</NamedSets>

</Cube>

<!-- 角色权限 -->

<Role name=*"ROLE\_POP"*>

<SchemaGrant access=*"none"* />

</Role>

<Role name=*"ROLE\_ADOP"*>

<SchemaGrant access=*"none"* />

</Role>

<Role name=*"ROLE\_ADOPN"*>

<SchemaGrant access=*"none"* />

</Role>

</Schema>

fad.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Schema name=*"%{fad}"* metamodelVersion=*"4.0"* xmlns=*"http://mondrian.pentaho.com/schema"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://mondrian.pentaho.com/schema http://mondrian.pentaho.com/schema/mondrian.xsd"*>

<!-- 物理模型 -->

<PhysicalSchema>

<Query alias=*"v\_date"* keyColumn=*"id"*>

<ExpressionView>

<SQL dialect=*"mysql"*><![CDATA[SELECT \* FROM dim\_date WHERE id >= 20170305 AND id <= CURDATE()]]></SQL>

</ExpressionView>

</Query>

<Query alias=*"v\_hour"* keyColumn=*"id"*>

<ExpressionView>

<SQL><![CDATA[SELECT id FROM dim\_cohort WHERE id >= 0 AND id <= 23]]></SQL>

</ExpressionView>

</Query>

<Table name=*"dim\_city"* keyColumn=*"id"* />

<Table name=*"dim\_channel"* keyColumn=*"id"* />

<Table name=*"dim\_product"* keyColumn=*"id"* />

<Table name=*"dim\_version"* keyColumn=*"id"* />

<Table name=*"dim\_network"* keyColumn=*"name"* />

<Table name=*"dim\_cohort"* keyColumn=*"id"* />

<Table name=*"dim\_yorn"* keyColumn=*"id"* />

<Table name=*"dim\_yorn"* keyColumn=*"id"* alias=*"yorn1"* />

<Table name=*"dim\_yorn"* keyColumn=*"id"* alias=*"yorn2"* />

<Table name=*"dim\_yorn"* keyColumn=*"id"* alias=*"yorn3"* />

<Table name=*"dim\_device"* keyColumn=*"udid"*>

<ColumnDefs>

<CalculatedColumnDef name=*"is\_rooted"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(is\_root = 1, 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"has\_gp"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(has\_gplay = 1, 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"has\_gaid"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(gaid > '', 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"has\_sim"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(imsi > '', 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Link source=*"dim\_network"* target=*"dim\_device"* foreignKeyColumn=*"network"* />

<Link source=*"dim\_yorn"* target=*"dim\_device"* foreignKeyColumn=*"is\_rooted"* />

<Link source=*"yorn1"* target=*"dim\_device"* foreignKeyColumn=*"has\_gp"* />

<Link source=*"yorn2"* target=*"dim\_device"* foreignKeyColumn=*"has\_gaid"* />

<Link source=*"yorn3"* target=*"dim\_device"* foreignKeyColumn=*"has\_sim"* />

<Table name=*"dim\_apppath"* keyColumn=*"id"* />

<Table name=*"dim\_level"* keyColumn=*"id"* />

<Table name=*"dim\_ad"* keyColumn=*"id"* />

<Table name=*"dim\_adver"* keyColumn=*"id"* />

<Table name=*"dim\_resource"* keyColumn=*"id"*>

<ColumnDefs>

<CalculatedColumnDef name=*"res\_source"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE sdktype WHEN 1 THEN '自主运营' WHEN 3 THEN '自动运营' ELSE '其他' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"res\_type"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE restype WHEN 1 THEN 'apk' WHEN 2 THEN 'wap' ELSE '其他' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"dim\_position"* keyColumn=*"id"* />

<Link source=*"dim\_adver"* target=*"dim\_ad"* foreignKeyColumn=*"adverid"* />

<Link source=*"dim\_resource"* target=*"dim\_ad"* foreignKeyColumn=*"resoid"* />

<Link source=*"dim\_position"* target=*"dim\_ad"* foreignKeyColumn=*"advtype"* />

<Table name=*"fact\_client"*>

<ColumnDefs>

<CalculatedColumnDef name=*"is\_upgraded"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(version != init\_version, 1, 0)</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"fact\_active"* />

<Table name=*"fact\_blacklist"* />

<Table name=*"fact\_runlevel"* />

<Table name=*"fact\_ad"* />

<Query alias=*"v\_show"*>

<ExpressionView>

<SQL><![CDATA[SELECT \* FROM fact\_ad WHERE show\_date IS NOT NULL]]></SQL>

</ExpressionView>

</Query>

<Query alias=*"v\_click"*>

<ExpressionView>

<SQL><![CDATA[SELECT \* FROM fact\_ad WHERE click\_date IS NOT NULL]]></SQL>

</ExpressionView>

</Query>

<Query alias=*"v\_install"*>

<ExpressionView>

<SQL><![CDATA[SELECT \* FROM fact\_ad WHERE install\_date IS NOT NULL]]></SQL>

</ExpressionView>

</Query>

<Query alias=*"v\_close"*>

<ExpressionView>

<SQL><![CDATA[SELECT \* FROM fact\_ad WHERE close\_date IS NOT NULL]]></SQL>

</ExpressionView>

</Query>

<Query alias=*"v\_uninstall"*>

<ExpressionView>

<SQL><![CDATA[SELECT \* FROM fact\_ad WHERE uninstall\_date IS NOT NULL]]></SQL>

</ExpressionView>

</Query>

<Table name=*"fact\_upgrade"*>

<ColumnDefs>

<CalculatedColumnDef name=*"is\_upgraded"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(upgrade\_date IS NULL, 0, 1)</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"agg\_l\_01\_client"* />

<Table name=*"agg\_l\_02\_client"* />

<Table name=*"agg\_l\_03\_client"* />

<Table name=*"agg\_l\_04\_client"* />

<Table name=*"agg\_l\_05\_client"* />

</PhysicalSchema>

<!-- 逻辑模型 -->

<!-- 共享维度 -->

<!-- 日期维度 -->

<Dimension name=*"Date"* table=*"v\_date"* key=*"Date"* type=*"TIME"* caption=*"%{date}"*>

<Attributes>

<Attribute name=*"Year"* keyColumn=*"the\_year"* levelType=*"TimeYears"* hasHierarchy=*"false"* />

<Attribute name=*"HalfYear"* levelType=*"TimeHalfYear"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"half\_year"* />

</Key>

<Name>

<Column name=*"half\_year"* />

</Name>

</Attribute>

<Attribute name=*"Quarter"* levelType=*"TimeQuarters"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"quarter\_num"* />

</Key>

<Name>

<Column name=*"en\_quarter"* />

</Name>

</Attribute>

<Attribute name=*"Month"* levelType=*"TimeMonths"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"month\_num"* />

</Key>

<Name>

<Column name=*"month\_num"* />

</Name>

</Attribute>

<Attribute name=*"Week"* levelType=*"TimeWeeks"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"week\_num"* />

</Key>

<Name>

<Column name=*"week\_num"* />

</Name>

</Attribute>

<Attribute name=*"Day"* keyColumn=*"id"* nameColumn=*"day\_num"* levelType=*"TimeDays"* hasHierarchy=*"false"* />

<Attribute name=*"Date"* keyColumn=*"id"* nameColumn=*"the\_day"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Monthly"*>

<Level attribute=*"Year"* caption=*"%{year}"* />

<Level attribute=*"HalfYear"* caption=*"%{half\_year}"* />

<Level attribute=*"Quarter"* caption=*"%{quarter}"* />

<Level attribute=*"Month"* caption=*"%{month}"* />

<Level attribute=*"Day"* caption=*"%{day}"* />

</Hierarchy>

<Hierarchy name=*"Daily"*>

<Level attribute=*"Date"* caption=*"%{date}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 设备维度 -->

<Dimension name=*"Device"* table=*"dim\_device"* key=*"UDID"* caption=*"%{device}"*>

<Attributes>

<Attribute name=*"Platform"* keyColumn=*"platform"* hasHierarchy=*"false"* />

<Attribute name=*"OSVersion"* keyColumn=*"os\_version"* hasHierarchy=*"false"* />

<Attribute name=*"Network"* table=*"dim\_network"* keyColumn=*"name"* orderByColumn=*"priority"* hierarchyCaption=*"%{network}"* caption=*"%{type}"* />

<Attribute name=*"Brand"* keyColumn=*"vender"* caption=*"%{brand}"* />

<Attribute name=*"Model"* keyColumn=*"model"* caption=*"%{model}"* />

<Attribute name=*"Resolution"* keyColumn=*"src"* caption=*"%{resolution}"* />

<Attribute name=*"IsRoot"* table=*"dim\_yorn"* keyColumn=*"id"* nameColumn=*"name"* hierarchyCaption=*"%{is\_rooted}"* caption=*"%{yesorno}"* />

<Attribute name=*"HasGPlay"* table=*"yorn1"* keyColumn=*"id"* nameColumn=*"name"* hierarchyCaption=*"%{has\_gplay}"* caption=*"%{yesorno}"* />

<Attribute name=*"HasGaid"* table=*"yorn2"* keyColumn=*"id"* nameColumn=*"name"* hierarchyCaption=*"%{has\_gaid}"* caption=*"%{yesorno}"* />

<Attribute name=*"HasSim"* table=*"yorn3"* keyColumn=*"id"* nameColumn=*"name"* hierarchyCaption=*"%{has\_sim}"* caption=*"%{yesorno}"* />

<Attribute name=*"UDID"* keyColumn=*"udid"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Platform"* caption=*"%{platform}"*>

<Level attribute=*"Platform"* caption=*"%{name}"* />

<Level attribute=*"OSVersion"* caption=*"%{version}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 日期间隔维度 -->

<Dimension name=*"DateDiff"* table=*"dim\_cohort"* key=*"Day"* caption=*"%{date\_diff}"*>

<Attributes>

<Attribute name=*"Day"* keyColumn=*"id"* caption=*"%{day\_diff}"* />

</Attributes>

</Dimension>

<!-- 小时维度 -->

<Dimension name=*"Hour"* table=*"v\_hour"* key=*"Name"* caption=*"%{hour}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* caption=*"%{hour}"* />

</Attributes>

</Dimension>

<!-- App版本维度 -->

<Dimension name=*"Version"* table=*"dim\_version"* key=*"Num"* caption=*"%{version}"*>

<Attributes>

<Attribute name=*"Num"* keyColumn=*"id"* caption=*"%{version\_num}"* />

</Attributes>

</Dimension>

<!-- 渠道维度 -->

<Dimension name=*"Channel"* table=*"dim\_channel"* key=*"Id"* caption=*"%{channel}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"channel\_name"* hasHierarchy=*"false"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Channel"*>

<Level attribute=*"Name"* caption=*"%{name}"* />

<Level attribute=*"Id"* caption=*"%{code}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 产品维度 -->

<Dimension name=*"Product"* table=*"dim\_product"* key=*"Id"* caption=*"%{product}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* hasHierarchy=*"false"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Product"*>

<Level attribute=*"Name"* caption=*"%{name}"* />

<Level attribute=*"Id"* caption=*"%{code}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 是否维度 -->

<Dimension name=*"Yorn"* table=*"dim\_yorn"* key=*"Id"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{yesorno}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 网络类型维度 -->

<Dimension name=*"Network"* table=*"dim\_network"* key=*"Name"* caption=*"%{network}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"name"* orderByColumn=*"priority"* caption=*"%{type}"* />

</Attributes>

</Dimension>

<!-- 次数维度 -->

<Dimension name=*"Times"* table=*"dim\_cohort"* key=*"Id"*>

<Attributes>

<Attribute name=*"Range\_1"* keyColumn=*"times\_1"* nameColumn=*"times\_1\_desc"* caption=*"%{times\_range}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 广告维度 -->

<Dimension name=*"Ad"* table=*"dim\_ad"* key=*"Id"* caption=*"%{ad}"*>

<Attributes>

<Attribute name=*"Position"* table=*"dim\_position"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{ad\_position}"* />

<Attribute name=*"Name"* keyColumn=*"name"* caption=*"%{name}"* />

<Attribute name=*"Adver"* table=*"dim\_adver"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{adver}"* />

<Attribute name=*"ResSource"* table=*"dim\_resource"* keyColumn=*"res\_source"* caption=*"%{res\_source}"* />

<Attribute name=*"ResType"* table=*"dim\_resource"* keyColumn=*"res\_type"* caption=*"%{res\_type}"* />

<Attribute name=*"ResName"* table=*"dim\_resource"* keyColumn=*"name"* caption=*"%{res\_name}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 新增用户Cube -->

<Cube name=*"Client"* caption=*"%{new\_user}"*>

<Dimensions>

<Dimension source=*"Date"* caption=*"%{create\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Version"* caption=*"%{current\_version}"* />

<Dimension source=*"Product"* />

<Dimension name=*"Upgrade"* source=*"Yorn"* caption=*"%{is\_upgraded}"* />

<!-- App安装路径维度 -->

<Dimension name=*"AppPath"* table=*"dim\_apppath"* key=*"Id"* caption=*"%{app\_path}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"path"* caption=*"%{app\_path}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_client"* name=*"%{base\_group}"*>

<Measures>

<!-- 新增用户 -->

<Measure name=*"New User Count"* column=*"udid"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{new\_user}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"AppPath"* foreignKeyColumn=*"app\_path"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgraded"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_01\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_02\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_03\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<NoLink dimension=*"AppPath"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_04\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<NoLink dimension=*"AppPath"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgraded"* />

</DimensionLinks>

</MeasureGroup>

<MeasureGroup table=*"agg\_l\_05\_client"* type=*"aggregate"*>

<Measures>

<MeasureRef name=*"Fact Count"* aggColumn=*"fact\_count"* />

<MeasureRef name=*"New User Count"* aggColumn=*"fact\_count"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<NoLink dimension=*"Device"* />

<NoLink dimension=*"Channel"* />

<NoLink dimension=*"InitVersion"* />

<NoLink dimension=*"Version"* />

<ForeignKeyLink dimension=*"AppPath"* foreignKeyColumn=*"app\_path"* />

<NoLink dimension=*"Upgrade"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 活跃用户Cube -->

<Cube name=*"Active"* caption=*"%{active\_user}"*>

<Dimensions>

<Dimension name=*"CreateDate"* source=*"Date"* caption=*"%{create\_date}"* />

<Dimension name=*"ActiveDate"* source=*"Date"* caption=*"%{active\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Version"* caption=*"%{current\_version}"* />

<Dimension source=*"Product"* />

<Dimension source=*"DateDiff"* />

<Dimension name=*"VisitTimes"* source=*"Times"* caption=*"%{visit\_times}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_active"* name=*"%{base\_group}"*>

<Measures>

<!-- 活跃用户 -->

<Measure name=*"Active User Count"* column=*"udid"* aggregator=*"distinct-count"* caption=*"%{active\_user}"* />

<!-- 访问次数 -->

<Measure name=*"Visit Times"* column=*"visit\_times"* aggregator=*"sum"* caption=*"%{visit\_times}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"CreateDate"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"ActiveDate"* foreignKeyColumn=*"active\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Version"* foreignKeyColumn=*"version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"DateDiff"* foreignKeyColumn=*"date\_diff"* />

<ForeignKeyLink dimension=*"VisitTimes"* foreignKeyColumn=*"visit\_times"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<!-- 人均访问次数 -->

<CalculatedMember name=*"Avg Visit Times"* dimension=*"Measures"* formatString=*"#,###.0"* caption=*"%{user\_visit\_times}"*>

<Formula>[Measures].[Visit Times] / [Measures].[Active User Count]</Formula>

</CalculatedMember>

<!-- 新增用户 -->

<CalculatedMember name=*"New User Count"* dimension=*"Measures"* formatString=*"#,###"* caption=*"%{new\_user}"*>

<Formula>[DateDiff].[Day].[0]</Formula>

</CalculatedMember>

<!-- 新增用户占比 -->

<CalculatedMember name=*"New User Percent"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{new\_user\_pct}"*>

<Formula>[Measures].[New User Count] / [Measures].[Active User Count]</Formula>

</CalculatedMember>

</CalculatedMembers>

</Cube>

<!-- 留存用户Cube -->

<Cube name=*"Keep"* caption=*"%{keep\_user}"*>

<Dimensions>

<Dimension name=*"CreateDate"* source=*"Date"* caption=*"%{create\_date}"* />

<Dimension source=*"Device"* />

<Dimension source=*"Channel"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension source=*"Product"* />

<Dimension source=*"DateDiff"* visible=*"false"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_active"* name=*"%{base\_group}"*>

<Measures>

<!-- 活跃用户 -->

<Measure name=*"Active User Count"* column=*"udid"* aggregator=*"distinct-count"* visible=*"false"* caption=*"%{active\_user}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"CreateDate"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"init\_version"* />

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"DateDiff"* foreignKeyColumn=*"date\_diff"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<!-- 新增用户 -->

<CalculatedMember name=*"New User Count"* dimension=*"Measures"* formatString=*"#,###"* caption=*"%{new\_user}"*>

<Formula>[DateDiff].[Day].[0]</Formula>

</CalculatedMember>

<!-- 次日留存率 -->

<CalculatedMember name=*"Keep User Percent 1"* dimension=*"Measures"* caption=*"%{keep\_pct\_1}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[1] / [Measures].[New User Count], NULL)</Formula>

<CalculatedMemberProperty name=*"FORMAT\_STRING"* expression=*"Iif([Measures].[Keep User Percent 1] &lt; 0.2, '|0.00%|style=green', '|0.00%|style=red')"* />

</CalculatedMember>

<!-- 2日留存率 -->

<CalculatedMember name=*"Keep User Percent 2"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_2}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[2] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 3日留存率 -->

<CalculatedMember name=*"Keep User Percent 3"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_3}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[3] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 4日留存率 -->

<CalculatedMember name=*"Keep User Percent 4"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_4}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[4] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 5日留存率 -->

<CalculatedMember name=*"Keep User Percent 5"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_5}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[5] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 6日留存率 -->

<CalculatedMember name=*"Keep User Percent 6"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_6}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[6] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 7日留存率 -->

<CalculatedMember name=*"Keep User Percent 7"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_7}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[7] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 30日留存率 -->

<CalculatedMember name=*"Keep User Percent 30"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_30}"*>

<Formula>Iif([Measures].[New User Count] > 0, [DateDiff].[Day].[30] / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 3日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 3"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in3}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[3]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 7日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 7"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in7}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[7]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 14日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 14"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in14}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[14]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

<!-- 30日内留存率 -->

<CalculatedMember name=*"Keep User Percent In 30"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{keep\_pct\_in30}"*>

<Formula>Iif([Measures].[New User Count] > 0, Aggregate([DateDiff].[Day].[1] : [DateDiff].[Day].[30]) / [Measures].[New User Count], NULL)</Formula>

</CalculatedMember>

</CalculatedMembers>

</Cube>

<!-- 黑名单Cube -->

<Cube name=*"Blacklist"* caption=*"%{blacklist}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Date"* caption=*"%{stat\_date}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_blacklist"* name=*"%{base\_group}"*>

<Measures>

<!-- 黑名单量 -->

<Measure name=*"Black Count"* column=*"black\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{black\_count}"* />

<!-- 待释放量 -->

<Measure name=*"Release Count"* column=*"release\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{release\_count}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"stat\_date"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 运营Cube -->

<Cube name=*"Operation"* caption=*"%{operation}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Date"* caption=*"%{create\_date}"* />

<Dimension name=*"Level"* table=*"dim\_level"* key=*"Id"* caption=*"%{run\_level}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{name}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_runlevel"* name=*"%{base\_group}"*>

<Measures>

<Measure name=*"User Count"* column=*"user\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{user\_count}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Level"* foreignKeyColumn=*"runlevel"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-下发Cube -->

<Cube name=*"AdSend"* caption=*"%{ad\_send}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_ad"* name=*"%{base\_group}"*>

<Measures>

<!-- 下发个数 -->

<Measure name=*"Send Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{send\_count}"* />

<!-- 下发人数 -->

<Measure name=*"Send User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{send\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-展示Cube -->

<Cube name=*"AdShow"* caption=*"%{ad\_show}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

<Dimension name=*"ShowDate"* source=*"Date"* caption=*"%{show\_date}"* />

<Dimension name=*"ShowHour"* source=*"Hour"* caption=*"%{show\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"v\_show"* name=*"%{base\_group}"*>

<Measures>

<!-- 展示个数 -->

<Measure name=*"Show Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{show\_count}"* />

<!-- 展示人数 -->

<Measure name=*"Show User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{show\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

<ForeignKeyLink dimension=*"ShowDate"* foreignKeyColumn=*"show\_date"* />

<ForeignKeyLink dimension=*"ShowHour"* foreignKeyColumn=*"show\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-点击Cube -->

<Cube name=*"AdClick"* caption=*"%{ad\_click}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

<Dimension name=*"ClickDate"* source=*"Date"* caption=*"%{click\_date}"* />

<Dimension name=*"ClickHour"* source=*"Hour"* caption=*"%{click\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"v\_click"* name=*"%{base\_group}"*>

<Measures>

<!-- 点击个数 -->

<Measure name=*"Click Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{click\_count}"* />

<!-- 点击人数 -->

<Measure name=*"Click User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{click\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

<ForeignKeyLink dimension=*"ClickDate"* foreignKeyColumn=*"click\_date"* />

<ForeignKeyLink dimension=*"ClickHour"* foreignKeyColumn=*"click\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-安装Cube -->

<Cube name=*"AdInstall"* caption=*"%{ad\_install}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

<Dimension name=*"InstallDate"* source=*"Date"* caption=*"%{install\_date}"* />

<Dimension name=*"InstallHour"* source=*"Hour"* caption=*"%{install\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"v\_install"* name=*"%{base\_group}"*>

<Measures>

<!-- 安装个数 -->

<Measure name=*"Install Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{install\_count}"* />

<!-- 安装人数 -->

<Measure name=*"Install User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{install\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

<ForeignKeyLink dimension=*"InstallDate"* foreignKeyColumn=*"install\_date"* />

<ForeignKeyLink dimension=*"InstallHour"* foreignKeyColumn=*"install\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-关闭Cube -->

<Cube name=*"AdClose"* caption=*"%{ad\_close}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

<Dimension name=*"CloseDate"* source=*"Date"* caption=*"%{close\_date}"* />

<Dimension name=*"CloseHour"* source=*"Hour"* caption=*"%{close\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"v\_close"* name=*"%{base\_group}"*>

<Measures>

<!-- 关闭个数 -->

<Measure name=*"Close Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{close\_count}"* />

<!-- 关闭人数 -->

<Measure name=*"Close User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{close\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

<ForeignKeyLink dimension=*"CloseDate"* foreignKeyColumn=*"close\_date"* />

<ForeignKeyLink dimension=*"CloseHour"* foreignKeyColumn=*"close\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 广告-卸载Cube -->

<Cube name=*"AdUninstall"* caption=*"%{ad\_uninstall}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Ad"* />

<Dimension source=*"Device"* />

<Dimension name=*"SendDate"* source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"SendHour"* source=*"Hour"* caption=*"%{send\_hour}"* />

<Dimension name=*"UninstallDate"* source=*"Date"* caption=*"%{uninstall\_date}"* />

<Dimension name=*"UninstallHour"* source=*"Hour"* caption=*"%{uninstall\_hour}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"v\_uninstall"* name=*"%{base\_group}"*>

<Measures>

<!-- 卸载个数 -->

<Measure name=*"Uninstall Count"* column=*"adcode"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{uninstall\_count}"* />

<!-- 卸载人数 -->

<Measure name=*"Uninstall User"* column=*"udid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{uninstall\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Ad"* foreignKeyColumn=*"adkey"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"SendDate"* foreignKeyColumn=*"send\_date"* />

<ForeignKeyLink dimension=*"SendHour"* foreignKeyColumn=*"send\_hour"* />

<ForeignKeyLink dimension=*"UninstallDate"* foreignKeyColumn=*"uninstall\_date"* />

<ForeignKeyLink dimension=*"UninstallHour"* foreignKeyColumn=*"uninstall\_hour"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 升级反馈Cube -->

<Cube name=*"Upgrade Feedback"* caption=*"%{upgrade\_feedback}"*>

<Dimensions>

<Dimension source=*"Product"* />

<Dimension source=*"Channel"* />

<Dimension source=*"Device"* />

<Dimension name=*"InitVersion"* source=*"Version"* caption=*"%{initial\_version}"* />

<Dimension name=*"UpVersion"* source=*"Version"* caption=*"%{upgrade\_version}"* />

<Dimension source=*"Date"* caption=*"%{send\_date}"* />

<Dimension name=*"Upgrade"* source=*"Yorn"* caption=*"%{is\_upgraded}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_upgrade"* name=*"%{base\_group}"*>

<Measures>

<Measure name=*"User Count"* column=*"udid"* aggregator=*"count"* formatString=*"#,###"* caption=*"%{user\_count}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"Product"* foreignKeyColumn=*"app\_key"* />

<ForeignKeyLink dimension=*"Channel"* foreignKeyColumn=*"clnt"* />

<ForeignKeyLink dimension=*"Device"* foreignKeyColumn=*"udid"* />

<ForeignKeyLink dimension=*"InitVersion"* foreignKeyColumn=*"version"* />

<ForeignKeyLink dimension=*"UpVersion"* foreignKeyColumn=*"up\_version"* />

<ForeignKeyLink dimension=*"Date"* foreignKeyColumn=*"create\_date"* />

<ForeignKeyLink dimension=*"Upgrade"* foreignKeyColumn=*"is\_upgraded"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 角色权限 -->

<Role name=*"ROLE\_POP"*>

<SchemaGrant access=*"none"* />

</Role>

<Role name=*"ROLE\_ADOPN"*>

<SchemaGrant access=*"none"* />

</Role>

</Schema>

yaya.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Schema name=*"%{yaya}"* metamodelVersion=*"4.0"* xmlns=*"http://mondrian.pentaho.com/schema"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://mondrian.pentaho.com/schema http://mondrian.pentaho.com/schema/mondrian.xsd"*>

<!-- 物理模型 -->

<PhysicalSchema>

<Query alias=*"v\_date"* keyColumn=*"id"*>

<ExpressionView>

<SQL dialect=*"mysql"*><![CDATA[SELECT \* FROM dim\_date WHERE id >= 20151219 AND id <= CURDATE()]]></SQL>

</ExpressionView>

</Query>

<Table name=*"dim\_info"* keyColumn=*"id"*>

<ColumnDefs>

<CalculatedColumnDef name=*"type\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE content\_type WHEN 0 THEN '资讯' WHEN 1 THEN '视频' ELSE '其他' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"dim\_info\_cat"* keyColumn=*"id"*>

<ColumnDefs>

<CalculatedColumnDef name=*"type\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE content\_type WHEN 0 THEN '资讯' WHEN 1 THEN '视频' ELSE '其他' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Link source=*"dim\_info\_cat"* target=*"dim\_info"* foreignKeyColumn=*"category\_id"* />

<Table name=*"fact\_info"* />

<Table name=*"fact\_info\_detail"*>

<ColumnDefs>

<ColumnDef name=*"id"* type=*"Integer"* internalType=*"long"* />

<CalculatedColumnDef name=*"type\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE fact\_info\_detail.content\_type WHEN 0 THEN '资讯' WHEN 1 THEN '视频' ELSE '其他' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"platform\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>CASE source\_platform WHEN 1 THEN '自媒体' WHEN 2 THEN '其他' ELSE '牙牙资讯' END</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"index\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(is\_index = 1, '是', '否')</SQL>

</ExpressionView>

</CalculatedColumnDef>

<CalculatedColumnDef name=*"push\_name"*>

<ExpressionView>

<SQL dialect=*"mysql"*>IF(is\_push = 1, '是', '否')</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"fact\_info\_cat"*>

<ColumnDefs>

<CalculatedColumnDef name=*"user\_pct"*>

<ExpressionView>

<SQL dialect=*"generic"*>sub\_count / total\_count</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

<Table name=*"fact\_interact"*>

<ColumnDefs>

<ColumnDef name=*"id"* type=*"Integer"* internalType=*"long"* />

</ColumnDefs>

</Table>

<Table name=*"stat\_star"* />

<Table name=*"stat\_search"*>

<ColumnDefs>

<CalculatedColumnDef name=*"search\_pct"*>

<ExpressionView>

<SQL dialect=*"generic"*>search\_count / all\_count</SQL>

</ExpressionView>

</CalculatedColumnDef>

</ColumnDefs>

</Table>

</PhysicalSchema>

<!-- 逻辑模型 -->

<!-- 共享维度 -->

<!-- 日期维度 -->

<Dimension name=*"Date"* table=*"v\_date"* key=*"Date"* type=*"TIME"* caption=*"%{date}"*>

<Attributes>

<Attribute name=*"Year"* keyColumn=*"the\_year"* levelType=*"TimeYears"* hasHierarchy=*"false"* />

<Attribute name=*"HalfYear"* levelType=*"TimeHalfYear"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"half\_year"* />

</Key>

<Name>

<Column name=*"half\_year"* />

</Name>

</Attribute>

<Attribute name=*"Quarter"* levelType=*"TimeQuarters"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"quarter\_num"* />

</Key>

<Name>

<Column name=*"en\_quarter"* />

</Name>

</Attribute>

<Attribute name=*"Month"* levelType=*"TimeMonths"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"month\_num"* />

</Key>

<Name>

<Column name=*"month\_num"* />

</Name>

</Attribute>

<Attribute name=*"Week"* levelType=*"TimeWeeks"* hasHierarchy=*"false"*>

<Key>

<Column name=*"the\_year"* />

<Column name=*"week\_num"* />

</Key>

<Name>

<Column name=*"week\_num"* />

</Name>

</Attribute>

<Attribute name=*"Day"* keyColumn=*"id"* nameColumn=*"day\_num"* levelType=*"TimeDays"* hasHierarchy=*"false"* />

<Attribute name=*"Date"* keyColumn=*"id"* nameColumn=*"the\_day"* hasHierarchy=*"false"* />

</Attributes>

<Hierarchies>

<Hierarchy name=*"Monthly"*>

<Level attribute=*"Year"* caption=*"%{year}"* />

<Level attribute=*"HalfYear"* caption=*"%{half\_year}"* />

<Level attribute=*"Quarter"* caption=*"%{quarter}"* />

<Level attribute=*"Month"* caption=*"%{month}"* />

<Level attribute=*"Day"* caption=*"%{day}"* />

</Hierarchy>

<Hierarchy name=*"Daily"*>

<Level attribute=*"Date"* caption=*"%{date}"* />

</Hierarchy>

</Hierarchies>

</Dimension>

<!-- 资讯维度 -->

<Dimension name=*"Info"* table=*"dim\_info"* key=*"Id"* caption=*"%{info}"*>

<Attributes>

<Attribute name=*"Type"* keyColumn=*"content\_type"* nameColumn=*"type\_name"* caption=*"%{info\_type}"* />

<Attribute name=*"Category"* table=*"dim\_info\_cat"* keyColumn=*"id"* nameColumn=*"name"* caption=*"%{info\_category}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 资讯分类维度 -->

<Dimension name=*"InfoCategory"* table=*"dim\_info\_cat"* key=*"Id"* caption=*"%{info\_category}"*>

<Attributes>

<Attribute name=*"Type"* keyColumn=*"type\_name"* caption=*"%{info\_type}"* />

<Attribute name=*"Name"* keyColumn=*"name"* caption=*"%{category\_name}"* />

<Attribute name=*"Id"* keyColumn=*"id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

<!-- 资讯点击Cube -->

<Cube name=*"Info"* caption=*"%{info}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

<Dimension name=*"PublishDate"* source=*"Date"* caption=*"%{publish\_date}"* />

<Dimension source=*"InfoCategory"* />

<Dimension name=*"Info"* table=*"fact\_info\_detail"* key=*"Id"* caption=*"%{info}"*>

<Attributes>

<Attribute name=*"Type"* keyColumn=*"content\_type"* nameColumn=*"type\_name"* caption=*"%{info\_type}"* />

<Attribute name=*"SrcPlatform"* keyColumn=*"source\_platform"* nameColumn=*"platform\_name"* caption=*"%{src\_platform}"* />

<Attribute name=*"Creater"* keyColumn=*"create\_by"* caption=*"%{creater}"* />

<Attribute name=*"Title"* keyColumn=*"title"* hasHierarchy=*"false"* />

<Attribute name=*"Source"* keyColumn=*"source"* hasHierarchy=*"false"* />

<Attribute name=*"IsIndex"* keyColumn=*"index\_name"* hasHierarchy=*"false"* />

<Attribute name=*"IsPush"* keyColumn=*"push\_name"* hasHierarchy=*"false"* />

<Attribute name=*"Id"* keyColumn=*"id"* caption=*"%{info\_id}"*>

<Property attribute=*"Title"* />

<Property attribute=*"Source"* />

<Property attribute=*"IsIndex"* />

<Property attribute=*"IsPush"* />

</Attribute>

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_info\_detail"* name=*"%{base\_group}"*>

<Measures>

<!-- 点击次数 -->

<Measure name=*"Click Times"* column=*"click\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_times}"* />

<!-- 点击人数 -->

<Measure name=*"Click User Count"* column=*"click\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_users}"* />

<!-- 评论次数 -->

<Measure name=*"Comment Times"* column=*"comment\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{comment\_times}"* />

<!-- 评论人数 -->

<Measure name=*"Comment User Count"* column=*"comment\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{comment\_users}"* />

<!-- 分享次数 -->

<Measure name=*"Share Times"* column=*"share\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{share\_times}"* />

<!-- 分享人数 -->

<Measure name=*"Share User Count"* column=*"share\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{share\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"stat\_date"* />

<ForeignKeyLink dimension=*"PublishDate"* foreignKeyColumn=*"publish\_date"* />

<ForeignKeyLink dimension=*"InfoCategory"* foreignKeyColumn=*"category\_id"* />

<FactLink dimension=*"Info"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<CalculatedMember name=*"Info Title"* dimension=*"Measures"* caption=*"%{title}"*>

<Formula>[Info].[Id].CurrentMember.Properties('Title')</Formula>

</CalculatedMember>

<CalculatedMember name=*"Info Source"* dimension=*"Measures"* caption=*"%{source}"*>

<Formula>[Info].[Id].CurrentMember.Properties('Source')</Formula>

</CalculatedMember>

<CalculatedMember name=*"Info IsIndex"* dimension=*"Measures"* caption=*"%{is\_index}"*>

<Formula>[Info].[Id].CurrentMember.Properties('IsIndex')</Formula>

</CalculatedMember>

<CalculatedMember name=*"Info IsPush"* dimension=*"Measures"* caption=*"%{is\_push}"*>

<Formula>[Info].[Id].CurrentMember.Properties('IsPush')</Formula>

</CalculatedMember>

</CalculatedMembers>

</Cube>

<!-- 资讯点击分类Cube -->

<Cube name=*"InfoCategory"* caption=*"%{info\_category}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

<Dimension name=*"Info"* source=*"Info"* caption=*"%{info}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_info"* name=*"%{base\_group}"*>

<Measures>

<Measure name=*"User Count"* column=*"deviceid"* aggregator=*"distinct-count"* formatString=*"#,###"* caption=*"%{user\_count}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"stat\_date"* />

<ForeignKeyLink dimension=*"Info"* foreignKeyColumn=*"id"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<CalculatedMember name=*"User Percent"* dimension=*"Measures"* formatString=*"Percent"* caption=*"%{user\_pct}"*>

<Formula>[Info].[Category].CurrentMember / [Info].[Category].CurrentMember.Parent</Formula>

</CalculatedMember>

</CalculatedMembers>

</Cube>

<!-- 资讯标签点击Cube -->

<Cube name=*"InfoTag"* caption=*"%{info\_tag}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

<Dimension name=*"InfoCategory"* source=*"InfoCategory"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_info\_cat"* name=*"%{base\_group}"*>

<Measures>

<Measure name=*"User Count"* column=*"sub\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{user\_count}"* />

<Measure name=*"User Percent"* column=*"user\_pct"* aggregator=*"sum"* formatString=*"Percent"* caption=*"%{user\_pct}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"stat\_date"* />

<ForeignKeyLink dimension=*"InfoCategory"* foreignKeyColumn=*"acc"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 活动Cube -->

<Cube name=*"Interact"* caption=*"%{interact}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

<Dimension name=*"StartDate"* source=*"Date"* caption=*"%{start\_date}"* />

<Dimension name=*"EndDate"* source=*"Date"* caption=*"%{end\_date}"* />

<Dimension name=*"Interact"* table=*"fact\_interact"* key=*"Id"* caption=*"%{interact}"*>

<Attributes>

<Attribute name=*"Id"* keyColumn=*"id"* caption=*"%{interact\_id}"*>

<Property attribute=*"Title"* />

</Attribute>

<Attribute name=*"Title"* keyColumn=*"title"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"fact\_interact"* name=*"%{base\_group}"*>

<Measures>

<!-- 点击次数 -->

<Measure name=*"Click Times"* column=*"click\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_times}"* />

<!-- 点击人数 -->

<Measure name=*"Click User Count"* column=*"click\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_users}"* />

<!-- 评论次数 -->

<Measure name=*"Comment Times"* column=*"comment\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{comment\_times}"* />

<!-- 评论人数 -->

<Measure name=*"Comment User Count"* column=*"comment\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{comment\_users}"* />

<!-- 分享次数 -->

<Measure name=*"Share Times"* column=*"share\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{share\_times}"* />

<!-- 分享人数 -->

<Measure name=*"Share User Count"* column=*"share\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{share\_users}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"stat\_date"* />

<ForeignKeyLink dimension=*"StartDate"* foreignKeyColumn=*"start\_date"* />

<ForeignKeyLink dimension=*"EndDate"* foreignKeyColumn=*"end\_date"* />

<FactLink dimension=*"Interact"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

<CalculatedMembers>

<CalculatedMember name=*"Interact Title"* dimension=*"Measures"* caption=*"%{title}"*>

<Formula>[Interact].[Id].CurrentMember.Properties('Title')</Formula>

</CalculatedMember>

</CalculatedMembers>

</Cube>

<!-- 明星Cube -->

<Cube name=*"Star"* caption=*"%{star}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

<Dimension name=*"Star"* table=*"stat\_star"* key=*"Id"* caption=*"%{star}"*>

<Attributes>

<Attribute name=*"Name"* keyColumn=*"real\_name"* caption=*"%{real\_name}"* />

<Attribute name=*"Id"* keyColumn=*"star\_id"* hasHierarchy=*"false"* />

</Attributes>

</Dimension>

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"stat\_star"* name=*"%{base\_group}"*>

<Measures>

<!-- 点击次数 -->

<Measure name=*"Click Times"* column=*"click\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_times}"* />

<!-- 点击人数 -->

<Measure name=*"Click User Count"* column=*"click\_user"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{click\_users}"* />

<!-- 粉丝数 -->

<Measure name=*"Fans Count"* column=*"fans\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{fans\_count}"* />

<!-- 动态数 -->

<Measure name=*"Dynamic Count"* column=*"dynamic\_num"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{dynamic\_count}"* />

<!-- 帖子数 -->

<Measure name=*"Post Count"* column=*"post\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{post\_count}"* />

<!-- 评论数 -->

<Measure name=*"Comment Count"* column=*"comment\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{comment\_count}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"create\_date"* />

<FactLink dimension=*"Star"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 搜索用户Cube -->

<Cube name=*"SearchUser"* caption=*"%{search\_user}"*>

<Dimensions>

<Dimension name=*"StatDate"* source=*"Date"* caption=*"%{stat\_date}"* />

</Dimensions>

<MeasureGroups>

<MeasureGroup table=*"stat\_search"* name=*"%{base\_group}"*>

<Measures>

<Measure name=*"Search User Count"* column=*"search\_count"* aggregator=*"sum"* formatString=*"#,###"* caption=*"%{search\_user\_cnt}"* />

<Measure name=*"Search User Percent"* column=*"search\_pct"* aggregator=*"sum"* formatString=*"Percent"* caption=*"%{search\_user\_pct}"* />

</Measures>

<DimensionLinks>

<ForeignKeyLink dimension=*"StatDate"* foreignKeyColumn=*"stat\_date"* />

</DimensionLinks>

</MeasureGroup>

</MeasureGroups>

</Cube>

<!-- 角色权限 -->

<Role name=*"ROLE\_ADOP"*>

<SchemaGrant access=*"none"* />

</Role>

<Role name=*"ROLE\_ADOPN"*>

<SchemaGrant access=*"none"* />

</Role>

</Schema>

昨日资讯点击创建人TopN

SELECT

NON EMPTY {

[Measures].[Click Times],

[Measures].[Click User Count],

[Measures].[Comment Times],

[Measures].[Comment User Count],

[Measures].[Share Times],

[Measures].[Share User Count],

[Measures].[Info Title],

[Measures].[Info Source],

[Measures].[Info IsIndex],

[Measures].[Info IsPush]

} ON COLUMNS,

NON EMPTY Order(

Generate(

[Info].[Creater].[Creater].Members,

TopCount(

[Info].[SrcPlatform].[SrcPlatform].Members \*

[Info].[Creater].[Creater].CurrentMember \*

[InfoCategory].[Name].[Name].Members \*

[Info].[Id].[Id].Members,

3,

[Measures].[Click User Count]

)

),

[Measures].[Click User Count],

DESC

) ON ROWS

FROM [Info]

WHERE (

[Info].[Type].[资讯],

StrToTuple("[PublishDate].[Daily].[" || Format(Now(), 'yyyy-mm-dd') || "]", [PublishDate].[Daily]).Lag(1),

StrToTuple("[StatDate].[Daily].[" || Format(Now(), 'yyyy-mm-dd') || "]", [StatDate].[Daily]).Lag(1)

)