Bazy danych - sprawozdanie z projektu

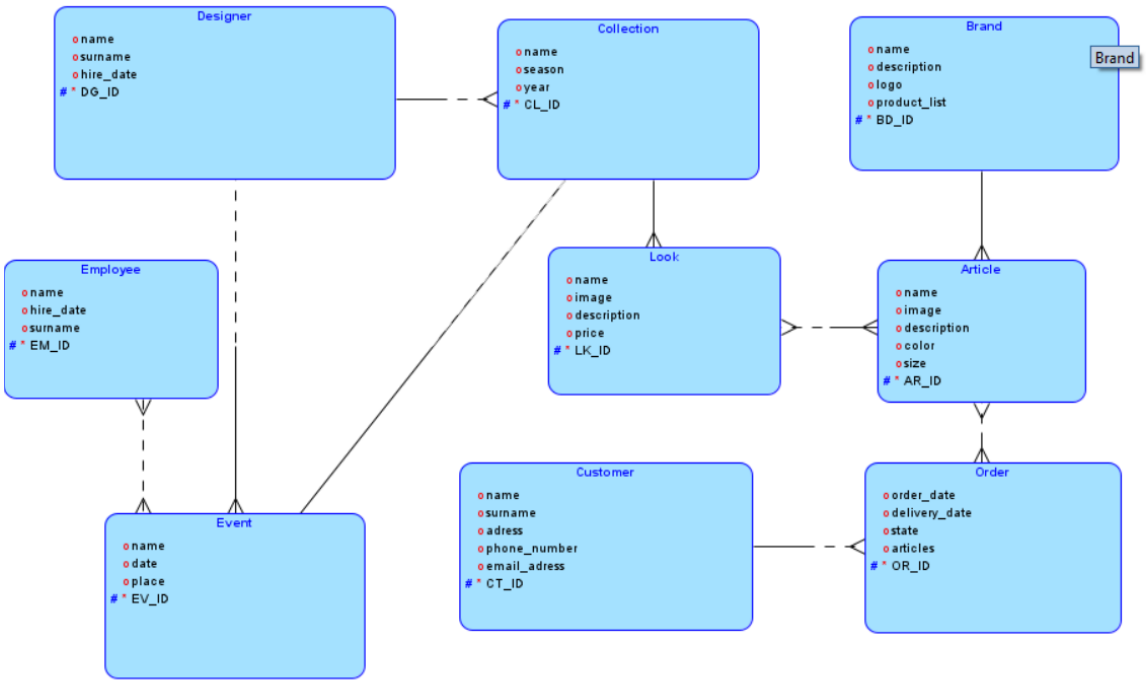
1. Podstawowe informacje

- **Autorzy projektu:** Szymon Pawłowski, Mateusz Pliszka, Jakub Więckowski.....
- **Nazwa grupy:** Analityka Gospodarcza.....
- **Temat:** Dom mody .....
- **Wybrany zakres (ocena):** 5 (w razie braku spełnienia, któregoś z wymagań prosimy o kontakt, ponieważ bardzo zależy nam na ocenie bardzo dobrej) .....
- **Baza danych:** ie83685 SP83685 .....

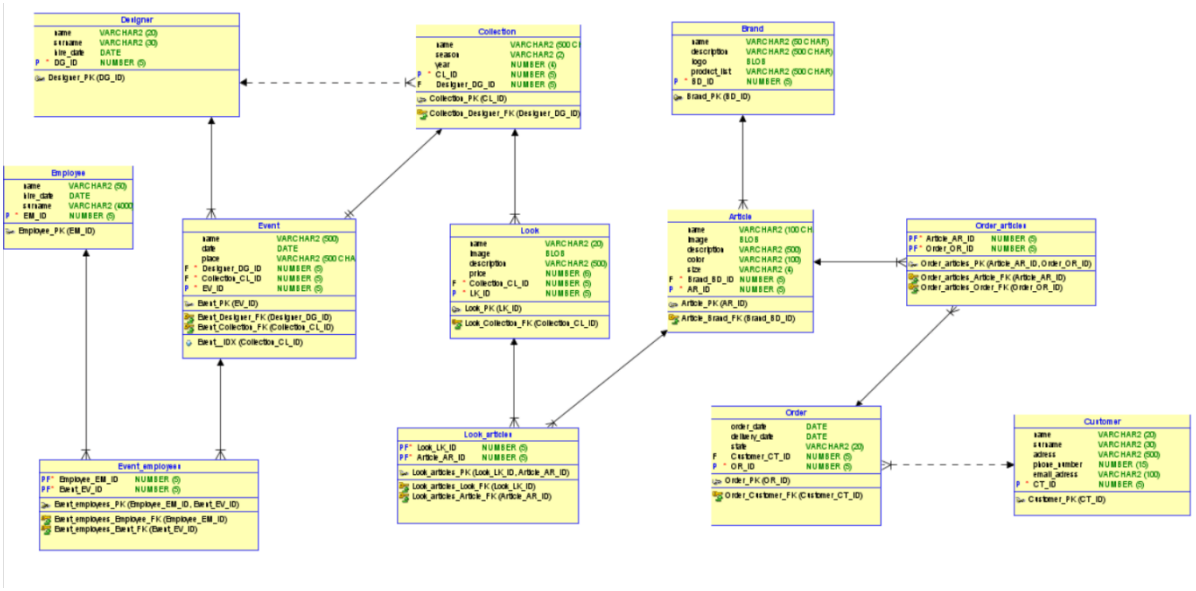
2. Wprowadzenie – opis studium przypadku

W domu mody mamy kolekcje, które mają swojego designera, swoją stylizacje i są przedstawiane na eventach. Designer przedstawia swoje kolekcje na eventach, które są przygotowywane przez pracowników. Stylizacje składają się z różnych artykułów, które mają swoją markę i są zamawiane przez klientów.

3. Model związków encji



4. Model relacyjnej bazy danych



5. Schemat bazy - tabele

DDL

```
-- Generated by Oracle SQL Developer Data Modeler 22.2.0.165.1149
--   at:        2023-01-31 13:11:38 CET
--   site:      Oracle Database 11g
--   type:      Oracle Database 11g



-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE TABLE article (
    name        VARCHAR2(100 CHAR),
    image       BLOB,
    description VARCHAR2(500),
    color       VARCHAR2(100),
    "size"      VARCHAR2(4),
    brand_bd_id NUMBER(5) NOT NULL,
    ar_id       NUMBER(5) NOT NULL
);

ALTER TABLE article ADD CONSTRAINT article_pk PRIMARY KEY ( ar_id );

CREATE TABLE brand (
    name         VARCHAR2(50 CHAR),
    description  VARCHAR2(500 CHAR),
    logo         BLOB,
    product_list VARCHAR2(500 CHAR),
    bd_id        NUMBER(5) NOT NULL
);

ALTER TABLE brand ADD CONSTRAINT brand_pk PRIMARY KEY ( bd_id );
```

```
CREATE TABLE collection (
    name          VARCHAR2(500 CHAR),
    season        VARCHAR2(2),
    year          NUMBER(4),
    cl_id         NUMBER(5) NOT NULL,
    designer_dg_id NUMBER(5)
);

ALTER TABLE collection ADD CONSTRAINT collection_pk PRIMARY KEY ( cl_id );

CREATE TABLE customer (
    name         VARCHAR2(20),
    surname      VARCHAR2(30),
    adress       VARCHAR2(500),
    phone_number NUMBER(15),
    email_adress VARCHAR2(100),
    ct_id        NUMBER(5) NOT NULL
);

ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( ct_id );

CREATE TABLE designer (
    name      VARCHAR2(20),
    surname   VARCHAR2(30),
    hire_date DATE,
    dg_id     NUMBER(5) NOT NULL
);

ALTER TABLE designer ADD CONSTRAINT designer_pk PRIMARY KEY ( dg_id );

CREATE TABLE employee (
    name      VARCHAR2(50),
    hire_date DATE,
    surname   VARCHAR2(4000),
    em_id     NUMBER(5) NOT NULL
);

ALTER TABLE employee ADD CONSTRAINT employee_pk PRIMARY KEY ( em_id );

CREATE TABLE event (
    name            VARCHAR2(500),
    "date"          DATE,
    place           VARCHAR2(500 CHAR),
    designer_dg_id   NUMBER(5) NOT NULL,
    collection_cl_id NUMBER(5) NOT NULL,
    ev_id           NUMBER(5) NOT NULL
);

CREATE UNIQUE INDEX event__idx ON
    event (
        collection_cl_id
    ASC );

ALTER TABLE event ADD CONSTRAINT event_pk PRIMARY KEY ( ev_id );
```

```
CREATE TABLE event_employees (
    employee_em_id NUMBER(5) NOT NULL,
    event_ev_id    NUMBER(5) NOT NULL
);

ALTER TABLE event_employees ADD CONSTRAINT event_employees_pk PRIMARY KEY (
employee_em_id,
                                                event_ev_id );

CREATE TABLE look (
    name           VARCHAR2(20),
    image          BLOB,
    description    VARCHAR2(500),
    price          NUMBER(6),
    collection_cl_id NUMBER(5) NOT NULL,
    lk_id          NUMBER(5) NOT NULL
);

ALTER TABLE look ADD CONSTRAINT look_pk PRIMARY KEY ( lk_id );

CREATE TABLE look_articles (
    look_lk_id    NUMBER(5) NOT NULL,
    article_ar_id NUMBER(5) NOT NULL
);

ALTER TABLE look_articles ADD CONSTRAINT look_articles_pk PRIMARY KEY ( look_lk_id,
                                                article_ar_id );

CREATE TABLE "Order" (
    order_date     DATE,
    delivery_date  DATE,
    state          VARCHAR2(20),
    articles       VARCHAR2(500),
    customer_ct_id NUMBER(5),
    or_id          NUMBER(5) NOT NULL
);

ALTER TABLE "Order" ADD CONSTRAINT order_pk PRIMARY KEY ( or_id );

CREATE TABLE order_articles (
    article_ar_id NUMBER(5) NOT NULL,
    order_or_id   NUMBER(5) NOT NULL
);

ALTER TABLE order_articles ADD CONSTRAINT order_articles_pk PRIMARY KEY (
article_ar_id,
                                                order_or_id );

ALTER TABLE article
    ADD CONSTRAINT article_brand_fk FOREIGN KEY ( brand_bd_id )
        REFERENCES brand ( bd_id );

ALTER TABLE collection
    ADD CONSTRAINT collection_designer_fk FOREIGN KEY ( designer_dg_id )
        REFERENCES designer ( dg_id );
```

```
ALTER TABLE event
   ADD CONSTRAINT event_collection_fk FOREIGN KEY ( collection_cl_id )
      REFERENCES collection ( cl_id );

ALTER TABLE event
   ADD CONSTRAINT event_designer_fk FOREIGN KEY ( designer_dg_id )
      REFERENCES designer ( dg_id );

ALTER TABLE event_employees
   ADD CONSTRAINT event_employees_employee_fk FOREIGN KEY ( employee_em_id )
      REFERENCES employee ( em_id );

ALTER TABLE event_employees
   ADD CONSTRAINT event_employees_event_fk FOREIGN KEY ( event_ev_id )
      REFERENCES event ( ev_id );

ALTER TABLE look_articles
   ADD CONSTRAINT look_articles_article_fk FOREIGN KEY ( article_ar_id )
      REFERENCES article ( ar_id );

ALTER TABLE look_articles
   ADD CONSTRAINT look_articles_look_fk FOREIGN KEY ( look_lk_id )
      REFERENCES look ( lk_id );

ALTER TABLE look
   ADD CONSTRAINT look_collection_fk FOREIGN KEY ( collection_cl_id )
      REFERENCES collection ( cl_id );

ALTER TABLE order_articles
   ADD CONSTRAINT order_articles_article_fk FOREIGN KEY ( article_ar_id )
      REFERENCES article ( ar_id );

ALTER TABLE order_articles
   ADD CONSTRAINT order_articles_order_fk FOREIGN KEY ( order_or_id )
      REFERENCES "Order" ( or_id );

ALTER TABLE "Order"
   ADD CONSTRAINT order_customer_fk FOREIGN KEY ( customer_ct_id )
      REFERENCES customer ( ct_id );
```

DML

Wszystkie wartości wstawione do tabel zostały uzupełnione za pomocą funkcji insert.into… value…

6. Przypadki użycia bazy danych

-- Zapytania --

```
-- 1. Suma jaką klient wydał na zamówienia --
SELECT Customer.name, Customer.surname, SUM(Article.price) AS total_spent
FROM Customer
JOIN Orders ON Customer.ct_id = Orders.customer_ct_id
JOIN Order_articles ON Orders.or_id = Order_articles.order_or_id
```

```
JOIN Article ON Order_articles.article_ar_id = Article.ar_id
GROUP BY Customer.name, Customer.surname
ORDER BY total_spent DESC;

-- 2. Ilość dni, w których dostarczono zamówienie i informacje o opóźnieniu dostawy--
SELECT or_id, delivery_date - order_date AS delivery_time,
CASE
WHEN (delivery_date - order_date) >= 3 THEN (delivery_date - order_date) - 3
ELSE 0
END AS delivery_delay
FROM orders
WHERE state = 'completed';

-- 3. Kontakt do ludzi, którzy przez ostatnie 2 lata zamówili chinosy --
SELECT Customer.name, Customer.surname, Customer.phone_number, Customer.email_address
FROM Customer
JOIN orders ON Customer.ct_id = orders.customer_ct_id
JOIN order_articles ON orders.or_id = order_articles.order_or_id
JOIN Article ON order_articles.article_ar_id = Article.ar_id
WHERE
Article.name = 'Chinosy'
AND orders.order_date >= ADD_MONTHS(SYSDATE, -24);



-- 4. Informacja o najbliższym evencie i pracownikach, pracujących krócej niż rok w celu przeszkolenia
--
WITH
  emp_seniority AS (
    SELECT em_id, hire_date, ROUND(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) AS
seniority_in_years
    FROM employee
  ),
  events_and_seniority AS (
    SELECT e.name, e.event_date, e.place, es.seniority_in_years, es.em_id
    FROM event e
    JOIN event_employees ee ON e.ev_id = ee.event_ev_id
    JOIN emp_seniority es ON ee.employee_em_id = es.em_id
  )
SELECT *
FROM events_and_seniority
WHERE event_date = (
  SELECT MIN(event_date)
  FROM events_and_seniority
)
AND seniority_in_years < 1;

-- 5. Informację o sprzedaży dla konkretnych kolekcji i ich projektancie --
SELECT
  Collection.name as collection_name,
  Designer.name AS designer_name,
  Designer.surname AS designer_surname,
  SUM(Article.price) AS total_revenue
FROM
  Article
```

```
  JOIN look_articles ON Article.ar_id = look_articles.article_ar_id
  JOIN Look ON look_articles.look_lk_id = Look.lk_id
  JOIN Collection ON Look.collection_cl_id = Collection.cl_id
  JOIN Designer ON Collection.designer_dg_id = Designer.dg_id
GROUP BY
  Collection.name,
  Designer.name,
  Designer.surname
ORDER BY
  total_revenue DESC;

-- 6. Wyświetl looki, których cena jest większa niż średnia cena dla wszystkich looków --
SELECT look.name, SUM(article.price) as total_price
FROM look
JOIN look_articles ON look.lk_id = look_articles.look_lk_id
JOIN article ON look_articles.article_ar_id = article.ar_id
GROUP BY look.name
HAVING SUM(article.price) > (SELECT AVG(total_price) FROM (
    SELECT SUM(article.price) as total_price
    FROM look
    JOIN look_articles ON look.lk_id = look_articles.look_lk_id
    JOIN article ON look_articles.article_ar_id = article.ar_id
    GROUP BY look.name
));

-- 7. Informacje o pracownikach zatrudnionych przy największej liczbie eventów i informacje o tych eventach --
SELECT employee.name, employee.surname, event.name as event_name, event.event_date, event.place as event_place
FROM (SELECT employee_em_id, COUNT(*) AS num_events
FROM event_employees
GROUP BY employee_em_id
ORDER BY num_events DESC
FETCH FIRST 1 ROW ONLY) most_events
INNER JOIN employee ON most_events.employee_em_id = employee.em_id
INNER JOIN event_employees ON most_events.employee_em_id = event_employees.employee_em_id
INNER JOIN event ON event_employees.event_ev_id = event.ev_id;

-- 8. Informacje o artykułach wchodzacych w sklad najbardziej skomplikowanego looku --
SELECT l.name AS look_name, l.description, a.name AS article_name, a.description AS article_description, a.color
FROM Look l
JOIN look_articles la ON l.lk_id = la.look_lk_id
JOIN article a ON la.article_ar_id = a.ar_id
WHERE l.lk_id = (SELECT look_lk_id
FROM look_articles
GROUP BY look_lk_id
ORDER BY COUNT(*) DESC
FETCH FIRST 1 ROW ONLY);
```

7. Pozostałe obiekty bazy danych

-- Utworzenie widoków --

```
-- 1. Suma wydana na zamówienia przez danych klientów --
CREATE VIEW TOTAL_SPENT AS
SELECT Customer.name, Customer.surname, SUM(Article.price) AS total_spent
FROM Customer
JOIN Orders ON Customer.ct_id = Orders.customer_ct_id
JOIN Order_articles ON Orders.or_id = Order_articles.order_or_id
JOIN Article ON Order_articles.article_ar_id = Article.ar_id
GROUP BY Customer.name, Customer.surname
ORDER BY total_spent DESC;

-- 2. Informacje o dostawie --
CREATE VIEW DELIVERY_INFO AS
SELECT or_id, delivery_date - order_date AS delivery_time,
CASE
WHEN (delivery_date - order_date) >= 3 THEN (delivery_date - order_date) - 3
ELSE 0
END AS delivery_delay
FROM orders
WHERE state = 'completed';

-- 3. Dane kontaktowe do ludzi, którzy w ciagu 2 lat zamowili dany produkt --
CREATE VIEW SPECIFIC_ARTICLE_CUSTOMER_INFO AS
SELECT Customer.name, Customer.surname, Customer.phone_number, Customer.email_address
FROM Customer
JOIN orders ON Customer.ct_id = orders.customer_ct_id
JOIN order_articles ON orders.or_id = order_articles.order_or_id
JOIN Article ON order_articles.article_ar_id = Article.ar_id
WHERE
Article.name = 'Chinosy'
AND orders.order_date >= ADD_MONTHS(SYSDATE, -24);

-- 4. Przeszkolenie nowych pracowników podczas eventu --
CREATE VIEW NEW_EMPLOYEE_TRAINING AS
WITH
  emp_seniority AS (
    SELECT em_id, hire_date, ROUND(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) AS
seniority_in_years
    FROM employee
  ),
  events_and_seniority AS (
    SELECT e.name, e.event_date, e.place, es.seniority_in_years, es.em_id
    FROM event e
    JOIN event_employees ee ON e.ev_id = ee.event_ev_id
    JOIN emp_seniority es ON ee.employee_em_id = es.em_id
  )
SELECT *
FROM events_and_seniority
WHERE event_date = (
  SELECT MIN(event_date)
  FROM events_and_seniority
)
```

```
AND seniority_in_years < 1;

-- 5. Sprzedaż dla kolekcji --
CREATE VIEW COLLECTION_REVENUE AS
SELECT
  Collection.name as collection_name,
  Designer.name AS designer_name,
  Designer.surname AS designer_surname,
  SUM(Article.price) AS total_revenue
FROM
  Article
  JOIN look_articles ON Article.ar_id = look_articles.article_ar_id
  JOIN Look ON look_articles.look_lk_id = Look.lk_id
  JOIN Collection ON Look.collection_cl_id = Collection.cl_id
  JOIN Designer ON Collection.designer_dg_id = Designer.dg_id
GROUP BY
  Collection.name,
  Designer.name,
  Designer.surname
ORDER BY
  total_revenue DESC;

-- 6. Looki posiadające cene powyżej średniej --
CREATE VIEW OVER_AVERAGE_LOOKS AS
SELECT look.name, SUM(article.price) as total_price
FROM look
JOIN look_articles ON look.lk_id = look_articles.look_lk_id
JOIN article ON look_articles.article_ar_id = article.ar_id
GROUP BY look.name
HAVING SUM(article.price) > (SELECT AVG(total_price) FROM (
    SELECT SUM(article.price) as total_price
    FROM look
    JOIN look_articles ON look.lk_id = look_articles.look_lk_id
    JOIN article ON look_articles.article_ar_id = article.ar_id
    GROUP BY look.name
));


-- 7. Pracownicy przypisani do największej liczby eventów --
CREATE VIEW MOST_EVENT_EMPLOYEE AS
SELECT employee.name, employee.surname, event.name as event_name, event.event_date,
event.place as event_place
FROM (SELECT employee_em_id, COUNT(*) AS num_events
FROM event_employees
GROUP BY employee_em_id
ORDER BY num_events DESC
FETCH FIRST 1 ROW ONLY) most_events
INNER JOIN employee ON most_events.employee_em_id = employee.em_id
INNER JOIN event_employees ON most_events.employee_em_id =
event_employees.employee_em_id
INNER JOIN event ON event_employees.event_ev_id = event.ev_id;

-- 8. Najbardziej skomplikowany look --
CREATE VIEW MOST_COMPLICATED_LOOK AS
```

```
SELECT l.name AS look_name, l.description, a.name AS article_name, a.description AS article_description, a.color
FROM Look l
JOIN look_articles la ON l.lk_id = la.look_lk_id
JOIN article a ON la.article_ar_id = a.ar_id
WHERE l.lk_id = (SELECT look_lk_id
FROM look_articles
GROUP BY look_lk_id
ORDER BY COUNT(*) DESC
FETCH FIRST 1 ROW ONLY);
```

-- Sekwencery --

```
-- 1. Sekwencer artykułów --
CREATE SEQUENCE article_sequencer
START WITH 50006
INCREMENT BY 1;
-- Wykorzystanie sekwencera --
INSERT INTO ARTICLE(name, description, color, article_size, brand_bd_id, ar_id, price)
VALUES ('Czapka z daszkiem', 'Nakrycie głowy', 'black', 'M', 30026, article_sequencer.NEXTVAL, 80);

-- 2. Sekwencer brandów --
CREATE SEQUENCE brand_sequencer
START WITH 30001
INCREMENT BY 1;
-- Wykorzystanie sekwencera --
INSERT INTO BRAND(name, description, bd_id) VALUES ('House', 'Tani brand odzieżowy', brand_sequencer.NEXTVAL);
```