DATABASES REPORT HOMEWORK – 3

By Noman Noor | Album number: 302343 Signed on 2020-06-02

DECLARATION:

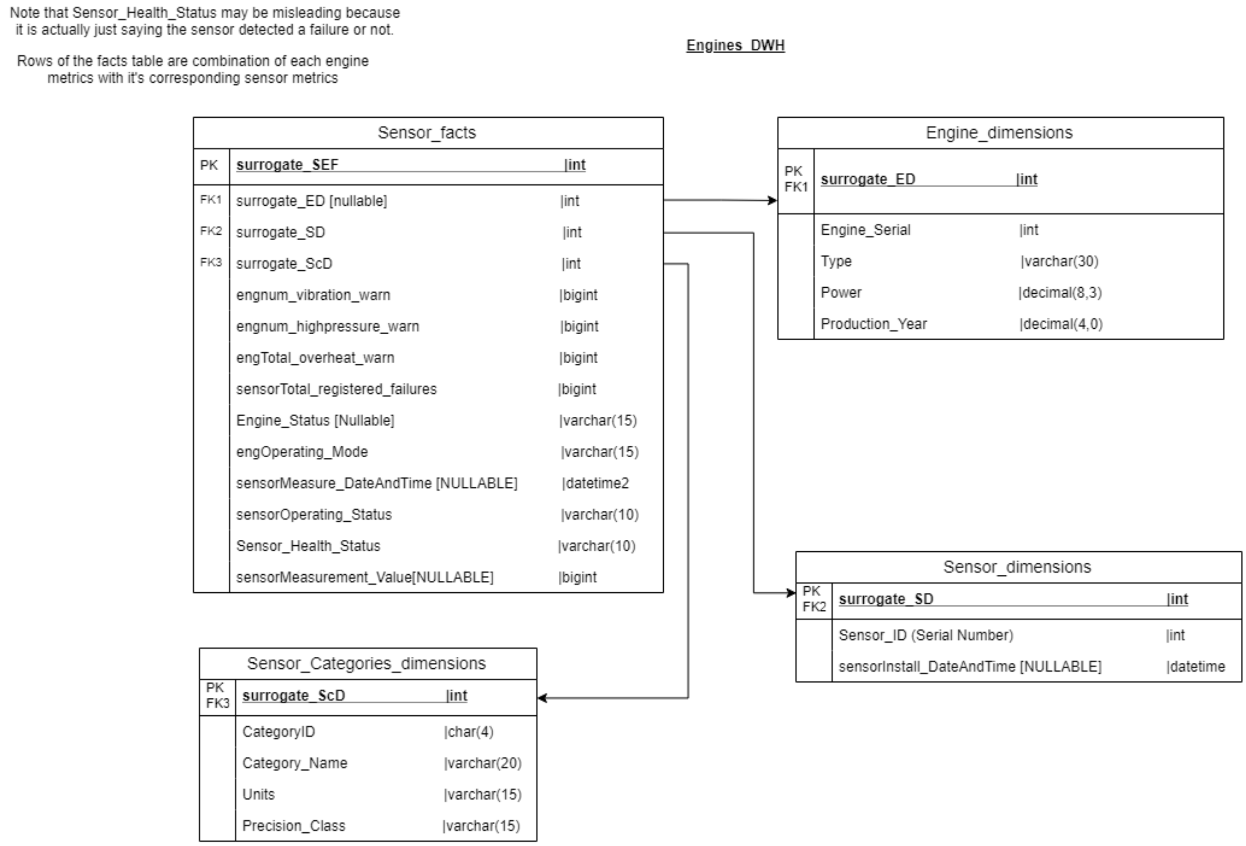
'I certify that this assignment is entirely my own work, performed independently and without any help from sources which are not allowed.'

(Signed by Noman Noor)

NOMAN NOOR

DESIGN OF THE DATA WAREHOUSE:

Database name: EnginesDWH



Note that Sensor_Health_Status may be misleading because it is actually just saying the sensor detected a failure or not.

Rows of the facts table are combination of each engine metrics with it's corresponding sensor metrics

Engines_DWH

Sensor_facts		
PK	**surrogate_SEF**	**int**
FK1	surrogate_ED [nullable]	int
FK2	surrogate_SD	int
FK3	surrogate_ScD	int
	engnum_vibration_warn	bigint
	engnum_highpressure_warn	bigint
	engTotal_overheat_warn	bigint
	sensorTotal_registered_failures	bigint
	Engine_Status [Nullable]	varchar(15)
	engOperating_Mode	varchar(15)
	sensorMeasure_DateAndTime [NULLABLE]	datetime2
	sensorOperating_Status	varchar(10)
	Sensor_Health_Status	varchar(10)
	sensorMeasurement_Value[NULLABLE]	bigint

Engine_dimensions		
PK FK1	**surrogate_ED**	**int**
	Engine_Serial	int
	Type	varchar(30)
	Power	decimal(8,3)
	Production_Year	decimal(4,0)

Sensor_dimensions		
PK FK2	**surrogate_SD**	**int**
	Sensor_ID (Serial Number)	int
	sensorInstall_DateAndTime [NULLABLE]	datetime

Sensor_Categories_dimensions		
PK FK3	**surrogate_ScD**	**int**
	CategoryID	char(4)
	Category_Name	varchar(20)
	Units	varchar(15)
	Precision_Class	varchar(15)

RATIONALE:

-- UPDATE: --

Basically, the facts table describes the metrics of the sensor along with corresponding metrics of the corresponding engine. The table represents the fact that the sensor measured _____ where the state of its engine was _____, also talking about the category of what the sensor measured. The dimension tables give more information about the Sensor, Engine, and the Category (We only keep information that doesn't change with time because it would be bad to store them inside the facts table as it would just be repeated data). For the fact table itself, our goal is to represent the fact the specific state(e.g. with a specific measurement) of each sensor while the engines were in some state. Same goes for sensors.

This fact table hence gives us the relationship of what each sensor measured for each engine, what a sensor measured for a specific category, vice-versa for both the previous statement, and so on.

One could say that one of the most important facts that our fact table conveys is that some sensor measures some value (or has some state) of some category for some state of some engine.

Note that the engine reference is nullable but it still conveys the useful fact that the server measured something even when it was not installed anywhere (probably some error or somebody forgot to reset the sensor from a business standpoint).

-- UPDATE ENDS --

I kept all metrics/measurements in the measurement_facts columns, and the things that would describe or explain them in the other databases “with the suffix _dimensions” because they were the dimensions that described the data in the facts table.

I also chose to use only surrogate keys because after research I found out that it a good practice to do so.

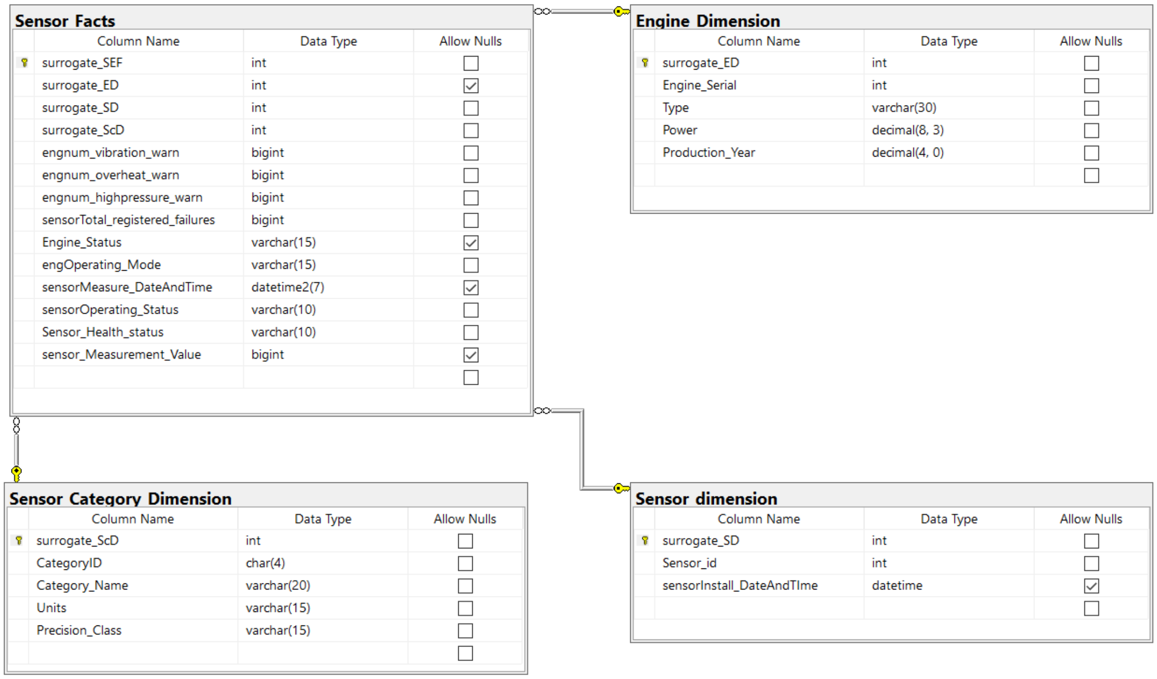
Quoting https://www.kimballgroup.com/1998/05/surrogate-keys/:

“

- Production may reuse keys that it has purged but that you are still maintaining, as I described.
- Production may make a mistake and reuse a key even when it isn’t supposed to. This happens frequently in the world of UPCs in the retail world, despite everyone’s best intentions.
- Production may re-compact its key space because it has a need to garbage-collect the production system. One of my customers was recently handed a data warehouse load tape with all the production customer keys reassigned!
- Production may legitimately overwrite some part of a product description or a customer description with new values but not change the product key or the customer key to a new value. You are left holding the bag and wondering what to do about the revised attribute values. This is the Slowly Changing Dimension crisis, which I will explain in a moment.
- Production may generalize its key format to handle some new situation in the transaction system. Now the production keys that used to be integers become alphanumeric. Or perhaps the 12-byte keys you are used to have become 20-byte keys.
- Your company has just made an acquisition, and you need to merge more than a million new customers into the master customer list. You will now need to extract from two production systems, but the newly acquired production system has nasty customer keys that don’t look remotely like the others.

”

RESULTING DESIGN:



Sensor Facts

	Column Name	Data Type	Allow Nulls
🔑	surrogate_SEF	int	☐
	surrogate_ED	int	☑
	surrogate_SD	int	☐
	surrogate_ScD	int	☐
	engnum_vibration_warn	bigint	☐
	engnum_overheat_warn	bigint	☐
	engnum_highpressure_warn	bigint	☐
	sensorTotal_registered_failures	bigint	☐
	Engine_Status	varchar(15)	☑
	engOperating_Mode	varchar(15)	☐
	sensorMeasure_DateAndTime	datetime2(7)	☑
	sensorOperating_Status	varchar(10)	☐
	Sensor_Health_status	varchar(10)	☐
	sensor_Measurement_Value	bigint	☑
			☐

Engine Dimension

	Column Name	Data Type	Allow Nulls
🔑	surrogate_ED	int	☐
	Engine_Serial	int	☐
	Type	varchar(30)	☐
	Power	decimal(8, 3)	☐
	Production_Year	decimal(4, 0)	☐
			☐

Sensor Category Dimension

	Column Name	Data Type	Allow Nulls
🔑	surrogate_ScD	int	☐
	CategoryID	char(4)	☐
	Category_Name	varchar(20)	☐
	Units	varchar(15)	☐
	Precision_Class	varchar(15)	☐
			☐

Sensor dimension

	Column Name	Data Type	Allow Nulls
🔑	surrogate_SD	int	☐
	Sensor_id	int	☐
	sensorInstall_DateAndTIme	datetime	☑
			☐

APPENDIX:

All the SQL source code for generating this data warehouse database is provided in the attachment to the zip file containing this.