

WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI

SPRAWOZDANIE
z projektu

Temat pracy: BAZA DANYCH KINA

Wykonał:

Rafał KOSTRZEWA

GRUPA:

WCY21KA1S1

Nr albumu:

80878

Warszawa 2024

1.WSTĘP

Temat bazy to Kino.

W bazie znajdują się informacje o różnych placówkach kin. Dokładniej mamy informacje o znajdujących się tam salach, miejscach (Fotelach) i biletach. Również mamy podgląd do repertuaru filmowego wraz z obsadą.

Zaprojektowanie bazy danych tworzy solidną podstawę do analizy efektywności operacyjnej kina, identyfikacji trendów w preferencjach widzów, a także wspomaga podejmowanie decyzji dotyczących repertuaru, cen biletów czy rozwoju biznesu.

Baza umożliwia:

1. Sprzedaż Biletów:

- Baza danych umożliwia monitorowanie sprzedaży biletów, co pozwala na identyfikację okresów o największym popycie.
- Analiza średniej ceny biletu pozwala na dostosowanie strategii cenowej do preferencji klientów.

2. Popularność Filmów:

- Dzięki zgromadzonym danym można ocenić popularność poszczególnych filmów, co jest kluczowe dla planowania repertuaru.

3. Wykorzystanie Miejsc w Salach Kinowych:

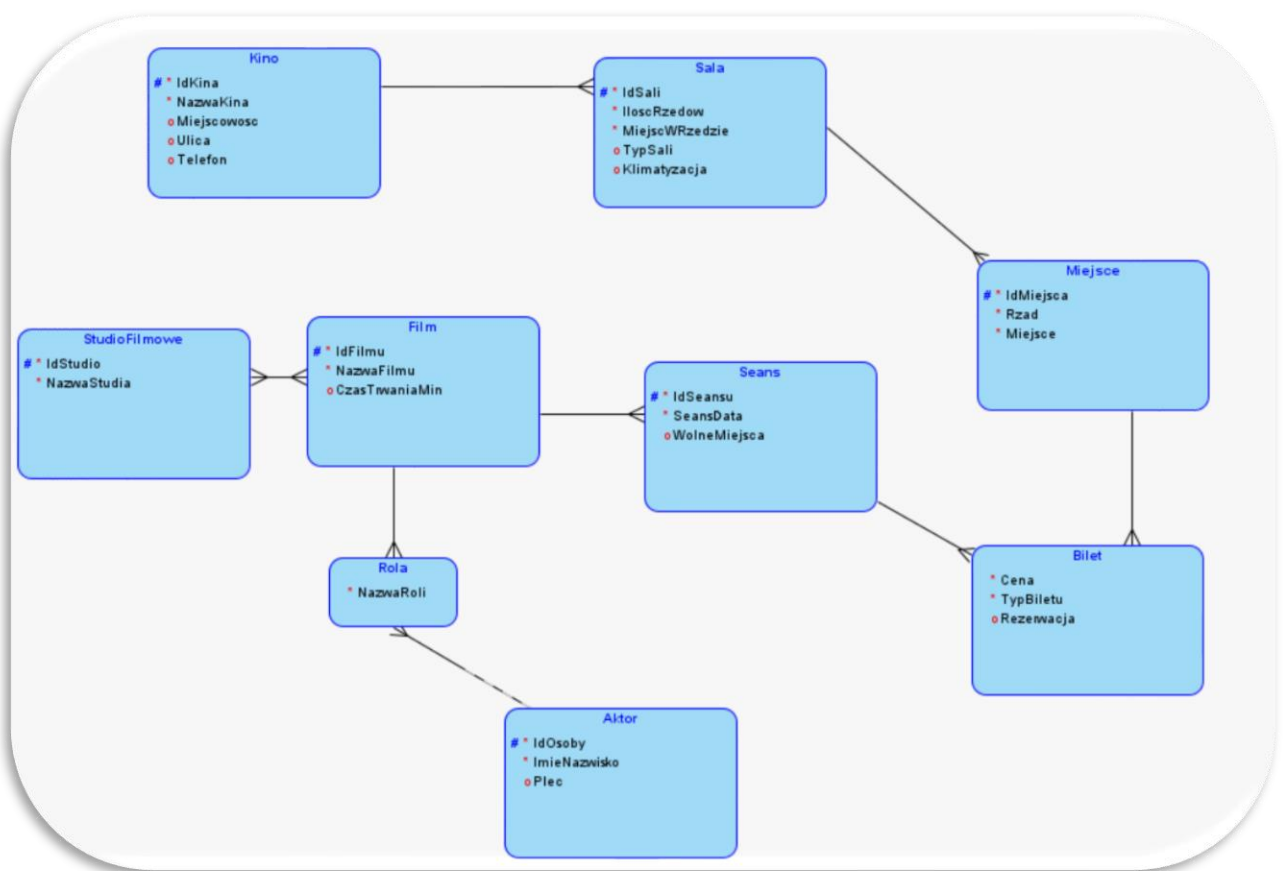
- System pozwala monitorować efektywność wykorzystania miejsc w poszczególnych salach kinowych.
- Informacje te są istotne do optymalizacji przestrzeni kinowej oraz zarządzania dostępnymi seansami.

4. Dochód Kina:

- Analiza dochodu kina w różnych okresach pozwala na śledzenie ogólnej rentowności działalności.
- Porównanie dochodu per salę kinową umożliwia ocenę efektywności zarządzania dostępnymi zasobami.

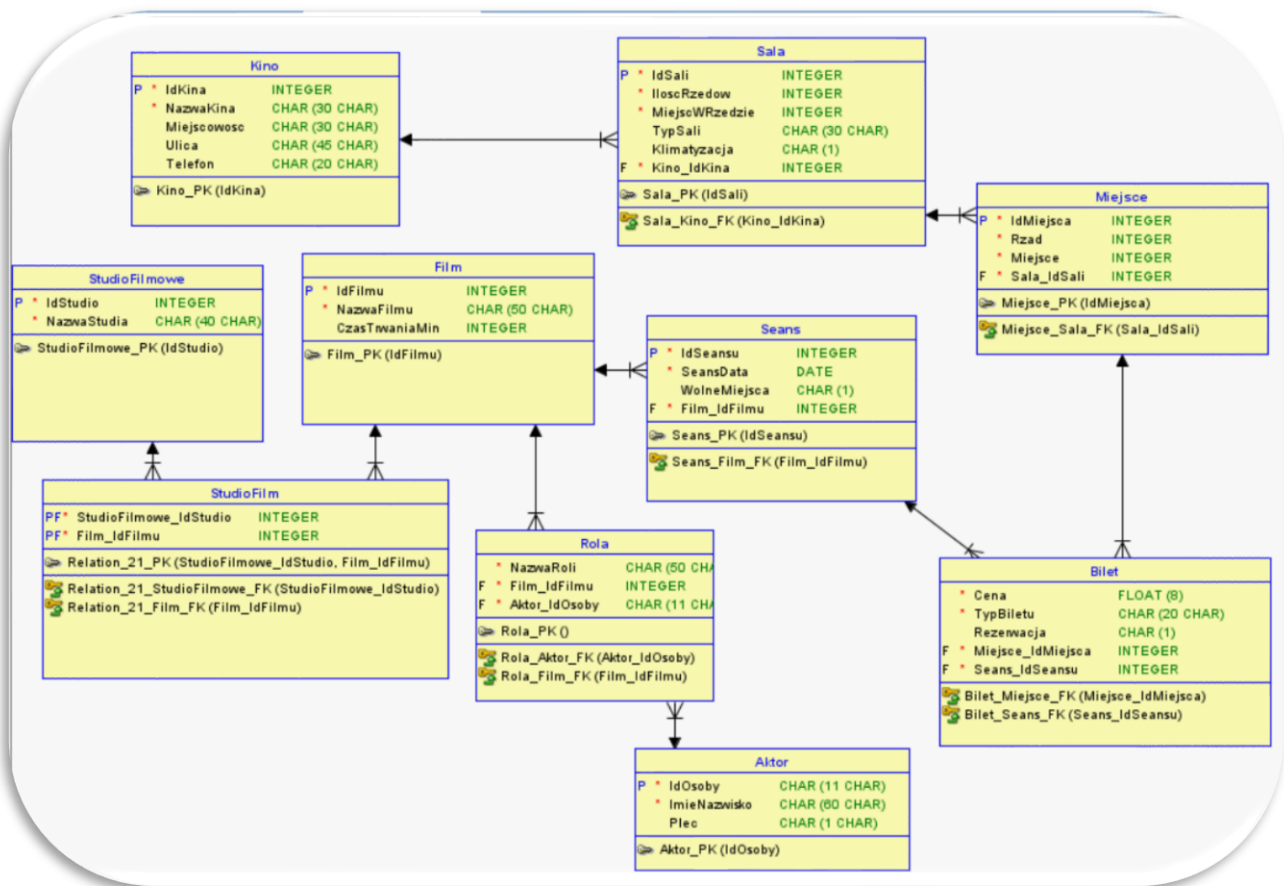
2.MODEL LOGICZNY

Model logiczny w projektowaniu baz danych jest etapem, który koncentruje się na reprezentacji struktury danych w sposób bardziej abstrakcyjny niż model fizyczny. W tym etapie skupiamy się na tym, jak dane powinny być zorganizowane i jak będą ze sobą powiązane, nie biorąc jeszcze pod uwagę konkretnych aspektów implementacyjnych, takich jak typy danych czy indeksy. Inaczej model logiczny dostarcza abstrakcyjną reprezentację struktury danych i relacji między nimi, co umożliwia projektantom baz danych zrozumienie, jak dane są ze sobą powiązane, zanim zajmą się szczegółami implementacyjnymi w modelu fizycznym.



3.MODEL RELACYJNY

Model relacyjny to struktura organizacji danych w bazie danych, opierająca się na relacjach między tabelami. W tym modelu dane są reprezentowane za pomocą tabel, gdzie każda tabela reprezentuje pewną encję, a kolumny tabel odpowiadają atrybutom tej encji. Klucze główne i obce są wykorzystywane do określenia relacji między tabelami. Model relacyjny pomaga w strukturyzacji informacji, ułatwia proces minimalizacji redundancji danych oraz zapewnieniu integralności i spójności danych.



4. OPROGRAMOWANIE

Oracle SQL Data Modeler: Oracle SQL Data Modeler to narzędzie do projektowania baz danych, które umożliwia modelowanie danych i generowanie skryptów SQL. Główne zastosowania obejmują:

1. **Projektowanie Baz Danych:** Umożliwia projektowanie struktury bazy danych, tworzenie tabel, relacji, indeksów itp. Przy użyciu graficznego interfejsu użytkownika, projektanci mogą definiować i wizualizować schemat bazy danych.
2. **Generowanie Skryptów SQL:** SQL Data Modeler umożliwia generowanie skryptów SQL na podstawie zaprojektowanego modelu danych. Te skrypty mogą być wykorzystywane do utworzenia struktury bazy danych.

Oracle SQL Developer: Oracle SQL Developer to środowisko programistyczne do zarządzania i rozwijania baz danych Oracle. Główne zastosowania obejmują:

1. **Programowanie SQL:** SQL Developer oferuje zaawansowany edytor SQL z funkcjami takimi jak kolorowanie składni, podpowiedzi, formatowanie kodu, co ułatwia pisanie, testowanie i debugowanie zapytań SQL.

2. **Zarządzanie Bazą Danych:** Umożliwia zarządzanie bazą danych Oracle, w tym tworzenie, edycję i usuwanie obiektów bazodanowych (takich jak tabele, widoki, procedury składowane).
3. **Monitorowanie Wydajności:** Narzędzie pozwala na monitorowanie wydajności bazy danych, analizę planów wykonania zapytań oraz optymalizację zapytań.
4. **Migracja Danych:** SQL Developer zapewnia narzędzia do migracji danych, co ułatwia przenoszenie danych między różnymi środowiskami.
5. **Rozwijanie Aplikacji:** Działa jako środowisko programistyczne do rozwijania aplikacji bazodanowych, wspierając języki takie jak PL/SQL.

5. SKRYPTY WDROŻENIOWE I DEINSTALUJĄCE

• SKRYPT WDROŻENIOWY INSTALACJĘ

```
-- Generated by Oracle SQL Developer Data Modeler 23.1.0.087.0806
-- at:    2024-01-23 11:58:34 CET
-- site:   Oracle Database 12cR2
-- type:   Oracle Database 12cR2

-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE TABLE aktor (
  idosoby   CHAR(11 CHAR) NOT NULL,
  imienazwisko CHAR(60 CHAR) NOT NULL,
  plec     CHAR(1 CHAR)
);

COMMENT ON COLUMN aktor.idosoby IS
  'PESEL';

ALTER TABLE aktor ADD CONSTRAINT aktor_pk PRIMARY KEY ( idosoby );

CREATE TABLE bilet (
  cena      FLOAT(8) NOT NULL,
  typbiletu CHAR(20 CHAR) NOT NULL,
  rezerwacja NUMBER,
  miejsce_idmiejsca INTEGER NOT NULL,
  seans_idseansu  INTEGER NOT NULL
);

CREATE TABLE film (
  idfilmu   INTEGER NOT NULL,
  nazwafilmu CHAR(50 CHAR) NOT NULL,
  czastrwaniamin INTEGER
);

ALTER TABLE film ADD CONSTRAINT film_pk PRIMARY KEY ( idfilmu );

CREATE TABLE kino (
  idkina   INTEGER NOT NULL,
  nazwakina CHAR(30 CHAR) NOT NULL,
  miejscowosc CHAR(30 CHAR),
  ulica     CHAR(45 CHAR),
  telefon   CHAR(20 CHAR)
);

ALTER TABLE kino ADD CONSTRAINT kino_pk PRIMARY KEY ( idkina );

CREATE TABLE miejsce (
  idmiejsca INTEGER NOT NULL,
  rzad      INTEGER NOT NULL,
  miejsce   INTEGER NOT NULL,
  sala_idsali INTEGER NOT NULL
);

ALTER TABLE miejsce ADD CONSTRAINT miejsce_pk PRIMARY KEY ( idmiejsca );

CREATE TABLE rola (
  nazwaroli CHAR(50 CHAR) NOT NULL,
  film_idfilmu INTEGER NOT NULL,
  aktor_idosoby CHAR(11 CHAR) NOT NULL
);
```



```

CREATE TABLE sala (
    idsali    INTEGER NOT NULL,
    ilosczerwcow  INTEGER NOT NULL,
    miejscwczedzie INTEGER NOT NULL,
    typsali   CHAR(30 CHAR),
    klimatyzacja  NUMBER,
    kino_idkina  INTEGER NOT NULL
);

ALTER TABLE sala ADD CONSTRAINT sala_pk PRIMARY KEY ( idsali );

CREATE TABLE seans (
    idseansu  INTEGER NOT NULL,
    seansdata  DATE NOT NULL,
    wolnemiesca  NUMBER,
    film_idfilmu  INTEGER NOT NULL
);

ALTER TABLE seans ADD CONSTRAINT seans_pk PRIMARY KEY ( idseansu );

CREATE TABLE studiofilm (
    studiofilmowe_idstudio  INTEGER NOT NULL,
    film_idfilmu            INTEGER NOT NULL
);

ALTER TABLE studiofilm ADD CONSTRAINT relation_21_pk PRIMARY KEY ( studiofilmowe_idstudio,
                                                                    film_idfilmu );

CREATE TABLE studiofilmowe (
    idstudio  INTEGER NOT NULL,
    nazwastudia  CHAR(40 CHAR) NOT NULL
);

ALTER TABLE studiofilmowe ADD CONSTRAINT studiofilmowe_pk PRIMARY KEY ( idstudio );

ALTER TABLE bilet
    ADD CONSTRAINT bilet_miejsce_fk FOREIGN KEY ( miejsce_idmiejsca )
        REFERENCES miejsce ( idmiejsca )
        NOT DEFERRABLE;

ALTER TABLE bilet
    ADD CONSTRAINT bilet_seans_fk FOREIGN KEY ( seans_idseansu )
        REFERENCES seans ( idseansu )
        NOT DEFERRABLE;

ALTER TABLE miejsce
    ADD CONSTRAINT miejsce_sala_fk FOREIGN KEY ( sala_idsali )
        REFERENCES sala ( idsali )
        NOT DEFERRABLE;

ALTER TABLE studiofilm
    ADD CONSTRAINT relation_21_film_fk FOREIGN KEY ( film_idfilmu )
        REFERENCES film ( idfilmu )
        NOT DEFERRABLE;

ALTER TABLE studiofilm
    ADD CONSTRAINT relation_21_studiofilmowe_fk FOREIGN KEY ( studiofilmowe_idstudio )
        REFERENCES studiofilmowe ( idstudio )
        NOT DEFERRABLE;

ALTER TABLE rola
    ADD CONSTRAINT rola_aktor_fk FOREIGN KEY ( aktor_idosoby )
        REFERENCES aktor ( idosoby )
        NOT DEFERRABLE;

ALTER TABLE rola
    ADD CONSTRAINT rola_film_fk FOREIGN KEY ( film_idfilmu )
        REFERENCES film ( idfilmu )
        NOT DEFERRABLE;

```

```
ALTER TABLE sala
ADD CONSTRAINT sala_kino_fk FOREIGN KEY ( kino_idkina )
REFERENCES kino ( idkina )
NOT DEFERRABLE;

ALTER TABLE seans
ADD CONSTRAINT seans_film_fk FOREIGN KEY ( film_idfilmu )
REFERENCES film ( idfilmu )
NOT DEFERRABLE;
```

• SKRYPT DEINSTALUJĄCY

```
-- Generated by Oracle SQL Developer Data Modeler 23.1.0.087.0806
-- at:    2024-01-23 11:59:59 CET
-- site:  Oracle Database 12cR2
-- type:  Oracle Database 12cR2
```

```
DROP TABLE aktor CASCADE CONSTRAINTS;
```

```
DROP TABLE bilet CASCADE CONSTRAINTS;
```

```
DROP TABLE film CASCADE CONSTRAINTS;
```

```
DROP TABLE kino CASCADE CONSTRAINTS;
```

```
DROP TABLE miejsce CASCADE CONSTRAINTS;
```

```
DROP TABLE rola CASCADE CONSTRAINTS;
```

```
DROP TABLE sala CASCADE CONSTRAINTS;
```

```
DROP TABLE seans CASCADE CONSTRAINTS;
```

```
DROP TABLE studiofilm CASCADE CONSTRAINTS;
```

```
DROP TABLE studiofilmowe CASCADE CONSTRAINTS;
```

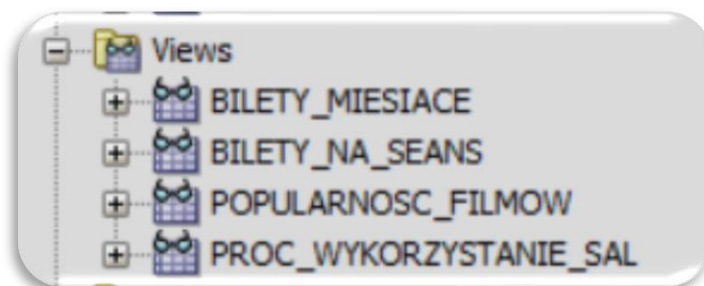
6. INSTALACJA ORAZ SPRAWDZENIE POPRAWNOŚCI DZIAŁANIA BAZY

Instalacja:

- Należy uruchomić na kliencie SKRYPT WDROŻENIOWY INSTALACJĘ „kino_create.ddl”

Sprawdzenie poprawności działania bazy:

- **Wprowadzanie testowych danych:**
 - Należy uruchomić na kliencie skrypt uzupełniający bazę przykładowymi(testowymi) danymi „kino_populate.sql”
- **Przetestowanie perspektyw:**
 - Wprowadź następujące zdania i porównaj z docelowymi wynikami:



1. BILETY_MIESIACE

Zdanie:

```
SELECT * FROM bilety_miesiace  
FETCH FIRST 2 ROWS ONLY;
```

Docelowy wynik:

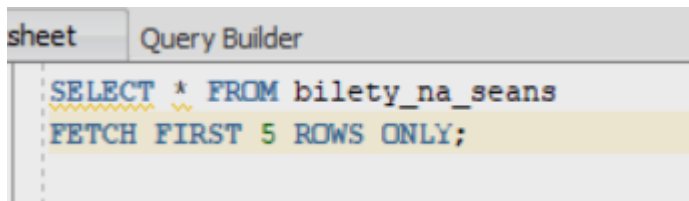
	MIESIAC	LICZBA_BILETOW	DOCHOD_Z_BILETOW	SREDNIA_CENA_BILETU
1	2024-01	5	139	27.8
2	2024-02	5	125.5	25.1

Perspektywa:

```
SELECT  
    TO_CHAR(seansdata, 'YYYY-MM') AS miesiac,  
    COUNT(*) AS liczba_biletow,  
    SUM(cena) AS dochod_z_biletow,  
    AVG(cena) AS srednia_cena_biletu  
FROM bilet  
JOIN seans ON bilet.seans_idseansu = seans.idseansu  
GROUP BY TO_CHAR(seansdata, 'YYYY-MM')  
ORDER BY miesiac;
```

2. BILETY_NA_SEANS

Zdanie:



sheet Query Builder

```
SELECT * FROM bilety_na_seans  
FETCH FIRST 5 ROWS ONLY;
```

Docelowy wynik:



	ID Seansu	Liczba Biletów	Typ Biletu
1	1	2	Normalny
2	2	1	Ulgowy
3	2	1	VIP
4	3	1	Ulgowy
5	6	1	Normalny

Perspektywa:

```
SELECT se.idseansu AS "ID Seansu",  
       COUNT(b.typbiletu) AS "Liczba Biletów",  
       b.typbiletu AS "Typ Biletu"  
FROM seans se  
JOIN bilet b ON se.idseansu = b.seans_idseansu  
GROUP BY se.idseansu, b.typbiletu  
ORDER BY "ID Seansu";
```

3. POPULARNOSC_FILMOW

Zdanie:

```
SELECT * FROM popularnosc_filmow
```

Docelowy wynik:

NAZWAFILMU	LICZBA_SEANSOW	UDZIAL_PROCENTOWY
1 Matrix	2	20
2 Pulp Fiction	2	20
3 Interstellar	2	20
4 Avengers: Endgame	2	20
5 Incepcja	2	20

Perspektywa:

```
SELECT
  f.nazwafilmu,
  COUNT(*) AS liczba_seansow,
  ROUND(COUNT(*) * 100 / (SELECT COUNT(*) FROM
seans), 2) AS udzial_procentowy
FROM film f
JOIN seans s ON f.idfilmu = s.film_idfilmu
GROUP BY f.nazwafilmu
ORDER BY liczba_seansow DESC;
```

4. PROC_WYKORZYSTANIE_SAL

Zdanie:

```
SELECT * FROM proc_wykorzystanie_sal
```

Docelowy wynik:

	NAZWAKINA	IDSALI	PROCENT_WYKORZYSTANIA
1	Kino 2	2	16.67
2	Kino 1	1	15
3	Kino 3	3	5

Perspektywa:

```
SELECT
  k.nazwakina,
  s.idsali,
  ROUND(SUM(se.wolnemiejsca) * 100 /
SUM(s.iloscRzedow*s.miejscwrzedzie), 2) AS
procent_wykorzystania
FROM kino k
JOIN sala s ON k.idkina = s.kino_idkina
JOIN miejsce m ON s.idsali = m.sala_idsali
JOIN seans se ON s.idsali = se.film_idfilmu
LEFT JOIN bilet b ON se.idseansu = b.seans_idseansu
AND m.idmiejsca = b.miejsce_idmiejsca
GROUP BY k.nazwakina, s.idsali
ORDER BY procent_wykorzystania DESC;
```