KINO AURORA

Informacijski sistem neodvisnega kina

DBMS	Firebird 5 Embedded
Uporabniški vmesnik	Flask · HTML · CSS · JavaScript
Model	13 relacijskih tabel · 2 pogleda · 2 proceduri · 5 sprožilcev
Namen	Rezervacije sedežev, spored, blagajna in poslovna poročila

Kratko poročilo za praktični del kontrolne naloge

1. Opis informacijskega problema

Fiktivni neodvisni kino Aurora mora usklajevati filme, žanre, dvorane, sedeže, projekcije, kupce, rezervacije, plačila in zaposlene. Če so podatki ločeni v več preglednicah, lahko pride do dvojne prodaje istega sedeža, prekrivanja terminov in napačnih poročil. Cilj projekta je enoten relacijski sistem, ki zagotavlja referenčno integriteto in omogoča praktično uporabo skozi spletni vmesnik.

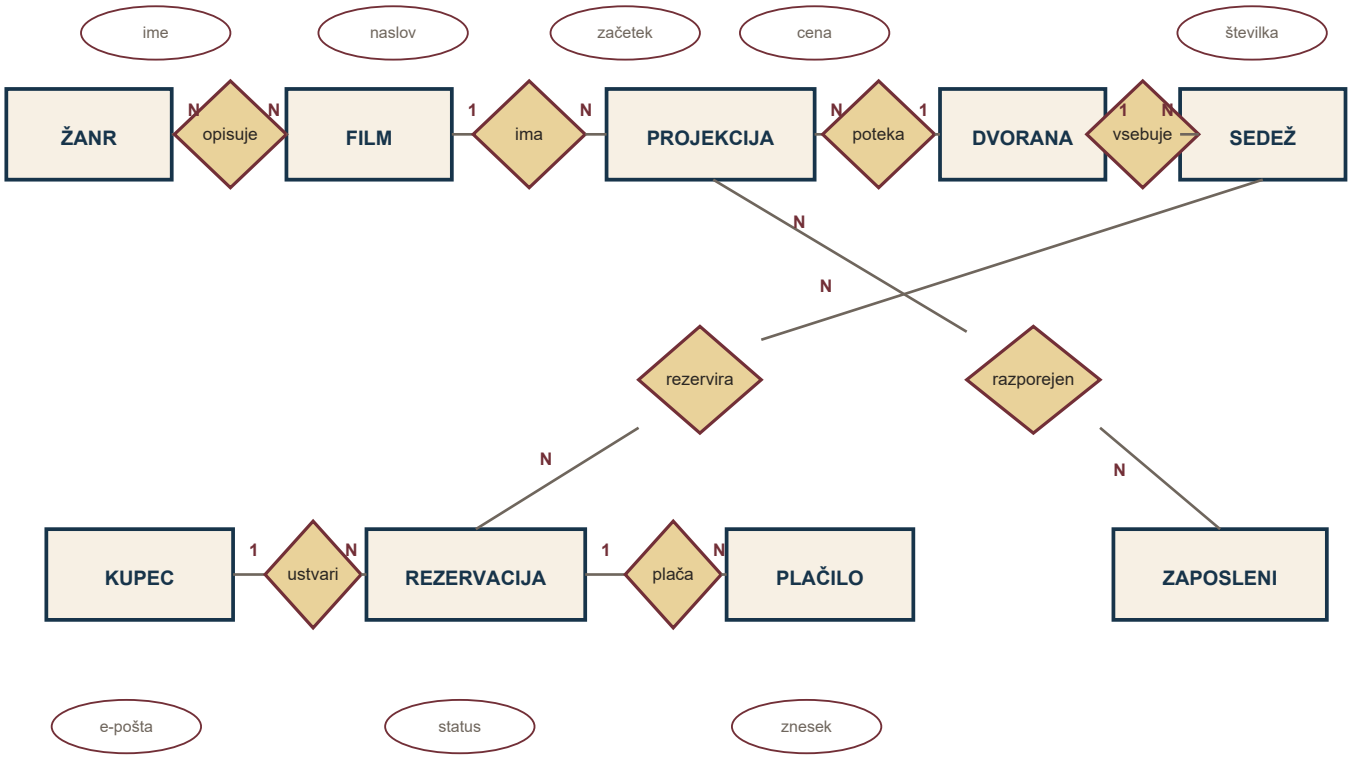
Rešitev

Firebird 5 Embedded hrani vse podatke v lokalni datoteki. Primarni in tuji ključi določajo povezave, omejitev UNIQUE(SCREENING_ID, SEAT_ID) preprečuje dvojno rezervacijo, transakcija pa zagotovi, da se kupec, rezervacija, sedeži in plačilo shranijo skupaj ali se ne shrani nič. Sistem je napolnjen s ponovljivimi vzorčnimi podatki.

Uporabnik	Dejanje	Vpliv na bazo
Obiskovalec	Izbere projekcijo in sedeže	SELECT, INSERT, transakcija, sprožilec
Blagajnik	Poišče ali prekliče rezervacijo	JOIN, CONTAINING, shranjena procedura, DELETE
Vodja kina	Doda film in razporedi termin	INSERT, M:N povezava, preverjanje konflikta
Vodja kina	Pregleda uspešnost	CTE, GROUP BY, podpoizvedba, RANK() OVER

2. Konceptualni model (ER-diagram)

Konceptualni model določi entitete, njihove pomembne atribute, razmerja in kardinalnosti. Razmerja M:N se v logičnem modelu pretvorijo v povezovalne tabele.



Ključna poslovna pravila

Film ima več projekcij, projekcija pa pripada enemu filmu in eni dvorani. Dvorana vsebuje več sedežev. Kupec lahko ustvari več rezervacij. Rezervacija vsebuje enega ali več sedežev; isti sedež je za isto projekcijo lahko aktiven samo v eni rezervaciji. Film ima več žanrov, projekcija pa lahko vključuje več zaposlenih.

3. Logični model relacijske baze

Vsaka entiteta ima primarni ključ. Tuji ključi izvajajo povezave, sestavljeni ključi pa enolično določajo povezovalne zapise.

FILM
PK film_id; title, director, release_year, duration_minutes, status

GENRE
PK genre_id; name UNIQUE

FILM_GENRE
PK/FK film_id + PK/FK genre_id

HALL
PK hall_id; name UNIQUE, is_active

SEAT
PK seat_id; FK hall_id; row_label, seat_number, seat_type

SCREENING
PK screening_id; FK film_id, FK hall_id; start_time, base_price

CUSTOMER
PK customer_id; first_name, last_name, email UNIQUE

BOOKING
PK booking_id; FK customer_id, FK screening_id; code, status, total

BOOKING_SEAT
PK/FK booking_id + PK/FK seat_id; FK screening_id; price_paid

PAYMENT
PK payment_id; FK booking_id; amount, method, status, paid_at

EMPLOYEE
PK employee_id; name, role, email

SCREENING_EMPLOYEE
PK/FK screening_id + PK/FK employee_id; duty

AUDIT_LOG
PK audit_id; event_time, entity, action, details

4. Primer uporabe poizvedbe

Poročilo o uspešnosti filmov najprej v skupnem tabelnem izrazu (CTE) združi projekcije in prodane sedeže. Zatem okenska funkcija RANK razvrsti filme po prihodku, ne da bi izgubila posamezne vrstice rezultata.

```
WITH FILM_STATS AS (
  SELECT F.FILM_ID, F.TITLE,
         COUNT(DISTINCT SC.SCREENING_ID) AS SCREENINGS,
         COUNT(BS.SEAT_ID) AS SOLD_SEATS,
         COALESCE(SUM(BS.PRICE_PAID), 0) AS REVENUE
  FROM FILM F
  LEFT JOIN SCREENING SC ON SC.FILM_ID = F.FILM_ID
  LEFT JOIN BOOKING_SEAT BS ON BS.SCREENING_ID = SC.SCREENING_ID
  GROUP BY F.FILM_ID, F.TITLE
)
SELECT RANK() OVER (ORDER BY REVENUE DESC) AS RANKING,
       TITLE, SCREENINGS, SOLD_SEATS, REVENUE
FROM FILM_STATS
ORDER BY RANKING, TITLE;
```

Druge uporabljene možnosti SQL

Aplikacija uporablja INSERT ... RETURNING za pridobitev novega ključa, CHECK in DOMAIN za veljavne vrednosti, poglede V_SCREENING_OVERVIEW in V_BOOKING_OVERVIEW, sprožilce za izračun skupnega zneska in revizijsko sled ter proceduro SP_CANCEL_BOOKING za varen preklic rezervacije.

Predstavitveni način

Preklop na vrhu spletnega vmesnika odpre konzolo, ki v realnem času pokaže SQL, dejanske vezane parametre, čas izvajanja in število spremenjenih vrstic. Tako je pri praktičnem prikazu jasno vidna povezava med uporabniškim dejanjem in operacijo v bazi.

5. Zagon in praktična predstavitev

Projekt je namenjen lokalnemu zagonu brez Dockerja ali strežniške namestitve. Firebird Embedded teče v istem procesu kot aplikacija, datoteka baze pa je shranjena v mapi data.

Postopek zagona

1. Ob prvi uporabi zaženemo setup.bat. Skripta ustvari lokalno mapo .venv, namesti odvisnosti in napolni bazo z vzorčnimi podatki.
2. Za vsako naslednjo predstavitev zaženemo run.bat. Odpre se naslov http://127.0.0.1:5000.
3. Po potrebi reset_demo.bat obnovi začetne podatke.

Predlagan prikaz pri pouku

Najprej pokažemo program in rezerviramo dva sedeža. Nato vključimo predstavitveni način ter razložimo transakcijo, INSERT ... RETURNING in sprožilce. V blagajni prekličemo rezervacijo s shranjeno proceduro. Na strani O projektu pokažemo ER-diagram in logični model, predstavitev pa zaključimo s poročilom CTE + RANK v SQL laboratoriju.

Viri

– Gimnazija Vič: Inf 4 - Praktični del kontrolne naloge (20 %) - baza.
– Učno gradivo: Podatkovne baze (konceptualni ER-model, kardinalnost in pretvorba v relacijski model).
– Firebird Project: Firebird 5 Language Reference in dokumentacija gonilnika firebird-driver.