Университет ИТМО

Разработка базы данных склада товаров

Курсовая работа

по дисциплине Информационные системы и базы данных

направления 09.03.01 «Информатика и вычислительная техника»

Выполнил:
Неграш Андрей Владимирович (P33301)

Преподаватель:
Харитонова Анастасия Евгеньевна

Санкт-Петербург, 2022

Оглавление

Введение

Автоматизация процессов отчётности и учёта товаров на складе призвана помочь сократить расходы на количестве сотрудников и увеличить прибыль за счёт максимально эффективного использования пространства и более детального прогнозирования загруженности на основе большого количества данных. Для работы такой системы необходимо много составляющих, но одной из важнейших является грамотно спроектированная база данных, на основе которой и будет работать всё остальное. Есть случаи, когда предприятию необходимо лишь перейти в цифровую среду, а все процессы уже отлажены в различного рода журналах и бумажных отчётах, которые большими количествами хранятся в архивах. Чаще всего на таких предприятиях нужно просто в точности перенести существующие процессы в электронную систему. Однако цель нашей работы – рассмотреть ситуацию, когда склад начинает свою работу с самого начала и все процессы нужно продумать заранее для максимального удобства работников.

Актуальность исследования – рассматриваемая тема весьма актуальна, ведь процесс цифровизации бизнеса является одним из важнейших аспектов современного предпринимательства как на территории России, так и за её пределами.

Объектом исследования данной курсовой работы является склада товаров.

Предмет исследования – база данных для нового склада товаров.

Цель работы – разработать базу данных склада товаров.

Анализ предметной области

Для грамотного создания базы данных склада необходимо понимать некоторые его характеристики, а также направленность деятельности. Рассмотрим следующий сценарий: склад принадлежит частному предпринимателю, который сотрудничает лишь с одной крупной сетью продуктовых гипермаркетов и является переходной точкой для товаров на пути от производителя до полок в магазине. Продукция с данного склада поступает в 2 магазина. Для минимизации рисков заключены договоры о перевозке с несколькими индивидуальными предпринимателями, занимающимися частными перевозками, а также с одной крупной транспортной компанией.

План рассматриваемого склада выглядит следующим образом:

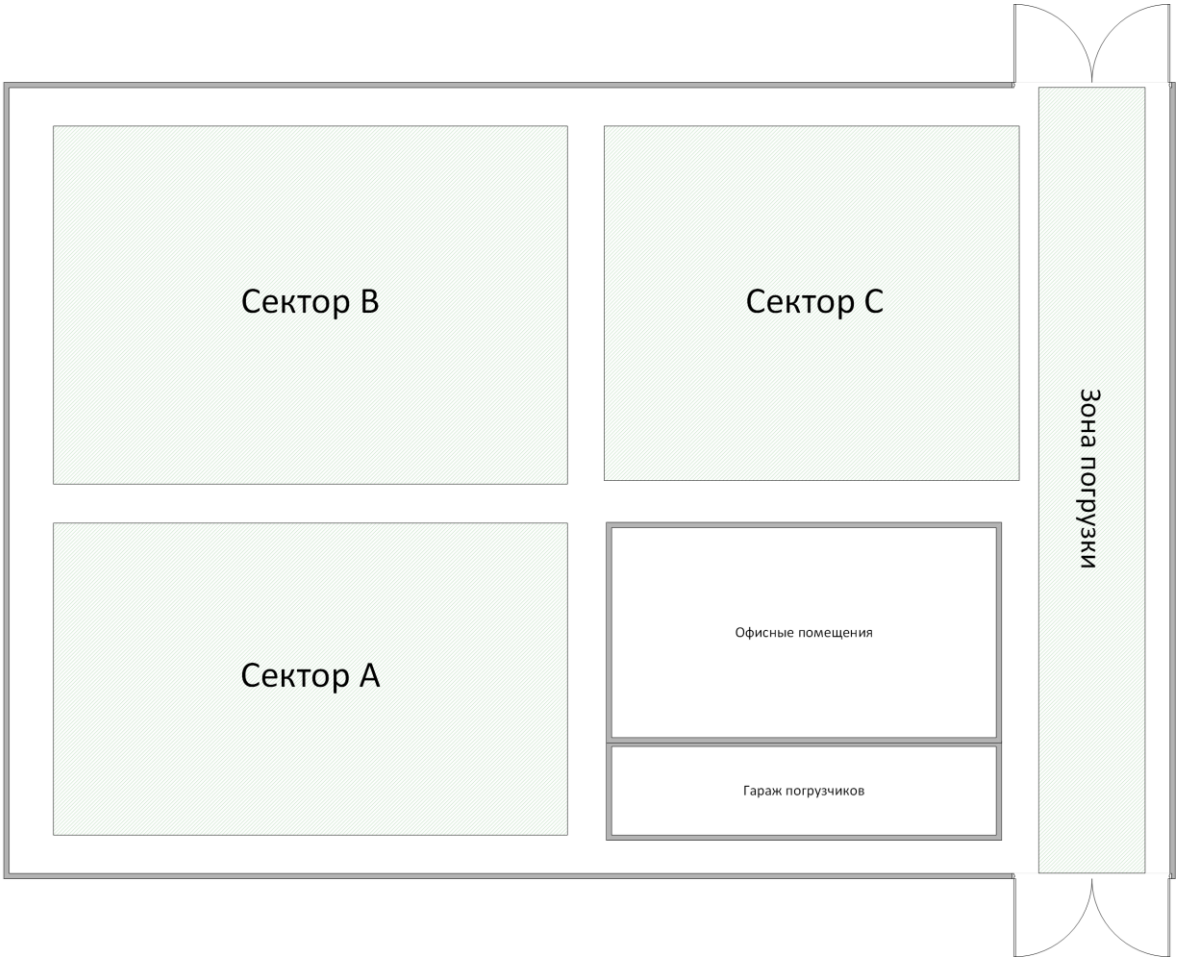

Рисунок 1. План склада

Склад для удобства работников разделён на 3 сектора, каждый из которых может принимать товары не больше определённых размеров. Всего есть 4 категории размеров, для наглядности использована международная система размерности (S, M, L, XL). Сектор A позволяет располагать в себе товары размером S и M, сектор B отведён для долгосрочного хранения (более недели) товаров размером S, M или L, сектор C рассчитан на крупногабаритные товары, маркируемые размером XL. С такими исходными данными приступаем к работе.

Бизнес-процессы

С помощью разрабатываемой базы данных необходимо решать и упрощать выполнение ряда задач, необходимых для нормального функционирования склада. Ниже приведено несколько основных задач, называемых бизнес-процессами, которые позволяет выполнять наша база данных.

Поступление товара на склад

1) Оформление менеджером акта приёма с указанием компании-поставщика и перевозчика
2) Создание транзакции поступления согласно акту
3) Внесение контролирующим работником всех продуктов в заказ данной транзакции (списка поступающих на склад продуктов)
4) Разгрузка в необходимые места хранения
5) Подписание акта приёма перевозчиком и контролирующим работником

Отправка товара в магазин

1) Оформление менеджером акта отправки с указанием конечной компании-получателя (магазина) и перевозчика
2) Создание транзакции отправки согласно акту
3) Внесение контролирующим работником всех продуктов в заказ данной транзакции (списка отправляемых продуктов)
4) Подготовка и перемещение продуктов из мест хранения в место погрузки
5) Погрузка
6) Подписание акта отправки перевозчиком и контролирующим работником

Оформление возврата товара

1) Оформление менеджером акта возврата товара по форме с указанием причин, компании-отправителя (магазина) и перевозчика
2) Создание транзакции по типу «Поступление товара на склад»
3) Внесение контролирующим работником всех продуктов в заказ данной транзакции (списка поступающих на склад продуктов)
4) Разгрузка в необходимые места хранения
5) Подписание акта приёма перевозчиком и контролирующим работником

Инфологическая схема

Подготовим инфологическую схему для будущей базы данных, указав все связи между сущностями. Для создания модели воспользуемся веб-приложением PhpMyAdmin в режиме «Дизайнер».

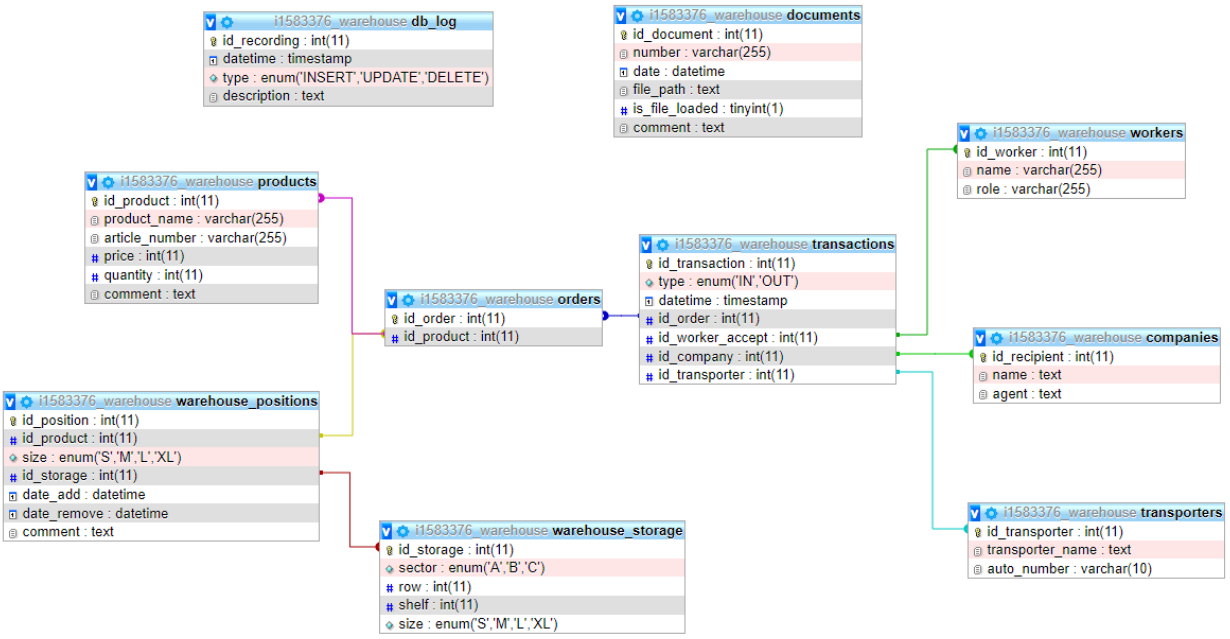

Рисунок 2. Инфологическая схема базы данных

Описание сущностей

Реализовать разрабатываемую систему можно с использованием любой СУБД, в том числе — нереляционной (NoSQL), однако в связи с достаточно небольшим объёмом данных и простотой настройки и внедрения мы будем использовать MySQL.

Таблица companies

Имя поля	Тип данных	Ключ	Комментарий
id_recipient	int	Первичный	id компании-партнёра в базе данных
name	text	-	официальное название компании
agent	text	-	ФИО официального представителя

Таблица содержит первичные данные компаний-партнёров – поставщиков товара и конечные магазины продажи. Доступные данные позволят быстрее составлять акты приёма или отправки, при этом не несут в себе всей информации о компании, которая могла бы быть использована не по назначению.

Таблица db_log

Имя поля	Тип данных	Ключ	Комментарий
id_recording	int	Первичный	id записи
datetime	timestamp	-	дата и время записи
type	enum(‘INSERT’, ’UPDATE’, ’DELETE’)	-	тип произошедшего действия
description	text	-	описание операции

Таблица существует для логирования событий, произошедших в основных таблицах системы. Каждая запись означает произведённое действие над той или иной таблицей данных.

Таблица documents

Имя поля	Тип данных	Ключ	Комментарий
id_document	int	Первичный	id документа в базе данных
number	varchar(255)	-	номер документа согласно внутреннему реестру
date	datetime	-	дата добавления/последней редакции
file_path	text	-	путь к файлу документа (если он загружен в базу, иначе null)
is_file_loaded	tinyint(1)	-	загружен ли файл в базу
comment	text	-	комментарий к документу

Таблица необходима для учёта текущих документов и сохранения электронных копий. К релевантным документам относятся акты приёма и выдачи товара со склада, при этом документы, которые связаны с работой отдела кадров или бухгалтерии в данный реестр попадать не должны.

Таблица orders

Имя поля	Тип данных	Ключ	Комментарий
id_order	int	Первичный	id заказа в базе данных
id_product	int	-	id продукта в базе данных
quantity	int	-	количество продуктов в заказе

В данной таблице находится список продуктов, объединённых одним заказом. Не имеет значения, поступает ли этот заказ на склад или отправляется в магазин.

Таблица products

Имя поля	Тип данных	Ключ	Комментарий
id_product	int	Первичный	id продукта в базе данных
product_name	varchar(255)	-	название продукта
article_number	varchar(255)	-	артикул

price	int	-	цена товара за штуку
quantity	int	-	количество единиц товара
comment	text	-	комментарий к товару

В данной таблице хранятся все основные параметры товаров, которые лежат на складе. Поиск по базе при формировании заказа может осуществляться как по названию товара, так и по его артикулу. Цена необходима для быстрой оценки стоимости в случае, если товар был испорчен по вине работников склада.

Таблица transactions

Имя поля	Тип данных	Ключ	Комментарий
id_transaction	int	Первичный	id операции по приёму/выдаче товара
type	enum('IN', 'OUT')	-	тип операции (IN – приём, OUT – выдача)
datetime	timestamp	-	дата и время операции
id_order	int	-	id заказа со всеми продуктами
id_worker_accept	int	-	id работника, одобрившего операцию
id_company	int	-	id компании отправителя/получателя
id_transporter	int	-	id перевозчика

Таблица содержит данные об операциях приёма и выдачи товара. Операция всегда содержит в себе id заказа, в котором есть перечень всех принимаемых или выдаваемых товаров, также есть информация, какой компании принадлежит данный заказ, какой перевозчик занимается доставкой, и какой работник проверил и подтвердил соответствие перечня товаров в заказе их реальному наличию согласно акту.

Таблица transporters

Имя поля	Тип данных	Ключ	Комментарий
id_transporter	int	Первичный	id перевозчика
transporter_name	text	-	ФИО водителя или название компании-перевозчика
car_number	varchar(10)	-	номер автомобиля

В данной таблице содержится первичная информация о перевозчике – ФИО водителя и государственный номер автомобиля (в формате A001AA 198).

Таблица warehouse_positions

Имя поля	Тип данных	Ключ	Комментарий
id_position	int	Первичный	id местоположения продукта
id_product	int	-	id продукта
size	enum(‘S’, ’M’, ’L’, ’XL’)	-	размер продукта (S – маленький, лежит на полках, XL – большой, занимает целый сектор сам)
id_storage	int	-	id места на складе
date_add	datetime	-	дата добавления
date_remove	datetime	-	дата перемещения
comment	text	-	комментарий

Таблица содержит информацию о текущем местоположении того или иного товара на складе для упрощения поиска при сборке заказа. О каждом товаре известно: размер (одна из 4 категорий), место на складе, дата, когда его туда поместили и дата, когда убрали. Такая структура поможет легко перемещать товары внутри склада не теряя возможность в будущем отследить, когда товар пришёл на склад, когда перенесён с одного места на другое и когда был отправлен со склада.

Таблица warehouse_storage

Имя поля	Тип данных	Ключ	Комментарий
id_storage	int	Первичный	id местоположения продукта
sector	enum(‘A’,’B’,’C’)	-	сектор
row	int	-	номер ряда
shelf	int	-	номер полки
size	enum(‘S’,’M’,’L’,’XL’)	-	размер продукта

В данной таблице содержатся данные о каждом месте хранения товаров. Известны размеры, которые может вмещать в себя это место, а также однозначное местоположение, включая сектор, ряд в нём и конкретную полку.

Таблица workers

Имя поля	Тип данных	Ключ	Комментарий
id_worker	int	Первичный	id работника в системе
name	varchar(255)	-	ФИО работника
role	varchar(255)	-	должность работника

Таблица содержит первичные данные о работниках. При желании на основании роли можно давать определённый уровень доступа к системе.

Создание таблиц на языке MySQL

```
CREATE TABLE `companies` (
  `id_recipient` int(11) NOT NULL,
  `name` text COLLATE utf8_unicode_ci NOT NULL,
  `agent` text COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `db_log` (
  `id_recording` int(11) NOT NULL,
  `datetime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  `type` enum('INSERT','UPDATE','DELETE') COLLATE utf8_unicode_ci NOT
NULL,
  `description` text COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `documents` (
  `id_document` int(11) NOT NULL,
  `number` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `date` datetime NOT NULL,
  `file_path` text COLLATE utf8_unicode_ci NOT NULL,
  `is_file_loaded` tinyint(1) NOT NULL,
  `comment` text COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
CREATE TABLE `orders` (
  `id_order` int(11) NOT NULL,
  `id_product` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `products` (
  `id_product` int(11) NOT NULL,
  `product_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `article_number` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `price` int(11) NOT NULL,
  `quantity` int(11) NOT NULL,
  `comment` text COLLATE utf8_unicode_ci
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `transactions` (
  `id_transaction` int(11) NOT NULL,
  `type` enum('IN','OUT') COLLATE utf8_unicode_ci NOT NULL,
  `datetime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  `id_order` int(11) NOT NULL,
  `id_worker_accept` int(11) NOT NULL,
  `id_company` int(11) NOT NULL,
  `id_transporter` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
CREATE TABLE `transporters` (
  `id_transporter` int(11) NOT NULL,
  `transporter_name` text COLLATE utf8_unicode_ci NOT NULL,
  `auto_number` varchar(10) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `warehouse_positions` (
  `id_position` int(11) NOT NULL,
  `id_product` int(11) NOT NULL,
  `size` enum('S','M','L','XL') COLLATE utf8_unicode_ci NOT NULL,
  `id_storage` int(11) NOT NULL,
  `date_add` datetime NOT NULL,
  `date_remove` datetime NOT NULL,
  `comment` text COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


CREATE TABLE `warehouse_storage` (
  `id_storage` int(11) NOT NULL,
  `sector` enum('A','B','C') COLLATE utf8_unicode_ci NOT NULL,
  `row` int(11) NOT NULL,
  `shelf` int(11) NOT NULL,
  `size` enum('S','M','L','XL') COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
CREATE TABLE `workers` (
  `id_worker` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `role` varchar(255) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


ALTER TABLE `companies`
  ADD PRIMARY KEY (`id_recipient`);


ALTER TABLE `db_log`
  ADD PRIMARY KEY (`id_recording`);


ALTER TABLE `documents`
  ADD PRIMARY KEY (`id_document`);


ALTER TABLE `orders`
  ADD PRIMARY KEY (`id_order`),
  ADD KEY `id_product` (`id_product`);


ALTER TABLE `products`
  ADD PRIMARY KEY (`id_product`);


ALTER TABLE `transactions`
  ADD PRIMARY KEY (`id_transaction`),
  ADD KEY `id_order` (`id_order`) USING BTREE,
  ADD KEY `id_worker_accept` (`id_worker_accept`),
```

```
  ADD KEY `id_transporter` (`id_transporter`),
  ADD KEY `id_company` (`id_company`);


ALTER TABLE `transporters`
  ADD PRIMARY KEY (`id_transporter`);


ALTER TABLE `warehouse_positions`
  ADD PRIMARY KEY (`id_position`),
  ADD KEY `id_storage` (`id_storage`),
  ADD KEY `id_product` (`id_product`);


ALTER TABLE `warehouse_storage`
  ADD PRIMARY KEY (`id_storage`);


ALTER TABLE `workers`
  ADD PRIMARY KEY (`id_worker`);


ALTER TABLE `companies`
  MODIFY `id_recipient` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `documents`
  MODIFY `id_document` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `orders`
  MODIFY `id_order` int(11) NOT NULL AUTO_INCREMENT;
```

```
ALTER TABLE `products`
  MODIFY `id_product` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `transactions`
  MODIFY `id_transaction` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `transporters`
  MODIFY `id_transporter` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `warehouse_positions`
  MODIFY `id_position` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `warehouse_storage`
  MODIFY `id_storage` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `workers`
  MODIFY `id_worker` int(11) NOT NULL AUTO_INCREMENT;


ALTER TABLE `companies`
  ADD CONSTRAINT `companies_ibfk_1` FOREIGN KEY (`id_recipient`)
REFERENCES `transactions` (`id_company`);
```

```
ALTER TABLE `orders`
  ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`id_order`) REFERENCES
`transactions` (`id_order`),
  ADD CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`id_product`)
REFERENCES `warehouse_positions` (`id_product`);


ALTER TABLE `products`
  ADD CONSTRAINT `products_ibfk_1` FOREIGN KEY (`id_product`)
REFERENCES `orders` (`id_product`);


ALTER TABLE `transporters`
  ADD CONSTRAINT `transporters_ibfk_1` FOREIGN KEY (`id_transporter`)
REFERENCES `transactions` (`id_transporter`);


ALTER TABLE `warehouse_storage`
  ADD CONSTRAINT `warehouse_storage_ibfk_1` FOREIGN KEY (`id_storage`)
REFERENCES `warehouse_positions` (`id_storage`);


ALTER TABLE `workers`
  ADD CONSTRAINT `workers_ibfk_1` FOREIGN KEY (`id_worker`)
REFERENCES `transactions` (`id_worker_accept`);


COMMIT;
```

Запросы

Поиск товара по артикулу

```
SELECT * FROM `products` WHERE `article_number` = 'article_number'
```

Поиск товаров по названию с совпадениями

```
SELECT * FROM `products` WHERE `product_name` LIKE '%searching_string%'
```

Вывод списка из 50 документов, отсортированных по дате последнего изменения

```
SELECT * FROM `documents` ORDER BY `date` DESC LIMIT 50
```

Вывод списка товаров, отсортированных по дате появления на складе

```
SELECT
    p.product_name,
    p.article_number,
    p.price
FROM
    warehouse_positions as wp, products as p
WHERE
    wp.id_product = p.id_product
ORDER BY
    wp.date_add DESC
```

Вывод 3 лучших работников по количеству обработанных актов отправки/приёма

```
SELECT
    w.name,
    w.role,
```

```
    (SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept =
w.id_worker AND t.type = 'IN') as count_in_transactions,
    (SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept =
w.id_worker AND t.type = 'OUT') as count_out_transactions,
    (SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept =
w.id_worker) as count_all
FROM
    workers as w
ORDER BY
    count_all DESC LIMIT 3
```

Вывод списка водителей по убыванию количества выполненных заказов

```
SELECT
    t.transporter_name,
    t.auto_number,
    (SELECT COUNT(*) FROM transactions as tr WHERE tr.id_transporter =
t.id_transporter) as count_orders
FROM
    transporters as t
ORDER BY
    count_orders DESC LIMIT 25
```

Вывод 5 заказов, содержащих товар (по его артикулу)

```
SELECT t.*
FROM
    transactions as t,
    orders as o,
    products as p
WHERE
    t.id_order = o.id_order AND
```

```
        o.id_product = p.id_product AND
        p.article_number = 'article_number'
ORDER BY t.datetime DESC LIMIT 5
```

Вывод всех поступлений товара за октябрь 2022 года

```
SELECT *
FROM
        transactions as t
WHERE
        MONTH(t.datetime) = 10 AND
        YEAR(t.datetime) = 2022 AND
        t.type = 'IN'
ORDER BY t.datetime DESC
```

Просмотр товара, лежащего в секторе A на 2 полке в 3 ряду

```
SELECT p.*
FROM
        warehouse_positions as wp,
        products as p,
        warehouse_storage as ws
WHERE
        wp.id_product = p.id_product AND
        ws.sector = 'A' AND
        ws.row = 3 AND
        ws.shelf = 2 AND
        ws.id_storage = wp.id_storage
ORDER BY wp.date_add DESC
```

Вывод всех свободных полок на складе необходимого размера XL

```
SELECT ws.*
FROM warehouse_storage as ws
WHERE
    (SELECT COUNT(ALL wp.date_add)
    FROM warehouse_positions as wp
    WHERE ws.id_storage = wp.id_storage
    ) = (
    SELECT COUNT(ALL wp.date_remove)
    FROM warehouse_positions as wp
    WHERE ws.id_storage = wp.id_storage) AND
    ws.size = 'XL'
ORDER BY
    ws.sector ASC,
    ws.row ASC,
    ws.shelf ASC
```

Глобальный поиск по всей системе по введённой строчке текста

```
SELECT
    c.id_recipient as id,
    c.name as name,
    'COMPANY' as type
FROM companies as c
WHERE c.name LIKE '%searching_string%'
UNION SELECT
    d.id_document as id,
    CONCAT_WS(' ', 'Документ №', d.number) as name,
    'DOCUMENT' as type
```

```
FROM documents as d
WHERE d.number LIKE '%searching_string%'
UNION SELECT
        p.id_product as id,
        p.product_name as name,
        'PRODUCT' as type
FROM products as p
WHERE
        p.product_name LIKE '%searching_string%' OR
        p.article_number LIKE '%searching_string%'
UNION SELECT
        t.id_transporter as id,
        t.transporter_name as name,
        'TRANSPORTER' as type
FROM transporters as t
WHERE
        t.transporter_name LIKE '%searching_string%' OR
        t.auto_number LIKE '%searching_string%'
UNION SELECT
        w.id_worker as id,
        w.name as name,
        'WORKER' as type
FROM workers as w
WHERE w.name LIKE '%searching_string%'
```

Хранимые процедуры

Часто в информационных системах одному простому действию, описанному в бизнес-процессе, соответствует несколько операций. Например, действие «добавить товар в заказ» в рамках информационной системы подразумевает несколько шагов:

1) Проверка наличия товара на складе в нужном количестве
2) Добавление в корзину заказа нового товара
3) Выбрать местоположение данного товара

Эти операции выполняются именно в таком порядке и выполняются всякий раз, когда любой работник добавляет товар в заказ. Чтобы упростить читаемость кода и следовать принципу инкапсулирования, логично объединить все эти операции в единый объект. Именно для этого и существуют хранимые процедуры.

Также хранимые процедуры целесообразно использовать для часто выполняемых запросов, так как весь код компилируется единожды при запуске сервера, а каждый следующий вызов использует уже готовый скомпилированный вид – благодаря этому выполнение запроса ускоряется.

Процедура вывода всех документов

```
CREATE PROCEDURE `AllDocuments`() READS SQL DATA
SELECT * FROM `documents` ORDER BY `date` DESC
```

Процедура вывода свободных мест на складе

```
CREATE PROCEDURE `AllEmptyPlaces`() READS SQL DATA
SELECT ws.* FROM warehouse_storage as ws WHERE
(SELECT COUNT(ALL wp.date_add)
FROM warehouse_positions as wp
WHERE ws.id_storage = wp.id_storage) = (
SELECT COUNT(ALL wp.date_remove)
FROM warehouse_positions as wp
WHERE ws.id_storage = wp.id_storage)
ORDER BY ws.sector ASC, ws.row ASC, ws.shelf ASC
```

Процедура добавления продукта на склад

```
CREATE PROCEDURE `AddProduct`
   @name NVARCHAR(255),
   @article NVARCHAR(255),
   @price INT,
   @quantity INT,
   @comment TEXT
AS
INSERT INTO products (product_name, article_number, price, quantity, comment)
VALUES (@name, @article, @price, @quantity, @comment)
```

Процедура добавления продукта в заказ

```
CREATE PROCEDURE AddProductToOrder
   @article VARCHAR(255),
   @product_id INT OUTPUT,
   @order_id INT
AS
   SELECT @product_id = `id_product` FROM `products` WHERE `article_number`= @article
   INSERT INTO `orders` (id_order, id_product)
   VALUES(@order_id, @ product_id)
```

Процедура вывода списка работников с их показателями за этот месяц

```
CREATE PROCEDURE `AllWorkersWithStats`() READS_SQL_DATA
SELECT w.name, w.role,
(SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept = w.id_worker AND t.type = 'IN' AND MONTH(t.datetime) = MONTH(NOW()) AND YEAR(t.datetime) = YEAR(NOW())) as count_in_transactions,
```

(SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept = w.id_worker AND t.type = 'OUT' AND MONTH(t.datetime) = MONTH(NOW()) AND YEAR(t.datetime) = YEAR(NOW())) as count_out_transactions,

(SELECT COUNT(*) FROM transactions as t WHERE t.id_worker_accept = w.id_worker AND MONTH(t.datetime) = MONTH(NOW()) AND YEAR(t.datetime) = YEAR(NOW())) as count_all

FROM workers as w ORDER BY w.name ASC

Собственные функции

В SQL есть большое количество встроенных функций, например, SUM() – возвращающий сумму элементов или MAX()/MIN(), возвращающий максимальный или минимальный элемент набора соответственно. Однако бывает, что в проекте необходимо часто выводить данные, являющиеся сложными или составными. И для того, чтобы каждый раз не писать внутри запроса сложную формулу, можно создать собственную функцию, которая будет высчитывать нужные данные и возвращать только результат в необходимом виде.

Функция общей стоимости товара по его id

```
CREATE FUNCTION ProductSum(@id INT) RETURNS INT
BEGIN
  DECLARE total INT;
    SELECT total = p.price * p.quantity
    FROM products as p
    WHERE p.id_product = @id;
  RETURN total;
END;
```

Функция общей стоимости всех товаров на складе

```
CREATE FUNCTION AllProductsPrice() RETURNS TABLE
AS RETURN
(SELECT
        p.product_name,
        p.article_number,
        p.price*p.quantity as sum,
        p.comment
FROM products as p)
```

Функция вывода всех записей логирования за текущий месяц

```
CREATE FUNCTION AllLogRecordings() RETURNS TABLE
AS RETURN
(SELECT * FROM `db_log`
WHERE
MONTH(`datetime`) = MONTH(NOW()) AND
YEAR(`datetime`) = YEAR(NOW()))
```

Триггеры

В информационных системах также используется удобный инструмент под названием «Триггер». Этот инструмент позволяет задавать инструкции, которые будут выполнены для операций INSERT, UPDATE или DELETE в выбранной таблице. Данный инструмент очень часто используется для логирования операций над базой данных. К нашей системе это также применимо.

Триггер надбавки цены

```
CREATE TRIGGER `ProductPriceIncrease`
BEFORE INSERT, UPDATE
ON `products`
UPDATE products as p
SET p.price = p.price + 250
WHERE p.id_product = @@IDENTITY
```

Триггер логирования – добавление нового продукта

```
CREATE TRIGGER `log_insert_product`
AFTER INSERT
ON `products`
FOR EACH ROW
INSERT INTO `db_log` (`datetime`, `type`, `description`)
VALUES (NOW(), 'INSERT', 'Новая запись добавлена в таблицу `products`')
```

Заключение

Разработка полноценной электронной системы для бизнеса любого размера – очень трудоёмкий и долгий процесс. Необходимо продумать множество деталей: основные бизнес-процессы, удобство использования, разделение ролей в администрировании системы. В то же время важно не делать большое количество ненужных функций, которые будут лишь создавать дополнительные проблемы и неудобства.

В процессе данной курсовой работы было прописано несколько основных бизнес-процессов для реализации потребностей склада товаров. На их основе создавались инфологическая и физическая модели базы данных, а также был разработан ряд запросов и процедур, необходимых для полноценной работы системы.

Внедрение в реальный проект системы, работающей на базе данных, которая была создана в рамках данной курсовой работы, на данный момент скорее невозможно без доработок, поскольку в качестве объекта исследования был выбран гипотетический новый склад товаров, который только начинает осуществлять свою деятельность. Необходимо проконсультироваться с управляющим складом, выяснить его виденье процессов, основанное на опыте, и подготовить большее количество сущностей с необходимым количеством параметров. Однако созданная база данных может являться надёжной основой для дальнейшей разработки и развития в реальном бизнесе.