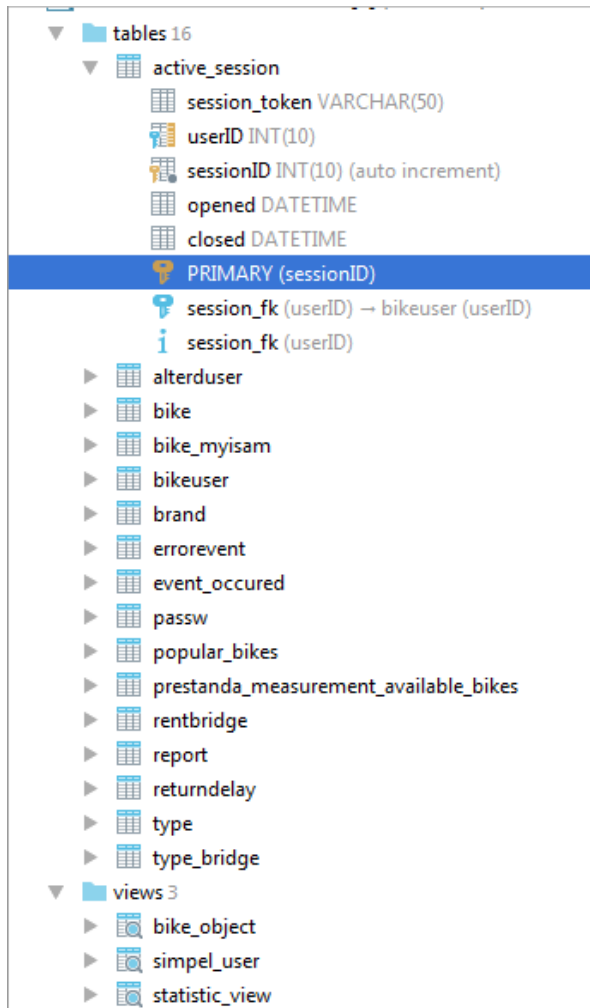


Översikt

Ex-arbetet bestod i att utsätta ett system jag byggt tillsammans med kursare Niklas Karlsson för en UX-granskning.

Programmet har en MySQL databas, en server i Java med REST framework Jax-RS, Jersey implementation, och en klient i JavaFX. Systemet kan skapa användare, låna ut cyklar, lägga till och ta bort cyklar i databasen, skicka mail.



En MySQL databas med totalt 16 tabeller. Databasen är normaliserad och foreign keys används konsekvent.

Ett antal event körs med bestämda intervaller, exempelvis körs ett event som kontrollerar om ett session varit öppen i mer än ett dygn. I så fall stängs den aktuella inloggningen.

Lösenord är krypterade med salt som utgörs av en 24 tecken lång randomiserad sträng.

Vidare finns tre vyer varav två ungefär motsvarar dataobjekt.

Databasen är inte särskilt stor, ändå har jag lagt till index på några kolumner.

På tabellen bikeuser finns en trigger som körs efter insert. Denna lägger in den gamla informationen i tabellen altereduser.

I tabellen bike sparas även en bild som en blob.

<div> <div></div> disabled_percent DECIMAL(5,2) <div></div> total_number_of_users INT(11) <div></div> male_users_percent DECIMAL(5,2) <div></div> female_users_percent DECIMAL(5,2) <div></div> other_users_percent DECIMAL(5,2) <div></div> male_20_to_30_percent DECIMAL(5,2) <div></div> male_30_to_40_percent DECIMAL(5,2) <div></div> male_40_to_50_percent DECIMAL(5,2) <div></div> male_50_or_more_percent DECIMAL(5,2) <div></div> female_20_to_30_percent DECIMAL(5,2) <div></div> female_30_to_40_percent DECIMAL(5,2) <div></div> female_40_to_50_percent DECIMAL(5,2) <div></div> female_50_or_more_percent DECIMAL(5,2) <div></div> other_20_to_30_percent DECIMAL(5,2) <div></div> other_30_to_40_percent DECIMAL(5,2) <div></div> other_40_to_50_percent DECIMAL(5,2) <div></div> other_50_or_more_percent DECIMAL(5,2) <div></div> new_loans_30_days_back_total INT(11) <div></div> new_loans_30_days_back_male_percent DECIMAL(5,2) <div></div> new_loans_30_days_back_female_percent DECIMAL(5,2) <div></div> new_loans_30_days_back_other_percent DECIMAL(5,2) <div></div> most_active_user_group_total VARCHAR(100) <div></div> least_active_user_group_total VARCHAR(100) <div></div> most_active_user_group_30_days_back VARCHAR(100) <div></div> least_active_user_group_30_days_back VARCHAR(100) <div></div> nr_1_popular_bike VARCHAR(70) <div></div> nr_2_popular_bike VARCHAR(70) <div></div> nr_3_popular_bike VARCHAR(70) <div></div> nr_4_popular_bike VARCHAR(70) <div></div> nr_5_popular_bike VARCHAR(70) <div></div> nr_1_least_popular_bike VARCHAR(70) <div></div> generated_by VARCHAR(40) <div></div> generated_by_id INT(10) </div>	<p>Statistick_view består av 38 kolumner, i princip alla kolumner är satta med funktioner. Jag har använt enum lokalt i databasen för att kategorisera användare efter kön och ålder.</p> <p>Ett event som körs en gång i veckan sparar alla kolumner i en rapporttabell.</p>
---	---

<div> ▼ routines 57 </div> <ul style="list-style-type: none"> bike_maintenance_percent() category_percent(VARCHAR, INT, INT): DECIMAL check_open_sessions() check_password_get_bikeuser(VARCHAR, VARCHAR, OUT INT) delete_bike(INT, OUT TINYINT) disabled_bikes_percent() execute_bike_loan(INT, INT, DATE, OUT DATE) get_all_bikes() get_bike_returnedDate_from_ID(INT, OUT TINYINT) get_category(INT): VARCHAR get_millisec() get_num_of_curr_available_bikes() get_num_of_total_bikes() get_report(INT, VARCHAR) get_total_num_bikes() getUserFromUserName(VARCHAR) insert_bike(VARCHAR, VARCHAR, SMALLINT, VARCHAR, SMALLINT, LONGBLOB, INT) insert_new_user(VARCHAR, VARCHAR, VARCHAR, DATE, VARCHAR, VARCHAR, VARCHAR, EN insert_new_user_old(VARCHAR, VARCHAR, VARCHAR, VARCHAR, VARCHAR, VARCHAR, VARC insert_prestanda_measurement(DECIMAL, DECIMAL, DECIMAL, DECIMAL, DECIMAL, DECIMAL is_bike_available(INT): TINYINT largest_change() least_active_user_group_30_days_back() least_active_user_group_total() loans_in_order(INT): VARCHAR loans_in_order_least_popular(INT): VARCHAR most_active_user_group_30_days_back() most_active_user_group_30_days_back_2() most_active_user_group_total() most_common_type(INT, OUT VARCHAR) new_loans_30_days_back_percent(VARCHAR): DECIMAL new_loans_30_days_back_total() num_of_curr_available_bikes() num_of_curr_available_bikes_2() num_of_usable_bikes() number_of_users() popular_bikes_in_order(INT) popular_bikes_in_order_2() popular_bikes_order_func(INT): VARCHAR return_bike(INT, INT, OUT TINYINT) return_bike_by_rent_id(INT) search_available_bikes(OUT MEDIUMTEXT) search_available_bikes_test(OUT MEDIUMTEXT) search_by_string(VARCHAR) search_next_available_bikes(INT, INT, OUT INT, OUT INT) search_next_available_bikes_not_previous_choise(INT, INT, INT, OUT INT, OUT INT) search_next_available_bikes_previous_choise(INT, INT, INT, OUT INT, OUT INT) size_of_search_available_bikes_mb() 	<p>Jag har 57 funktioner och procedurer, vissa är nästlade.</p>
--	---

I ex-arbetet ingick även att skriva select-satser för att få fram visst underlag av slaget: vilken vecka ökade kundbasen mest?

```
SELECT year(bu1.memberSince), monthname(bu1.memberSince),
week(bu1.memberSince), count(*) - (SELECT count(*) from bikeuser as
```

```
bu2 WHERE  
    DATE_FORMAT(bu1.memberSince, '%y %m %w') =  
    DATE_FORMAT(DATE_ADD(bu2.memberSince, INTERVAL -1  
WEEK), '%y %m %w'))  
    FROM bikeuser as bu1  
GROUP BY year(bu1.memberSince), month(bu1.memberSince),  
week(bu1.memberSince)  
ORDER BY 4 DESC;
```