Matt Miller & Kai Blumberg
HW #3
INFO 523

**4.2 Briefly compare the following concepts. You may use an example to explain your point(s).**

**(a) Snowflake schema, fact constellation, starnet query model**

A snowflake schema contains a fact table, which contains keys to different dimension tables. Each dimension table stores data about each attribute (sales, time, etc.) for each key. The snowflake schema allows keys to other dimension tables to be stored in dimension tables, so the model grows outward like a snowflake. While a snowflake schema has only 1 fact table, a fact constellation has multiple fact tables, with dimension tables shared between the fact tables. A fact constellation can also have dimension tables with keys to other dimension tables, like in the snowflake schema. A starnet query model uses a concept hierarchy, unlike the snowflake and fact constellation, to organize its data. Each attribute is grouped into different levels of a hierarchy, with each attribute representing a dimension in the model, and each level of the hierarchy known as a footprint. Starnet allows users to do typical OLAP operations such as roll-up or drill-down, on the hierarchy and on the data's dimensions, while the snowflake and fact constellation schemas can only do OLAP operations on the dimensions, either increasing or decreasing the number of dimensions observed.

**(b) Data cleaning, data transformation, refresh**
All three of these techniques are used to transfer data from operational databases to the data warehouse. Data cleaning identifies missing, noisy, or erroneous and attempts to fix the data, or throw it out if it is unable to do so. On the other hand, data transformation converts data from the operational databases, which can be in many different forms with different attributes, query keywords, and return values, into a uniform format used by the warehouse. Unlike in data cleaning, where data will have values changed, data transformation just reorganizes the data into a different format. Refresh is the operation used to update the data warehouse. Data warehouses contain time-series data from the operational databases, so every so often the data warehouse needs to update based on the new data in the operational databases. As part of a refresh, data cleaning and data transformation will be used.

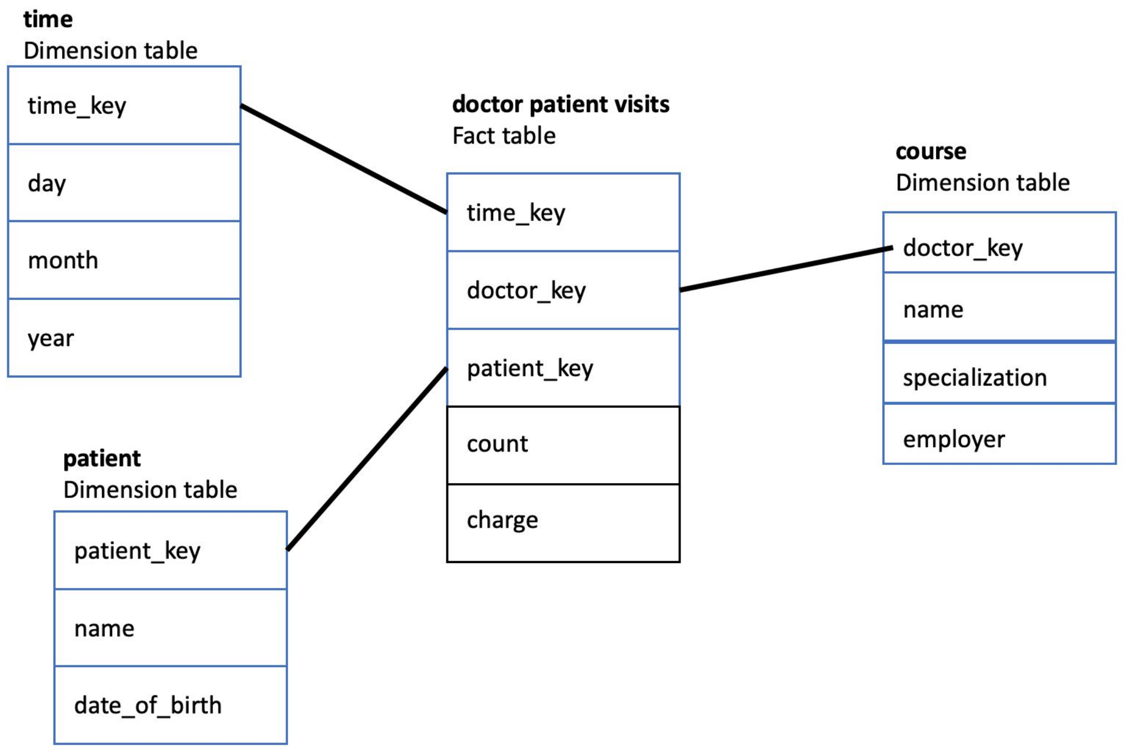
**4.3 Suppose that a data warehouse consists of the three dimensions time, doctor, and patient, and the two measures count and charge, where charge is the fee that a doctor charges a patient for a visit.**

**(a) Enumerate three classes of schemas that are popularly used for modeling data warehouses.**

The three popular classes of schemas used for modeling data warehouses are: the star schema, the snowflake schema, and the fact constellation schema.

**(b) Draw a schema diagram for the above data warehouse using one of the schema classes listed in (a).**

Star schema for the time, doctor, patient datawarehouse:

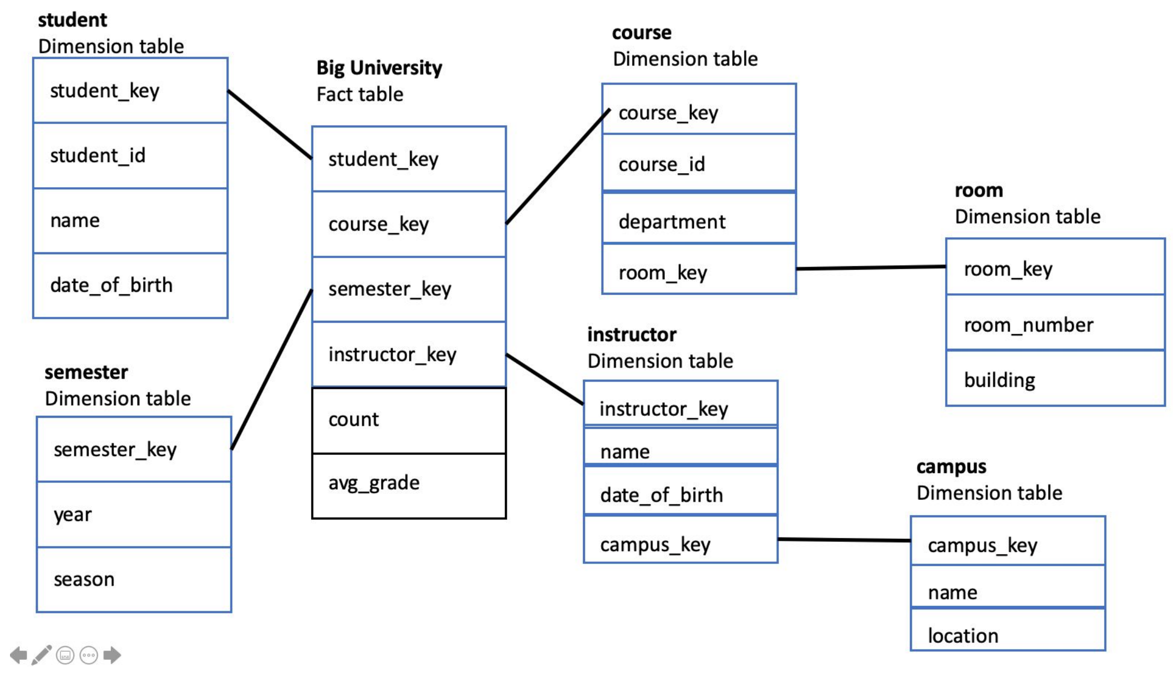


**(c) Starting with the base cuboid [day,doctor,patient], what specific OLAP operations should be performed in order to list the total fee collected by each doctor in 2010?**

In order to list the total fee collected by each doctor in 2010, we can perform a **roll-up** OLAP operation. Rolling up on the time dimension using a concept hierarchy we can condense the data down from the individual time units the data is in, such as days, weeks or months, and “roll” it “up” to the year level, look at 2010 and then we can see how much each doctor collected that year.

**4.4 Suppose that a data warehouse for Big University consists of the four dimensions student, course, semester, and instructor, and two measures count and avg grade. At the lowest conceptual level (e.g., for a given student, course, semester, and instructor combination), the avg grade measure stores the actual course grade of the student. At higher conceptual levels, avg grade stores the average grade for the given combination.**

**(a) Draw a snowflake schema diagram for the data warehouse.**



**(b) Starting with the base cuboid [student,course,semester,instructor], what specific OLAP operations (e.g., roll-up from semester to year) should you perform in order to list the average grade of CS courses for each Big University student.**

Starting with the base cuboid [student,course,semester,instructor], Roll up to [department,semester,instructor], and set department == "CS". Roll up the dates by year or a higher-level time unit if possible, such as "all time". Then list the "avg_grade"

**(c) If each dimension has five levels (including all), such as "student < major < status < university < all", how many cuboids will this cube contain (including the base and apex cuboids)?**

Total number of cuboids = $\prod_{i=1}^{n} (L_i + 1)$

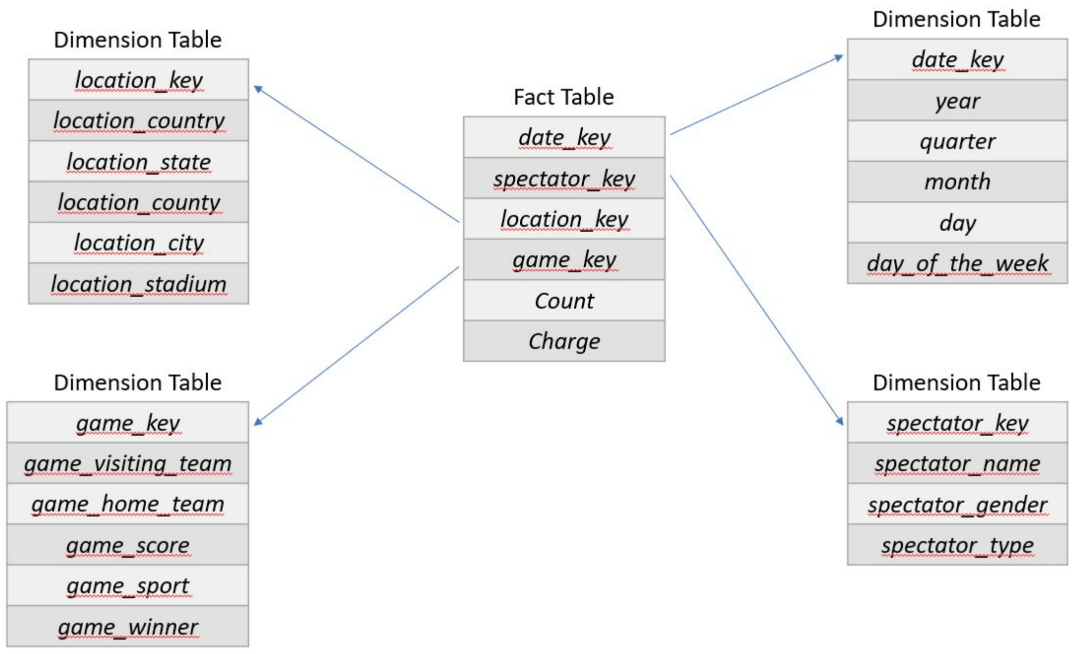
Where for each dimension i from 1 to n, $L_i$ represents the number of hierarchical levels (from the concept hierarchy) associated with each dimension. Taken together we have:

Total number of cuboids = $(L_1 + 1) * (L_2 + 1) * (L_3 + 1) * (L_4 + 1)$

Total number of cuboids = $(5+1) * (5+1) * (5+1) * (5+1) = 6^4 = 1296$

**4.5 Suppose that a data warehouse consists of the four dimensions date, spectator, location, and game, and the two measures count and charge, where charge is the fare that a spectator pays when watching a game on a given date. Spectators may be students, adults, or seniors, with each category having its own charge rate.**

**(a) Draw a star schema diagram for the data warehouse.**



**(b) Starting with the base cuboid [date,spectator,location, game], what specific OLAP operations should you perform in order to list the total charge paid by student spectators at GM Place in 2010?**

Something like:

Roll-up to [date, spectator, location]. Then, roll-up dates to the year level (2009, 2010, 2011, etc.), and dice the dataset for (location_stadium == GM Place) and (date == 2010) and (spectator_type == student). Finally, sum up all of the charges of this dataset, and you will have the total charge paid by student spectators at GM Place in 2010.

**(c) Bitmap indexing is useful in data warehousing. Taking this cube as an example, briefly discuss advantages and problems of using a bitmap index structure.**

Bitmap indexing essentially is a one-hot representation. For each attribute, bitmap indexing encoded an *n*-sized bit vector, where *n* is the cardinality of the attribute. If a given value *v* is present, that bit of the vector is 1, the rest of the bits 0. For example, the spectator_type attribute would look like:

| Tuple | adult | student | senior |
|---|---|---|---|
| T1 (student) | 0 | 1 | 0 |
| T2 (adult) | 1 | 0 | 0 |
| T3 (senior | 0 | 0 | 1 |

Bitmap indexing is very useful, because it can significantly reduce space and I/O since a string of characters is represented by a single bit (spectator_type: student vs. spectator_type: 0 1 0). However, if the cardinality is exceptionally large, this vector will also become very large. For example, if our attribute could be any of the values in the set (“aa”, “ab”, “ac”, … “ba”, “bb”,... “zz”), there are 676 (26*26) possible values, so a 676-bit vector would be needed. Instead, if you used strings, it’d require a 24-bit vector (chars are 8-bits, 2 letters + terminating character). However, the issues with high cardinality can be avoided with data compression using run-length encoding. This would transform some long vector like: 0000000000000000000000001000 to 24-0_1-1_3-0 (compressing numbers with RLE is confusing, underscores/hyphens wouldn’t be in the compressed representation).
The main advantage of bitmap indexing is it allows bitwise operations to be done for comparison, join, etc. operations, which significantly cuts down on compute time. Bitmap indexing may not be great if the data wasn’t properly cleaned/transformed. For example, the date’s year attribute could have “January”, “Jan”, “01”, “Jan.”, “january”, etc. for the month of January if the data wasn’t properly cleaned and transformed, which would increase the size of the bit vector.