BI - Analiza ocen

Analityka i eksploracja danych

Autor:	Piotr Szczypior
Kontakt:	264468@student.pwr.edu.pl
Data:	11 stycznia 2026
Prowadzący:	Prof. dr hab. inż. Henryk Maciejewski

1 Wstęp

1.1 Cel i zakres projektu

Niniejsze sprawozdanie dokumentuje proces projektowania i implementacji systemu Business Intelligence opartego o technologie Microsoft SQL Server Integration Services (SSIS) oraz SQL Server Analysis Services (SSAS). Głównym celem projektu było stworzenie kostki analitycznej umożliwiającej wielowymiarową analizę danych dotyczących ocen studentów z kierunku na Wydziale Elektronicznym Politechniki Wrocławskiej.

System został zaprojektowany w celu umożliwienia analizy wyników akademickich w różnych przekrojach, takich jak studenci, prowadzący zajęcia, rodzaje egzaminów oraz kursy. Implementacja obejmuje kompletny proces ETL (Extract, Transform, Load) realizowany w środowisku SSIS oraz wielowymiarową kostkę OLAP (Online Analytical Processing) zbudowaną w technologii SSAS.

Projekt pozwala na efektywną agregację i eksplorację danych akademickich, wspierając procesy podejmowania decyzji oraz identyfikację trendów w wynikach nauczania. Zastosowanie technologii Microsoft zapewnia skalowalność rozwiązania oraz możliwość integracji z innymi systemami uczelnianymi.

1.2 Źródła danych

Dane wejściowe do systemu zostały przygotowane w formacie CSV i obejmują następujące zbiory:

- `course_group.csv` – dane grup kursów
- `grades.csv` – dane z ocenami
- `students.csv` – dane o studentach
- `teacher_title.csv` – dane z tytułami nauczycieli
- `teachers.csv` – dane nauczycieli

2 ETL

W ramach realizacji systemu Business Intelligence opracowano proces ETL (Extract, Transform, Load), odpowiedzialny za pozyskanie, przetworzenie oraz załadowanie danych do hurtowni. Dane źródłowe w formacie CSV zostały poddane wieloetapowej obróbce, obejmującej ekstrakcję z plików, walidację oraz transformację do struktury zgodnej z modelem analitycznym.

Implementacja procesu ETL obejmowała następujące etapy:

1. **Stworzenie schematu bazy danych** – utworzenie struktury relacyjnej bazy danych w środowisku Microsoft SQL Server.
2. **Załadowanie danych źródłowych** – wykorzystanie narzędzia SQL Server Integration Services (SSIS) do automatycznego importu danych z plików CSV do tabel pośrednich (staging tables) w bazie danych.
3. **Czyszczenie i walidacja danych** – identyfikacja oraz usunięcie niespójności, duplikatów i wartości odstających w danych źródłowych, a także standaryzacja formatów i uzupełnienie brakujących wartości.
4. **Transformacja i załadowanie do hurtowni** – przekształcenie danych z tabel pośrednich do docelowego modelu wymiarowego, obejmującego tabele wymiarów i tabele faktów, stanowiącego podstawę dla kostki analitycznej SSAS.

Wszystkie operacje ETL zostały zautomatyzowane w postaci pakietów SSIS, w celu zapewnia powtarzalność procesu oraz możliwość regularnej aktualizacji danych w systemie.

2.1 Stworzenie tabel Staging

2.2 Czyszczenie danych

Tabela Teacher

Kolumna `gender` w pliku `teachers.csv` przyjmowała wartości 0 i 1. W celu ujednolicenia reprezentacji danych dokonano konwersji na typ znakowy oraz mapowania wartości na oznaczenia M (Man) i F (Female).

```
ALTER TABLE stg_teachers
ALTER COLUMN gender VARCHAR(1);
```

Następnie przeprowadzono aktualizację wartości w kolumnie `gender` według ustalonego schematu mapowania:

```
UPDATE stg_teachers
SET gender = CASE
  WHEN gender = '1' THEN 'M'
  WHEN gender = '2' THEN 'K'
  ELSE 'U'
END;
```

W celu zachowania spójności danych uzupełniono brakujące wartości w kolumnie `institute`, przypisując oznaczenie `UNK` (unknown) dla rekordów z wartością NULL lub pustym ciągiem znaków:

```
UPDATE stg_teachers
SET institute = 'UNK'
WHERE institute IS NULL OR institute = '';
```

Analogicznie, dla kolumny `faculty` zastąpiono wartości NULL wartością domyślną -1, oznaczającą brak przypisania do wydziału:

```
UPDATE stg_teachers
SET faculty = -1
WHERE faculty IS NULL;
```

Tabela Grades

Usunięto rekordy zawierające nieprawidłowe wartości ocen (0 i 1), które nie mieszczą się w standardowej skali oceniania:

```
DELETE FROM stg_grades
WHERE grade IN (0, 1);
```

Dla kolumny `exam` uzupełniono brakujące wartości domyślnym oznaczeniem 'T' (test):

```
UPDATE stg_grades
SET exam = 'T'
WHERE exam IS NULL OR exam = '';
```

W celu standaryzacji nazw kursów dokonano konwersji na wielkie litery:

```
UPDATE stg_grades
SET course = UPPER(course);
```

Tabela Students

Uzupełniono brakujące wartości w kolumnie `sub_spec2` oznaczeniem 'NA' (not applicable):

```
UPDATE stg_students
SET sub_spec2 = 'NA'
WHERE sub_spec2 IS NULL OR sub_spec2 = '';
```

Tabela Course Group

Standaryzacja nazw kursów poprzez konwersję na wielkie litery:

```
UPDATE stg_course_group
SET course = UPPER(course);
```

2.3 Standaryzacja relacji między tabelami

W celu zapewnienia integralności referencyjnej wprowadzono rekord tytułu domyślnego dla nauczycieli bez przypisanego tytułu naukowego:

```
IF NOT EXISTS (SELECT 1 FROM stg_teacher_title WHERE title_id = 0)
BEGIN
    INSERT INTO stg_teacher_title (title_id, title_long, title)
    VALUES (0, 'Unknown Title', 'UNK');
END
```

Następnie uzupełniono tabelę `stg_teachers` o brakujące rekordy nauczycieli występujących w tabeli ocen, ale nieobecnych w tabeli nauczycieli. Rekordom tym przypisano wartości domyślne:

```
INSERT INTO stg_teachers (teacher_id, gender, faculty, institute, title_id)
SELECT g.teacher_id, 'U', NULL, 'UNK', 0
FROM stg_grades AS g
LEFT JOIN stg_teachers AS t ON g.teacher_id = t.teacher_id
WHERE t.teacher_id IS NULL
GROUP BY g.teacher_id;
```

Analogicznie uzupełniono tabelę `stg_course_group` o kursy występujące w ocenach, ale nieobecne w tabeli grup kursów. Przypisano im domyślną grupę o identyfikatorze 5:

```
INSERT INTO stg_course_group (course, course_group)
SELECT g.course, 5
FROM stg_grades AS g
LEFT JOIN stg_course_group AS cg ON g.course = cg.course
WHERE cg.course IS NULL
GROUP BY g.course;
```

2.4 Wzbogacenie danych o kolumny analityczne

W celu umożliwienia zaawansowanych analiz rozszerzono strukturę tabel o kolumny pochodne zawierające przetworzone wartości analityczne.

Tabela Grades

Dodano trzy nowe kolumny umożliwiające analizę w kontekście roku studiów oraz statusu zaliczenia:

```
ALTER TABLE stg_grades
ADD [semester_type] varchar(1),
    [study_year] int,
    [pass] bit;
```

Wypełniono kolumnę `semester_type` oznaczeniem typu semestru (W – zimowy, S – letni) na podstawie parzystości numeru semestru:

```
UPDATE stg_grades
SET semester_type = CASE
      WHEN semester % 2 = 1 THEN 'W'
      ELSE 'S'
END;
```

Obliczono rok studiów na podstawie numeru semestru:

```
UPDATE stg_grades
SET study_year = (semester + 1) / 2;
```

Dodano kolumnę binarną `pass` wskazującą status zaliczenia przedmiotu (1 – zaliczony, 0 – niezaliczony):

```
UPDATE stg_grades
SET pass = CASE
    WHEN grade >= 3.0 THEN 1
    WHEN grade = 2.0 THEN 0
END;
```

Tabela Course Group

Rozszerzono tabelę o kolumnę `course_type` identyfikującą typ zajęć na podstawie ostatniego znaku nazwy kursu:

```
ALTER TABLE stg_course_group
ADD course_type varchar(1);
```

Wypełniono kolumnę `course_type` według schematu mapowania typów zajęć (W – wykład, L – laboratorium, C – ćwiczenia, P – projekt, S – seminarium, E – egzamin, U – nieznany):

```
UPDATE stg_course_group
SET course_type = CASE
      WHEN course IS NULL THEN 'U'
      WHEN UPPER(RIGHT(course, 1)) NOT IN ('W', 'L', 'C', 'P', 'S', 'E')
           THEN 'U'
      ELSE UPPER(RIGHT(course, 1))
END;
```

2.5 Tworzenie tabel wymiarowych

Tabela Semester

Została utworzona nowa tabela wymiarowa `dim_semester` agregująca informacje o semestrach:

```
DROP TABLE IF EXISTS dim_semester;

CREATE TABLE [dim_semester] (
     [semester_id] bigint identity(1,1) primary key,
     [semester] int,
     [year] int,
     [semester_type] varchar(1),
     [study_year] int
);
```

Załadowano dane do tabeli wymiarowej na podstawie unikalnych kombinacji wartości z tabeli `stg_grades`:

```
INSERT INTO dim_semester (semester, year, semester_type, study_year)
SELECT DISTINCT
    semester,
    year,
    semester_type,
    study_year
FROM stg_grades
ORDER BY year, semester;
```

Tabela Teacher

Pozostałe tabele wymiarów zostały utworzone na podstawie tabel pośrednich. Etap ten został przeprowadzony w celu standaryzacji nazw, zapewnienia odpowiedniej struktury tabel oraz utworzenia kluczy.
Poniżej znajduje się przykładowy kod SQL dla tabeli wymiarowej `dim_teachers`.

```
DROP TABLE IF EXISTS dim_teachers;

CREATE TABLE dim_teachers (
    teacher_id BIGINT NOT NULL PRIMARY KEY,
    gender VARCHAR(1),
    faculty INT,
    institute VARCHAR(5),
    title_id INT
);
```

Załadowano dane do tabeli wymiarowej na podstawie tabeli `stg_teachers`:

```
INSERT INTO dim_teachers (teacher_id, gender, faculty, institute, title_id)
SELECT teacher_id, gender, faculty, institute, title_id
FROM stg_teachers;
```

Podobny mechanizm został zastosowany dla tabel `stg_students`, `stg_course_group` oraz `stg_teacher_title`.

2.6 Tworzenie tabeli faktów

Utworzono tabelę faktów `fact_grades` będącą centralnym elementem modelu wymiarowego, przechowującą szczegółowe informacje o wystawionych ocenach wraz z kluczami obcymi do odpowiednich wymiarów:

```
DROP TABLE IF EXISTS fact_grades;

CREATE TABLE [fact_grades] (
      [grade_id] bigint identity(1,1) primary key,
      [student_id] bigint,
      [teacher_id] bigint,
      [semester_id] bigint,
      [course] varchar(50),
      [grade] decimal(3,1),
      [exam] varchar(1),
      [pass] bit,
);
```

Załadowano dane do tabeli faktów na podstawie połączenia tabeli `stg_grades` z wymiarem `dim_semester`:

```
INSERT INTO fact_grades
    (student_id, teacher_id, semester_id, course, grade, exam, pass)
SELECT
    g.student_id,
    g.teacher_id,
    s.semester_id,
    g.course,
    g.grade,
    g.exam,
    g.pass
FROM stg_grades AS g
JOIN dim_semester AS s
ON g.semester = s.semester AND g.year = s.year;
```

2.7 Dodatkowe tabele wymiarów

Tabela Teacher Workload

Utworzono dodatkowo nową tabelę wymiarową `dim_teacher_workload` agregującą metryki obciążenia nauczycieli w poszczególnych semestrach, obejmujące liczbę wystawionych ocen, prowadzonych kursów, ocenianych studentów, liczbę egzaminów i testów oraz współczynnik zdawalności:

```
DROP TABLE IF EXISTS dim_teacher_workload;

CREATE TABLE [dim_teacher_workload] (
      [teacher_id] bigint,
      [semester_id] bigint,
      [grade_count] int,
      [course_count] int,
      [student_count] int,
      [exam_count] int,
      [test_count] int,
      [pass_rate] DECIMAL(5,2)

      CONSTRAINT PK_teacher_workload PRIMARY KEY (teacher_id, semester_id)
);
```

Załadowano dane do tabeli wymiarowej na podstawie agregacji danych z tabeli faktów `fact_grades`:

```
INSERT INTO dim_teacher_workload
SELECT
  teacher_id,
  semester_id,
  COUNT(*) as grade_count,
  COUNT(DISTINCT course) as course_count,
  COUNT(DISTINCT student_id) as student_count,
  SUM(CASE WHEN exam = 'E' THEN 1 ELSE 0 END) as exam_count,
```

```
  SUM(CASE WHEN exam = 'T' THEN 1 ELSE 0 END) as test_count,
  CAST(SUM(CAST(pass as int)) * 100.0 / COUNT(*) AS DECIMAL(5,2)) as pass_rate
FROM fact_grades
GROUP BY teacher_id, semester_id
ORDER BY teacher_id, semester_id;
```

Tabela Student Workload

Podobnie została utworzona tabela `Student Workload` agregująca metryki obciążenia studentów w poszczególnych semestrach.

```
DROP TABLE IF EXISTS [dim_student_workload];

CREATE TABLE [dim_student_workload] (
      [student_id] bigint,
      [semester_id] bigint,
      [grade_count] int,
      [course_count] int,
      [teacher_count] int,
      [exam_count] int,
      [test_count] int,
      [pass_rate] DECIMAL(5,2)

      CONSTRAINT PK_student_workload PRIMARY KEY (student_id, semester_id)
);
```

Załadowano dane do tabeli wymiarowej na podstawie agregacji danych z tabeli faktów fact grades:

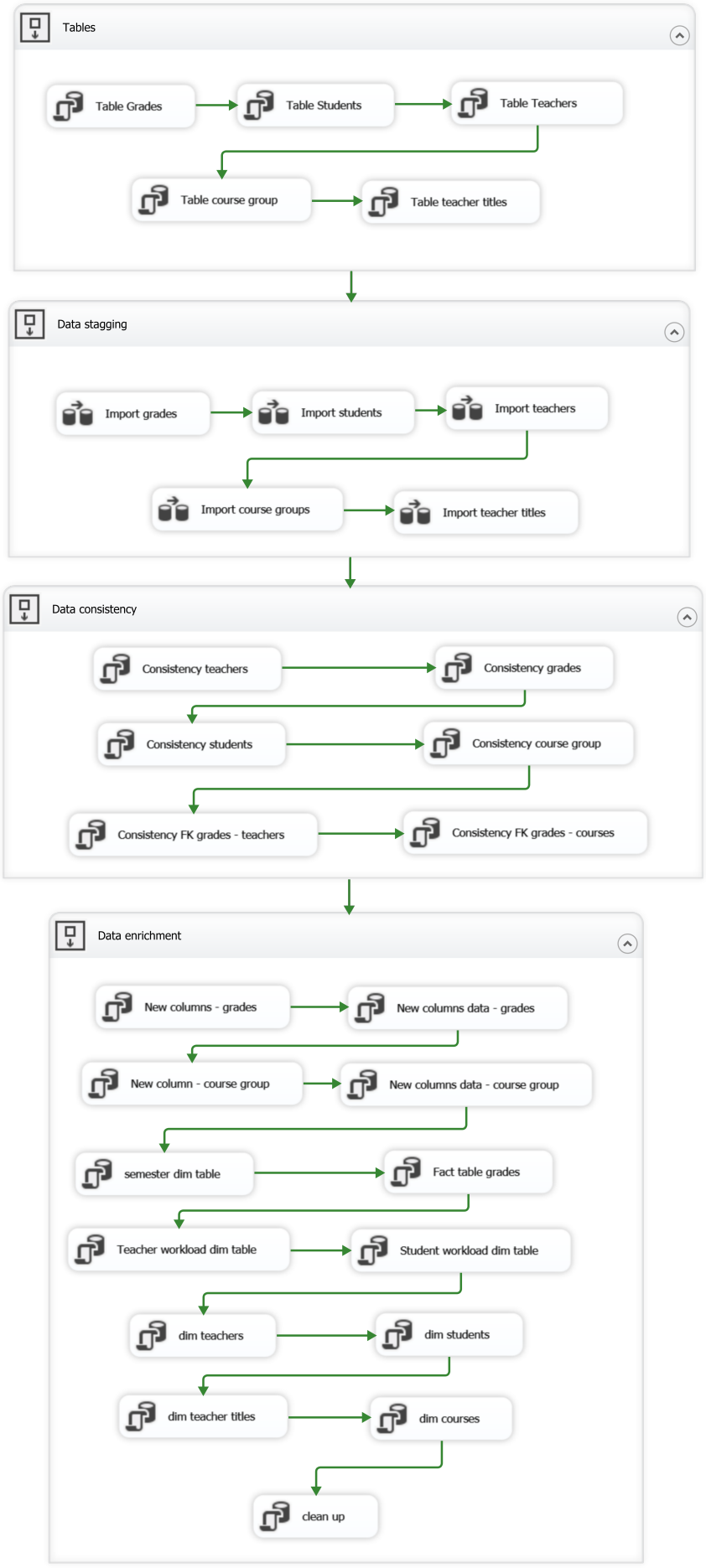
```
INSERT INTO dim_student_workload
SELECT
    student_id,
    semester_id,
    COUNT(*) as grade_count,
    COUNT(DISTINCT course) as course_count,
    COUNT(DISTINCT teacher_id) as teacher_count,
    SUM(CASE WHEN exam = 'E' THEN 1 ELSE 0 END) as exam_count,
    SUM(CASE WHEN exam = 'T' THEN 1 ELSE 0 END) as test_count,
    CAST(SUM(CAST(pass as int)) * 100.0 / COUNT(*) AS DECIMAL(5,2)) as pass_rate
FROM fact_grades
GROUP BY student_id, semester_id
ORDER BY student_id, semester_id;
```

2.8 Usunięcie tabel pośrednich

Na zakończenie procesu ETL w narzędziu SSIS przeprowadzono czyszczenie bazy danych. Tabele pośrednie (staging) zostały usunięte, pozostawiając jedynie tabele wymiarowe oraz tabelę faktów stanowiące finalną strukturę hurtowni danych.
Przykładowe usunięcie tabeli staging:

```
DROP TABLE IF EXISTS stg_students;
```

2.9 Diagram procesu ETL



Rysunek 1: Diagram przepływu danych w procesie ETL zrealizowanym w narzędziu SSIS

Literatura