数据仓库的设计（一）

该系列文章是阅读Kimball的《The Data Warehouse Toolkit 3rd Edition》的笔记，做了一个整理。

Kimball的这本大作，是数据仓库领域的经典。从书中可以学习到很多关于数据仓库设计的道与术。虽然现在都在往大数据NoSQL方向挤，但NoSQL也是Not only SQL,可见，SQL所代表的关系型数据库，依然是整个数据世界的基石，而数据仓库的相关技术，也可以为我们进行大数据的结构设计时提供参考。

本书主要围绕着星型模型（Star Schema）的设计，结合各领域的实际应用，讨论了维度表(Dimensional Table)与事实表(Fact Table)的设计方法。

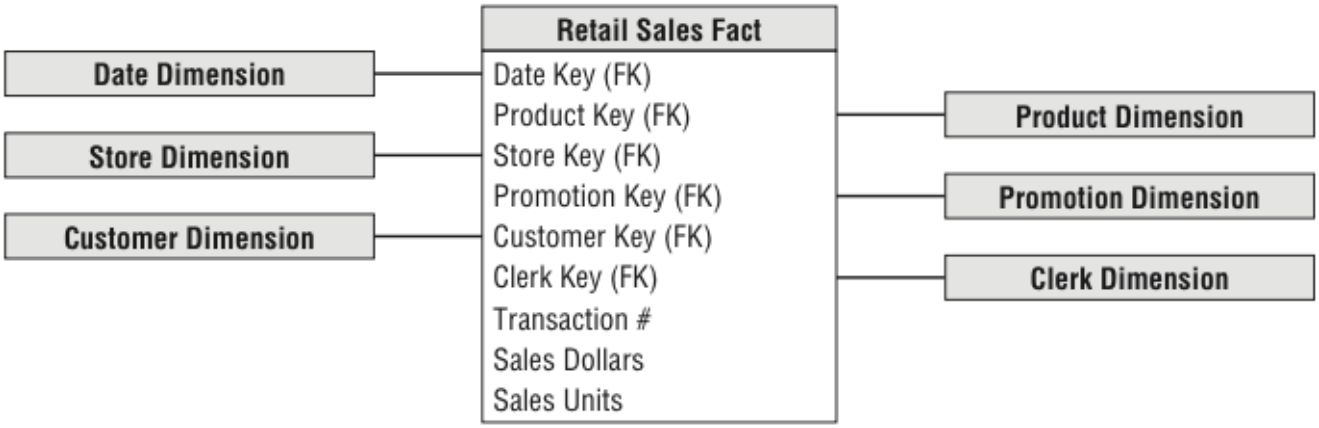
事实表

事实表是用来度量的。
事实表里存储的是业务处理的记录和需要进行度量的结果。事实表中的每一条记录都代表着一个度量（measurement）的事件。
事实表一般会有两个以上的外键，这些外键做为其它维度表的主键。

维度表

维度表用来描述内容（Descriptive Context），它用来说明事实表中度量值的各个维度的属性。
数据仓库的设计，很大程度上取决于维度及其属性的设计。

维度表与事实表组合在一起，形成了星型模型，如下图：



星型模型

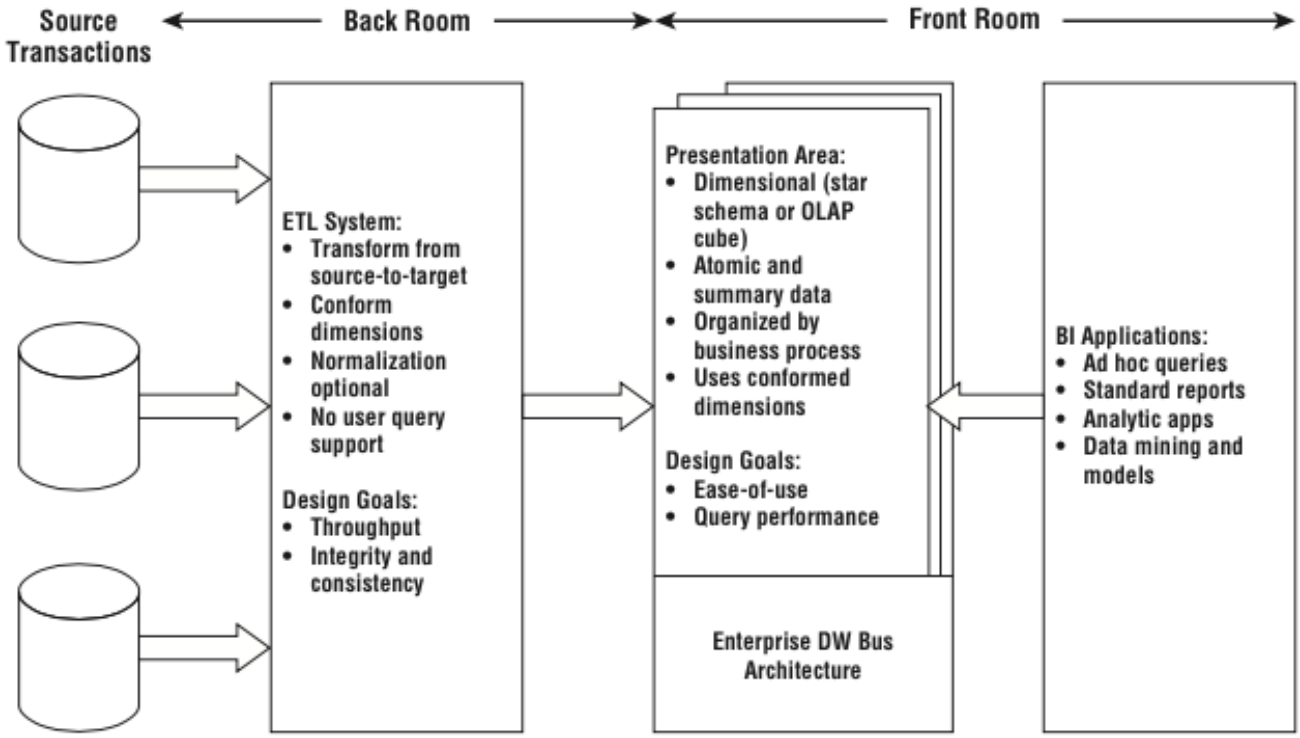
对其进行查询操作的SQL语句如下：

```
SELECT
    store.district_name,
    product.brand,
    sum(sales_facts.sales_dollars) AS "Sales Dollars"
FROM store,
    product,
    date,
    sales_facts
WHERE
    date.month_name="January" AND
    date.year=2013 AND
    store.store_key = sales_facts.store_key AND
    product.product_key = sales_facts.product_key AND
    date.date_key = sales_facts.date_key
GROUP BY
    store.district_name,
    product.brand
```

上面语句中，FROM的是相关维度表与事实表，WHERE的条件是各维度的属性过滤条件以及事实表与维度表的绑定，GROUPBY是想要统计的维度属性，最终sum出事实表中的度量值。

Kimball的DW/BI架构

Kimball提出的DW/BI架构，由四部分组成，大类上分为Back room和Front room。详见下图：



DW/BI架构图

业务源系统(Opertional source system)

这里指业务系统，它其实是在数据仓库之外的业务系统，比如企业的销售系统、客户管理系统等。这些系统的数据结构都由系统本身来决定，我们不能对其进行修改，只能将数据照搬过来。

ETL系统(Extract, Transformation, and Load System)

Extract即抽取，是将业务系统的数据抽取到ETL系统中来；
Transformation是转换，从各系统进来的数据千奇百怪，我们需要对数据进行各种清洗、组合、去重等操作；
Load是将转换后的数据加载进目标的维度模型，即更新维度表和事实表。

展现区域(Presentation Area)

DW/BI的展现区域，是数据经过组织后所存储的地方，它可以直接供用户进行查询操作。

BI应用(BI Application)

BI应用通过上面的展现区域查询数据，同时它又可需要支持即席查询（ad hoc）甚至提供数据挖

掘或建模的功能。

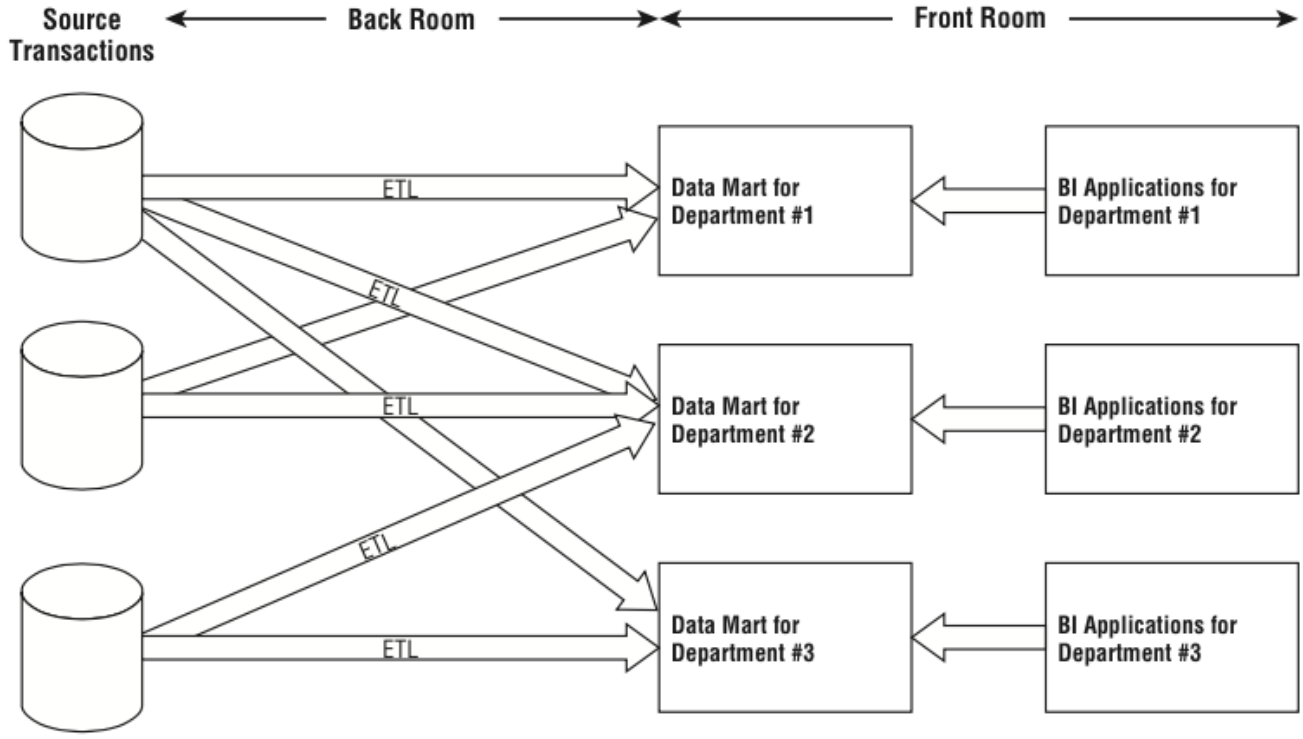
一个餐馆的隐喻

Kimball将DW/BI架构比喻成了一个餐馆，ETL系统就是厨房，他们将来自外部的采购进行各种加工处理，而展现区域和BI应用就是餐馆的大堂，厨房做好的菜都将来到大堂，展现到每位顾客的面前，而顾客们也是在大堂里吃到食物、接受到餐饮的服务等。

其它的DW/BI架构介绍

独立数据集市架构

独立数据集市的架构(Independent Data Mart Architecture)，这种一般在企业的各个部门间会出现。某个部门需要进行数据分析的工作，于是就从业务系统中进行ETL操作，建立了一个仅供部门自己使用的数据集市。如果同时还有别的部门也有同样的想法，那么情况就会如下图所示：

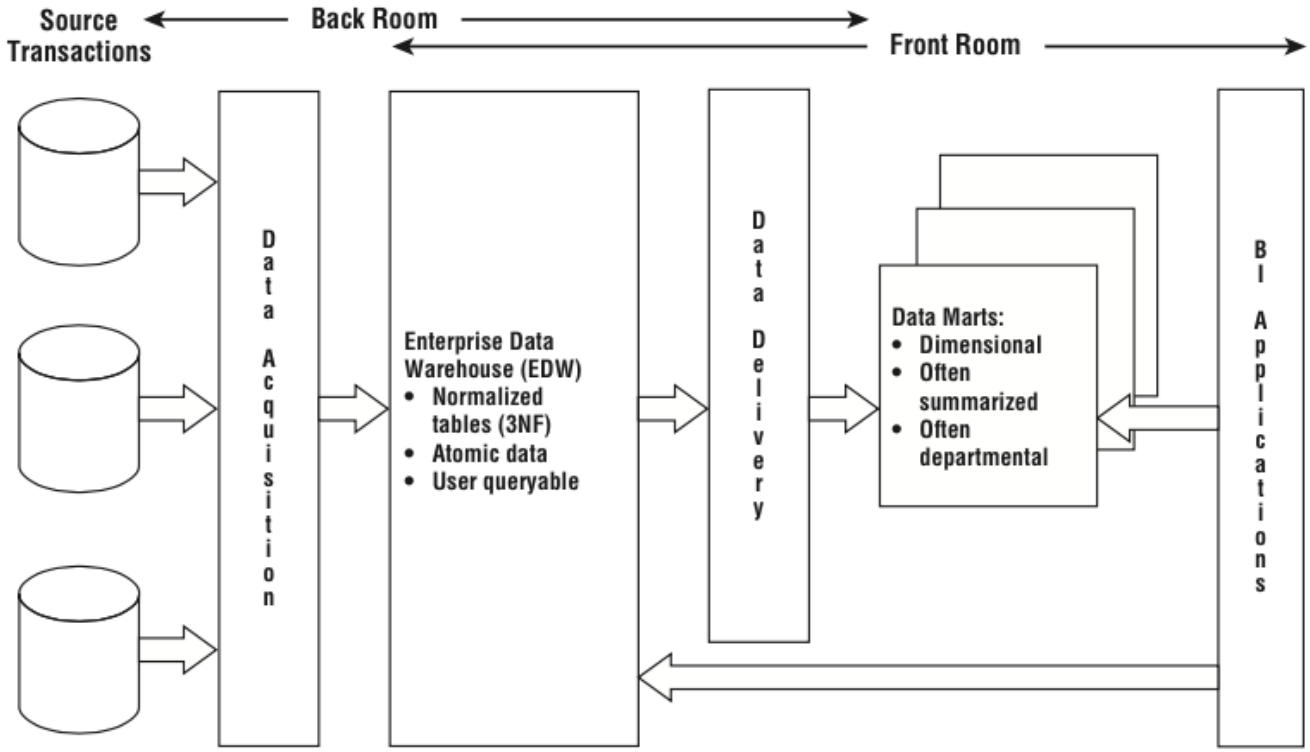


独立数据集市架构

由于业务规则和处理方式的不统一，结果会造成各部门的报表上的数据对不上。Kimball也不建议使用这种方式。

CIF架构

Hub-and-Spoke Corporate Information Factory Inmon Architecture，它是由另一位数据仓库教父Inmon提出的。他提出在企业层面上建立起一个企业数据仓库(EDW)。

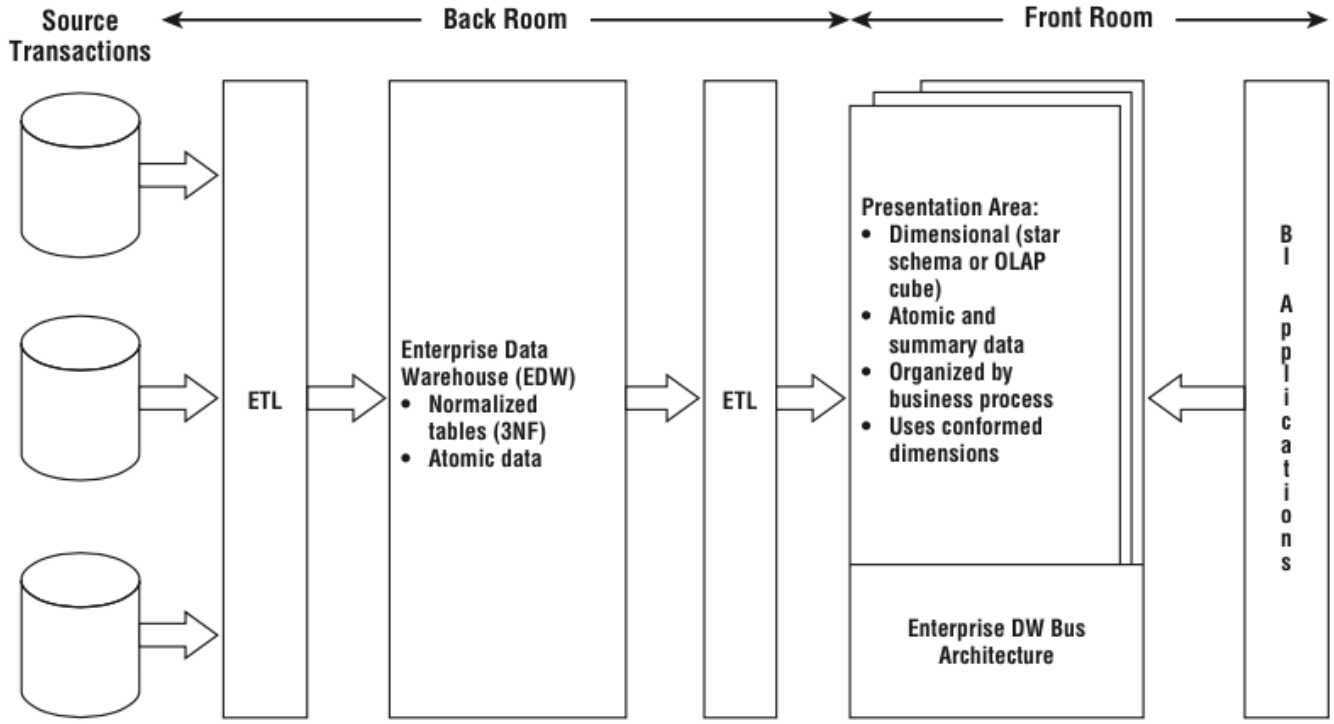


CIF架构

与Kimball不同的是，Inmon要求EDW的数据是符合3NF规范化的，而Kimball则提出了遵照维度设计的企业总线(Enterprise Bus)的形式，维度的设计却是去规范化的，里面经常有大量的数据冗余。

超越的架构

这种架构是将Inmon和Kimball的架构进行了结合，采用了Inmon的EDW概念，但同时去除了DM，而是将Kimball的维度模型OLAP加了进来。



Hybrid架构

不过Kimball仍在书中讽刺地写道，这种架构会花费更多的时间和金钱，如果你真的不差钱，那么还是可以去做的：）