# https://scontent-waw1-1.xx.fbcdn.net/v/t34.0-12/18716448_1454443317950417_1556560772_n.png?oh=0408673626e9314791a2d43fc7e6b0dc&oe=592A1B76

# Widoki

## book\_status

Wyświetla książki z ich aktualnym statusem, inne niż „dostępna”.

Łączymy tabele book i loan po id\_book, z tabeli loan wybieramy wszystkie rekordy które mają status niższy niż 4, a więc są aktywne (status 4 to zwrócona a 5 to anulowana). Do tego dołączamy z tabeli loan\_status nazwę konkretnego statusu. Ze zwróconych rekordów wybieramy najwyższy status (baza jest zaprojektowana w taki sposób, że oprócz 3 „wypożoczona”, możliwe są też niższe statusy „zamówiona” i „zarezerwowana” – u nas nie używane). Wyniki grupowane są po id\_book, więc max zwracany jest dla danej książki a nie dla całego zbioru.

CREATE

VIEW `book\_status` AS

SELECT

`b`.`title` AS `title`,

MAX(`l`.`status\_id`) AS `status`,

`s`.`sts\_name` AS `sts\_name`,

`b`.`id\_book` AS `id\_book`

FROM

((`book` `b`

JOIN `loan\_list` `l`)

JOIN `loan\_status` `s`)

WHERE

((`b`.`id\_book` = `l`.`book\_id`)

AND (`l`.`status\_id` < 4)

AND (`l`.`status\_id` = `s`.`sts\_number`))

GROUP BY `b`.`id\_book`

## book\_status\_list

Zwraca listę wszystkich książek wraz z ich statusami.

Jeżeli dany id\_book znajduje się w widoku book\_status to zwraca status z tamtej tabeli. W przeciwnym wypadku podaje status jako „dostępna”. W użyciu left join book z book\_status, więc wszystkie rekordy z book wejdą do wyników, niezależnie od tego, czy znajdą się w dołączanym widoku czy nie.

CREATE

VIEW `book\_status\_list` AS

SELECT

`b`.`title` AS `title`,

`b`.`id\_book` AS `id\_book`,

(CASE

WHEN

`b`.`id\_book` IN (SELECT

`book\_status`.`id\_book`

FROM

`book\_status`)

THEN

`s`.`sts\_name`

ELSE 'dostepna'

END) AS `status`

FROM

(`book` `b`

LEFT JOIN `book\_status` `s` ON ((`b`.`id\_book` = `s`.`id\_book`)))

## bookauthors

Tworzy „przyjemniejszą” wersję autorów – scala imię i nazwisko do jednego pola.

Pobiera dane z tabeki book\_authors, a więc tej która do danej książki z tabeli book przypisuje dowolną ilość autorów z tabeli author. Wyświetla id\_book oraz pole „autor” które jest wynikiem operacji: jeśli imię jest niepuste, podaje „nazwisko, imię”. Jeśli imię jest puste, podaje tylko „nazwisko” bez przecinka i bez imienia. Takie połączone pole wchodzi po dalszych obróbkach do widoku „wypożyczalnia2” z którego pobieramy dane do naszego katalogu w programie.

CREATE

VIEW `bookauthors` AS

SELECT

(CASE

WHEN (`a`.`fname` <> '') THEN CONCAT(`a`.`sname`, ', ', `a`.`fname`)

ELSE `a`.`sname`

END) AS `autor`,

`b`.`id\_book` AS `id\_book`

FROM

(`book\_authors` `b`

JOIN `author` `a`)

WHERE

(`b`.`id\_aut` = `a`.`id\_aut`)

## ConsAuthor

Tworzy pole, które zawiera połączonych autorów z kilku rekordów.

Jeżeli do książki jest przypisany więcej niż jeden autor, funkcja GROUP\_CONCAT połączy wszystkich autorów z widoku bookauthors (a więc tych już scalonych, jako pole „nazwisko, imię”) i poszczególne wystąpienia oddziela średnikiem. Grupowanie według id\_book, a więc połączeni zostaną tylko ci autorzy, którzy występują przy danej książce.

CREATE

VIEW `ConsAuthor` AS

SELECT

GROUP\_CONCAT(`bookauthors`.`autor`

SEPARATOR '; ') AS `autorzy`,

`bookauthors`.`id\_book` AS `id\_book`

FROM

`bookauthors`

GROUP BY `bookauthors`.`id\_book`

## loan\_list

Lista wszystkich wypożyczeń z ich aktualnym statusem.

Zbiera dane z poszczególnych tabel i łączy je w całość – id wypożyczenia (loan\_id z tabeli loan), tytuł książki i book\_id z tabeli book, imię i nazwisko użytkownika z tabeli users, status\_id, status name i data ostatniego statusu.

W klauzuli where wykonywane jest dodatkowe zapytanie – z tabeli loan\_history, a więc tam gdzie rejestrowane są poszczególne etapy danego wypożyczenia i ich daty, wybierane jest loan\_id i najwyższy przypisany do niego status. Do widoku wejdą tylko te rekordy gdzie loan\_id i status\_id znajdą się w wyniku tego wyszukiwania – odfiltrujemy występowanie wszystkich niższych statusów, czyli jeśli książka została już zwrócona, to nie pojawi się nam rekord dla jej zamówienia czy wypożyczenia.

CREATE

VIEW `loan\_list` AS

SELECT

`l`.`loan\_id` AS `loan\_id`,

`b`.`title` AS `title`,

`b`.`id\_book` AS `book\_id`,

`u`.`imie` AS `fname`,

`u`.`nazwisko` AS `sname`,

`h`.`status\_id` AS `status\_id`,

`s`.`sts\_name` AS `sts\_name`,

`h`.`sts\_date` AS `sts\_date`

FROM

((((`loan` `l`

JOIN `book` `b`)

JOIN `users` `u`)

JOIN `loan\_history` `h`)

JOIN `loan\_status` `s`)

WHERE

((`h`.`loan\_id` , `h`.`status\_id`) IN (SELECT

`loan\_history`.`loan\_id`, MAX(`loan\_history`.`status\_id`)

FROM

`loan\_history`

GROUP BY `loan\_history`.`loan\_id`)

AND (`l`.`book\_id` = `b`.`id\_book`)

AND (`l`.`user\_id` = `u`.`user\_id`)

AND (`l`.`loan\_id` = `h`.`loan\_id`)

AND (`h`.`status\_id` = `s`.`sts\_number`))

GROUP BY `l`.`loan\_id`

## loan\_list\_all

Lista wszystkich wypożyczeń z informacją czy jest nadal otwarte (czyli status inny niż zwrócona/anulowane) i połączonymi danymi z kilku tabel.

Podobnie jak poprzednia formuła, łączy dane o książce, użytkowniku i statusie. Status danego wypożyczenia jest wybierany dokładnie tak samo jak w poprzedniej formule, dodane są jeszcze pola login i user\_id na potrzeby naszego programu. Nowością jest pole open. W zależności od statusu przypisuje wartość open (status < 4) lub closed (status >=4).

CREATE

VIEW `loan\_list\_all` AS

SELECT

`l`.`loan\_id` AS `loan\_id`,

`b`.`title` AS `title`,

`b`.`id\_book` AS `book\_id`,

`u`.`imie` AS `fname`,

`u`.`nazwisko` AS `sname`,

`u`.`login` AS `user`,

`u`.`user\_id` AS `user\_id`,

`h`.`status\_id` AS `status\_id`,

`s`.`sts\_name` AS `sts\_name`,

`h`.`sts\_date` AS `sts\_date`,

(CASE

WHEN (`h`.`status\_id` >= 4) THEN 'CLOSED'

WHEN (`h`.`status\_id` < 4) THEN 'OPEN'

END) AS `open`

FROM

((((`loan` `l`

JOIN `book` `b`)

JOIN `users` `u`)

JOIN `loan\_history` `h`)

JOIN `loan\_status` `s`)

WHERE

((`h`.`loan\_id` , `h`.`status\_id`) IN (SELECT

`loan\_history`.`loan\_id`, MAX(`loan\_history`.`status\_id`)

FROM

`loan\_history`

GROUP BY `loan\_history`.`loan\_id`)

AND (`l`.`book\_id` = `b`.`id\_book`)

AND (`l`.`user\_id` = `u`.`user\_id`)

AND (`l`.`loan\_id` = `h`.`loan\_id`)

AND (`h`.`status\_id` = `s`.`sts\_number`))

GROUP BY `l`.`loan\_id`

## loan\_list\_open

Pokazuje tylko aktywne wypożyczenia.

Korzysta z poprzedniego widoku, loan\_list\_all. Pobiera z niego wszystkie dane identyfikacyjne użytkownika, książki i wypożyczenia, ale zafiltrowuje na rekordy ze statusem „OPEN”.

! Wykorzystywany w programie w klasie WindowAccount w tabeli „Aktualne wypożyczenia”

CREATE

VIEW `loan\_list\_open` AS

SELECT

`loan\_list\_all`.`loan\_id` AS `loan\_id`,

`loan\_list\_all`.`title` AS `tytul`,

`loan\_list\_all`.`book\_id` AS `book\_id`,

`loan\_list\_all`.`fname` AS `imie`,

`loan\_list\_all`.`sname` AS `nazwisko`,

`loan\_list\_all`.`user` AS `user`,

`loan\_list\_all`.`user\_id` AS `user\_id`,

`loan\_list\_all`.`status\_id` AS `status\_id`,

`loan\_list\_all`.`sts\_name` AS `status`,

`loan\_list\_all`.`sts\_date` AS `data`

FROM

`loan\_list\_all`

WHERE

(`loan\_list\_all`.`open` = 'OPEN')

## loans\_history\_list

Wyświetla listę wszystkich akcji – zamówień, rezerwacji, wypożyczeń, zwrotów.

Łączy dane z kilku tabel i pokazuje je w bardziej przystępnej wersji. Żadne dane nie są odfiltrowywane, pokazuje wszystkie statusy, te aktualne i te zakończone. Wyświetla dane „robocze” – identyfikator wypożyczenia, książki, wpisu w tabeli loan\_history, jak i te bardziej ludzkie, czyli tytuł, imię i nazwisko użytkownika, nazwę statusu i jego datę.

CREATE

VIEW `loans\_history\_list` AS

SELECT

`h`.`entry\_id` AS `entry\_id`,

`l`.`loan\_id` AS `loan\_id`,

`b`.`id\_book` AS `book\_id`,

`b`.`title` AS `title`,

`u`.`imie` AS `fname`,

`u`.`nazwisko` AS `sname`,

`s`.`sts\_name` AS `sts\_name`,

`s`.`sts\_number` AS `sts\_number`,

`h`.`sts\_date` AS `sts\_date`,

`l`.`user\_id` AS `user\_id`,

`u`.`login` AS `login`

FROM

((((`loan` `l`

JOIN `loan\_history` `h`)

JOIN `loan\_status` `s`)

JOIN `users` `u`)

JOIN `book` `b`)

WHERE

((`l`.`loan\_id` = `h`.`loan\_id`)

AND (`l`.`book\_id` = `b`.`id\_book`)

AND (`l`.`user\_id` = `u`.`user\_id`)

AND (`h`.`status\_id` = `s`.`sts\_number`))

## Historia

Ograniczona wersja widoku loans\_history\_list.

Pokazuje wybrane kolumny i odfiltrowuje roboczy status 0 – „zainicjowana” (używany w procedurze wypożyczenie, bo nie bardzo miałam pomysł jak inaczej ją zbudować)

! Wykorzystywany w programie w klasie WindowAccount w tabeli „Historia wypożyczeń”

CREATE

VIEW `historia` AS

SELECT

`loans\_history\_list`.`title` AS `tytul`,

`loans\_history\_list`.`sts\_name` AS `status`,

`loans\_history\_list`.`sts\_date` AS `data`,

`loans\_history\_list`.`login` AS `user`,

`loans\_history\_list`.`user\_id` AS `user\_id`

FROM

`loans\_history\_list`

WHERE

(`loans\_history\_list`.`sts\_number` <> 0)

## wypozyczalnia2

Widok stworzony do tabeli „Katalog” w naszym programie.

Wyświetla wszystkie te dane, które są nam przydatne i odfiltrowuje te, których nie chciałabym prezentować profesorowi na ekranie. Pobiera id książki, tytuł, autorów z widoki ConsAuthor (czyli w jednym polu wyświetleni są wszyscy autorzy danej książki, z imienia i nazwiska, oddzieleni średnikami), rok wydania, isbn, nazwę wydawcy i kategorii (w formie nazwy a nie ID), status książki (dostępna/wypożyczona).

Z widokiem ConsAuth założony jest left join, żeby mieć pewność, że książki bez autorów nie zostaną pominięte. Wydawca i kategoria powinny być zawsze wypełnione, ale jeśli będziemy chcieli zostawić im opcję pustych pól, to zmienię to też na left join.

Kolumna cenzura domyślnie jest ustawiona na 0, kilka książek zaznaczyłam 1, bo niezależnie od tego jak posortowałam dane, ciągle na stronie startowej pojawiały się jakieś nieładne tytuły, ale nie chciałabym ich na stałe wyrzucać z tabeli.

CREATE

VIEW `wypozyczalnia2` AS

SELECT

`b`.`id\_book` AS `book\_id`,

`b`.`title` AS `tytul`,

`ca`.`autorzy` AS `autorzy`,

`b`.`pub\_year` AS `rok`,

`b`.`isbn` AS `isbn`,

`p`.`name` AS `wydawca`,

`c`.`name` AS `kategoria`,

`bs`.`status` AS `status`

FROM

((((`book` `b`

LEFT JOIN `ConsAuthor` `ca` ON ((`ca`.`id\_book` = `b`.`id\_book`)))

JOIN `publisher` `p`)

JOIN `category` `c`)

JOIN `book\_status\_list` `bs`)

WHERE

((`bs`.`id\_book` = `b`.`id\_book`)

AND (`b`.`id\_pub` = `p`.`id\_pub`)

AND (`b`.`id\_cat` = `c`.`id\_cat`)

AND (`b`.`cenzura` = '0'))

# Procedury

…opisy coming soon

## add\_new\_loan

CREATE DEFINER=`sql11171543`@`%` PROCEDURE `add\_new\_loan`(IN bookid int, IN userid int)

BEGIN

IF

(select loan\_id from (select l.loan\_id, max(h.status\_id) as maxsts from loan l, loan\_history h where l.book\_id=bookid and l.user\_id=userid and l.loan\_id=h.loan\_id group by l.loan\_id) as tmp where tmp.maxsts<4) IS NULL

THEN

BEGIN

INSERT INTO loan (book\_id, user\_id) VALUES (bookid, userid);

INSERT INTO loan\_history (loan\_id, status\_id, sts\_date) values ((select l.loan\_id from loan l where book\_id=bookid and user\_id=userid and l.loan\_id NOT IN (select h.loan\_id from loan\_history h)),0, current\_date);

END;

END IF;

END

## Borrow

CREATE DEFINER=`sql11171543`@`%` PROCEDURE `borrow`(IN bookid int, IN userid int, IN stsid int)

BEGIN

call add\_new\_loan(bookid, userid);

-- SET @max\_status\_all\_loans = (select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (@all\_loans\_user\_book) group by loan\_id) as tmp where tmp.maxstat <4);

SET @status\_still\_open =(select max(h.status\_id) from loan\_history h where h.loan\_id IN (select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4) group by loan\_id);

SET @currentloan=(select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4);

IF stsid > @status\_still\_open THEN

INSERT INTO loan\_history (loan\_id, status\_id, sts\_date) VALUES (@currentloan,stsid, current\_date);

IF (stsid)>=4 THEN

SET @reserved=(select loan\_id from loan\_list\_open where book\_id=bookid and status\_id=1 order by sts\_date asc limit 1);

insert into loan\_history (loan\_id, status\_id, sts\_date) values(@reserved,2,current\_date);

END IF;

END IF;

END

## retrn

CREATE DEFINER=`sql11171543`@`%` PROCEDURE `retrn`(IN bookid int, IN userid int, IN stsid int)

IF (stsid>=4) THEN

BEGIN

call add\_new\_loan(bookid, userid);

SET @status\_still\_open =(select max(h.status\_id) from loan\_history h where h.loan\_id IN (select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4) group by loan\_id);

SET @currentloan=(select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4);

IF stsid > @status\_still\_open THEN

INSERT INTO loan\_history (loan\_id, status\_id, sts\_date) VALUES (@currentloan,stsid, current\_date);

END IF;

SET @reserved=(select loan\_id from loan\_list\_open where book\_id=bookid and status\_id=1 order by sts\_date asc limit 1);

insert into loan\_history (loan\_id, status\_id, sts\_date) values(@reserved,2,current\_date);

END;

END IF

## retrn2

CREATE DEFINER=`sql11171543`@`%` PROCEDURE `rtrn2`(IN bookid int, IN userid int)

BEGIN

call add\_new\_loan(bookid, userid);

SET @status\_still\_open =(select max(h.status\_id) from loan\_history h where h.loan\_id IN (select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4) group by loan\_id);

SET @currentloan=(select loan\_id from (select max(h.status\_id) as maxstat, loan\_id from loan\_history h where h.loan\_id IN (SELECT l.loan\_id from loan l where l.book\_id=bookid AND l.user\_id=userid) group by loan\_id) as tmp where tmp.maxstat <4);

IF @status\_still\_open < 4 THEN

INSERT INTO loan\_history (loan\_id, status\_id, sts\_date) VALUES (@currentloan,4, current\_date);

END IF;

SET @reserved=(select loan\_id from loan\_list\_open where book\_id=bookid and status\_id=1 order by sts\_date asc limit 1);

IF @reserved is not null THEN

insert into loan\_history (loan\_id, status\_id, sts\_date) values(@reserved,2,current\_date);

END IF;

END