

A sample project to the subject Database and
information systems

Information system of auctions

(c) 2012-2019, version: 0.15, 20190325

Radim Bača, Peter Chovanec, Michal Krátký

Department of Computer Science, FEECS, VŠB – Technical University
of Ostrava

<http://dbedu.cs.vsb.cz/>

Contents

1	Assignment Specification	3
2	Conceptual Model	4
3	Data Models	5
4	State Analysis	8
5	Functional Analysis	9
5.1	List of Functions	9
5.2	Detail Description of Functions	11
6	Design of User Interface	16
6.1	Menu	16
6.2	Auction Detail Form	17
7	Often Issues in Analysis	18

General rules

- The goal of the analysis is to describe a future system, the goal **is not to describe as many pages as possible** (a good analysis can include 20 pages, a bad analysis 100 pages).
- A developer must be able to implement the system according to the analysis – **the analysis must be complete**.
- A analysis has not to include a description of trivial aspects of the system, e.g. trivial operations (**C**reate**R**ead**U**pdate**D**ele**t**e) for codebooks, we must describe only complicated parts of the system in more detail, e.g. complicated functions.
- The same problem as an *incomplete analysis* is an analysis describing everything in the system – we call it *analysis-paralysis*. Why? We are not able to find complex (and important) features of the system.
- The functional analysis is anyway important as the data analysis. We are not able to implement the system without the functional analysis.
- Why? The number of functions **is not equal** to the number of tables $\times 4$ (since we have 4 database operations) or 5 (since we often use one select retrieving a record for primary key and one select retrieving all records of a table); a system can include a number of other functions.
- Complicated functions are described by a tool like the minispecification, sequence UML diagram or data flow diagram (DFD). We select the best one for a concrete situation; however, it seems that doing the minispecification takes the lowest time.
- We define SQL commands of the functions in the analysis, as a result the analysis is more concrete and the implementation of such a function is possible without any doubt. Moreover, it helps us to tune data and physical design of the system.

- **Summary:** Two ways of poor analysis:
 - An analysis describing nothing.
 - An analysis describing everything.
- **A good idea how to start a functional analysis** is to design forms of the user interface, and, in such a way, to identify functions of the system.
- **A good idea how to design a function** is to minimize a number of database operations (e.g., to use one complex select instead of a number of selects).

1 Assignment Specification

WHY? We need an information system for the management of the electronic auctions.

WHO? The system can be used only by the users created by the administrator. The system recognizes two categories of the users. The first category represents the users that can only create auctions (further only *Auctioneer*) and the second category represents the users that can only bid at the auctions (further only *User*). *Administrator* has the special privileges for the management of the users, categories and for the delete of an auction.

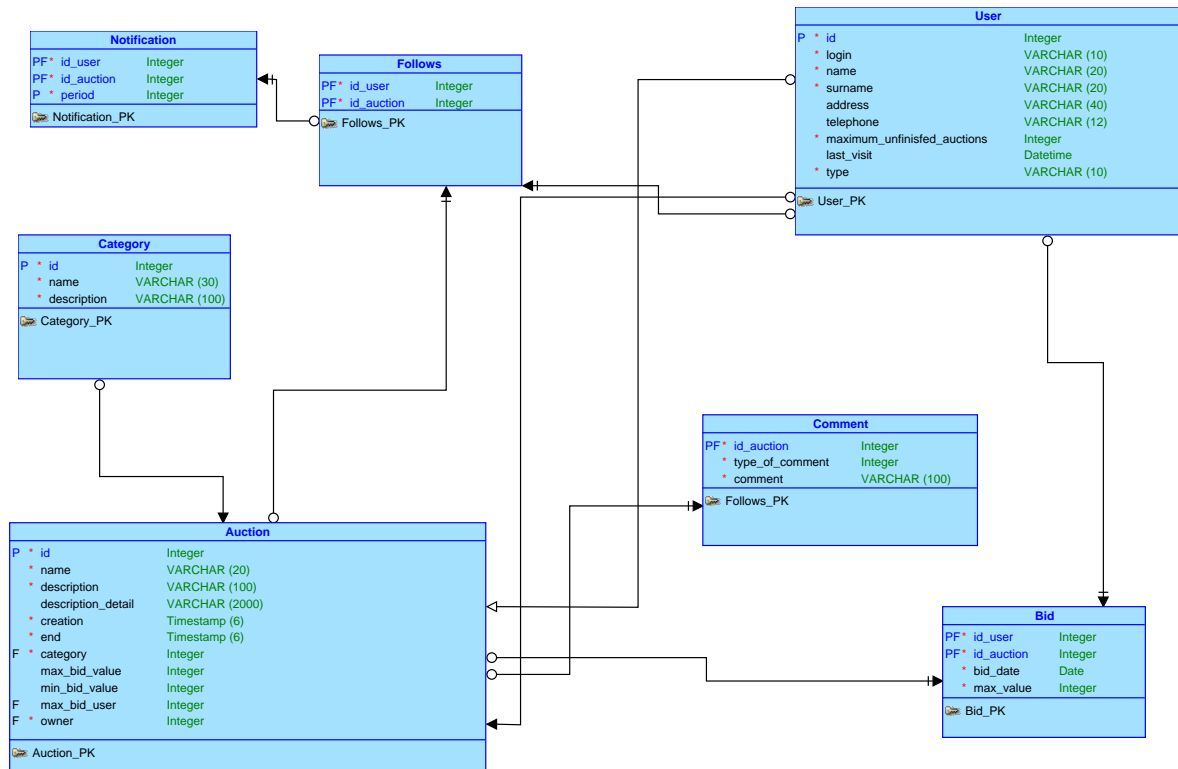
INPUTS: Detail information about the auctions and their bids. Moreover, the users can set an observation on the auction and also comment the auctions after they won.

OUTPUTS: Information about the status of the auctions, comments about the auctioneers and the list of the observed auctions.

FUNCTIONS: The system has to provide the bids at the auctions, where it is necessary to compare current bid with the maximal values of the other bids, update the information about the maximal current bid and automatic bidding for an auction. Moreover, the system has to check the observed auctions in the regular intervals and if the observed auction comes to its end, the system has to notify the observers of the auction. The system has to remember the date of the last user visit and to show him the new auctions after his log in.

2 Conceptual Model

ER Diagram



Entity Types

Legend: **Table**, Primary Key, *foreign key*, attribute

User(id, login, name, surname, address, telephone, type, maximum_unfinished_auctions, last_visit)

Auction(id, name, description, description_detail, creation, end, *owner*, *category*, *id_user_max_bid*, max_bid_value, min_bid_value)

Bid(*id_user*, *id_auction*, bid_date, max_value)

Comment(*id_user*, *id_auction*, type_of_comment, comment)

Follows(*id_user*, *id_auction*)

Category(id, name, description)

Notification(*id_user*, *id_auction*, period)

3 Data Models

Description of tables is depicted in the following tables.

Table **User**

	Data type	Length	Key	Null	Index	IC	Description
id	Int		Primary	N	A		
login	Varchar	10		N			Login of a user
name	Varchar	20		N			User name
surname	Varchar	20		N			Last name of the user
address	Varchar	40		A			Street and city of the user
telephone	Number	12		A			
type	Varchar	10		N		1	User type
maximum_unfinished_auctions	Integer			N		2	The maximum auctions owned by the user at once
last_visit	Timestamp			A			Timerstamp of the last visit of the system

Table **Auction**

	Data type	Length	Key	Null	Index	IC	Description
id	Int		Primary	N	A		
name	Varchar	20		N			Auction name
description	Varchar	100		N			A short description
description_detail	Varchar	2000		A			A long description
creation	Timestamp			N		3, 4	Date of creation
end	Timestamp			N		3, 4	Auction finish date
owner	Int		FK (User)	N			Bidder
category	Int		FK (Category)	N			Category of the auction
max_bid_user	Int		FK (User)	N			Who has the highest bid for the auction
max_bid_value	Int			N			The value of the highest bid of the auction
min_bid_value	Int			N			The minimal value of bid for the auction

Table **Bid**

	Data type	Length	Key	Null	Index	IC	Description
id_user	Int		PK, FK (User)	N	A		
id_auction	Int		PK, FK (Auction)	N	A		
bid_date	Timestamp					5	Bid date
max_value	Int						Maximal value of the bid

Table **Notification**

	Data type	Length	Key	Null	Index	IC	Description
id_user	Int		PK, FK (User)	N	A		
id_auction	Int		PK, FK (Auction)	N	A		
period	Int	Primary	N	A			The number of hours before the end of the auction, when the user has been notified

Table **Comment**

	Data type	Length	Key	Null	Index	IC	Description
id_auction	Int		PK, FK (Auction)	N	A		
type_of_comment	Int						
comment	Varchar	100					Content of the comment

Table **Follows**

	Data type	Length	Key	Null	Index	IC	Description
id_user	Int		PK, FK (User)	N	A		
id_auction	Int		PK, FK (Auction)	N	A		

Table **Category**

	Data type	Length	Key	Null	Index	IC	Description
id	Int		Primary	N	A		
name	Varchar	30		N			
description	Varchar	100		N			The description of the auctions in the specified category

Integrity Constraints:

1. Type: admin, auctioneer or user.
2. `maximum_unfinished_auctions < 100`.
3. `creation < end`.
4. `end - creation < 21 days`.
5. `bid_date < #auction.end`, where `#auction` is the auction defined by `id_auction` (i.e. the bid auction).

4 State Analysis

We define these status of auctions:

- **New** – new inserted auction.
- **Running** – a new auction where the start of the auction `Auction.creation` \leq the current date (and time) and the end of the auction `Auction.end` \geq the current date.
- **Finished** – a running auction where `Auction.end` \leq the current date. Two types of an auction **Winning auction without comment** and **Winning auction** are related to this state.

We define these types of auctions from the User and Auctioneer point of view:

- **Own** – an auction created by the user (attribute `Auction.owner`).
- **Followed** – an auction is followed by a user if a record including the auction and user exists in the table `Follows`.
- **With a bid** – an auction is 'with a bid' if a record including the user and auction is in the table `Bid`.
- **Winning auction without comment** – a finished auction where `max_bid_user` is the id of the user and there is no record with `id_auction` in the table `Comment`.
- **Winning auction** – the same, but the record with `id_auction` exists in the table `Comment`.

5 Functional Analysis

5.1 List of Functions

1. User Management

Table: User, Responsibility: Admin

- **1.1 User Insert**
- **1.2 User Update**
- **1.3 User Delete**
- **1.4 List of Users** with a definition of a filter to search users
- **1.5 User Detail**
Responsibility: Admin; User and Auctioneer see only the own record

2. Auction Management

Table: Auction, Category, User

- **2.1 New Auction**
Responsibility: Auctioneer
- **2.2 Auction Update** – it is possible to update only some auctions
Responsibility: Auctioneer
- **2.3 Auction Delete**
Responsibility: Admin – it is possible to delete only some auctions; however, it is possible to use CASCADE to delete all auctions and related records.
Notice: In a real information system, delete is not often used, records are flagged as deleted or they are moved to archive tables.
- **2.4 Auction Details**
Table: all, Responsibility: Admin, User, Auctioneer

3. List of Auctions

Tables: all

- **3.1 Continuing auctions** – a list of continuing auctions for selected categories.

Notice: In a real system, users requires to search auctions according to other attributes, e.g. description.

- **3.2 Observed auctions**
- **3.3 Auctions with bid**
- **3.4 List of winning auctions without comment**
- **3.5 List of winning auctions**
- **3.6 Auctions of auctioneer**

SQL commands returning these lists are defined in Section 5.2.

4. Bidding

Table: Bid, Responsibility: User

- **4.1 New bid** – new bid to an auction is executed for a user only if `max_value > min_bid_value` of the auction.
- **4.2 Bid update** – `max_value` is increased, `bid_date` is automatically updated.
- **4.3 Automatic bid** – this function is executed by functions 4.1 and 4.2. The function is described in Section 5.2.
- **4.4 Bid cancellation** – although it is a common function, we ignore it in the system. Only owner of an auction can cancel the auction (the attribute `owner`).
- **4.5 Auction bids** – this function is required if the detail of an auction is displayed. It returns a list of bids of the auctions from the newest one to the oldest one.

5. Category Management

Although a hierarchy of categories must be supported in a real system, we do not support it in our system. Since this function includes only CRUD operations implemented by one SQL statement, we do not described them in more detail.

Table: Category, Responsibility: Admin

- **5.1 New category**
- **5.2 Category update**

- **5.3 Category delete**
- **5.4 List of categories**

6. Comment Management

Table: Comment, Responsibility: A user to win an auction

- **6.1 New Comment**
See the function: **3.4 A list of winning auctions without comment**
- **6.2 Comment update**
See the function: **3.5 A list of winning auctions**
- **6.3 Comment delete**
See the function: **3.5 A list of winning auctions**
- **6.4 Comment of auctions**
Responsibility: Auctioneer

7. Other Functions

- **7.1 User Notification** – this function is executed in a time period, it finds auctions to finish soon and send an e-mail to all observers. In our system, mails are not sent, only a record is inserted in the table Notification. This function is described in Section 5.2.

5.2 Detail Description of Functions

Function 3. List of auctions

Input: the user id \$idBidder for functions 3.2 – 3.5

- **3.1 Continuing auctions**

```
select a.* from auction a, subcategory sb
where a.end < CURRENT_TIMESTAMP and sb.id=idCategory
      and sb.id=a.subcategory
order by (CURRENT_TIMESTAMP-end) asc;
```

- **3.2 Observed auctions**

```
select a.* from auction a, follows f, subcategory sb
where a.id=f.ID_auction and f.ID_user=$idBidder
      and sb.id=idCategory and sb.id=a.subcategory
order by (CURRENT_TIMESTAMP-end) asc;
```

- **3.3 Auctions with bid**

```
select a.* from auction a, bid b, subcategory sb
where a.id=b.ID_auction and b.ID_user=$idBidder
      and sb.id=idCategory and sb.id=a.subcategory
order by (CURRENT_TIMESTAMP-end) asc;
```

- **3.4 List of winning auctions without comment**

```
select a.* from auction a, subcategory sb
where a.ID_user_max_bid=$idBidder
      and a.id not in
      (select ID_auction from Comment
       where ID_buyer =$idBidder)
      and sb.id=idCategory and sb.id=a.subcategory
order by creation asc;
```

- **3.5 List of winning auctions**

```
select a.* from auction a, subcategory sb
where a.ID_user_max_bid=$idBidder
      and a.id in
      (select ID_auction from CommentUser
       where ID_buyer =$idBidder)
      and sb.id=idCategory and sb.id=a.subcategory
order by creation asc;
```

- **3.6 Actions of auctioneer**

Input: \$idOwner – the id of the auctioneer

```
select a.* from auction a, subcategory sb
where a.owner = $idOwner
      and sb.id=idCategory
      and sb.id=a.subcategory
order by creation asc;
```

Function 4.3 Automatic bid

Parameters: \$id_user, \$id_auction, \$max_value

Function Description:

This function is automatically executed in the case of insertion of a new record in the table Bid or the update of Bid.max_value. This function updates the attributes Auction.max_bid_value and Auction.max_bid_user and it is a transaction.

1. Read max_bid_value of the auction with \$id_auction:

```
SELECT max_bid_value FROM bid
      WHERE bid.id_auction = $id_auction
```

2. If max_bid_value is null then max_bid_value = \$max_value.
3. If \$max_value < max_bid_value this function is finished.
4. Find a new value of max_bid_value for the auction using:

```
SELECT MAX(max_value) FROM bid
      WHERE bid.id_auction = $id_auction and
      max_value NOT IN
      (SELECT MAX(max_value) FROM bid
      WHERE bid.id_auction = $id_auction)
```

It is the second highest bid. If MAX(max_value) = NULL, we set max_bid_value of the auction to min_bid_value, otherwise we set MAX(max_value) + 10.

5. Set max_bid_user of the auction with the highest first bid of the auction:

```
SELECT id_user, bid_date FROM bid
      WHERE bid.id_auction = $id_auction and
      max_value IN
      (
        SELECT MAX(max_value) FROM bid
          WHERE bid.id_auction = $id_auction
      ) and
```

```

bid_date <= all
(
    SELECT bid_date FROM bid
    WHERE max_value IN
    (
        SELECT MAX(max_value) FROM bid
        WHERE bid.id_auction = $id_auction
    )
)

```

Function 7.1 User Notification

Parameters: \$period includes the number of hours to a finishing auction when the notification must be repeatedly sent, e.g. 12h.

Function description:

1. This function is executed at an interval, e.g. 30min.
2. The current time can be changed during execution of this functions; therefore, we store it at the beginning of the function, e.g. `$p_current_datetime = CURRENT_TIMESTAMP`.
3. We create a cursor scans continuing auctions to finish soon and an e-mail is sent only one time per hour.

The function *GetPeriod* returns the number of hours between two times, e.g. it returns 0 if the difference is 45min, 1 in the case of 68min etc.

```

SELECT * FROM Auction a, Follows f WHERE
-- we consider only auctions to finish soon
a.end >= $p_current_datetime AND
GetPeriod(a.end, $p_current_datetime) > $period AND
-- scans all observers of these auctions
a.id_auction = f.id_auction AND
-- find out if the notification has been sent this hour
AND (
    SELECT Count(*) FROM Notification
    WHERE n.id_auction = f.id_auction AND

```

```

        n.id_user = f.id_user AND
        n.period = GetPeriod(a.end, $p_current_datetime)) = 0

```

Notice: It is necessary to check the performance of the query, since the number of records compared for `end >= CURRENT_TIMESTAMP AND GetPeriod(a.end, $p_current_datetime) > period` depends on the DBMS used. In the case of the low performance, e.g. tables are sequentially scanned, it is necessary to change the physical database design. We must keep in mind that locking tables Auction a Follows at a longer interval, negatively influence the performance of the system; therefore, it is executed at an interval.

4. A record (f.id_auction, f.id_user, GetPeriod(a.end, \$p_current_datetime)) is inserted in the table Notification for each record found. In a real system, an e-mail is sent.
5. This function automatically deletes notifications for already finished auctions (it is necessary due to a possibly huge number of records in the table and we do not need to handle history of notifications).

```

DELETE FROM Notification WHERE id_auction, id_user IN
(
    SELECT n.id_auction, n.id_user
    FROM Auction a, Follows f, Notification n WHERE
        -- only finished auctions are considered
        a.end < CURRENT_TIMESTAMP AND
        -- only observers of these auctions are considered
        a.id_auction = f.id_auction
        -- only observers receiving a notification
        -- are considered
        f.id_auction = n.id_auction AND f.id_user = n.id_user
)

```

6 Design of User Interface

6.1 Menu

1. **Auctions** (*Responsibility*: Admin, User, Auctioneer)
 - (a) **Continuing auctions** – action: 3.1 Continuing auctions
 - (b) **My auctions** – action: 3.6 Auctions of auctioneer
 - (c) **Observed auctions** – action: 3.2 Observed auctions
 - (d) **Auctions with my bid** – action: 3.3 Auctions with bid
 - (e) **List of winning auctions without comment** – action: 3.4 List of winning auctions without comment
 - (f) **List of winning auctions** – action: 3.5 List of winning auctions

The system offers a link

- showing a detail of each auction (action 2.4 Auction Detail, see Section 6.2).
 - to an update form of the auction for auctions of the Auctioneer: action 2.2 Auction update.
 - to a delete operation of the function: action 2.3 Auction delete
2. **New auction** (*Responsibility*: User) – action: 2.1 New Auction
 3. **My profile** (*Responsibility*: User) – action: 1.5 User Detail
 4. **Administration** (*Responsibility*: Admin)
 - (a) **User Management**
 - i. **User insert** – action: 1.1 User insert
 - ii. **List of users** – action: 1.4 List of users
These action are offers for each user: 1.2 User update, 1.3 User delete, 1.5 User detail
 - (b) **Category management** – similar to the user management.

6.2 Auction Detail Form

Function 2.4
Auction Detail



Auction name

Auction description

Auction Status:

- New
- Continuing
- Finished

When the auction
status is finished
bids are not
displayed.

Kindle Keyboard 3G		
Prodávám novou, nerozbalenou čtečku Amazon Kindle 3G, která umožňuje přístup na internet zdarma přes Wifi i 3G síť.		
 Auction status: Continuing 2 days 7 hours to the end	 Auctioneer: Marcus Auctions of auctioneer	
Bid		
Current cost: 4100 Kč	Bid: <input type="text" value="4200"/> Kč <input type="button" value="Bid"/>	
Automatic bids		
Maximum cost:	<input type="text" value="6000"/> Kč	<input type="button" value="Run"/>
History fo bids		
User	Cost	Date
laski	4 100,00 Kč	Čt 12 dub 2012 09:58:52
Indy	4 000,00 Kč	Po 09 dub 2012 20:29:44

Login of auctioneer
Function 3.6. Auctions of
auctioneer

Function 4.1.
New bid

Function 4.3.
Automatic bids

Function 4.5 History of
bids

7 Often Issues in Analysis

Code	Group	Description
1.1	Incomplete analysis	It does not include any assignment specification.
1.2		It does not include any data model.
1.3		It does not include any state analysis.
1.4		It does not include any functional analysis.
1.5		It does not include any design of user interface (UI).
2.1	Project header	It does not include any header – the project name, course, login, date.
3.1	State analysis	The individual states are not related to the content of the tables.
4.1	UI design	The number of UI forms is low.
4.2		A form is trivial.
4.3		A form does not include the number and title of functions for individual UI components.

Group of issues: Detail function description

Code	Description
5.1	Forbidden (cascade) delete.
5.2	A function is trivial.
5.3	A function is not divided to individual atomic parts.
5.4	A function is described only using words (no SQL command is used).
5.5	A function is described only using SQL.
5.6	An SQL command to a function's part is missing.
5.7	A function is described using PL/SQL or T-SQL (parts specific for PL/SQL or T-SQL, e.g. the function GetDate(), is possible to replace by a word description, e.g. current_date).
5.8	There no parameters of a function.
5.9	A description of more atomic parts is included into one point of the function description.
5.10	A description of an SQL SELECT is included in more points – SELECT is an atomic operation.
5.11	An inappropriate usage of a cursor – the cursor is used instead of the aggregation functions (SUM, COUNT, an so on).
5.12	An inappropriate usage of a cursor – one INSERT can be used.
5.13	An inappropriate usage of a cursoru – one UPDATE can be used.
5.14	An inappropriate usage of a cursor – one DELETE can be used.
5.15	A syntax error in an SQL command.
5.16	A function header does not correspond to the individual parts of the function.
5.17	A function describes individual actions of UI.
5.18	SQL Formatting – it is necessary to distinguish SQL commands (e.g. using another font type) from other text to make reading more clear.