**VICTORIA UNIVERSITY OF WELLINGTON**
***Te Whare Wananga o te Upoko o te Ika a Maui***

# *Basic OLAP Queries*

*Lecturer : Dr. Pavle Mogin*

*SWEN 432*
*Advanced Database Design and Implementation*

# Plan for OLAP Queries

- Basic multidimensional OLAP queries:
  - Roll – up and Drill – down
  - Slice and Dice
  - Pivoting

  - *Readings :*
    - *Ramakrishnan, Gehrke: "Database Management Systems", Chapter 25, Sections 25.3, 25.4, and 25.6*
    - *Chaudhuri and Dayal* : "An Overview of Datawarehousing and OLAP Technologies"

# *Roll – Up and Drill - Down*

- ***Roll – up*** operation corresponds to taking the current aggregation level of fact values and doing a further aggregation (getting coarser grained data)
- Each roll-up operation can be expressed using a SQL/92 SELECT…AGG()… GROUP BY… statement
- ***Drill – down*** is just an opposite operation of roll – up, but it requires the access to data of finer granularity

# *Classification of Roll – Up*

- ***Roll – up*** operations can be classified onto:
  - **Dimensional** roll-ups that are done by dropping one or more dimensions
  - **Hierarchical** roll-ups that are done by climbing up the attribute hierarchies of dimensions
  - **Mixed** roll-ups that combine the previous two techniques
- Note that in an ultimate case, the hierarchical roll-up above the top level of an attribute hierarchy (to the attribute “all”) can be viewed as converting to a dimensional roll-up

# Example Star Schema

- In the examples that follow, we shall consider the following star schema:
  - Fact Table :
    - *Sales* (*ShopId, ProdId*, *TimeId*, *Total*)
    - *Total* is a fact
  - Dimension tables:
    - *Location*(*ShopId, City ),*
    - *Product*(*ProdId*, *ProdName*, *Type)* ,
    - *Time*(*TimeId*, *Week*)
  - Dimension attribute hierarchies:
    - *ShopId*→*City* (location hierarchy)
    - *ProdId*→*Type* (product hierarchy)
    - *TimeId*→*Week* (time hierarchy)

# *Dimensional Roll – Ups*

- One dimension dropped:

  SELECT *ProdId*, *ShopId*, SUM(*Total* ) AS ***Total_by_Prod_Loc***

  FROM *Sales*

  GROUP BY *ProdId*, *ShopId* ;

- Two dimensions dropped:

  SELECT *ProdId,* SUM(*Total* ) AS ***Total_by_Prod***

  FROM *Sales*

  GROUP BY *ProdId* ;

- All dimensions dropped:

  SELECT SUM(*Total* ) AS ***Overall_Total***

  FROM *Sales*;

# Hierarchical Roll- Ups (one dimension)

- One roll – up would give sales by product, ***city*** and day

  SELECT *ProdId*, *City*, *TimeId*, SUM(*Total*) AS ***City_Total***
  FROM *Sales NATURAL JOIN Location*
  GROUP BY *City*, *ProdId*, *TimeId* ;

- The second roll – up would give sales by ***product type***, shop and day

  SELECT *Type*, *ShopId*, *TimeId*, SUM(*Total*) AS ***Type_Total***
  FROM *Sales NATURAL JOIN Product*
  GROUP BY *Type*, *ShopId*, *TimeId* ;

- The third roll – up ***Week_Total*** would give sales by product, shop and ***week***

  SELECT *ProductId*, *ShopId*, *Week*, SUM(*Total*) AS ***Week_Total***
  FROM *Sales NATURAL JOIN Time*
  GROUP BY *Week, ShopId, ProductId* ;

# *Hierarchical Roll-Ups (2d and 3d)*

- Two dimensional roll-up:

  SELECT *Type*, *City*, *TimeId*, SUM(*Total* ) AS ***Type_City_Tot***
  FROM *Sales NATURAL JOIN Product NATURAL JOIN Location*
  GROUP BY *Tipe*, *City*, *TimeId* ;

  (There would be also two other 2d roll-ups possible (***Type_Week*** and ***City_Week*** ) but these are not shown)

- Three dimensional roll-up:

  SELECT *Type*, *City*, *Week*, SUM(*Total* ) AS ***Type_City_Week_Tot***
  FROM *Sales NATURAL JOIN Product NATURAL JOIN Location NATURAL JOIN Time*
  GROUP BY *Type*, *City*, *Week* ;

# Roll – Up Summary

- Generally, if there are *k* dimensions, it is possible to define more than $2^k$ roll - up SQL queries of the type:

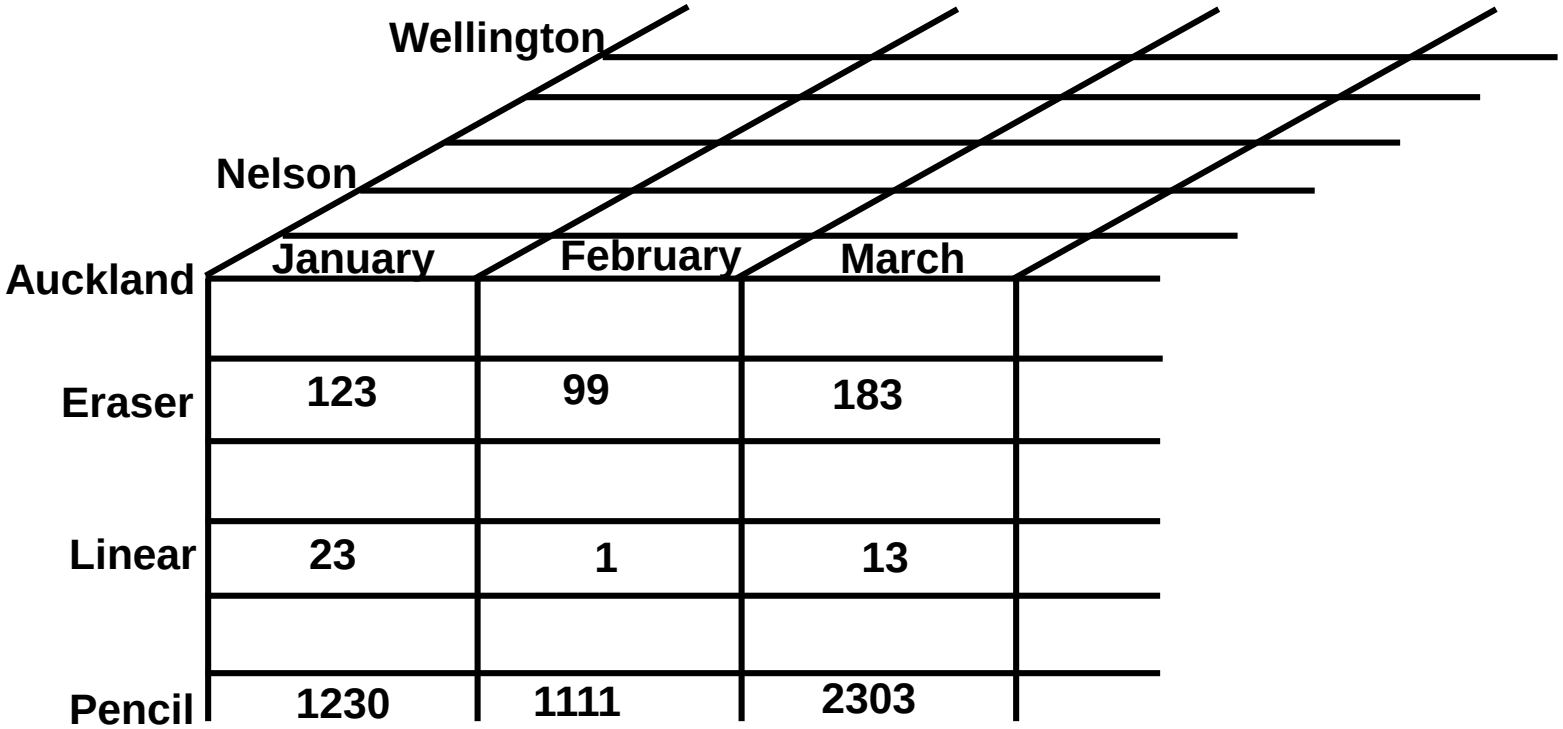
```
SELECT <attribute_list> AGG(attribute_name)
FROM <list_of_tables>
WHERE <list_of_conditions>
GROUP BY <grouping_list>
```

- There are exactly $2^k$ dimensional roll – ups
- The number of hierarchical roll – ups depends on the number of levels of dimension attribute hierarchies
- A simple (with no parallel branches) attribute hierarchy with *n* levels, gives raise to *n* roll-ups

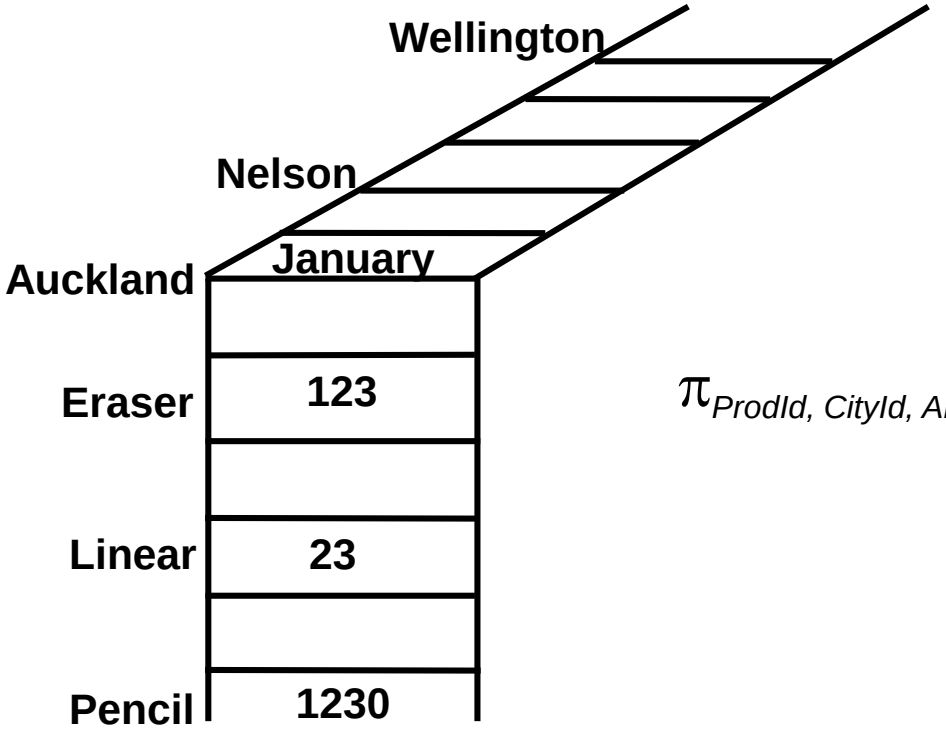
# *Slice_and_Dice Operations*

- ***Slice*** operation corresponds to reducing the number of dimensions by taking a projection of facts on a proper subset of dimensions for some selected attribute values of dimensions that are being dropped
- ***Slice*** operation amounts to equality select condition
- ***Dice*** operation amounts to range select condition on one dimension, or to equality select condition on more than one dimension
- Both terms come from visualizing the effect of these operations on a hypercube data set
- Here, the term **select** relates to the $\sigma$ operator of the relational algebra

# A Sales Hypercube

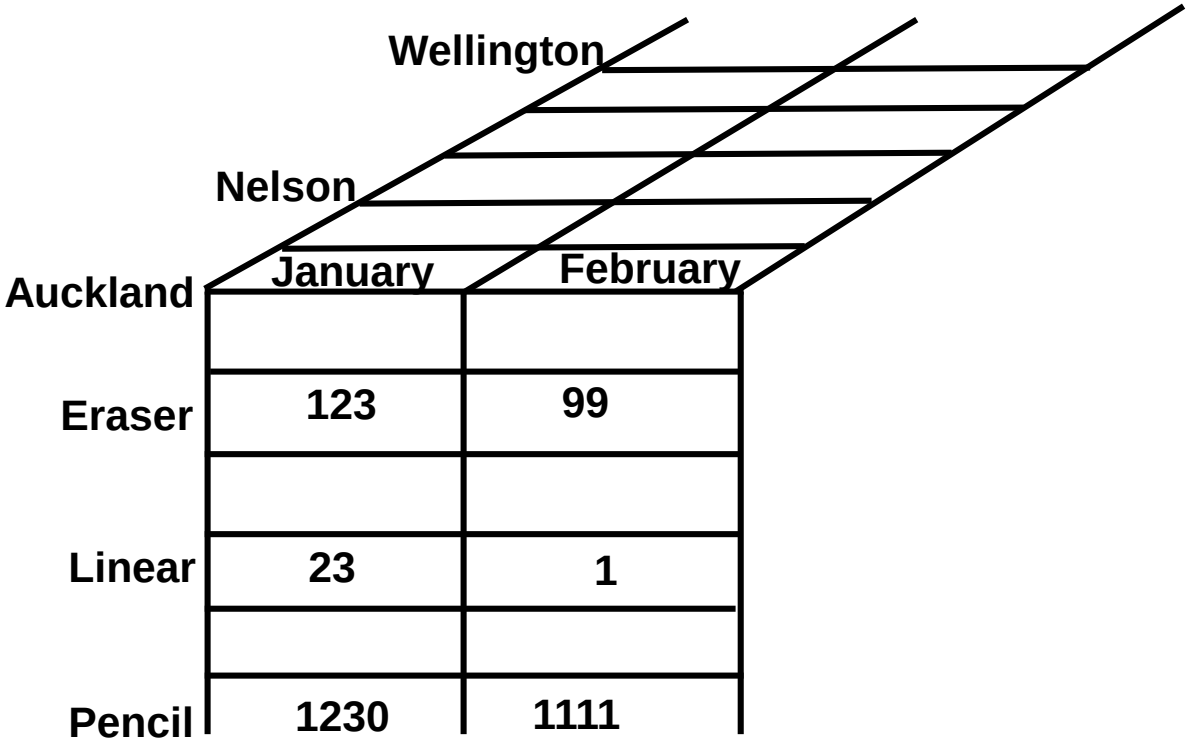


# Slice Operation Visualization




$$\pi_{ProdId,\ CityId,\ Amount}\ (\sigma_{MonthId\ =\ January}(Sales))$$

# Dice Operation Visualization




$\pi_{ProdId,\ CityId,\ MonthId,\ Amount}\ (\sigma_{MonthId \in \{January,\ February\}}(Sales))$

# SQL and Slice_and_Dice Operations

- Consider *Sales* fact table with *Product*, *City*, and *Time* as dimensions:

  *Sales* (*ProdId, CityId, MonthId, Amount* )

- A slice operation can be described in the following way

```
SELECT ProdId, CityId, Amount
FROM Sales
WHERE MonthId = 'January' ;
```

- A dice operation would be:

```
SELECT ProdId, CityId, MonthId, Amount
FROM Sales
WHERE MonthId ='January' OR MonthId = 'February' ;
```

or

```
SELECT CityId, Amount
FROM Sales
WHERE MonthId ='January' AND ProdId = 'Pencil' ;
```

# *Pivoting*

- In a fact table, each tuple corresponds to at least one measure value and there is a column for each of dimensions
- The simplest view of ***pivoting*** is that it selects two dimensions to aggregate one of measures
- The aggregated values are often displayed in a grid where each point in the (*x*, *y* ) coordinate system corresponds to an aggregated value of the measure
- The *x* and *y* coordinate values are the values of the selected two dimensions
- The result of pivoting is also called ***cross – tabulation***

# Extension of a Star Schema

Location

| CityId | City |
|---|---|
| 1 | Well |
| 2 | Nels |
| 3 | Auck |

SalesPerson

| PerId | Name |
|---|---|
| 1 | John |
| 2 | Susan |
| 3 | James |
| 4 | Susan |
| 5 | Ann |

Sales

| CityId | PerId | TimId | Amnt |
|---|---|---|---|
| 1 | 1 | 1 | 230 |
| 1 | 1 | 2 | 300 |
| 1 | 1 | 8 | 310 |
| 1 | 2 | 7 | 50 |
| 2 | 3 | 1 | 550 |
| 2 | 3 | 5 | 100 |
| 3 | 4 | 6 | 880 |
| 3 | 5 | 1 | 60 |
| 3 | 5 | 2 | 60 |
| 3 | 5 | 4 | 140 |

Time

| TimId | Day |
|---|---|
| 1 | Mon |
| 2 | Tue |
| 3 | Wed |
| 4 | Thu |
| 5 | Fri |
| 6 | Sut |
| 7 | San |
| 8 | Mon |

# Pivoting on City and Day

| | Mon | Tue | Wed | Thu | Fri | Sat | San | SubTotal |
|---|---|---|---|---|---|---|---|---|
| Auckland | *60* | *60* | *0* | *140* | *0* | *880* | *0* | *1140* |
| Nelson | *550* | *0* | *0* | *0* | *100* | *0* | *0* | *650* |
| Wellington | *540* | *300* | *0* | *0* | *0* | *0* | *50* | *890* |
| SubTotal | *1150* | *360* | *0* | *140* | *100* | *880* | *50* | *2680* |

# Expressing Pivoting by SQL (body)

```
SELECT City, Day, SUM(Amnt) AS Sales
FROM Sales NATURAL JOIN Location NATURAL JOIN Time
GROUP BY City, Day;
```

| *City* | *Day* | *Sales* |
|---|---|---|
| Auckland | Monday | *60* |
| Auckland | Tuesday | *60* |
| Auckland | Thursday | *140* |
| Auckland | Saturday | *880* |
| Nelson | Monday | *550* |
| Nelson | Friday | *100* |

| *City* | *Day* | *Sales* |
|---|---|---|
| Wellington | Monday | *540* |
| Wellington | Tuesday | *300* |
| Wellington | Sunday | *50* |

**But, this is only the body of the pivoting table**

# *Expressing Pivoting by SQL (subs&total)*

```
SELECT Day, SUM(Amnt) AS Total
FROM Sales NATURAL JOIN Time
GROUP BY Day;
```

| Day | Total |
|---|---|
| Mon | *1150* |
| Tue | *360* |
| Thu | *140* |
| Fri | *100* |
| Sat | *880* |
| San | *50* |

```
SELECT City, SUM(Amnt) AS Total
FROM Sales NATURAL JOIN Location
GROUP BY City;
```

| City | Total |
|---|---|
| Auck | *1140* |
| Nels | *650* |
| Well | *890* |

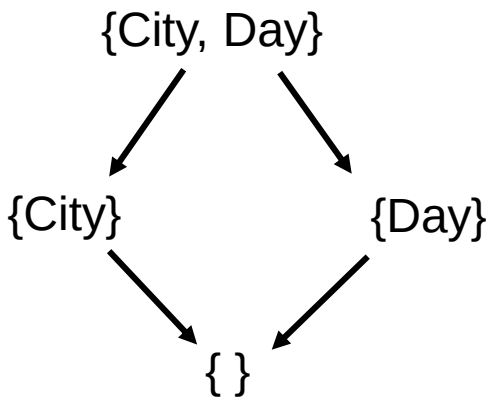
```
SELECT SUM(Amnt)
FROM Sales;
```

| SUM(Amnt) |
|---|
| 2680 |

# Summary of the Cross Tab Example

SELECT City, Day, SUM(Amnt)… GROUP BY City, Day

SELECT City
…
GROUP BY City;




SELECT Day
…
GROUP BY Day;

SELECT SUM(Amnt) FROM Sales

Structures like that one are often called ***cube lattice***

# Replacing Pivot Operation by SQL

- To express a pivot operation involving two dimensions, we would have to issue the following SQL/92 statement:

# Replacing Pivot Operation by SQL

```
(SELECT City, Day, SUM(Amnt) FROM Sales NATURAL JOIN
Location NATURAL JOIN Time
GROUP BY City, Day)
UNION ALL
(SELECT City, ‘ ’, SUM(Amnt) FROM Sales NATURAL JOIN Location
GROUP BY City)
UNION ALL
(SELECT ‘ ’, Day, SUM(Amnt) FROM Sales NATURAL JOIN Time
GROUP BY Day)
UNION ALL
(SELECT ‘ ’, ‘ ’, SUM(Amnt) FROM Sales);
```

# A Relational Representation of Pivoting

| City | Day | Sales |
|---|---|---|
| Auckland | Monday | *60* |
| Auckland | Tuesday | *60* |
| Auckland | Thursday | *140* |
| Auckland | Saturday | *880* |
| Nelson | Monday | *550* |
| Nelson | Friday | *100* |
| Wellington | Monday | *540* |
| Wellington | Tuesday | *300* |
| Wellington | Sunday | *50* |

| City | Day | Sales |
|---|---|---|
| Auckland | | *1140* |
| Nelson | | *650* |
| Wellington | | *890* |
| | Monday | *1150* |
| | Tuesday | *360* |
| | Thursday | *140* |
| | Friday | *100* |
| | Saturday | *880* |
| | Sunday | *50* |
| | | *2680* |

# *Summary of Basic OLAP Operations*

| OLAP operation | Use of attribute hierarchy | No of di-mensions | Use of aggregate functions | Use of $\sigma$ | May be replaced by |
|---|---|---|---|---|---|
| Roll-up | Y | Same or smaller | Y | N | SELECT… AGG GROUP BY |
| Drill-down | Y | Same or greater | May use | N | SELECT… AGG GROUP BY |
| Slice | N | Smaller | N | Y | SELECT… WHERE cond |
| Dice | N | May be Smaller | N | Y | SELECT… WHERE cond |
| Pivot / cross tab | Y | Smaller | Y (on many Subsets) | N | Multiple roll-up |