VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

A sample project to the subjects Database Systems I and II

Information System of Auctions

Radim Bača, Peter Chovanec, Michal Krátký, Petr Lukáš

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

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

1 Project Specification

MOTIVATION

We need an information system for the management of simple electronic auctions in the company. Using these auctions, the system will provide the sale of unnecessary assets. The main purpose of the system will be to create an auction with the possibility of bidding. Moreover, the system will also allow to monitor auctions and add comments to them.

ROLES

The role with the highest privileges will be **company manager**, who provide the management of the company assets and their depreciation. Furthermore, the system will be used only by users created by the company manager. These users will be divided into **auction manager** and **auctioneer**. Auction managers will create auctions of depreciated assets from a certain category. In summary, we will call all roles in the system **user**. All users will be able to participate in auctions and bid on them.

INPUTS:

The system will mainly deal with auctions and bids. In the case of **auction**, we will be obliged to record its name, description, start and end timestamps, initial price and the auction manager who created the auction.

Each auction will belong to exactly one **category**, which is characterized by its name and description. Many auctions can belong to the same category. Only a company manager or auction manager will be able to create a new auction. The company manager will be able to update or delete any auction that has not yet started. The auction manager will only be able to update or delete the auction created by him, but only if the auction has not yet started. The management of the categories will be provided only by the company manager.

In the case of **user**, we will be interested in his e-mail, name and surname, address (street, city, postal code, state), role (it means if he is company manager, auction manager or auctioneer) and a password in encrypted form. Optionally, the user will be provided with a telephone contact and a timestamp of the last login to the system. Only a company manager can create or delete a user. The system will allow to restore the deleted users. Each user can update his data. The company manager can update another user's information except for the password.

Each user can make more **bids** at the auction, where we record the amount of bids and the timestamp. The user can follow the selected auctions and can add **comments** to the auctions. For a comment, we will save when and by whom it was created, and the comment may be a response to another comment.

OUTPUTS:

The main output will be a **list of auctions** with the possibility of filtering, which will be available to all users. In addition to the basic information about the auction, the name of the category will also be part of the list. Moreover, a timestamp and the value of the last bid will be included in the list. In addition, the company manager will see in the list the contact details of the user who created the auction and the contact details of the user who made the last bid. Auctions will be listed in descending order by end date. Only open auctions and auctions in a certain category can be displayed in the list. The user will be able to set the filter to: (i) the auctions he has created, (ii) the auctions he has bid on, and (iii) the auctions he is following. The list will support searching by keyword, the match of which will be searched in the substrings of the name and description of the auction. The list will distinguish between unstarted, ongoing and closed auctions.

Another output available to all users will be **auction detail**. It will contain a list of bids and a list of comments. The list of bids will be sorted in descending order according to the timestamp, comments will be sorted in ascending order according to the creation time. By default, only comments that do not respond to another comment will be visible. The detail will indicate the name of the user who created the auction. Comments will indicate the name of the commenting user. Within the auction detail, it will be possible to view the history of its description (see the FUNCTIONS section).

Auction statistics will be available for the company manager. The total amount that was obtained from the auctions will be displayed in the bar graph by individual calendar months or weeks. Similarly, a graph with the number of closed auctions in each month or week will be displayed in the auction statistics. By default, statistics will be displayed for the current calendar year, however the user will be able to select any year. The statistics will also include a list of users who participated in an auction in the selected year. The list of winning auctions will include the total amount the user paid for their winning auctions. It will be possible to sort the list in descending order according to both of these data.

FUNCTIONS:

The main task of the system will be to provide **bidding on auctions**, which will be accessible to all users. The system will not allow a bid to be made by the user who made the last bid. Therefore, the user cannot bid twice in a row at the same auction. The system also checks whether the value of the bid meets the minimum increase with respect to the previous bid, or to the initial value of the auction in the case of the first bid. The minimum increment is given by Table 1. In the case of successful bid, the user of the previous bid will be informed about the newly inserted bid, it means he will know that he is no longer the current winner of the auction.

From	To	Min. increase
0,00 CZK	19,90 CZK	1,00 CZK
20,00 CZK	99,99 CZK	5,00 CZK
100,00 CZK	499,99 CZK	10,00 CZK
500,00 CZK	1 999,99 CZK	20,00 CZK
2 000,00 CZK	4 999,99 CZK	50,00 CZK
5 000,00 CZK	9 999,99 CZK	100,00 CZK
10 000,00 CZK	19 999,99 CZK	200,00 CZK
20 000,00 CZK	49 999,99 CZK	500,00 CZK
50 000,00 CZK	99 999,99 CZK	1 000,00 CZK
100 000,00 CZK	–	2 000,00 CZK

Tabulka 1: Table of minimum increments[1]

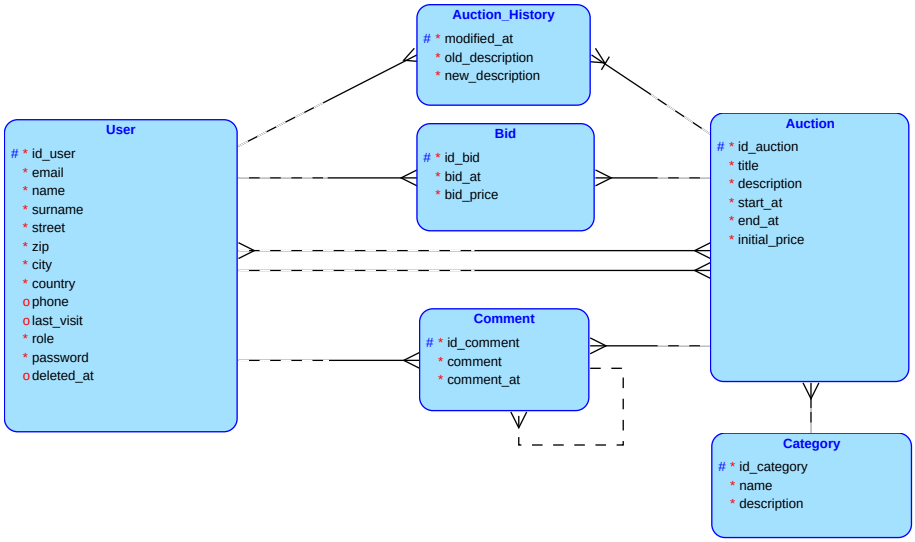
To prevent frauds, the system will check **modifications of auctions**. It will not be possible to make any changes to the auction that are already in progress or have ended. In the case of modification of the description of an upcoming auction, the system will save **change history**. The system automatically records who made this update and when, and what the description looked like before and after the change.

The system will check the following auctions at regular intervals, and if the following auction approaches the end, it will send **notifications to users** following the auction. For closed auctions, all following users will be removed at the same time.

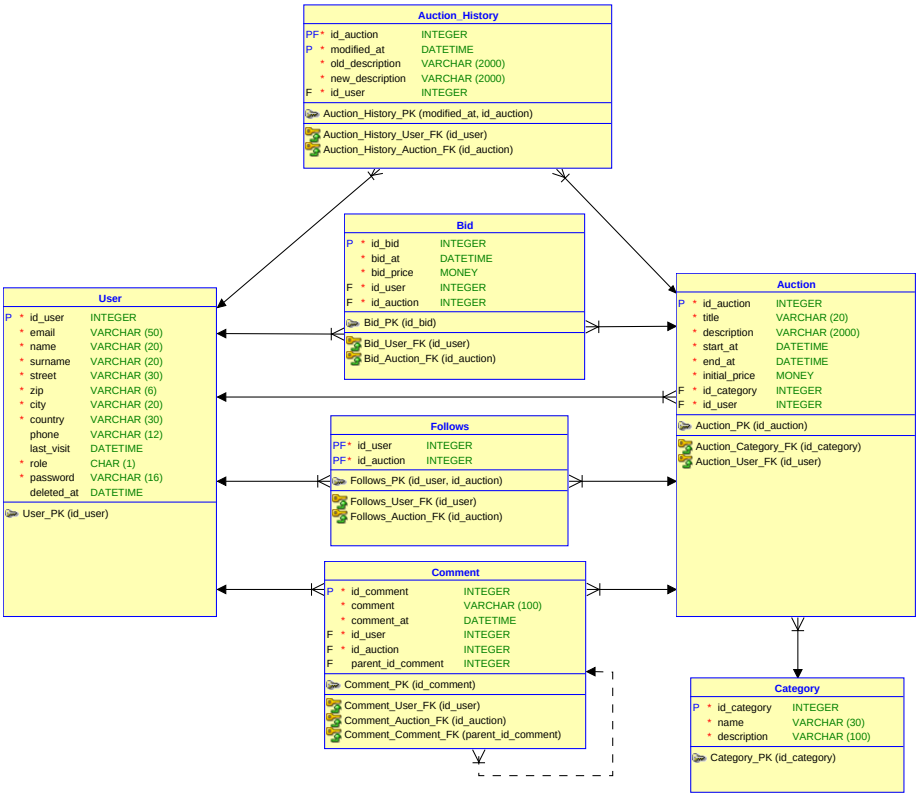
[1] It is excepted, the table of minimum increments can change over time.

2 Data Analysis

Conceptual Data Model



Relational Data Model



Data Dictionary

The description of individual tables is given in the following data dictionary.

Table **User**

Attribute name	Data type	Length	Key	Null	IR	Description
id_user	INTEGER		Primary	No		Automatically incremented PK
email	VARCHAR	50		No		E-mail used for login
name	VARCHAR	20		No		First name
surname	VARCHAR	20		No		Last name
street	VARCHAR	30		No		Street
zip	VARCHAR	6		No		ZIP
city	VARCHAR	20		No		City
country	VARCHAR	30		No		Country
phone	VARCHAR	12		Yes		Phone number
last_visit	DATETIME			Yes		Timestamp of last login
role	CHAR	1		Ne	1	Role
password	VARCHAR	16		No		Password encrypted by algorithm MD5
deleted_at	DATETIME			Yes		Timestamp of user delete

Table **Auction**

Attribute name	Data type	Length	Key	Null	IR	Description
id_auction	INTEGER		Primary	No		Automatically incremented PK
title	VARCHAR	20		No		Title of auction
description	VARCHAR	2000		No		Description of auction
start_at	DATETIME			No	2	Timestamp of start
end_at	DATETIME			No	2	Timestamp of end
initial_price	MONEY			No	3	Initial price of auction
id_category	INTEGER		Foreign (Category)	No		Category of the auction
owner_id_user	INTEGER		Foreign (User)	No		Auction manager

Table **Bid**

Attribute name	Data type	Length	Key	Null	IR	Description
id_bid	INTEGER		Primary	No		Automatically incremented PK
bid_at	DATETIME			No	4, 5	Timestamp of bid
bid_price	MONEY			No		Price of auction after bid
id_user	INTEGER		Foreign (User)	No		Bidding user
id_auction	INTEGER		Foreign (User)	No	5	Bid auction

Table **Comment**

Attribute name	Data type	Length	Key	Null	IR	Description
id_comment	INTEGER		Primary	No		Automatically incremented PK
comment	VARCHAR	100		No		Content of comment
comment_at	DATETIME			No		Timestamp of creation
id_user	INTEGER		Foreign (User)	No		Commenting user
id_auction	INTEGER		Foreign (Auction)	No		Commented auction
parent_id_comment	INTEGER		Foreign (Comment)	Yes		Commented comment

Table **Follows**

Attribute name	Data type	Length	Key	Null	IR	Description
id_user	INTEGER		Primary, Foreign (User)	No		Following user
id_auction	INTEGER		Primary, Foreign (Auction)	No		Followed auction

Table **Category**

Attribute name	Data type	Length	Key	Null	IR	Description
id_category	INTEGER		Primary	No		Automatically incremented PK
name	VARCHAR	30		No		Name of category
description	VARCHAR	100		No		Description of category

Table **Auction_History**

Attribute name	Data type	Length	Key	Null	IR	Description
id_auction	INTEGER		Primary, Foreign (User)	No		Modified auction
modified_at	DATETIME			No		Timestamp of modification
old_description	VARCHAR	2000		No		Original auction description
new_description	VARCHAR	2000		No		New auction description
id_user	INTEGER		Foreign (User)	No		User who made the change

Integrity restrictions:

1. `role` must contain values „V“ for company manager , „S“ for auction manager, or „D“ for auctioneer.
2. `start_at` $<$ `end_at`.
3. default value `initial_price` $= 0$.
4. `bid_date` $<$ `#auction.start_at` where `#auction` je the auction defined by the value `id_auction` (it means the auction we are bidding on).
5. Combination of values (`bid_at`, `auction_id`) is unique.

3 Formal Analysis

Table **User**

1. **A set of FDs**:
   - `id_user` → `email`, `name`, `surname`, `street`, `zip`, `city`, `country`, `phone`, `last_visit`, `role`, `password`, `deleted_at`
   - `email` → `id_user`, `name`, `surname`, `street`, `zip`, `city`, `country`, `phone`, `last_visit`, `role`, `password`, `deleted_at`
   - `zip` → `city`, `country`
2. **Minimalization of the set of FDs**:
   - `id_user` → `email`, `name`, `surname`, `street`, `zip`, `phone`, `last_visit`, `role`, `password`, `deleted_at`
     Based on the transitivity rule on the right side, we will omit `city` and `country`.
   - `email` → `id_user`
     Based on the transitivity rule on the right side, we will omit everything except `id_user`.
   - `zip` → `city`, `country`
3. **Covers**:
   - {`id_user`}+ = {`id_user`, `email`, `name`, `surname`, `street`, `zip`, `city`, `country`, `phone`, `last_visit`, `role`, `password`, `deleted_at`}
   - {`email`}+ = {`email`, `id_user`, `name`, `surname`, `street`, `zip`, `city`, `country`, `phone`, `last_visit`, `role`, `password`, `deleted_at`}
4. **Keys**:
   - $K_1 =$ {`id_user`}
   - $K_2 =$ {`email`}
5. **Normal form**:
   2NF – there is a dependency between non-key attributes: `zip` → `city`, `country`.
6. **Decomposition**:
   - $R_1 =$ (`zip`, `city`, `country`)
   - $R_2 =$ (`id_user`, `email`, `name`, `surname`, `street`, `zip`, `phone`, `last_visit`, `role`, `password`, `deleted_at`)

   Note: With respect to the purpose of the database, we will not consider the decomposition of the table `User`.

Table **Auction**

1. **A set of FDs**:
   - `id_auction` → `title`, `description`, `start_at`, `end_at`, `initial_price`, `id_category`, `owner_id_user`

2. **Minimalization of the set of FDs**:
   This set can not be further minimized.
3. **Covers**:
   - $\{$id_auction$\}+ = \{$id_auction, title, description, start_at, end_at, initial_price, id_category, owner_id_user$\}$
4. **Keys**:
   - $K_1 = \{$id_auction$\}$
5. **Normal form**:
   BCNF – for each FD, its left side is a key or its superset.

Table **Bid**

1. **A set of FDs**:
   - id_bid $\rightarrow$ bid_at, bid_price, id_user, id_auction
   - id_auction, bit_at $\rightarrow$ id_bid, bid_price, id_user
2. **Minimalization of the set of FDs**:
   - id_bid $\rightarrow$ bid_at, bid_price, id_user, id_auction
   - id_auction, bit_at $\rightarrow$ id_bid
     Based on the transitivity rule on the right side, we will omit everything except id_bid.
3. **Covers**:
   - $\{$id_bid$\} \rightarrow \{$id_bid, bid_at, bid_price, id_user, id_auction$\}$
   - $\{$id_auction, bid_at$\} \rightarrow \{$id_auction, bid_at,id_bid, bid_price, id_user$\}$
4. **Keys**:
   - $K_1 = \{$id_bid$\}$
   - $K_2 = \{$id_auction, bid_at$\}$
5. **Normal form**:
   BCNF – for each FD, its left side is the key (i.e. one of the keys) or its superset.

Table **Comment**

1. **A set of FDs**:
   - id_comment $\rightarrow$ comment, comment_at, id_user, id_auction, parent_id_comment
2. **Minimalization of the set of FDs**:
   This set can not be further minimized.
3. **Covers**:

- $\{$id_comment$\}+ = \{$id_comment, comment, comment_at, id_user, id_auction, parent_id_comment$\}$

4. **Keys**:
   - $K_1 = \{$id_comment$\}$
5. **Normal form**:
   BCNF – for each FD, its left side is the key (i.e. one of the keys) or its superset.

Table **Follows**

Table `Follows` does not contain non-trivial FD and is therefore in **BCNF** (i.e. there is no FD that would violate the condition for BCNF).

Table **Category**

1. **A set of FDs**:
   - id_category $\rightarrow$ name, description
2. **Minimalization of the set of FDs**:
   This set can not be further minimized.
3. **Covers**:
   - $\{$id_category$\}+ = \{$id_category, name, description$\}$
4. **Keys**:
   - $K_1 = \{$id_category$\}$
5. **Normal form**:
   BCNF – for each FD, its left side is the key or its superset.

Table **Auction_History**

1. **A set of FDs**:
   - id_auction, modified_at $\rightarrow$ old_description, new_description, id_user
2. **Minimalization of the set of FDs**:
   This set can not be further minimized.
3. **Covers**:
   - $\{$id_auction, modified_at$\}+ = \{$id_auction, modified_at, old_description, new_description, id_user$\}$
4. **Keys**:
   - $K_1 = \{$id_auction, modified_at$\}$
5. **Normal form**:
   BCNF – for each FD, its left side is the key or its superset.

4 State Analysis

We define following status of auctions and status of users:

Status of Auction:

- **Upcoming** – `current_timestamp` $<$ `#auction.started_at`.
- **Running** – `current_timestamp BETWEEN #auction.started_at AND #auction.end_at`.
- **Finished** – `current_timestamp` $>$ `#auction.end_at`.

Note: `#auction` represents the record of auction, where we define the status in table `Auction`.

Status of User:

- **Active** – `#user.deleted_at IS NULL`.
- **Deleted** – `#user.deleted_at IS NOT NULL`.

Pozn.: `#user` represents the record of user, where we define the status in table `User`.

5 Functional Analysis

5.1 List of Functions

1. **User Management**
   Responsibility: Company Manager

   - **1.1 User Insert**
   - **1.2 List of Users**
   - **1.3 User Detail**
     Responsibility: Company Manager; Auction Manager and Auctioneer can see only the own record
   - **1.4 User Update**
     Responsibility: Company Manager; Auction Manager and Auctioneer can see only the own record
   - **1.5 User Disable** – set the attribute `deleted_at` to current timestamp; the system does to allow to provide operation `DELETE` on database.
   - **1.6 User Enable** – set the attribute `deleted_at` to NULL.
   - **1.7 List of Roles** – returns table with the list of roles according to IR 1 (see Chapter 2).

2. **Auction Management**

   - **2.1 New Auction**
     Responsibility: Auction Manager
   - **2.2 List of Auctions** – non-trivial query, see Chapter 5.2.
     Responsibility: User
   - **2.3 Auction Detail**
     Responsibility: User
   - **2.4 Auction Update** – transaction, see Chapter 5.2.
     Responsibility: Company Manager; Auction Manager only his auctions
   - **2.5 Auction Delete** – delete is allowed only in 'Upcoming' status; all relating records in table `Auction_History` will be cascade deleted[2].
     Responsibility: Company Manager
   - **2.6 History of Changes** – the list of auction updates
     Responsibility: Auction Manager

3. **Auction Following**
   Responsibility: User

   - **3.1 Follow the Auction** – insert record into table `Follows` for specified auction and user.
   - **3.2 Unfollow the Auction** – delete record from table `Follows` about specified auction and user.
   - **3.3 Is Auction Followed** – check in table `Follows`, if specified user follows specified auction.

4. **Bidding**
   Responsibility: User

[2] Cascade deleting in table `Bid` is not necessary since, the auction does not have any bids yet.

- **4.1 New Bid** – transaction, see Chapter 5.2.
- **4.2 List of Auction Bids** – function invoked by detail of the auction and it returns the list of bids for the specified auction ordered from the newest to the oldest bid.
  Note: Update neither delete of the bids is not allowed.

5. **Category Management**
   Responsibility: Company Manager

   - **5.1 New Category**
   - **5.2 List of Categories** – the list contains all information about the categories including the number of auctions in the categories.
     Responsibility: User
   - **5.3 Detail of Category**
   - **5.4 Update Category**
   - **5.5 Delete Category** – function allowed only, if does not exist auction of this category.

6. **Comment Management**
   Responsibility: User

   - **6.1 New Comment** – the comment is allowed to create to any auction and to any comment.
   - **6.2 List of Comments to Auction** – function invoked by detail of the auction and it returns the list of comments to specified auction together with the number of comments to them.
   - **6.3 List of Comments to Comment** – function invoked by detail of the comment and it returns the list of comments to specified comment with the number of comments to them.
     Note: Update neither delete of the comments is not allowed.

7. **Auction Statistics**
   Responsibility: Company Manager

   - **7.1 Auction Statistics** – non-trivial query, see Chapter 5.2.
   - **7.2 User Statistics** – non-trivial query, see Chapter 5.2.

8. **Other Functions**
   Responsibility: System

   - **8.1 Notification of Following Users** – transaction, see Chapter 5.2.
   - **8.2 Email** – function with input parameters `#email`, `#subject` and `#body` providing the sending email to specified `#email` with specified `#subject` and `#body`. Implementation is dependent on chosen DBMS.

5.2 Detail Description of Functions

Function 2.2 List of Auctions
Inputs:

- `#keyword` – keyword searched in name of description of the auctions; can be unspecified `NULL`
- `#only_unfinished` – function can return all or only unfinished auctions (values 0 or 1)
- `#id_category` – ID of the category or `NULL`
- `#id_user` – ID of the user or `NULL`
- `#id_creator` – ID of the user or `NULL`
- `#id_bidder` – ID of the user or `NULL`
- `#id_follower` – ID of the user or `NULL`

The function returns a list of auctions together with information about the category, the user who created the auction, the last bid and the user who made the last bid. The condition `WHERE` is dynamically created according to the input data.

```
SELECT
  Auction.id_auction, Auction.title, Auction.description, Auction.start_at,
  Auction.end_at, Auction.initial_price, Category.id_category,
  Category.name AS category_name, Creator.id_user AS id_creator,
  Creator.name AS creator_name, Creator.email AS creator_email,
  CASE
    WHEN CURRENT_TIMESTAMP < Auction.start_at THEN 'Upcoming'
    WHEN CURRENT_TIMESTAMP BETWEEN Auction.start_at AND Auction.end_at THEN
      'Running'
    ELSE 'Finished'
  END AS auction_state,
  LastBid.bid_at AS last_bid_at, LastBid.bid_price AS last_bid_price,
  LastBidder.id_user AS last_id_bidder, LastBidder.name AS last_bidder_name,
  LastBidder.email AS last_bidder_email
FROM
  Auction
  JOIN Category ON Auction.id_category = Category.id_category
  JOIN User Creator ON Auction.id_user = Creator.id_user
  LEFT JOIN Bid LastBid ON Auction.id_auction = LastBid.id_auction AND
    LastBid.bid_at >= ALL (
      SELECT Bid.bid_at
      FROM Bid
      WHERE Bid.id_auction = Auction.id_auction
    )
  LEFT JOIN User LastBidder ON LastBid.id_user = LastBidder.id_user
WHERE (creation of condition WHERE presented below)
ORDER BY Auction.end_at DESC
```

The condition `WHERE` is composed of the following expressions joined by a logical `AND`:

- If `#keyword IS NOT NULL`:
  `(Auction.title LIKE '%' + #keyword + '%' OR Auction.description LIKE '%' + #keyword + '%')`
- If `#only_unfinished = 1`:
  `(CURRENT_TIMESTAMP <= Auction.end_at)`
- If `#id_category IS NOT NULL`:
  `(Category.id_category = #id_category)`
- If `#id_creator IS NOT NULL`:
  `(Auction.id_user = #id_creator)`
- If `#id_bidder IS NOT NULL`:
  `EXISTS (SELECT * FROM Bid WHERE Auction.id_auction = Bid.id_auction AND Bid.id_user = #id_bidder)`
- If `#id_follower IS NOT NULL`:
  `EXISTS (SELECT * FROM Follows WHERE Auction.id_auction = Follows.id_auction AND Follows.id_user = #id_bidder)`

Function 2.4 Auction Update

Inputs:

- `#id_auction` – ID of the auction
- `#id_user` – ID of the user
- `#auction_new` – new values of the auctions after the update (structured variable[a])

Outputs:

- `#error_msg` – output error message (default value is `NULL`)

The function according to auction status checks whether an update can be performed, performs the update itself, and records the description change into the history. The function is treated as a transaction.

1. Select current values of all auction attributes into the structured variable `#auction_old`:

   ```
   SELECT *
   FROM Auction
   WHERE id_auction = #id_auction
   ```

2. Set the variable `#now` to current timestamp.

3. If `#auction_old.started_at < #now`, set the error message `#error_msg` to „The auction in the Upcoming neither Finished status cannot be updated!“ and terminate the transaction.

4. If `#auction_old.description <> #auction_new.description`, record the change history of the auction description:

   ```
   INSERT INTO Auction_History (id_auction, modified_at, old_description,
   new_description, id_user)
   VALUES (#id_auction, #now, #auction_old.description,
   #auction_new.description, #id_user)
   ```

5. Update of the auction is executed:

   ```
   UPDATE Auction
   SET
     title = #auction_new.title,
     description = #auction_new.description,
     start_at = #auction_new.start_at,
     end_at = #auction_new.end_at,
     initial_price = #auction_new.initial_price,
     id_category = #auction_new.id_category,
     id_user = #auction_new.id_user
   WHERE id_auction = #id_auction
   ```

[a] If the selected database system does not support structured variables, the input variables correspond to all the attributes of the auction.

Function 4.1 New Bid

Inputs:
- `#id_auction` – ID of the auction
- `#id_user` – ID of the user
- `#value` – value of the bid

Outputs:
- `#error_msg` – output error message (default value is `NULL`)

The function checks whether the user is not bidding on the auction where he made the last bid, and whether the bid value is higher than the allowable bid value. Consequently, the function inserts the bid and sends an informative e-mail to the user with the last highest bid. The function will be treated as a transaction.

1. Set the variable `#now` to current timestamp.
2. Set variables `#id_user_prev`, `#user_email` and `#current_price` to selected ID of the user with the last bid, his email and value of his bid. We consider only the first record of the query [a]:
   ```
   SELECT User.id_user, User.email, Bid.bid_price
   FROM Bid JOIN User ON Bid.id_user = User.id_user
   WHERE Bid.id_auction = #id_auction
   ORDER BY Bid.bid_at DESC
   ```
   If the query returns an empty result (the auction does not have a bid yet), the values `#id_user_prev`, `#user_email` a `#current_price` are set to `NULL`.
3. Set variables `#initial_price` and `#title` to initial price and the title of the auction:
   ```
   SELECT initial_price, title
   FROM Auction
   WHERE id_auction = #id_auction
   ```
4. If `#id_user_prev` = `#id_user`, set error message `#error_msg` to „It is not possible to bid on the same auction twice in a row.“ and terminate the transaction.
5. If `#current_price IS NULL`, the variable `#current_price` is set to `#initial_price`.
6. Determine the minimum value of a new bid `#min_bid` at the auction according to the Table 1 and the `#current_price`.
7. If `#value` < `#current_price + #min_bid`, set error message `#error_msg` to „The minimum bid value at the auction is {`#current_price + #min_bid`}.“ and terminate the transaction.
8. Insert bid into database:
   ```
   INSERT INTO Bid (id_auction, id_user, bid_at, bid_price)
   VALUES (#id_auction, #id_user, #now, #value).
   ```
9. If `#user_email` is not `NULL` and at the same time the following query returns any result:
   ```
   SELECT *
   FROM Follows
   WHERE id_user = #id_user AND id_auction = #id_auction,
   ```
   call Function 8.2 with input parameters:
   - `#email` ← `#user_email`,
   - `#subject` ← „New bid to auction {`#title`}“,
   - `#body` ←„A new bid was made for the auction {`#title`} to {`#value`} CZK.“.

[a] Integrity constraint 5 ensures an order of bids, it means there are no two bids at the same time for a given auction

Function 6.2 List of Comments to Auction
Inputs:
- #id_auction – ID of the auction

```
SELECT C1.id_comment,
       C1.comment,
       C1.comment_at,
       U1.name,
       U1.surname,
       (SELECT COUNT(*)
        FROM Comment C2
        WHERE C2.parent_id_comment = C1.id_comment) AS comments_count
FROM Comment C1
JOIN User U1 ON C1.id_user = U1.id_user
WHERE C1.id_auction = #id_auction
ORDER BY C1.comment_at DESC;
```

Function 6.3 List of Comments to Comment
Inputs:
- #id_comment – ID of the comment

```
SELECT C1.id_comment,
       C1.comment,
       C1.comment_at,
       U1.name,
       U1.surname,
       (SELECT COUNT(*)
        FROM Comment C2
        WHERE C2.parent_id_comment = C1.id_comment) AS comments_count
FROM Comment C1
JOIN User U1 ON C1.id_user = U1.id_user
WHERE C1.parent_id_comment = #id_comment
ORDER BY C1.comment_at DESC;
```

Function 7.1 Auction Statistics

Inputs:
- #year – selected year
- #stat_type – value „W“ or „M“

```
SELECT period(Auction.end_at) AS period, SUM(B1.bid_price) AS sum_price,
       COUNT(*) AS cnt
FROM Auction
JOIN Bid B1 ON Auction.id_auction = B1.id_auction
WHERE year(Auction.end_at) = #year
AND B1.bid_at >= ALL (SELECT B2.bid_at
                      FROM Bid B2
                      WHERE B2.id_auction = Auction.id_auction)
GROUP BY period(Auction.end_at);
```

- Depending on the value #stat_type „W“, resp. „M“, the function returns *period*(#date) week (according to the standard ISO 8601) or month from #date.
- Function *year*(#date) returns year from #date.

Function 7.2 User Statistics

Inputs:

- #year – selected year

```
SELECT User.id_user, User.name, User.surname, SUM(B.bid_price) AS sum_price,
       COUNT(*) AS cnt
FROM User
JOIN Auction ON User.id_user = Auction.id_user
JOIN Bid B1 ON Auction.id_auction = B1.id_auction
WHERE year(Auction.end_date) = #year
AND B1.bid_at >= ALL (SELECT B2.bid_at
                      FROM Bid B2
                      WHERE B2.id_auction = Auction.id_auction)
GROUP BY User.id_user, User.name, User.surname;
```

Function *year*(#date) returns year from #date.

Function 8.1 Notification of Following Users

Inputs: *(none)*

The function searches the auctions, which end in less than 5 hours, and sends an information email to all users who follow these auctions. The function is executed every 30 minutes. The function is treated as a transaction.

1. Set the variable #now to current timestamp.
2. Create a cursor that goes through auctions ending in less than 5 hours. The current record will be stored in a structured variable #row.

   ```
   SELECT user.email, auction.title, auction.end_at
   FROM user, follows, auction
   WHERE user.id_user = follows.id_user
     AND follows.id_auction = auction.id_auction
     AND hours_between(a.end_at, #now) <= 5;
   ```

3. In the loop, move in the cursor and call the Function 8.2 for each record with input parameters:
   - #email ← #row.email,
   - #subject ← „Auction #row.title is comming to end.“,
   - #body ← „Auction #row.title will be closed at #row.end_at.“.
4. This feature will also automatically delete users following already finished auctions.

   ```
   DELETE FROM follows
   WHERE id_auction IN (
      SELECT id_auction FROM auction
      WHERE end_at < #now
   );
   ```

The function *hours_between()* returns the number of whole hours between two times, e.g. if the difference is 45min, the function returns 0, if 68min, the function returns 1, etc.

6 Design of User Interface

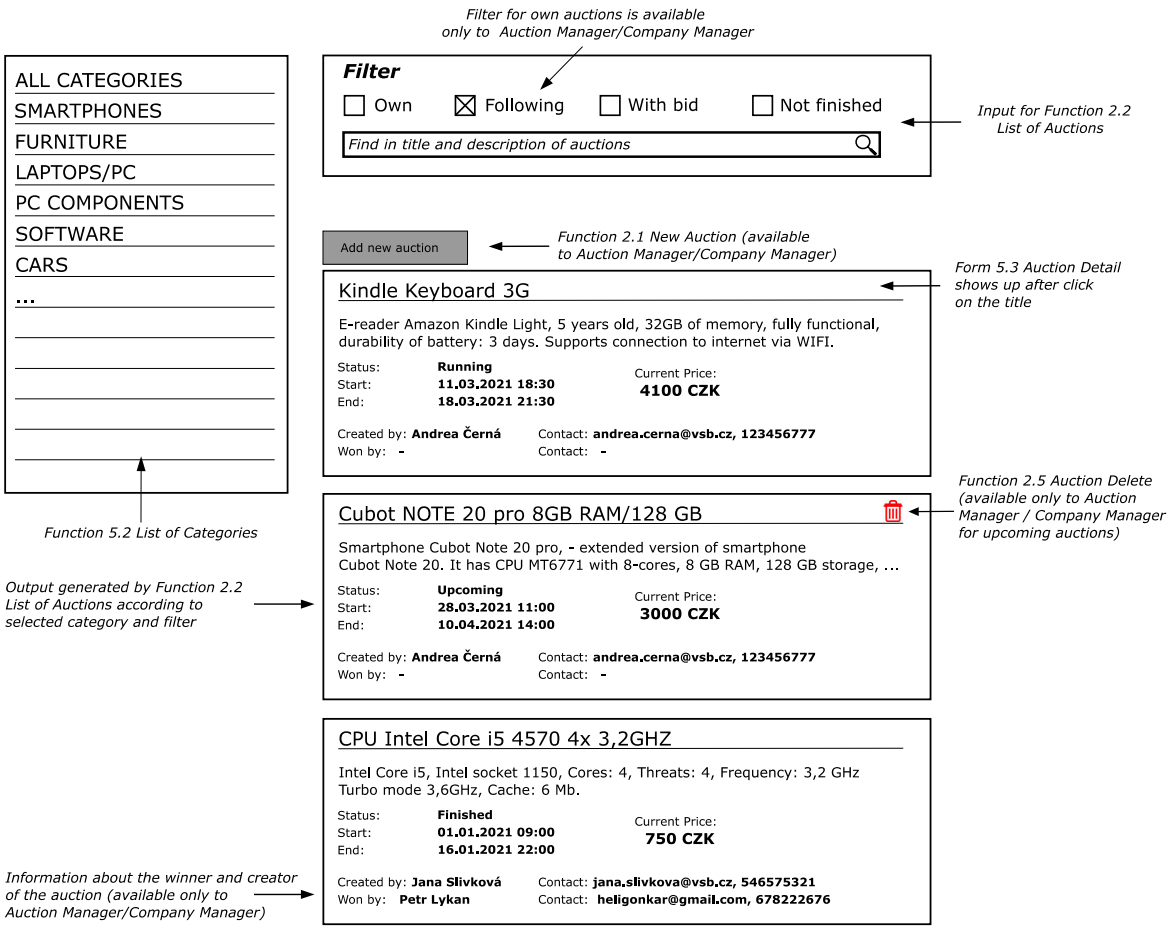
6.1 Menu

1. **Auction Overview** – visible to all users, see Form 5.1.
   The Auction Detail will be offered for each auction, see Form 5.2.
2. **User Management** – visible only to Company Manager, see Form 5.3.
3. **Category Management** – visible only to Company Manager, see Form 5.4.
4. **Statistics** – visible only to Company Manager, see Form 5.5.
5. **User Profile** – visible to all users, see Form 5.6.

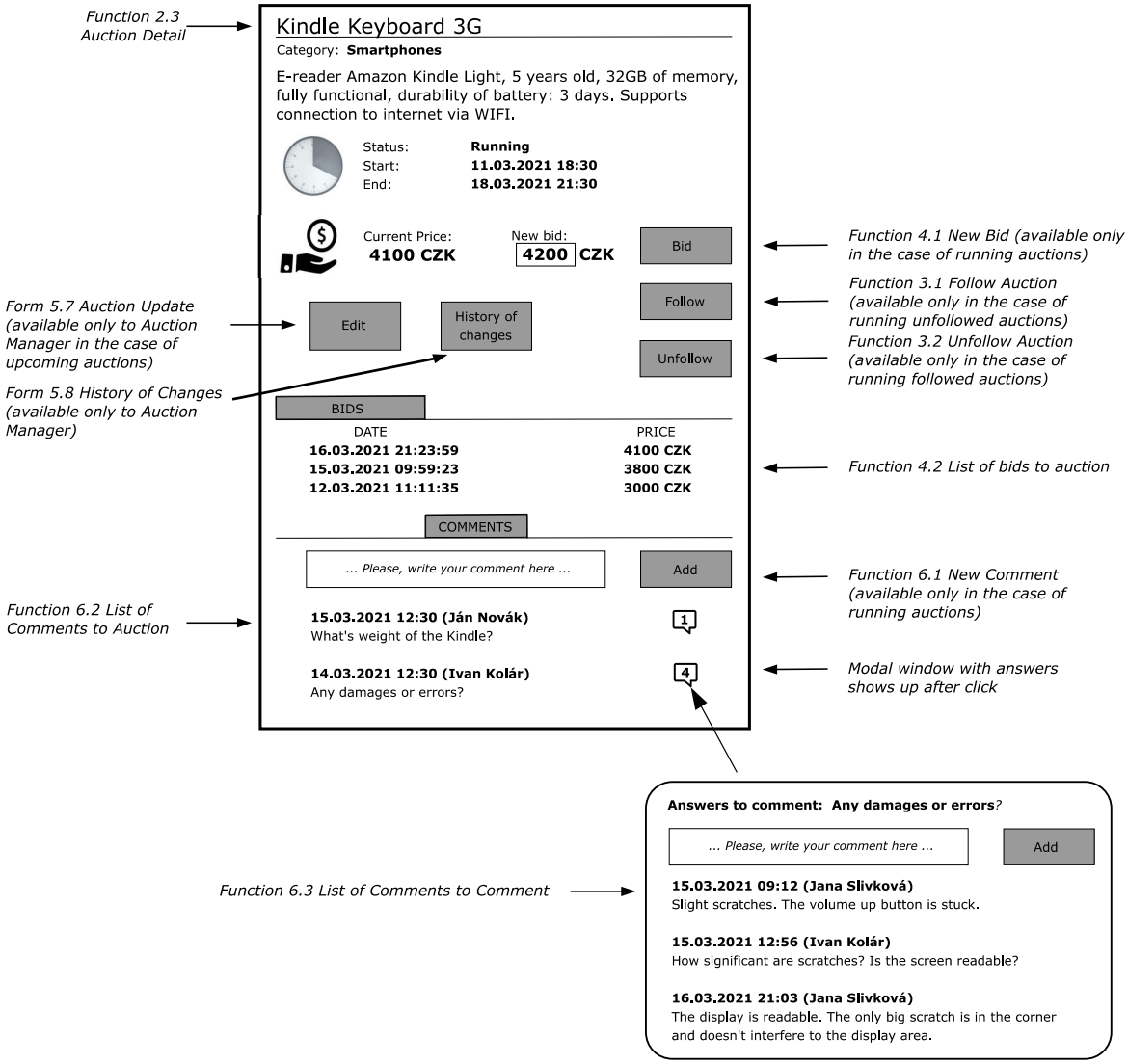
Note: Unless otherwise stated, the input parameters `id_user` and `id_auction` for the functions in the following forms represent the ID of the logged user and the ID of the viewed auction.

6.2 Detail Description of Forms

Form 5.1 Auction Overview



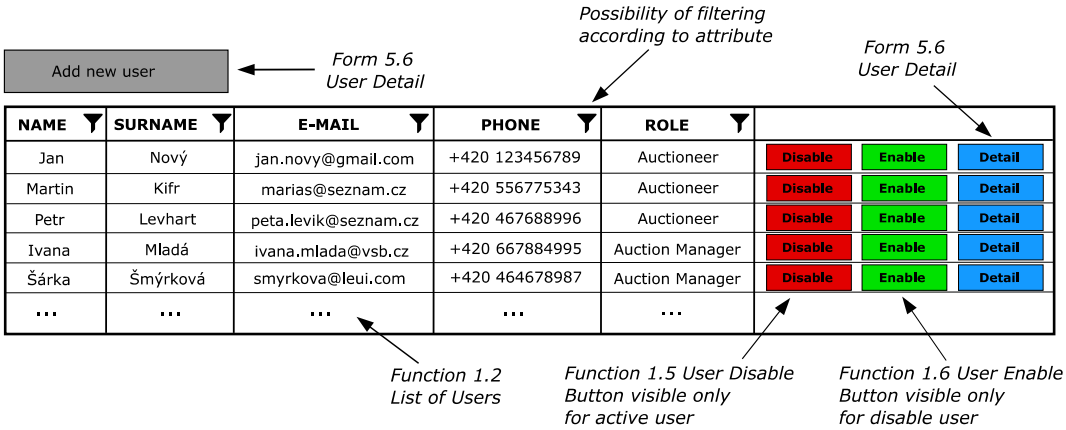
Form 5.2 Auction Detail



Notes:

- Bids and comments will be placed on tabs. For better the description, we present these components below each other.
- **Form 5.7 Auction Update** will be similar to Form 5.6 User detail, the update will use Function 2.4 Auction Update.
- **Form 5.8 History of Changes** will contain a simple list of the history of auction changes, and the Function 2.6 History of changes will be used to obtain the required data. The list will be read only.

Form 5.3 User Management



Notes:

- The button Add new user triggers Form 5.6 User Detail with empty edit fields. The form will contain a button Add User and will use Function 1.1 User Insert.
- The button Detail triggers Form 5.6 User Detail with edit fields filled by Function 1.3 User Detail. The form will contain a button Update User and will use Function 1.4 User Update.

Form 5.4 Category Management

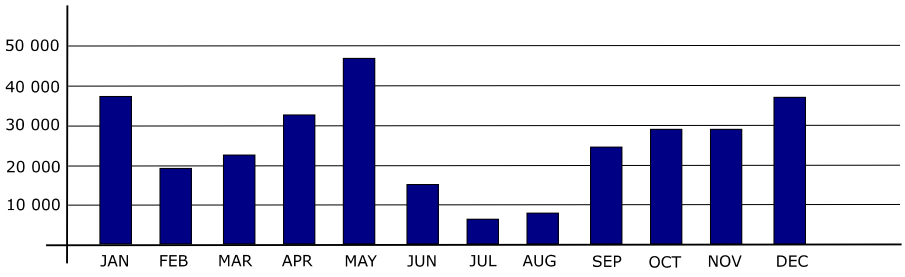
The category management will operate in a similar way as the user management described in Forms 5.3 and 5.6. The only difference is in the function for deleting categories (Function 5.5 Delete Category). In the case of categories, the category is physically deleted, and the delete button will not be available for categories with some auctions.

Form 5.5 Statistics

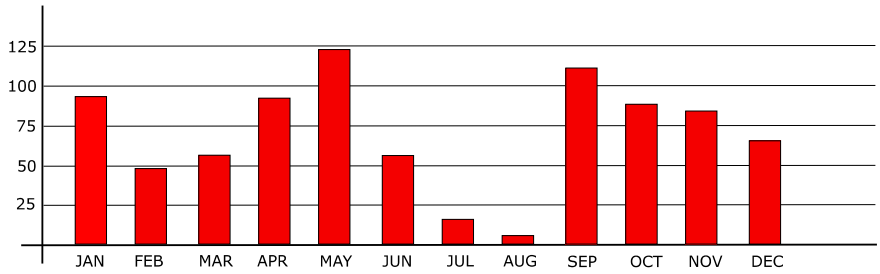
Year of view: **2020**

Type of view: (●) Months ( ) Weeks

Revenue from sold property



Number of finished auctions



Function 7.1
Auction Statistics

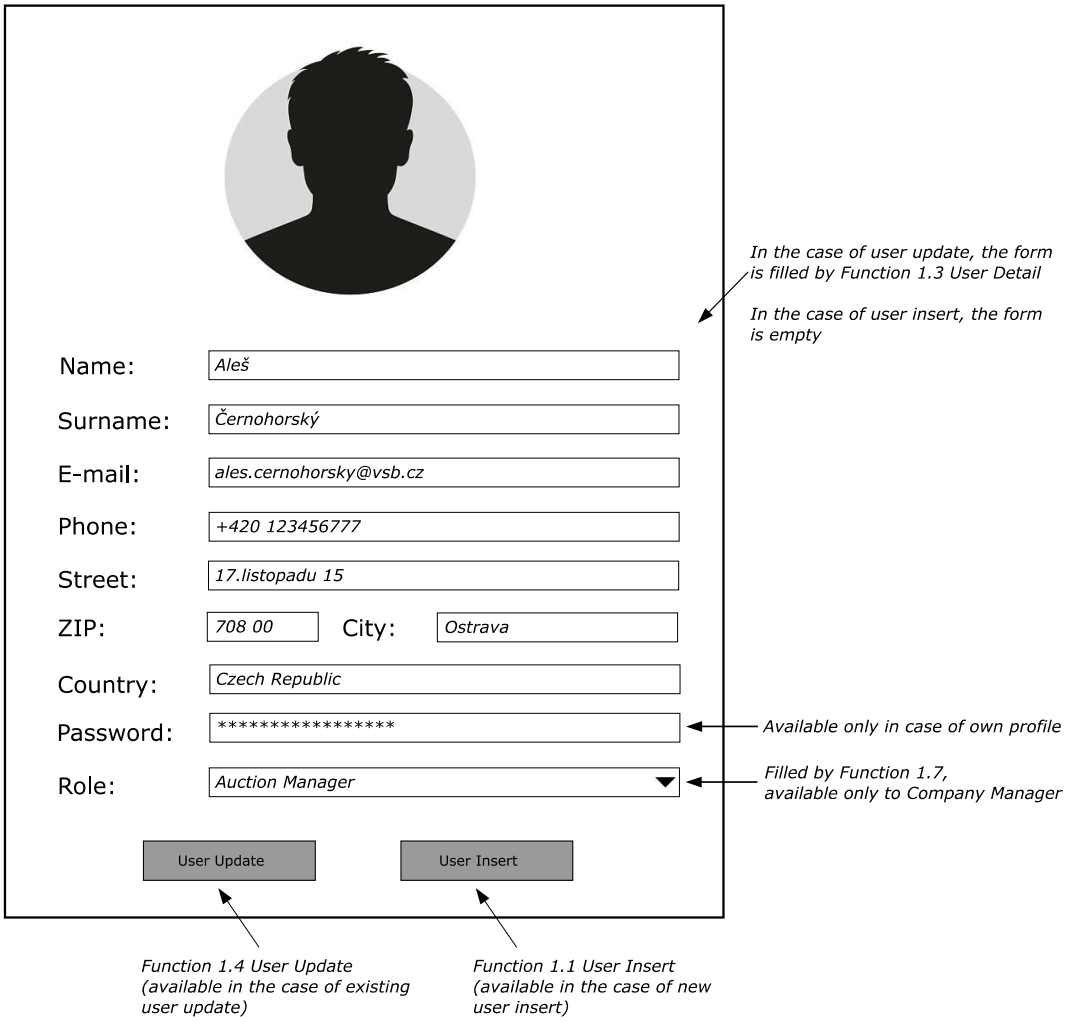
Possibility of sorting according to attribute

User Statistics

NAME	SURNAME	TOTAL PAID	WON AUCTIONS
Jan	Nový	8 100 Kč	3
Martin	Kifr	1 300 Kč	1
Petr	Levhart	2 700 Kč	5
Ivana	Mladá	4 440 Kč	6
Šárka	Šmýrková	3 100 Kč	2
...	...	...	...

Function 7.2
User Statistics

Form 5.6 User Profile



Note:

- In the case of adding a new user, the password is generated by the system and sent to the user by e-mail.