# COURSE NAME:
# DATA WAREHOUSING & DATA MINING

# LECTURE 13
# TOPICS TO BE COVERED:

- DMQL – Data Mining Query language
- Data specification
- Specifying knowledge
- Hierarchy specification
- Pattern presentation & visualisation specification
- Data mining languages and standardisation of data mining.

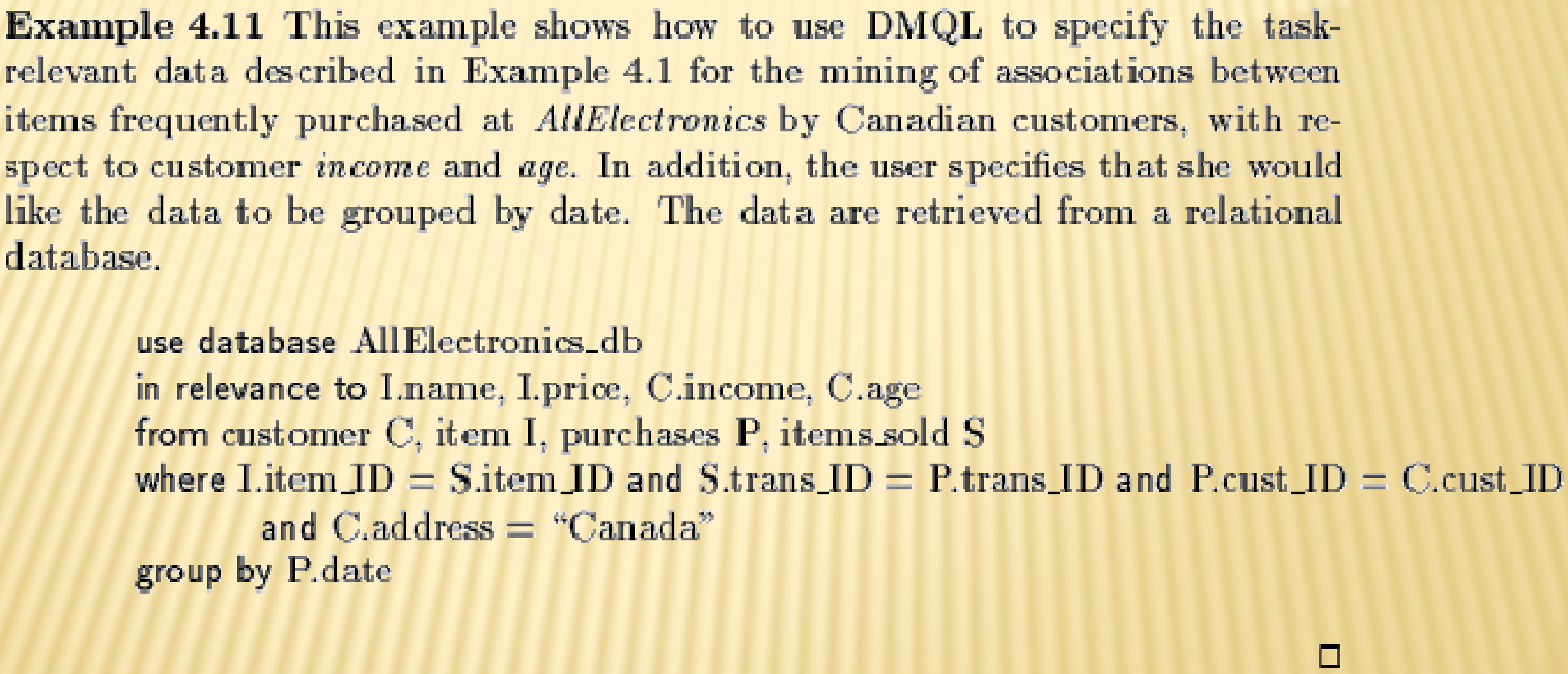
# SYNTAX FOR DMQL

- Syntax for specification of
  - task-relevant data
  - the kind of knowledge to be mined
  - concept hierarchy specification
  - interestingness measure
  - pattern presentation and visualization
- Putting it all together — a DMQL query

# SYNTAX FOR TASK-RELEVANT DATA SPECIFICATION

- *use database* database_name, or *use data warehouse* data_warehouse_name
- *from relation*(s)/cube(s) [*where* condition]
- *in relevance to* att_or_dim_list
- *order by* order_list
- *group by* grouping_list
- *having* condition

# SPECIFICATION OF TASK-RELEVANT DATA



**Example 4.11** This example shows how to use DMQL to specify the task-relevant data described in Example 4.1 for the mining of associations between items frequently purchased at *AllElectronics* by Canadian customers, with respect to customer *income* and *age*. In addition, the user specifies that she would like the data to be grouped by date. The data are retrieved from a relational database.

```
use database AllElectronics_db
in relevance to I.name, I.price, C.income, C.age
from customer C, item I, purchases P, items_sold S
where I.item_ID = S.item_ID and S.trans_ID = P.trans_ID and P.cust_ID = C.cust_ID
          and C.address = "Canada"
group by P.date
```

□

# SYNTAX FOR SPECIFYING THE KIND OF KNOWLEDGE TO BE MINED

- Characterization

  Mine_Knowledge_Specification **::=**

  ***mine characteristics* [*as* pattern_name]**
  ***analyze* measure(s)**

- Discrimination

  Mine_Knowledge_Specification **::=**

  ***mine comparison* [*as* pattern_name]**
  ***for* target_class *where* target_condition**
  **{*versus* contrast_class_*i* *where* contrast_condition_*i*}**
  ***analyze* measure(s)**

- Association

  Mine_Knowledge_Specification **::=**

  **mine associations [as pattern_name]**

# SYNTAX FOR SPECIFYING THE KIND OF KNOWLEDGE TO BE MINED (CONT.)

- Classification

  Mine_Knowledge_Specification ::=
  *mine classification* [*as* pattern_name]
  *analyze* classifying_attribute_or_dimension

- Prediction

  Mine_Knowledge_Specification ::=
  *mine prediction* [*as* pattern_name]
  *analyze* prediction_attribute_or_dimension
  {*set* {attribute_or_dimension_*i*= value_*i*}}

# SYNTAX FOR CONCEPT HIERARCHY SPECIFICATION

- To specify what concept hierarchies to use

  use hierarchy **<hierarchy>** for **<attribute_or_dimension>**
- We use different syntax to define different type of hierarchies
  - schema hierarchies

    define hierarchy **time_hierarchy** on **date** as **[date,month quarter,year]**
  - set-grouping hierarchies

    define hierarchy **age_hierarchy** for **age** on **customer** as

    **level1: {*young, middle_aged, senior*} < level0:** all

    **level2: {20, ..., 39} < level1: *young***

    **level2: {40, ..., 59} < level1: *middle_aged***

    **level2: {60, ..., 89} < level1: *senior***

# SYNTAX FOR CONCEPT HIERARCHY SPECIFICATION (CONT.)

- operation-derived hierarchies

  define hierarchy **age_hierarchy** for **age** on **customer** as

  **{age_category(1), ..., age_category(5)} := cluster(default, age, 5) <** all**(age)**

- rule-based hierarchies

  define hierarchy **profit_margin_hierarchy** on **item** as

  **level_1: low_profit_margin < level_0:** all

  **if (price - cost)< $50**

  **level_1: medium-profit_margin < level_0:** all

  **if ((price - cost) > $50) and ((price - cost) <= $250))**

  **level_1: high_profit_margin < level_0:** all

  **if (price - cost) > $250**

# SYNTAX FOR INTERESTINGNESS MEASURE SPECIFICATION

- Interestingness measures and thresholds can be specified by the user with the statement:

  with **<interest_measure_name>** threshold = **threshold_value**

- **Example:**

  **with support** threshold **= 0.05**

  **with confidence** threshold **= 0.7**

# SYNTAX FOR PATTERN PRESENTATION AND VISUALIZATION SPECIFICATION

- We have syntax which allows users to specify the display of discovered patterns in one or more forms

  display as <**result_form>**

- To facilitate interactive viewing at different concept level, the following syntax is defined:

  Multilevel_Manipulation ::= *roll up on*<br>attribute_or_dimension<br>| *drill down on*<br>attribute_or_dimension<br>| *add* attribute_or_dimension<br>| *drop* attribute_or_dimension

# PUTTING IT ALL TOGETHER: THE FULL SPECIFICATION OF A DMQL QUERY

```
use database AllElectronics_db
use hierarchy location_hierarchy  for B.address
mine characteristics as  customerPurchasing
analyze  count%
in relevance to C.age, I.type, I.place_made
from  customer C,  item I, purchases P, items_sold S, works_at W,
   branch
where I.item_ID = S.item_ID  and S.trans_ID = P.trans_ID
       and P.cust_ID = C.cust_ID and P.method_paid = ``AmEx''
       and P.empl_ID = W.empl_ID and W.branch_ID = B.branch_ID
       and B.address = ``Canada''  and I.price >= 100
with noise threshold = 0.05
display as table
```

# OTHER DATA MINING LANGUAGES & STANDARDIZATION EFFORTS

- Association rule language specifications
  - MSQL (Imielinski & Virmani'99)
  - MineRule (Meo Psaila and Ceri'96)
  - Query flocks based on Datalog syntax (Tsur et al'98)
- OLEDB for DM (Microsoft'2000)
  - Based on OLE, OLE DB, OLE DB for OLAP
  - Integrating DBMS, data warehouse and data mining
- CRISP-DM (CRoss-Industry Standard Process for Data Mining)
  - Providing a platform and process structure for effective data mining
  - Emphasizing on deploying data mining technology to solve business problems

# DESIGNING GRAPHICAL USER INTERFACES BASED ON A DATA MINING QUERY LANGUAGE

- What tasks should be considered in the design GUIs based on a data mining query language?
  - Data collection and data mining query composition
  - Presentation of discovered patterns
  - Hierarchy specification and manipulation
  - Manipulation of data mining primitives
  - Interactive multilevel mining
  - Other miscellaneous information

# A Data Mining Query Language, DMQL: Language Primitives

How can you define following schema in DMQL

- Fact Table
- Dimension Table
- Star Schema
- Snowflake Schema
- Fact Constellation

# A DATA MINING QUERY LANGUAGE, DMQL: LANGUAGE PRIMITIVES

- Cube Definition (Fact Table)

  define cube <cube_name> [<dimension_list>]: <measure_list>

- Dimension Definition ( Dimension Table )

  define dimension <dimension_name> as (<attribute_or_subdimension_list>)

- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

# DEFINING A STAR SCHEMA IN DMQL

define cube sales_star [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

# DEFINING A SNOWFLAKE SCHEMA IN DMQL

define cube sales_snowflake [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city(city_key, province_or_state, country))

# DEFINING A FACT CONSTELLATION IN DMQL

```
define cube sales [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars), avg_sales =
          avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state,
   country)
define cube shipping [time, item, shipper, from_location, to_location]:
        dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location
   in cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
```

# MEASURES: THREE CATEGORIES

- distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning.
  - E.g., count(), sum(), min(), max().
- algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function.
  - E.g., avg(), min_N(), standard_deviation().
- holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank().