**Written exam, DBD, January 2006**

**Open book exam: Any books or notes are allowed.**

**The assignment contains 5 questions. Total time available: 4 hours.**

To ease photo copying: Use a soft pencil on blank or ruled paper. **Not** on squared paper.

There is a time estimate for each question. 30 minutes have been set off for reading the text and as spare. The time estimates are **not** weights. The final marks reflect an overall assessment.

Note: This English version has small changes compared to the original Danish version.

**Hotline support system**

Many companies have a hotline that help users with IT issues. Several applications on the market promise to support the hotline, but they rarely do it well.

Your job is to design a good support system for hotlines. The system will not be big and complex with a lot of data classes and virtual windows, but many details must be in place to provide effective support. This shows up for instance in the function details.

**Users**

Users encounter problems of many kinds. For instance they may have forgotten their password, so they cannot start their work, or the printer lacks toner, or they cannot remember how to make Word write in two columns. The problem may also be to repair something, for instance a printer, or to order a program the user needs.

The easiest solution is to call hotline. In many cases this solves the problem right away. However, hotline prefers to receive the problem request by email to *hotline@...* Sometimes this is impossible, for instance if the problem is that the user has forgotten his password.

If you cannot have your problem solved right away, it is annoying not knowing when it will be solved. How often will you for instance have to go to the printer to check whether it has got toner now? In many cases the problem has been solved, but the user doesn't know.

Some support systems allow the user to look up his problem request in the support system to see what has happened, but it is inconvenient and how often should he look? On the other hand, there are cases where the user wants to deliver further information, or has solved the problem in some other way. Here the user might open the problem request and record what is necessary.

Assume that there are at most 1000 users in the company. There is an employee database with the user's name, phone number, email address, user name and password. The support system can retrieve data from it.

**Hotline**

Hotline is staffed by *supporters*. Some supporters are *first line,* others are *second line*. First-line supporters receive the requests by phone or email, or in person when the user turns up at the hotline desk.

In busy periods, a first-line supporter may receive around 50 requests a day. Around 80% of the requests can be dealt with right away, and for these problems it is particularly hard to ensure that supporters record them for statistical use.

The remaining 20% of the requests are passed on to the second line. Based on the problem description and talks with the user, first line can often give the request a priority and maybe an estimated time for the solution. (Experience shows that users shouldn't be allowed to define the priority themselves, because they tend to give everything a high priority.)

Half of the second-line requests are in principle easy, but cannot be dealt with immediately. The supporter may have to move out of the office, for instance to change toner in the printer or help the user at his own PC. Usually this ends the request, but it may also turn into a long request.

Around 10% of all problems are long requests because the problem has to be transferred to a hotline person with special expertise, or because spare parts and expertise have to be ordered from external sources. Transferring the problem often fails. In some hotlines they place yellow stickers on the expert's desk, but the stickers often disappear. Or the expert misunderstands the problem. For this reason it is important that the expert in person or by phone can talk with the supporter who initially received the request, or with the user himself.

There are at most 10-15 employees that occasionally or all the time serve as supporters. They know each other and know who is expert in what. The supporters frequently change between first and second line, for instance to get variation.

The request is sometimes lost because a supporter has started working on it, but becomes ill or goes on vacation before it is finished.

Most hotlines dream of making statistics of frequent and time-consuming requests in order to find ways to prevent the problems. However, it is hard to record the correct data. Gathering data also makes it possible to measure how long hotline takes to handle each request. This also encourages hotline to be careful to tell the system when requests have been completed.

In busy periods, around 100 requests may be open (unresolved). Then it is hard for the individual supporter to survey the problems he is working on and see which problems are most urgent.

**Existing support systems**

There are several support systems on the market. Typically they automatically collect all emails sent to *hotline@ ...* and put them in a list of open requests. Figure 1 shows an example of such a list. You can see the request number (Req), the email address of the sender, the supporter working on it (Owner), and how long ago the request was received (Age).

You can also see when someone last looked at the request (ActOn). However, this is not really useful. It would be better to see when the request should be closed according to what the user expects. It would be nice if the system warned about requests that are not completed on time.

As you can tell from the figure, the system cannot handle national letters (Danish), and it is not intuitive what the various functions do.

Anyway, the basic principle is okay. A supporter keeps the list on the screen so he can follow what is going on. He can open an incoming request (much the same way as you open an email), maybe take on the request (for instance by sending a reply mail), classify the case according to the cause of the problem (printer, login, etc.), give it a priority, transfer it to someone else, etc. When the request has been completed, the supporter closes it, and the request will no longer be on the usual list of open requests.

As you can see on the figure, *Status* isn't used at all. It is too cumbersome and it is obscure how to use it. Here is a reasonable proposal for how to use the status field:

First line A first-line supporter must take on the request, for instance because it just arrived.

Second line A second-line supporter must take on the request.

Taken The request is handled by a supporter (the owner). The owner may change from one line to another while he is handling the request.

Parked The request awaits something, for instance an external delivery, and hotline need not do anything meanwhile.

Reminder The request hasn't been closed in due time, or the external delivery wasn't received in due time.

Closed The request has been handled. However, it may be opened again, for instance because the user doesn't think the problem has been solved.

Open requests are those that are neither parked, nor closed.

When a user calls by phone or in person, the supporter creates a new request. It will appear in the normal list of requests. For statistical and other purposes, it is useful to keep track of when requests change state.

While a long request is handled, it may receive additional information from the original user as well as from supporters. The user and the supporters should at any time be able to see the request history, including state changes.

A supporter can set the system to send an ordinary email to himself when he has to look at some request. This is particularly useful for second-line supporters who concentrate on other tasks until they are needed for support.

In most support systems it is cumbersome to classify requests according to the problem cause. One reason is that the true cause may not be known at the beginning. Some hotlines work with causes in several levels, for instance:

printer -> color printer -> toner

**Question 1. Data model (estimated time: 30 minutes)**

Make an E/R model for the system. Show the attributes for each class. Omit artificial keys and foreign keys.

**Question 2. Task descriptions (estimated time: 60 minutes)**

Describe the tasks for users as well as supporters as a task list (annotated). Add sub-tasks and problems where relevant. Because of question 4, it is a good idea to make space for notes about the solution. (In this and the rest of the questions, you may ignore tasks that deal with request statistics, creating/deleting employees, and editing the catalog of request causes.)

**Question 3. Virtual windows (estimated time: 60 minutes)**

Design virtual windows in graphical form for the supporters in hotline. Assume that the system will run on a Windows-like platform, i.e. without html-look, links, etc.

**Question 4. Functions and problem resolution (estimated time: 40 minutes)**

For each virtual window, show the semantic functions and search functions that are needed. Briefly describe how the functions and windows solve the problems that exist today in a hotline. Where needed, use a short mini-spec to explain the details of a function.

**Question 5. Usability test (estimated time: 20 minutes)**

During development, usability tests must be carried out. Specify two test tasks that would be suited as the first tasks during the test.



Figure 1. A list of hotline requests in an existing support system.

Søren Lauesen 2006-10-01

DBDexam05A.doc