CSI4142 Data Science

Data Analytics: Business Intelligence and OLAP

(Notes by Hl Viktor © References indicated in text)

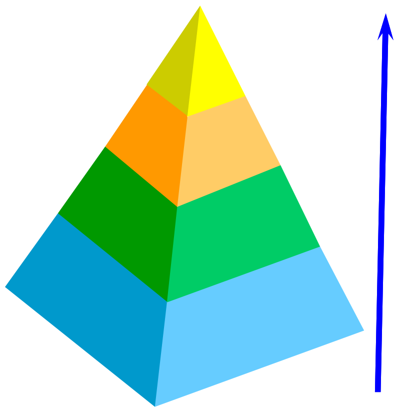
1

Overview

- Decision support and Business Intelligence
- Three types of users
- Dashboards and scorecards
- SQL and MDX for OLAP queries

2

The role of BI Applications



Strategic: **Ad hoc power users*****

"Push-button" knowledge workers

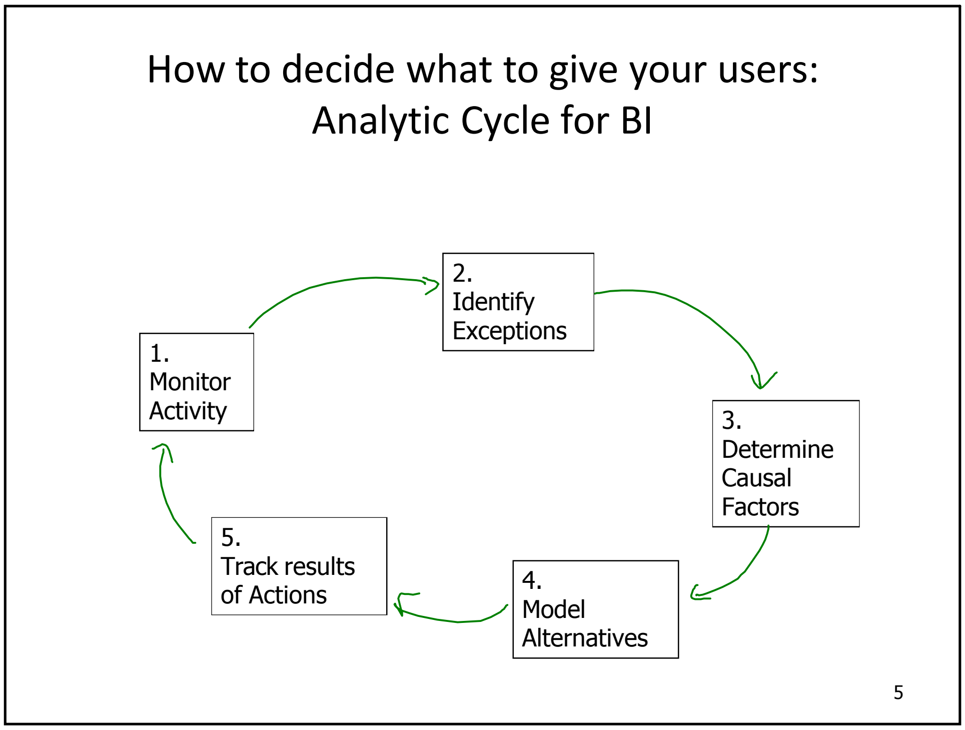
Operational: standard report consumers

3

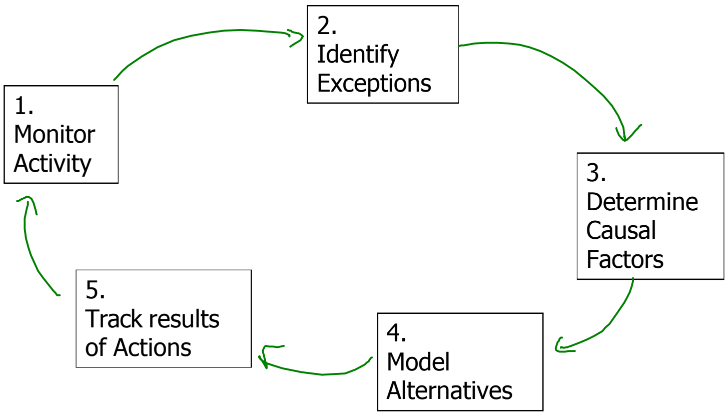
Types of BI applications

Direct access queries, reports and data mining	Strategic: Ad hoc power users
Standard reports Analytic Applications Dashboards and Scorecards	Push-button knowledge workers
Operation BI	Operational: standard report consumers

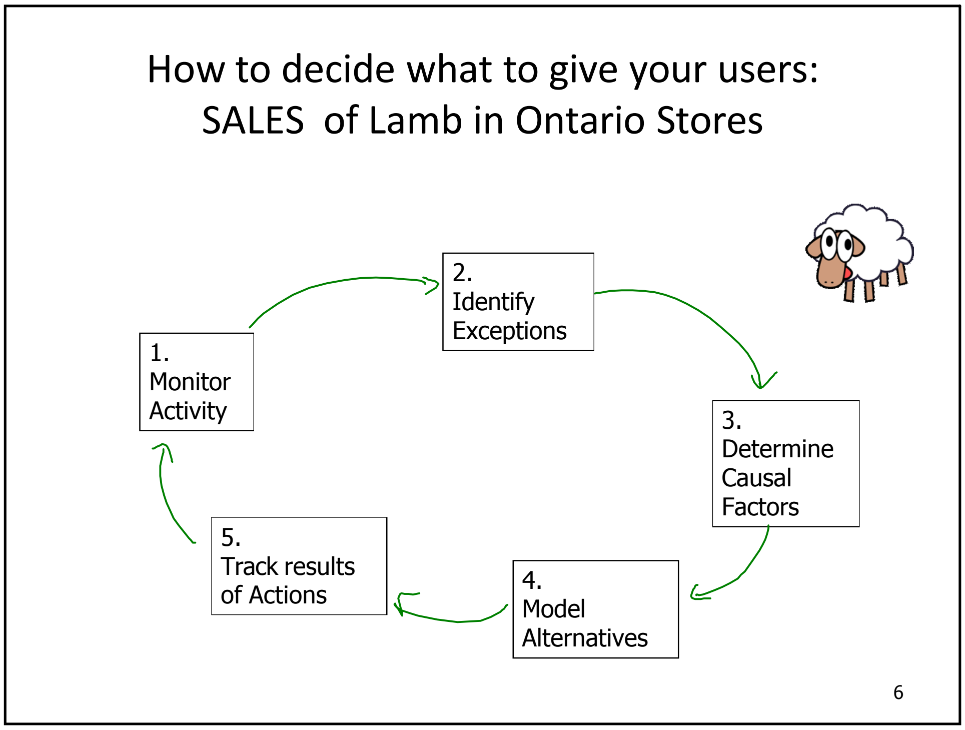
4



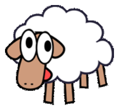
How to decide what to give your users: Analytic Cycle for BI

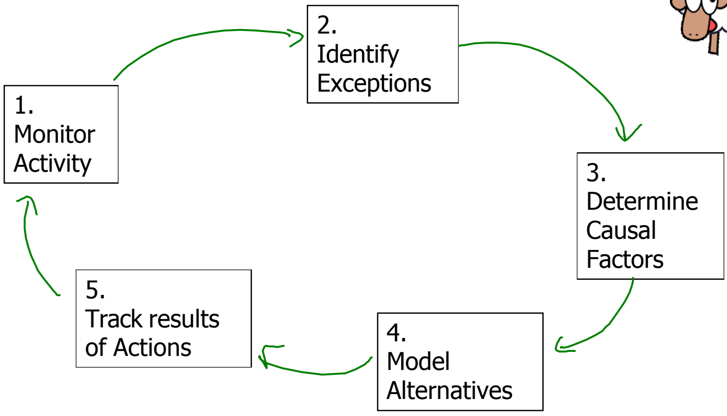


5



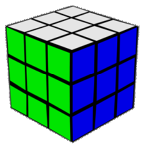
How to decide what to give your users: SALES of Lamb in Ontario Stores





6

Typical BI questions



- Explore using concept hierarchies
- Drill down, roll up, slice and dice

1. Compare the Sales in Ottawa versus Montreal for 2016
2. Compare the Sales in Ottawa versus Montreal in Dec 2016
3. Compare the Sales in Ottawa versus Toronto in Dec 2015 versus Dec 2016
4. Compare the Sales in Ottawa versus Vancouver for Lamb in Dec 2016 versus Nov 2016
5. Compare the Sales in Ottawa versus Vancouver for Lamb versus Chicken in Dec 2016 versus Nov 2016
6. Consider the Sales for 2016 → Q1 → Feb → 14 Feb
7. Find the 10% products with the highest Sales volume
8. Etc.

7

Visualize: Dashboards and scorecards

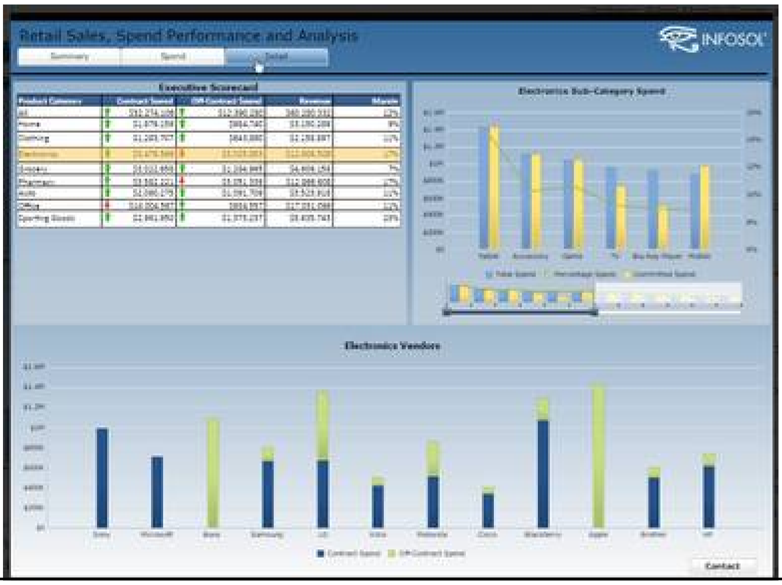
- https://www.klipfolio.com/ (in downtown Ottawa)
- http://prod.qlik.com/us/products/qlikview (in Kanata)

So how is this actually implemented?

A word about SQL and MDX…

Review your prior knowledge..

http://www.infosol.com/retail-sales-sample-dashboard/



Retail Sales, Spend Performance and Analysis
INFOSOL
Summary
Spend
Detail
Electronics Sub-Category Spend
Electronics Vendors
Contact

Executive Scorecard

Product Category	Contract Spend	Off-Contract Spend	Revenue	Margin
All	$12,274,108	$12,390,280	$60,390,332	12%
Home	$1,879,158	$984,740	$3,190,289	9%
Clothing	$1,283,707	$643,880	$2,158,897	11%
Electronics	$1,476,566	$1,523,283	$12,804,508	12%
Grocery	$3,022,658	$1,284,845	$4,804,154	7%
Pharmacy	$3,582,221	$3,051,556	$12,988,600	17%
Auto	$2,086,175	$1,091,709	$3,523,910	11%
Office	$16,206,567	$858,557	$17,031,088	12%
Sporting Goods	$2,861,892	$1,373,237	$8,835,743	19%

Typical OLAP queries

- SQL aggregate operations (recall DB1)
  - We use `SUM(), AVG(), MIN(), MAX` and `Group BY`
- SQL extensions:
  - Cubes, Rollups and Grouping Sets
  - Windowing
- MDX (used by Microsoft SQL Server)
- Iceberg queries: Top and Bottom

The two variants have “similar” syntaxes
Beware: portability issues between vendors

11

Typical OLAP queries

```
SELECT { [Measures].[Sales Amount] }
ON COLUMNS,
{[Date].[Year].&[2014],
 [Date].[Year].&[2015] }
ON ROWS FROM [Grocery Fact]
WHERE ( [Province].[Ontario] );
```

Partial Schema
Grocery Fact(Date-key, Store-key, … ,Sales Amount)
Date(Date-Key, …, month, year)
Store(Store-key, …, city, province)

12

Typical OLAP queries

```
SELECT { [Measures].[Sales Amount] }
ON COLUMNS,
{[Date].[Month].&[Dec], [Date].[Year].&[2014],
 [Date].[Month].&[Dec], [Date].[Year].&[2015]}
ON ROWS FROM [Grocery Fact]
WHERE ( [Province].[Ontario] );
```

Partial Schema:
Grocery Fact(Date-key, Store-key, …, Sales Amount)
Date(Date-Key, …, month, year)
Store(Store-key, …, city, province)

13

Typical OLAP query: Windowing

- Compare each product's price with the average price in its brand:

```
SELECT p.brand, p.name, p.price, avg(p.price)
OVER (PARTITION BY p.brand)
FROM product p;
```

Brand	Name	Price	Avg(Price)
Natrel	Milk 2%	2.00	2.833
Natrel	Cream	4.00	2.833
Natrel	Milk 1 %	2.50	2.833
Quebon	Milk 2 %	2.20	2.600
Quebon	Milk Soya	3.00	2.600

14

Typical OLAP query: Windowing

- Compare each product's price with the average price in its brand and provide a rank:

```
SELECT p.brand, p.name, p.price, avg(p.price)
RANK() OVER (PARTITION BY p.brand)
FROM product p;
```

Brand	Name	Price	Avg(Price)	Rank
Natrel	Milk 2%	2.00	2.833	1
Natrel	Cream	4.00	2.833	3
Natrel	Milk 1 %	2.50	2.833	2
Quebon	Milk 2 %	2.20	2.600	1
Quebon	Milk Soya	3.00	2.600	2

15

The WINDOW Clause

```
SELECT L.state, T.month, AVG(S.sales) OVER W AS movavg
FROM  Sales S, Times T, Locations L
WHERE S.timeid=T.timeid AND S.locid=L.locid
WINDOW W AS (PARTITION BY L.state
       ORDER BY T.month
       RANGE BETWEEN INTERVAL `1' MONTH PRECEDING
       AND INTERVAL `1' MONTH FOLLOWING)
```

- This example shows moving average of sales over 3 months
- Let the result of the FROM and WHERE clauses be “Temp”.
- (Conceptually) Temp is partitioned according to the PARTITION BY clause.
  - Similar to GROUP BY, but the answer has one row for each row in a partition, not one row per partition!

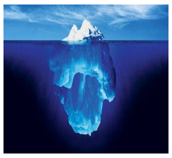
Typical OLAP queries: `TopCount` in MDX

```
SELECT {TOPCOUNT([Product].[Name], 5,
[Measures].[Sales Amount] }
ON ROWS FROM [Grocery Fact]
WHERE ( [Province].[Ontario] );
// returns the products (with their sales) of
the 5 best sellers
```

Partial Schema:
Grocery Fact(Date-key, Store-key, Product-key, …, Sales Amount)
Date(Date-Key, …, month, year)
Store(Store-key, …, city, province)
Product(Product-key, name, brand, …)

17

Top *N* Queries in SQL

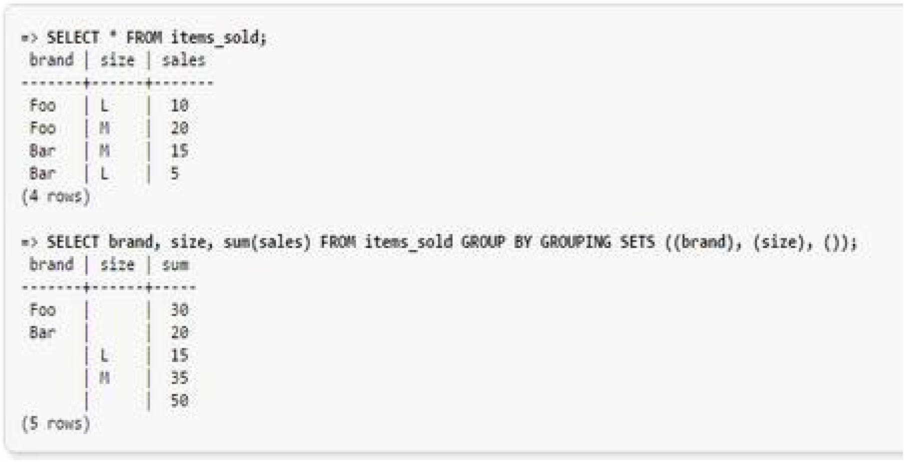


- Query optimizer is used to find the top *n* rows
- So-called Iceberg Queries

```
SELECT P.pid, P.pname, S.sales
FROM Sales S, Products P
WHERE S.pid=P.pid AND S.locid=1 AND
S.timeid=3
ORDER BY S.sales DESC
OPTIMIZE FOR 10 ROWS
```

Grouping sets

(https://www.postgresql.org/docs/devel/static/queries-table-expressions.html)



```
=> SELECT * FROM items_sold;
 brand | size | sales
-------+------+-------
 Foo   | L    |  10
 Foo   | M    |  20
 Bar   | M    |  15
 Bar   | L    |  5
(4 rows)

=> SELECT brand, size, sum(sales) FROM items_sold GROUP BY GROUPING SETS ((brand), (size), ());
 brand | size | sum
-------+------+-----
 Foo   |      |  30
 Bar   |      |  20
       | L    |  15
       | M    |  35
       |      |  50
(5 rows)
```

// note that () denotes ALL

19

Rollup operation

`ROLLUP (e1, e2, e3);`

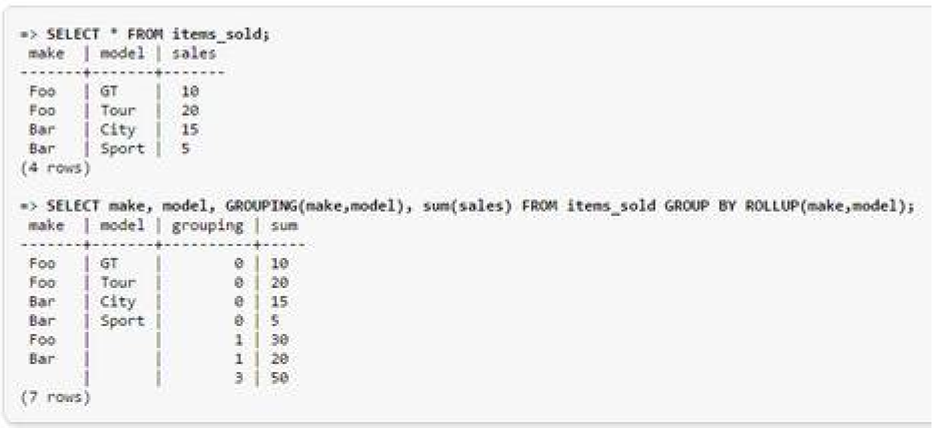
Is equivalent to:

`GROUPING SETS ((e1,e2,e3),(e1,e2), (e1),());`

20

Rollup operation

(https://www.postgresql.org/docs/devel/static/queries-table-expressions.html)



```
=> SELECT * FROM items_sold;
 make  | model | sales
-------+-------+-------
 Foo   | GT    |  10
 Foo   | Tour  |  20
 Bar   | City  |  15
 Bar   | Sport |  5
(4 rows)

=> SELECT make, model, GROUPING(make,model), sum(sales) FROM items_sold GROUP BY ROLLUP(make,model);
 make  | model | grouping | sum
-------+-------+----------+-----
 Foo   | GT    |        0 | 10
 Foo   | Tour  |        0 | 20
 Bar   | City  |        0 | 15
 Bar   | Sport |        0 | 5
 Foo   |       |        1 | 30
 Bar   |       |        1 | 20
       |       |        3 | 50
(7 rows)
```

21

Cube(): Powerset of Grouping Sets

```
cube(a, b, c);
```
is equivalent to:
```
GROUPING SETS
(  (a,b,c),
   (a,b),(a,c),
   (a),
   (b,c),(b),
   (c),
   ());
```

// note that () denotes ALL

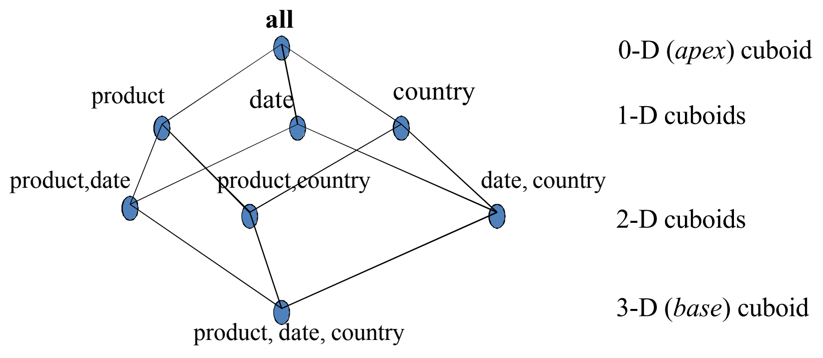
22

The CUBE Operator

- CUBE (product-key, store-key, date-key) BY SUM (Sales Amount)
  - if there are k dimensions, we have 2^k possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.
  - Equivalent to rolling up Sales on all eight subsets of the set {product-key, store-key, date-key}; each roll-up corresponds to an SQL query of the form:

```
SELECT SUM(Sales Amount)
FROM   Grocery Fact
GROUP BY grouping-list
```

Cube with 3 dimensions: 8 times `group by`



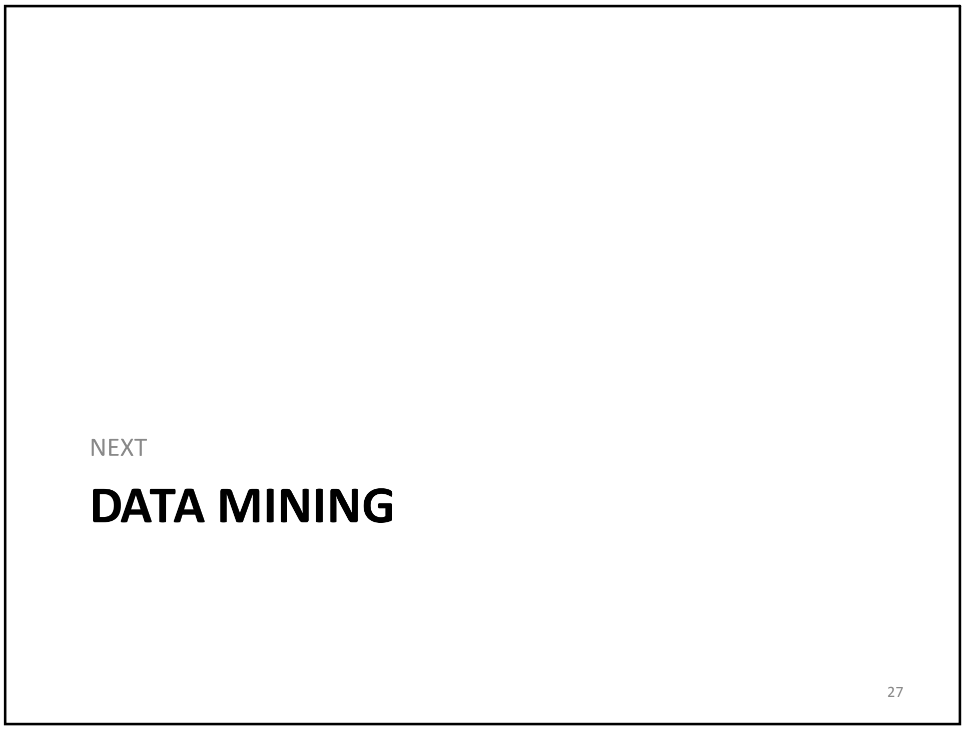
24

Implementing BI applications: the bottom line

- Use a dashboard template
- Use a tool such as PowerPivot
- Write the SQL or MDX code yourself

Summary

- BI applications target “knowledge workers”
- Dashboards and scorecards
- OLAP operations for Analytics
  - Aggregates and Group By
  - Windowing and Top N
  - Groupings Sets, Roll Ups and Cubes
- For the Project: Practical class on Feb 13th



NEXT

DATA MINING