Szymon Kasprzycki 193065, Data Engineering

Subject of the project:

Original project subject was: 56. On-line shop: orders, packages, customers, opinions.

I changed the scope a little to – online smartphone shop: orders, packages, customers, returns.

Topic analysis:

Customer for the database: Mid/Large sized online smartphones shop.

Users of the database: All employees of the company – directors, magazine workers, data administrators, data analysts, accountants, order/return processing employees.

Purpose of the database: Manage and organize work of an online smartphone shop. Handle information related to orders, packages, customers, deliveries and invoices. Database should provide seamless and organized operating of the business. The primary goals include order processing, storing customers' information, inventory management, tracking payments, and handling returns.

Real life context: This database is prepared for the online store already having a working website. It does not support the possibility of stationary sales. It could support the efficient and stable work of the mentioned shop.

Possible use scenarios:

- Keeping track of orders/returns history,
- Retrieving a list of customers,
- Performing analysis on the daily/monthly/annual sales,
- Keeping track of the magazine state for each product,
- Fraud of the products detection,
- Calculating the shop revenue for specific period,
- Customer segmentation for targeted marketing based on analysis of their orders,
- Popular products identification,
- Returns analysis for product range improvement,
- Possibility of dynamic pricing based on analysis of order data.

Assumptions and limitations:

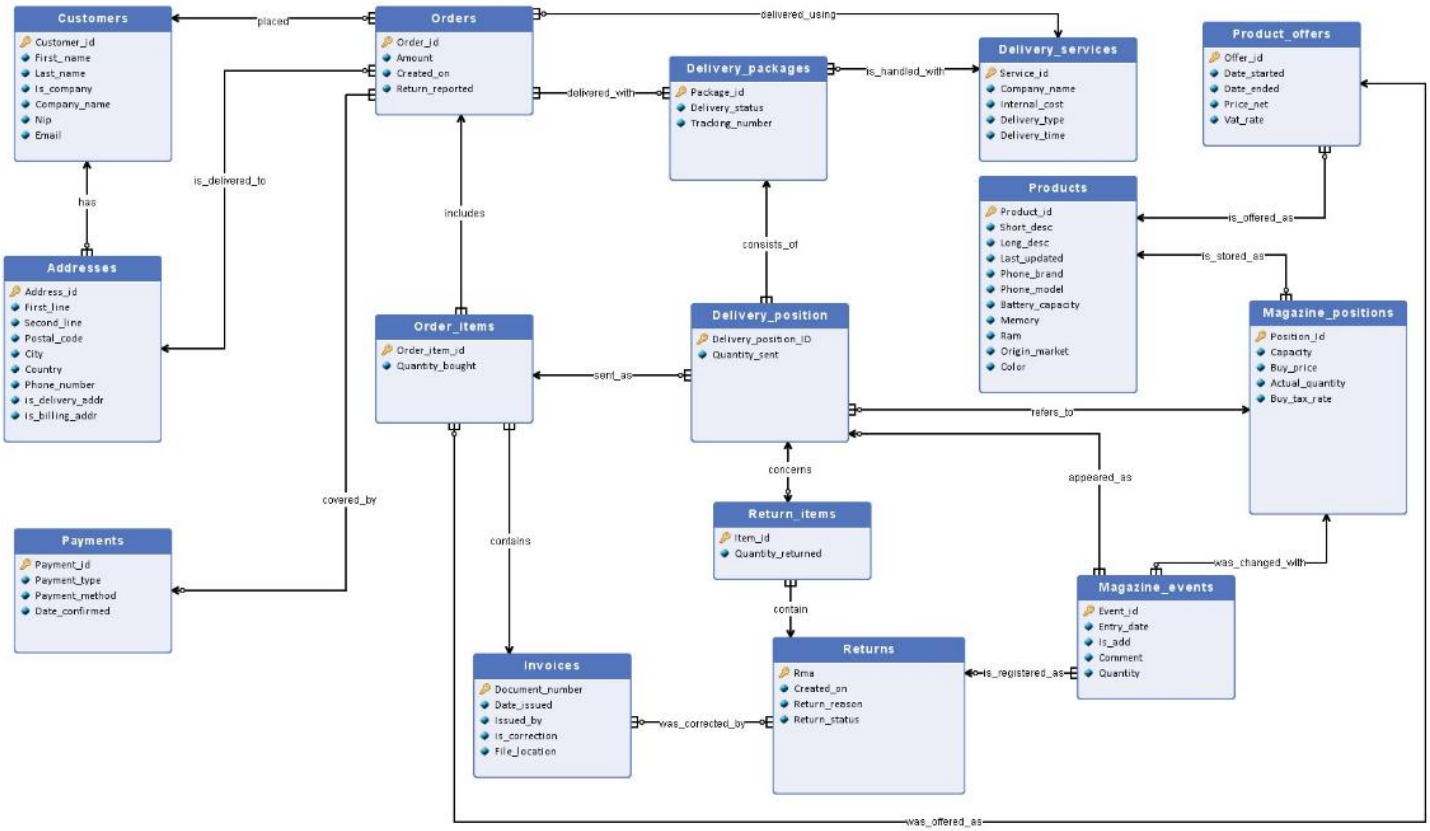
- We assume that the company has a working web store with working authentication process so we do not have to handle the way of authentication, password storing etc.
- We assume that database would be stored with appropriate level of security, because of storing the confidential data.
- We assume that database would have consistent backups.
- Database does not take part in any maintenance breaks and system changes, its work is stable and consistent.
- Database has connection only with the internal system of the company, it is achievable only from custom software available in magazine and office.
- Important note – database should be protected again SQL injection on the used software side.

Example inquiries:

- Retrieve customer information by e.g. phone number

- Retrieve customers living in specific country
- Check product availability in the magazine
- Calculate monthly/annual revenue
- Identify top-selling products
- Monitor low stock products
- List orders with the lower limit for the price
- List orders delivered by specific delivery service
- Check if the order was successfully paid
- Check if customer has defined a billing address
- Check the price for product between date in the specific day

ER Diagram:



Description of the entity sets:

Entity set: Customers
Description: Entity set containing all customers that have ever placed an order in the company or were manually added to the database.

Columns:

- **Customer_id** – integer, unique for each row: synthetic identifier for each customer
- First_ name – varchar(32): letters a-z, A-Z as well as "-" sign
- Last_name – varchar(48) : letters a-z, A-Z as well as "-" sign
- Is_company – bit: Defines whether customer is registered as a company or not (1 means company, 0 means private person)
- Company_name – varchar(200): if customer is defined as a company - its name is stored here; it must be defined as a set of ASCII characters.
- Nip – varchar(10): if customer is defined as a company - its tax identification number is stored here, it consists of 10 digits.
- Email – varchar(72): customer email used to send notification regarding orders and other communication. Should match the pattern xxxxx@yyyyy.zzz where x, y may be lowercase letters, capital letters, numbers and some specials signs like (-_.+) and z should be a lowercase or capital letter. X, y and z may occur one or multiple times each.

Entity set: Orders
Description: The list of all orders ever received by the company. Each order is connected with exactly one client, may have multiple packages and may include multiple positions of multiple products.

Columns:

- **Order_id** – int: synthetic, unique Identifier.
- Amount – float: The total cost of an order with a dot as a decimal point separator.
- Created_on – datetime: Full date in "YYYY-MM-DD hh:mm:ss[.nnn]" format.
- Return_reported – bit: Defines if there was any return connected with order. Can take only two values: 1 or 0 (1 means true, 0 false).

Entity set: Payments
Description: The list of ever received payments (for orders) and sent payments (for returns). Payment may cover the order price after some time, after registering the order in the database.

Columns:

- **Payment_id** - int: Synthetic, unique Identifier for the payment, used for internal purposes
- Payment_type – varchar(8) - It could be "received" or "sent" payment.
- Payment_method – varchar(32): One of the following – "bank transfer", "BLIK", "credit card", "cash", "fast bank transfer".
- Date_confirmed – datetime: Specifies when money achieved its destination account. Full date in "YYYY-MM-DD hh:mm:ss[.nnn]" format.

Entity set: Products
Description: The list of all products (smartphones) available in the shop with actual information and description

Columns:

- **Product_id** – int: Synthetic, unique Identifier for the product in internal systems
- Short_desc – varchar(1000): Short description of the product. Contains ASCII characters.
- Long_desc – varchar(50000): Long description of the product. Contains ASCII characters.
- Last_updated – datetime: The datetime of last entity update in "YYY-MM-DD hh:mm:ss[.nnn]" format.
- Phone_brand – varchar(100): Producer of the device. Contains ASCII characters.
- Phone_model – varchar(100): producer model identifier. Contains ASCII characters.
- Battery_capacity – int: Smartphone battery capacity in mAh, always in range 0->20000
- Memory – int: The number of gigabytes of ROM memory of smartphone e.g. 256, always in range 0->100000
- Ram – float: The number of gigabytes of RAM memory of smartphone (may contain halves), always in range 0.00->100.00
- Origin_market – varchar(50): Specifies the market where device was meant to be sold - e.g. "Asia", "Europe", "US". Contains only letters.
- Color – varchar(50): The name of color that the phone has, defined by producer. Contains ASCII characters.

Entity set: Addresses
Description: List of all customers' addresses ever received by the company - multiple addresses may be connected with one customer. Each address may be either billing or delivery or both at the same time. Each of the addresses is connected with exactly one client.

Columns:

- **Address_id** – int: Synthetic, unique Identifier of the address entity
- First_line – varchar(250): First line of the address in ISO-8859-2 standard. Contains ASCII characters.
- Second_line – varchar(250): Second line of the address in ISO-8859-2 standard. Contains ASCII characters.
- Postal_code - varchar(6): Zip (postal) code in XX-XXX format where every X is a number in range 0-9.
- City – varchar(100): Full city name in ISO-8859-2 standard. Contains ASCII characters.
- Country – varchar(100): Full country name in ISO-8859-2 standard. Consists of letters only.
- Phone_number – varchar(16): Customers phone number preceded with +XX country Identifier - field have fixed length at 16 digits as this is the longest possible phone numbers length (in China) with +XX country code included. May contain only digits.
- Is_delivery_addr – bit: Defines if address is used for delivering purpose (1 means true, 0 false)
- Is_billing_addr – bit: Defines if address is used for billing purpose (1 means true, 0 false)

Entity set: Returns
Description: The list of all returns processed ever by the company . One return may contain multiple positions from the order. Also, the return should contain at least one correction invoice (but created after receiving the products and processing the return payment).

Columns:

- **Rma** – varchar(20): Synthetic Identifier for each return e.g. RMA/XX/YYYY/ZZZZ.

- Created_on – datetime: Full date of return report by customer in "YYYY-MM-DD hh:mm:ss[.nnn]" format.
- Return_reason – varchar(500): Description of the reason of sending product back (may be blank if no reason specified). Contains ASCII characters.
- Return_status – varchar(20): As the name states, it contains internal status of processed return. May be: "return_submitted", "return_received", "return_accepted", "return_rejected").

Entity set: Order_items
Description: A list of all invoice positions - connections of specific product offer and bought quantity of this particular product.

Columns:

- **Order_item_id** – bigint: Synthetic, unique Identifier.
- Quantity_bought – int: Amount of the ordered pieces of specified product.

Entity set: Invoices
Description: A set of billing documents containing either sell invoices or correction invoices (for returns).

Columns:

- **Document_number** – varchar(20): Synthetic, unique Identifier for document e.g. FV/MM/YYYY/XXXXXX, where MM is month, YYYY is year, XXXX is the invoice number.
- Date_issued – datetime: Full date in "YYYY-MM-DD hh:mm:ss[.nnn]" format.
- Issued_by – varchar(100): First and last name of the person who issued document. Contains only letters.
- is_correction – bit: Defines whether the document is correcting previous one or not. Connected with returns. Takes only value of 0 or 1. (1 means the document is correction, 0 means it is normal invoice)
- File_location – varchar(200): Location of the file in internal system data, url or path to the file. Contains ASCII characters.

Entity set: Delivery_packages
Description: The list of shipped packages in the company. List consists of either sent and received packages (orders and returns). One package may contain items from one or multiple orders/returns.

Columns:

- **Package_id** – int: Synthetic, unique ID for internal identification
- Delivery_status – varchar(30): Package status, may be: "created_shipping_document", "packed", "handed_over_to_the_delivery", "delivered".
- Tracking_number – varchar(50): External Identifier received from shipping company. May be repetitive due to the variety of carriers. Contains ASCII characters.

Entity set: Magazine_positions
Description: The set of all magazine positions - products but from the same delivery series

Columns:

- **Position_id** – int: Synthetic, unique Identifier
- Capacity - int: The amount of the product that we can maximally store
- Buy_price - float: The price we bought product for (netto)

- Actual_quantity – int: Actual magazine state
- Buy_tax_rate – float: : The tax rate of product for the time when we bought it (e.g. 0.23)

Entity set: Magazine_events
Description: This entity set contains all events that have impact on magazine positions. E.g. adding pieces of some magazine position or removing some pieces from magazine.

Columns:

- **Event_id** – int: Synthetic, unique identifier
- Entry_date – datetime: Timestamp of the event in "YYYY-MM-DD hh:mm:ss[.nnn]" form
- Is_add – bit: Defines whether event added items to magazine or taken from magazine (1-added, 0-taken)
- Quantity – int: Defines the amount with which the magazine stock was changed.
- Comment – varchar(1000): Comment defining the purpose of event and person who issued the event. Contains ASCII characters.

Entity set: Product_offers
Description: This entity set contains all actual and historical data of sell offers of specific products.

Columns:

- **Offer_id** – int: Synthetic, unique identifier
- Date_started - datetime: The date when offer started in "YYYY-MM-DD hh:mm:ss[.nnn]" format
- Date_ended – datetime: The date when offer ended in "YYYY-MM-DD hh:mm:ss[.nnn]" format
- Price_net – float: Netto (without tax) offer selling price
- Vat_rate – float: Vat rate for this product that should be considered in total selling price (e.g. 0.23)

Entity set: Delivery_services
Description: This entity set consists of possible delivery services from which client may choose appropriate to get his parcel.

Columns:

- **Service_id** – int: Synthetic, unique Identifier
- Company_name – varchar(100): Name of the delivery company. Contains ASCII characters.
- Internal_cost – float: Cost that our company must pay for the package (including discounts). May be useful to calculate the margin of the delivery cost for customer
- Delivery_type – varchar(20): Specifies whether it is "courier" or "parcel_machine".
- Delivery_time – int: Approximate delivery time based on information provided by the delivery company

Entity set: Delivery_position
Description: This entity set consists of positions put in specific packages. It allows to retrieve the information if how many pieces of the product were sent in specific delivery package.

Columns:

- **Delivery_position_ID** – int: Synthetic, unique Identifier for internal purposes
- Quantity_sent – int: Quantity of items from the order_item (part of it or whole) sent in one package

Entity set: Return_items
Description: This entity set consists of positions returned by customer in specific return statement. It allows to retrieve the information if how many pieces of the product were returned in specific return submission.

Columns:

- **Item_id** – int: Synthetic, unique identifier
- Quantity_returned – int: Quantity of the returned items (a number between 1 and delivery_position quantity)

Relationships description:

- **Has** (Addresses – Customers, 0..n - 1) – Makes connection between specific address and customer that this address belongs to. Customer may have zero or multiple addresses, because he may want another delivery address and yet another to bill or he may still not given his address. One address has to be connected to one customer (and only one).
- **Placed** (Customers – Orders, 1 – 0..n) – Connects orders to the specified customer that submitted it. Order has to have connection with only one customer while customer may have placed zero or many orders.
- **Includes** (Orders – Order_items, 1 – 1..n) – Specifies items and their quantity in the specific order. One order item may appear in only one order, while order may contain multiple order items (but at least one).
- **Contains** (Invoices – Order_items, 1 – 1..n) – Specifies items and their quantity on the invoice. Order item here may be also called an invoice position. One order may appear only in one invoice, but invoice may contain multiple positions (but at least one).
- **Covered_by** (Orders – Payments, 1..n – 0..1) – Matches the payment and the order it was made for. One payment may cover multiple orders while one order may be covered with just one payment.
- **Delivered_with** (Orders – Deliver_packages, 1..n – 0..n) – Ties the package entity with the order it is a part of. Order may contain zero or multiple packages (optionality, because the package is not sent in the same moment as the order is submitted). While one package may contain at least one but also multiple orders (but to the same address and client).
- **Concerns** (Delivery_position – Return_items, 1 – 0..1) – Matches items submitted to return with the corresponding delivered positions. Delivery position may have zero or one corresponding return item (because not all of the items are always returned) and return item should have exactly one corresponding delivery position that is being returned.
- **Was_corrected_by** (Returns – Invoices, 0..n – 0..n) – Connects the return that was completed with correcting invoice that was made for its documentation. One return may have zero (if it is not completed) correcting invoices or multiple if for example there was an error while creating the first correction. One invoice may have none returns connected (if it is not a correcting invoice) or multiple if it concerns items from multiple returns.
- **Is_handled_with** (Delivery_packages – Delivery_services, 0..n – 1) – Connects the package with carrier that delivers it. One package may be delivered by only one carrier, but carrier may have zero or multiple delieveries of our packages.
- **Is_offered_as** (Products – Product_offers, 1 – 0..n) – Ties a product with its selling offers for specific periods. One product may have zero offers (if it was never meant to be sold) or multiple (when the price has been changing through the time). Product offer may connect only with one product it is meant for.
- **Is_stored_as** (Products – Magazine_positions, 1 – 0..n) – Matches the product with its corresponding magazine positions. One magazine position may be identified as only one product, but product may be stored as multiple magazine positions (from different suppliers, for different buy price, etc.)
- **Was_changed_with** (Magazine_positions – Magazine_events, 1 – 0..n) – Matches events occurring in magazine with corresponding positions that were changed. Event can change only one magazine

position while one magazine position may be changed by multiple magazine events (for example at different days).

- **Is_registered_as** (Returns – Magazine_events, 0..1 – 1..n) – Connects a return with a corresponding magazine event that registered returned products back on magazine. One return may be registered as multiple magazine events, because it may concern multiple products. Magazine may have only one corresponding return or don't have it when it is not connected with a return.
- **Is_delivered_to** (Addresses – Orders, 1 – 0..n) – Ties an order to the address that it should be delivered to. One order can be delivered only to one address, but one address may be a delivery address for zero or more orders.
- **Sent_as** (Order_items – Delivery_position, 1 – 0..n) – Specifies which positions (or parts of positions) from the order were sent in which package. One order position may have zero (because items are not sent immediately after order submission) or more delivery positions (because it may be splitted if item stock has run out). One delivery position corresponds to exactly one order_item.
- **Refers_to** (Delivery_position – Magazine_position, 0..n – 1) – Connects directly the delivered item to the magazine position it was. Multiple delivery positons may correspond to one magazine position (magazine position may be sent multiple times before running out of stock). However, only one magazine position may be connected to delivery position as it represents only one product.
- **Consists_of** (Delivery_position - Delivery_packages, 1..n – 1) – Specifies which positions prepared to delivery are included in the specific parcel. One parcel can contain at least one delivery position (but may contain more) while delivery position may only be included in one parcel (we cannot split it).
- **Was_offered_as** (Order_items - Product_offers, 0..n – 1) – Specifies the historical offer of the product that was active when order was made. One order item may be described only with one offer (it was ordered for specific price when this offer was active). Product offer may be connected with zero or multiple order items (it depends on the amount of orders that were made when this offer was active).
- **Delivered_using** (Orders – Delivery_services, 0..n – 1) – Defines the carrier chosen by customer for the specific order. One order can be delivered only with one carrier while delivery service may deliver multiple orders.
- **Contain** (Return_items – Returns, 1..n – 1) – Return (submission) is connected with specific items that are being returned using it. So one return item must be connected to only one return submission while return submission may contain one or more items.
- **Appeared_as** (Delivery_position – Magazine_events, 0..1 – 1..n) – Connects items prepared to be sent with an magazine event that was executed to pack these things. One delivery position can be connected with one or more magazine events because it can change states for multiple magazine positions (that are the same product but for example bought for different prices). However one magazine event can be connected with zero or one delivery position (it may not be connected with sent product. That is the reason for optionality).

RDB Schema

Customers (**Customer_id**, First_name, Last_name, is_company, Company_name, Nip, Email)

Addresses (**Address_id**, First_line, Second_line, Postal_code, City, Country, Phone_number, is_delivery_addr, is_billing_addr, CustomerConnected REF Customers)

Orders (**Order_id**, Created_on, Return_reported, CustomerID REF Customers, DeliveryService REF Delivery_Services, DeliveryAddressID REF Addresses, PaymentID REF Payments)

Payments (**Payment_id**, Payment_method, Payment_type, Date_confirmed)

Delivery_services (**Delivery_service_id**, Company_name, Internal_cost, Delivery_type, Delivery_time)

Delivery_packages (**Package_id**, Delivery_status, Is_return, Tracking_number, DeliveryServiceID REF Delivery_services)

Order_items (**Order_item_id**, Quantity_bought, OrderID REF Orders, OfferID REF Product_offers, InvoiceID REF Invoices)

Product_offers (**Offer_id**, Date_started, Date_ended, Price_net, Vat_rate, ProductID REF Products)

Products (**Product_id**, Short_desc, Long_desc, Last_updated, Phone_brand, Phone_model, Battery_capacity, Memory, Ram, Origin_market, Color)

Magazine_positions (**Position_id**, Capacity, Buy_price, Actual_quantity, Buy_tax_rate, ProductID REF Products)

Magazine_events (**Event_id**, Entry_date, Is_add, Quantity, Comment, MagazinePositionID REF Magazine_positions, DeliveryPositionID REF Delivery_Position)

Returns (**Rma**, Created_on, Return_reason, Return_status)

Return_items (**Item_id**, Quantity_returned, ReturnID REF Returns, DeliveryPositionID REF Delivery_position)

Invoices (**Document_number**, Date_issued, Issued_by, is_correction, file_location)

Delivery_position (**Delivery_position_id**, Quantity_sent, packageID REF Delivery_packages, orderItemID REF Order_items, magazinePositionID REF Magazine_Position)

Deliveries (**OrderID** REF Orders, **PackageID** REF Delivery_packages)

Invoice_corrects (**InvoiceID** REF Invoices, **ReturnID** REF Returns)

Returned_items_events (Rma REF Returns, **MagazineEventID** REF Magazine_events)