System zarządzania konferencjami

Spis treści

Opis projektu

System służy do zarządzania, śledzenia i raportowania konferencji, zapewniając integralność, historię zmian i gotowe podsumowania okresowe. Składa się z następujących elementów:

- Tabele
  - **KONFERENCJE** – informacje o wydarzeniach (nazwa, data, miejsce, opis)
  - **PRELEGENCI** – dane prelegentów (imię, nazwisko, biografia)
  - **SALE** – sale konferencyjne (nazwa, lokalizacja, pojemność)
  - **SESJE** – poszczególne wystąpienia w ramach konferencji (tytuł, data, opis), powiązane z konferencją i salą
  - **UCZESTNICY** – uczestnicy (imię, nazwisko, e-mail, PESEL)
  - **REJESTRACJE** – zapisy uczestników na konferencje
  - **SESJE_PRELEGENCI** – relacja N:M między sesjami a prelegentami
- Mechanizmy wspierające
  - **Archiwizacja**: każda usunięta krotka trafia do odpowiadającej tabeli *_ARCH z datą usunięcia
  - **Audyt**: wszystkie zmiany DML logowane są w tabeli **AUDIT_LOG** przez triggery i pakiet PKG_AUDIT
  - **Walidacja PESEL**: trigger i pakiet PKG_EXCEPTIONS blokują wstawienie nieprawidłowego numeru
  - **Raporty**: pakiet PKG_REPORTS generuje zestawienia miesięczne, kwartalne i roczne, zapisane w tabelach PODSUMOWANIA_*
- Integracja z Pythonem
  - Skrypt loader.py ładuje dane z plików CSV do tabel

Cel projektu

Celem jest stworzenie kompletnego rozwiązania do zarządzania konferencjami, obejmującego:

- **Rejestrację obiektów** biznesowych: konferencji, sal, prelegentów, uczestników, sesji i rejestracji.
- **Logikę biznesową** i utrzymanie spójności danych w bazie Oracle (procedury CRUD, archiwizacja, audyt, walidacje).
- **Automatyczny import danych** z plików CSV przy pomocy skryptu Python + Instant Client.
- **Raportowanie** miesięczne, kwartalne i roczne z liczbą sesji i unikalnych uczestników.

Architektura systemu

1. Baza danych

- Tabele główne i archiwalne
- Relacje FK zapewniające spójność referencyjną

2. Logika w bazie (SQL)

- Pakiety PKG_* z procedurami CRUD
- Triggery archiwizacyjne (BEFORE DELETE) i audytowe (AFTER INSERT/UPDATE/DELETE)
- Pakiet PKG_EXCEPTIONS do walidacji PESEL
- Pakiet PKG_REPORTS do generowania podsumowań

3. Warstwa ETL (Python + python-oracledb)

- loader.py do masowego importu plików CSV/JSON
- Walidacja e-mail, obsługa błędów, archiwizacja plików po wczytaniu

4. Interfejs użytkownika

- SQL Developer / SQL*Plus do wywoływania procedur i przeglądania wyników
- (Opcjonalnie) harmonogramowanie w cron lub DBMS_SCHEDULER

Schemat bazy danych

KONFERENCJE

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_konferencji	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
nazwa	VARCHAR2(200 BYTE)			Nie	NOT NULL
data_rozpoczecia	DATE			Nie	NOT NULL
miejsce	VARCHAR2(200 BYTE)			Tak	
opis	CLOB			Tak	

PRELEGENCI

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_prelegenta	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
imie	VARCHAR2(100 BYTE)			Nie	NOT NULL
nazwisko	VARCHAR2(100 BYTE)			Nie	NOT NULL
biografia	CLOB			Tak	

SALE

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_sali	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
nazwa	VARCHAR2(100 BYTE)			Nie	NOT NULL
lokalizacja	VARCHAR2(200 BYTE)			Tak	
pojemnosc	NUMBER			Tak	

SESJE

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_sesji	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
konferencja_id	NUMBER		→ KONFERENCJE(id_konferencji) (CASCADE)	Nie	
tytul	VARCHAR2(200 BYTE)			Nie	NOT NULL
opis	CLOB			Tak	
data_sesji	TIMESTAMP			Nie	NOT NULL
sala_id	NUMBER		→ SALE(id_sali)	Tak	

UCZESTNICY

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_uczestnika	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
imie	VARCHAR2(100 BYTE)			Nie	NOT NULL
nazwisko	VARCHAR2(100 BYTE)			Nie	NOT NULL
email	VARCHAR2(200 BYTE)			Nie	NOT NULL, UNIQUE
pesel	VARCHAR2(11 BYTE)			Tak	walidowany trig.

REJESTRACJE

Kolumna	Typ	PK	FK	Nullable	Uwagi
id_rejestracji	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
konferencja_id	NUMBER		→ KONFERENCJE(id_konferencji)	Nie	
uczestnik_id	NUMBER		→ UCZESTNICY(id_uczestnika)	Nie	
data_rejestracji	DATE			Tak	DEFAULT SYSDATE

SESJE_PRELEGENCI

Kolumna	Typ	PK	FK	Nullable	Uwagi
id	NUMBER GENERATED ALWAYS AS IDENTITY	✓		Nie	
sesja_id	NUMBER		→ SESJE(id_sesji) (ON DELETE CASCADE)	Nie	
prelegent_id	NUMBER		→ PRELEGENCI(id_prelegenta)	Nie	

Relacje między tabelami

1. KONFERENCJE → SESJE

- **PK**: KONFERENCJE.id_konferencji
- **FK**: SESJE.konferencja_id → odniesienie do konferencji, **ON DELETE CASCADE**
- Relacja: 1 konferencja może mieć wiele sesji (1 : N).

2. SALE → SESJE

- **PK**: SALE.id_sali
- **FK**: SESJE.sala_id → odniesienie do sali
- Relacja: 1 sala może gościć wiele sesji (1 : N); sesja może nie mieć przypisanej sali (NULL).

3. UCZESTNICY → REJESTRACJE

- **PK**: UCZESTNICY.id_uczestnika
- **FK**: REJESTRACJE.uczestnik_id → odniesienie do uczestnika
- Relacja: 1 uczestnik może dokonać wielu rejestracji (1 : N).

4. KONFERENCJE → REJESTRACJE

- **PK**: KONFERENCJE.id_konferencji
- **FK**: REJESTRACJE.konferencja_id → odniesienie do konferencji
- Relacja: 1 konferencja może mieć wielu uczestników (poprzez rejestracje) (1 : N).

5. **SESJE ↔ PRELEGENCI** przez SESJE_PRELEGENCI

- **Tabela SESJE_PRELEGENCI** to tabela pośrednia z dwoma FK:
  - sesja_id → SESJE.id_sesji
  - prelegent_id → PRELEGENCI.id_prelegenta
- Relacja: wiele prelegentów może występować na wielu sesjach (N :M).

Główne moduły

Pakiet	Zawartość	Zadanie
PKG_CONFERENCES	add_conference, update_conference, delete_conference	CRUD dla tabeli konferencje
PKG_SPEAKERS	procedury add_speaker, update_speaker, delete_speaker	CRUD dla tabeli prelegenci
PKG_ROOMS	procedury add_room, update_room, delete_room	CRUD dla tabeli sale
PKG_SESSIONS	procedury add_session, update_session, delete_session	CRUD dla tabeli sesje
PKG_PARTICIPANTS	procedury add_participant, update_participant, delete_participant	CRUD dla tabeli uczestnicy + walidacja PESEL (trigger)
PKG_REGISTRATIONS	register_participant, cancel_registration	CRUD dla tabeli rejestracje
PKG_SESSION_SPEAKERS	assign_speaker_to_session, unassign_speaker	CRUD dla tabeli łącznikowej sesje_prelegenci
PKG_AUDIT	log_it(p_table, p_op, p_rec_id)	Wstawia wpis do audit_log przy każdej operacji INSERT/UPDATE/DELETE via trigger'y
PKG_EXCEPTIONS	check_pesel(p_pesel)	Walidacja formatu i sumy kontrolnej PESEL, podnosi wyjątek -20001
PKG_REPORTS	gen_monthly, gen_quarterly, gen_yearly	Zlicza unikalnych uczestników i sesje w zadanym okresie i zapisuje do tabel PODSUMOWANIA_*

Triggery

- **Archiwizacyjne** (_BEFORE_DELETE):

Dla każdej głównej tabeli (np. uczestnicy, prelegenci, sale, sesje, sesje_prelegenci, rejestracje, konferencje) istnieje trigger:

```
CREATE OR REPLACE TRIGGER trg_<tabela>_before_delete
    BEFORE DELETE ON <tabela>
    FOR EACH ROW
BEGIN
    INSERT INTO <tabela>_arch (...) VALUES (:OLD...., SYSTIMESTAMP);
END;
```

Dzięki temu usunięte wiersze trafiają do <tabela>_arch z polem deleted_at.

- **Audytowe** (_AFTER INSERT OR UPDATE OR DELETE):

Dla najważniejszych tabel (KONFERENCJE, SESJE itd.):

```
CREATE OR REPLACE TRIGGER trg_<tabela>_audit
    AFTER INSERT OR UPDATE OR DELETE ON <tabela>
    FOR EACH ROW
BEGIN
    IF INSERTING THEN
        pkg_audit.log_it('TABELA','INSERT', :NEW.id);
    ELSIF UPDATING THEN
        pkg_audit.log_it('TABELA','UPDATE', :NEW.id);
    ELSIF DELETING THEN
        pkg_audit.log_it('TABELA','DELETE', :OLD.id);
    END IF;
END;
```

Wpisy lądują w audit_log z user_name i ts_operacji.

- **Walidacja PESEL** (TRG_UCZESTNICY_VALIDATE):

```
CREATE OR REPLACE TRIGGER trg_uczestnicy_validate
    BEFORE INSERT OR UPDATE ON uczestnicy
    FOR EACH ROW
BEGIN
    IF :NEW.pesel IS NOT NULL THEN
        pkg_exceptions.check_pesel(:NEW.pesel);
    END IF;
END;
```

Tabele podsumowań i raporty

Tabele:

1. podsumowania_mies (konferencja_id, rok, miesiac, liczba_uczestnikow, liczba_sesji, wygenerowano)
2. podsumowania_kwarta (... kwartal ...)
3. podsumowania_rok (... rok ...)

Raporty (pkg_reports):

1. gen_monthly(p_konf_id, p_rok, p_miesiac)
2. gen_quarterly(p_konf_id, p_rok, p_kwartal)
3. gen_yearly(p_konf_id, p_rok)

Procedury obliczają zakres dat, zliczają rekordy z tabel rejestracje + sesje, a następnie MERGE do tabel podsumowań.

Skrypt ETL – loader.py

- **Środowisko**: Python 3 + python-oracledb + Oracle Instant Client
- **Struktura katalogów**:

/project

/incoming - nowe pliki CSV

/archive - przetworzone pliki z timestampem

/logs - loader.log

/loader.py - skrypt ETL

/config.py - dane logowania

/test_connect.py – skrypt sprawdzający połączenie z bazą danych

- **Proces**:
  1) Skrypt czyta pliki w ustalonej kolejności
  2) Dla każdego pliku uruchamia process_*(), który wstawia dane do odpowiedniej tabeli
  3) Błędy logowane w *loader.log*, pliki po wczytaniu przenoszone do *archive/*.