ETL流程概述及常用实现方法

参考 ： http://blog.csdn.net/btkuangxp/article/details/48224187

Extract-Transform-Load ： 抽取-转换-加载
常见于数据仓库开发中将数据有业务系统归集到数据仓库或数据集市的过程。
花费时间最长的是T （transform清洗，转换）部分。

1. 抽取作业
从源数据库（常为业务系统）获得数据的过程
1.1 手工开发抽取作业常用方法 ：
1.1.1 数据源和数据仓库为同一类数据库
通过数据库链接功能
可以在数据源（业务系统）和DW内建立数据库链接（如DB2的联邦数据库NICKNAME），然后在DW内直接SELECT访问。
优点是实现使用简单，逻辑简单；
缺点是容易被滥用，对源数据库造成较大的负载压力。
1.1.2 数据源和ODS为不同类型数据库
1） 将源数据库的数据导出为文本文件，利用FTP协议进行传输导入ODS区域。
优点是实现简单，对源系统压力较小。缺点是传输步骤增加了，处理需要的时间增加。
2）部分数据库间能通过ODBC建立源数据库和目标数据库链接，此时也能直接使用SELECT获取数据。
优点是实现使用简单，逻辑简单；
缺点是容易被滥用对源数据库造成较大的负载压力，且建立时较为复杂。
1.2 更新数据的时间和数量问题
1.2.1 实时抽取数据
这类抽取方式在数据仓库中很少见到，因为一般来说数据仓库对数据的实时性要求并不高。实时抽取常见于BI中的CRM系统，比如在实时营销中，客户一旦进行了某类操作就实时触发对应的营销行为。
o 时间戳方式
要求源表中存在一个或多个字段(时间戳),其值随着新纪录的增加而不断增加，执行数据抽取时，程序定时循环检查通过时间戳对数据进行过滤，抽取结束后，程序记录时间戳信息。
这种方式的优点是对源系统的侵入较小，缺点是抽取程序需要不断扫描源系统的表，对其 有一定压力。
o 触发器方式
要求用户在源数据库中有创建触发器和临时表的权限，触发器捕获新增的数据到临时表中，执行抽取时，程序自动从临时表中读取数据。
这种方式的优点是实时性极高，缺点是对源系统的侵入性较大，同时会对源数据库造成很大的压力（行级触发器），很可能影响源系统的正常业务。
o 程序接口方式
改造源系统，在修改数据时通过程序接口同步发送数据至目标库，发送数据的动作可以跟业务修改数据动作脱耦，独立发送。
这种方法的优点是对源系统的造成压力较小，实时性较强；缺点是需要对源系统的侵入性较强，需要源系统做较大的改造。
1.2.2 批量抽取数据
为了保证数据抽取时数据的准确性、完整性和唯一性，同时降低抽取作业对源数据库造成的压力，抽取作业的加载必须避开源数据的生成时间。这种方法一般用于实时性要求不高的数据。比如T+1或者每月1日进行抽取。
1.2.2.1 常用实现
o 日志检查
需要源数据库生成数据完毕之后，在外部生成日志。抽取程序定时检查源系统的执行日志，发现完成标志后发起抽取作业。
这种方式优点是可靠性高，对源数据库造成的压力较小。缺点是需要源数据库配合生成可供检查的外部日志。
o 约定时间抽取
可以直接约定一个加载完毕同时对源数据库压力较小的时间（如每日凌晨2点），抽取程序建立定时任务，时间一到自动发起抽取作业。
这种方式优点是对源数据库的侵入性和造成的压力较小；缺点是可靠性不高，可能会发生数据未生成完毕也直接进行抽取的情况。
1.2.2.2根据下载时候对数据的筛选方式可以分为
o 全量下载
用于：
• 源数据量较小，如维表。
• 数据变化较大，比如90%的数据都产生了变化的表。
• 变化的数据不能预期，无法标示，如账户表。
的时候。
优点在于下载较为简单且能容纳任何情况的数据变化；缺点是如果数据量较大，需要抽取相当长的时间，同时会占用大量的IO和网络资源。
o 增量下载
常用于数据只增不减的表，如交易明细表等。

・时间戳

源系统在修改或添加数据时更新对应的时间戳字段（如交易表的日期字段），抽取程序根据时间戳选择需要更新的数据进行抽取。

・触发器方式

要求用户在源数据库中有创建触发器和临时表的权限，触发器捕获新增的数据到临时表中，到执行抽取的时间时，程序自动从临时表中读取数据。

占用资源较多，不建议使用。

优点是下载的数据较小，速度较快，占用资源少；缺点是使用限制较大，有时候需要源系统进行改造支持。

2. 转换作业

包含了数据清洗和转换

2.1 数据清洗

任务是过滤不符合条件或者错误的数据。

这一步常常出现在刚刚开始建立数据仓库或者源业务系统仍未成熟的时候，此时发现错误数据需要联系源业务系统进行更正，部分可预期的空值或者测试用数据可以过滤掉。

2.2数据转换

这一步是整个ETL流程中最为占用时间和资源的一步。

数据转换包含了简单的数据不一致转换，数据粒度转换和耗时的数据关联整合或拆分动作。这里可能存在各种各样千奇百怪的需求。对于核心数据仓库来说，里面往往是对数据进行按照主题划分合并的动作。同时，也会添加一些为了提升执行效率而进行反范式化添加的冗余字段。

根据实现方式的不同，可以区分为使用数据库存储过程转换和使用高级语言转换

o 使用数据库存储过程转换

使用SQL开发存储过程完成转换作业是很多银行常用的方法。

它的优点是开发简单、能支持绝大部分转换场景；缺点在于占用资源多且受制于单一数据库性能，无法做到横向扩展。

因此，除了业务的理解能力外，对SQL海量数据处理的优化能力在此也非常重要。比如:

- 利用数据库的分区性，选择良好的分区键。
- 建表时合理选择主键和索引，关联时候必须使用主键或索引进行关联。
- 关注数据库对SQL的流程优化逻辑，尽量选择拆分复杂SQL，引导数据库根据你选择流程进行数据处理
- 合理反范式化设计表，留出适当的冗余字段，减少关联动作。

具体的优化根据不同的数据库有着不同的处理方式，根据所选用的数据库不同而定。

o 使用高级语言转换

使用高级语言包含了常用的开发C/C++/JAVA等程序对抽取的数据进行预处理。

自行使用高级语言开发的优点是运行效率较高，可以通过横向扩展服务器数量来提高系统的转换作业处理能力；缺点是开发较为复杂，同时虽然能进行较为复杂的逻辑的开发，但是对于大数据量的关联的支持能力较弱，特别是有复数的服务器并行处理的时候。

3加载作业

转换作业生成的数据有可能直接插入目标数据库，一般来说，这种情况常见于使用数据库存储过程进行转换作业的方案。此时，ETL作业位于目标数据库上，加载作业只需要使用INSERT或者LOAD的方式导入目标表即可。此时转换作业和加载作业往往是在同一加工中完成的。

当使用高级语言开发时，ETL作业有着专门的ETL服务器，此时，转换作业生成的往往是文本文件，在转换作业完成后需要使用目标库特有的工具导入或者通过INSERT入目标库。

同时，根据抽取作业的数据抽取方式的不同（全量、增量），对目标表进行替换或者插入动作。

4流程控制

抽取加载和转换作业需要一个集中的调度平台控制他们的运行，决定执行顺序，进行错误捕捉和处理。

较为原始的ETL系统就是使用CRON做定时控制，定时调起相应的程序或者存储过程。但是这种方式过于原始，只能进行简单的调起动作，无法实现流程依赖行为，同时按步执行的流程控制能力也弱，错误处理能力几乎没有。只适合于极其简单的情况。

对于自行开发的较为完善的ETL系统，往往需要具有以下几个能力：

- 流程步骤控制能力

调度平台必须能够控制整个ETL流程（抽取加载和转换作业），进行集中化管理，不能有流程游离于系统外部。

- 系统的划分和前后流程的依赖

由于整个ETL系统里面可能跨越数十个业务系统，开发人员有数十拨人，必须支持按照业务系统对ETL流程进行划分管理的能力。

同时必须具有根据流程依赖进行调度的能力，使得适当的流程能在适当的时间调起。

- 合理的调度算法

同一时间调起过多流程可能造成对源数据库和ETL服务器还有目标数据库形成较大负载压力，故必须有较为合理的调度算法。

- 日志和警告系统

必须对每一步的流程记录日志，起始时间，完成时间，错误原因等，方便ETL流程开发人员检查错误。对于发生错误的流程，能及时通知错误人员进行错误检查和修复。

- 较高可靠性

5常用商业ETL工具

常用的ETL工具有Ascential公司的Datastage、Informatica公司的Powercenter、 NCR Teradata公司的ETL Automation等。

- Datastage

是使用高级语言进行开发ETL服务器的代表。使用JAVA进行开发E/T/L的整个流程，同时支持平行添加服务器提升处理效率的方法。

- Automation

基于Teradata的TD数据库的ETL调度框架。其ETL流程是使用DSQL的存储过程进行开发，利用TD数据库的海量数据处理能力，也具有一定的平行扩展能力。