

Homework 4

Björn Þór Jónsson

Introduction to Database Design, Fall 2019

Do as many of the following exercises as you have time for. Focus on the exercises you feel will benefit your preparation the most.

1 Hardware and DBMS Design

Question 1.A Select the correct statements below:

- (a) SSDs improve performance of unclustered indexes.
- (b) Before a transaction can be committed, all the disk pages it has updated must be written to disk.
- (c) In a main memory system, uncommitted changes may be written to disk.
- (d) Google claims to have built a distributed system that offers both consistency and availability. The CAP theorem says this is possible.

Question 1.B Reflect upon why ACID transactions are rarely used in distributed systems.

2 Data Systems for Analytics

Question 2.A Select the correct statements below:

- (a) Key-value stores can support all needs of analytics applications.
- (b) In big data, “velocity” means that it is necessary to react quickly to the large amounts of data being added to the system.
- (c) Multimedia is inherently very unstructured data.
- (d) ACID transactions are a fundamental method to support analytics applications.

Question 2.B Consider a scenario where you have 1PB of raw data files that have just been produced as a result of a scientific experiment. You are trying to make scientific discoveries using this data, using complex computations, and also determine the experiments you would like to perform in the future. You have a cluster of machines in your lab (100ish nodes), which have 128GB main-memory and 16 cores each. Explain (with convincing arguments) what type of data management/processing system you would you pick for this scenario.

3 Normalisation

Consider the SQL script, and the associated description, in Section 3 of Homework 3. In Homework 3, you normalised two relations of the five in the database. Now, you are asked to normalise one more relation. For the Projects relation, take the following steps:

1. Find all the FDs in the relations, given the constraints and assumptions above.

Note: We strongly recommend to make a script in your favourite programming language to generate the SQL script, based on a list of column names.

2. Decompose the relation until each sub-relation is in BCNF, or in 3NF but not BCNF, while preserving all non-redundant FDs. Write down the results schema description in a simple Relation(columns) format.
3. Write the detailed SQL commands to create the resulting tables (with primary keys and foreign keys) and populate them, by extracting the relevant data from the original relations.

Note: In this homework, create a new relation also for the “original” relation, so that you can compare the decomposition result with the original relation. Thus, you do not need to alter any table at any point.

4. Select the correct normal form for the decomposed schema.

4 Godly strife (by Johan von Tangen Sivertsen)

Zeus is tired of all the infighting among the gods. He has brought you to mount Olympus to help deal with the problem. To avoid conflicts Zeus wants a simple database that tracks all the promises that the gods have made to humans. That way a god can quickly check for conflicts before making a new promise. In case conflicts arise anyway the database should track the various sacrifices made by humans to gods. The time, place and what was sacrificed should all be recorded. Sacrifices are divisible into three categories, flesh(cooked,livestock), wine and valuables(gold,gemstones,etc.). Each category has different attributes that describe them but they all have value. It should be possible to sum up

the total value of sacrifices on a pr. human pr. god level to help resolve any conflicting promises.

Further, some humans are appointed priests, a priest is always promised protection by his associated deity and if a sacrifice is conducted in a ceremony presided over by a priest, it is twice as valuable as normally. Any priest serves a single deity. Finally Poseidon has requested that the database also tracks if a human attacks any sea-monster or cyclops and in that case a hecatomb should be deducted from their total sacrifice value for all gods.

1. Draw an ER-diagram that supports the requirements.
2. Write the DDL for a database according to your design.

Remember that Zeus will likely strike you down with lightning if he is not pleased with your work.

5 SQL

Consider the *Northwind* database that comes with the homework. Write SQL queries to answer the following information requests.

Note that most of these queries are intentionally relatively easy to bring you back to SQL (only 3.3 is really difficult). Revisit old homeworks and exams for some challenging queries, including division queries.

5.1 Clients

Client information is found in the table *customers*.

Q1.1 How many customers are there in the database?

Q1.2 How many distinct customer *company* names are in the database?

Q1.3 If you have answered the previous two queries correctly, you should realize that some two customers have the same *company* name. Write a query that will return the common *company* name and their IDs. Please make sure that each pair only appears once; the query should only return one record!

5.2 Orders

For each order, there is one record (order head) in *orders*. This record shows, among other information, the customer and order ID. The orders for individual products are found in *order_details*, which shows the order ID, product ID, and price information (line price = unit price * quantity * (1 - discount)).

Q2.1 What is the total price for all orders?

Q2.2 All the orders are from 2006. What is the total price for all orders of each month (ordered by the month)?

Hint: You can use the MONTH() function to extract the month from a date.

Q2.3 The queries above would have been easier to write with an appropriate view on order_details, where the line price is already calculated. Create such a view. Use this view when appropriate in the following.

Q2.4 Which order has the most expensive order detail, and what is the value of that order detail?

Q2.5 Which order is the most expensive, and what is the value of the order?

5.3 Product Sales

Information about products is found in the table Products; the table Categories contains categorical information about the products.

Q3.1 What are the total sales of each product? Order the results such that the highest sellers are shown first. Then modify the query to include products that have not been sold at all.

Q3.2 Redo Query Q3.1, but this time only for products in categories that contain the word ‘Pasta’ in their description.

Q3.3 Write a query that yields an ordered list of the 10 products that are most frequently ordered (by number of records in *order_details*). The list should contain a rank (1-10, with ties) and should look like the list in Figure 1.

Note: PostgreSQL has a RANK() function, that might actually work correctly. However, try to do this query without it, as that is not a standard function. As an example, the MySQL RANK() function will yield a different result.

Hint: It is helpful to create a new view, which uses the view from Q2.3 to count the occurrences of each product in *order_details*. The query is then a (complex) self-join of this new view.



Result Grid				Filter Rows:	<input type="text" value="Search"/>	Export
	rnk	id	product_name	cnt		
▶	1	43	Northwind Traders Coffee	5		
	1	48	Northwind Traders Chocolate	5		
	3	8	Northwind Traders Curry Sauce	4		
	3	19	Northwind Traders Chocolate Biscuits Mix	4		
	3	41	Northwind Traders Clam Chowder	4		
	3	80	Northwind Traders Dried Plums	4		
	3	81	Northwind Traders Green Tea	4		
	8	34	Northwind Traders Beer	3		
	8	40	Northwind Traders Crab Meat	3		
	10	4	Northwind Traders Cajun Seasoning	2		
	10	6	Northwind Traders Boysenberry Spread	2		
	10	7	Northwind Traders Dried Pears	2		
	10	51	Northwind Traders Dried Apples	2		
	10	56	Northwind Traders Gnocchi	2		
	10	1	Northwind Traders Chai	2		
	10	72	Northwind Traders Mozzarella	2		

Figure 1: The ranked list for Query 3.3.