DM Final – Jisoo Lee

**<Data>: stored representations of meaningful objects/events**

1. structured: organized str. #, dates, forms

2. unstructured: free from, images, audio, video

Big Data: volume/variety/velocity

**<Info>: processed/analyzed data, report, sum, analytics, visual**

**<Knowledge> >> <Decision Making> >> <Action> >> <Data>**

**<Data Management>: better allocation of attention to support effective decision making/strategy**

Database: collection of data related/managed as single unit

DBMS: Database Mgmt System; software managing user DB

- Solve problems with file-based processing; (1) program-data dependence; (2) limited data sharing

- ☺ for large data; program-data independence; layers of independence; planned data redundancy; improved data consistency; improved data sharing; increased productivity of application development; improved data quality; reduced program maintenance

Metadata: data describing properties of end-user data; data about data

Data Model: graphics capturing nature/relationship among data; ER

**<Systems Development Life Cycle - 6>**

Planning >> Analysis (conceptual) – ER Modeling >> Logical Design – Relational data model & Normalization >> Physical Design >> Implementation – SQL >> Maintenance

**<ER: Entity-Relationship>**

**Entity**: represents data

- strong: independent on other entities; identifier is self-contained and does not reference an attribute of another entity

- weak: dependent on other entities; identifier corresponds to the identifier of strong entity (Double lines)

**Attributes**: property of entity or relationship type

- Required attributes: required value for each instance of entity

- **Identifier**/**Key**: unique/not null value; uniquely identifies individual instances of entity

**Relationship**: relationship associates entities; relationship instances associate entity instances

- Degree: num of entities that participate in relationship; unary, binary, ternary

**Cardinality** **Constraint**: specifies num of instances of target entity that can be associated with each instance of source entity

- many to many/ternary >> associative entity

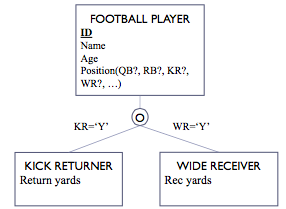
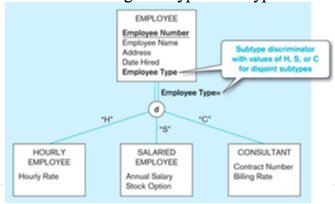
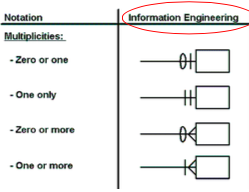
**Supertype/subtype** (Generalization/Specialization)

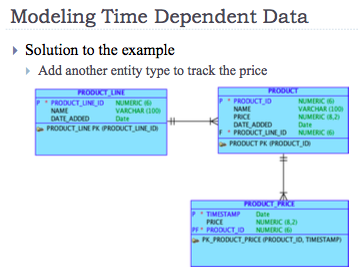
- subtype: subgrouping of entities sharing common attributes distinct from other subgroupings

- subtype discriminator attribute: attribute of supertype; its value determines target subtype

- **Completeness**: **Total** (Double lines) vs. **Partial** (Single line): instance of supertype must vs. may or may not correspond to instance of one of subtypes

- **Disjoint** vs. **Overlap**: instance of supertype cannot vs. can be member of more than one subtype -> for disjoint, can add one attribute as discriminator; for overlap, use composite attribute whose subparts pertain to different subtypes, each subpart contains Boolean value to indicate whether instance belongs to associated subtype





**<Relational Data Model>**

**Relation**: named, two-dimensional table of data

**Primary** **Key**: unique attribute used to identify a single instance of entity

**Foreign** **Key**: attribute in relation that serves as PK of another relation in same DB

**Schema**: description of logical str of DB – text vs. graphic

**Integrity constraints**: limits acceptable values in DB; e.g. domain constraints (chr), entity integrity (PK), referential integrity (FK valid)

**<Normalization>**

:formal process for creating well structured relations so that redundancies and anomalies can be avoided

- 1st: No multivalued attributes (each cell has exactly one value)

- 2nd: Non-key attributes are identified by the whole PK (no partial functional dependencies)

- 3rd: Non-key attributes are identified by only PK (no transitive dependencies)

**<SQL: Structured Query Language>**

☺reduced training cost and higher productivity

app portability and cross-sys communication

1.Familiarize with data model, entities, relationship

2. understand result

3. select > from > relation > where > grup by/ having/ distinct/ not in/ in etc.

1.Understand how indexes are used in query

2.Keep optimizer statistics uptodate

3.Use compatible data types for fields

4. Write simple queries

5.Break complex queries into multiple simple parts

6. Don’t nest one query inside another query >union ..

7. Don’t combine a table with itself

8. Create temporary tables for groups of queries

9. Combine update operations

10. Retrieve only the data you need

11. Don’t have the DBMS sort without an index

12.Learn

13. Consider the total query processing time for ad hoc queries

-DDLC: Data Definition Lang Command; table

CREATE TABLE Student

(SID NUMBER(8,0) NOT NULL,

fName VARCHAR2(50),

lName VARCHAR2(50),

Dept NUMBER(3,0),

CONSTRAINT StuPK PRIMARY KEY (SID),

CONSTRAINT StuFK FOREIGN KEY (Dept) REFERENCES Dept(DID));

ALTER TABLE Student

ADD Gender CHAR(1);

DROP COLUMN Gender;

RENAME COLUMN Gender TO Sex;

MODIFY Gender VARCHAR2(6);

DROP TABLE Student CASCADE CONSTRAINTS;

-DMLC: Data Manipulation Lang Command; data

INSERT INTO Student

VALUES (1001, ‘Jisoo’, ‘Lee’, 123);

INSERT INTO Student (SID, fName)

VALUES (1001, ‘Jisoo’);

UPDATE Student

SET fName = ‘JS’

WHERE SID = 1001;

DELETE FROM Student <이것만 하면 all

WHERE SID = 1001;

SELETE CName, Cstate

FROM Customer\_T

WHERE (LOWER(CName) LIKE ‘%furniture%’ //String F

OR (CName) LIKE ‘%INTERIOR%’)

AND CState IN (‘CA’, ‘FL’, ‘IA’) //Membership Condition

ORDER BY CName DESC; //ASC default

SELETE \*

FROM Student

WHERE GradDate >= ’24-Oct-10’; //Comparison Operator

SELETE PID, PStdPrice\*.06 As Tax //Expression, Alias

FROM Product\_T;

//String Function

//LOWER, UPPER, INITCAP

//SUBSTR: extract a portion of a given string

@SUBSTR(Zipcode, 2 //start, 3 //2 end) = 50‘22’4

@SUBSTR(14.87,-3,2) >> .8

//CONCAT: combine two strings

//LTRIM(str, # or str): remove unnecessary blanks

@LTRIM(‘00012310’,’01’) >> ‘2310’

//LPAD(str, #, str) : LPAD(‘tech’,8,0) >>’0000tech’; LPAD(‘tech’,2)>>te

//Mathematical Function

//AVG, SUM for numeric data only

//ROUND(150.75,0) >> 151; ROUND(150.75,0,1) >>150

Q. Find the number of customers living in each postal code region, which is defined as the first three digits of the postal code.

SELECT SUBSTR(CustomerPostalCode, 1, 3) AS Region, COUNT(\*) AS NumberOfCustomers

FROM Customer\_T

GROUP BY SUBSTR(CustomerPostalCode, 1, 3);

￼

Q. Find the number of customers from each state, but only for those states which have more than 1 customer.

SELECT CustomerState, COUNT(\*) AS NumOfCust

FROM Customer\_T

GROUP BY CustomerState

HAVING COUNT(\*) > 1

ORDER BY NumofCust desc;

Q. Limit the number of rows to appear in ordered query results

SELECT \* from

(SELECT productstandardprice

FROM product\_t

ORDER BY productstandardprice DESC)

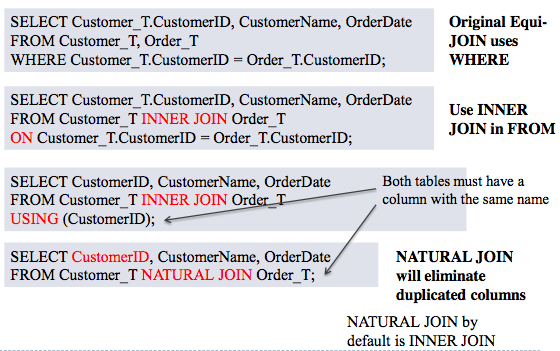
WHERE ROWNUM <= 5;

-DCLC: Data Control Lang Command; access, control

JOIN

-Cartesian: Ta 15 row \* Tb 10 row = 150 rows

-Equi/Inner



Q. Find the names of all of the customers that have purchased more than one product. Also, provide the number of unique products that such customers have purchased.

SELECT CustomerName, COUNT(DISTINCT ProductID) AS NumberOfProducts

FROM Customer\_T, Order\_T, OrderLine\_T

WHERE Customer\_T.CustomerID = Order\_T.CustomerID AND Order\_T.OrderID = OrderLine\_T.OrderID

GROUP BY CustomerName

HAVING COUNT(DISTINCT ProductID) > 1;

-Outer: Left, Right, Full

Q. Provide the details of customers who have never placed an order

SELECT Customer\_T.\*, OrderDate

FROM Customer\_T LEFT OUTER JOIN Order\_T

ON Customer\_T.CustomerID = Order\_T.CustomerID WHERE OrderDate IS NULL;

Q. Find names of students who take courses in the Pappajohn building without using JOIN

SELETE Name

FROM Student

WHERE SID IN (SELECT SID FROM Registration

WHERE CID IN (SELECT CID

FROM Course WHERE building = ‘PBB’));

-Self: a table with two dif alias

Q. Provide the ID and name of each employee along with the ID and name of their supervisor

SELECT E.EmployeeID, E.EmployeeName, E.EmployeeSupervisor AS SupervisorID, S.EmployeeName AS SupervisorName

FROM Employee\_T E, Employee\_T S

WHERE E.EmployeeSupervisor = S.EmployeeID;

Q. Find the age of those who follow BOTH 1001 and 1002

SELECT Userid, UAge

FROM User\_T

WHERE UserID IN (SELECT FID From Follow\_T where UserID=1001)

AND UserID IN (SELECT FID From Follow\_T where UserID=1002);

>> INTERSECT

Q. Provide the user ID and city of each user whose state and age are the same as those of user 1001

SELECT userID, city

FROM TW\_USER WHERE state = (SELECT state

FROM TW\_USER WHERE userID=1001)

and age= (SELECT age

FROM TW\_USER WHERE userID=1001)

AND userID!=1001;

SELECT userID, city

FROM TW\_USER

WHERE (state, age) =(SELECT state, age FROM TW\_USER WHERE userID=1001)

AND userID != 1001;

Q. Provide the ID, state, and age of each user whose age is more than the minimum age in her/his own state.

SELECT userID, state, age FROM TW\_USER

WHERE (state, age) NOT IN

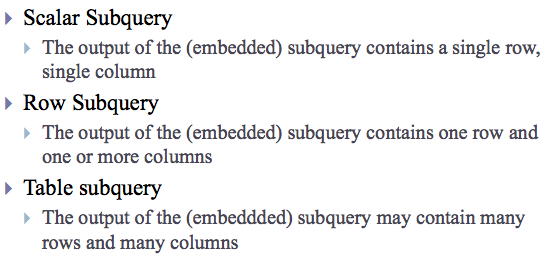
(SELECT state, min(age) FROM TW\_USER GROUP BY state);

Q. How much does the price of each product differs from the average price?

SELECT ProductStandardPrice-AvgPrice

FROM Product\_T, (SELECT AVG(ProductStandardPrice) AS AvgPrice FROM Product\_T);

SUB



Q.Find the Num of Students from each dept who have GPA 50% higher than the avg GPA?

SELECT Dept, COUNT(\*)

FROM Student\_t

WHERE GPA > (SELECT AVG(GPA)\*1.5

FROM Student\_t)

GROUP BY Dept;

Q. What are the name and address of the customer who placed order number 1008?

JOIN: SELECT CustomerName, CustomerAddress, CustomerCity, CustomerState, CustomerPostalCode

FROM Customer\_T, Order\_T

WHERE Customer\_T.CustomerID = Order\_T.CustomerID

AND OrderID = 1008;

SUB: SELECT CustomerName, CustomerAddress, CustomerCity, CustomerState, CustomerPostalCode

FROM Customer\_T WHERE CustomerID =

(SELECT CustomerID FROM Order\_T WHERE OrderID=1008);

**<Web and Databases>**

*World Wide Web*

-one of most pop app built to work over internet

-\*Web Client(Browser)

>> send/display \*HTTP req in \*HTML form

@IE, Firefox, Chrome, Safari..

>> \*Internet <<

-\*Web Server

<< handle/respond \*HTTP req

@Apache, MS IIS

-Static: Web Server + HTML pages

-App: Web + Server-side (App) + Database

*App on Cloud*

-Advantage

-managed by provider

-flexible resource assignment

-network accessible

-sustainable

-managed through self-service

-Disadvantage

-security

**<Physical Database Design>**

*Choosing Data Types*

-represent all possible values

-improve data integrity

-support needed manipulations

-minimize storage size (capacity of data type, precision & scale) @NUM(8,3)=12345.123

*Adding Tables for Coding*

-few unique values – adding coding save space

Conceptual Design: ER/EER

Logical Design: Relational schema

-systematic structure to store

-reduce redundancy

-avoid anomalies and improve integrity

Implementation: SQL

-data def/manipulation

Physical Design: Data file

-improve performance (faster access)

-enhance security

-easy backup and recovery

*Denormalization*

☺ balance

☹ cost of maintenance/ storage space

1.combine data of >= tables into 1 table

2.partition large table into smaller table

-horizontally

-divide one relation into multiple tables

-break relation into balanced partitions

-useful for when dif users need access to dif row

-Nrow = X+Yrow

-vertically

-primary key must be repeated in each file

-useful for when dif users need access to dif col

-Mcol+1 = X + Y col

☺efficiency: relative records grouped together

local optimization

security

less recovery and backup time

load balancing

☹inconsistent access speed

complexity: non-transparent partitioning

extra space or update time: duplicate

*File Organizations*

: org records in a table in physical files

-Sequential: records stored in sequence by pk values

-(time = N/2)

☺non-div

-Indexed: records with index to locate

-index: a data structure used to determine in a file the location of records that satisfy some condition-automatically pk

-(time = logN)

-larger table; index pk of each table; index search field (where, order by, group by); variety in index values >100; join index

@CREATE INDEX NameIDX ON C\_T(CustName)

@DROP INDEX NameIDX

-Hashed: physical address of record determined by hash algorithm

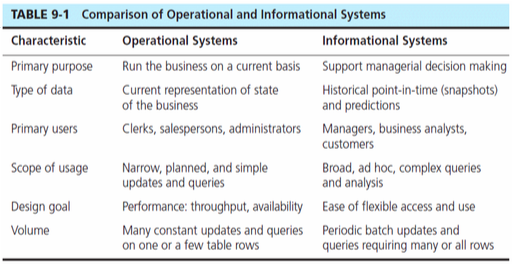
-algorithm: input(col value) >> output(index); records with same position are grouped in lists; need special mechanism to avoid conflicts

**<Data Warehouse>**

:Subject-oriented, integrated(consistent naming), time-variant(study trend and change), non-updatable(read-only, refreshed) collection of data used in support of management decision-making processes

-operational system: (system of record) a sys used to run a biz in real time, based on current data

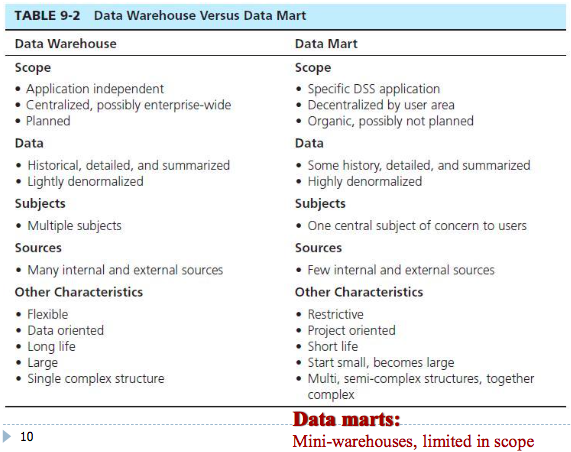
-informational system: a sys designed to support decision making based on historical point-in-time and prediction data for complex queries or data-mining app

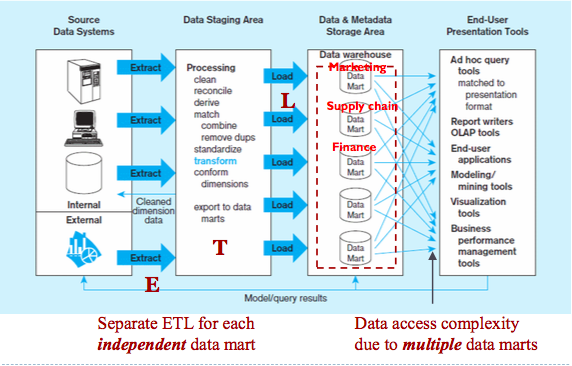


-Operation>>informational: gather(extract)>process(transform)>load data(ETL)

-architecture

1.Indipendent Data Mart





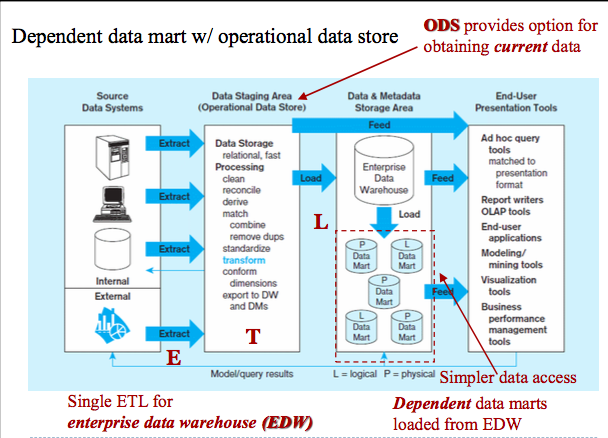
☹separate ETL process for each DM(redundancy)

data inconsistency

difficult to relate data from dif DM

scalability issue when new apps needed

2.Dependent data mart w/ operational data store



Data in DW

-Status Data

-Event Data: for a given time period

-event: a DB action (create/updata/delete) that results from a transaction

-Transient: care only about now – destroying the previous data content

-periodic: keep all historical data, need significant more space

Advance in DB/DW

-NoSQL: Not only SQL; deal with unstructured data

-OLAP: online analytical processing

**<Data Mining>**

-Queries

@Find all credit applicants with last name of Smith.

@Identify customers who have purchased more than $10,000 in the last month.

@Find all customers who have purchased milk

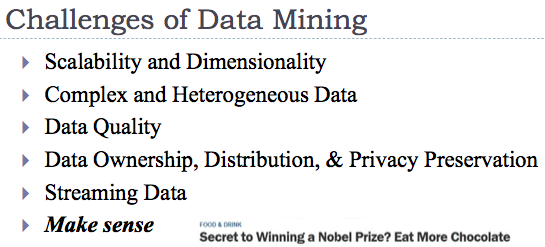
-Data Mining

-core of knowledge discovery of hidden information from data(explore the data, find patterns, make prediction)

-dw facilitates data mining, can be applied to data from other sources

@Identify customers with similar buying habits.

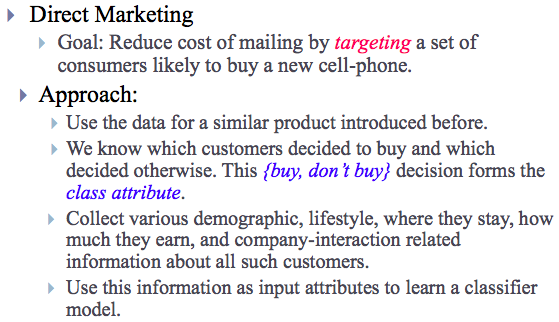
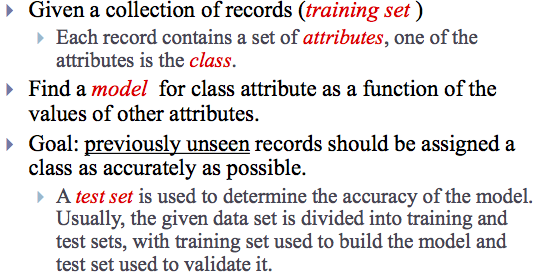
@Find all items which are frequently purchased with milk.

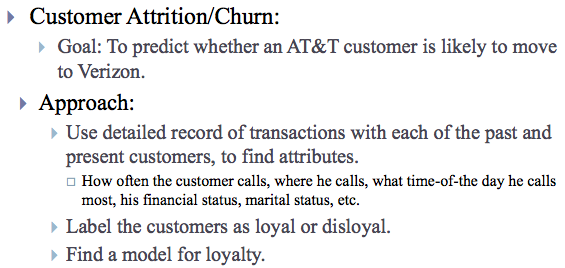


-Task

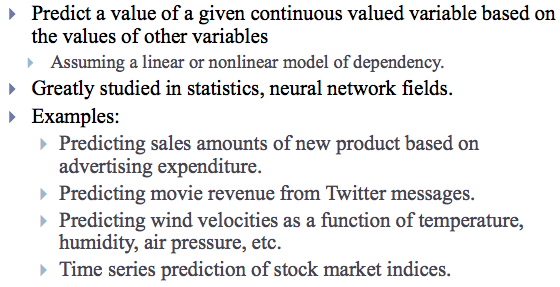
-Prediction: use variables to predict unknown future values

-classification





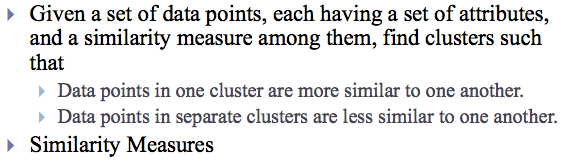
-regression

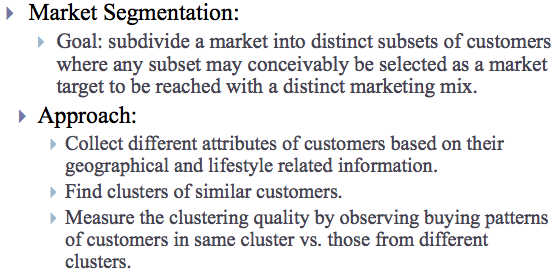


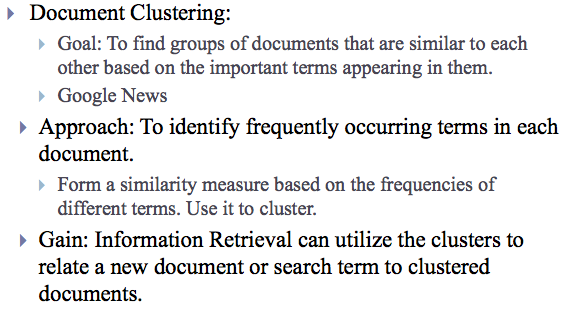
-deviation detection

-Description: find human-interpretable patterns describing the data

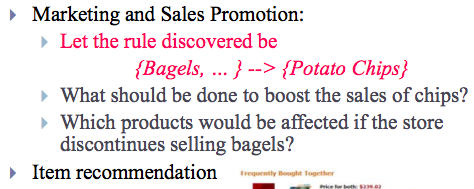
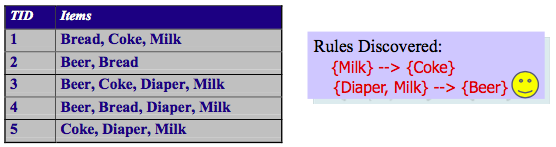
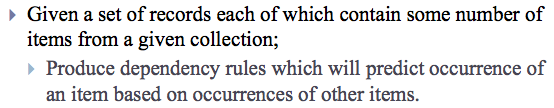
-clustering







-association rule discovery



-sequential pattern discovery