**CHAPTER 1** -- Data object represents an entity with relationships (1:1,1:M…), **Data objects** (rows) **described by attributes** (columns) - Types of attributes: **nominal**: categories, name of things, i.e occupation, zip code, hair color…, **ordinal**: values have meaningful rankings, i.e army rankings, grades, size={small,m,L)... **binary**: only has 2 states, **symmetric** (gender) vs **asymmetric** (medical test positive/negative) can lead to imbalanced data… **numeric**: quantity, **interval** (temperatures in C or F, calendar dates...) vs **ratio** (length, counts, monetary...) -**Discrete** (finite/countable infinite set of values)vs **continuous** (real numbers, i.e. temperatures, height, weight...) attributes. We often convert these to bands for data analysis. -Descriptive **data summarization**: general idea to get an overall picture of your data. See how it’s distributed. 1) **initial steps**: basic statistical analysis, visualization, 2) gives us a feeling of our data: relationships, patterns, trends

**CHAPTER 2: Online analytical processing** (OLAP):, data uploaded periodically, **queries**: aggregates (sum, max, min, count, average), trends and icebergs. **Data warehouse** is based on a multidimensional data model which views data in the form of a **data cube** -**FACT table** contains measures and keys to the related dimension

-why a separate data warehouse? -high performance for both systems, -different functions and data: ,– missing data: Decision support requires historical data which operational DBs do not typically maintain – data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources – data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

-**data warehouse**: 1. a decision support database that is maintained separately from the organization’s database. 2. Support information processing by providing a solid platform of consolidated, historical data. -**data warehousing**: the process of constructing and using data warehouses. -data warehouse (DW) subject oriented: organized around major subject (customer, products, sales) Focused on modelling and analyzing of data for decision makers Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

-DW consists of one or more **data marts**. Data marts corresponds to a subject.

-DW integrated: **1**. constructed by integrating multiple, heterogeneous data sources. **2**. data cleaning and data integration techniques are applied (naming convention, units, encoding structures)

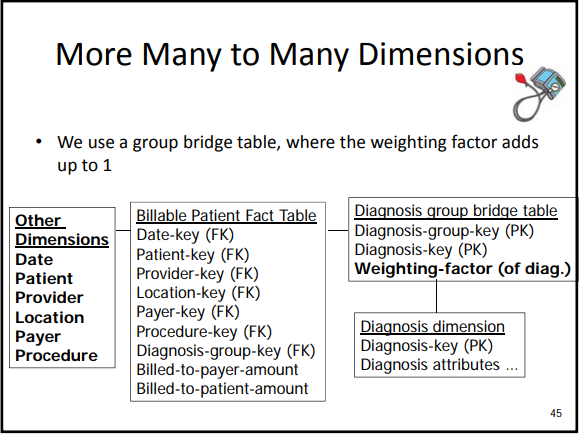
-DW time variant: **1**. Time horizon for the data warehouse is significantly longer than that of operational systems. Operational database=current value data. DW data=information from a historical perspective (5-10years) **2**. Every key structure in the DW contains an element of time, not necessarily the case for operational DB.

-DW nonvolatile: **1**. a **physically separate store** of data transformed from the operational environment **2**. operational **update of data does not occur** in the data warehouse environment, – Does not require transaction processing, recovery, and concurrency control mechanisms, – Requires only two operations in data accessing (initial loading of data)

-building data warehouse: -**dimensional modeling,** -physical design, - data staging (ETL)

-**Dimensional modeling**: -FACT table: primary table where performance measures for business process are stored (composite PK from many FK, facts: business measure (numeric and additive). -DIMENSIONAL tables: contains textual description of business, MANY dimensional attributes, used to specify query constraints.

**4 steps to designing an individual dimensional model**: **1**. Chose the business process to model, **2.** Declare the grain: How to describe a single row in the fact table?, **3.** Choose the dimensions: How do business people describe the data that results from the business process?, **4.** Choose the fact: What are we measuring? Usually numeric and additive

Types of dimensions: -**Causal**: promotion, contract, deal, etc. **Multiple** date or timestamp: date shipped, date received, etc. **Degenerate**: ticket number, order number • Role‐playing: one table acting in many “views”, **Status**: account status, **Audit**: data quality and record lineage (“when the record was loaded for the first time”), **Junk**: indicators and flag 

-attribute banding: used to answer “banded queries”, use of > and <=. Used to Avoid monster dimensions!

-Many to many dimension: use of bridge table, i.e. see blow

-snowflaking, happens when we try to normalize the dimension. AVOID AS FAR AS POSSIBLE because of high cost of JOIN operation!

**CHAPTER 3:** -**aggregates** (cube/cuboids) are used to speed up frequent queries -One **cuboid** corresponds to one aggregate, -data are **summed** using **concept hierarchies**

-pre-calculated and pre-stored summaries that are stored in DW. -used for query optimization when doing OLAP operations

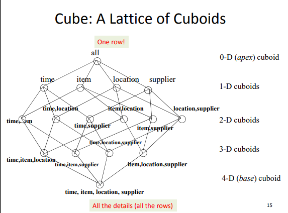
-Aggregate will periodically, dynamically change since it depends on the frequent queries (frequent business requests, statistical distribution of data)

-data mart = based dimensional model+aggregate dimensional model

-Each aggregate occupies its own fact table. -**aggregate navigator:** choose the right cuboid, transparently intercepts the end user code and uses the best aggregate

How does it work? 1. Rank order all the aggregate fact tables for the smallest to the largest. (Cuboids) 2. Find the smallest aggregate fact table and proceed to step 2. 3. For the smallest, see if all the dimensional attributes of the query can be found. 1. If yes, we are done. 2. If not, find the next smallest aggregate fact table and retry step 2. 4. Execute the altered SQL. (If no aggregate fact tables found, use the Base Cuboid.)

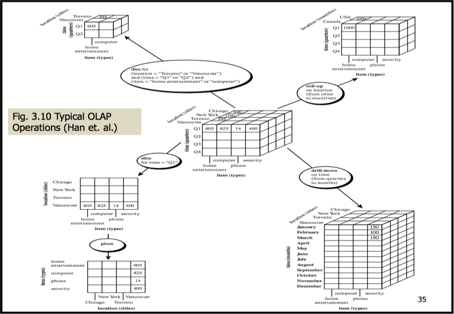
-an n-D base cube is called a **base cuboid** -top most 0-D cuboid which holds the highest level of summarization is called the **apex** cuboid

-**Lattice** of cuboids forms an OLAP data cube.

**-typical OLAP operations**: **roll-up** (climbing up hierarchy), **drill-down** (higher level to more detailed), **slice and dice**, **pivot**(rotate, reorient cube…)

-Benefits: performance gain, not much extra data storage, benefit all users, transparent to users, low impact on data staging and on DBA’s workload

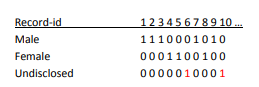
-draw back: what aggregate to materialize (store)?, what to aggregate? Business requirements change all the time…

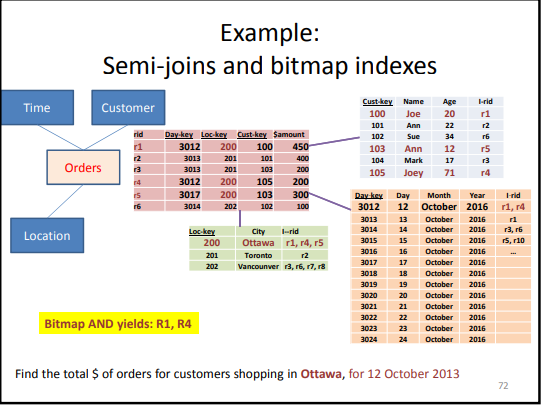
**-Recipe for aggregates:** **1**. Identify set of frequent queries **2**. Identify concept hierarchies used (in queries in 1) **3**. Determine levels in concept hierarchies to be used to speed up the queries (month, year)? **4**. Decide on initial set of aggregates **5**. If your system allows: a) Implement aggregate strategy and aggregate navigator (e.g. write the code) (or) b) Verify appropriateness of actual aggregates used in OLAP cube engine (if allowed by system) **6.**Monitor adapt

-Major difference between logical/physical model: **data types, table segmentation**, **buffer size**, **disk page size**, **table organization**… Logical (dimensional) Physical actually choosing the data types (varchar), if nulls, pk? Etc.

-Indexing: **-B+ tree indexes on primary keys**, **Clustered versus unclustered** (generally cluster on date key),

 **Bitmap indexes**, -Indexes for n‐way joins (**star joins**) good if small, can be compressed. Take intersection for query.

-sort merge steps: Joins are expensive. Sort R and S on the join column • Scan R and S to do a ``merge’’ (on join col.), and output result tuples. ‐ Advance scan of R until (current R‐tuple >= current S) tuple, Then advance scan of S until (current S‐tuple >= current R) tuple; Do this until (current R tuple = current S) tuple. – At this point, all R tuples with same value in Ri (current R group) and all S tuples with same value in Sj (current S group) match; Output for all pairs of such tuples. – Then resume scanning R and S. • R is scanned once; each S group is scanned once per matching R tuple. • Multiple scans of an S group are likely to find needed pages in buffer.

****-total cost for sort-merge join = N log N + M log M + (M + N) I/Os

-Total cost for n-way sort-merge join **HUGE! Solution → star joins**: goal is to reduce the number of rows from the dimension prior to joining, -use idea of **selectivity** from query optimization, -use of reducers: bitmap, semi-join, inverted indexes, **example**: Select sum(totalorders) From Where date = today And city = ‘Ottawa’ And Joining Date=today instead of 10 years! -**semijoin**: only returns the id that match the query in each dimension

-inverted index: index on one or more attribute (example of document from information retrieval)

**CHAPTER 5:** Goal of data staging: get the **right** data from source A (sources) to source B (data marts).

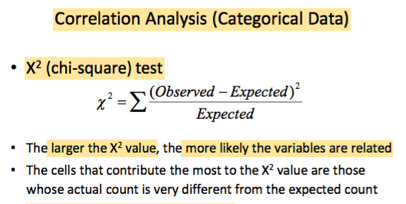
Steps for data staging:

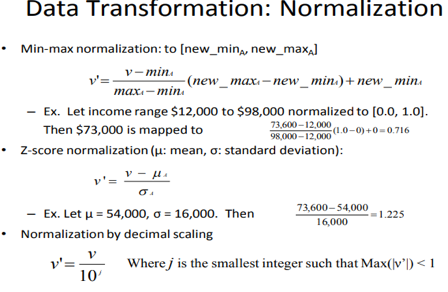
**A1.** **High-level planning:** -Create a very high-level, one-page schematic of the source-to-target flow, -Identify starting and ending points, -Label known data sources, -Include placeholders for sources yet to be determined, -Label targets, -Include notes about known problems, **A2.** **Choose data staging tool**: all major data vendor offers one **A3**. **General planning: extraction** from multiple sources (timing, information fusion), archiving (when?), data quality management, change management(when?how?),**A4**. **Detailed planning by table**: -Drill down by target table, graphically sketching any complex data restructuring or transformations, -Identify attribute hierarchies (normalize the source), -Graphically illustrate the surrogate-key generation process, -Develop a preliminary job sequencing, -unbucketize, -first historic draft load,-For each dimension processing, Use surrogate keys (autonumbers), not uniqueID given, -final data link dimension and fact table through surrogate keys

**B1.Develop one time historic load**:-build and test the historic dimension table loads, -build and test the historic fact table loads **including** surrogate key lookup and substitution **B2.** **Populate dimension table**, -**surrogate key** assignment, -validation relationship, -load (bulk loader, aggregations…),-index management (drop and re-index),**B3**. **Historic load of Atomic-level DM**, -fact table processing (surrogate key lookup), -ensure referential integrity, **Pipelining**: replace id with surrogate key

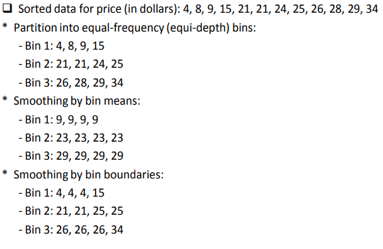
Transforming the data for data mining and analytics: -flag normal, anormal, impossible, out of bounds facts, -conversion (text to numeric…), -map values into ranges, Steps to transform the data:**1. Data cleaning**: why? Eliminate noise (random error or variance in a measured variable), incomplete, inconsistent, remove duplicates…

Considered the number one problem in data science

-handle missing values by: ignoring, use default value (unknown), use mean value, use most probable value. **PROBLEM: may not be correct...,** Eliminate anomalies. -handle noise: binning (place data in buckets of neighbors) sort data and partition into (equal-frequency) bins then one can smooth by bin means, smooth by bin median. use cluster to find outlier. Should involve human inspection…**2. Data integration and information fusion**: -top-down (schema level) vs bottum-up (data driven) vs hybrid, Goal is to keep original information as far as possible,



PT2



**Always requires human insp (PT1)**

ex:

-attribute construction: use ranges. domain dependant

**3. Data reduction:** -no need to keep similar attributes, i.e. only need to keep one of age and date-of-birth: **-sampling**: obtain small sample S to represent data set N. If not carefully designed, then sampling may not reduce database I/Os (page at a time) use stratified sampling or else can be skewed -**sampling with replacement vs without replacement** (doesn’t represent the data as well)

-Handle data change: -**Type 0**: no change,-**Type 1**: overwrite (ex mispelled name Ann -> Anne), -**Type 2**: add new row (someone changed city but we want to keep all records, could an we attribute “current” or “effective-date”), -**Type 3**: keep history\_dimension (change in marital status want to keep history, i.e single to married),-**Type 4**: add history table to track customer history, **requires to update keys, 4.Last data staging step: automation:** -typical operational functions (logging, monitoring, job definition flow dependency…), -record metadata, -ensure data quality, -disk space management procedures...

**PtA.Q1:** 1. Explain what ordinal data are and give an example of ordinal data contained in the table. **PtA.A1:** Ordinal data values have a meaningful order (ranking) but magnitude between successive values is not known. In this case, we have size = {compact, intermediate, full-size}.

**PtA.Q2:** Show the steps you would follow to determine whether the distribution of the prices is symmetric or skewed.**PtA.A2:**  Here, you need to calculate the mean, median and mode. If they are the same, then the distribution is symmetric. Otherwise, it is skewed. (calculate mean median mode) **PtA.Q4:** The above table contains some missing values, notably for the Make attribute. Explain the approach that would be the best to use, when handling such missing values for a Car Dealer, and be sure to motivate your answer. **PtA.A4:** Missing values such as these provide us with many “headaches” when we wish to conduct analytics. The first reason is that the results of aggregate functions such as count(), sum() and so on are now uncertain. Second, for machine learning, there is the possibility to generalize over missing values. However, if the number is large, then a machine learning algorithm will have difficulty to construct accurate models. Most logical thing to do is inspect cars personally **PTB.Q1:** Explain what the grain of a data mart is and declare the grain of the data mart you will design **PTB.A1:** The grain of the data mart refers to the level of detail that we store data. Specifically, it refers to the level of details of a single row in the Fact table. In this case, the grain is a Sale of a single Car to a Customer on a particular Day, as sold by a Salesperson. **PTB.Q2:** Explain what a fact/measure is and identify one fact/measure that you will include in your data mart. **PTB.A2:** A fact/measure is an additive (and often numeric) value or measurement that is stored in the Fact table. It is mainly used for aggregation and subsequent analytics. In this case study, the “dollar-amount” is an example of a fact/measure. Others could be the “cost-price” or the “profit-made”. **PTB.Q4:** Explain the benefit of aggregates and provide one aggregate that would aid *Bumpy Used Car Dealership* in their data analytics. This choice should be based on the above‐mentioned “potential queries”. **PTB.A4:** Aggregates (also know as Cubes) are mainly used in order to improve query performance. They are pre-calculated and often implemented as materialized views from queries. Specifically, they involve SUM() and GROUP BY operations. In this case, any one of the following aggregate s would potentially speed up performance (based on the frequent queries): **a**. Monthly Sales **b**. Seasonal Sales c. Sales per Car Maker (e.g. BMW versus FIAT) **PTB.Q5:** Explain, (a) what snowflaking is, and, (b) why it should be avoided **PTB.A5:** Snowflaking means that we normalize a dimension into two tables, and then link them using foreign keys. For example, we could split the CAR dimension and store details about the MANUFACTURER separately. It is not such a great idea, because it involves the joining of tables and (as shown in class) this is a very expensive database operation. The one place where we would use snowflaking is to avoid “monster” (i.e. very huge) dimensions. **PTB.Q6:** Explain, (a) what a surrogate key is, and, (b) why it should be used in all data marts.**PTB.A6:** A surrogate key is an auto-number that is used as a primary key in the data mart. The reason for using this is that we wish to avoid using production keys that has a “meaning” in the operational system. Production keys may be changed by the source system and also even be reused. If this happen, it may cause inconsistencies in our data mart. For example, consider we have a Meet-number that is re-used. This may lead to data from two totally separate Meets being considered as one. Clearly, this will lead to inconsistencies in the data. **PTB.Q7:** Suppose that your data mart contains a Customer dimension that includes the Gender attribute. Illustrate, by means of your own example, how you would use a bitmap index to speed up the queries against the Gender attribute. **PTB.A7:** The bitmap index is created during data staging and would look something like this (see diagram): In this case, we know that we only have to consider the records with “1”s in our query. Suppose we have a query that calculates the total dollar-amount (SUM()) of cars that were purchased by Customer where Gender = “Undisclosed”. In this toy example, we know that we only have to consider records the CustId6 and CustId10. These records are selected and a semi-join is performed to link it to the FACT TABLE. Here, the FACT TABLE rows with CustId6 and CustId10 are retrieved and the query proceeds to calculate the SUM(). This can speed up our queries, especially if we use hardware to calculate the AND operations  **Create a very high-level, one-page schematic of the source-to-target flow.** Identify starting and ending points. Label known data sources. Include placeholders for sources yet to be determined. Label targets. Include notes about known problems**. The surrogate key pipeline** refers to the data staging steps we take to convert the Fact table from the transactional input data format to the dimensional model. Refer to slide 26 of the Data Staging nodes, as well as the textbook by Kimball et. al. **Pearson’s Coefficient. Alternatively, one could draw a scatter plot and see if there are any strong correlations between the two dimensions. Attribute banding** refers to the process of discretizing numeric data into groups that are then useful for decision support. This is also sometimes done to avoid so-called monster dimensions. Examples in this data mart could be to convert the dates of birth of members into age ranges, the exact time of purchase to hourly bands, the exact prices of products into price bands, and so on.

**Finding outliers and Big data:** Outlier**:** data object that deviated significantly from the normal objects as if it were generated by a different mechanism. DIfferent than noise. Noise is random error and should be removed before outlier detection. One object may belong to more than one type of outlier.

->Global Outliers: significantly deviates from the rest of the data set. Issue: find appropriate measurement of deviation.

->Contextual outliers: deviates significantly based on a selected context. (temperature of 25c in Ottawa during winter). Generalization Of local outliers. Divided into two groups: contextual attributes and behavioural attributes. Issue: how to define meaningful context.

-->Mining contextual outliers: In some cases, context is unclear. Model the normal behaviour wrt to contexts. -Training model - train a model that predicts the expected behaviour attribute values wrt to contextual attr. -obj contextual attr if its behavior attribute values significantly deviate from values predicted by model. -Using prediction model that links contexts & behaviours:avoid explicit identification of specific contexts. Methods: Regression, Markov, Finite State Automaton

->Collective Outliers: subset of data objects collectively deviate significantly from the whole data set, even if the individual data objects may not be outliers. Need to consider behaviour of groups of objects.

-->Mining Collective Outliers: Models the expected behaviour of the structure units directly.COllective outlier detective subtle due to the challenge of exploring structures in data. Exploration uses heuristic and may be application dependent. Methods to solve; **Regression**, **Markov Model, Finite State Automaton**

Outlier detection methods: **1.** **Statistical methods:** assume normal data follow statistical model. Use gaussian distribution to model normal data. Estimate probability that the object fits the Gaussian distribution. Effectiveness depends on whether assumption of statistical model holds in the real data. **2. Parametric method**: *2.1 Grubbs test*: univariate outlier detection. Non-parametric: using histograms: problem: choosing the bin size. Adapt kernel density estimation to estimate the probability density of the data. If  high then normal, otherwise outlier. **3. Proximity-based** approaches: objects that are far from the others are outliers. Distance-based outlier detection: outlier if its neighbourhood does not have enough other points. Density-based outlier detection: outlier if density is relatively much lower than that of its neighbours. Local outliers: comparing to their local neighbourhoods, instead of the global data. Density around an outlier is different from the density around its neighbours. **4. Clustering-based:** normal data belong to large and dense clusters, outliers belong to small or sparse clusters -- or none at all. **5. Classification-based:**Outlier detection is fast. Quality depends on the availability and quality of the training set. **5.1 One-class Model**: train a classification model that can distinguish “normal” data from outliers. Classifier built to described only the normal class. Learn the decision boundary of the normal class using classification methods. Any sample that does not belong to normal class are outliers. Can detect new outliers that may not appear close to any outlier objects in the training set.  **5.2Semi-supervised learning**: combining classification-based and clustering-based methods.  Using clustering based approach, find a large cluster C, and small cluster C1. Treat all objects in C as normal. Then use one-class model to identify normal objects in outlier detection. Since some objects in C1 outlier, label all as outliers. Any object that doesn’t fall into model for C, considered an outlier as well. Strength: fast outlier detection. Bottleneck: Quality heavily depends on availability and quality of training set, but often difficult to obtain representative and high quality data.

->Outliers in high dimension: without saying why they are outliers is not useful in high-d due to many features are involved in high-dimensional data set. Data in high-d are often sparse. Distance between objects are dominated by noise as the dimensionality increases. Data Subspaces: adaptive to the subspaces signifying the outliers. Scalable with respect to dimensionality. # of subspaces increases exponentially. →Finding outliers in subspaces: Find outliers in  much lower dimensional subspaces: easy to interpret why and to what extend the object is an outlier.