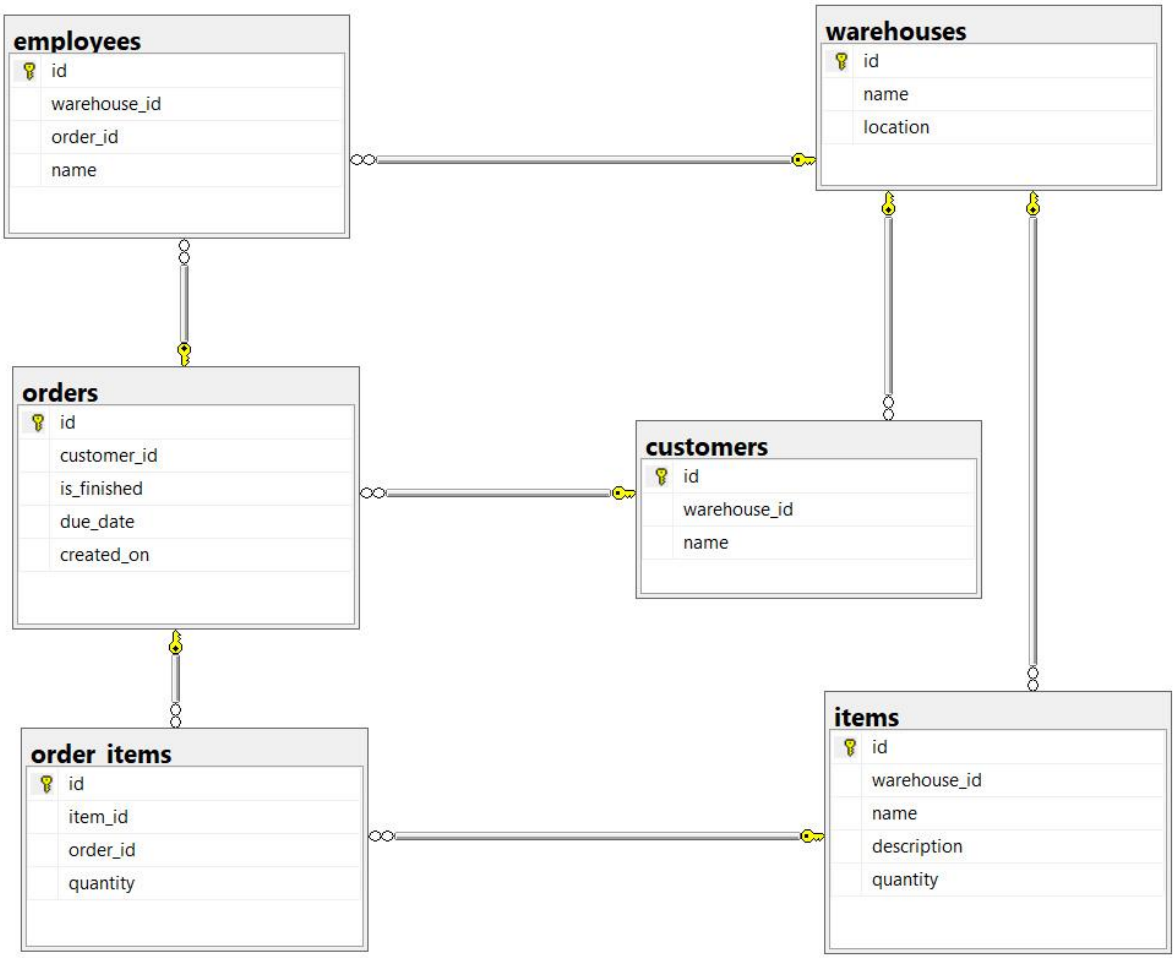
Проек Бази Данни 2020

Георги Демирев, 62296

Склад



Склад (warehouse) има служители(employee), клиенти(customer) и продукти(item). Клиентите имат поръчки(order), които поръчки имат продукти които са в списъка на поръчката(order_item), който продукт(order_item) е продукт от склада(item). От друга страна служителите на склада обработват поръчките и така всяка поръчка има един или повече служители които отговарят за нея.

Самия код има коментари които обясняват ясно какви са създадените Views и Triggers.

Също примерните заявки които са написани са документирани с коментари в sql скриптовете.

Скриптовете:

----------1----------

```
USE master
GO
if exists
(
	select *
	from
		sysdatabases
	where
		name='db'
)
DROP DATABASE db
GO
CREATE DATABASE db
GO
USE db
GO
CREATE TABLE warehouses
	(
		id         int NOT NULL IDENTITY
	  , "name"     char(30) NOT NULL
	  , "location" varchar(100) NOT NULL
	)
;

CREATE TABLE customers
	(
		id           int NOT NULL IDENTITY
	  , warehouse_id int NOT NULL
	  , "name"       char(30) NOT NULL
	)
;

CREATE TABLE "orders"
	(
		id          int NOT NULL IDENTITY
	  , customer_id int NOT NULL
	  , is_finished bit NOT NULL
	  , due_date datetime2(0)
	  , created_on datetime
	)
;

CREATE TABLE items
	(
		id            int NOT NULL IDENTITY
	  , warehouse_id  int NOT NULL
	  , "name"        char(30) NOT NULL
	  , "description" char(255)
	  , quantity      int NOT NULL
```

```
	)
;

CREATE TABLE order_items
	(
		id	int NOT NULL IDENTITY
	  , item_id  int NOT NULL
	  , order_id int NOT NULL
	  , quantity int NOT NULL
	)
;

CREATE TABLE employees
	(
		id		int NOT NULL IDENTITY
	  , warehouse_id int NOT NULL
	  , order_id	int
	  , "name"	char(30) NOT NULL
	  ,
	 )
;

------PRIMARY_KEYS----------------------------------------------------------------------
ALTER TABLE warehouses ADD CONSTRAINT PK_warehouses PRIMARY KEY(id)
;

ALTER TABLE customers ADD CONSTRAINT PK_customers PRIMARY KEY(id)
;

ALTER TABLE "orders" ADD CONSTRAINT PK_orders PRIMARY KEY (id)
;

ALTER TABLE order_items ADD CONSTRAINT PK_order_items PRIMARY KEY(id)
;

ALTER TABLE items ADD CONSTRAINT PK_items PRIMARY KEY(id)
;

ALTER TABLE employees ADD CONSTRAINT PK_employees PRIMARY KEY(id)
;

------FEREIGN_KEYS----------------------------------------------------------------------
--one to many
ALTER TABLE customers ADD CONSTRAINT FK_customers_warehouses FOREIGN KEY(warehouse_id)
REFERENCES warehouses(id)
;

ALTER TABLE "orders" ADD CONSTRAINT FK_orders_customers FOREIGN KEY(customer_id)
REFERENCES customers(id)
;

ALTER TABLE order_items ADD CONSTRAINT FK_order_items_orders FOREIGN KEY(order_id)
REFERENCES "orders"(id)
;

ALTER TABLE items ADD CONSTRAINT FK_items_warehouses FOREIGN KEY(warehouse_id) REFERENCES
warehouses(id)
;
```

```
ALTER TABLE employees ADD CONSTRAINT FK_employees_warehouses FOREIGN KEY(warehouse_id)
REFERENCES warehouses(id)
;

ALTER TABLE employees ADD CONSTRAINT FK_employees_orders FOREIGN KEY(order_id) REFERENCES
"orders"(id)
;

--one to one
ALTER TABLE order_items ADD CONSTRAINT FK_order_items_items FOREIGN KEY(item_id)
REFERENCES items(id)
;
```

----------2----------

```
CREATE TRIGGER dbo.TriggerCustomerUpdate
ON
	orders AFTER
INSERT
  ,
 UPDATE
	AS BEGIN
UPDATE
	dbo.orders
SET created_on = GETDATE()
FROM
	orders o
	JOIN
		inserted i
		ON
			o.id = i.id
WHERE
	i.created_on IS NULL
END
--Trigger to set created_on date for orders, every time when new order is persisted
------TRIGGERS--------------------------------------------------------------------------------
```

----------3----------

```
CREATE VIEW dbo.free_employees AS
SELECT *
FROM
	employees
WHERE
	employees.order_id IS NULL
;

go
CREATE VIEW dbo.ordered_items AS
SELECT
	i.id
  , i.warehouse_id
  , i."name"
  , i."description"
  , p.total_quantity
FROM
```

```
	items AS i
	INNER JOIN
		(
			SELECT
				item_id
			  , SUM(quantity) total_quantity
			FROM
				order_items
			GROUP BY
				item_id
		)
		AS p
		ON
			i.id = p.item_id
;

------VIEWS---------------------------------------------------------------------------
--all free employees
--view for listing all items, that are already booked for orders;
------INDEXES-------------------------------------------------------------------------
--I have not added indexes, because there is not need for this complexity of the database.
--If in future version of the database i have for example a need for second id with
different purpose,
--I will make this column indexable

-------------------------------------------------4-------------------------------------------------

USE db
GO

----Warehouses-------------------------------------------------------------------------
insert into warehouses ("name", "location") values('second warehous','location 2');
insert into warehouses ("name", "location") values('first warehous','location 1');
------Customers-----------------------------------------------------------------------
insert into customers (warehouse_id, "name") values('1','Ivan');
insert into customers (warehouse_id, "name") values('1','Georgi');
insert into customers (warehouse_id, "name") values('1','Ivan');
insert into customers (warehouse_id, "name") values('2','Stoqn');
------Orders--------------------------------------------------------------------------
insert into "orders" (customer_id, is_finished, due_date) values('2','0','2020-09-15
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('2','0','2020-07-15
00:00:00');
insert into "orders" (customer_id, is_finished) values('1','0');
insert into "orders" (customer_id, is_finished, due_date) values('2','1','2020-08-05
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('3','0','2020-08-12
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('1','1','2020-11-15
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('1','1','2020-11-15
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('4','0','2020-08-01
00:00:00');
insert into "orders" (customer_id, is_finished, due_date) values('4','0','2020-12-15
00:00:00');
insert into "orders" (customer_id, is_finished) values('4','1');
```

```
------Items-------------------------------------------------------------------------
insert into items (warehouse_id, "name", "description", quantity) values('1','pc','good
pc','1111');
insert into items (warehouse_id, "name", "description", quantity) values('1','pc','not
good pc','2222');
insert into items (warehouse_id, "name", "description", quantity) values('1','mouse','good
mouse','3333');
insert into items (warehouse_id, "name", "description", quantity) values('1','pc','very
good pc','4444');
insert into items (warehouse_id, "name", "description", quantity) values('1','pc','bad
pc','5555');
insert into items (warehouse_id, "name", "description", quantity) values('1','mouse','good
mouse','23525');
insert into items (warehouse_id, "name", "description", quantity) values('1','mouse','very
good mouse','3525235');
insert into items (warehouse_id, "name", "description", quantity) values('1','mouse','not
good mouse','22266');
insert into items (warehouse_id, "name", "description", quantity) values('2','mouse','good
mouse','6666');
insert into items (warehouse_id, "name", "description", quantity) values('2','pc','not
good pc','7777');


------Order_items-----------------------------------------------------------------------
insert into order_items (item_id, order_id, quantity) values('1','2','200');
insert into order_items (item_id, order_id, quantity) values('2','2','444');
insert into order_items (item_id, order_id, quantity) values('3','2','333');
insert into order_items (item_id, order_id, quantity) values('4','3','200');
insert into order_items (item_id, order_id, quantity) values('5','3','200');
insert into order_items (item_id, order_id, quantity) values('6','2','200');
insert into order_items (item_id, order_id, quantity) values('7','3','200');
insert into order_items (item_id, order_id, quantity) values('1','7','333');
insert into order_items (item_id, order_id, quantity) values('2','7','2700');
insert into order_items (item_id, order_id, quantity) values('3','7','789');


  ------Employees-------------------------------------------------------------------------
insert into employees(warehouse_id, "name") values('1','Gosho');
insert into employees(warehouse_id, "name") values('1','Pesho');
insert into employees(warehouse_id, order_id, "name") values('1','1','Mitko');
insert into employees(warehouse_id, order_id, "name") values('1','1','Gosho');
insert into employees(warehouse_id, order_id, "name") values('1','2','Pesho');
insert into employees(warehouse_id, "name") values('1','Stefan');
insert into employees(warehouse_id, order_id, "name") values('2','2','Stoqn');
insert into employees(warehouse_id, order_id, "name") values('2','3','Ceco');
insert into employees(warehouse_id, order_id, "name") values('2','4','Boiko');
insert into employees(warehouse_id, "name") values('2','Liuben');
```

----------5----------

Това е скрипта със заявките за тестването на функционалността на базата

По-долу са снимки на таблиците отговарящи на заявките тук

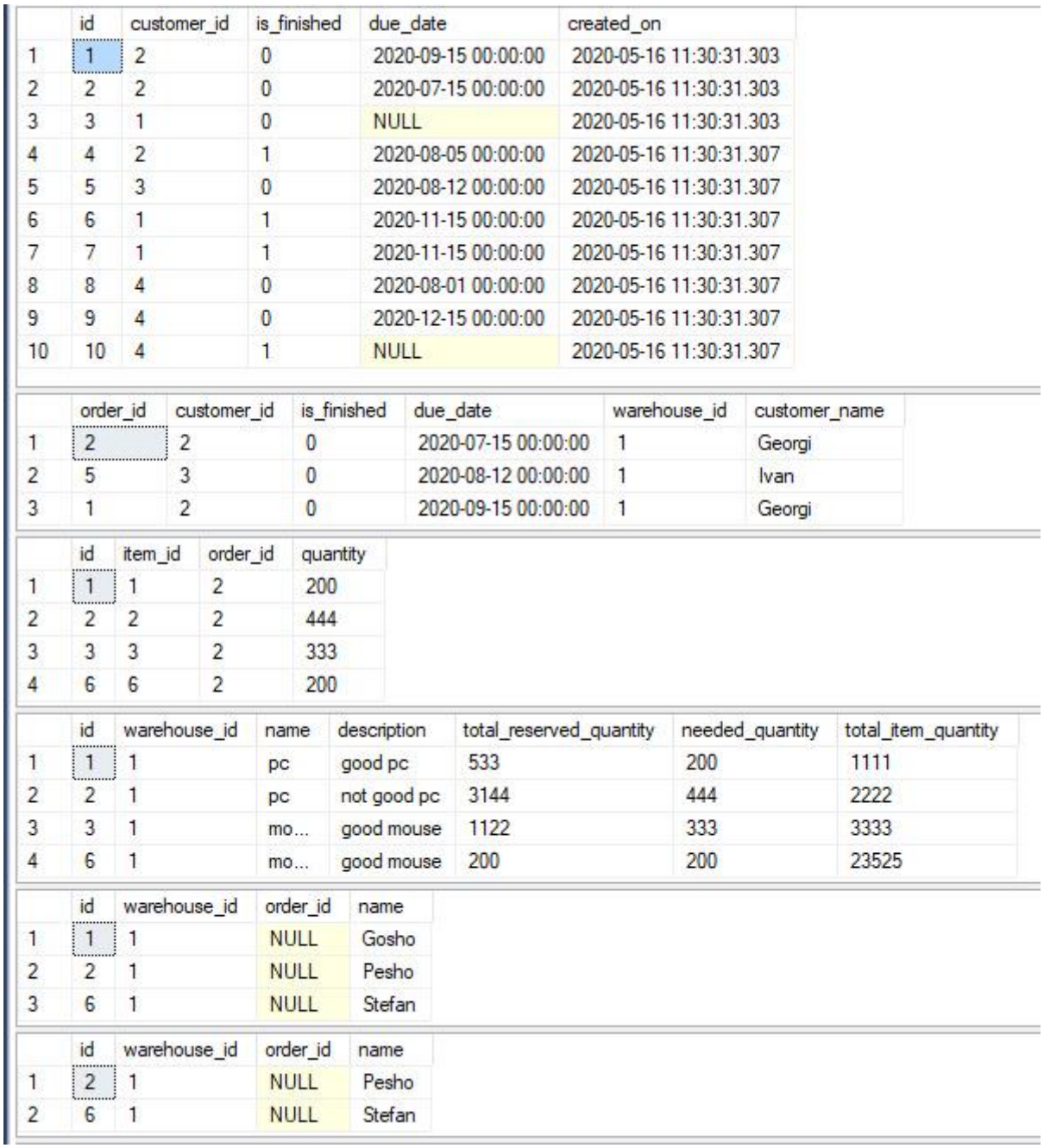
```
USE db
```

```
GO

--All the operations shown bellow are for one warehouse, because the idea is not to mess
warehouses-----------------------------------------------------
--I am performing one usual operation in a wearhouse

--show all the orders to make sure the trigger is working
select * from "orders";
-- here we can see that the milisecond are different so the trigger is working


--show not finished orders ordered by due date in order to start working on that order
select
       orders.id as order_id
  , orders.customer_id
  , orders.is_finished
  , orders.due_date
  , customers.warehouse_id
  , customers."name" as customer_name
from
       orders
       join
              customers
              on
                     orders.customer_id = customers.id
where
       is_finished                = 0
       and due_date     IS NOT NULL
       and customers.warehouse_id = 1
order by
       due_date asc
;


--show the list of items needed for the most urgent order
select *
from
       order_items
where
       order_id = 2
;
-- here order_id = 2 is the most urgent order


--see if there is enough items to fulfil the orders needs
select
       ordered_items.id
  , ordered_items.warehouse_id
  , ordered_items."name"
  , ordered_items."description"
  , ordered_items.total_quantity as total_reserved_quantity
  , p.quantity                   as needed_quantity
  , items.quantity               as total_item_quantity
from
       ordered_items
       join
              (
                     select *
```

```
                from
                        order_items
                where
                        order_id = 2
        )
        as p
        on
                ordered_items.id = p.item_id
    join
        items
        on
                p.item_id = items.id
;
-- here order_id = 2 is the most urgent order
-- and from the difference between total_item_quantity and reserved_quantity we can
-- see if there is enough quantity for the order for every material
-- In this case it is seen that not all needed items are available
-- In future versions of the database-> the warehouse should be able to make orders to
another warehouse to add more items in stock
-- In this version items are added when we see that we need them from this query


--show free employees ordered by name (and then by id)
select *
from
        free_employees
where
        warehouse_id = 1
order by
        "name"
  , id asc
;

--assign an employee to the most urgent order
update
        employees
set order_id = 2
where
        employees.id                = 1
        and employees.warehouse_id = 1
;

-- here order_id = 2 is the most urgent order
--show free employees again to make sure the employee is not available now because he is
working on the order
select *
from
        free_employees
where
        warehouse_id = 1
order by
        "name"
  , id asc
;

--show the employees again to make sure everything is okay and the employee is assigned to
the order
select *
```

```
from
	employees
;

--show employees and their assigned order
select
	employees.id as employee_id
  , employees."name"
  , orders.id as order_id
  , employees.warehouse_id
from
	employees
	join
		orders
		on
			employees.order_id = orders.id
where
	employees.warehouse_id = 1
;
```

Последователността на таблиците отговаря на последователността на заявките.



	id	customer_id	is_finished	due_date	created_on
1	1	2	0	2020-09-15 00:00:00	2020-05-16 11:30:31.303
2	2	2	0	2020-07-15 00:00:00	2020-05-16 11:30:31.303
3	3	1	0	NULL	2020-05-16 11:30:31.303
4	4	2	1	2020-08-05 00:00:00	2020-05-16 11:30:31.307
5	5	3	0	2020-08-12 00:00:00	2020-05-16 11:30:31.307
6	6	1	1	2020-11-15 00:00:00	2020-05-16 11:30:31.307
7	7	1	1	2020-11-15 00:00:00	2020-05-16 11:30:31.307
8	8	4	0	2020-08-01 00:00:00	2020-05-16 11:30:31.307
9	9	4	0	2020-12-15 00:00:00	2020-05-16 11:30:31.307
10	10	4	1	NULL	2020-05-16 11:30:31.307

	order_id	customer_id	is_finished	due_date	warehouse_id	customer_name
1	2	2	0	2020-07-15 00:00:00	1	Georgi
2	5	3	0	2020-08-12 00:00:00	1	Ivan
3	1	2	0	2020-09-15 00:00:00	1	Georgi

	id	item_id	order_id	quantity
1	1	1	2	200
2	2	2	2	444
3	3	3	2	333
4	6	6	2	200

	id	warehouse_id	name	description	total_reserved_quantity	needed_quantity	total_item_quantity
1	1	1	pc	good pc	533	200	1111
2	2	1	pc	not good pc	3144	444	2222
3	3	1	mo...	good mouse	1122	333	3333
4	6	1	mo...	good mouse	200	200	23525

	id	warehouse_id	order_id	name
1	1	1	NULL	Gosho
2	2	1	NULL	Pesho
3	6	1	NULL	Stefan

	id	warehouse_id	order_id	name
1	2	1	NULL	Pesho
2	6	1	NULL	Stefan

	id	warehouse_id	order_id	name
1	1	1	2	Gosho
2	2	1	NULL	Pesho
3	3	1	1	Mitko
4	4	1	1	Gosho
5	5	1	2	Pesho
6	6	1	NULL	Stefan
7	7	2	2	Stoqn
8	8	2	3	Ceco
9	9	2	4	Boiko
10	1...	2	NULL	Liuben

	employee_id	name	order_id	warehouse_id
1	1	Gosho	2	1
2	3	Mitko	1	1
3	4	Gosho	1	1
4	5	Pesho	2	1