

Advanced Data Warehousing

Data Warehouse Architecture

1. Introduction

Why do we build a data warehouse?

Because operational databases are not suitable for decision support queries. Therefore, operational databases are extracted, cleaned, and transformed into data warehouses.

Why do fact tables contain highly aggregated data?

Because decision support queries normally focus on aggregated data (i.e. **low level of granularity** or **low level of details**), fact tables often contain **highly aggregated data**, such as total lab usage (in the Uselog case study), or total fuel used (in the Robcor case study).

What is drill down in data warehouses?

When the management finds an interesting data in the fact table, they often like to drill down into more detail information. For example, in the Uselog case study, if the total lab usage in the evening in semester 1 is rather high, we might want to find out the reasons (such as the break down of the total lab usage by class and major, or even to a student level; who are the students who used the labs in the evening?).

How do we drill down in data warehouses?

This is an important question. Can we do it? Using the Uselog example, we can retrieve the total lab usage based on semester, time session, major, and class. For example, retrieve the total lab usage in semester 1, night session, major accounting, and master degree. Supposed this figure is interesting, and we would like to drill down further to find out the exact time distribution at night (perhaps because the information on night is not that precise – whether this is early evening, like 6-8pm, or very late at night, like after midnight).

Unfortunately, because the fact table does not store this information, the obvious answer is to drill down into the operational database. However, this is not a preferable practice, because the operational database might not be easily accessible (Note that a data warehouse might be built from several operational databases, and some operational databases are not even a database system, e.g. an excel spreadsheet). Therefore, accessing the operational database for a drill down purposes is not desirable.

So, how can we really do it?

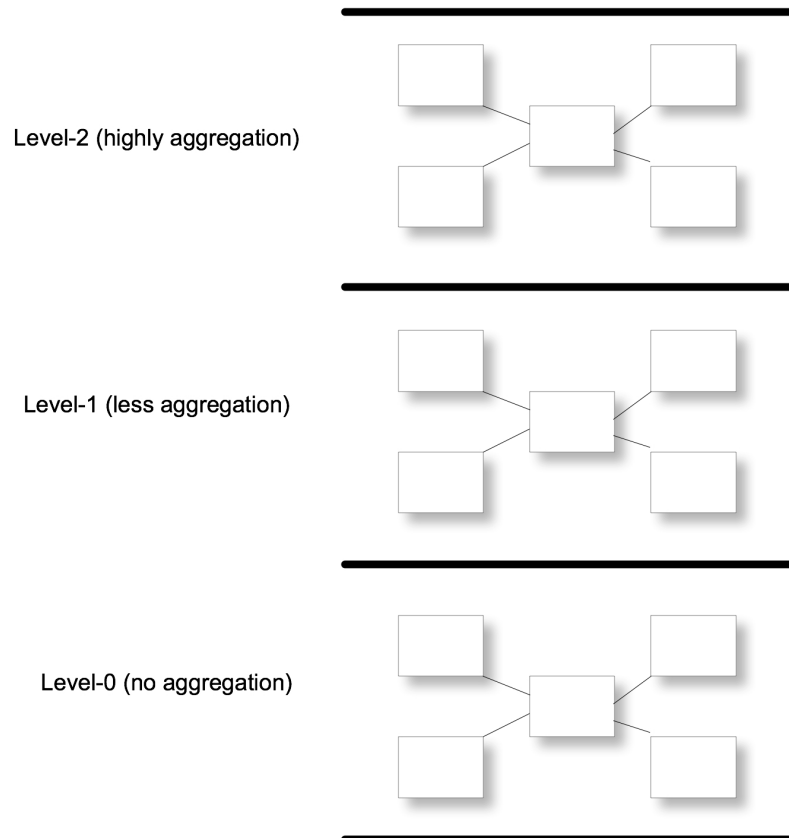
The answer is explained through the notion of “**Data Warehouse Architecture**”.

What is a data warehouse architecture?

A data warehouse architecture consists of a number of granularity layers of data warehouses. The highest granularity star schema, which contains the most detail data, is level-0. Level-1 builds on top of level-0 star schemas, and level-1 star

schemas have a lower granularity of the fact measures (e.g. less detail data, as the data is already aggregated). Subsequently, level-2 star schemas are built on top of level-1 star schemas, and have even a lower granularity of the fact measures.

Data Warehouse Architecture



So, level-0 has the highest level of granularity and lowest level of aggregation, level-1 has a lower level of granularity (and a higher level of aggregation) than level-0, level-2 has a lower level of granularity (and a higher level of aggregation) of level-1, etc. In short, highest level of granularity means lowest level of aggregation.

In summary:

Level 2 – high level of aggregation by (i) incorporating selected dimension tables only based on specifications, and (ii) incorporating user-defined grouping for time, distance etc (e.g. yearly/monthly only, short/med/long distance only).

Level 1 – medium level of aggregation by incorporating ALL domain tables from the operation database but may still incorporate some user-defined groupings.

Level 0 – lowest level of aggregation where ALL domain tables are incorporated as dimension tables AND no grouping in the dimension attributes. Basically, level-0 means no aggregation

In this lesson, we will use the term “**level of aggregation**” to refer to level-0, level-1, level-2, etc (instead of level of granularity).

2. The USELOG Case Study

What is the level of aggregation of star schema in the Uselog case study?

In the Uselog case study, the fact table is built using the following SQL:

```
create table fact_uselog as
select
  t.semId,
  t.timeId,
  t.class_id,
  t.major_code,
  count(t.student_id) as total_usage
from tempfact_uselog t
group by t.semId, t.timeId, t.class_id, t.major_code;
```

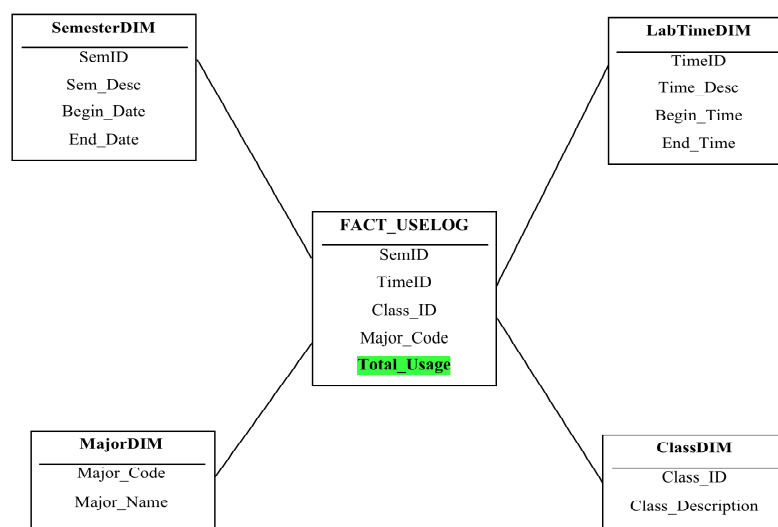
The fact measure in the fact_uselog table is number of students, using the count function. Therefore, the fact_uselog table cannot be in the lowest level of aggregation; it must be a higher level of aggregation (e.g. level-1 or level-2 or higher), because the fact measure is an aggregated information. This is the reason why we are not able to drill down the total_usage using this star schema. To be able to drill down to a lower level of aggregation, we need to query the lower level of aggregation of the star schema, which at the moment we don't have.

How would a lower level of aggregation of the Uselog star schema look like?

We need to have a lower level of aggregation of the Uselog star schema, so that we will be able to drill down the fact measure.

The fact_uselog table is highly aggregated, and therefore we need to build a lower level of aggregation of the Uselog star schema. At the moment, the fact_uselog star schema is level-2, which is as follows:

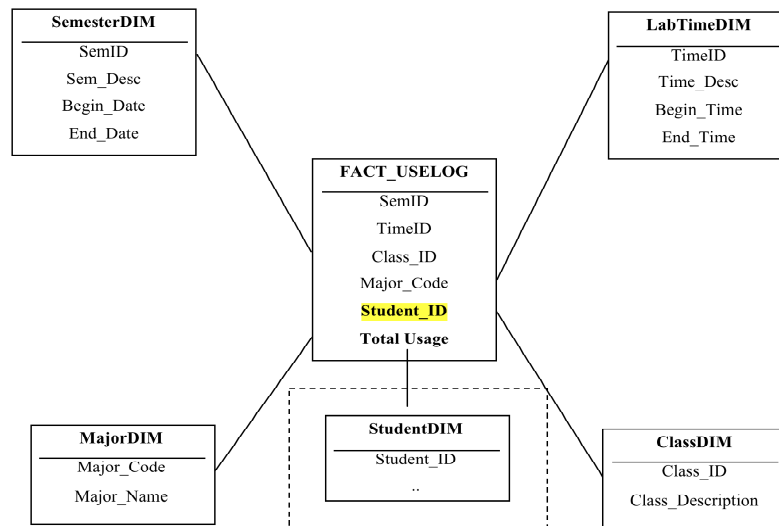
Level-2 Uselog Star Schema:



Note: In the assignment, we always start from a higher level of aggregation, such as level-2 aggregation in this example.

Level-1 Uselog Star Schema:

Level-1 Uselog star schema is basically the drill-down of the `total_usage` fact measure, which provides the details of who (which students) contribute to this `total_usage` fact measure. In other words, the fact table now contains the `Student_ID`.



This level-1 star schema is basically created by adding other information available from the tables in the operational database. In the Uselog example, we can add the student information and this will subsequently lower the level of aggregation of the Uselog fact table. Note that the lower the aggregation level, the more data you will have in the fact table.

```
create table tempfact_uselog as
select U.log_date, U.log_time, S.class_id, S.major_code, U.student_ID
from dw.uselog U, dw.student S
where U.student_id = S.student_id;
```

```
alter table tempfact_uselog
add (timeid number);
```

```
update tempfact_uselog
set timeid = 1
where to_char(log_time, 'HH24:MI') >= '06:01'
and to_char(log_time, 'HH24:MI') <='12:00';
```

```
update tempfact_uselog
set timeid = 2
where to_char(log_time, 'HH24:MI') >= '12:01'
and to_char(log_time, 'HH24:MI') <='18:00';
```

```
update tempfact_uselog
set timeid = 3
where to_char(log_time, 'HH24:MI') >= '18:01'
or to_char(log_time, 'HH24:MI') <='06:00';
```

```

alter table tempfact_uselog
add (semid varchar2(10));

update tempfact_uselog
set semid = 'S1'
where to_char(log_date, 'MMDD') >= '0101'
and to_char(log_date, 'MMDD') <= '0715';

update tempfact_uselog
set semid = 'S2'
where to_char(log_date, 'MMDD') >= '0716'
and to_char(log_date, 'MMDD') <= '1231';

```

The fact table is built using the following SQL:

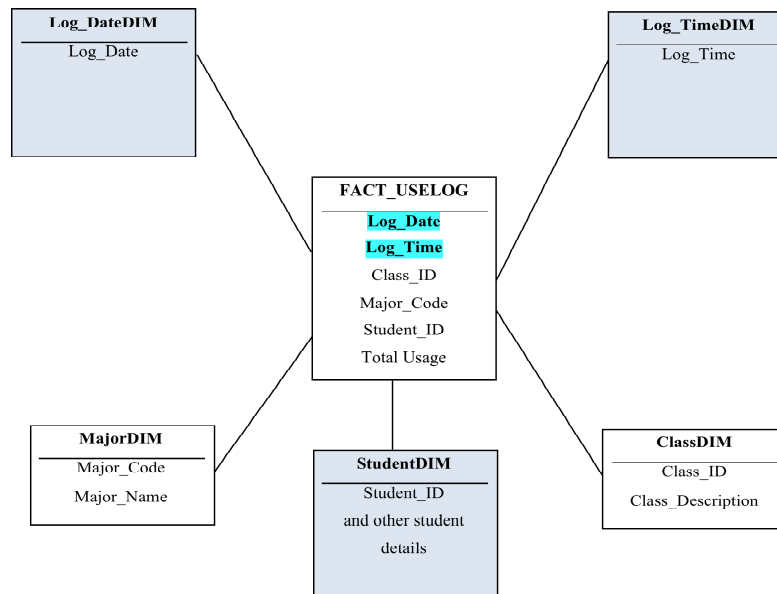
```

create table fact_uselog_level_1 as
select
    t.semide,
    t.timeid,
    t.class_id,
    t.major_code,
    t.student_id
    count(*) as total_usage
from tempfact_uselog t
group by t.semide, t.timeid, t.class_id, t.major_code,
t.student_id;

```

Level-0 Uselog Star Schema:

Now, the most detail star schema (the lowest level of aggregation) is the drill-down of the semester and labtime dimensions, which should contain the actual log_date (instead of semester), and log_time (instead of lab session time). This is the level where we do not have any grouping of dimension attributes.



```

create table fact_uselog_level_0 as
select
    U.log_date,
    U.log_time,
    S.class_id,
    S.major_code,
    U.student_ID,
    count(*) as Total_Usage
from dw.uselog U, dw.student S
where U.student_id = S.student_id
group by
    U.log_date,
    U.log_time,
    S.class_id,
    S.major_code,
    U.student_ID;
  
```

In summary, level-0 star schema provides the most detailed information about the data warehouse. The upper levels provide some levels of aggregated information, which has a higher level of aggregation. There is no particular guideline on how many levels should we have in the data warehouse architecture. It can be less or more than three, depending on what level of aggregation is needed. Additionally, there is no particular rule for upper levels star schemas. The only rule is that level (x+1) star schema is more aggregated than level x star schema. Moreover, there could be more than one star schema in

level ($x+1$). For example, level-0 has one star schema, but level-1 might have 5 star schemas, and level-2 have 10 star schemas.

The upper level star schema is often implemented as a “dashboard” in the data warehouse architecture, where users are able to interact with the dashboard, which provides users highly aggregated information for decision making. In the implementation, level-0 star schema (dimensions and fact) is implemented by the create table statement in SQL, whereas upper level star schemas are often implemented as a view (implemented by the create view statement in SQL).

Comparisons:

Now let's compare the differences between the three levels of star schemas. A sample data in the fact table of level-2 star schema is shown as follows:

Level-2 Fact Table

SemID	TimeID	Class_ID	Major_Code	Total_Usage
S1	3	Bachelor	IT	1500
S1	3	Bachelor	Bus	1250
S1	3	Bachelor	Law	788
S1	3	Bachelor	Science	980

Notes: S1 is Semester 1, and time ID 3 is evening

Suppose that the management retrieves this data from the fact table, and would be interested in knowing further details about total_usage of 1500 which is number of students in lab in Semester 1, evening, doing Bachelor of IT. In other words, the management would like to drill down further to find out more information about this 1500 of total_usage. We cannot do it in level-2, because the level-2 fact table has only data about semester ID, time ID, class ID, and major code. In order to drill down further, we need a lower level of aggregation of star schema (e.g. level-1 star schema).

So, a sample data for level-1 star schema is as follows. It basically breaks down the 1500 total_usage (semester 1, evening, Bachelor of IT) into a number of total_usage figures based on student ID. The total of total_usage of semester 1, evening, Bachelor of IT must be equal to 1500.

Level-1 Fact Table

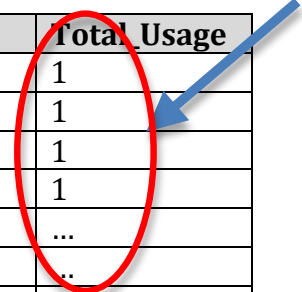
SemID	TimeID	Class_ID	Major_Code	StudentID	Total_Usage
S1	3	Bachelor	IT	21002	75
S1	3	Bachelor	IT	21023	10
S1	3	Bachelor	IT	21025	55
S1	3	Bachelor	IT	21166	120
S1	3	Bachelor	IT
S1	3	Bachelor	IT

The sum is 1500



Now supposed the management would like to drill down the total_usage of 75 (see the above highlight), in particular the management would like to see the actual date and actual time of login. Then we must go to level-0 star schema:

Level-0 Fact Table



LogDate	LogTime	Class_ID	Major_Code	StudentID	Total Usage
4-Apr	19:00	Bachelor	IT	21002	1
6-Apr	21:20	Bachelor	IT	21002	1
20-Apr	02:30	Bachelor	IT	21002	1
3-May	18:55	Bachelor	IT	21002	1
19-May	19:30	Bachelor	IT	21002	...
25-May	19:45	Bachelor	IT	21002	..

The sum is 75

It is natural that the fact measure in the level-0 star schema contains 1s only. Note that fact tables must have fact measures.

Actually, there is no rule in giving a number to the level. The rule of thumb is that level x has higher aggregation than level $(x-1)$. The most important is that level-0 is the lowest level of aggregation, which basically means no aggregation at all, and hence cannot be drilled down further.

Additionally, as mentioned earlier, on each level, there could be more than one star schema.

For example, we could have a new level in between level-0 and level-1 in the Uselog example above. The current level-1 Uselog has semesterID ('S1' and 'S2') and timeID (morning, afternoon, night), whereas level-0 has logdate (the actual date) and logtime (the actual time). We could have an intermediate level, where the semester is broken down into months, rather than directly into the actual date. Similarly, the timeID (morning, afternoon, night) can be broken down into hours, rather than directly into the actual time.

Hence, we could have two star schemas in the new level-1 (the old level-1 becomes level-2).

The following is the data in each level:

Level-3 Fact Table

SemID	TimeID	Class_ID	Major_Code	Total_Usage
S1	3	Bachelor	IT	1500
S1	3	Bachelor	Bus	1250
S1	3	Bachelor	Law	788
S1	3	Bachelor	Science	980

The sum is 1500

Level-2 Fact Table

SemID	TimeID	Class_ID	Major_Code	StudentID	Total_Usage
S1	3	Bachelor	IT	21002	75
S1	3	Bachelor	IT	21023	10
S1	3	Bachelor	IT	21025	55
S1	3	Bachelor	IT	21166	120
S1	3	Bachelor	IT
S1	3	Bachelor	IT

The sum will not be 75

Level-1a Fact Table

Month	TimeID	Class_ID	Major_Code	StudentID	Total_Usage
Jan	3	Bachelor	IT	21002	10
Feb	3	Bachelor	IT	21002	5
Mar	3	Bachelor	IT	21002	5
Apr	3	Bachelor	IT	21002	10
May	3	Bachelor	IT	21002	25
Jun	3	Bachelor	IT	21002	20
Jul	3	Bachelor	IT	21002	34

3=night

Level-1b Fact Table

SemID	TimePeriod	Class_ID	Major_Code	StudentID	Total_Usage
S1	6-7pm	Bachelor	IT	21002	25
S1	7-8pm	Bachelor	IT	21002	10
S1	8-9pm	Bachelor	IT	21002	5
S1	9-10pm	Bachelor	IT	21002	5
S1	10-11pm	Bachelor	IT	21002	3
...	...	Bachelor	IT	21002	...

The sum is 75

Note that if one month (e.g. July) is divided into two semesters (e.g. S1 and S2), when S1 is broken down into months (e.g. Jan-Jul), the total usage for S1 will not be equal to the sum of total usage from January to July, because some part of July will go to S1 and the other will go to S2.

Level-0 Fact Table

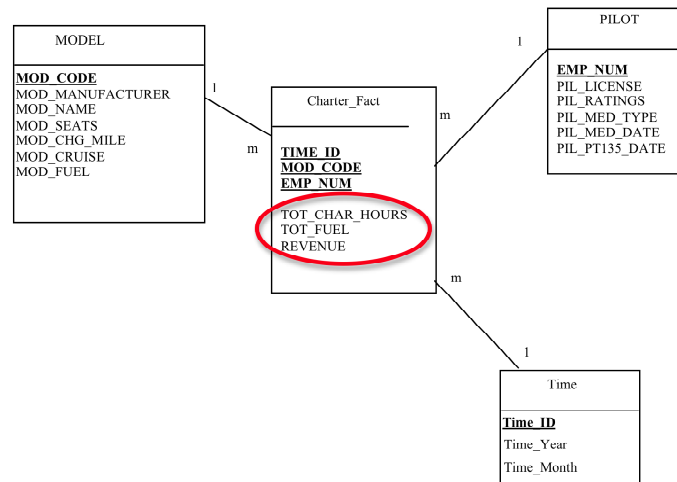
The sum is 75

LogDate	LogTime	Class_ID	Major_Code	StudentID	Total Usage
4-Apr	19:00	Bachelor	IT	21002	1
6-Apr	21:20	Bachelor	IT	21002	1
20-Apr	02:30	Bachelor	IT	21002	1
3-May	18:55	Bachelor	IT	21002	1
19-May	19:30	Bachelor	IT	21002	...
25-May	19:45	Bachelor	IT	21002	..

3. The ROBCOR Case Study

How does the data warehouse architecture look like for the Robcor case study?

Level-2 Robcor Star Schema:



Level-2 Robcor star schema's fact table is created by the following create table statement:

```

create table charter_fact_level_2 as
select C.Char_Pilot as EMP_Num,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM') as Time_ID,
       sum(C.Char_Hours_Flown) as Tot_Char_Hours,
       sum(C.Char_Fuel_Gallons) as Tot_Fuel,
       sum(C.Char_Distance * M.Mod_chg_mile) as Revenue
from   dw.Charter C, dw.Model M, dw.Aircraft A
where  C.AC_Number=A.AC_Number and A.Mod_Code=M.Mod_Code
group by C.Char_Pilot, M.Mod_Code, to_char(C.Char_Date, 'YYYYMM');
  
```

Note that the three fact measures: total hour flown, total fuel used, and total revenue, are all highly aggregated.

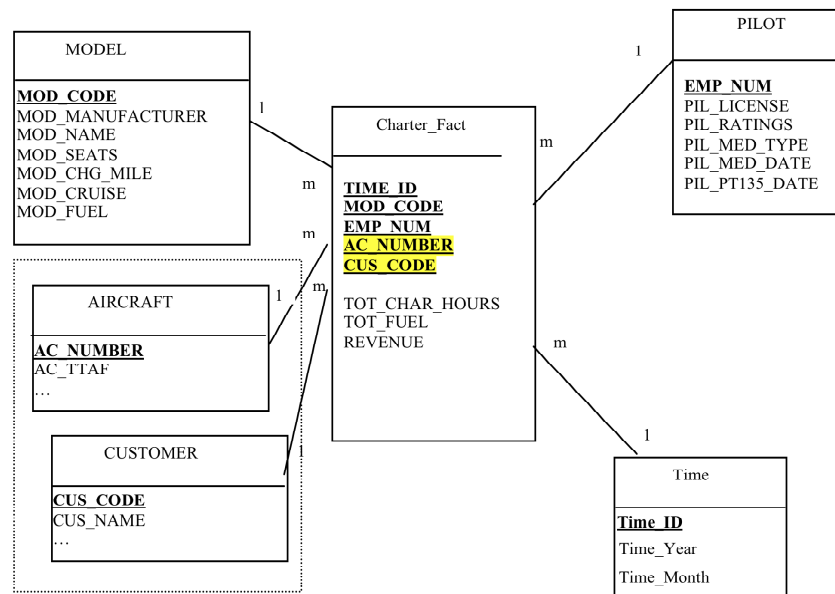
Level-1 Robcor Star Schema:

Level 1 star schema adds unique attributes from other tables in the operational database, thus reducing the level of granularity in the star schema. Level-1 Robcor star schema's fact table is created by the following create table statement:

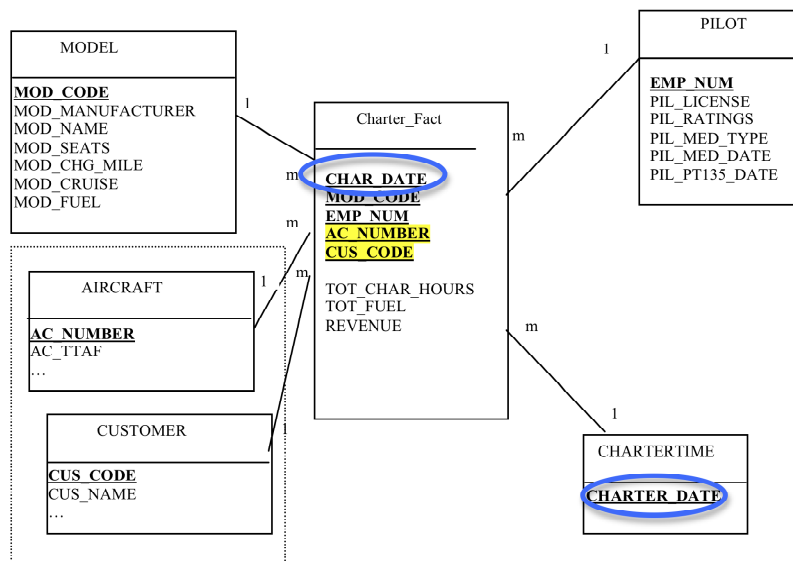
```

create table charter_fact_level_1 as
select C.Char_Pilot as EMP_Num,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM') as Time_ID,
       A.AC_Number,
       C.Cus_Code,
       sum(C.Char_Hours_Flown) as Tot_Char_Hours,
       sum(C.Char_Fuel_Gallons) as Tot_Fuel,
       sum(C.Char_Distance * M.Mod_chg_mile) as Revenue
from   dw.Charter C, dw.Model M, dw.Aircraft A
where  C.AC_Number=A.AC_Number and A.Mod_Code=M.Mod_Code
group by
       C.Char_Pilot,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM'),
       A.AC_Number,
       C.Cus_Code;

```



Level-0 Robcor Star Schema:



Level-0 star schema has the lowest level of aggregation, where the TIME_ID in level-1 is replaced by the actual charter travel date.

In level-1, the time dimension has a higher level of aggregation, namely month. Note that the granularity is month/year. This means that the records in the fact table are grouped based on month.

In level-0 schema, we need to drill down into each individual actual travel date. Therefore, we cannot use time dimension in level-1. We need to get the actual travel date as the dimension.

4. Summary

There are two methods to lower down the level of aggregation of a fact measure:

1. *Add a new dimension*
 - In the Uselog case study, from level-2 to level-1, we add a new dimension, called StudentDIM.
 - In the Robcor case study, from level-2 to level-1, we add two new dimensions: AircraftDIM, and CustomerDIM.
2. *Replace a dimension with a lower level of aggregation of the same dimension*
 - In the Uselog case study, from level-1 to level-0, we change the level of aggregation of SemesterDIM to LogDate, and LabTimeDIM to LogTime.
 - In the Robcor case study, from level-1 to level-0, we change the level of aggregation of the TimeDIM, from a month level to the actual date level.

THE END