

Демонстрационная БД "MARKETPLACE" СУБД РЕД База Данных

Демонстрационная БД **MARKETPLACE** дает пример информационной поддержки цифровой торговой площадки. На этой торговой площадке покупатели ищут и заказывают товары, выставленные на продажу различными продавцами и хранящиеся на отдельных складах. Товары могут иметь произвольное описание с множеством атрибутов. Заказанный и оплаченный товар может быть доставлен в один из пунктов, который указал покупатель. Покупатели могут оставлять отзывы о товарах в заказе и их оценку. Эти оценки и отзывы могут быть использованы при поиске и подборе товаров для заказа другими покупателями.

Прикладные задачи, которые она решает:

- [*] добавление товара на склад
- [*] поиск товара и просмотр деталей описания
- [*] ликвидация товара со склада
- [*] инвентаризация склада / товара
- [*] новый заказ
- [*] оплата заказа
- [*] доставка / завершение заказа
- [*] отмена заказа
- [*] возврат оплаты
- [*] ведение справочной информации
- [*] бизнес-отчеты по товарам и продавцам
- [*] выставление оценок и отзывов

Цель: Демонстрационная база данных предназначена для показа возможностей СУБД RedDatabase 5. Она наглядно демонстрирует основные операции с данными, а также базовые возможности и функции управления базой данных.

База данных может использоваться, например,

- для самостоятельного изучения языка запросов SQL и работы с СУБД RedDatabase,
- для подготовки книг, пособий и учебных курсов по языку SQL и СУБД RedDatabase,
- для демонстрации возможностей RedDatabase в публикациях и онлайн.

Назначение: Основное назначение — предоставить готовую, продуманную среду для экспериментов на примере RedDatabase в приложении e-commerce и для демонстрации основных CRUD операций, работы с JSON, триггерами и процедурами.

Функциональность: База данных демонстрирует современные функции СУБД, включая ограничения, триггеры для автоматизации действий, хранимые процедуры для инкапсуляции логики, работу с JSON для гибкого хранения данных и базовую оптимизацию запросов через индексы.

Демонстрируемые возможности СУБД РЕД База Данных

- [*] целостность БД, включая триггеры для автоматизации действий
- [*] использование разнообразных типов данных
- [*] SQL JSON
- [*] хранимые процедуры для инкапсуляции логики
- [*] базовая оптимизация запросов через индексы
- [*] пакеты

Дополнительные демонстрируемые возможности:

- [*] Начальный набор синтетических данных в таблицах для практического освоения
- [*] Частичный индекс только на актуальные заказы
- [*] Представления с отчётами, отчёты (оборот товаров за год и подобное, оконные функции и т. д.)
- [*] Мониторинг производительности запросов

База данных наполнена синтетическими данными, напоминающими реальные, по которым можно строить разнообразные отчеты и достаточно сложные запросы.

Особенности Демонстрационной БД

Демо-базу можно рассматривать как основу для бэкенда торгового приложения.

Работа с данными

1. **JSON-интерфейсы:** Многие функции принимают JSON для упрощения интеграции:

- CREATE_PRODUCT - создание товара
- CREATE_ORDER - создание заказа
- UPDATE_PRODUCT_INFO_SIMPLE - обновление товара

В настоящее время JSON-данные хранятся в текстовом формате BLOB SUB_TYPE TEXT. Это позволяет хранить произвольные JSON-структуры, но ограничивает возможности выполнения сложных запросов и фильтрации непосредственно на уровне базы данных.

2. **Целостность:** Заложены ограничения для поддержки ссылочной целостности данных и допустимости значений.

3. **Безопасность транзакций**

- Использование `IN AUTONOMOUS TRANSACTION` для критичных операций
- `FOR UPDATE WITH LOCK` для предотвращения race conditions при создании заказов
- Проверки `row_count` для подтверждения операций

4. **Триггеры и функции:** В базе данных присутствуют базовые триггеры для автоматизации некоторых процессов (например, генерация UUID для трек-номера доставки, проверка наличия товара на складе и валидация атрибутов продукта). Также имеются хранимые процедуры и функции для выполнения сложных операций (создание заказа, получение истории заказов, расчета дохода продавца и т. д.).

Администрирование БД

1. **Производительность:** Используется ограниченное количество индексов для оптимизации запросов. Также применяются:

- Частичные индексы для часто запрашиваемых данных
- Вычисляемые поля для часто используемых расчетов
- Представления для сложных запросов

2. **Логи:** Операции возврата отслеживаются автоматически. Логи изменений данных фиксируются вручную через создание таблицы внешних логов.

Демонстрационная база данных распространяется под [лицензией](#).

Демонстрационную БД MARKETPLACE можно скачать на [сайте](#) компании Red Soft. См. раздел [Где взять и как установить](#)

Свои замечания и пожелания направляйте нам по электронной почте по адресу education@red-soft.ru.

Где взять и как установить

Демонстрационную БД MARKETPLACE можно скачать на [сайте](#) компании Red Soft. Поставляемая версия БД MARKETPLACE соответствует СУБД РЕД База Данных версии 5, работающей на компьютере под управлением ОС Windows 10/11 или Linux. Рекомендуется RedOS 8.

Установка демонстрационной БД MARKETPLACE

Скачанный с сайта архив необходимо разархивировать в каталог на диске на рабочей станции или сервере СУБД РЕД База Данных и назначить права доступа к каталогу и файлам. Разархивированный файл нужно перенести в любой каталог на сервере, к которому сервер имеет доступ. После этого демо-база данных готова к соединению.

В Linux пользователь reddatabase должен быть владельцем всех каталогов в пути к файлу БД и иметь соответствующие права доступа. Например:

```
sudo chown reddatabase:redatabase /var/DemoDB
sudo chmod 770 /var/DemoDB
# копируем файл, например
sudo cp MPS_3000.FDB /var/DemoDB/
sudo chmod 660 /var/DemoDB/MPS_3000.FDB
```

После этого можно проверить готовность и полноту демо-базы

```
/opt/RedDatabase/bin/isql -u sysdba
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect localhost/3050:/var/_db/MPS_3000.FDB;
Database: localhost/3050:/var/_db/MPS_3000.FDB, User: SYSDBA
SQL>
```

Демонстрационная БД "MARKETPLACE"

```

~ : isql — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
[+] Новая вкладка  [ ] Разделить окно  [ ] Копировать  [ ] Вставить

1: Io: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
[abc@localhost ~]$ /opt/RedDatabase/bin/isql -u sysdba
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect localhost/3050:/var/_db/MPS_3000.FDB;
Database: localhost/3050:/var/_db/MPS_3000.FDB, User: SYSDBA
SQL> select * from install.get_tables_record_counts;

TABLE_NAME                                RECORD_COUNT  ERROR_INFO
=====
ADDRESSES                                3000
CUSTOMERS                                1492
CUSTOMER_ADDRESSES                        1492
GOODS                                    3000
MEDIA                                    14556
ORDERS                                   30000
ORDER_CANCELLATIONS                       565
ORDER_ITEMS                               349025
ORDER_STATUSES                             6
PAYMENTS                                33225
PAYMENT_STATUSES                           4
PICKUP_MEDIA                              1061
PICKUP_POINTS                             359
PRODUCTS                                 4500
PRODUCTS_MEDIA                           13495
REVIEWS                                235887
SELLERS                                  832
SHIPPING_METHODS                          2
TMPATT                                    0
WAREHOUSES                               317

TABLE_NAME                                RECORD_COUNT  ERROR_INFO
=====
WAREHOUSE_STOCK                          20747

```

В различных дистрибутивах количество записей может отличаться.

Тестовые данные

Хотя демо-база данных "MARKETPLACE" наполнена синтетическими данными, они очень похожи на реальные. По ним можно строить разнообразные отчеты и достаточно сложные запросы. Количество записей в каждом отношении для разных наборов (дистрибутивов) указано в таблице ниже.

ОТНОШЕНИЕ	ОПИСАНИЕ	"3000"	"5000"
REVIEWS	Отзывы покупателей о товарах и покупках	235887	399237
PAYMENTS	Оплаты заказов	33225	55513
SELLERS	Продавцы	832	1432
PICKUP_MEDIA	Фото, схемы и пр., относящиеся к пунктам выдачи	1061	1609
PAYMENT_STATUSES	Статус оплаты	4	4
WAREHOUSE_STOCK	Запасы товаров на складах	20747	34406
ORDER_ITEMS	Строки заказов - товары и их количество	349025	581199
PRODUCTS_MEDIA	Фото, схемы и пр., относящиеся к товарам	13495	22579
WAREHOUSES	Склады	317	472
CUSTOMERS	Покупатели	1492	2545
TMPATT	временная для загрузки	0	0
CUSTOMER_ADDRESSES	Почтовые адреса покупателей	1492	2545
PRODUCTS	Предложения товаров от продавцов	4500	7500
ORDERS	Заказы	30000	50000
SHIPPING_METHODS	Способы доставки	2	2
MEDIA	Мультимедийные материалы	14556	24188
ORDER_STATUSES	Статусы заказа	6	6
PICKUP_POINTS	Пункты выдачи заказов (ПВЗ)	359	551
ADDRESSES	Почтовые адреса	3000	5000
GOODS	Основные характеристики товаров	3000	5000
ORDER_CANCELLATIONS	Записи об отмене заказа	565	923

При желании пользователи могут повторить заполнение БД или создать свои расширенные данные на основе поставляемых скриптов. Из-за ограничений ссылочной целостности важно соблюдать последовательность выполнения загрузки.

Описание данных

На этой торговой площадке *покупатели* ищут и заказывают *товары*, выставленные на продажу различными *продавцами*. Товары могут иметь произвольное описание и множество *атрибутов*, в том числе *изображения* и другие *медиа*.

Товары продавцов хранятся на различных *складах*.

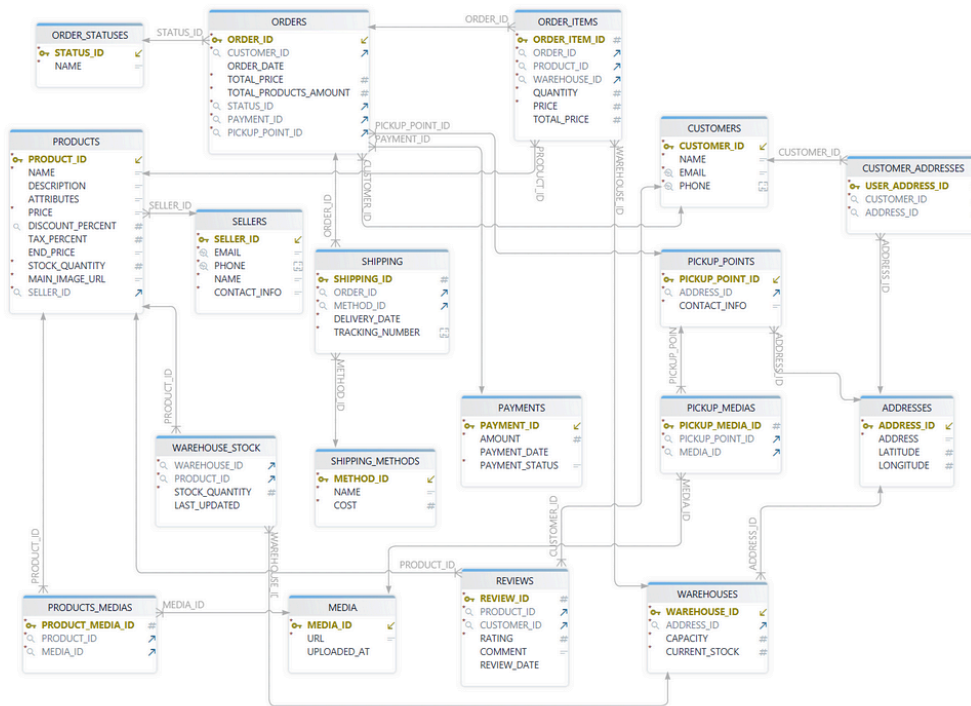
Каждый *заказ* может содержать несколько разных товаров в нужном количестве. Оплаченный заказ может быть доставлен в один из *пунктов выдачи заказов*, который указал *покупатель*. Любой пункт может иметь дополнительное описание, например *изображение* или *маршрут*.

Покупатели могут оставлять *отзывы* о товарах и их *оценку*. Эти оценки и отзывы могут быть использованы при поиске и подборе товаров для заказа другими покупателями.

Список отношений

ОТНОШЕНИЕ	ОПИСАНИЕ
REVIEWS	Отзывы покупателей о товарах и покупках
PAYMENTS	Оплаты заказов
SELLERS	Продавцы
PICKUP_MEDIAS	Фото, схемы и пр., относящееся к пунктам выдачи
PAYMENT_STATUSES	Статус оплаты
WAREHOUSE_STOCK	Запасы товаров на складах
ORDER_ITEMS	Строки заказов - товары и их количество
PRODUCTS_MEDIAS	Фото, схемы и пр., относящееся к товарам
WAREHOUSES	Склады
CUSTOMERS	Покупатели
TMPATT	временная для загрузки
CUSTOMER_ADDRESSES	Почтовые адреса покупателей
PRODUCTS	Предложения товаров от продавцов
ORDERS	Заказы
SHIPPING_METHODS	Способы доставки
MEDIA	Мультимедийные материалы
ORDER_STATUSES	Статусы заказа
PICKUP_POINTS	Пункты получения заказов
ADDRESSES	Почтовые адреса
GOODS	Основные характеристики товаров
ORDER_CANCELLATIONS	Записи об отмене заказа

Диаграмма схемы данных



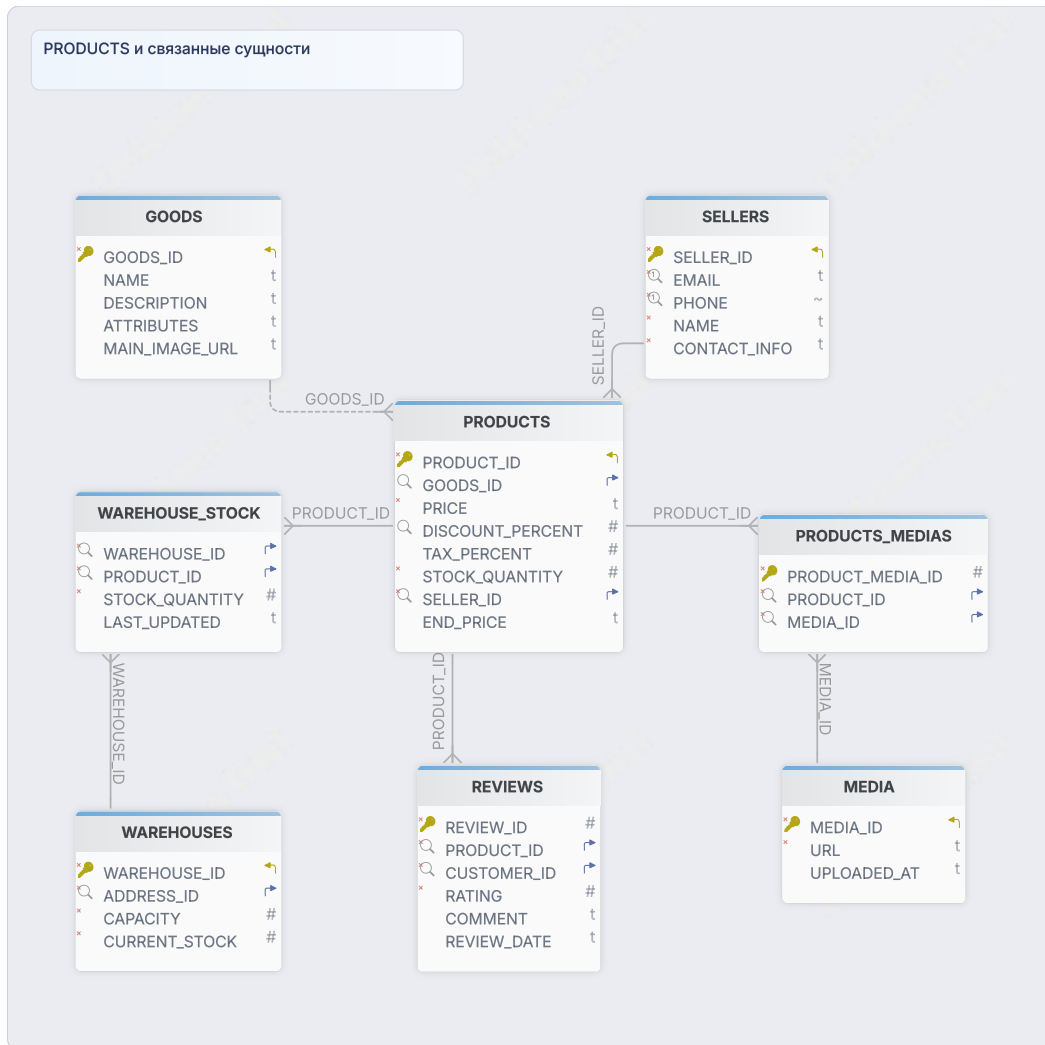
Основными сущностями являются:

- Продукты (PRODUCTS)
- Заказы (ORDERS)
- Покупатели (CUSTOMERS)

Подсистема "Продукты и связанные сущности"

Подсистема ведения товаров (Products и связанные сущности)

Диаграмма



На этой схеме представлена часть БД для хранения информации о продуктах, которыми торгуют на площадке.

Продавец (SELLER) предоставляет свой товар (PRODUCT) для продажи на торговой платформе (MARKETPLACE).

Данные о товарах хранятся в таблице PRODUCTS. В таблице GOODS для каждого товара сохраняются общие характеристики: название, описание и набор произвольных атрибутов (цвет, размер и т.п.). Все атрибуты товара можно запросить в представлении

GOODS_ATTRIBUTES. В таблице PRODUCTS — конкретные предложения товара (цена, скидки и т.п.). Каждому товару и предложению назначаются уникальные идентификаторы в момент добавления в БД.

Предложение товара принадлежит продавцу SELLER. Про продавца сохраняется вся контактная информация. Для товара могут храниться различные медиа-материалы (картинки, чертежи, звуковые описания, рекламные ролики). Ссылки на такие материалы сохраняются в таблице MEDIA и связываются с нужным товаром в таблице PRODUCT_MEDIA.

Запасы продуктов хранятся на складе - WAREHOUSE (один продукт может храниться партиями на нескольких складах). Запись в таблице WAREHOUSE_STOCK соответствует одной партии данного товара на указанном складе в наличествующем количестве (запас на складе). Про каждый склад известен его адрес и максимальная емкость. При добавлении товара на конкретный склад специальным триггером CHECK_WAREHOUSE_CAPACITY проверяется, не превысит ли новое количество максимально допустимое.

В системе предусмотрена возможность оставлять отзывы о товаре в таблице REVIEWS. Покупатели из CUSTOMERS могут добавить отзывы и указать свою оценку этого товара (REVIEW.RATING)

Список товаров, продающихся со значительными скидками (распродажа), можно получить с помощью представления SELLOUT

Вот полный список объектов базы данных, относящихся исключительно к работе с товарами (без заказов и покупателей):

ТАБЛИЦЫ

1. Основные таблицы товаров

1. GOODS - Основные характеристики товаров (название, описание, атрибуты)
2. PRODUCTS - Конкретные предложения товаров от продавцов
3. SELLERS - Продавцы

2. Склады и запасы

4. WAREHOUSE_STOCK - Запасы товаров на конкретных складах
5. WAREHOUSES - Информация о складах

3. Медиафайлы

6. MEDIA - Медиафайлы (изображения, видео)
7. PRODUCTS_MEDIA - Связь товаров с медиафайлами

ПРЕДСТАВЛЕНИЯ (VIEWS)

1. GOODS_ATTRIBUTES - Показ JSON атрибутов товаров
2. SELLOUT - Товары со скидкой $\geq 75\%$
3. TOPCAPS - Анализ загрузки складов (включает товарные остатки)
4. CATEGORY_OF_GOODS - категории товаров (у кого присутствует такой атрибут)

ФУНКЦИИ

Создание и управление товарами

1. `CREATE_PRODUCT` - Создание продукта с JSON-описанием

Информация о товарах

2. `CALCULATE_REVENUE` - Расчет выручки продавца за период (косвенно через товары)
3. `ATTR_GET_STRING` – показывает значение названного атрибута для указанного по ID товара

ХРАНИМЫЕ ПРОЦЕДУРЫ

Управление товарами

1. `ADD_PRODUCT_MEDIA` - Добавление медиафайлов к товару
2. `ADD_PRODUCT_MEDIA_STAT` - Добавление медиафайлов с возвратом статистики
3. `UPDATE_PRODUCT_INFO` - Обновление информации о продукте через JSON
4. `DEL_PRODUCT_FROM_WAREHOUSE` - Удаление товара со склада (основная версия)

Информационные

5. `GET_PRODUCT_INFO` - Получение полной информации о товаре
6. `GET_AVERAGE_PRODUCT_RATING` - Получение средней оценки товара

ТРИГГЕРЫ

1. `CHECK_ATTRIBUTES_VALUES` - Валидация JSON атрибутов при вставке/обновлении в `GOODS`
2. `CHECK_WAREHOUSE_CAPACITY` - Проверка вместимости склада перед добавлением товара
3. `UPDATE_STOCK_TRIGGER` - Обновление общего количества товара на складе

ИСКЛЮЧЕНИЯ (EXCEPTIONS)

1. `EXC_INVALID_QUANTITY` - Недопустимое количество товара
2. `EXC_INVALID_JSON` - Недопустимый формат JSON (используется в работе с атрибутами товаров)
3. `EXC_INVALID_ATTRIBUTES` - Недопустимый список атрибутов товара
4. `EXC_WAREHOUSE_CAPACITY_EXCEEDED` - Недостаточно товаров на складе / превышена вместимость
5. `EXC_ILLEGAL_GOODS_ID` - Товар с указанным ID отсутствует в БД
6. `EXC_INVALID_WAREHOUSE_PRODUCT` - Товар отсутствует на складе
7. `EXC_INSUFFICIENT_STOCK` - Недостаточно товара на складе
8. `EXC_DUPLICATE_PRODUCT_ON_WAREHOUSE` - Товар с этого склада уже заказан
9. `EXC_INVALID_WAREHOUSE` - Недопустимый склад

ИНДЕКСЫ

1. `IDX_DISCOUNTED_PRODUCTS` - Частичный индекс для товаров со скидкой
2. `IDX_POSITIVE_STOCK_QUANTITY` - Частичный индекс для положительных остатков на складе

СВЯЗИ МЕЖДУ ОБЪЕКТАМИ ТОВАРОВ

Основные связи

- `PRODUCTS.GOODS_ID` → `GOODS.GOODS_ID` (FK_PRODUCT_GOODS1)
- `PRODUCTS.SELLER_ID` → `SELLERS.SELLER_ID` (FK_PRODUCT_SELLER)
- `PRODUCTS_MEDIAS.PRODUCT_ID` → `PRODUCTS.PRODUCT_ID` (FK_PRODUCT_MEDIA_PRODUCT)
- `PRODUCTS_MEDIAS.MEDIA_ID` → `MEDIA.MEDIA_ID` (FK_PRODUCT_MEDIA_MEDIA)
- `WAREHOUSE_STOCK.PRODUCT_ID` → `PRODUCTS.PRODUCT_ID` (FK_STOCK_PRODUCT)
- `WAREHOUSE_STOCK.WAREHOUSE_ID` → `WAREHOUSES.WAREHOUSE_ID` (FK_STOCK_WAREHOUSE)

Поток данных

`GOODS` (основные характеристики)

↓

`PRODUCTS` (конкретные предложения) → `PRODUCTS_MEDIA` (фото/видео) → `MEDIA` (файлы)

↓

`WAREHOUSE_STOCK` (наличие на складах) → `WAREHOUSES` (информация о складах)

КЛЮЧЕВЫЕ БИЗНЕС-ПРОЦЕССЫ ДЛЯ ТОВАРОВ

1. **Создание товара** → `CREATE_PRODUCT()` → `GOODS` + `PRODUCTS` + `WAREHOUSE_STOCK`
2. **Добавление изображений** → `ADD_PRODUCT_MEDIA()` → `MEDIA` + `PRODUCTS_MEDIA`
3. **Управление остатками** → `DEL_PRODUCT_FROM_WAREHOUSE()` → `WAREHOUSE_STOCK`
4. **Получение информации** → `GET_PRODUCT_INFO()` ← `PRODUCTS` + `GOODS` + `SELLERS`
5. **Анализ скидок** → Представление `SELLOUT` ← `PRODUCTS` + `GOODS`

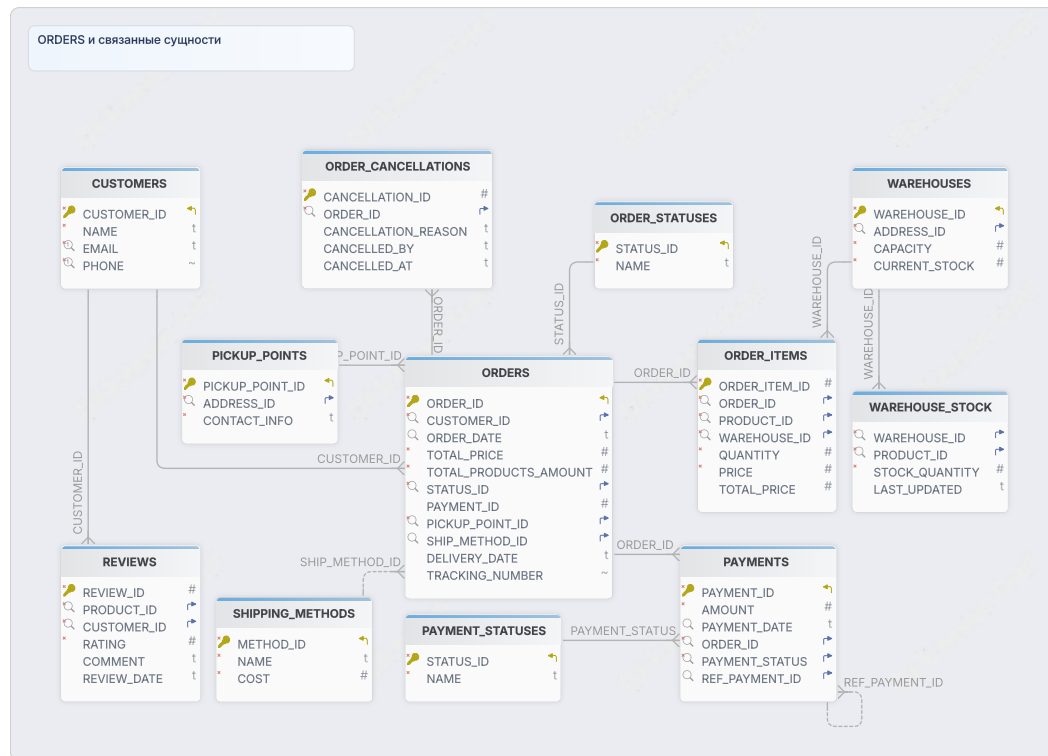
Все эти объекты полностью независимы от заказов и покупателей и могут использоваться для:

- Управления каталогом товаров
- Контроля складских остатков
- Работы с медиаконтентом товаров
- Анализа товарной матрицы
- Управления ценами и скидками

Подсистема "Заказы и связанные сущности"

Подсистема ведения заказов (Orders и связанные сущности)

Диаграмма



На этой схеме отображено хранение информации о заказах, их оплате и доставках.

Покупатель (CUSTOMER) делает заказы (ORDERS) на товары (PRODUCTS). Каждый товар в заказе заносится в таблицу (ORDER_ITEMS) с указанием количества, цены, цены со скидками и склада. Для получения заказа покупатель выбирает удобный ему пункт выдачи заказа (PICKUP_POINTS). После этого заказ переходит в статус "AWAITING_PAYMENT" и ожидает оплаты, а товару для доставки назначается *Tracking Number*, дата доставки и вид доставки (Base | Express).

Оплата заносится в таблицу PAYMENTS, где сохраняется информация обо всех платежах, включая неудачные попытки. После завершения покупателем полной оплаты суммы TOTAL_PRICE, заказ считается оплаченным (PAID).

Для регламентированного создания записи заказа служит процедура CREATE_ORDER, в которую передается вся информация о заказе в формате JSON. Для получения отчета по всем

заказам конкретного покупателя служит процедура GET_CUSTOMER_ORDER_HISTORY. Список всех заказов можно выбрать в представлении ORDER_HISTORY_VIEW.

Полностью оплаченный и доставленный заказ можно ЗАКРЫТЬ (перевести его статус в 'CLOSED'). Имеется возможность настроить автоматическое закрытие всех таких заказов с помощью периодического запуска процедуры CLOSE_PENDING_ORDERS.

Заказ можно отменить (CANCEL). Для оплаченного, но не доставленного заказа при отмене производится возврат оплаты (REFUND).

Для вычисления средней прибыли продавца за интервал дат служит функция CALCULATE_REVENUE

ТАБЛИЦЫ

1. Основные таблицы заказов

1. ORDERS - Основная таблица заказов
2. ORDER_STATUSES - Справочник статусов заказов
3. ORDER_CANCELLATIONS - Отменённые заказы

2. Платежи

4. PAYMENTS - Платежи по заказам
5. PAYMENT_STATUSES - Справочник статусов платежей

3. Доставка и логистика

6. SHIPPING_METHODS - Способы доставки
7. PICKUP_POINTS - Пункты выдачи заказов
8. ADDRESSES - Адреса (используются для пунктов выдачи)
9. PICKUP_MEDIA - Медиафайлы пунктов выдачи

4. Покупатели

10. CUSTOMERS - Покупатели
11. CUSTOMER_ADDRESSES - Адреса покупателей

5. Продавцы

12. SELLERS - Продавцы (для связи с платежами/аналитикой)

ПРЕДСТАВЛЕНИЯ (VIEWS)

1. ORDERS_UNREDEEMED - Неоплаченные заказы с расчётом долга
2. ORDER_HISTORY_VIEW - История заказов
3. SELLERS_INFO - Информация о продавцах (для отчётов по заказам)

ФУНКЦИИ

Управление заказами

1. COUNT_ORDERS_BY_STATUS - Подсчёт заказов по статусу
 2. ORDER_PAID_SUM - Сумма оплат по заказу
 3. IS_PAID - Проверка оплаты заказа
 4. IS_DELIVERED - Проверка доставки заказа
 5. GET_ORDER_PAYMENT_INFO - Информация об оплате заказа
 6. ADD_PAYMENT_GET_ID - Создание платежа для заказа
-

ХРАНИМЫЕ ПРОЦЕДУРЫ

Создание и обработка заказов

1. CREATE_ORDER - Создание заказа из JSON
2. CREATE_ORDER_REVIEW - Создание отзыва о заказе
3. CANCEL_ORDER - Отмена заказа с возвратом товаров
4. CANCEL_ORDER_WITH_REFUND - Отмена с полным возвратом
5. CHECK_ORDER_CANCELLATION_STATUS - Проверка возможности отмены

Управление статусами

6. CLOSE_PENDING_ORDERS - Закрытие ожидающих заказов
7. CLOSE_ORDER_WITH_RESULT - Закрытие конкретного заказа
8. CHECK_ORDER_PAYMENT_STATUS - Проверка статуса оплаты

Платежи и возвраты

9. PAYMENTS_REFUND - Возврат платежей по заказу
10. PAY_FOR_ORDER - Оплата заказа

Управление клиентами

11. CUSTOMER_ADD_OR_UPDATE - Добавление/обновление покупателя
12. CUSTOMER_UPDATE - Обновление данных покупателя
13. CUSTOMER_DROP - Удаление покупателя
14. GET_CUSTOMER_ORDER_HISTORY - История заказов покупателя

Управление продавцами

16. SELLER_ADD_OR_UPDATE - Добавление/обновление продавца
17. SELLER_DROP - Удаление продавца

Управление пунктами выдачи

18. PICKUP_POINT_ADD_OR_UPDATE - Добавление/обновление данных пункта выдачи
 19. PICKUP_POINT_UPDATE - Обновление данных пункта выдачи
 20. PICKUP_POINT_DELETE - Удаление пункта выдачи
-

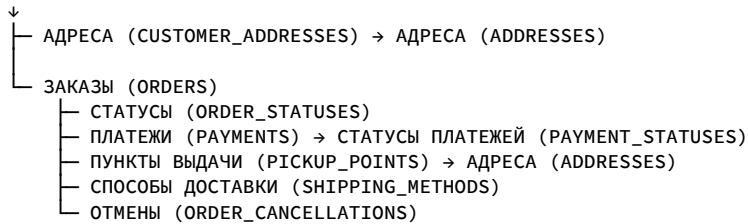
ИСКЛЮЧЕНИЯ (EXCEPTIONS)

1. EXC_INVALID_SHIPPING_METHOD - Недопустимый метод доставки
2. EXC_HAS_DEPENDENCIES - Имеются зависимые записи
3. EXC_INVALID_PICKUP_POINT - Недопустимый пункт самовывоза
4. EXC_NOVALID_ORDER - Заказ не найден

5. EXC_INVALID_CUSTOMER - Недопустимый клиент
 6. EXC_INVALID_SELLER - Недопустимый продавец
 7. EXC_INVALID_QUANTITY - Недопустимое количество
 8. EXC_INVALID_JSON - Недопустимый формат JSON
 9. EXC_INVALID_WAREHOUSE - Недопустимый склад
 10. EXC_INSUFFICIENT_STOCK - Недостаточно товара
 11. EXC_DUPLICATE_PRODUCT_ON_WAREHOUSE - Дублирование товара со склада
-

КЛЮЧЕВЫЕ СВЯЗИ МЕЖДУ ОБЪЕКТАМИ ЗАКАЗОВ

ПОКУПАТЕЛИ (CUSTOMERS)



ОСНОВНЫЕ БИЗНЕС-ПРОЦЕССЫ

1. ЖИЗНЕННЫЙ ЦИКЛ ЗАКАЗА:

CREATE_ORDER() → PAY_FOR_ORDER_0() → CLOSE_ORDER_WITH_RESULT()
 ↓
 CANCEL_ORDER_WITH_REFUND() (при отмене)

2. ПЛАТЕЖНЫЕ ПРОЦЕССЫ:

SET_PAYMENT_ID() → CHECK_ORDER_PAYMENT_STATUS() → PAYMENTS_REFUND()

3. АНАЛИТИКА:

GET_CUSTOMER_ORDER_HISTORY() + ORDER_HISTORY_VIEW + ORDERS_UNREDEEMED

4. УПРАВЛЕНИЕ ПОКУПАТЕЛЯМИ:

CUSTOMER_ADD_OR_UPDATE() → GET_CUSTOMER_ORDER_HISTORY() → CUSTOMER_DROP()

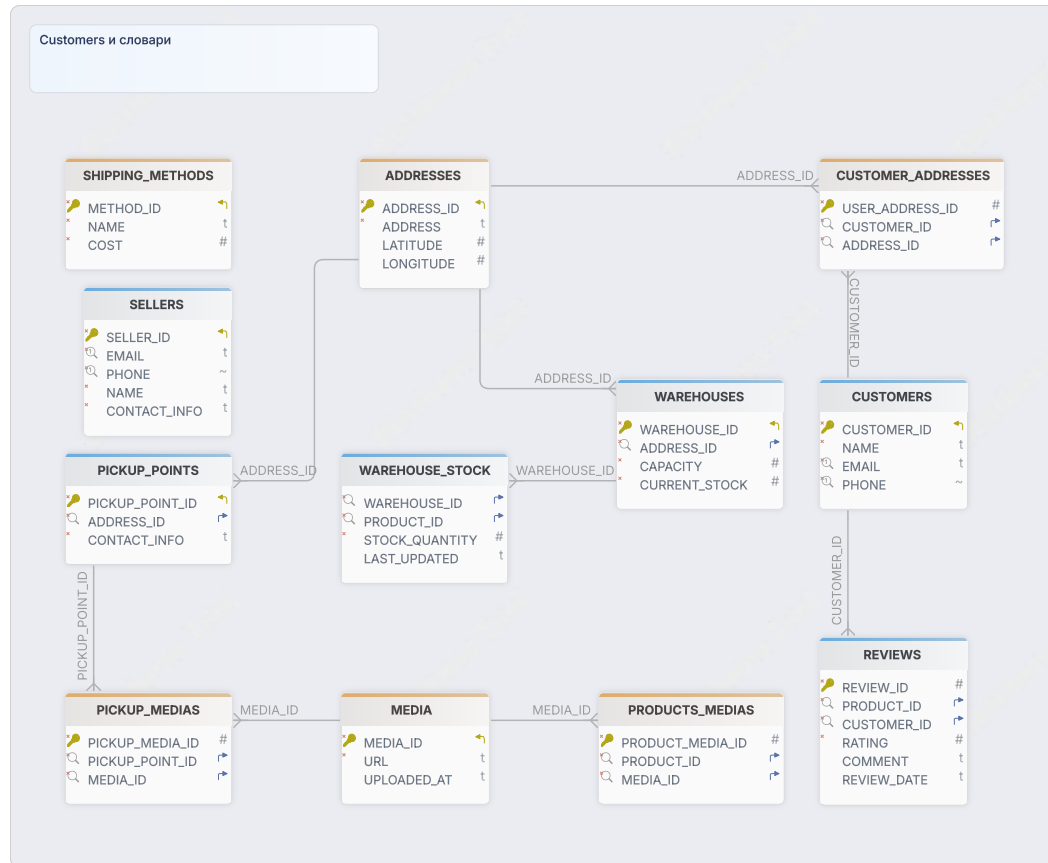
5. УПРАВЛЕНИЕ ПУНКТАМИ ВЫДАЧИ:

PICKUP_POINT_ADD_OR_UPDATE() → PICKUP_POINT_UPDATE() → PICKUP_POINT_DELETE()

Подсистема "Покупатель и другие сущности"

Покупатели и другие сущности

Диаграмма



На этой схеме отображено хранение информации о покупателях и ряд таблиц-справочников.

Покупатель (CUSTOMER) делает заказы на товары. Про покупателя хранится имя, контактная информация (email и номер телефона), а также его адрес (ADDRESS). Допустимо иметь для одного покупателя несколько сохраненных адресов. Их можно использовать для разных заказов. Адреса "привязываются" к покупателю в таблице CUSTOMER_ADDRESSES.

Покупатели могут оставить отзывы (REVIEWS) о товарах из заказа (PRODUCT) и дать им численную оценку (REVIEWS.RATING).

Заказанные товары доставляются на пункты выдачи заказов (PICKUP_POINTS), адреса которых также хранятся в таблице ADDRESSES. Пункты выдачи заказов также могут иметь фото или схемы проезда, которые хранятся в таблице MEDIA и "привязываются" к пункту выдачи в таблице PICKUP_MEDIA.

Медиа-материалы вводятся в систему вместе с данными о других объектах и связываются с ними в момент добавления.

В системе также используется ряд справочников.

Справочники — это таблицы, содержащие относительно статичные данные, используемые для классификации, категоризации или предоставления допустимых значений.

Статусы и классификаторы

1. `ORDER_STATUSES` - Справочник статусов заказов
2. `PAYMENT_STATUSES` - Справочник статусов платежей
3. `SHIPPING_METHODS` - Справочник способов доставки
4. `ADDRESSES` - Справочник адресов
5. `PICKUP_POINTS` - Справочник пунктов выдачи заказов. Статичные точки самовывоза.

Имеют признаки как справочников, так и операционных таблиц:

6. `SELLERS` - Справочник продавцов
7. `GOODS` - Базовый справочник товаров

Некоторые представления можно считать производными справочниками:

7. `CATEGORY_OF_GOODS` – товары и категории (если 'категория' присутствует в списке атрибутов товара)

Эти объекты образуют **конфигурационный слой** системы, который определяет бизнес-правила и допустимые состояния, в то время как операционные таблицы содержат фактические бизнес-данные.

Стандартные операции с данными

Добавление товара на склад (CREATE_PRODUCT)

Товар принадлежит продавцу SELLER. Для товара могут храниться различные медиа-материалы (картинки, чертежи, звуковые описания, рекламные ролики). Ссылки на такие материалы сохраняются в таблице MEDIA и связываются с нужным товаром в таблице PRODUCT_MEDIA.

Запасы товара хранятся на складах (один товар может храниться партиями на нескольких складах - WAREHOUSE). Запись в таблице WAREHOUSE_STOCK соответствует одной партии данного товара на указанном складе в наличествующем количестве (запас на складе).

Регламентное добавление товара на склад выполняется функцией CREATE_PRODUCT. На вход ей нужно передать:

- ID склада, на который будет добавлен товар,
- ID описания товара GOODS_ID, если оно уже имеется в БД,
- полную информацию о партии товара в формате JSON. Например:

```
{
  "NAME": "Один популярный товар для питания.",
  "DESCRIPTION": "Сделан профессионалами для профессионалов. Только положительные отзывы",
  "ATTRIBUTES": { "attributes": [
    { "name": "категория", "value": "Новинки" },
    { "name": "color", "value": "blue" },
    { "name": "stars", "value": 1 }
  ] },
  "PRICE": 12.76,
  "DISCOUNT_PERCENT": 6,
  "TAX_PERCENT": 13,
  "END_PRICE": 13.6532,
  "STOCK_QUANTITY": 1403,
  "MAIN_IMAGE_URL": "http://example.com/img242.jpg",
  "SELLER_ID": 83
}
```

Синтаксический разбор JSON и получение значений параметров выполняются средствами SQL JSON. Выполняются проверки:

- существует ли в SELLERS запись про указанного продавца?
- имеется ли запись в WAREHOUSES про указанный склад?
- Если GOODS_ID указан → проверка существования товара

Негативные ответы приводят к созданию исключения и прекращению дальнейшей обработки.

Если GOODS_ID указан → используем существующий. Иначе — создается новое описание в таблице GOODS. Создается новая запись в таблице PRODUCTS, в поля которой помещаются значения из полей JSON входных данных. Функция возвращает PRODUCT_ID созданной записи. В таблицу WAREHOUSE_STOCK добавляется запись с партией нового товара и указанием его количества.

Каждый склад имеет максимальную емкость. При добавлении товара на конкретный склад специальным триггером CHECK_WAREHOUSE_CAPACITY проверяется, не превысит ли новое количество максимально допустимое.

Обновление информации о товаре может быть выполнено с помощью процедуры UPDATE_PRODUCT_INFO с аналогичными входными параметрами.

Процедура ADD_PRODUCT_MEDIA служит для добавления и "привязки" медиафайлов к товару.

Ликвидация товара на складе (D, U)

Удаление партии товара на складе выполняется хранимой процедурой DEL_PRODUCT_FROM_WAREHOUSE, на вход которой передаются: ID склада, ID товара, удаляемое количество товара. Производится проверка на допустимость значений указанных параметров, в частности ненулевое и положительное количество.

Если на данном складе есть несколько партий этого товара, он будет забираться из самой маленькой подходящей (размер наименьшей партии, который равен или превышает запрошенное количество).

Если такой не нашлось, вызывается исключение, операция прерывается.

Количество товара в найденной партии уменьшается на указанное количество. Запись может быть удалена, если количество = 0

Общее количество товара на складе пересчитывается.

Учет товара (R)

Процедура GET_PRODUCT_INFO выдает полную информацию о товаре, включая данные о продавце.

Создание и добавление заказа (C)

Покупатель (CUSTOMER) делает заказы (ORDERS) на товары (PRODUCTS). Для регламентированного создания записи заказа служит процедура CREATE_ORDER, в которую передается вся информация о заказе в формате JSON. Например:

```

{
  "CUSTOMER_ID": 330,
  "PICKUP_POINT_ID": 1,
  "SHIPPING_METHOD_ID": 1,
  "WAREHOUSES": [
    {
      "WAREHOUSE_ID": 7,
      "PRODUCTS": [
        {"PRODUCT_ID": 36, "QUANTITY": 99},
        {"PRODUCT_ID": 354, "QUANTITY": 82},
        {"PRODUCT_ID": 385, "QUANTITY": 40}
      ]
    },
    {
      "WAREHOUSE_ID": 8,
      "PRODUCTS": [
        {"PRODUCT_ID": 320, "QUANTITY": 89}
      ]
    },
    {
      "WAREHOUSE_ID": 9,
      "PRODUCTS": [
        {"PRODUCT_ID": 169, "QUANTITY": 86}
      ]
    }
  ]
}

```

Данные в формате JSON могут быть получены, например, из ответа REST API, или сформированы на WEB-странице, или в мобильном приложении.

В заказе покупатель указывает удобный ему пункт выдачи (PICKUP_POINTS) и вид доставки (SHIPPING_METHOD). Процедура сначала проверяет:

- правильный ли синтаксис JSON во входном параметре,
- присутствует ли в БД указанный покупатель,
- имеется ли в БД запись про указанный склад,
- находится ли код указанного вида доставки в списке допустимых,
- имеется ли запись про указанный пункт выдачи

При негативном ответе на эти проверки создается исключение, и дальнейшие действия прекращаются. После синтаксического разбора JSON для каждого товара проверяется наличие достаточного запаса на указанном складе.

Если заказ еще не создан, создается запись в таблице ORDERS. Каждый товар из заказа заносится в уникальную запись в таблице (ORDER_ITEMS) с указанием ID заказа, количества, цены и склада. Процедура обновляет количество товара на складе с учетом попавшего в заказ, общее количество товара в заказе и полную сумму заказа.

Процедура создает запись об оплате заказа в таблице PAYMENTS. После этого заказ переходит в статус "AWAITING_PAYMENT" и ожидает оплаты, а товару для доставки назначается *Tracking_Number* в формате GUID, дата доставки и вид доставки (Base | Express).

Код из справочника статуса заказа заносится в поле STATUS_ID. Статус ACTIVE соответствует любому из (AWAITING_PAYMENT, PAID, DELIVERED).

Оплата товаров (U)

Оплата выполняется с помощью вызова хранимой процедуры PAY_FOR_ORDER с параметрами: ORDER_ID_PARAM , PAYMENT_AMOUNT

Оплата заносится в таблицу PAYMENTS, где сохраняется информация обо всех платежах, включая неудачные попытки. После завершения покупателем полной (сумма по всем прошедшим платежам) оплаты, заказ считается оплаченным (PAID). Подтверждение полной оплаты заказа можно получить функцией IS_PAID.

Доставка товаров(U)

В этой демонстрационной БД доставленным считается любой полностью оплаченный заказ, у которого прошло определенное количество дней после указанной даты доставки (по умолчанию - 30).

Проверка того, является ли заказ доставленным (DELIVERED), выполняется функцией IS_DELIVERED.

Закрытие заказа (U)

Полностью оплаченный и доставленный заказ считается полностью выполненным (завершенным) и может быть "закрыт", т.е. переведен в статус CLOSED . Чтобы закрыть заказ, следует вызвать процедуру CLOSE_ORDER_WITH_RESULT и передать ей order_id заказа.

Правила ее работы: После получения подтверждения полной оплаты функцией IS_PAID, значение поля STATUS_ID устанавливается в код, соответствующий 'PAID' , проверяется, доставлен ли заказ (вызов функции IS_DELIVERED), если заказ доставлен, то значение поля STATUS_ID заказа устанавливается в код, соответствующий статусу 'CLOSED' .

Для массового закрытия полностью выполненных заказов служит процедура CLOSE_PENDING_ORDERS - Закрытие ожидающих заказов, которая возвращает статистику своей работы.

Отмена заказа (U)

Уже отмененный (CANCELED) заказ отменить нельзя. Завершенный или доставленный заказ (CLOSED, DELIVERED) отменить нельзя (только через возврат товара, который в демонстрационный пример не входит). Отмена заказа выполняется вызовом процедуры CANCEL_ORDER_WITH_REFUND. Активный заказ переводится в статус 'CANCELED', товары из его строк возвращаются на склад (с разрешением конфликта, если место уже занято). Если отмененный заказ был оплачен, то выполняется *возврат платежей (все оплаты по заказу со статусом 'PAID')*. Запись об отмене заказа заносится в таблицу ORDER_CANCELATIONS.

Возврат платежей (U)

Если отмененный заказ был оплачен, то выполняется *возврат платежей*. Возврат платежей выполняется вызовом процедуры PAYMENT_REFUND.

Возвращается только та сумма, которая была уплачена к моменту возврата (все оплаты по заказу со статусом 'PAID'). Для каждой из таких прошедших оплат добавляются компенсационные записи с отрицательной суммой, которые переводятся в статус 'RFD' (возвращены). Компенсационная запись имеет ссылку на отмененную оплату.

Оставить отзыв и оценить (U)

Отзыв на товар из заказа создается процедурой CREATE_ORDER_REVIEW Покупатель присылает отзыв и указывает: товар, заказ и оценку (0..5)

Полезные запросы для анализа товаров и складов в системе MARKETPLACE:

1. Аналитические запросы по товарам

Товары с наибольшей скидкой

```
CREATE OR ALTER VIEW DISCOUNTED (PRODUCT_ID, "Название товара", "Цена", "Скидка %",
"Конечная цена", "Экономия %", "Продавец", "Наличие")
AS
SELECT
    p.product_id,
    g.name as "Название товара",
    p.price as "Цена",
    p.discount_percent as "Скидка %",
    p.end_price as "Конечная цена",
    ROUND((p.price - p.end_price) / p.price * 100, 2) as "Экономия %",
    s.name as "Продавец",
    ws.stock_quantity as "Наличие"
FROM products p
JOIN goods g ON p.goods_id = g.goods_id
JOIN sellers s ON p.seller_id = s.seller_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
WHERE p.discount_percent > 0
ORDER BY p.discount_percent DESC
;
```

Применение:

```
select * from DISCOUNTED where "Скидка %" > 75 FETCH FIRST 20 ROWS ONLY ;
--или (для скидок больше 75 процентов)
select * from SELLOUT;
```

Самые дорогие товары по категориям (атрибутам)

Такой вариант работает, даже если отсутствует атрибут с именем 'категория'

```

WITH bycat AS (
  SELECT
    COALESCE((
      SELECT
        jt."value"
      FROM JSON_TABLE(g.ATTRIBUTES, 'lax $.attributes[*]'
        COLUMNS (
          "name" VARCHAR(255) PATH '$.name',
          "value" VARCHAR(255) PATH '$.value'
        )) AS jt
      WHERE jt."name" = 'категория'
      ROWS 1
    ), 'nan') AS "Категория",
    g.name AS "Название",
    p.end_price
  FROM goods g
  JOIN products p ON g.goods_id = p.goods_id)
SELECT
  bc."Категория",
  bc."Название",
  MAX(bc.end_price) AS "Максимальная цена",
  MIN(bc.end_price) AS "Минимальная цена",
  AVG(bc.end_price) AS "Средняя цена",
  COUNT(*) AS "Количество товаров"
FROM bycat bc
WHERE bc."Категория" IS NOT NULL
GROUP BY bc."Категория", bc."Название"
ORDER BY "Максимальная цена" DESC

```

Товары с низким запасом (нуждаются в пополнении)

```

CREATE OR ALTER VIEW PRODUCT_INVENTORY
AS
SELECT
  p.product_id,
  g.name as "Товар",
  SUM(ws.stock_quantity) as "Общий остаток",
  COUNT(DISTINCT ws.warehouse_id) as "Количество складов",
  s.name as "Продавец",
  CASE
    WHEN SUM(ws.stock_quantity) <= 10 THEN 'КРИТИЧЕСКИ НИЗКИЙ'
    WHEN SUM(ws.stock_quantity) <= 25 THEN 'НИЗКИЙ'
    WHEN SUM(ws.stock_quantity) <= 50 THEN 'СРЕДНИЙ'
    ELSE 'НОРМАЛЬНЫЙ'
  END as "Уровень запаса"
FROM products p
JOIN goods g ON p.goods_id = g.goods_id
JOIN sellers s ON p.seller_id = s.seller_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
GROUP BY p.product_id, g.name, s.name
HAVING SUM(ws.stock_quantity) <= 50
ORDER BY "Общий остаток" ASC;

```

2. Анализ продавцов

Топ продавцов по количеству товаров

```
CREATE OR ALTER VIEW SELLERS_TOP_BY_PRODUCTS
AS
SELECT
    s.seller_id,
    s.name as "Продавец",
    s.email,
    COUNT(p.product_id) as "Количество товаров",
    SUM(COALESCE(ws.stock_quantity, 0)) as "Общий запас",
    count(distinct COALESCE(ws.WAREHOUSE_ID,0)) as "Количество складов",
    ROUND(AVG(p.end_price),2) as "Средняя цена товара",
    ROUND(MIN(p.end_price),2) as "Минимальная цена",
    ROUND(MAX(p.end_price),2) as "Максимальная цена"
FROM sellers s
LEFT JOIN products p ON s.seller_id = p.seller_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
GROUP BY s.seller_id, s.name, s.email
ORDER BY "Количество товаров" DESC;
```

Продавцы с товарами на конкретном складе

```
CREATE OR ALTER VIEW SELLERS_AT_WAREHOUSE
AS
SELECT
    w.warehouse_id,
    a.address as "Адрес склада",
    s.name as "Продавец",
    COUNT(DISTINCT p.product_id) as "Товаров на складе",
    SUM(ws.stock_quantity) as "Общее количество",
    SUM(ws.stock_quantity * p.end_price) as "Общая стоимость"
FROM warehouses w
JOIN addresses a ON w.address_id = a.address_id
JOIN warehouse_stock ws ON w.warehouse_id = ws.warehouse_id
JOIN products p ON ws.product_id = p.product_id
JOIN sellers s ON p.seller_id = s.seller_id
GROUP BY w.warehouse_id, a.address, s.name
ORDER BY w.warehouse_id, "Общая стоимость" DESC;
```

Вызов:

```
select * from SELLERS_AT_WAREHOUSE where warehouse_id = 3
```

3. Анализ складов

Загрузка складов

```
CREATE OR ALTER VIEW WAREHOUSES_WORKLOAD
AS
SELECT
    w.warehouse_id,
    a.address as "Адрес",
    w.capacity as "Вместимость",
    w.current_stock as "Текущий запас",
    ROUND(w.current_stock * 100.0 / w.capacity, 2) as "Загрузка %",
    COUNT(DISTINCT ws.product_id) as "Уникальных товаров",
    SUM(ws.stock_quantity) as "Всего единиц",
    AVG(ws.stock_quantity) as "Средний запас на товар"
FROM warehouses w
JOIN addresses a ON w.address_id = a.address_id
LEFT JOIN warehouse_stock ws ON w.warehouse_id = ws.warehouse_id
GROUP BY w.warehouse_id, a.address, w.capacity, w.current_stock
ORDER BY "Загрузка %" DESC;
```

Распределение товаров по складам

```
CREATE OR ALTER VIEW GOODS_AT_WAREHOUSES AS
SELECT
    g.name as "Товар",
    SUM(ws.stock_quantity) as "Общее количество",
    COUNT(DISTINCT ws.warehouse_id) as "Количество складов",
    LIST(
        'Склад ' || w.warehouse_id || ': ' || ws.stock_quantity || ' ед.',
        ', '
    ) as "Распределение по складам"
FROM goods g
JOIN products p ON g.goods_id = p.goods_id
JOIN warehouse_stock ws ON p.product_id = ws.product_id
JOIN warehouses w ON ws.warehouse_id = w.warehouse_id
GROUP BY g.name, g.goods_id
HAVING COUNT(DISTINCT ws.warehouse_id) > 1
ORDER BY "Общее количество" DESC;
```

4. Оптимизационные запросы

Товары, которые есть только на одном складе (риск потери доступности)

```
CREATE OR ALTER VIEW SINGLE_WAREHOUSE_GOODS AS
SELECT
    p.product_id,
    g.name as "Товар",
    w.warehouse_id as "Единственный склад",
    ws.stock_quantity as "Количество",
    a.address as "Адрес склада",
    s.name as "Продавец"
FROM products p
JOIN goods g ON p.goods_id = g.goods_id
JOIN warehouse_stock ws ON p.product_id = ws.product_id
JOIN warehouses w ON ws.warehouse_id = w.warehouse_id
JOIN addresses a ON w.address_id = a.address_id
JOIN sellers s ON p.seller_id = s.seller_id
WHERE p.product_id IN (
    SELECT product_id
    FROM warehouse_stock
    GROUP BY product_id
    HAVING COUNT(DISTINCT warehouse_id) = 1
)
ORDER BY ws.stock_quantity ASC;
```

Склады с потенциальной проблемой емкости

```
CREATE OR ALTER VIEW WAREHOUSE_CAPACITY_ISSUES
AS
SELECT
    w.warehouse_id,
    a.address,
    w.capacity,
    w.current_stock,
    ROUND(w.current_stock * 100.0 / w.capacity, 2) as "Текущая загрузка %",
    -- Прогноз загрузки при увеличении запасов на 20%
    ROUND((w.current_stock * 1.2) * 100.0 / w.capacity, 2) as "Прогноз +20%",
    -- Количество товаров с низким запасом на этом складе
    (
        SELECT COUNT(*)
        FROM warehouse_stock ws2
        WHERE ws2.warehouse_id = w.warehouse_id
        AND ws2.stock_quantity < 10
    ) as "Товаров с низким запасом"
FROM warehouses w
JOIN addresses a ON w.address_id = a.address_id
WHERE w.current_stock > w.capacity * 0.8 -- Более 80% загружено
ORDER BY "Текущая загрузка %" DESC;
```

5. Финансовые запросы

Стоимость товаров на каждом складе

```
CREATE OR ALTER VIEW WAREHOUSE_GOODS_COST AS
SELECT
    w.warehouse_id,
    a.address as "Адрес склада",
    COUNT(DISTINCT ws.product_id) as "Количество товаров",
    SUM(ws.stock_quantity) as "Всего единиц",
    SUM(ws.stock_quantity * p.end_price) as "Общая стоимость",
    ROUND(AVG(p.end_price), 2) as "Средняя цена за единицу",
    MAX(p.end_price) as "Самый дорогой товар",
    MIN(p.end_price) as "Самый дешевый товар"
FROM warehouses w
JOIN addresses a ON w.address_id = a.address_id
JOIN warehouse_stock ws ON w.warehouse_id = ws.warehouse_id
JOIN products p ON ws.product_id = p.product_id
GROUP BY w.warehouse_id, a.address
ORDER BY "Общая стоимость" DESC;
```

Анализ маржи по товарам

```
CREATE OR ALTER VIEW PRODUCT_MARGINS
AS
SELECT
    g.name as "Товар",
    p.price as "Цена продажи",
    -- Предположим, что закупочная цена = 70% от цены продажи
    (p.price * 0.7) as "Предполагаемая закупка",
    p.discount_percent as "Скидка %",
    p.end_price as "Конечная цена",
    (p.end_price - (p.price * 0.7)) as "Маржа",
    ROUND((p.end_price - (p.price * 0.7)) / p.end_price * 100, 2) as "Маржинальность
    %",
    SUM(ws.stock_quantity) as "Остаток",
    SUM(ws.stock_quantity * (p.end_price - (p.price * 0.7))) as "Потенциальная
    прибыль"
FROM goods g
JOIN products p ON g.goods_id = p.goods_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
GROUP BY g.name, p.price, p.discount_percent, p.end_price
ORDER BY "Потенциальная прибыль" DESC;
```

6. Практические запросы для оперативной работы

Поиск товара по названию и проверка наличия

```
CREATE OR ALTER VIEW GOODS_AVAILABILITY
AS
SELECT
    g.name as "Название",
    g.description as "Описание",
    p.product_id,
    p.end_price as "Цена",
    SUM(ws.stock_quantity) as "Общее наличие",
    COUNT(DISTINCT w.warehouse_id) as "Количество складов",
    LIST(
        'Склад ' || w.warehouse_id || ' (' || a.address || '): ' || ws.stock_quantity || '
шт.',
        ' ';
    ) as "Наличие на складах",
    s.name as "Продавец",
    s.contact_info as "Контакты продавца"
FROM goods g
JOIN products p ON g.goods_id = p.goods_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
LEFT JOIN warehouses w ON ws.warehouse_id = w.warehouse_id
LEFT JOIN addresses a ON w.address_id = a.address_id
JOIN sellers s ON p.seller_id = s.seller_id
GROUP BY g.name, g.description, p.product_id, p.end_price, s.name, s.contact_info
ORDER BY "Общее наличие" DESC;
```

Применение

```
select * from GOODS_AVAILABILITY where UPPER("Название") LIKE '%ТЕСТ%';

select * from GOODS_AVAILABILITY where PRODUCT_ID = 23;
```

Анализ товарных остатков для планирования закупок

```
-- Шаг 1: дневные продажи по каждому товару
WITH daily_product_sales AS (
    SELECT
        oi.product_id,
        o.order_date,
        SUM(oi.quantity) AS daily_sales --
    FROM order_items oi
    JOIN orders o ON o.order_id = oi.order_id
    GROUP BY oi.product_id, o.order_date
),
-- Шаг 2: средние продажи в день для каждого товара
avg_product_sales AS (
    SELECT
        product_id,
        AVG(1.0 * daily_sales) AS avg_daily_sales
    FROM daily_product_sales
    GROUP BY product_id
)
SELECT
    g.name AS "Товар",
    SUM(ws.stock_quantity) AS "Текущий остаток",
    COALESCE(av.avg_daily_sales, 0) AS "Средние продажи в день",
    CASE
        WHEN av.avg_daily_sales > 0
        THEN ROUND(SUM(ws.stock_quantity) / av.avg_daily_sales, 1)
        ELSE 999
    END AS "Дней остатка",
    s.name AS "Поставщик"
FROM goods g
JOIN products p ON g.goods_id = p.goods_id
JOIN sellers s ON p.seller_id = s.seller_id
LEFT JOIN warehouse_stock ws ON p.product_id = ws.product_id
LEFT JOIN avg_product_sales av ON p.product_id = av.product_id
GROUP BY
    g.name,
    p.product_id,
    s.name,
    av.avg_daily_sales
HAVING SUM(ws.stock_quantity) > 0
ORDER BY "Дней остатка" ASC
--OPTIMIZE FOR FIRST ROWS;
```

Эти запросы покрывают различные аспекты работы с товарами и складами: от оперативного контроля до стратегического анализа. Вы можете адаптировать их под конкретные бизнес-потребности вашего маркетплейса.

В демо-базе могут присутствовать другие представления, не описанные на данной странице, но предназначенные или пригодные для решения различных задач анализа, планирования и оптимизации работы электронного магазина.