



2014中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2014



大数据技术探索和价值发现

数据库设计模式变迁

高峡

2014-04-12



Who Am I?



- MS in Computer Science
- IBM Singapore
- 全球顶级市场调研公司
- 标签：
 - 数据库 多维数据集 商务智能 数据挖掘 NOSQL
 - 大数据 商业模式 数据库设计模式
 - 应用场景



- 新浪微博：@高峡之数据时代
- 高峡出平湖，神女应无恙！ --- 毛泽东

Agenda(1)

- 从范式到反范式
 - Pattern
 - Anti-Pattern
- 行列互转
 - 行列数据库
 - 行列互转 (Pivoting)
- 从行式数据库转为列式数据库
 - RDBMS
 - Sybase IQ
- 从数据库到数据仓库
 - 面向业务
 - 面向主题

Agenda (2)

- 从正向集合到反向集合
 - 正向集合
 - 反向集合 (Inverted File Index)
- 从二维数据库到多维数据集 (以空间换时间)
 - RDBMS (二维数据库)
 - OLAP (多维数据集)
- 从Schema->Flexible Schema->No Schema
 - DBMS
 - EAV
 - MongoDB

Agenda(3)

- 从结构化数据到半结构化数据，再到非结构化数据
 - RDBMS
 - NOSQL
 - MPP
- 进入大数据时代之前，Review在关系型数据库里面积累的所有的经验

Agenda(4)

- 数据库设计模式的持续演进
- 数据库设计模式 ----》 商业模式

商业场景和商业价值



手中只有锤子，
看到的都是钉子！

1. 集合思维

- 光标
- 序列表的引入
- SET （关系型数据库的精华！）

场景

- 需要把一个有分隔符的字符串转化为TABLE

String_to_table

```
ALTER FUNCTION [dbo].[string_to_table]
    (@param varchar(max),
    @delimiter varchar(100))

RETURNS TABLE AS
    RETURN(
        SELECT substring(@delimiter + @param + @delimiter, Number + 1,
            charindex(@delimiter, @delimiter + @param + @delimiter, Number
+ 1) - Number - 1)
            AS Value
        FROM Numbers
        WHERE Number <= len(@delimiter + @param + @delimiter) - 1
            AND substring(@delimiter + @param + @delimiter, Number, 1) =
@delimiter)
```

调用:

```
select * from dbo.string_to_table('gaoxia loves database', '')
```

建议:

- 避免光标式的程序员思维
- 拓宽数据库的SET集合思维

思考:

- 为什么Hadoop分析平台上面需要SQL?

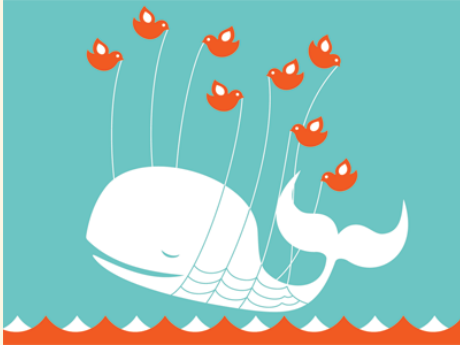
2. 范式设计

- 1NF
- 2NF
- 3NF
- 4NF
- 5NF

反范式设计 Anti-Pattern

- 增加冗余列
- 增加派生列
- 分割
 - 水平分割
 - 垂直分割
- 维护数据的完整性
 - 批处理
 - 应用逻辑
 - 触发器

案例: Twitter



- Twitter, which launched its service in 2006, is expanding at an amazing pace.
- According to official *About Twitter* page, as of September 2010 some **175 million users** use the service, generating about **95 million tweets a day**. This means about **1100 new tweets appear every second**. At **peak times**, over **3,000 tweets are generated per second**. More than **600 million searches are done in a day**.

场景:

- Tweets
- Timeline
- Social Graph
- Search Indices

Tweets

id	user_id	text	created_at
20	12	just setting up my twitter	2006-03-21 20:50:14
29	12	inviting coworkers	2006-03-21 21:02:56
34	16	Oh shit, I just twittered a little.	2006-03-21 21:08:09

In this situation, there are **two types** of easily applicable partitioning methods.

- 1.The user id-based partitioning
- 2.The tweet id-based partitioning

Partition: 分区

- 数据应用系统中，80%的时间是在处理20%的数据，20%的时间在处理另外80%的数据。
- 在设计时将主要精力关注在20%的数据上，通过对这20%的数据建立独立的分区 的数据建立独立的分区，建立索引等方式 建立索引等方式，提高其性能。
- 80-20法则意味着切割，对数据要进行切割，周期工作要切割，用户功能要切割，甚至硬件使用方式、资源的分配都可能进行调整。

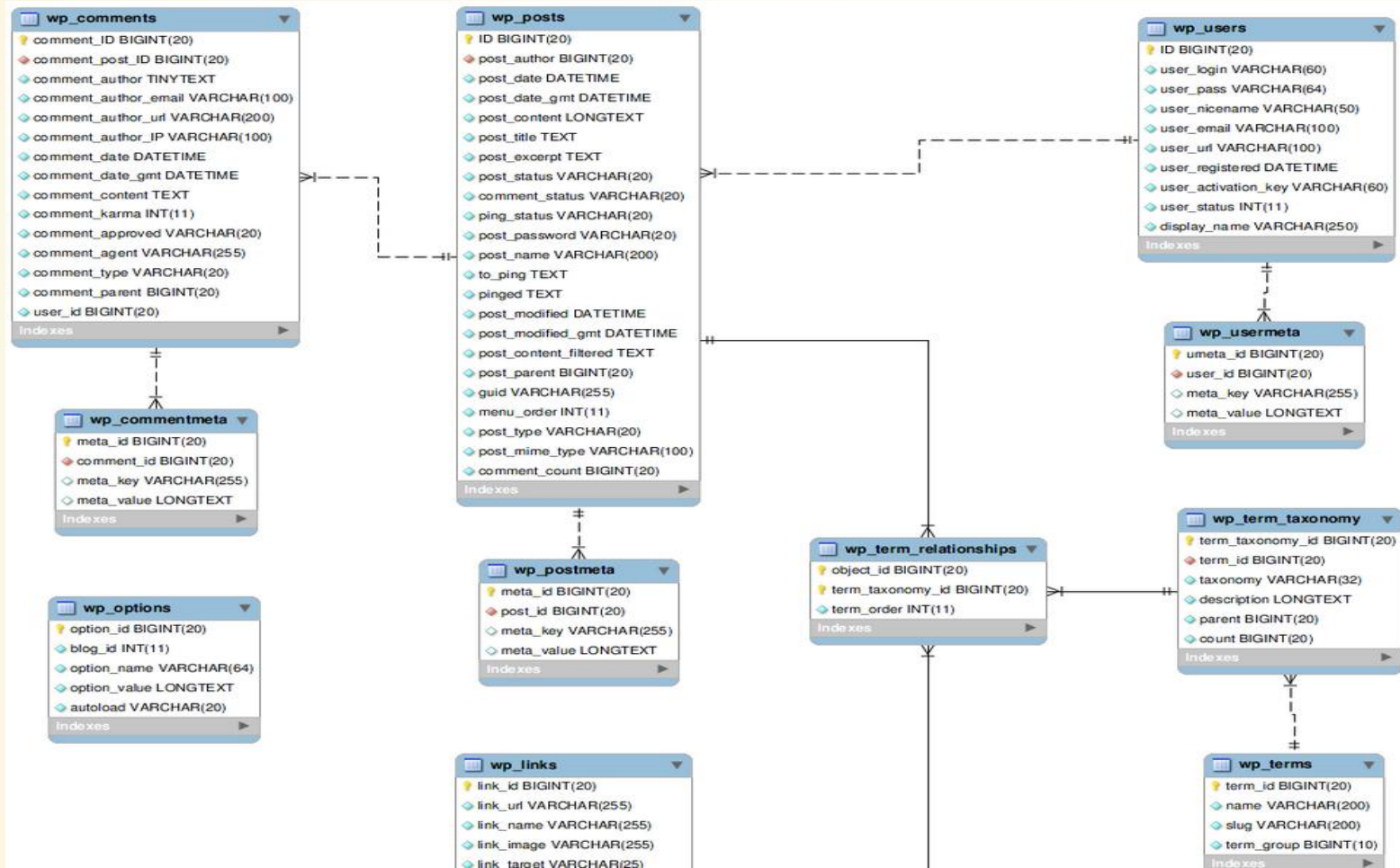
3. KV存储

- Key-Value 存储

案例: WordPress

- WordPress是一个注重美学、易用性和网络标准的个人信息发布平台。WordPress虽为免费的开源软件，但其价值无法用金钱来衡量。
- 使用WordPress可以搭建功能强大的网络信息发布平台，但更多的是应用于个性化的博客。针对博客的应用，WordPress能让您省却对后台技术的担心，集中精力做好网站的内容。

架构



设计

- **wp_users**

ID: 自增唯一ID

user_login: 登录名

user_pass: 密码

user_nicename: 昵称

user_email: Email

user_url: 网址

user_registered: 注册时间

user_activation_key: 激活码

user_status: 用户状态

display_name: 显示名称

wp_usermeta

umeta_id: 自增唯一ID

user_id: 对应用户ID

meta_key: 键名

meta_value: 键值

4. EAV

- Entiry-Attribute-Value

案例: Magento

- Magento 是一款新的专业开源电子商务平台，Magento电子商务平台采用php进行开发，使用Zend Framework框架。Magento设计得非常灵活，具有模块化架构体系和丰富的功能。易于与第三方应用系统无缝集成。在设计上，包含相当全面，以模块化架构体系，让应用组合变得相当灵活，功能也相当丰富。
- 被EBAY收购



功能

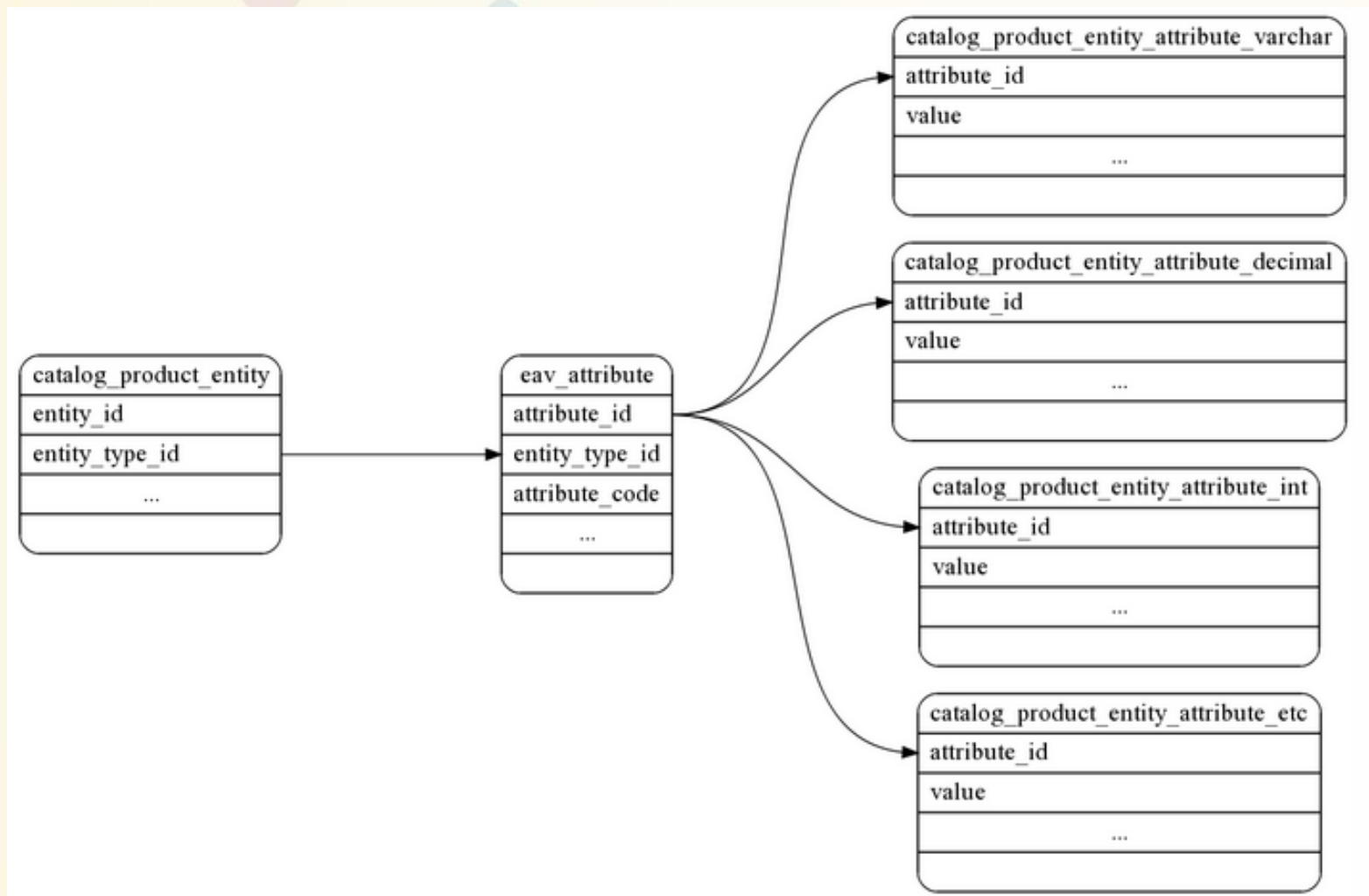


- 1.强大的商品属性组合
- 2.购物车价格规则
- 3.灵活的模板系统
- 4.多网店系统
- 5.完善的插件体系
- 6.安全加密
- 7.企业应用集成

场景

- 异构的产品种类和内容
- Schema 不要变化

架构



Entity

EntityType	EntityID	EntityName
计算机	1	DELL-100
计算机	2	IBM RISC 6000
运动鞋	3	360跑鞋
运动鞋	4	360休闲鞋
.....

Attribute

EntityType	AttributeID	AttributeName
计算机	1	CPU
计算机	2	内存
计算机	3	磁盘
计算机	4	显卡
运动鞋	1	款式
运动鞋	2	尺码
运动鞋	3	大小
.....

Value

EntityID	EntityType	Attribute	Value
DELL 100	计算机	CPU	Intel I5
DELL 100	计算机	内存	8G
DELL 100	计算机	磁盘	1T
DELL 100	计算机	显卡	SGI
360跑鞋	运动鞋	款式	运动
360跑鞋	运动鞋	尺码	42
360跑鞋	运动鞋	颜色	蓝色
.....

Query Pattern 查询模式

```
SELECT `main_table`.*,  
IFNULL(al.value, main_table.frontend_label) AS `store_label`  
FROM `eav_attribute` AS `main_table`  
INNER JOIN `catalog_eav_attribute` AS `additional_table`  
ON main_table.attribute_id = additional_table.attribute_id  
LEFT JOIN `eav_attribute_label` AS `al`  
ON al.attribute_id = main_table.attribute_id AND al.store_id = 1  
WHERE (main_table.entity_type_id='4')  
AND (additional_table.used_in_product_listing=1)
```

Pros and Cons

- Pros:

- 灵活的EAV架构支持异构商品
- 不用改动Schema
- 解决了SPARSE问题

- Cons:

- 复杂的查询
- 性能调优复杂

5. FTS – Full Text Search

- 开源的FTS引擎
 - Lucene
 - Sphinx
- FTS的数据库实现
 - Mysql
 - SQL Server
- Information Retrieval

场景:

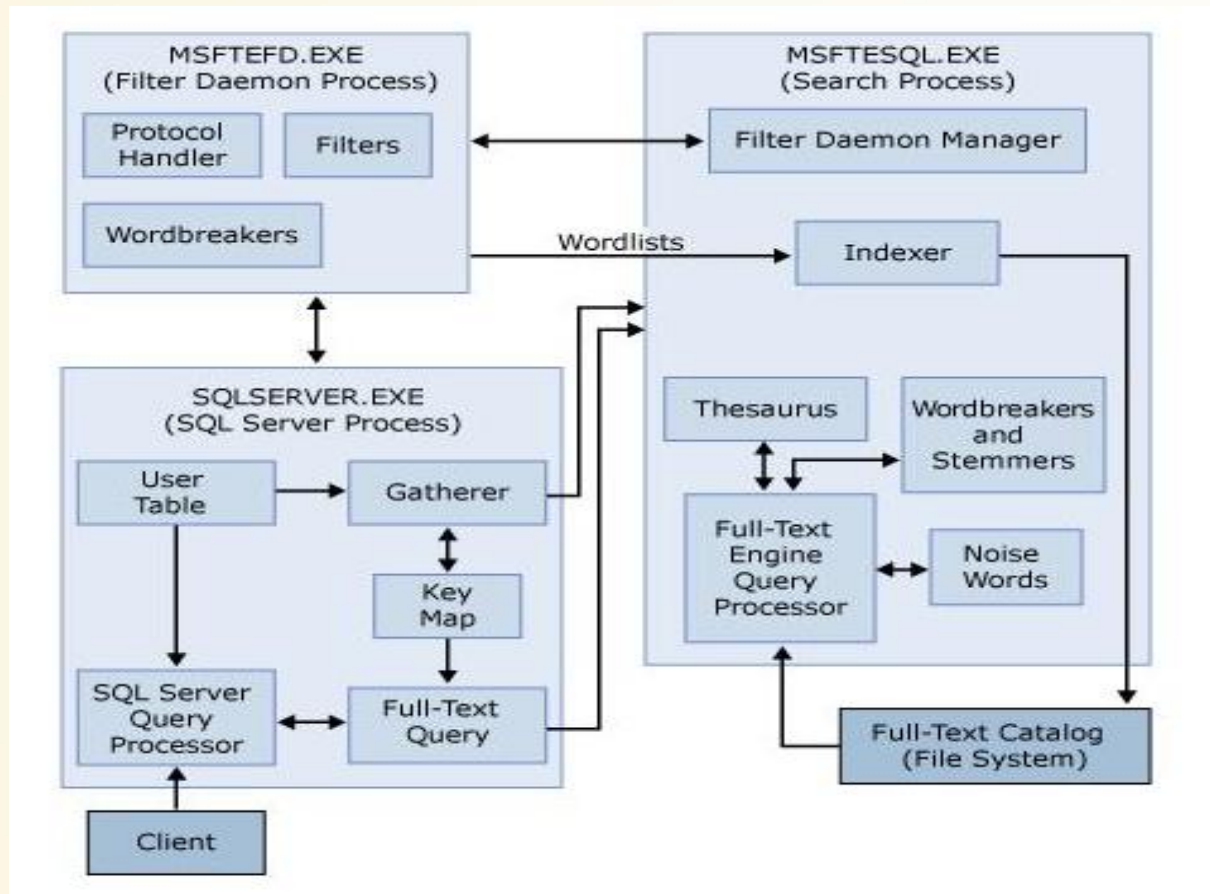
- Tag搜索
- Skill Set 搜索
 - LinkedIn

Inverted File Index – 倒排文件索引

doc		Term		
1	--->	product		
2	--->	product	windows	oracle
3	--->	windows	oracle	
4	--->	product	chognqing	oracle
5	--->	windows		
6	--->	...		
7	--->		

Term		doc						
product	--->	1	2	3	4			
windows	--->	2	3	5	6	8		
chognqing	--->	4	5	7	8			
oracle	--->	4	3	2	1	10	11	
sybase	--->	7	8	0	1	12	20	

案例: SQL Server Full Text Search



Keyword

	display_term	column_id	document_count
22	acceptable	6	1
23	accessories	6	1
24	add	6	1
25	adding	6	1
26	adjust	6	1
27	adjustable	6	2
28	adjusted	6	1
29	adjusting	5	1
30	adjusting	6	1
31	adjustments	6	1
32	adventure	5	4
33	adventure	6	7
34	again	6	1
35	air	6	1
36	alignment	6	1

```
SELECT display_term
       , column_id
       , document_count
FROM sys.dm_fts_index_keywords
     (DB_ID('AdventureWorks'),
      OBJECT_ID('ProductDocs'))
```

Keywords_by_document

	keyword	display_term	column_id	document_id	occurrence_count
19	0x003600310039	619	6	1	1
20	0x003600320030	620	6	1	1
21	0x00610062006F00760065	above	6	6	1
22	0x00610062006F00760065	above	6	8	1
23	0x006100620073006F0...	absorbing	6	7	1
24	0x00610063006300650...	acceptable	6	1	1
25	0x00610063006300650...	accessories	6	5	1
26	0x006100640064	add	6	1	1
27	0x00610064006400690...	adding	6	1	1
28	0x00610064006A00750...	adjust	6	3	1
29	0x00610064006A00750...	adjustable	6	5	2
30	0x00610064006A00750...	adjustable	6	7	1
31	0x00610064006A00750...	adjusted	6	7	1
32	0x00610064006A00750...	adjusting	5	7	1
33	0x00610064006A00750...	adjusting	6	7	2
34	0x00610064006A00750...	adjustments	6	7	3

```
select *  
from  
sys.dm_fts_index_keywords_by_document  
t  
  (DB_ID('AdventureWorks'),  
  OBJECT_ID('ProductDocs'))
```

Query Pattern 查询

```
select * from Sample  
where [Column] like '%重庆大数据%'
```

```
select [Column] as [result] from Sample  
where contains([Column], '重庆大数据')
```

结论:

- FTS性能提高百倍以上
- 自动维护Keyword, Document
- SET集合的运算
- Inverted File Index的逆袭

6. Pivoting 行列转换

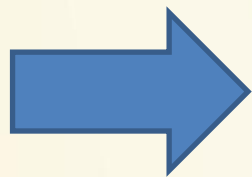
- 数据库论坛问得最多的问题
- 数据库行列本身的限制

解决Pivoting问题

- 在二维数据库里面解决Pivoting

案例：财务科目的旋转

companyid	accountid	balance
1	1121	100.00
1	1122	200.00
1	1123	300.00
1	1124	400.00
1	1125	500.00



companyid	1121	1122	1123	1124	1125
1	100.00	200.00	300.00	400.00	500.00

Pivoting: SQL

```
select companyid  
      ,[1121],[1122],[1123],[1124],[1125]  
from factaccount  
pivot  
(  
    max(balance)  
    for accountid in ([1121], [1122],[1123] ,[1124], [1125]  
) x
```

7. XML

- Impedence Mismatch 阻抗失效
- Multiple Result Set 多结果集的串接

XML 介绍

```
declare @xml xml
set @xml =
' <RecordStore>
    <Album id="1" category="Rock">
        <Artist>Rockers Utd.</Artist>
        <Title>Rock Until You Drop</Title>
        <ReleaseYear>2005</ReleaseYear>
        <Price>10</Price>
    </Album>
    <Album id="2" category="Oldies">
        <Artist>Oldies Inc.</Artist>
        <Title>Rock Like There Was A Tomorrow</Title>
        <ReleaseYear>1960</ReleaseYear>
        <Price>5</Price>
    </Album>
</RecordStore>'
```

XQuery

-- Get all album titles in Category = "Rock"

```
select @xml.query('/RecordStore/Album[@category="Rock"]/Title')
```

-- Get all artists that released albums in 2005

```
select @xml.query('/RecordStore/Album[ReleaseYear=2005]/Artist')
```

-- Get the price for the Album called ShockRock

```
select @xml.value('/RecordStore/Album[Title="ShockRock"]/ReleaseYear)[1]',  
          'int')
```

-- Get the title for the album with id 4

```
select @xml.value('/RecordStore/Album[@id="4"]/Title)[1]', 'varchar(100)')
```

案例: RightNow SAAS CRM



- 顶级SAAS CRM供应商
- 甲骨文**15亿美元**收购云客户服务提供商 RightNow

案例：场景

- 不同的用户有不同的客户字段定义

- <FirstName>Xia</FirstName>
- <LastName>Gao</LastName>
- <Education>Master</Education>
- <Gender>Male</Gender>
- <Marriage>Married</Marriage>
- <Address>
 - <Street>沙坪坝</Street>
 - <State>重庆</State>
 - <Zip>400030</Zip>
- </Address>
- <Phone>
 - <Business>123456789</Business>
 - <Family>987654321</Family>
- </Phone>

场景

- 支持自定义客户属性
- 快速查询
 - 属性的组合查询

实现方式

- XML实现
- 支持XML index
 - Path
 - Value
 - Property

8. 列式数据库



行式 VS 列式

行式	列式
数据与索引分离	数据与索引一体
数据几乎不压缩	数据压缩
操作某列必须读出整行	能直接读取某列数据

Sybase IQ



- Sybase IQ 排名列式数据库第一
- IQ通过列存储、革命性的位图索引方法以及智能的动态访问技术实现了快速的查询响应速度，比传统的数据库查询速度提高**10-1000倍**
- **减少磁盘I/O** IQ通过独特的列存储，索引与压缩技术，大大减少了查询中的磁盘I/O次数，其杰出的磁盘I/O效果带来了更快速的查询反应，更高的吞吐量和更低的成本。
- 被SAP以**58亿美元**收购

案例: Panel



- 精准营销
- Customer Segmentation 客户分群
- 微博为什么不盈利?
 - 缺乏结构化的人群标签数据
 - 无法做精准营销

我知道我的广告浪费了一半，
但我不知道浪费了哪一半。

盈利模式

- 市场调研
 - 微软
 - 沃尔玛
 - 谷歌
 - Motorola
- 300万的一个Panel到底应该值多少钱？

场景

- 300万行的客户数据
- 几千个客户属性
- 需要根据任何客户属性的组合进行查询
- 行式数据库的局限：
 - 索引的限制（256）

解决方案

- 从行式数据库转化为列式数据库
- 查询时间提高50倍
- 压缩率高达1:30

9. 数据仓库

- 定义:

- 数据仓库，由数据仓库之父比尔·恩门（**Bill Inmon**）于1990年提出，主要功能仍是将组织透过资讯系统之联机事务处理(OLTP)经年累月所累积的大量资料，透过数据仓库理论所特有的资料储存架构，作一有系统的分析整理，以利各种分析方法如联机分析处理(OLAP)、数据挖掘(Data Mining)之进行，并进而支持如决策支持系统(DSS)、主管资讯系统(EIS)之创建，帮助决策者能快速有效的自大量资料中，分析出有价值的资讯，以利决策拟定及快速回应外在环境变动，帮助建构商业智能(BI)。

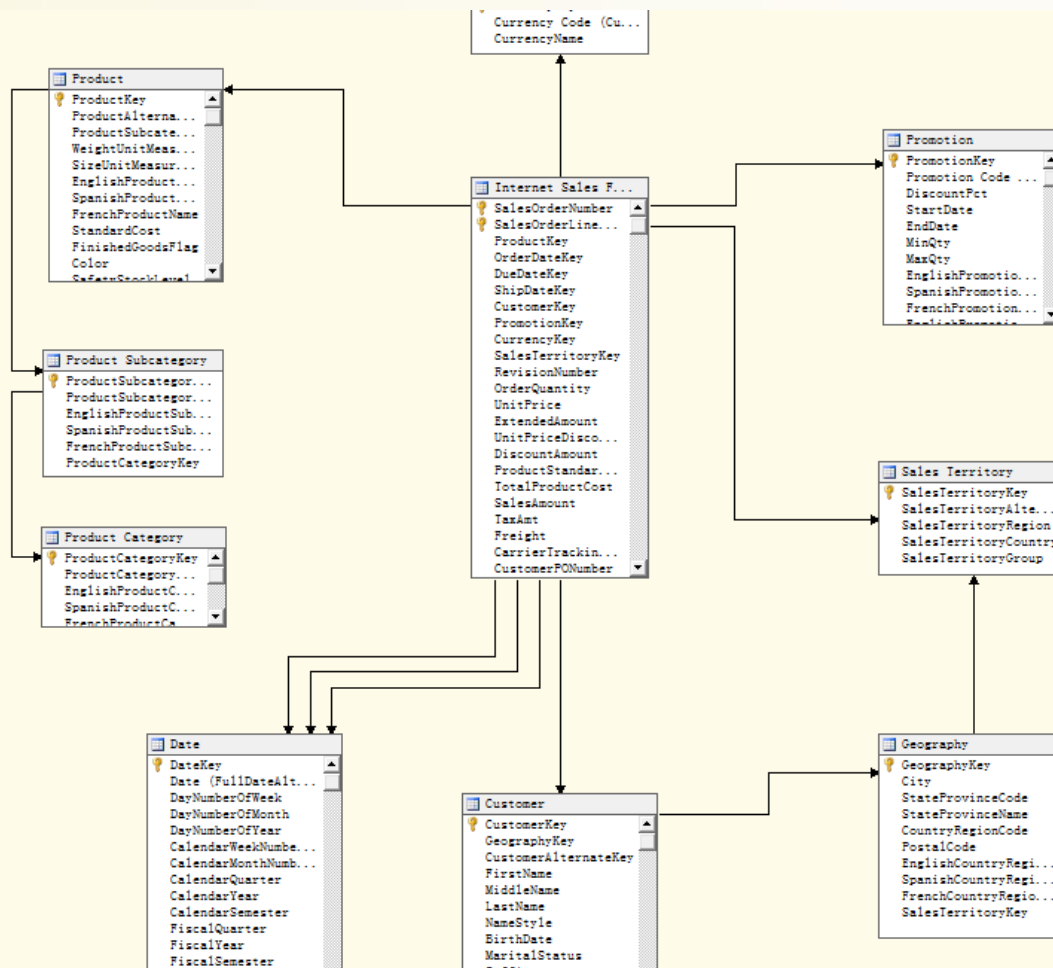
- 特征:

- 面向主题
- 集成
- 时变
- 不可更新

Query Pattern

- 并发
- 粒度
- 面向主题
- 集成
- 不易失

Star Schema



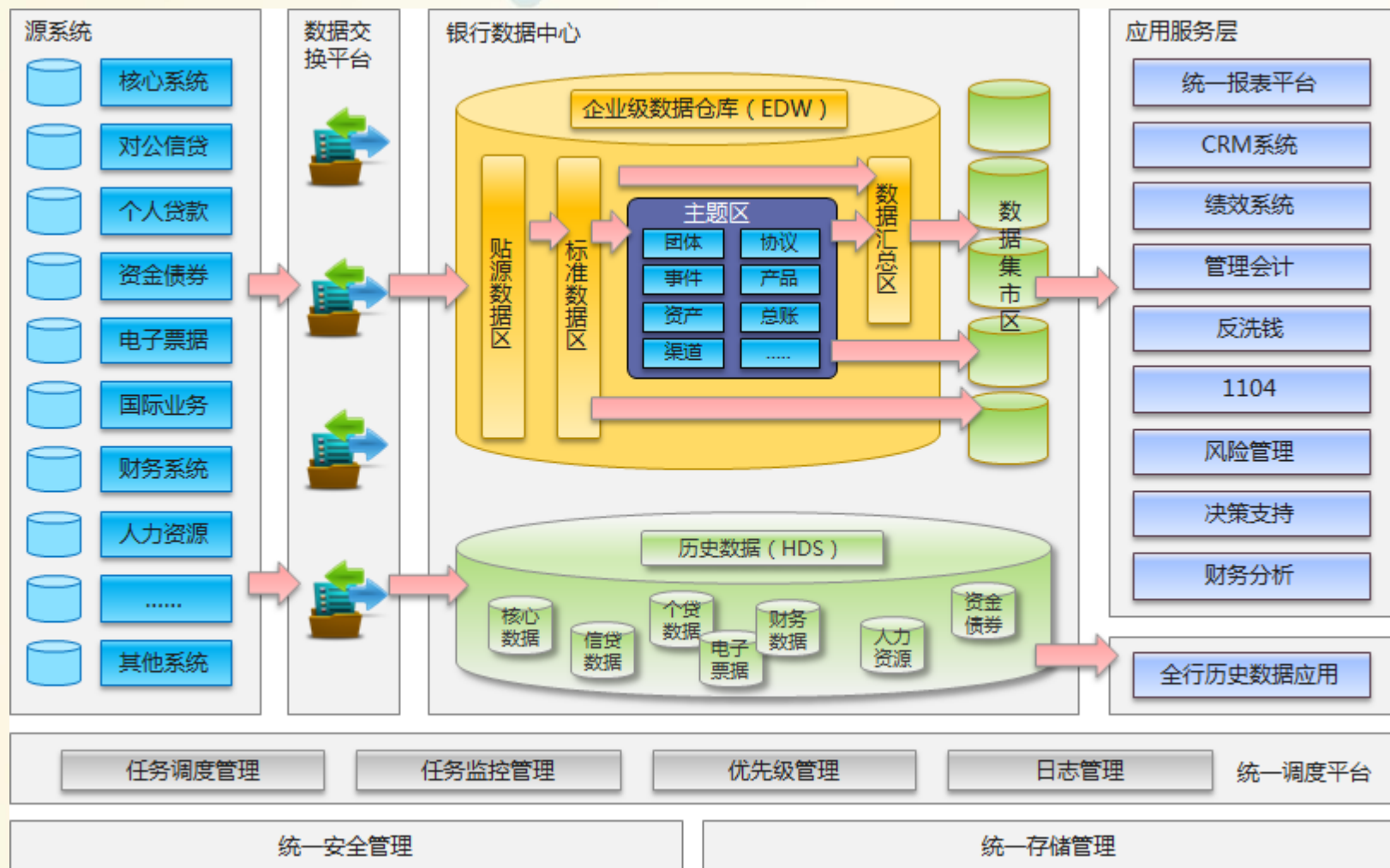
核心

- Schema
 - 星型 Star Schema
 - 瀑布 Snowflake Schema
- 事实表 （ Fact Tables ）
- 维度表 （ Dimensions ）

案例：BOA 美洲银行的数据仓库

- 美洲银行就是使用NCR Teradata建立数据仓库并获得成功应用的一个例子。该银行在几年的时间里曾先后兼并过十几家小银行，由于拥有的30多个OLTP业务系统太多而且分散，管理十分不容易，要找到准确的业务数据也很难。举例来说，它要准确地了解各个分行的客户资料就要花很多的时间，最后的结果还不一定完全准确。为此，美洲银行投资Teradata建立了一个中央的数据仓库，把各个分行系统中的数据都集中到中央库来，一些以前要几个星期才能得到答案的业务问题现在只需要几分钟甚至更少，效果非常明显。

业务架构



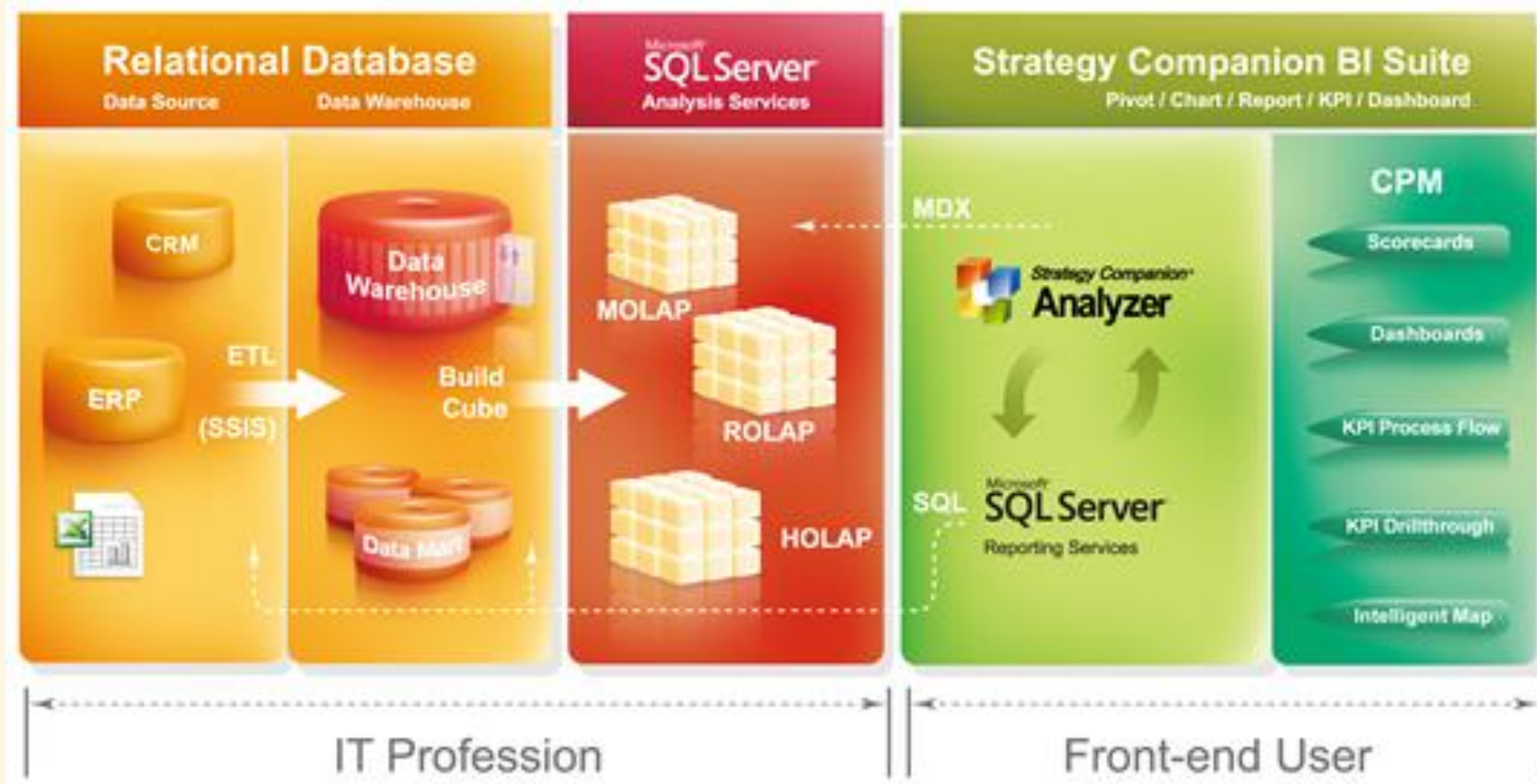
效果

- 例如，业务人员可以在分析银行的客户群中了解哪些类型或具有什么特征的客户最有可能购买哪一种产品或服务。美洲银行是**1986**年开始投资**Teradata**建立数据仓库的，采用循序渐进的方式实施数据仓库，刚开始时的数据库容量为**20GB**，后来逐步扩展成**3.4TB**的庞大系统。
- 在美洲银行的数据仓库中目前存有**280**亿份抵押贷款的资料。这套系统在**1994**年**1**月**17**日发生的洛杉矶大地震中充分显示了其价值。银行的住宅租赁部在几分钟内就确定了其损失。当时加州分行的副总裁**Charles Griffin**先生事后这样描述道：“我们可以根据邮政编码进到每个区，看看在遭受地震破坏的区域有多少以及有哪些类型的房产贷款。”根据这些信息，银行可以迅速作出反应，拿出相应的对策来。
- 统一的客户视图，便于企业做客户的精准营销

10. 多维数据库OLAP

- 联机分析处理 (OLAP) 的概念最早是由关系数据库之父 **E.F.Codd** 于1993年提出的, 他同时提出了关于OLAP的12条准则。OLAP的提出引起了很大的反响, OLAP作为一类产品同联机事务处理 (OLTP) 明显区分开来。
- Codd提出OLAP的12条准则来描述OLAP系统:
 - 联机分析处理
 - 联机分析处理
 - 准则1 OLAP模型必须提供多维概念视图
 - 准则2 透明性准则
 - 准则3 存取能力准则
 - 准则4 稳定的报表能力
 - 准则5 客户/服务器体系结构
 - 准则6 维的等同性准则
 - 准则7 动态的稀疏矩阵处理准则
 - 准则8 多用户支持能力准则
 - 准则9 非受限的跨维操作
 - 准则10 直观的数据操纵
 - 准则11 灵活的报表生成
 - 准则12 不受限的维与聚集层次

OLAP框架

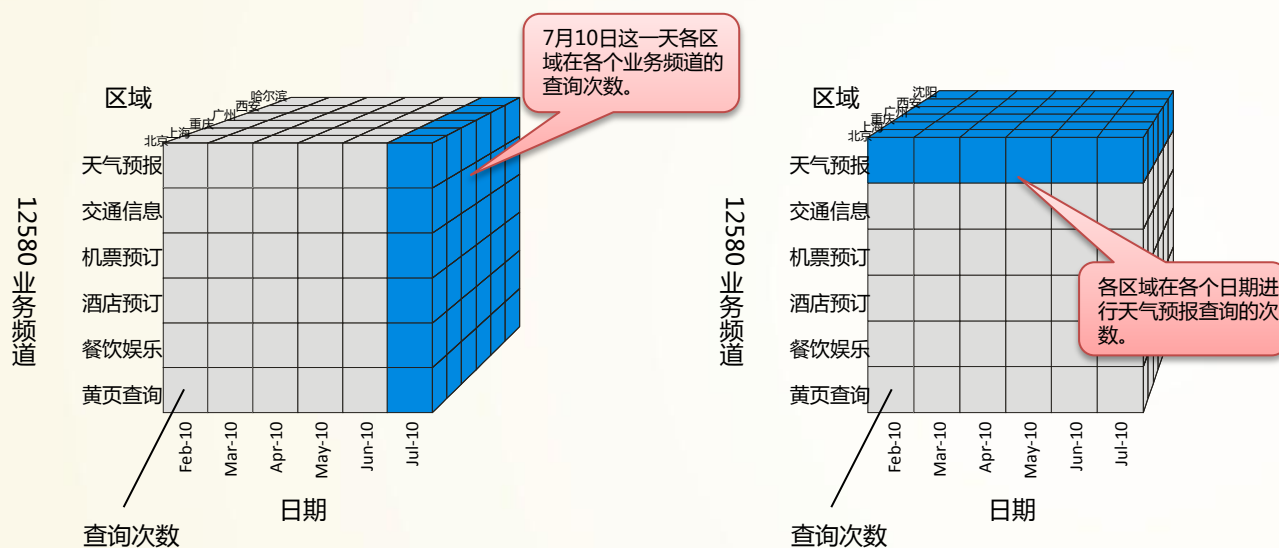


二维与多维数据切换

月份		2010年4月		2010年5月
地区	分公司	月份	查询次数	查询用户数
北方	Beijing	2010年5月	1,090,000	480,000
北方	Beijing	2010年4月	987,300	421,200
北方	Harbin	2010年5月	550,000	430,000
南方	Guangzhou	2010年5月	1,240,000	670,000
南方	Guangzhou	2010年4月	1,029,933	540,024
南方	Chongqing	2010年5月	560,000	430,000
南方	Chongqing	2010年4月	452,009	349,001

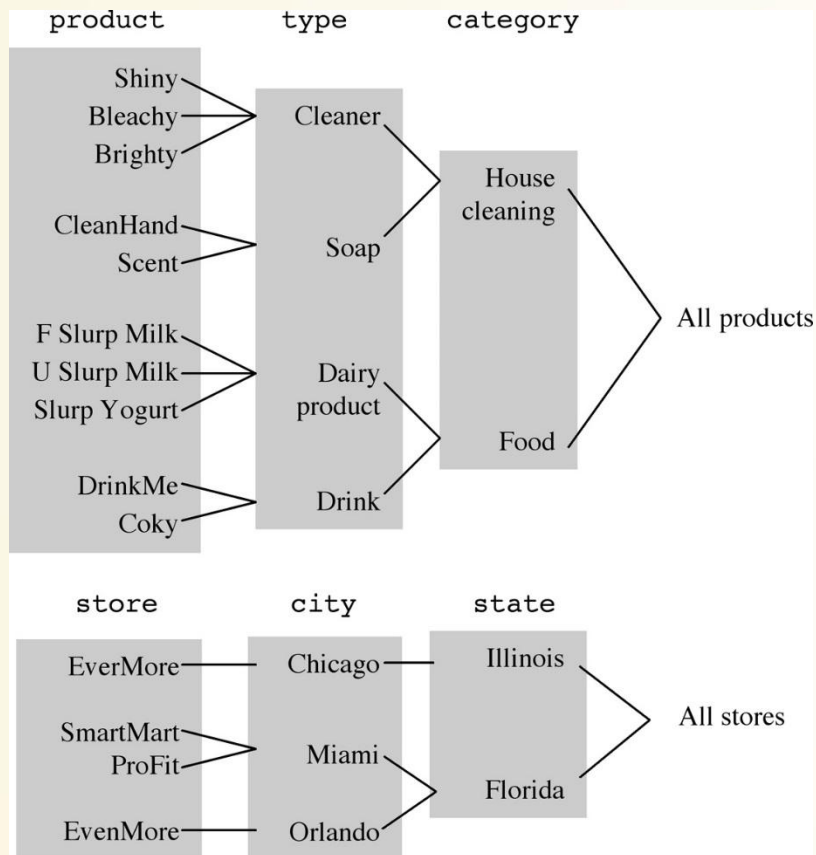
多维数据分析

- 切片



示意图：用切片的方法从不同的角度观察查询次数

为什么OLAP会快？ --- 聚集策略



MDX – 类SQL的多维查询语言

- SELECT
NON EMPTY {
[Product].[Product Category].[产品目录]
} ON COLUMNS,
NON EMPTY {
[Store].[地区]
} ON ROWS
FROM [Foodmart多维数据立方]
WHERE [Time By Day].[month_of_year].&[3]

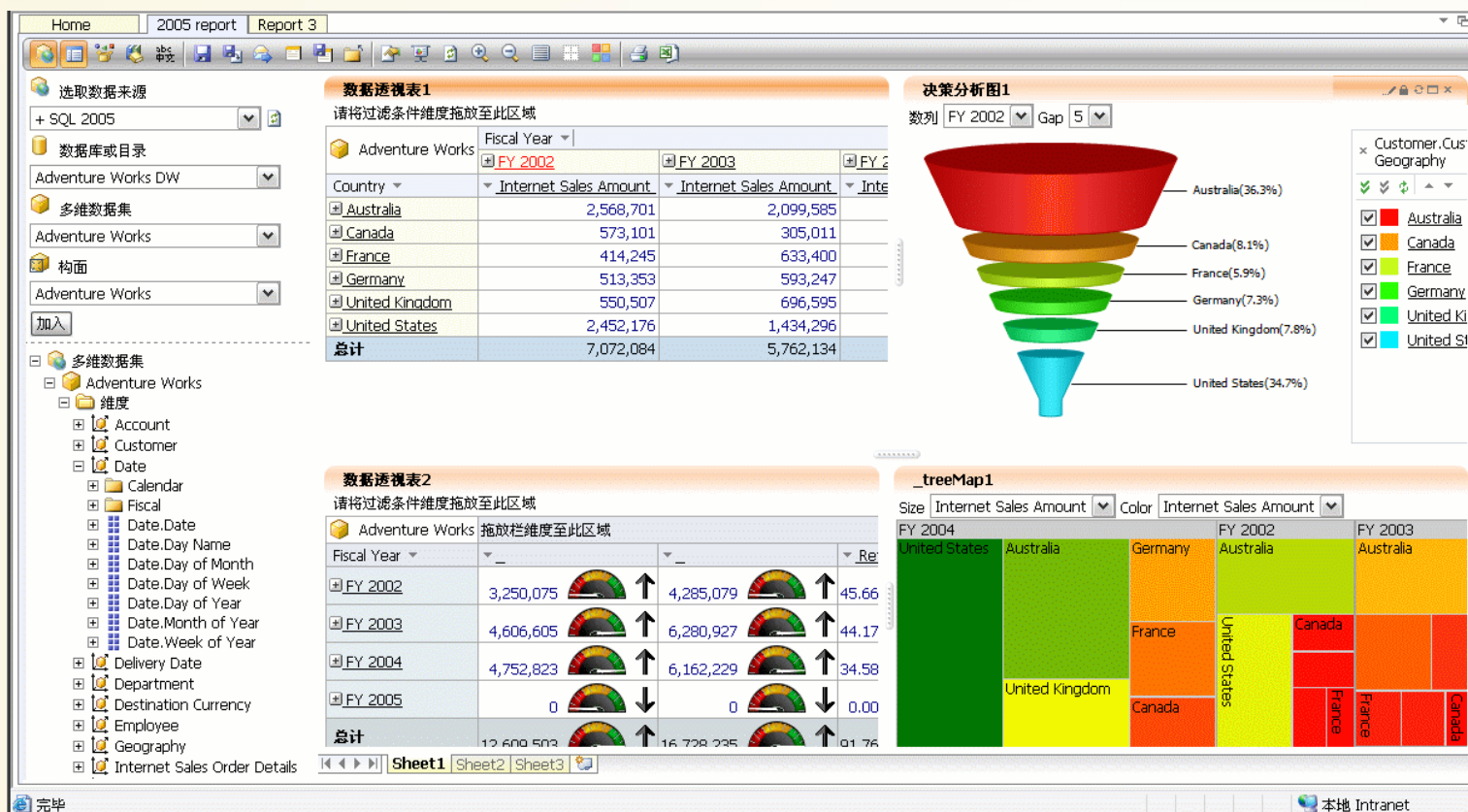
零售业务的多维数据展示

<div> 渠道 直营渠道 仓库层次 季节 产品年份 </div>									
<div> BSERPDW <div> <div>大类</div> <div>小类</div> </div> <div> <div>积分兑换礼品</div> <div>礼品</div> <div>男鞋</div> <div>男单</div> <div>男凉</div> <div>男棉</div> <div>男四季</div> <div>男鞋合计</div> <div>女鞋</div> <div>总计</div> </div> </div>									
年份	月	周	销售数量	销售数量	销售数量	销售数量	销售数量	销售数量	销售数量
2010年			-	927	72,957	4,462	5,886	1,454	84,759
2011年			22	1,694	78,918	4,257	6,065	1,745	90,985
2012年			1,028	2,434	74,589	5,350	5,852	1,609	87,400
2013年	1月		-	1,272	6,950	-	1,422	1	8,373
	2月		-	179	9,798	4	348	2	10,152
	3月		-	546	5,704	43	28	189	5,964
	4月		-	125	4,987	678	-	395	6,060
	5月		-	203	4,203	1,091	-	382	5,676
	6月		-	169	3,343	2,207	-	413	5,963
	7月	27周	-	45	505	431	-	45	981
		28周	-	29	592	616	-	52	1,260
		29周	-	29	669	494	-	50	1,213
		30周	-	50	658	506	-	38	1,202
		31周	-	44	403	298	-	25	726
	7月合计		-	197	2,827	2,345	-	210	5,382
	8月		-	129	3,433	1,226	-	128	4,787
	9月		-	160	9,585	60	23	19	9,687
	10月		-	113	6,729	1	147	5	6,882
	11月		-	449	6,302	-	799	2	7,103
	12月		716	378	6,173	2	2,256	-	8,431
									11,272

案例：进销存

- 场景：
 - ERP 系统只保存入库/出库记录
 - 需要查询每天的按商品SKU/店铺/时间/区域/商品种类的库存数量
- ERP的尴尬
 - 每次查询的时候需要进行时间的实时累积
 - 各种属性的组合非常费时间
 - 经常查某一天的数据，需要20-30分钟

解决方案：OLAP多维数据集



大数据 & NOSQL

- Not Only SQL!!!

- KV 数据库
- 文档型数据库
- 列式存储数据库
- 图形数据库

互联网三高

- 1. High performance
对数据库高并发读写的需求
- 2. Huge Storage
对海量数据的高效率存储和访问的需求
- 3. High Scalability & High Availability
对数据库的高可扩展性和高可用性的需求

关系型数据库面临的尴尬

- 1、数据库事务一致性需求
- 2、数据库的写实时性和读实时性需求
- 3、对复杂的SQL查询，特别是多表关联查询的需求

NOSQL解决的问题

- 1.满足极高读写性能需求的Key-Value数据库
- 2.满足海量存储需求和访问的面向文档的数据库
- 3.满足高可扩展性和可用性的面向分布式计算的数据库

11. 排序

- 排序是一个极其耗时的操作
- Does not Scale （假设是一千万的用户同时在线）

案例：Redis Sorted Set（有序集）

- 场景1：（排行榜）

Select top 10 username , score
from table
Order by score desc

- 场景2：（用户自身排名）

解决方案:

- 完美演绎Sorted Set集合

```
$b = $redis->zincrby('myadd',1,'b');  
//b的score加1，并返回当前b的score
```
















```
$c = $redis->zrange( 'myadd' ,score, 1, 10);  
//以score升序的方式,返回排名前十的
```

Reddit



- Reddit是个社交新闻站点，口号：提前于新闻发生，来自互联网的声音。其拥有者是Condé Nast Digital公司（Advance Magazine Publishers Inc的子公司）
- 用户（也叫**redditors**）能够浏览并且可以提交因特网上内容的链接或发布自己的原创或有关用户提交文本的帖子。其他的用户可对发布的链接进行高分或低分的投票，得分突出的链接会被放到首页。
- 另外，用户可对发布的链接进行评论以及回复其他评论者，这样就形成了一个在线社区。Reddit用户可以创造他们自己的论题部分，对发布链接和评论的人来说，就像Reddit用户提交的非正式也像社团的正式。

投票排名

- 2  3219 
 Every Bethesda game in particular... (i.imgur.com)
submitted 4 小时 ago by Jakewadewood to gaming
517 评论 分享
- 3  3846 
 It's amazing what rescue can do. (i.imgur.com)
submitted 5 小时 ago by KaptainKlutz to aww
317 评论 分享
- 4  3604 
 Selfie level 11 achieved. (i.imgur.com)
submitted 5 小时 ago by Zchavago to funny
361 评论 分享
- 5  3467 
 TIL John Swartzwelder, a writer for the Simpsons used to write episodes while sitting in a l
amounts of coffee and smoking endless cigarettes". When California passed an anti-smol
it in his house. (en.wikipedia.org)
submitted 7 小时 ago by BonerZero 🍷 to todayilearned
650 评论 分享
- 6  2756 
 Overheard my friend talking to his now ex-wife on the phone. (i.imgur.com)
submitted 7 小时 ago by bf13 to AdviceAnimals
260 评论 分享

商业模式

- 2012年，该网站取得了**370亿**页面浏览量，**4亿**独立访问者和**3000万**篇文章。单是10月，Reddit就取得了超过38亿的页面访问量超过4600万独立访问者，页面访问量比早前一年增长了一倍。
- 目前估值**4亿美金**

11. 异构数据 – No Schema

关系型数据库的不足

- 大量数据的写入操作
- Schema

案例: Survey

- Customer Insights



- 2013年1月美国在线调查服务公司SurveyMonkey已经以举债和股权融资的形式，募集资金8亿美元。
- 在这轮融资中，SurveyMonkey的估值达到**13亿美元**。

题型

题型样本

1. 您对我公司的企业文化是否了解？

☐ 是 ☐ 否

题型样本

1. 您手机支持的功能包括：

☐ 短信 ☐ 彩信 ☐ 上网 ☐ 游戏

☐ 电子邮件

题型样本

1. 请您在选购产品对以下方面的重要性进行排序(1 - 最重要, 7 - 最不重要)

	1	2	3	4	5	6	7
创新设计	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
视觉效果	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
功能	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
价格	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
品牌	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
价值	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
质量	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

题型:

题型样本

1. 愿景与使命

	非常不同意				非常同意
我对公司实现远景目标和使命充满信心	1	2	3	4	5
公司的使命目标使我觉得我的工作很重要	1	2	3	4	5
我认为公司制定的经营策略支持公司的远景目标	1	2	3	4	5
我认同公司的企业文化和核心价值观	1	2	3	4	5
公司致力于提供卓越的客户服务	1	2	3	4	5

题型样本

1. 你上个月的花费中，下列支出所占的百分比：（合计为100%）

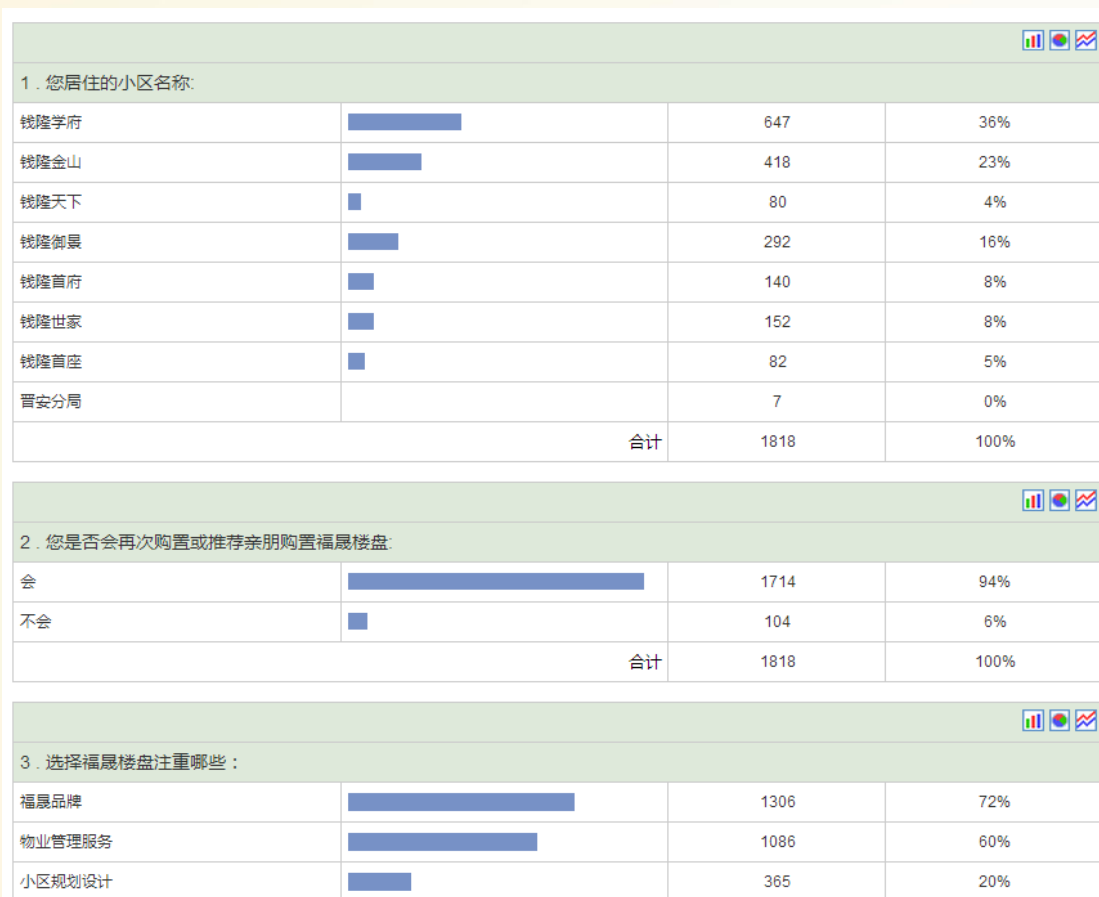
住宿	<input type="text"/>	<input type="button" value="▲"/> <input type="button" value="▼"/>
交通	<input type="text"/>	<input type="button" value="▲"/> <input type="button" value="▼"/>
饮食	<input type="text"/>	<input type="button" value="▲"/> <input type="button" value="▼"/>
娱乐	<input type="text"/>	<input type="button" value="▲"/> <input type="button" value="▼"/>
	<input type="text"/>	合计

题型样本

1. 总体而言，您对公司的总体满意度如何

非常不满意	非常满意			
1	2	3	4	5

Real-Time Analysis 实时跟踪



场景:

- 支持高达**20**种固定题型
 - 单选
 - 多选
 - 矩阵
 - 打分
 - 评论
 - 下拉框
- 同时支持新题型的扩展
 - 支持文档
 - 支持视屏
 - 支持声音
 - 支持图像
- 需要实时的分析
 - 答卷的实施统计与分析

NOSQL - MongoDB



- MongoDB 在最新一轮的融资中获得了 1.5 亿美元的 VC 资金，其估值也已达到 **12 亿** 美元。
- MongoDB 是一个基于分布式文件存储的数据库，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。
- Mongo 最大的特点是支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

Array

```
{ "_id" : 1234,  
  "dealershipName" : "Eric's Mongo Cars" ,  
  "cars" : [  
    { "year" : 2013,  
      "make" : "10gen" ,  
      "model" : "MongoCar" ,  
      "vin" : 3928056,  
      "mechanicNotes" : "Runs great!" },  
    { "year" : 1985,  
      "make" : "DeLorean" ,  
      "model" : "DMC-12" ,  
      "vin" : 8056309,  
      "mechanicNotes" : "Great Scott!" }  
  ]  
}
```

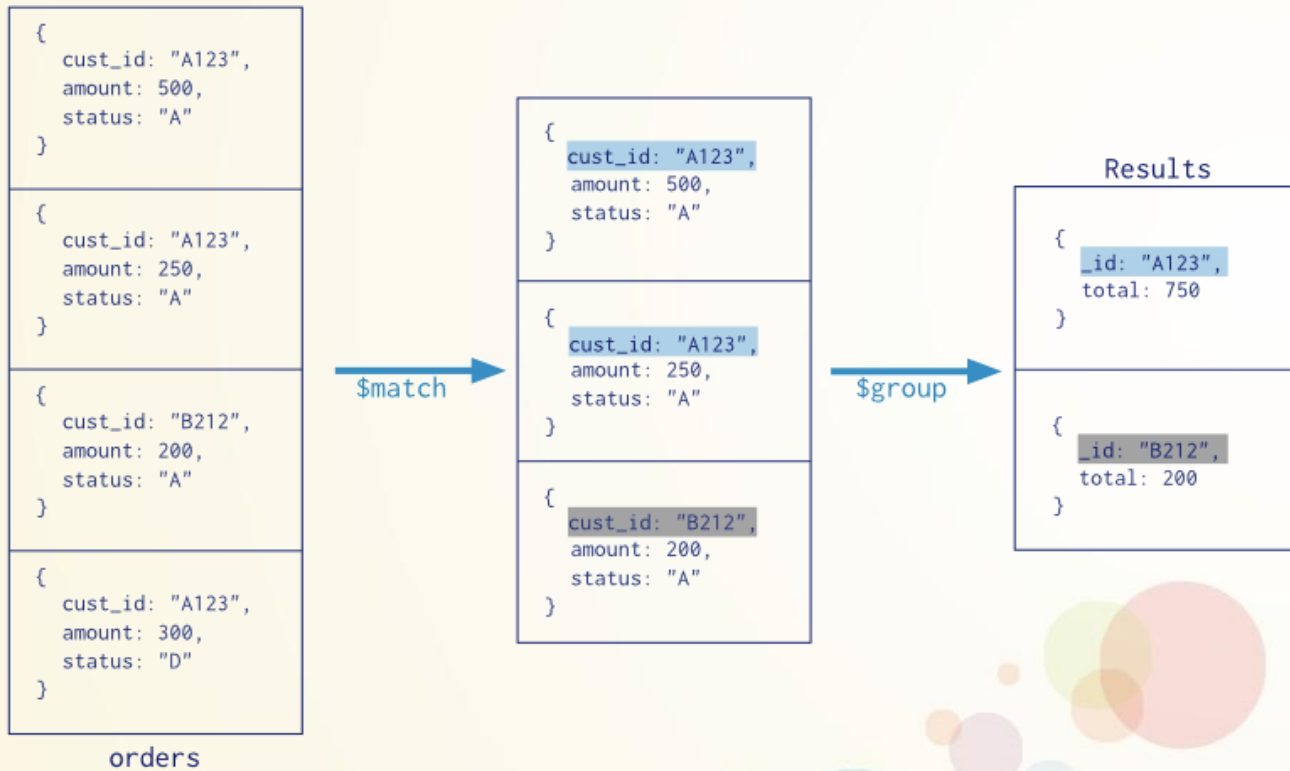
解决方案: MongoDB

```
• /* 0 */
• {
•   "_id" : ObjectId("52ea03baf7e4d32c58f1a7e8"),
•   "Title" : "2014年数据库技术大会调查问卷",
•   "Author" : "高峡",
•   "Questions" : [{
•     "id" : 1,
•     "type" : "单选",
•     "question" : "你最喜欢的数据库",
•     "answers" : ["Oracle", "SQL Server", "DB2", "Sybase"]
•   }, {
•     "id" : 2,
•     "type" : "多选",
•     "question" : "你感兴趣的大数据技术领域",
•     "answers" : ["社交", "电子商务", "能源化工", "互联网", "金融"]
•   }, {
•     "id" : 3,
•     "type" : "矩阵打分",
•     "question" : "你感兴趣的大数据技术领域",
•     "answers" : ["数据库设计模式的变迁", "OLAP的实践", "大数据的应用", "Oracle的运维"],
•     "options" : ["1.非常满意", "2.比较满意", "3.一般", "4.不满意", "5.非常不满意"]
•   }
• }
```

Aggregation Framework

Collection

```
db.orders.aggregate(  
  $match phase → { $match: { status: "A" } },  
  $group phase → { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }  
)
```



数据库其他领域的发展：

- 时序数据库
- 空间数据库

12. 时序数据库 – TimeSeries DB



程式交易

- 根据实时股价的变化，制定出相应的规则
 - 比如：如果股价连续N个时间段，每个时间段的股价比前一个时间段都高，就买入；

13. 空间数据库



场景:

- 保险公司需要根据飓风路径，算出有多少客户潜在的受影响，并通知客户撤离！
- 保险数据库里面保存客户的地址和地理坐标，可以快速计算出被影响的客户

14. MPP

NewSQL

- 列存储
- 关系型
- MPP

NoSQL

- Key-Value
- MapReduce
- MPP

OldSQL

- 行存储
- 关系型
- SMP

分布式计算，分布式文件系统

内存计算 (In Memory Computing)








新的硬件：Flash Card，SSD，Infiniband (40G/s)



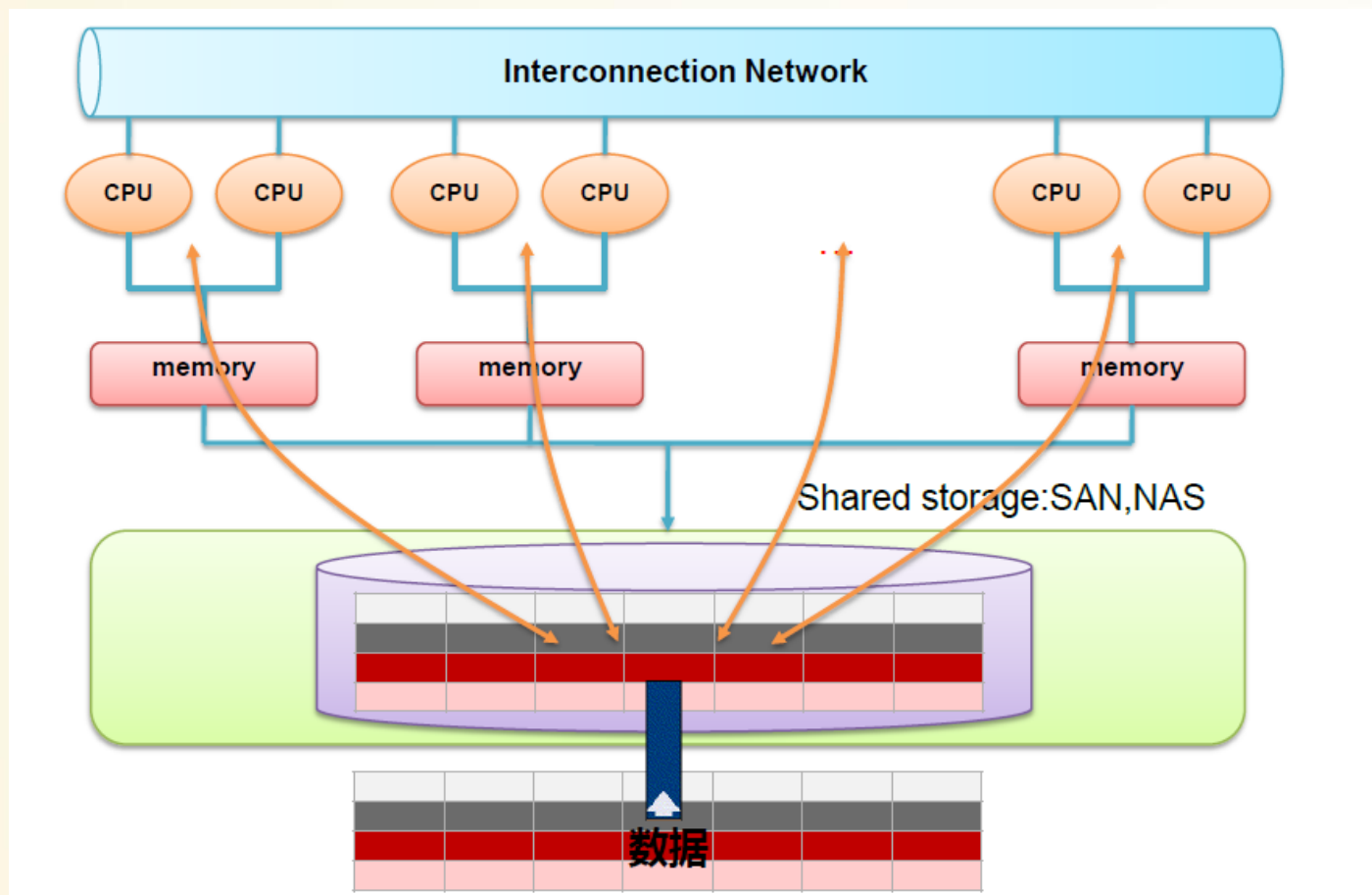
大数据激发了数据库行业技术创新的热情，主要的驱动力是对处理性能的强烈需求。为了提升性能，NewSQL阵营普遍采用了列存储技术；NoSQL阵营普遍采用了KV技术。三个阵营都不同程度地采用了分布式计算、分布式文件系统、内存计算技术，并积极地使用新的硬件技术，如大内存、Flash、SSD和高速网络连接（万兆交换机和Infiniband）

数据库市场

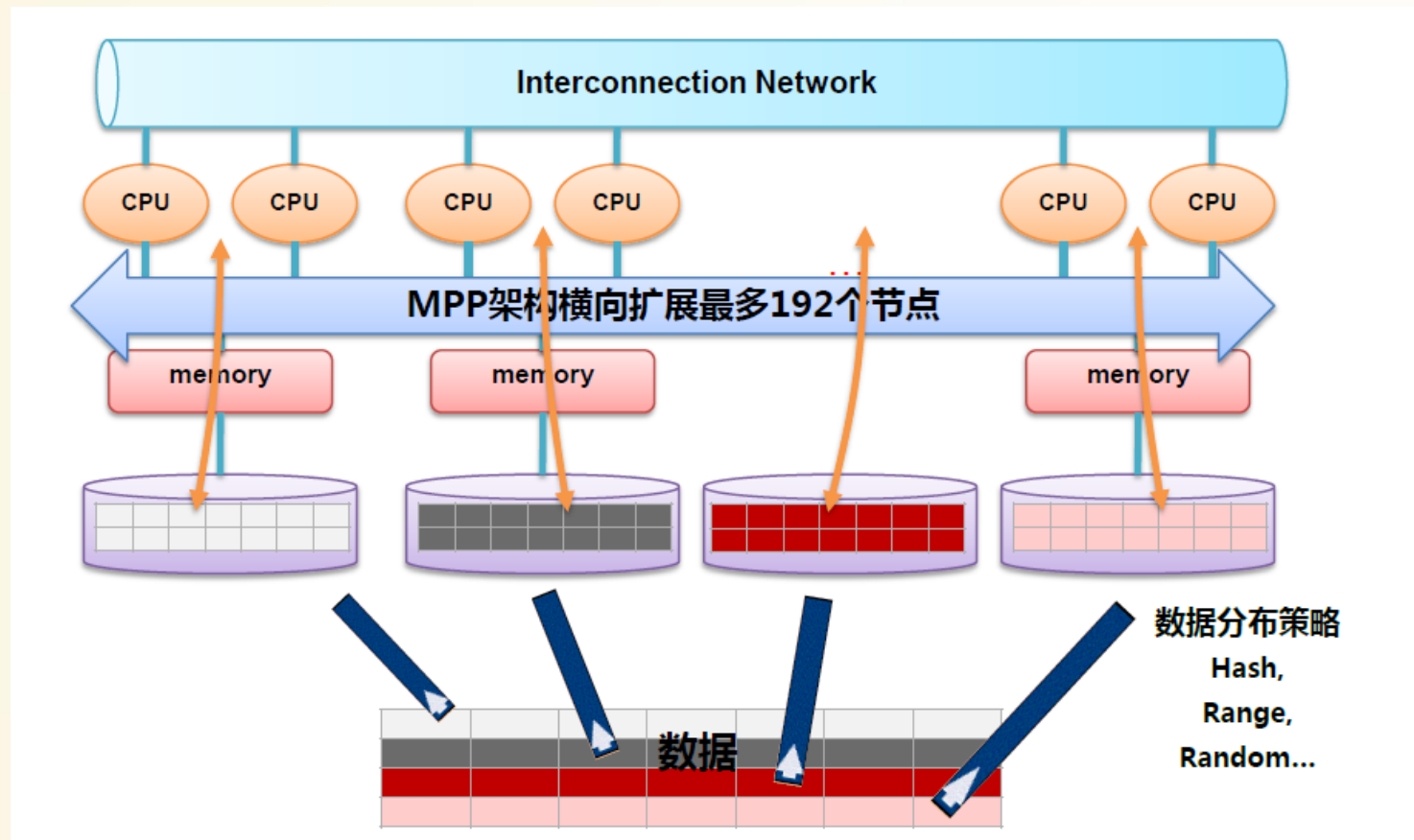
新型NewSQL数据库市场格局

		公司	产品	技术特点	产品来源
NewSQL	国外厂商		Sybase IQ	列存 + 共享磁盘	收购 (2010)
			HANA	内存数据库 + 列存	收购 (2009)
			Vertica	列存 + MPP	收购 (2011)
			Greenplum	行存 + 列存+MPP集群	收购 (2010)
			PDW	列存 + MPP集群	预计2013年初
	国内厂商		GBase 8a	列存 + MPP集群	国内唯一(2008年)
NoSQL	国外厂商		BigTable	列存 + Key Value	Google (2004年)
		 Apache Commons http://commons.apache.org/	HBase	列存 + Key Value	开源社区 (2006年)

Share Disk



Share Nothing + MPP



Vertica

企业数据分析挖掘系统的演变



传统数据仓库和分析系统

新一代数据仓库和分析基础架构

专有硬件设备	工业标准x86
价格昂贵	低成本
集中式专有结构	分布式集群
需要专门管理员	可以自助管理
批处理方式	实时查询
数据摘要和统计	原始数据深度挖掘
速度慢	速度快



10

成功案例：零售

Vertica in Retailing



GROUPON
Collective Buying Power

5ne

GUESS



挑战

- 顾客忠诚度
- 高度竞争的市场
- 以往基础设施能力的海量数据吞吐能力

解决方案

- 分析POS兆位元组的数据
- 更有效地管理库存
- 理解用户网上行为

收益

- 高用户忠诚度
- 极低的数据中心成本
- 以更低的成本就用户行为模式和竞争对手行动做出快速反应。



成功案例

Sample Customers

GUESS

Guess – 美国著名服装零售商

“服装零售市场瞬息万变，所以我们需要一个能够满足我公司的快节奏信息要求的高性能数据分析平台。在评估了其他几家供应商后，我们最终选择了Vertica公司，原因是Vertica公司的数据库能在创记录时间内给出我们业务上需要的数据查询结果，而成本却比其他供应商低得多。”

Guess 公司首席信息官---Mike Relich

- 我们使用Vertica数据库来分析公司全球多个数据中心内的数十TB的销售点 (POS)、制造、库存、客户和商店绩效数据。
- Vertica数据库查询速度是以前使用的数据库方案查询速度的200倍。
- 可以使用移动设备在店铺和仓库层访问Vertica数据库。

数据库设计模式的哲学

- 没有一颗银弹可以拯救世界
- 场景是最重要的
- 成功商业模式后面的数据库设计模式！
- 手中一定要拥有比锤子更多的工具！
- 集合（Set）思维
- 一颗永不泯灭的好奇心

欢迎

- 欢迎大家把日常碰到的难解数据库场景分享出来，我们一起找到合适的方案！
- 微博@高峡之数据时代

献给陪伴我们三十几年的关系型数据库



**You and I have memories
longer than the road that
stretches out ahead**

----- The Beatles

**前方蜿蜒无边的路再漫长，
也比不上你我之间的记忆**

Q&A

THANKS

