

Powtórka po wakacjach — zadanie diagnostyczne

Czas: 2 × 45 min

praca indywidualna (bez pomocy kolegów — to diagnoza, nie sprawdzian)

Cel: Sprawdzenie, co pamiętasz z pisania zapytań SELECT z roku 1, i przygotowanie własnej mapy zaległości przed nowymi tematami.

Po zajęciach potrafisz:

- pisze zapytania SELECT z warunkami, sortowaniem i funkcjami agregującymi
- łączy dwie i trzy tabele złączeniami wewnętrznymi i zewnętrznymi
- stosuje podzapytania skalarne, skorelowane i NOT EXISTS
- porównuje wynik z kontrolnym, odnajduje przyczynę różnicy i planuje powtórkę

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Przygotowanie przed zajęciami

- Przejrzyj notatki z roku 1 o złączeniach (JOIN) i podzapytaniach — bez uczenia się na pamięć.

Bezpieczeństwo

- W tym zadaniu wykonujesz tylko zapytania SELECT — nie zmieniaj danych bazy sklep (nie używaj UPDATE, DELETE, DROP).
- Jeśli przypadkiem zmienisz dane, zaimportuj `sklep.sql` ponownie, zanim porównasz wyniki z kontrolnymi.

Przebieg zadania

1 Przygotowanie

1. Uruchom w panelu XAMPP moduły Apache i MySQL. Otwórz phpMyAdmin albo konsolę `mysql -u root`.
2. Wykonaj `USE sklep;` i `SHOW TABLES;` — powinno być 6 tabel. Wykonaj `DESCRIBE` dla tabel `zamowienia` i `pozycje_zamowienia`.
3. Utwórz plik `r2_01_nazwisko.sql`. Każde zapytanie poprzedź komentarzem z numerem, np. `-- Z1`.

2 Część A — jedna tabela (Z1–Z6)

Lekcje 1.11–1.13: SELECT, WHERE, ORDER BY, agregaty, GROUP BY.

1. Z1. Wypisz nazwę, kategorię i cenę produktów z kategorii Monitory lub Akcesoria, które kosztują mniej niż 1000 zł. Posortuj od najdroższego. (Użyj IN.)
2. Z2. Wypisz imię, nazwisko i miasto klientów, których nazwisko kończy się na „ska”. Posortuj alfabetycznie po nazwisku.
3. Z3. Policz zamówienia złożone od 1 marca do 30 czerwca 2026 włącznie. Uwaga: kolumna data_zamowienia ma typ DATETIME.
4. Z4. Wypisz klientów, którzy nie podali miasta.
5. Z5. Dla każdej kategorii podaj liczbę produktów i średnią cenę zaokrągloną do 2 miejsc. Produkty bez kategorii pokaż jako „(bez kategorii)”.
6. Z6. Podaj statusy zamówień i liczbę zamówień w każdym statusie — tylko te statusy, które mają co najmniej 3 zamówienia.

3 Część B — złączenia (Z7–Z12)

Lekcja 1.14: INNER JOIN, LEFT JOIN, SELF JOIN.

1. Z7. Wypisz numer i datę każdego zamówienia oraz imię i nazwisko klienta, posortowane według daty.
2. Z8. Oblicz wartość każdego zamówienia (suma ilosc × cena_jedn). Posortuj malejąco według wartości.
3. Z9. Wypisz klientów, którzy nie złożyli żadnego zamówienia (LEFT JOIN).
4. Z10. Wypisz produkty, których nikt nigdy nie zamówił (NOT EXISTS albo NOT IN).
5. Z11. Wypisz produkty droższe od średniej ceny wszystkich produktów (podzapytanie).
6. Z12. Wypisz każdego pracownika (imię i nazwisko w jednej kolumnie), jego stanowisko i przełożonego. Dyrektor też ma się pojawić.

4 Część C — raporty (Z13–Z15)

Lekcje 1.13–1.15: łączenie wszystkiego.

1. Z13. W każdej kategorii znajdź najdroższy produkt (podzapytanie skorelowane). Wypisz kategorię, nazwę i cenę.
2. Z14. Podaj 3 klientów o największej łącznej wartości zamówień, pomijając zamówienia anulowane.
3. Z15. Podaj miasta, w których łączna wartość zamówień (bez anulowanych) przekracza 5000 zł, od największej.

5 Autokorekta i mapa zaległości

Porównaj swoje wyniki z wartościami kontrolnymi. Gdy się nie zgadzają, poszukaj przyczyny (warunek, typ złączenia, grupowanie) i popraw zapytanie. Poprawione zapytania oznacz w pliku komentarzem `-- poprawione`.

```
Z1 4 wiersze; pierwszy: Monitor 24" biurowy, 599.00
Z2 4 wiersze (jedna osoba bez miasta)
Z3 7
Z4 1 wiersz
Z5 7 wierszy; Laptopy: 3 produkty, średnio 4332.33
Z6 3 wiersze; wysłane: 5
Z7 12 wierszy; ostatnie: zamówienie 12
Z8 12 wierszy; największa wartość: 7996.00 (zamówienie 9)
Z9 2 wiersze
Z10 1 wiersz
Z11 5 wierszy (średnia cena: 1700.33)
Z12 8 wierszy; przełożony dyrektora: NULL
Z13 6 wierszy; Monitory: 4599.00
Z14 3 wiersze; pierwszy klient: 8241.00
Z15 4 wiersze; łączna wartość w Krakowie: 6792.00
```

1. Wpisz w sprawozdaniu przy każdym zapytaniu: zgodne od razu / zgodne po poprawce / niezgodne.
2. Wypełnij mapę zaległości: przy każdym temacie zaznacz, jak go oceniasz, i zapisz jedną rzecz do powtórzenia.
3. Zapisz plik `.sql` i oddaj go razem ze sprawozdaniem.

W sprawozdaniu umieść

- ☐ plik `.sql` z zapytaniami Z1–Z15 (z komentarzami numerów i oznaczeniem poprawek)
- ☐ tabela zapytań z informacją o zgodności z wynikiem kontrolnym
- ☐ mapa zaległości i plan powtórki
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego w Z3 warunek `BETWEEN '2026-03-01' AND '2026-06-30'` może pominąć część zamówień z 30 czerwca?
2. Czym różni się WHERE od HAVING? Podaj przykład z tego zadania.
3. Dlaczego w Z9 trzeba użyć LEFT JOIN, a nie JOIN?
4. Jaki błąd popełnisz, jeśli w Z13 zamiast podzapytania skorelowanego napiszesz `WHERE cena = (SELECT MAX(cena) FROM produkty)`?

Kryteria oceny

Kryterium	Punkty
Zapytania Z1–Z6 poprawne (wynik zgodny z kontrolnym)	5
Zapytania Z7–Z12 poprawne	6
Zapytania Z13–Z15 poprawne	3
Rzetelna autokorekta i mapa zaległości z planem	3
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Widoki — tworzenie i użycie

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się tworzenia widoków, które upraszczają złożone zapytania i ukrywają niepotrzebne kolumny, oraz sprawdzenie, że widok nie przechowuje danych.

Po zajęciach potrafisz:

- tworzy widoki poleceniem CREATE VIEW i zmienia je przez CREATE OR REPLACE VIEW
- korzysta z widoku jak z tabeli w zapytaniach z WHERE, GROUP BY i ORDER BY
- buduje widok na podstawie innego widoku i wyjaśnia skutki usunięcia widoku bazowego
- odczytuje definicję widoku (SHOW CREATE VIEW, information_schema.VIEWS)
- projektuje widok ukrywający dane osobowe

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Pracuj na świeżo zaimportowanej bazie sklep na swoim komputerze. Widoki usuwaj tylko te, które sam utworzyłeś.
- Polecenie z Z12 wykonuj w transakcji zakończonej ROLLBACK — dane mają wrócić do stanu wyjściowego.
- Widok ukrywający kolumny (Z5) chroni dane tylko wtedy, gdy użytkownik nie ma dostępu do samej tabeli — to temat zadania 3.

Przebieg zadania

1 Pierwszy widok

Widok to zapisane zapytanie: serwer wykonuje je przy każdym odczycie.

1. Zaimportuj ponownie `sklep.sql`, żeby zacząć od danych wyjściowych. Utwórz plik `r2_02_nazwisko.sql`.
2. Z1. Utwórz widok `v_katalog` z kolumnami: `id_produktu`, `nazwa`, `kategoria` (nazwa kategorii), `cena`, `stan`. Produkty bez kategorii też mają się w nim znaleźć.
3. Z2. Korzystając tylko z widoku `v_katalog`, wypisz monitory, a potem policz wszystkie wiersze widoku (powinno być 15).
4. Z3. Wyświetl listę widoków w bazie: `SHOW FULL TABLES WHERE Table_type = 'VIEW'`. Sprawdź, jak widok jest oznaczony w phpMyAdmin.

2 Widok z grupowaniem i widok bez danych osobowych

Widok `v_wartosc_zamowien` będzie potrzebny w kolejnych punktach — utwórz go dokładnie według poniższej definicji.

```
CREATE VIEW v_wartosc_zamowien AS
SELECT z.id_zamowienia, z.data_zamowienia, z.id_klienta,
       CONCAT(k.imie, ' ', k.nazwisko) AS klient, z.status,
       SUM(pz.ilosc * pz.cena_jedn) AS wartosc
FROM zamowienia z
JOIN klienci k ON k.id_klienta = z.id_klienta
JOIN pozycje_zamowienia pz ON pz.id_zamowienia = z.id_zamowienia
GROUP BY z.id_zamowienia, z.data_zamowienia, z.id_klienta,
         k.imie, k.nazwisko, z.status;
```

1. Z4. Utwórz widok `v_wartosc_zamowien` (kod powyżej). Napisz zapytanie do widoku, które pokaże 3 najdroższe zamówienia nieanulowane.
2. Z5. Utwórz widok `v_klienci_publiczni` z kolumnami: `id_klienta`, imie, nazwisko skrócone do pierwszej litery z kropką (np. „N.”), miasto. Bez adresu e-mail i daty rejestracji.
3. Z6. Spróbuj odczytać kolumnę `email` z widoku `v_klienci_publiczni`. Zapisz numer i treść błędu.

3 Zmiana widoku i widok na widoku

1. Z7. Zmień `v_katalog` poleceniem `CREATE OR REPLACE VIEW`: dodaj kolumnę `wartosc_magazynu` (`cena × stan`). Następnie oblicz z widoku wartość magazynu w każdej kategorii, od największej.
2. Z8. Utwórz widok `v_sprzedaz_klientow` na podstawie widoku `v_wartosc_zamowien`: `id_klienta`, klient, liczba_zamowien, razem — bez zamówień anulowanych. Wyświetl go posortowany malejąco według kwoty.
3. Z9. Utwórz widok `v_miesiace` z listą nazw kolumn w nagłówku `CREATE VIEW v_miesiace (miesiac, liczba, wartosc) AS ...`: miesiąc w formacie RRRR-MM, liczba i wartość zamówień nieanulowanych. Wyświetl go według miesięcy.

4 Jak widok działa w środku

1. Z10. Wykonaj `SHOW CREATE VIEW v_klienci_publiczni;` Odszukaj w wyniku `ALGORITHM`, `DEFINER` i `SQL SECURITY` i zapisz ich wartości.
2. Z11. Z tabeli `information_schema.VIEWS` wypisz nazwy widoków bazy sklep i kolumnę `IS_UPDATABLE`. Zastanów się, dlaczego część widoków ma wartość `NO`.
3. Z12. W konsoli rozpocznij transakcję, dodaj do zamówienia 8 pozycję (produkt 7, ilość 3, cena 129.00), odczytaj wartość zamówienia 8 z widoku, wykonaj `ROLLBACK` i odczytaj ją jeszcze raz. Co to mówi o przechowywaniu danych w widoku?
4. Z13. Usuń widok `v_wartosc_zamowien` i spróbuj odczytać `v_sprzedaz_klientow`. Zapisz błąd. Następnie odtwórz `v_wartosc_zamowien` i sprawdź, że `v_sprzedaz_klientow` znów działa.
5. Z14. Usuń widok `v_miesiace` poleceniem `DROP VIEW IF EXISTS`. Sprawdź, że tabele i dane bazy sklep zostały bez zmian.

W sprawozdaniu umieść

- ☐ plik `.sql` z poleceniami Z1–Z14
- ☐ zrzuty ekranu wyników Z4, Z7 i Z8 oraz komunikatów błędów z Z6 i Z13
- ☐ wartości `ALGORITHM`, `DEFINER` i `SQL SECURITY` z Z10
- ☐ wnioski z Z12 i Z13
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czy widok zajmuje miejsce na dysku tak jak tabela? Uzasadnij na podstawie Z12.
2. Podaj dwa powody, dla których programista aplikacji woląłby pisać `SELECT * FROM v_wartosc_zamowien` zamiast pełnego złączenia.
3. Dlaczego usunięcie widoku bazowego nie zostało zablokowane, choć zależy od niego inny widok? Jakie to niesie ryzyko?
4. Dlaczego widok `v_wartosc_zamowien` jest nieaktualizowalny (`IS_UPDATABLE = NO`)?

Kryteria oceny

Kryterium	Punkty
Widoki Z1–Z5 utworzone poprawnie i użyte w zapytaniach	6
Zmiana widoku i widoki na widokach (Z7–Z9)	5
Analiza działania widoków (Z10–Z14) z wnioskami	5
Odpowiedzi na pytania kontrolne	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Widoki — aktualizacja i uprawnienia

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Sprawdzenie, kiedy przez widok można zmieniać dane, jak WITH CHECK OPTION pilnuje warunku widoku i jak widok z uprawnieniami ogranicza dostęp użytkownika do danych.

Po zajęciach potrafisz:

- odróżnia widoki aktualizowalne od nieaktualizowalnych i wyjaśnia przyczynę
- stosuje WITH CHECK OPTION oraz porównuje LOCAL i CASCADED
- tworzy użytkownika z prawami tylko do wybranych widoków
- wyjaśnia różnicę między SQL SECURITY DEFINER i INVOKER

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **Konsola mysql** — potrzebne dwa okna: jedno jako root, drugie jako nowy użytkownik

Przygotowanie przed zajęciami

- Przypomnij sobie z zadania 27 (rok 1) polecenia CREATE USER, GRANT, REVOKE i SHOW GRANTS.

Bezpieczeństwo

- Zaimportuj świeżą bazę sklep przed zadaniem — będziesz zmieniać dane przez widoki.
- Nie zmieniaj uprawnień ani hasła konta root. Twórz tylko użytkowników z tego zadania i usuń ich na końcu (Z15).
- Hasła z zadania są przykładowe. Na prawdziwym serwerze każde konto ma własne, silne hasło, a użytkownik raportowy łączy się tylko z localhost lub zaufanej sieci.

Przebieg zadania

1 Zmiany danych przez widok

Widok z jednej tabeli, bez grupowania i bez wyrażeń w kolumnach, zwykle jest aktualizowalny: zmiana w widoku trafia do tabeli.

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_03_nazwisko.sql`.
2. Z1. Utwórz widok `v_tanie`: kolumny `id_produkty`, `nazwa`, `id_kategorii`, `cena`, `stan`, `data_dodania` z tabeli `produkty`, tylko produkty tańsze niż 500 zł. Ile ma wierszy?
3. Z2. Przez widok zwiększ o 5 stan produktu 7. Sprawdź w tabeli produkty, że zmiana tam trafiła.
4. Z3. Przez widok dodaj produkt „Podkładka pod mysz” (kategoria 3, cena 39.00, stan 50, data dzisiejsza). Potem dodaj w ten sam sposób „Tablet graficzny” za 899.00. Czy tablet widać w widoku? Czy jest w tabeli? Usuń go z tabeli produkty.
5. Z4. Spróbuj przez widok usunąć produkt 1 (`DELETE FROM v_tanie WHERE id_produkty = 1`). Ile wierszy usunięto i dlaczego?

2 WITH CHECK OPTION

Opcja sprawia, że przez widok nie da się wstawić ani zmienić wiersza tak, by wypadł poza warunek widoku.

1. Z5. Zmień `v_tanie` przez `CREATE OR REPLACE VIEW`, dodając na końcu `WITH CHECK OPTION`. Powtórz wstawienie tabletu za 899.00 i spróbuj zmienić cenę produktu 7 na 600. Zapisz kod błędu. Czy obniżka ceny produktu 7 do 119.00 przejdzie?
2. Z6. Utwórz widok `v_tanie_bez` (jak `v_tanie`, ale bez `CHECK OPTION`), a na nim dwa widoki akcesoriów (`id_kategorii = 3`): `v_akc_local` z `WITH LOCAL CHECK OPTION` i `v_akc_casc` z `WITH CASCADED CHECK OPTION`. Przez każdy z nich spróbuj wstawić akcesorium za 999.00 zł. Opisz różnicę.

3 Widoki nieaktualizowalne

Odtwórz widoki z zadania 2 według poniższych definicji.

```
CREATE VIEW v_wartosc_zamowien AS
SELECT z.id_zamowienia, z.data_zamowienia, z.id_klienta,
       CONCAT(k.imie, ' ', k.nazwisko) AS klient, z.status,
       SUM(pz.ilosc * pz.cena_jedn) AS wartosc
FROM zamowienia z
JOIN klienci k ON k.id_klienta = z.id_klienta
JOIN pozycje_zamowienia pz ON pz.id_zamowienia = z.id_zamowienia
GROUP BY z.id_zamowienia, z.data_zamowienia, z.id_klienta,
         k.imie, k.nazwisko, z.status;

CREATE VIEW v_klienci_publiczni AS
SELECT id_klienta, imie, CONCAT(LEFT(nazwisko, 1), '.') AS nazwisko, miasto
FROM klienci;
```

1. Z7. Spróbuj przez `v_wartosc_zamowien` zmienić status zamówienia 9 na 'wyslane'. Zapisz błąd.
2. Z8. Przez `v_klienci_publiczni` spróbuj zmienić nazwisko klienta 7, a potem jego miasto na „Łódź”. Które polecenie się udało i dlaczego?

4 Widok jako ograniczenie dostępu

Pracuj w dwóch oknach konsoli. W oknie 1 jesteś root, w oknie 2 logujesz się jako nowy użytkownik:

```
mysql -u raport -p sklep.
```

1. Z9. (root) Utwórz użytkownika 'raport'@'localhost' z hasłem Raport#2026 i nadaj mu tylko SELECT na widokach v_wartosc_zamowien i v_klienci_publiczni. Wyświetl SHOW GRANTS dla tego konta.
2. Z10. (raport) Wykonaj SHOW TABLES, odczytaj oba widoki, a potem spróbuj odczytać tabelę klienci i zmienić miasto przez v_klienci_publiczni. Zapisz kody błędów.
3. Z11. (root) Utwórz dwa widoki z kolumnami id_klienta, imie, miasto: v_klienci_definer z SQL SECURITY DEFINER i v_klienci_invoker z SQL SECURITY INVOKER. Klauzula stoi przed słowem VIEW: CREATE SQL SECURITY INVOKER VIEW Nadaj użytkownikowi raport SELECT na obu.
4. Z12. (raport) Odczytaj oba widoki. Który działa? Wyjaśnij, czyje uprawnienia sprawdza serwer w każdym przypadku.
5. Z13. (root) Utwórz użytkownika 'magazyn'@'localhost' (hasło Magazyn#2026) z prawami SELECT i UPDATE tylko do v_tanie. Jako magazyn zmniejsz o 1 stan produktu 8, spróbuj podnieść jego cenę do 999.00, zmienić stan w tabeli produkty i usunąć wiersz z widoku. Zapisz, co się udało.
6. Z14. (root) Odbierz użytkownikowi raport prawo do v_klienci_publiczni (REVOKE) i sprawdź SHOW GRANTS.
7. Z15. (root) Usuń obu użytkowników (DROP USER) i sprawdź w tabeli mysql.user, że ich nie ma.

W sprawozdaniu umieść

- ☐ plik .sql z poleceniami Z1–Z15 (polecenia wykonywane jako raport i magazyn oznacz komentarzem)
- ☐ zrzuty ekranu komunikatów błędów z Z5, Z7, Z10 i Z12
- ☐ tabela: które operacje przez widok się udały, a które nie i dlaczego
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Podaj trzy elementy zapytania, które sprawiają, że widok staje się nieaktualizowalny.
2. Co to jest „znikający wiersz” z Z3 i jak WITH CHECK OPTION mu zapobiega?
3. Kiedy SQL SECURITY DEFINER jest wygodny, a kiedy może być niebezpieczny?
4. Dlaczego użytkownik magazyn nie może zmienić ceny produktu na 999.00, choć ma prawo UPDATE do widoku?

Kryteria oceny

Kryterium	Punkty
Zmiany danych przez widok i ich analiza (Z1–Z4)	4
WITH CHECK OPTION, LOCAL i CASCADED (Z5–Z6)	4
Widoki nieaktualizowalne — błędy i wyjaśnienia (Z7–Z8)	2
Użytkownicy i uprawnienia do widoków, DEFINER/INVOKER (Z9–Z15)	6
Odpowiedzi na pytania kontrolne	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Warunki w SQL: CASE, IF, IFNULL, COALESCE, NULLIF

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się wybierania wartości według warunków w zapytaniach, budowania raportów przestawnych jednym zapytaniem i bezpiecznego obchodzenia się z NULL i dzieleniem przez zero.

Po zajęciach potrafisz:

- stosuje CASE w obu postaciach oraz funkcję IF w SELECT, ORDER BY i UPDATE
- kategoryzuje dane (przedziały cen, segmenty klientów) i grupuje po kategorii
- buduje raport przestawny (pivot) z SUM(CASE ...) i COUNT(CASE ...)
- zastępuje NULL wartością domyślną (IFNULL, COALESCE) i zapobiega dzieleniu przez zero (NULLIF)

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo
`C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisanie pliku .sql z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Polecenia UPDATE z Z12 i Z13 wykonuj tylko w transakcji (START TRANSACTION ... ROLLBACK) — ceny i stany mają wrócić do stanu wyjściowego.
- Przed każdym UPDATE z CASE sprawdź wynik tym samym wyrażeniem w SELECT. Brak gałęzi ELSE w CASE ustawia NULL — w kolumnie NOT NULL to błąd, a w innej utrata danych.

Przebieg zadania

1 CASE i IF

CASE ma dwie postacie: z warunkami (CASE WHEN warunek THEN ...) i z porównaniem do wartości (CASE kolumna WHEN wartość THEN ...).

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_04_nazwisko.sql`.
2. Z1. Każdemu produktowi przypisz przedział cenowy: „tani” (poniżej 300 zł), „średni” (300–1499,99 zł), „drogi” (od 1500 zł). Policz produkty w każdym przedziale i posortuj przedziały od najtańszego.
3. Z2. Dla zamówień 4, 8, 9 i 10 wypisz opis statusu (CASE z porównaniem): nowe → „czeka na płatność”, opłacone → „do wysłania”, wysłane → „zrealizowane”, pozostałe → „anulowane”.
4. Z3. Funkcją IF oznacz produkty o stanie poniżej 5 sztuk jako „dostępny” albo „brak”.
5. Z4. Wypisz zamówienia w kolejności obsługi: najpierw nowe, potem opłacone, wysłane i anulowane, a w obrębie statusu od najstarszego (CASE w ORDER BY). Pokaż 5 pierwszych.

2 NULL i wartości domyślne

1. Z5. Wypisz klientów bez miasta, pokazując zamiast NULL tekst „nie podano” (IFNULL). Wypisz produkt 1 i produkty bez kategorii z nazwą kategorii albo „bez kategorii” (COALESCE i LEFT JOIN).
2. Z6. Jednym SELECT-em sprawdź wartości: `COALESCE(NULL, NULL, 'trzeci', 'czwarty')`, `IFNULL(NULL, 'zastępstwo')`, `NULLIF(5, 5)`, `NULLIF(5, 3)`, `IF(NULL = NULL, 'tak', 'nie')`, `IFNULL(NULL = NULL, 'nieznane')`. Wyjaśnij każdy wynik.
3. Z7. Podziel klientów na segmenty według łącznej wartości zamówień bez anulowanych: VIP (od 5000 zł), stały (od 1000 zł), okazjonalny (powyżej 0), bez zakupów. Klienci bez zamówień też mają się pojawić z kwotą 0.00. Wskazówka: warunek `status <> 'anulowane'` umieść w ON złączenia, nie w WHERE.

3 Raporty przestawne

Wiersze grupujemy jak zwykle, a każda kolumna to osobny `SUM(CASE ...)` dla jednej wartości.

1. Z8. Raport: wiersze — miasta klientów, kolumny — liczba zamówień w statusach nowe, opłacone, wysłane, anulowane i razem. Posortuj według kolumny razem malejąco.
2. Z9. Raport sprzedaży (bez anulowanych): wiersze — kategorie (produkty bez kategorii jako „bez kategorii”), kolumny — wartość sprzedaży w kwartałach Q1, Q2, Q3 2026 i razem. Użyj funkcji `QUARTER()`.
3. Z10. Dla każdego klienta, który złożył zamówienie, podaj: liczbę wszystkich zamówień, liczbę anulowanych (`COUNT(CASE ...)`) oraz liczbę opłaconych lub wysłanych (`SUM(IF(...))`).

4 Dzielenie przez zero i CASE w UPDATE

1. Z11. Wykonaj `SELECT 10/0;` i `SHOW WARNINGS;`. Następnie dla każdego klienta oblicz średnią wartość zamówienia nieanulowanego: suma wartości nieanulowanych / liczba nieanulowanych (zbuduj najpierw tabelę pochodną z wartością każdego zamówienia). Klientka z samym anulowanym zamówieniem daje dzielenie przez zero — zabezpiecz mianownik funkcją `NULLIF` i porównaj liczbę ostrzeżeń przed i po.
2. Z12. Odczytaj `SELECT @@sql_mode;`. W transakcji spróbuj `UPDATE produkty SET stan = stan / 0 WHERE id_produkta = 1;` i zakończ `ROLLBACK`. Wyjaśnij, dlaczego w SELECT było tylko ostrzeżenie, a tu (przy `ERROR_FOR_DIVISION_BY_ZERO` i trybie ścisłym) jest błąd.
3. Z13. W transakcji jednym poleceniem `UPDATE` z `CASE` obniż ceny laptopów (kategoria 1) o 5% i podnieś ceny akcesoriów (kategoria 3) o 10%, z zaokrągleniem do groszy; pozostałe ceny bez zmian. Sprawdź liczbę zmienionych wierszy i nowe ceny, potem `ROLLBACK`.

W sprawozdaniu umieść

- ☐ plik `.sql` z zapytaniami Z1–Z13
- ☐ zrzuty ekranu wyników Z7, Z8 i Z9
- ☐ wyjaśnienie wyników Z6 i Z11–Z12
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się CASE z warunkami od CASE z porównaniem do wartości? Kiedy nie da się użyć tej drugiej postaci?
2. Co zwróci CASE bez ELSE, gdy żaden warunek nie jest spełniony? Jakie to ma skutki w UPDATE?
3. Czym różni się COALESCE od IFNULL?
4. Dlaczego w Z7 warunek na status trzeba umieścić w ON, a nie w WHERE?

Kryteria oceny

Kryterium	Punkty
CASE i IF w SELECT i ORDER BY (Z1–Z4)	4
Obsługa NULL i segmentacja klientów (Z5–Z7)	4
Raporty przestawne (Z8–Z10)	5
Dzielenie przez zero i UPDATE z CASE (Z11–Z13) z wyjaśnieniem	4
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Wyrażenia CTE i zapytania rekurencyjne

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się zapisywania złożonych zapytań jako czytelnych bloków WITH oraz przechodzenia przez hierarchie (pracownicy, drzewo kategorii) i generowania ciągów dat zapytaniem rekurencyjnym.

Po zajęciach potrafisz:

- zastępuje zagnieźdzone podzapytania jednym lub kilkoma wyrażeniami CTE
- pisze zapytanie WITH RECURSIVE z częścią początkową, częścią rekurencyjną i warunkiem stopu
- generuje kalendarz dat i łączy go z danymi, żeby pokazać okresy bez sprzedaży
- wypisuje hierarchię z poziomem i ścieżką, w dół i w górę drzewa

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)
- plik `r2_05_katalog_drzewo.sql` z folderu Pliki_do_zadan (tworzy bazę katalog)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku .sql z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Każde zapytanie rekurencyjne musi mieć warunek stopu. MySQL przerywa rekurencję po 1000 krokach (`cte_max_recursion_depth`), MariaDB od 10.6 też ma limit 1000 (`max_recursive_iterations`), ale **w MariaDB 10.4 z XAMPP limit jest praktycznie nieograniczony** — błędne zapytanie może obciążyć serwer na długo. Gdy zapytanie nie kończy się po kilku sekundach, przerwij je (Ctrl+C w konsoli) i popraw warunek.
- Zmianę danych w Z14 wykonuj tylko w transakcji zakończonej ROLLBACK.

Przebieg zadania

1 CTE zamiast podzapytań

Składnia: `WITH nazwa AS (SELECT ...), druga AS (SELECT ... FROM nazwa ...) SELECT ... FROM druga;`. Kolejne CTE mogą korzystać z wcześniejszych.

1. Zaimportuj `sklep.sql` i `r2_05_katalog_drzewo.sql`. Utwórz plik `r2_05_nazwisko.sql`. Pracuj w bazie `sklep`.
2. Z1. Za pomocą CTE `srednia` (średnia cena produktu) wypisz produkty droższe od średniej i o ile są droższe (zaokrąglaj do groszy).
3. Z2. Dwa CTE: `wartosci` (wartość każdego zamówienia nieanulowanego z `id_klienta`) i `klienci_sumy` (suma i liczba zamówień klienta, korzysta z `wartosci`). Wypisz imię, nazwisko, liczbę zamówień i sumę klientów, których suma jest większa od średniej sumy wszystkich klientów.
4. Z3. CTE `sprzedaz` (wartość sprzedaży w kategoriach, bez anulowanych) i `calosc` (suma wszystkiego). Wypisz kategorie z wartością i udziałem procentowym (1 miejsce po przecinku).

2 Rekurencja — generowanie ciągów

```
WITH RECURSIVE liczby AS (  
    SELECT 1 AS n                                -- część początkowa  
    UNION ALL  
    SELECT n + 1 FROM liczby WHERE n < 10        -- część rekurencyjna z warunkiem stopu  
)  
SELECT * FROM liczby;
```

1. Z4. Uruchom zapytanie z ramki. Zmień je tak, żeby jednym wierszem (`GROUP_CONCAT`) wypisało liczby 1–10 po przecinku.
2. Z5. Wygeneruj rekurencyjnie wszystkie miesiące 2026 roku (pierwszy dzień miesiąca, krok `+ INTERVAL 1 MONTH`) i połącz `LEFT JOIN`-em z liczbą i wartością zamówień nieanulowanych w miesiącu. Miesiące bez sprzedaży mają mieć 0. Wynik: 12 wierszy.
3. Z6. Wykonaj `SET lc_time_names = 'pl_PL';`. Wygeneruj wszystkie dni września 2026 i policz: soboty i niedziele (z polskimi nazwami dni — `DAYNAME`) oraz dni robocze (bez świąt). Użyj `DAYOFWEEK` (1 = niedziela, 7 = sobota).

3 Hierarchia pracowników

Tabela `pracownicy` ma kolumnę `id_przełożonego` — to drzewo zapisane w jednej tabeli.

1. Z7. Wypisz całą strukturę firmy od dyrektora: pracownik z wcięciem zależnym od poziomu (`REPEAT`), stanowisko, poziom i ścieżka nazwisk „Majewska > Olszewski > Król”. Posortuj po ścieżce. Uwaga: kolumnę ścieżki w części początkowej rzutuj na dłuższy tekst: `CAST(nazwisko AS CHAR(300))`.
2. Z8. Policz wszystkich podwładnych (na każdym poziomie) Marka Olszewskiego (`id` 2) i sumę ich pensji.
3. Z9. Wypisz drogę „w górę” od Igora Grabowskiego (`id` 7) do dyrektora z numerem kroku (0 = Igor).

4 Drzewo kategorii (baza katalog)

Przejdź do bazy katalog: `USE katalog;`. Kategoria główna ma `id_nadrzednej = NULL`.

1. Z10. Wypisz całe drzewo kategorii z wcięciami, poziomem i ścieżką „Komputery / Podzespoły / Dyski”, posortowane po ścieżce (18 wierszy).
2. Z11. Znajdź wszystkie kategorie w poddrzewie „Podzespoły” (razem z nią samą) i policz produkty w tym poddrzewie oraz ich najniższą i najwyższą cenę.
3. Z12. Zbuduj „okruszki” (breadcrumbs) dla produktu „Dysk SSD NVMe 2 TB”: jeden wiersz w postaci „Komputery > Podzespoły > Dyski > Dyski SSD”.
4. Z13. Dla każdej kategorii głównej policz produkty z całego jej poddrzewa (wskazówka: w CTE przenoś identyfikator korzenia).
5. Z14. W transakcji ustaw kategorii Podzespoły (id 6) kategorię nadrzędną 9 (Dyski SSD) — powstaje cykl. Napisz zapytanie o poddrzewo kategorii 6 z kolumną poziom i warunkiem stopu `WHERE p.poziom < 10` w części rekurencyjnej. Ile wierszy zwraca? Wykonaj ROLLBACK.

W sprawozdaniu umieść

- ☐ plik `.sql` z zapytaniami Z1–Z14
- ☐ zrzuty ekranu wyników Z5, Z7 i Z10
- ☐ wynik Z14 i wyjaśnienie, co by się stało bez warunku stopu
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Podaj dwie zalety CTE w porównaniu z podzapytaniem w FROM.
2. Z jakich dwóch części składa się CTE rekurencyjne i dlaczego łączy je UNION ALL?
3. Dlaczego w Z7 trzeba rzutować kolumnę ścieżki przez CAST w części początkowej?
4. Jak w bazie z cyklem (Z14) można wykryć, że wróciliśmy do już odwiedzonego węzła, bez ograniczania liczby poziomów?

Kryteria oceny

Kryterium	Punkty
CTE nierekurencyjne (Z1–Z3)	4
Generowanie ciągów i kalendarza (Z4–Z6)	4
Hierarchia pracowników (Z7–Z9)	4
Drzewo kategorii i cykl (Z10–Z14)	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Funkcje okna

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się liczenia rankingów, sum narastających, porównań z poprzednim okresem i udziałów w grupie bez utraty pojedynczych wierszy — za pomocą funkcji okna `OVER()`.

Po zajęciach potrafisz:

- odróżnia `GROUP BY` (skleja wiersze) od `OVER()` (zachowuje wiersze)
- numeruje i szereguje wiersze funkcjami `ROW_NUMBER`, `RANK` i `DENSE_RANK`, także w grupach (`PARTITION BY`)
- liczy sumy narastające, średnie kroczące i udziały procentowe w grupie
- porównuje wiersz z poprzednim i następnym (`LAG`, `LEAD`)
- wybiera top-N w każdej grupie przez CTE z funkcją okna

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder `Pliki_do_zadan`)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo
`C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Przygotowanie przed zajęciami

- Powtórz z zadania 5 zapis CTE (`WITH ... AS`).

Bezpieczeństwo

- Zadanie wymaga tylko odczytu danych i utworzenia jednego widoku. Nie zmieniaj danych w tabelach bazy sklep.

Przebieg zadania

1 Okno bez sklejania wierszy

Widok z wartością każdego zamówienia nieanulowanego będzie podstawą dalszych zapytań.

```
CREATE VIEW v_sprzedaz AS
SELECT z.id_zamowienia, z.id_klienta, z.data_zamowienia,
       SUM(pz.ilosc * pz.cena_jedn) AS wartosc
FROM zamowienia z
JOIN pozycje_zamowienia pz ON pz.id_zamowienia = z.id_zamowienia
WHERE z.status <> 'anulowane'
GROUP BY z.id_zamowienia, z.id_klienta, z.data_zamowienia;
```

1. Zaimportuj `sklep.sql`, utwórz plik `r2_06_nazwisko.sql` i widok `v_sprzedaz` (kod powyżej). Sprawdź: 11 zamówień o łącznej wartości 31639.00.
2. Z1. Dla każdego produktu z kategorią wypisz cenę, średnią cenę jego kategorii (`AVG ... OVER (PARTITION BY ...)`) i różnicę od tej średniej. Posortuj po kategorii i cenie malejąco.
3. Z2. Porównaj liczbę wierszy: `AVG(cena)` z `GROUP BY id_kategorii` oraz `AVG(cena) OVER (PARTITION BY id_kategorii)` bez `GROUP BY`. Wyjaśnij różnicę.
4. Z3. Spróbuj wybrać 3 najdroższe produkty warunkiem `WHERE ROW_NUMBER() OVER (ORDER BY cena DESC) <= 3`. Zapisz błąd.

2 Rankingi

1. Z4. Dla każdego klienta, który złożył zamówienie (wszystkie statusy), policz zamówienia i obok wypisz `ROW_NUMBER`, `RANK` i `DENSE_RANK` według tej liczby malejąco. W `ROW_NUMBER` dodaj drugie kryterium sortowania (nazwisko), żeby numeracja była powtarzalna. Opisz, czym różnią się trzy kolumny przy remisach.
2. Z5. Ponumeruj produkty w każdej kategorii od najdroższego (`ROW_NUMBER` z `PARTITION BY`).
3. Z6. Korzystając z Z5 i CTE, wypisz po 2 najdroższe produkty z każdej kategorii (top-N w grupie).
4. Z7. Podziel produkty na 3 grupy cenowe funkcją `NTILE(3)` (od najdroższych). Ile produktów trafiło do każdej grupy?

3 Sumy narastające i udziały

1. Z8. Z widoku `v_sprzedaz` wypisz zamówienia w kolejności dat z wartością i sumą narastającą sprzedaży (`SUM ... OVER (ORDER BY data_zamowienia)`).
2. Z9. Dla klientów 1, 2 i 4 wypisz ich zamówienia z sumą narastającą osobno dla każdego klienta oraz liczbą wszystkich zamówień klienta w każdym wierszu (`COUNT(*) OVER (PARTITION BY ...)`).
3. Z10. Dla każdego zamówienia z `v_sprzedaz` podaj jego udział procentowy w sprzedaży jego miesiąca (1 miejsce po przecinku).
4. Z11. Dla pracowników wypisz pensję, najwyższą pensję w zespole tego samego przełożonego i udział pensji w całym funduszu płac (`SUM(pensja) OVER ()`).

4 Porównania z sąsiednimi wierszami

1. Z12. W CTE policz sprzedaż w każdym miesiącu (z `v_sprzedaz`). Wypisz miesiąc, wartość, wartość z poprzedniego miesiąca (`LAG`), zmianę w złotych i w procentach (zabezpiecz dzielenie funkcją `NULLIF`).
2. Z13. Dla zamówień klientów 1, 2 i 4 (z tabeli `zamowienia`) wypisz datę następnego zamówienia tego samego klienta (`LEAD`) i liczbę dni do niego (`DATEDIFF`).
3. Z14. Dla zamówień z `v_sprzedaz` oblicz średnią kroczącą z 3 ostatnich zamówień: `AVG(wartosc) OVER (ORDER BY data_zamowienia ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)`.

W sprawozdaniu umieść

- ☐ plik `.sql` z zapytaniami Z1–Z14
- ☐ zrzuty ekranu wyników Z4, Z6, Z8 i Z12
- ☐ wyjaśnienie różnic ROW_NUMBER / RANK / DENSE_RANK na przykładzie Z4
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego funkcji okna nie można użyć w WHERE i jak to obejść?
2. Co zmienia PARTITION BY w porównaniu z samym OVER (ORDER BY ...)?
3. Klienci mają liczby zamówień 3, 2, 2, 1. Jakie numery dostaną od ROW_NUMBER, RANK i DENSE_RANK?
4. Dlaczego pierwsza wartość LAG w Z12 to NULL i jak to wpływa na zmianę procentową?

Kryteria oceny

Kryterium	Punkty
OVER z PARTITION BY i porównanie z GROUP BY (Z1–Z3)	3
Rankingi i top-N w grupie (Z4–Z7)	5
Sumy narastające i udziały (Z8–Z11)	5
LAG, LEAD i średnia krocząca (Z12–Z14)	4
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Wyrażenia regularne i walidacja danych

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się wyszukiwania danych według wzorca, wykrywania i poprawiania błędnych e-maili, telefonów i kodów pocztowych oraz blokowania złych danych ograniczeniem CHECK z REGEXP.

Po zajęciach potrafisz:

- zapisuje wzorce z kotwicami ^ \$, klasami znaków, alternatywą i kwantyfikatorami
- wykrywa błędne dane zapytaniem z NOT REGEXP
- normalizuje dane funkcjami REGEXP_REPLACE, REGEXP_SUBSTR i REGEXP_INSTR
- zabezpiecza tabelę ograniczeniem CHECK z wyrażeniem regularnym
- rozróżnia porównanie bez i z rozróżnianiem wielkości liter (COLLATE utf8mb4_bin)

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)
- plik `r2_07_zgloszenia_brudne.sql` z folderu Pliki_do_zadan (tworzy bazę rejestracja)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apacheFriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Dane poprawiaj tylko w kopii tabeli `zgloszenia_czyste` (Z9). Tabela `zgloszenia` zostaje bez zmian, żeby można było porównać wyniki i powtórzyć czyszczenie.
- Przed każdym UPDATE uruchom SELECT z tym samym warunkiem i sprawdź, które wiersze się zmieniają.
- Dane w pliku są fikcyjne. Prawdziwych danych osobowych z formularzy nie kopiuj na prywatne komputery ani do zrzutów ekranu w sprawozdaniu.

Przebieg zadania

1 Wzorce na bazie sklep

kolumna REGEXP 'wzorzec' zwraca 1, gdy wzorzec pasuje do dowolnego fragmentu tekstu. Kotwice `^` i `$` wymuszają dopasowanie od początku i do końca.

1. Zaimportuj `sklep.sql` i `r2_07_zgloszenia_brudne.sql`. Utwórz plik `r2_07_nazwisko.sql`.
2. Z1. W bazie sklep policz produkty, których nazwa zawiera cyfrę. Następnie wypisz produkty, których nazwa zaczyna się od „Laptop” albo „Monitor” (jeden wzorzec z alternatywą `|`).
3. Z2. Wypisz monitory z przekątną zapisaną jako dwie cyfry i cudzysłów (np. 27") oraz samą przekątną wyciętą funkcją REGEXP_SUBSTR.
4. Z3. Wypisz adresy e-mail klientów w formacie „litera.litera@...” (kropka między dwoma ciągami liter przed znakiem @).
5. Z4. Porównaj wyniki `'Nowak' REGEXP '^nowak$'` i `'Nowak' COLLATE utf8mb4_bin REGEXP '^nowak$'`. Wyjaśnij różnicę.

2 Wykrywanie błędów (baza rejestracja)

Wzorzec e-maila do tego zadania (uproszczony, ale wystarczający do wyłapania typowych pomyłek):

```
^[a-z0-9._%+-]+@[a-z0-9.-]+[.][a-z]{2,}$
```

1. Z5. `USE rejestracja;` Wypisz zgłoszenia z e-mailem niezgodnym ze wzorcem z ramki oraz osobno e-maile z dwiema kropkami obok siebie. Dlaczego zgłoszenia 4 i 19 też trafiły na listę?
2. Z6. Wypisz zgłoszenia z kodem pocztowym w złym formacie (poprawny: dwie cyfry, myślnik, trzy cyfry).
3. Z7. Dla każdego zgłoszenia pokaż telefon, same cyfry telefonu (REGEXP_REPLACE usuwający wszystko poza cyframi) i ich liczbę. Które numery nie dadzą się poprawić automatycznie?
4. Z8. Wypisz osoby, których imię lub nazwisko nie ma postaci „wielka litera + małe litery” (użyj COLLATE utf8mb4_bin i polskich liter w klasach znaków). Funkcją REGEXP_INSTR wskaż pozycję cyfry w nazwisku.

3 Czyszczenie danych w kopii

1. Z9. Utwórz kopię: `CREATE TABLE zgloszenia_czyste LIKE zgloszenia;` i przepis� dane `INSERT ... SELECT`. Jednym UPDATE popraw imiona i nazwiska (usuń spacje, pierwsza litera wielka, reszta mała) oraz e-maile (usuń spacje z brzegów, małe litery). Ile wierszy się zmieniło?
2. Z10. Ujednolicić telefony do 9 cyfr: najpierw zostaw same cyfry, potem z numerów zaczynających się od 48 lub 0048 i mających po nich 9 cyfr zostaw 9 ostatnich (RIGHT). Numery, które nadal nie mają 9 cyfr, ustaw na NULL.
3. Z11. Ujednolicić kody pocztowe: jeśli po usunięciu znaków innych niż cyfry zostaje 5 cyfr, zapisz je w postaci 00-000. Który kod został błędny i dlaczego?
4. Z12. Po czyszczeniu znajdź duplikaty zgłoszeń (ten sam e-mail więcej niż raz).

4 Ograniczenia CHECK

CHECK z REGEXP działa w MariaDB od 10.2 i w MySQL od 8.0.16. Wartość NULL nie narusza ograniczenia.

1. Z13. Dodaj do `zgloszenia_czyste` ograniczenie `chk_kod` sprawdzające format kodu pocztowego. Co się stało i dlaczego? Ustaw błędne kody na NULL i dodaj ograniczenie ponownie. Spróbuj wstawić zgłoszenie z kodem 12345 — zapisz kod błędu.
2. Z14. Błędne e-maile ustaw na NULL (do ręcznego wyjaśnienia), a potem dodaj ograniczenie `chk_email` (wzorzec z ramki i brak dwóch kropek). Sprawdź wstawienie e-maila `jan.test@example` (błąd) i `jan.test@example.pl` (poprawne).

W sprawozdaniu umieść

- ☐ plik `.sql` z poleceniami Z1–Z14
- ☐ zrzuty ekranu wyników Z5, Z7 oraz błędów z Z13 i Z14
- ☐ tabela błędów wykrytych w danych i sposobu ich poprawienia
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

- Czym różni się `nazwa LIKE '%7%'` od `nazwa REGEXP '7'`, a czym `nazwa REGEXP '^7$'`?
- Dlaczego bez `COLLATE utf8mb4_bin` wzorzec `^[A-Z][a-z]+$` nie wykryje imienia „piotr”?
- Dlaczego wzorzec e-maila w REGEXP nie gwarantuje, że adres istnieje? Jak aplikacja może to sprawdzić?
- Dlaczego przed dodaniem CHECK trzeba najpierw poprawić istniejące dane?

Kryteria oceny

Kryterium	Punkty
Wzorce na bazie sklep (Z1–Z4)	4
Wykrywanie błędnych danych (Z5–Z8)	5
Czyszczenie danych w kopii (Z9–Z12)	5
Ograniczenia CHECK z REGEXP (Z13–Z14)	3
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Procedury składowane — podstawy

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się tworzenia i wywoływania procedur składowanych z parametrami IN, OUT i INOUT, zmiennymi lokalnymi i instrukcją IF.

Po zajęciach potrafisz:

- tworzy procedurę z ogranicznikiem DELIMITER w konsoli i w phpMyAdmin
- stosuje parametry IN, OUT i INOUT oraz zmienne sesji @zmienna
- deklaruje zmienne lokalne, przypisuje wynik zapytania przez SELECT ... INTO i używa IF/ELSEIF
- pisze procedurę dodającą dane i procedurę raportu
- przegląda i usuwa procedury (SHOW PROCEDURE STATUS, SHOW CREATE PROCEDURE, DROP PROCEDURE)

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisanie pliku .sql z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Procedury zmieniające dane (Z8, Z10) testuj na świeżo zaimportowanej bazie, a Z10 w transakcji zakończonej ROLLBACK.
- Usuwać tylko procedury, które sam utworzyłeś.

Przebieg zadania

1 Pierwsza procedura

Średnik kończy polecenia wewnątrz procedury, więc na czas jej tworzenia zmieniamy znak końca polecenia na //. W phpMyAdmin zamiast DELIMITER wpisz // w polu „Ogranicznik” pod oknem SQL.

```
DELIMITER //
DROP PROCEDURE IF EXISTS nazwa //
CREATE PROCEDURE nazwa(IN p_param TYP, OUT p_wynik TYP)
BEGIN
    DECLARE v_zmienna TYP;          -- zmienna lokalna (tylko w procedurze)
    ...
END //
DELIMITER ;

CALL nazwa(wartosc, @wynik);      -- @wynik to zmienna sesji
SELECT @wynik;
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_08_nazwisko.sql`; wszystkie procedury twórz według szkieletu z ramki.
2. Z1. Utwórz procedurę `lista_kategorii()` wypisującą kategorie alfabetycznie i wywołaj ją.
3. Z2. Utwórz procedurę `produkty_kategorii(IN p_kategoria VARCHAR(50))` wypisującą nazwę, cenę i stan produktów podanej kategorii od najtańszego. Wywołaj dla kategorii Monitory.
4. Z3. Utwórz procedurę `produkty_w_cenie(IN p_od, IN p_do)` (typ DECIMAL(10,2)) i wywołaj ją dla przedziału 300–700 zł.

2 Parametry OUT i INOUT

1. Z4. Utwórz procedurę `liczba_zamowien(IN p_klient INT, OUT p_liczba INT)`. Wywołaj ją dla klienta 1 ze zmienną `@n` i wyświetl wynik.
2. Z5. Utwórz procedurę `statystyka_klienta(IN p_klient, OUT p_liczba, OUT p_suma)` zwracającą liczbę i łączną wartość zamówień nieanulowanych klienta (SELECT ... INTO dwóch parametrów). Wywołaj dla klientów 4 i 3.
3. Z6. Utwórz procedurę `dołącz_vat(INOUT p_kwota DECIMAL(10,2))`, która mnoży kwotę przez 1,23. Ustaw `SET @k = 100;`, wywołaj procedurę dwa razy i za każdym razem wyświetl `@k`. Wyjaśnij wynik.

3 Zmienne lokalne i IF

1. Z7. Utwórz procedurę `ocena_stanu(IN p_produkt INT)`: w zmiennych lokalnych zapamiętaj nazwę i stan produktu, a potem ustal ocenę: „nie ma takiego produktu” (brak wiersza), „brak — zamów” (stan 0), „niski stan” (poniżej 10), „stan dobry”. Wypisz id, nazwę, stan i ocenę. Sprawdź produkty 3, 2, 7 i 99.
2. Z8. Utwórz procedurę `dodaj_produkt(IN p_nazwa, IN p_kategoria, IN p_cena, IN p_stan, OUT p_id)`. Kategorię podaje się nazwą: jeśli nie istnieje, procedura ją dodaje. Data dodania — dzisiejsza. W `p_id` zwróć numer nowego produktu (LAST_INSERT_ID()). Dodaj „Kamera internetowa” (Akcesoria, 219.00, 15 szt.) i „Router Wi-Fi 7” (Sieci, 899.00, 6 szt.) i sprawdź wynik złączeniem produktów z kategoriami.

4 Raport, zmiana danych i zarządzanie procedurami

1. Z9. Utwórz procedurę `raport_miesiaca(IN p_rok INT, IN p_miesiac INT)` wypisującą zamówienia z danego miesiąca: numer, datę, klienta, status i wartość. Wywołaj dla kwietnia i grudnia 2026.
2. Z10. Utwórz procedurę `podwyzka(IN p_procent, IN p_stanowisko, OUT p_zmienionych)` podnoszącą pensję na stanowisku o podany procent i zwracającą liczbę zmienionych wierszy (`ROW_COUNT()`). W transakcji daj sprzedawcom 5%, sprawdź pensje, wykonaj `ROLLBACK`.
3. Z11. Wyświetl listę swoich procedur: `SHOW PROCEDURE STATUS WHERE Db = 'sklep'`; oraz z tabeli `information_schema.ROUTINES`. Wyświetl definicję jednej procedury: `SHOW CREATE PROCEDURE dodaj_produkt;`. Znajdź procedury w phpMyAdmin (baza sklep → Procedury).
4. Z12. Wywołaj `CALL liczba_zamowien(1);` (za mało parametrów), usuń procedurę `lista_kategorii` i spróbuj ją wywołać. Zapisz oba kody błędów.

W sprawozdaniu umieść

- ☐ plik `.sql` z definicjami procedur i wywołaniami Z1–Z12
- ☐ zrzuty ekranu wywołań Z5, Z7 (cztery produkty) i Z8
- ☐ wyjaśnienie różnicy między parametrem OUT, INOUT i zmienną sesji @
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Po co przed utworzeniem procedury zmieniamy ogranicznik poleceń (`DELIMITER`)?
2. Czym różni się zmienna lokalna `DECLARE v_x` od zmiennej sesji `@x`?
3. Dlaczego w Z8 trzeba zapamiętać `LAST_INSERT_ID()` zaraz po wstawieniu kategorii, a nie dopiero na końcu?
4. Podaj dwie zalety trzymania operacji (np. dodawania produktu) w procedurze zamiast w kodzie aplikacji.

Kryteria oceny

Kryterium	Punkty
Procedury z parametrami IN (Z1–Z3)	3
Parametry OUT i INOUT (Z4–Z6)	4
Zmienne lokalne, IF i procedura dodająca dane (Z7–Z8)	5
Raport, procedura zmieniająca dane, zarządzanie procedurami (Z9–Z12)	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Funkcje składowane i pętle

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się tworzenia własnych funkcji zwracających wartość, używania ich w zapytaniach oraz programowania pętli WHILE, REPEAT i LOOP.

Po zajęciach potrafisz:

- tworzy funkcję z RETURNS i RETURN oraz właściwą charakterystyką (DETERMINISTIC, NO SQL, READS SQL DATA)
- używa własnych funkcji w SELECT, WHERE, GROUP BY i ORDER BY
- programuje pętle WHILE, REPEAT ... UNTIL i LOOP z LEAVE i ITERATE
- odróżnia funkcję od procedury i rozpoznaje typowe błędy (przepętlenie, porównywanie napisów)

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo
`C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisanie pliku .sql z zapytaniami
code.visualstudio.com

Przygotowanie przed zajęciami

- Powtórz z zadania 8 zapis DELIMITER i różnicę między parametrem a zmienną lokalną.

Bezpieczeństwo

- Każda pętla musi mieć warunek zakończenia, który na pewno zostanie spełniony. Nieskończona pętla w funkcji wywołanej dla każdego wiersza zablokuje serwer — w konsoli przerwiesz ją Ctrl+C, w phpMyAdmin trzeba zatrzymać zapytanie (KILL) albo zrestartować MySQL w panelu XAMPP.
- Zadanie nie zmienia danych — tworzysz tylko funkcje.

Przebieg zadania

1 Funkcje bez dostępu do tabel

Funkcja zwraca jedną wartość i używa się jej jak wbudowanej funkcji SQL. Charakterystyka po RETURNS: DETERMINISTIC (dla tych samych argumentów zawsze ten sam wynik), NO SQL (nie czyta tabel) albo READS SQL DATA (czyta tabele).

```
DELIMITER //
DROP FUNCTION IF EXISTS brutto //
CREATE FUNCTION brutto(p_netto DECIMAL(10,2)) RETURNS DECIMAL(10,2)
DETERMINISTIC NO SQL
RETURN ROUND(p_netto * 1.23, 2) //
DELIMITER ;
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_09_nazwisko.sql`.
2. Z1. Utwórz funkcję `brutto` z ramki. Wypisz laptopy z ceną netto i brutto.
3. Z2. Utwórz funkcję `z1(p_kwota DECIMAL(12,2))` zwracającą tekst w formacie „3 299,00 zł” (spacja tysięcy, przecinek dziesiętny). Wskazówka: `FORMAT(x, 2)` daje „3,299.00” — zamień znaki funkcją `REPLACE` w odpowiedniej kolejności. Sprawdź dla 3299, 1234567.5 i 0.5 oraz w zapytaniu z Z1.
4. Z3. Utwórz funkcję `inicjaly(p_imie, p_nazwisko)` zwracającą np. „A.N.” (wielkie litery) i wypisz klientów w postaci „A.N. Nowak”.

2 Funkcje czytające dane

1. Z4. Utwórz funkcję `wartosc_zamowienia(p_zam INT) RETURNS DECIMAL(10,2) READS SQL DATA`, która zwraca sumę `ilosc × cena_jedn` pozycji zamówienia (0, gdy zamówienie nie istnieje). Wypisz 3 najdroższe zamówienia z wartością sformatowaną funkcją `z1`. Policz funkcją łączną wartość zamówień nieanulowanych (powinno być 31639.00).
2. Z5. Utwórz funkcję `segment_klienta(p_klient INT)` zwracającą VIP (od 5000 zł), stały (od 1000 zł), okazjonalny (powyżej 0) lub „bez zakupów” — według wartości zamówień nieanulowanych (porównaj z Z7 zadania 4). Policz klientów w każdym segmencie (`GROUP BY` po wyniku funkcji).
3. Z6. Wypisz nazwiska klientów VIP warunkiem `WHERE segment_klienta(id_klienta) = 'VIP'`. Jeśli pojawi się błąd 1267 (Illegal mix of collations), zapisz go i popraw warunek, dopisując po napisie `COLLATE utf8mb4_polish_ci`. Wyjaśnij przyczynę.

3 Pętle

WHILE warunek DO ... END WHILE (sprawdza przed krokiem), REPEAT ... UNTIL warunek END REPEAT (po kroku), etykieta: LOOP ... END LOOP z LEAVE (wyjście) i ITERATE (następny obrót).

1. Z7. Utwórz funkcję `silnia(p_n INT) RETURNS BIGINT` z pętlą WHILE. Sprawdź `silnia(0)`, `silnia(5)`, `silnia(10)`, `silnia(20)`, a potem osobno `silnia(21)`. Zapisz błąd i wyjaśnij.
2. Z8. Utwórz funkcję `suma_cyfr(p_liczba INT UNSIGNED)` z pętlą REPEAT (`MOD 10` i `DIV 10`). Sprawdź dla 2026, 0 i 99999. Dlaczego REPEAT, a nie WHILE, dobrze obsługuje liczbę 0?
3. Z9. Utwórz funkcję `parzyste(p_do INT) RETURNS VARCHAR(255)` z pętlą LOOP, która zwraca liczby parzyste od 2 do `p_do` oddzielone spacją (LEAVE po przekroczeniu `p_do`, ITERATE dla liczb nieparzystych). Sprawdź `parzyste(10)` i `parzyste(1)`.
4. Z10. Utwórz funkcję `dni_robocze(p_od DATE, p_do DATE)` liczącą dni od poniedziałku do piątku w przedziale (pętla po kolejnych dniach). Sprawdź wrzesień i październik 2026 (porównaj z Z6 zadania 5) oraz przedział z datą końcową wcześniejszą niż początkowa.

4 Funkcja czy procedura

1. Z11. Spróbuj wywołać funkcję jak procedurę: `CALL brutto(100);`. Zapisz błąd.
2. Z12. Wypisz z `information_schema.ROUTINES` nazwy, typ, `IS_DETERMINISTIC` i `SQL_DATA_ACCESS` swoich funkcji. Wyświetl `SHOW FUNCTION STATUS WHERE Db = 'sklep';` i usuń funkcję `parzyste` (`DROP FUNCTION`).

W sprawozdaniu umieść

- ☐ plik `.sql` z definicjami funkcji i zapytaniami Z1–Z12
- ☐ zrzuty ekranu wyników Z2, Z4, Z5 i Z10
- ☐ opis błędów z Z6, Z7 i Z11 z wyjaśnieniem
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Podaj trzy różnice między funkcją a procedurą składowaną.
2. Kiedy funkcję wolno oznaczyć jako `DETERMINISTIC`? Czy `wartosc_zamowienia` jest deterministyczna?
3. Dlaczego funkcja wywoływana w `WHERE` dla każdego wiersza może spowolnić zapytanie na dużej tabeli?
4. Jaką największą silnię zmieści `BIGINT`, a jaką `BIGINT UNSIGNED`? Jak można by policzyć większe?

Kryteria oceny

Kryterium	Punkty
Funkcje bez dostępu do tabel (Z1–Z3)	4
Funkcje czytające dane i ich użycie w zapytaniach (Z4–Z6)	5
Pętle <code>WHILE</code> , <code>REPEAT</code> i <code>LOOP</code> (Z7–Z10)	6
Funkcja a procedura, katalog funkcji (Z11–Z12)	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Obsługa błędów w procedurach

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się zgłaszania własnych błędów, przechwytywania błędów serwera w procedurach i wycofywania całej operacji, gdy jeden z jej kroków się nie uda.

Po zajęciach potrafisz:

- zgłasza własny błąd poleceniem SIGNAL SQLSTATE '45000' z komunikatem
- przechwytuje błędy procedurą obsługi CONTINUE i EXIT (kod błędu, SQLEXCEPTION, NOT FOUND)
- odczytuje kod i treść błędu przez GET DIAGNOSTICS i zapisuje je w dzienniku
- przekazuje błąd dalej przez RESIGNAL
- pisze procedurę z transakcją, która przy błędzie wycofuje wszystkie zmiany

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisanie pliku .sql z zapytaniami
code.visualstudio.com
- **Konsola mysql** — pokazuje pełne komunikaty błędów z SIGNAL — wygodniejsza niż phpMyAdmin

Przygotowanie przed zajęciami

- Powtórz z zadania 8 tworzenie procedur oraz z zadania 12 (rok 1) transakcje START TRANSACTION, COMMIT i ROLLBACK.

Bezpieczeństwo

- Zaimportuj świeżą bazę sklep — procedury zmieniają stany magazynowe i dodają zamówienia.
- W procedurze z transakcją ROLLBACK w obsłudze błędu jest obowiązkowy: bez niego po błędzie zostaje otwarta transakcja z częścią zmian, która może zablokować tabele innym użytkownikom.
- Komunikaty błędów pokazywane użytkownikowi aplikacji mają być zrozumiałe, ale nie mogą zdradzać budowy bazy (nazw tabel, zapytań). Szczegóły zapisuj w dzienniku błędów.

Przebieg zadania

1 SIGNAL — własne błędy

Ściąga z poleceniami tego zadania:

```
DECLARE CONTINUE HANDLER FOR 1062 SET v_dup = 1;      -- po błędzie idź dalej
DECLARE EXIT HANDLER FOR SQLEXCEPTION                -- po błędzie wyjdź z bloku
BEGIN
    GET DIAGNOSTICS CONDITION 1 @kod = MYSQL_ERRNO, @tekst = MESSAGE_TEXT;
    ROLLBACK;
    RESIGNAL;                                          -- przekaz błąd dalej
END;
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Własny komunikat';
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_10_nazwisko.sql`.
2. Z1. Utwórz procedurę `sprzedaj(IN p_produkt INT, IN p_ile INT)`, która zmniejsza stan produktu, a zgłasza błąd SIGNAL z komunikatem: „Nie ma takiego produktu”, „Ilość musi być dodatnia” albo „Za mało towaru w magazynie”. Sprawdź wywołania (2, 1), (2, 10), (99, 1) i (2, 0). Jaki numer błędu pokazuje serwer?

2 Procedury obsługi błędów

Procedura obsługi (handler) deklarowana jest po zmiennych, a przed pozostałymi poleceniami bloku.

1. Z2. Utwórz procedurę `dodaj_kategorie(IN p_nazwa, OUT p_komunikat)` z CONTINUE HANDLER FOR 1062 (duplikat klucza UNIQUE). Zwraca „Dodano kategorię ... (id ...)” albo „Kategoria ... już istnieje”. Sprawdź dla „Sieci” i „Laptopy”.
2. Z3. Utwórz procedurę `cena_produktu(IN p_produkt, OUT p_cena)` z CONTINUE HANDLER FOR NOT FOUND, która dla nieistniejącego produktu zwraca -1 zamiast ostrzeżenia „No data”. Sprawdź produkty 1 i 99.
3. Z4. Utwórz procedurę `dodaj_pozycje(IN p_zam, IN p_produkt, IN p_ilosc, OUT p_wynik)`, która wstawia pozycję zamówienia z aktualną ceną produktu, a dwie procedury EXIT HANDLER zamieniają błędy 1452 (klucz obcy) i 1062 (produkt już jest w zamówieniu) na czytelny komunikat. Sprawdź wywołania (9, 7, 1) dwa razy, (99, 7, 1) i (9, 99, 1). Cenę pobierz podzapytaniem z IFNULL(..., 0), żeby brak produktu dał błąd klucza obcego, a nie błąd NOT NULL.

3 Dziennik błędów i RESIGNAL

```
CREATE TABLE IF NOT EXISTS log_bledow (
    id INT AUTO_INCREMENT PRIMARY KEY,
    kiedy DATETIME NOT NULL,
    procedura VARCHAR(64) NOT NULL,
    kod INT,
    komunikat VARCHAR(255)
);
```

1. Z5. Utwórz tabelę `log_bledow` (kod powyżej) i procedurę `zmien_cene(IN p_produkt, IN p_cena)`. EXIT HANDLER FOR SQLEXCEPTION ma odczytać przez GET DIAGNOSTICS kod i treść błędu, zapisać je w `log_bledow`, a potem wykonać RESIGNAL. Zmień cenę produktu 1 na 3199.00, a potem na -5.
2. Z6. Wyświetl zawartość `log_bledow`. Wyjaśnij, dlaczego wpis w dzienniku pozostał, choć procedura zakończyła się błędem (bez transakcji każde polecenie zatwierdza się samo).

4 Transakcja w procedurze — składanie zamówienia

Zamówienie to kilka kroków: nagłówek w zamówienia, pozycja w pozycje_zamowienia i zmniejszenie stanu w produkty. Albo wszystkie się udają, albo żaden.

1. Z7. Utwórz procedurę `zloz_zamowienie(IN p_klient, IN p_produkt, IN p_ilosc, OUT p_id): EXIT HANDLER FOR SQLEXCEPTION z ROLLBACK i RESIGNAL; START TRANSACTION; sprawdzenie klienta (SIGNAL); dodanie zamówienia ze statusem nowe; odczyt stanu i ceny produktu ... FOR UPDATE; SIGNAL przy braku produktu lub za małym stanie; dodanie pozycji, zmniejszenie stanu; COMMIT.`
2. Z8. Policz zamówienia i zapisz stan produktu 4. Złóż zamówienie klienta 10 na 2 sztuki produktu 4. Sprawdź numer zamówienia, pozycję i nowy stan.
3. Z9. Spróbuj zamówić 50 sztuk produktu 5, produkt 99 i zamówienie dla klienta 99. Po każdej próbie sprawdź, czy liczba zamówień i stan produktu 5 się nie zmieniły.
4. Z10. Złóż jeszcze jedno poprawne zamówienie (klient 10, produkt 4, 1 sztuka). Jaki numer dostało? Wyjaśnij lukę w numeracji.

W sprawozdaniu umieść

- ☐ plik `.sql` z definicjami procedur i wywołaniami Z1–Z10
- ☐ zrzuty ekranu błędów z Z1 i Z9 oraz zawartości `log_bledow`
- ☐ tabela testów procedury `zloz_zamowienie` (wywołanie, komunikat, liczba zamówień, stan)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się CONTINUE HANDLER od EXIT HANDLER? Podaj przykład, kiedy użyjesz każdego z nich.
2. Co oznacza SQLSTATE '45000' i jaki numer błędu (MYSQL_ERRNO) dostaje komunikat zgłoszony przez SIGNAL?
3. Co stałoby się w Z9, gdyby obsługa błędu w `zloz_zamowienie` nie miała ROLLBACK, a co — gdyby nie miała RESIGNAL?
4. Dlaczego numer zamówienia z Z10 nie jest kolejnym po zamówieniu z Z8?

Kryteria oceny

Kryterium	Punkty
SIGNAL z własnymi komunikatami (Z1)	3
Procedury obsługi CONTINUE i EXIT (Z2–Z4)	5
Dziennik błędów, GET DIAGNOSTICS i RESIGNAL (Z5–Z6)	3
Procedura <code>zloz_zamowienie</code> z transakcją i testami (Z7–Z10)	6
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Wyzwalacze — dziennik zmian (audyt)

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się tworzenia wyzwalaczy, które automatycznie zapisują historię zmian danych, oraz poznanie ograniczeń wyzwalaczy.

Po zajęciach potrafisz:

- tworzy wyzwalacze BEFORE i AFTER dla INSERT, UPDATE i DELETE
- korzysta z wartości OLD i NEW i porównuje je bezpiecznie dla NULL (⇔)
- projektuje tabelę dziennika i zapisuje w niej kto, kiedy i co zmienił
- rozpoznaje ograniczenia wyzwalaczy: błąd 1442 i brak reakcji na kaskadowe usunięcie

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Zaimportuj świeżą bazę sklep — w zadaniu zmieniasz ceny, klientów i usuwasz zamówienie. Na koniec zaimportuj ją ponownie, żeby kolejne zadania zaczynały od danych wyjściowych.
- Dziennik zmian zawiera dane osobowe (e-maile). W prawdziwym systemie dostęp do tabel audytu ma tylko administrator, a dane przechowuje się tylko tak długo, jak wymagają tego przepisy (RODO).

Przebieg zadania

1 Historia cen

W wyzwalaczu UPDATE dostępne są dwie wersje wiersza: OLD (przed zmianą) i NEW (po zmianie).

```
DELIMITER //
DROP TRIGGER IF EXISTS trg_produkty_cena //
CREATE TRIGGER trg_produkty_cena
AFTER UPDATE ON produkty
FOR EACH ROW
BEGIN
    IF NEW.cena <> OLD.cena THEN
        INSERT INTO historia_cen (id_produkty, stara_cena, nowa_cena, kto, kiedy)
        VALUES (OLD.id_produkty, OLD.cena, NEW.cena, CURRENT_USER(), NOW());
    END IF;
END //
DELIMITER ;
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_11_nazwisko.sql`.
2. Z1. Utwórz tabelę `historia_cen` (id AUTO_INCREMENT, id_produkty, stara_cena, nowa_cena, kto VARCHAR(100), kiedy DATETIME) i wyzwalacz `trg_produkty_cena` (kod powyżej). Zmień cenę produktu 1 na 3199.00, a potem tylko jego stan (+1). Ile wierszy jest w historii i dlaczego?
3. Z2. Jednym poleceniem podnieś o 10% ceny akcesoriów (kategoria 3). Ile wierszy dopisał wyzwalacz? Co oznacza FOR EACH ROW?
4. Z3. Wypisz historię zmian z nazwą produktu, starą i nową ceną, zmianą w procentach i użytkownikiem, który zmienił cenę.

2 Audyt tabeli klienci

Jedna tabela dziennika dla trzech operacji. Dane „przed” i „po” zapisujemy jako tekst e-mail; miasto.

```
CREATE TABLE audyt_klienci (
    id INT AUTO_INCREMENT PRIMARY KEY,
    operacja ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
    id_klienta INT UNSIGNED NOT NULL,
    stare VARCHAR(255) NULL,
    nowe VARCHAR(255) NULL,
    kto VARCHAR(100) NOT NULL,
    kiedy DATETIME NOT NULL
);
```

1. Z4. Utwórz tabelę `audyt_klienci` (kod powyżej) i trzy wyzwalacze AFTER: `trg_klienci_ai` (INSERT — tylko nowe), `trg_klienci_au` (UPDATE — stare i nowe) i `trg_klienci_ad` (DELETE — tylko stare). Pomocna funkcja: `CONCAT_WS(';', ' ', NEW.email, NEW.miasto)`.
2. Z5. Utwórz wyzwalacz BEFORE INSERT `trg_klienci_bi`, który przed zapisem zamienia e-mail na małe litery bez spacji na brzegach, a puste miasto (same spacje) na NULL. Dodaj klientkę Ola Nowicka z e-mailem „Ola.Nowicka@Example.PL ” i miastem „ ”. Sprawdź, co zapisało się w tabeli klienci i w dzienniku.
3. Z6. Zmień miasto Oli Nowickiej na Lublin, „zmień” miasto klientki 1 na Kraków (to samo, co już ma), usuń Olę Nowicką i spróbuj usunąć klientkę 1. Wyświetl dziennik. Dlaczego próba usunięcia klientki 1 nie zostawiła wpisu?
4. Z7. Popraw `trg_klienci_au` tak, żeby zapisywał wpis tylko wtedy, gdy e-mail lub miasto naprawdę się zmieniły. Użyj porównania odpornego na NULL: `IF NOT (OLD.email <=> NEW.email AND OLD.miasto <=> NEW.miasto) THEN ...`. Sprawdź: ponownie Kraków dla klientki 1, miasto NULL dla klienta 2, zmiana samego imienia klientki 7.

3 Przegląd i ograniczenia wyzwalaczy

1. Z8. Wyświetl `SHOW TRIGGERS;` oraz z `information_schema.TRIGGERS` kolumny `TRIGGER_NAME`, `ACTION_TIMING`, `EVENT_MANIPULATION`, `EVENT_OBJECT_TABLE`. Znajdź wyzwalacze w phpMyAdmin (tabela → Wyzwalacze).
2. Z9. Utwórz wyzwalacz AFTER INSERT na produkty, który próbuje zaktualizować tabelę produkty (np. `UPDATE produkty SET stan = stan + 0 WHERE id_produktu = NEW.id_produktu`). Dodaj dowolny produkt. Zapisz błąd i czy produkt został dodany. Usuń ten wyzwalacz.
3. Z10. Utwórz tabelę `log_pozycji` (id, id_zamowienia, id_produktu, kiedy) i wyzwalacz AFTER DELETE na pozycje_zamowienia zapisujący usunięte pozycje. Usuń zamówienie 4 (jego pozycje usuwa kaskada ON DELETE CASCADE), a potem bezpośrednio pozycję produktu 7 z zamówienia 1. Ile wpisów jest w log_pozycji?
4. Z11. Usuń wyzwalacz z Z10 (DROP TRIGGER) i sprawdź SHOW TRIGGERS LIKE 'pozycje%'. Zaimportuj ponownie `sklep.sql`.

W sprawozdaniu umieść

- ☐ plik `.sql` z tabelami, wyzwalaczami i testami Z1–Z11
- ☐ zrzuty ekranu historii cen (Z3) i dziennika klientów po Z6 i Z7
- ☐ opis ograniczeń z Z9 i Z10
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Kiedy użyjesz wyzwalacza BEFORE, a kiedy AFTER? Podaj przykład z tego zadania.
2. Które z wartości OLD i NEW są dostępne w wyzwalaczach INSERT, UPDATE i DELETE? Które można zmieniać?
3. Dlaczego w Z7 zwykłe `OLD.miasto <> NEW.miasto` nie wykryje zmiany z „Warszawa” na NULL?
4. Wyzwalacz nie zadziałał przy kaskadowym usunięciu pozycji (Z10). Jak zapewnić pełny audyt usuwanych pozycji?

Kryteria oceny

Kryterium	Punkty
Historia cen — tabela, wyzwalacz i raport (Z1–Z3)	4
Audyt klientów: wyzwalacze AFTER i BEFORE z testami (Z4–Z6)	6
Poprawiony wyzwalacz z porównaniem $\Leftrightarrow$ (Z7)	2
Przegląd i ograniczenia wyzwalaczy (Z8–Z11)	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Wyzwalacze walidujące i zdarzenia (EVENT)

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Nauczenie się pilnowania reguł biznesowych wyzwalaczami (stan magazynu, dozwolone zmiany statusu, limit obniżki) oraz automatyzowania zadań okresowych zdarzeniami serwera.

Po zajęciach potrafisz:

- pisze wyzwalacze BEFORE, które blokują niedozwoloną zmianę poleceniem SIGNAL
- utrzymuje stan magazynu wyzwalaczami INSERT, UPDATE i DELETE na pozycjach zamówień
- włącza harmonogram zdarzeń i tworzy zdarzenia jednorazowe i cykliczne
- projektuje archiwizację starych danych w procedurze uruchamianej przez zdarzenie

Sprzęt i materiały

- komputer w pracowni z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza sklep zaimportowana z pliku `sklep.sql` (folder Pliki_do_zadan)

Oprogramowanie

- **XAMPP (Apache + MariaDB + phpMyAdmin)** — panel sterowania XAMPP, konsola: przycisk Shell albo `C:\xampp\mysql\bin\mysql.exe -u root`
apachefriends.org/download.html
- **phpMyAdmin** — zakładka SQL do uruchamiania zapytań; wchodzi w skład XAMPP
localhost/phpmyadmin
- **VS Code lub Notepad++** — do pisania pliku `.sql` z zapytaniami
code.visualstudio.com

Bezpieczeństwo

- Zaimportuj świeżą bazę sklep przed zadaniem i po nim — wyzwalacze zmieniają stany magazynowe, a archiwizacja usuwa zamówienia.
- Zdarzenie usuwające dane najpierw sprawdź ręcznie (CALL procedury) na kopii bazy. Po zajęciach usuń zdarzenia testowe i wyłącz harmonogram.
- `SET GLOBAL` zmienia ustawienie całego serwera — wykonuj je tylko na własnym serwerze XAMPP, nie na serwerze szkolnym.

Przebieg zadania

1 Stan magazynu pilnowany przez wyzwalacze

Wyzwalacz BEFORE INSERT na pozycje_zamowienia sprawdza stan i zmniejsza go. Błąd SIGNAL w wyzwalaczu przerywa całe polecenie.

```
DELIMITER //
DROP TRIGGER IF EXISTS trg_pozycje_stan //
CREATE TRIGGER trg_pozycje_stan
BEFORE INSERT ON pozycje_zamowienia
FOR EACH ROW
BEGIN
    DECLARE v_stan INT;
    SELECT stan INTO v_stan FROM produkty WHERE id_produkту = NEW.id_produkту;
    IF v_stan < NEW.ilosc THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Brak towaru na stanie';
    END IF;
    UPDATE produkty SET stan = stan - NEW.ilosc WHERE id_produkту = NEW.id_produkту;
END //
DELIMITER ;
```

1. Zaimportuj `sklep.sql`. Utwórz plik `r2_12_nazwisko.sql`.
2. Z1. Utwórz wyzwalacz `trg_pozycje_stan` (kod powyżej). Dodaj do zamówienia 9 pozycję: produkt 7, 5 sztuk, cena 129.00 i sprawdź stan produktu 7. Potem spróbuj dodać do zamówienia 11 produkt 3 (stan 0).
3. Z2. Jednym poleceniem INSERT spróbuj dodać do zamówienia 11 dwie pozycje: (produkt 7, 1 szt.) i (produkt 3, 1 szt.). Sprawdź stan produktu 7 i liczbę pozycji zamówienia 11. Czy pierwsza pozycja została dodana?
4. Z3. Utwórz wyzwalacz `trg_pozycje_zwrot` AFTER DELETE, który oddaje na stan ilość z usuniętej pozycji. Usuń pozycję z Z1 i sprawdź stan produktu 7.
5. Z4. Utwórz wyzwalacz `trg_pozycje_zmiana` BEFORE UPDATE: zabrania zmiany produktu w pozycji (SIGNAL), a przy zmianie ilości koryguje stan o różnicę i blokuje zwiększenie ponad stan. Uwaga: kolumna ilosc jest UNSIGNED — różnicę licz w zmiennej INT: `CAST(NEW.ilosc AS SIGNED) - CAST(OLD.ilosc AS SIGNED)`. Testy: zamówienie 12, produkt 7 → ilość 3; zamówienie 8, produkt 9 → ilość 10, potem 1; zmiana produktu 9 na 1 w zamówieniu 8.

2 Reguły dla statusu i ceny

1. Z5. Utwórz wyzwalacz `trg_zamowienia_status` BEFORE UPDATE na zamowienia, który nie pozwala zmienić statusu zamówienia wysłanego lub anulowanego. Sprawdź zmianę zamówienia 1 na 'nowe' i zamówienia 9 na 'opłacone'.
2. Z6. Utwórz wyzwalacz `trg_produkty_obnizka` BEFORE UPDATE na produkty, który nie odrzuca zmiany, ale ogranicza obniżkę ceny do 50% (ustawia NEW.cena). Zmień cenę produktu 2 na 100.00, a potem na 3000.00. Jakie ceny zapisano?
3. Z7. Zapisz stan produktu 11 i usuń zamówienie 12 (pozycje usuwa kaskada). Czy wyzwalacz `trg_pozycje_zwrot` oddał towar na stan? Porównaj z Z10 zadania 11.

3 Zdarzenia (EVENT)

Zdarzenia uruchamia wątek harmonogramu. W XAMPP (MariaDB) jest on domyślnie wyłączony, w MySQL 8.4 — włączony. Na stałe włącza go wpis `event_scheduler=ON` w sekcji [mysqld] pliku my.ini.

1. Z8. Sprawdź `SHOW VARIABLES LIKE 'event_scheduler'`. Utwórz tabelę `log_zdarzen` (id, kiedy DATETIME, opis VARCHAR(100)) i zdarzenie `ev_test` wykonywane co 10 sekund, które dopisuje wiersz do `log_zdarzen`. Wyświetl `SHOW EVENTS`. Włącz harmonogram `SET GLOBAL event_scheduler = ON;`, odczekaj ok. 30 sekund i policz wiersze.
2. Z9. Wyłącz zdarzenie (`ALTER EVENT ev_test DISABLE;`), odczekaj 10 sekund i sprawdź, że wierszy nie przybywa.
3. Z10. Utwórz tabele archiwum i procedurę `archiwizuj_anulowane(IN p_dni, OUT p_file)` (kod w ramce poniżej). Najpierw sprawdź ją ręcznie: `CALL` z `p_dni = 30`, potem zawartość obu tabel archiwum i liczbę zamówień. Wywołaj ją drugi raz — ile zamówień przeniesiono?
4. Z11. Utwórz zdarzenie `ev_archiwizacja` wykonywane codziennie o 2:00, zaczynając od jutra: `ON SCHEDULE EVERY 1 DAY STARTS CURRENT_DATE + INTERVAL 1 DAY + INTERVAL 2 HOUR DO CALL archiwizuj_anulowane(30, @file);`. Utwórz też zdarzenie jednorazowe `ev_sprzatanie` (`AT CURRENT_TIMESTAMP + INTERVAL 5 SECOND`), które usuwa wiersze z `log_zdarzen`. Wypisz zdarzenia z `information_schema.EVENTS` (typ, harmonogram, status). Po 10 sekundach sprawdź `log_zdarzen` i listę zdarzeń.
5. Z12. Usuń zdarzenia `ev_test` i `ev_archiwizacja` (`DROP EVENT`), wyłącz harmonogram (`SET GLOBAL event_scheduler = OFF;`) i zaimportuj ponownie `sklep.sql`.

4 Procedura archiwizacji do Z10

```
CREATE TABLE zamowienia_archiwum LIKE zamowienia;
CREATE TABLE pozycje_archiwum LIKE pozycje_zamowienia;
DELIMITER //
DROP PROCEDURE IF EXISTS archiwizuj_anulowane //
CREATE PROCEDURE archiwizuj_anulowane(IN p_dni INT, OUT p_file INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL; END;
    START TRANSACTION;
    INSERT INTO zamowienia_archiwum
        SELECT * FROM zamowienia
        WHERE status = 'anulowane' AND data_zamowienia < NOW() - INTERVAL p_dni DAY;
    SET p_file = ROW_COUNT();
    INSERT INTO pozycje_archiwum
        SELECT pz.* FROM pozycje_zamowienia pz
        JOIN zamowienia_archiwum za ON za.id_zamowienia = pz.id_zamowienia;
    DELETE FROM zamowienia
    WHERE id_zamowienia IN (SELECT id_zamowienia FROM zamowienia_archiwum);
    COMMIT;
END //
DELIMITER ;
```

W sprawozdaniu umieść

- ☐ plik `.sql` z wyzwalaczami, procedurą, zdarzeniami i testami Z1–Z12
- ☐ tabela testów wyzwalaczy (polecenie, oczekiwany wynik, wynik, stan magazynu)
- ☐ zrzuty ekranu `SHOW EVENTS` / `information_schema.EVENTS` i wyniku archiwizacji
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego w Z2 nie została dodana także pierwsza, poprawna pozycja?

2. Czym różni się wyzwalacz, który blokuje zmianę (Z5), od wyzwalacza, który ją poprawia (Z6)?
Który sposób jest bardziej przejrzysty dla użytkownika aplikacji?
3. Co się stało w Z7 i jak można to naprawić?
4. Co stanie się ze zdarzeniem jednorazowym po wykonaniu (ON COMPLETION NOT PRESERVE), a co, gdy w chwili planowanego uruchomienia harmonogram jest wyłączony?

Kryteria oceny

Kryterium	Punkty
Wyzwalacze stanu magazynu z testami (Z1–Z4)	6
Wyzwalacze reguł statusu i ceny, analiza kaskady (Z5–Z7)	4
Zdarzenia cykliczne i jednorazowe, archiwizacja (Z8–Z12)	7
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Transakcje i ACID — przelewy między kontami

Czas: 2 × 45 min

praca indywidualna (dwa okna konsoli na jednym komputerze)

Cel: Przećwiczenie sterowania transakcjami w MySQL/MariaDB i sprawdzenie w praktyce, co gwarantują niepodzielność, spójność, izolacja i trwałość (ACID).

Po zajęciach potrafisz:

- grupuje polecenia w transakcję i kończy ją poleceniem COMMIT albo ROLLBACK
- stosuje punkty zapisu (SAVEPOINT) do częściowego wycofania zmian
- wyjaśnia działanie autocommit i niejawnego zatwierdzania przez polecenia DDL
- rozpoznaje, że tabela MyISAM nie obsługuje transakcji
- pisze procedurę przelewu, która przy błędzie wycofuje całą transakcję

Sprzęt i materiały

- komputer z zainstalowanym XAMPP (MariaDB 10.4) lub MySQL 8.4
- plik `r2_13_bank.sql` z folderu `Pliki_do_zadan`

Oprogramowanie

- **XAMPP (Apache + MariaDB + PHP)** — konsola: przycisk „Shell” w XAMPP Control Panel, polecenie `mysql -u root`
apachefriends.org/download.html
- **phpMyAdmin** — tylko do importu pliku i podglądu tabel
localhost/phpmyadmin
- **Edytor tekstu (VS Code lub Notatnik++)** — plik `r2_13_transakcje.sql` z zapytaniami
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.07 znaczenie liter A, C, I, D oraz polecenia START TRANSACTION, COMMIT i ROLLBACK.

Bezpieczeństwo

- Plik `r2_13_bank.sql` usuwa i tworzy od nowa bazę `bank` — importuj go tylko na swoim komputerze. Bazy `sklep` nie zmieniaj.
- Transakcje wykonuj w konsoli `mysql`, a nie w phpMyAdmin: phpMyAdmin po każdym kliknięciu „Wykonaj” kończy połączenie, więc otwarta transakcja zostaje po cichu wycofana.
- Na końcu zajęć zamknij wszystkie otwarte transakcje (COMMIT lub ROLLBACK) — niezakończona transakcja blokuje wiersze innym sesjom.

Przebieg zadania

1 Przygotowanie bazy i dwóch konsol

Okno A to „aplikacja banku”, okno B to „inny użytkownik”, który podgląda dane.

1. Zaimportuj plik `r2_13_bank.sql` (phpMyAdmin → Import albo w konsoli `SOURCE` ścieżka/`r2_13_bank.sql`).
2. Otwórz dwa okna konsoli (dwa razy „Shell”) i w obu wpisz `mysql -u root`, a potem `USE bank;`. Ustaw okna obok siebie.
3. Z1. W oknie A sprawdź wartość zmiennej `@@autocommit`, wyświetl wszystkie konta i policz sumę sald. Zapisz sumę — to kontrola spójności na całe zajęcie.

2 COMMIT, ROLLBACK i błąd w środku transakcji

Przelew to zawsze dwa polecenia UPDATE: zdjęcie pieniędzy z jednego konta i dopisanie do drugiego.

1. Z2. W oknie A wykonaj w jednej transakcji przelew 200 zł z konta 1 na konto 2 i dopisz go do tabeli `operacje`. Zatwierdź. Sprawdź salda i sumę.
2. Z3. Rozpocznij przelew 1000 zł z konta 3 na konto 5, ale go nie kończ. Sprawdź salda w oknie A, a potem w oknie B. Wycofaj transakcję i sprawdź ponownie.
3. Z4. Rozpocznij transakcję, w której najpierw dopisujesz 5000 zł do konta 2, a potem zdejmujesz 5000 zł z konta 3. Zanotuj numer błędu. Sprawdź saldo konta 2 w tej samej transakcji, a następnie wykonaj ROLLBACK. Zapisz wniosek: czy błąd jednego polecenia wycofuje całą transakcję?
4. Z5. W jednej transakcji: przelej 100 zł z konta 4 na konto 5, utwórz punkt zapisu `po_pierwszym`, przelej 300 zł z konta 4 na konto 1, wróć do punktu zapisu i zatwierdź. Sprawdź salda kont 1, 4 i 5.

3 Autocommit, awaria i niejawne zatwierdzenie

1. Z6. W oknie A wyłącz autocommit i nalicz 1% odsetek na koncie 3 (`saldo * 1.01`). Sprawdź saldo w oknie B, potem w oknie A wykonaj COMMIT i sprawdź w B jeszcze raz. Włącz autocommit z powrotem.
2. Z7. Symulacja awarii: w oknie A rozpocznij transakcję i ustaw saldo konta 4 na 0. Nie zatwierdzaj — zamknij okno konsoli krzyżykiem. Otwórz nową konsolę i sprawdź saldo konta 4.
3. Z8. Rozpocznij transakcję, dopisz 50 zł do konta 5, utwórz tabelę `notatki (id INT)` i wykonaj ROLLBACK. Sprawdź saldo konta 5 i sumę sald. Wyjaśnij wynik, a potem przywróć poprawny stan (odejmij 50 zł i usuń tabelę `notatki`).

4 MyISAM i AUTO_INCREMENT a transakcje

1. Z9. Rozpocznij transakcję, zdejmij 500 zł z konta 1 w tabeli `konta_myisam` i wykonaj ROLLBACK. Wyświetl ostrzeżenie (`SHOW WARNINGS`) i saldo konta 1 w tej tabeli.
2. Z10. Rozpocznij transakcję, dodaj wiersz do `operacje` (z konta 4 na 5, 10 zł), odczytaj `LAST_INSERT_ID()` i wykonaj ROLLBACK. Zapamiętaj numer — wrócisz do niego w Z12.

5 Procedura przelewu z obsługą błędów

Uzupełnij szkielet procedury tak, żeby każdy błąd (CHECK, brak konta, ujemna kwota) wycofywał całą transakcję i był zgłaszany dalej. Przypomnij sobie DECLARE EXIT HANDLER, ROLLBACK i RESIGNAL z lekcji 2.05.

```
DELIMITER //
```

```
CREATE PROCEDURE przelew(IN p_z INT, IN p_na INT, IN p_kwota DECIMAL(12,2))
```

```
BEGIN
```

```
-- 1. handler: przy dowolnym błędzie SQL wycofaj i zgłoś błąd dalej
```

```
-- 2. START TRANSACTION
```

```
-- 3. zdejmij kwotę z konta p_z, dopisz do konta p_na
```

```
-- 4. jeśli drugi UPDATE nie zmienił żadnego wiersza (ROW_COUNT() = 0):
```

```
--     SIGNAL SQLSTATE '45000' z komunikatem 'Nie ma konta odbiorcy'
```

```
-- 5. zapisz operację w tabeli operacje, COMMIT
```

```
END //
```

```
DELIMITER ;
```

1. Z11. Utwórz procedurę `przelew`.
2. Z12. Wywołaj kolejno: `CALL przelew(4, 2, 250);`, `CALL przelew(5, 1, 1000);`, `CALL przelew(1, 99, 100);`, `CALL przelew(2, 3, -10);`. Dla każdego wywołania zapisz komunikat i sprawdź, czy salda się zmieniły. Wyświetl tabelę `operacje` — jaki numer dostał przelew 250 zł i dlaczego?
3. Z13. Policz sumę sald i porównaj z Z1. Wyjaśnij każdą różnicę (odsetki z Z6, operacja z Z8).
4. Z14. Trwałość: zatwierdź dowolny mały przelew procedurą, a w drugim oknie zostaw otwartą transakcję z niezatwierdzoną zmianą. Zatrzymaj i uruchom ponownie MySQL w XAMPP Control Panel (Stop → Start) i sprawdź, które zmiany przetrwały.
5. Zapisz wszystkie polecenia w pliku `r2_13_transakcje.sql` z komentarzami Z1–Z14.

W sprawozdaniu umieść

- ☐ tabela zapytań Z1–Z14 z treścią poleceń i wynikiem
- ☐ zrzuty ekranu dwóch okien konsoli z Z3 i Z6 (różne salda w oknie A i B)
- ☐ numer i treść błędu z Z4 oraz wnioski o wycofywaniu transakcji
- ☐ kod procedury `przelew` i wyniki czterech wywołań z Z12
- ☐ plik `r2_13_transakcje.sql`
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Wyjaśnij na przykładzie przelewu każdą z czterech właściwości ACID.
2. Dlaczego w Z4 saldo konta 2 było zwiększone mimo błędu i co by się stało, gdyby zamiast ROLLBACK wykonać COMMIT?
3. Które polecenia powodują niejawnie zatwierdzenie transakcji? Podaj trzy przykłady.
4. Dlaczego po ROLLBACK w Z10 numer AUTO_INCREMENT nie został ponownie użyty i czy to jest błąd?

Kryteria oceny

Kryterium	Punkty
Poprawne transakcje Z2–Z5 (COMMIT, ROLLBACK, SAVEPOINT)	5
Doświadczenia Z6–Z10 wykonane i poprawnie wyjaśnione	5
Działająca procedura przelew i analiza wywołań Z12	5
Kontrola spójności Z13 i test trwałości Z14	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Współbieżność: poziomy izolacji i blokady

Czas: 2 × 45 min

praca indywidualna (dwa okna konsoli A i B na jednym komputerze)

Cel: Zaobserwowanie w dwóch oknach konsoli problemów współbieżnego dostępu do danych i sprawdzenie, jak chronią przed nimi poziomy izolacji i blokady InnoDB.

Po zajęciach potrafisz:

- ustawia poziom izolacji sesji i rozpoznaje brudny odczyt, niepowtarzalny odczyt i fantom
- pokazuje utraconą aktualizację i usuwa ją atomowym UPDATE albo SELECT ... FOR UPDATE
- interpretuje błędy 1205 (lock wait timeout) i 1213 (deadlock)
- odczytuje informacje o blokadach z information_schema i SHOW ENGINE INNODB STATUS
- stosuje stałą kolejność blokowania, aby uniknąć zakleszczeń

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- plik `r2_13_bank.sql` z folderu Pliki_do_zadan (ten sam co w zadaniu 13)

Oprogramowanie

- **XAMPP — konsola mysql** — dwa okna „Shell”, w każdym `mysql -u root` apachefriends.org/download.html
- **Edytor tekstu (VS Code)** — plik `r2_14_wspolbieznosc.sql` code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.08 cztery poziomy izolacji i pojęcia: brudny odczyt, niepowtarzalny odczyt, fantom, utracona aktualizacja.

Bezpieczeństwo

- Zaimportuj od nowa `r2_13_bank.sql` — plik usuwa i tworzy bazę `bank`. Nie wykonuj ćwiczeń na bazie `sklep`.
- Po każdym scenariuszu zakończ transakcje w obu oknach (COMMIT lub ROLLBACK). Zawieszona transakcja trzyma blokady i zatrzymuje inne sesje.
- Poziom izolacji zmieniaj tylko poleceniem `SET SESSION ...` — `SET GLOBAL` zmienia ustawienie całego serwera.

Przebieg zadania

1 Przygotowanie

Zapisuj, w którym oknie wykonujesz polecenie (A albo B). Zapis „B:” w poleceniu oznacza okno B.

1. Zaimportuj `r2_13_bank.sql`. Otwórz dwa okna konsoli i w obu wykonaj `USE bank;`.
2. Z1. Sprawdź w oknie B domyślny poziom izolacji: w MariaDB `SELECT @@tx_isolation;`, w MySQL 8.4 `SELECT @@transaction_isolation;` oraz wartość `@@innodb_lock_wait_timeout`.

2 Poziomy izolacji

Poziom izolacji ustawiaj w oknie B przed START TRANSACTION, np. `SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;`.

1. Z2. Brudny odczyt — B: poziom READ UNCOMMITTED. A: rozpocznij transakcję i zdejmij 1000 zł z konta 1 (bez COMMIT). B: odczytaj saldo konta 1. A: ROLLBACK. B: odczytaj saldo jeszcze raz.
2. Z3. Powtórz Z2 z poziomem READ COMMITTED w oknie B. Co widzi B?
3. Z4. Niepowtarzalny odczyt — B: READ COMMITTED, START TRANSACTION i odczyt salda konta 2. A: dopisz 100 zł do konta 2 (zwykły UPDATE, autocommit). B: odczytaj saldo konta 2 ponownie w tej samej transakcji, potem COMMIT.
4. Z5. Powtórz Z4 z poziomem REPEATABLE READ. Sprawdź saldo w B przed COMMIT i po COMMIT.
5. Z6. Fantom — B: READ COMMITTED, START TRANSACTION, policz konta z saldem co najmniej 1000 zł. A: dodaj konto `(6, 'Maja Kamińska', 3000)`. B: policz ponownie, COMMIT.
6. Z7. Powtórz Z6 z poziomem REPEATABLE READ i kontem `(7, 'Tomasz Lewandowski', 4000)`. W oknie B wykonaj policzenie trzy razy: zwykłe, z dopiskiem `FOR UPDATE` i znowu zwykłe. Wyjaśnij różnicę (migawka MVCC a odczyt blokujący).

3 Utracona aktualizacja i FOR UPDATE

Symulujemy aplikację, która odczytuje saldo, liczy nową wartość „u siebie” i zapisuje liczbę.

1. Z8. W obu oknach START TRANSACTION i `SELECT saldo INTO @s FROM konta WHERE nr_konta = 3;`. A: `UPDATE konta SET saldo = @s - 100 WHERE nr_konta = 3; COMMIT;`. B: to samo z kwotą 200 i COMMIT. Jakie saldo powinno zostać, a jakie zostało? Przywróć saldo 2300.
2. Z9. Powtórz Z8, ale odczyt w obu oknach zakończ `FOR UPDATE`. Zaobserwuj, że B czeka, aż A zatwierdzi. Jaką wartość odczytał B i jakie jest saldo końcowe?
3. Z10. Zapisz, jak poprawić Z8 bez blokady jawnej — jednym atomowym poleceniem UPDATE — i wykonaj je w obu oknach.

4 Czekanie na blokadę i zakleszczenie

1. Z11. B: `SET SESSION innodb_lock_wait_timeout = 30;`. A: START TRANSACTION i zablokuj konto 4 przez `SELECT ... FOR UPDATE`. B: START TRANSACTION, dopisz 1 zł do konta 5, potem spróbuj zdjąć 1 zł z konta 4. Gdy B czeka, w oknie A wyświetl `trx_id, trx_state, trx_query` z `information_schema.INNODB_TRX`. Odczekaj do błędu w B i zanotuj jego numer. Sprawdź w B, czy zmiana konta 5 nadal istnieje. Zakończ obie transakcje ROLLBACK.
2. Z12. A: zablokuj konto 4 jak w Z11. B: spróbuj `SELECT ... FOR UPDATE NOWAIT` na koncie 4. Zakończ obie transakcje.
3. Z13. Zakleszczenie — A i B: START TRANSACTION. A: zdejmij 10 zł z konta 1. B: zdejmij 10 zł z konta 2. A: dopisz 10 zł do konta 2 (A czeka). B: dopisz 10 zł do konta 1. Zanotuj błąd w B, sprawdź w B saldo konta 2, zakończ obie transakcje COMMIT i wyświetl salda kont 1 i 2.
4. Z14. Wyświetl `SHOW ENGINE INNODB STATUS\G` i odszukaj sekcję LATEST DETECTED DEADLOCK. Wypisz, które polecenie czekało na którą blokadę.
5. Z15. Powtórz Z13, ale w obu oknach najpierw zmieniaj konto o niższym numerze. Czy zakleszczenie wystąpiło?

W sprawozdaniu umieść

- ☐ tabela zapytań Z1–Z15 z oznaczeniem okna (A/B) i wynikiem
- ☐ tabela porównania poziomów izolacji wypełniona na podstawie Z2–Z7
- ☐ zrzuty ekranu dwóch okien z Z8 (utracona aktualizacja) i Z13 (błąd 1213)
- ☐ fragment sekcji LATEST DETECTED DEADLOCK z Z14 z opisem
- ☐ plik `r2_14_wspolbieznosc.sql`

☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego brudny odczyt jest groźny? Podaj przykład decyzji podjętej na podstawie danych, które nigdy nie zostały zatwierdzone.
2. Czym różni się zwykły SELECT od SELECT ... FOR UPDATE w transakcji REPEATABLE READ?
3. Co serwer robi z transakcją po błędzie 1213, a co po błędzie 1205? Jak powinna zareagować aplikacja?
4. Podaj dwa sposoby uniknięcia utraconej aktualizacji w sklepie internetowym.

Kryteria oceny

Kryterium	Punkty
Scenariusze poziomów izolacji Z2–Z7 i tabela porównania	6
Utracona aktualizacja i jej usunięcie (Z8–Z10)	4
Blokady, timeout i zakleszczenie z analizą (Z11–Z15)	6
Odpowiedzi na pytania kontrolne	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Indeksy — EXPLAIN i pomiar czasu

Czas: 2 × 45 min

praca indywidualna

Cel: Zmierzenie, jak indeks zmienia czas wykonania zapytania na tabeli z milionem wierszy, i nauczenie się czytania planu EXPLAIN.

Po zajęciach potrafisz:

- mierzy czas zapytania w konsoli i porównuje wyniki przed i po założeniu indeksu
- zakłada i usuwa indeks (CREATE INDEX, DROP INDEX, SHOW INDEX)
- odczytuje z EXPLAIN kolumny type, possible_keys, key, rows i Extra
- wyjaśnia, dlaczego optymalizator czasem nie używa istniejącego indeksu
- ocenia koszt indeksów przy zapisie danych i zajmowane miejsce

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4, min. 1 GB wolnego miejsca
- plik `r2_15_demo.sql` z folderu Pliki_do_zadan

Oprogramowanie

- **XAMPP — konsola mysql** — czas zapytania widać pod wynikiem, np. `(0.105 sec)`
apachefriends.org/download.html
- **phpMyAdmin** — opcjonalnie: „Wykonanie zapytania trwało ... s” i przycisk „Wyjaśnij SQL”
localhost/phpmyadmin
- **Arkusz kalkulacyjny (LibreOffice Calc lub Excel)** — tabela i wykres czasów
libreoffice.org/download/download-libreoffice

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.09 zasadę działania indeksu B-tree i kolumny wyniku EXPLAIN.

Bezpieczeństwo

- Plik `r2_15_demo.sql` usuwa i tworzy od nowa bazę `demo`. Importuj go tylko na swoim komputerze.
- Na słabym komputerze lub przy braku miejsca zmień w pliku `SET @ile = 1000000;` na `200000` — nie przerywaj importu w połowie.
- Nie twórz indeksów na bazie `sklep` używanej w innych zadaniach — eksperymentuj tylko w bazie `demo`.

Przebieg zadania

1 Import i stan wyjściowy

Import 1 mln wierszy trwa od kilkunastu sekund do kilku minut — w tym czasie przeczytaj kroki 2 i 3.

```
-- konsola XAMPP (przycisk Shell):
mysql -u root
SOURCE C:/sciezka/do/r2_15_demo.sql
USE demo;
```

1. Zimportuj plik w konsoli (phpMyAdmin może przerwać tak duży import po przekroczeniu limitu czasu). Zapisz w sprawozdaniu, ile wierszy wygenerowano i ile trwał import.
2. Z1. Wyświetl szacowaną liczbę wierszy oraz rozmiar danych i indeksów w MB dla tabeli `odwiedziny` (`information_schema.TABLES`, kolumny `table_rows`, `data_length`, `index_length`) i listę indeksów (`SHOW INDEX`).

2 Zapytania bez indeksu

Każde zapytanie z pomiarem czasu wykonaj trzy razy i zapisz najlepszy wynik.

1. Z2. Policz odwiedzin klienta o numerze 4242.
2. Z3. Wyświetl plan EXPLAIN dla Z2. Zapisz type, key, rows i Extra.
3. Z4. Wyświetl wiersz o id = 500000 i jego plan EXPLAIN. Dlaczego to zapytanie jest szybkie już teraz?
4. Z5. Policz odwiedzin z 1 marca 2026 (od 00:00:00 do 23:59:59) i wyświetl plan EXPLAIN.
5. Z6. Wyświetl 5 najnowszych odwiedzin (id i czas) i plan EXPLAIN. Zwróć uwagę na Extra.

3 Zakładamy indeksy

1. Z7. Utwórz indeks `idx_klient` na kolumnie `id_klienta` i indeks `idx_czas` na kolumnie `czas`. Zapisz czas tworzenia każdego z nich.
2. Z8. Powtórz Z2 i Z3. Porównaj czas i plan z wynikami bez indeksu.
3. Z9. Powtórz Z5 i Z6. Jaki typ dostępu (type) pokazał EXPLAIN teraz?
4. Z10. Policz średni czas ładowania strony (`AVG(czas_ms)`) dla odwiedzin od 1 lutego 2026. Wyświetl EXPLAIN — czy serwer użył `idx_czas`? Zmierz czas tego samego zapytania z `FORCE INDEX (idx_czas)` po nazwie tabeli i wyjaśnij wynik.
5. Z11. Wykonaj `ANALYZE TABLE odwiedziny;` i powtórz Z1. Ile miejsca zajmują teraz indeksy?

4 Koszt indeksów i zadanie samodzielne

1. Z12. Utwórz dwie puste kopie tabeli: `kopia_bez` (bez indeksów poza kluczem głównym) i `kopia_z` (z czterema indeksami: `id_klienta`, `czas`, `strona`, `czas_ms`). Wstaw do każdej po 300 000 pierwszych wierszy z `odwiedziny` (`INSERT ... SELECT`) i porównaj czasy.
2. Z13. Samodzielnie: odwiedzin trwające ponad 2000 ms — policz je, zmierz czas, załóż odpowiedni indeks, zmierz ponownie i pokaż EXPLAIN. Na koniec usuń ten indeks oraz obie kopie z Z12.
3. Zapisz wszystkie polecenia w pliku `r2_15_indeksy.sql`. Czasy z Z2, Z5, Z6 i Z13 (bez i z indeksem) przedstaw w arkuszu na wykresie słupkowym. Nie usuwaj bazy `demo` — wykorzystasz ją w zadaniach 16 i 17.

W sprawozdaniu umieść

- ☐ liczba wierszy i czas importu
- ☐ tabela zapytań Z1–Z13 z wynikami
- ☐ tabela porównania czasów i planów EXPLAIN bez indeksu i z indeksem
- ☐ zrzuty ekranu EXPLAIN z Z3 i Z8

- ☐ wykres słupkowy czasów i wnioski
- ☐ plik `r2_15_indeksy.sql` i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Co oznaczają wartości ALL, index, range, ref i const w kolumnie type?
2. Dlaczego w Z10 optymalizator zrezygnował z indeksu, choć był na liście possible_keys?
3. Dlaczego indeksy spowalniają INSERT, UPDATE i DELETE?
4. Co znaczy „Using index” i „Using filesort” w kolumnie Extra?

Kryteria oceny

Kryterium	Punkty
Pomiary bez indeksu i z indeksem (Z2–Z9) w tabeli	6
Poprawne odczytanie planów EXPLAIN	4
Analiza Z10–Z12 (selektywność, rozmiar, koszt zapisu)	4
Zadanie samodzielne Z13	2
Wykres, wnioski i odpowiedzi na pytania	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Indeksy złożone, pokrywające i pełnotekstowe

Czas: 2 × 45 min

praca indywidualna

Cel: Zaprojektowanie indeksów złożonych i pokrywających dla konkretnych zapytań, rozpoznanie sytuacji, w których indeks nie działa, oraz wyszukiwanie słów w tekście indeksem FULLTEXT.

Po zajęciach potrafisz:

- dobiera kolejność kolumn w indeksie złożonym zgodnie z zasadą lewego prefiksu
- tworzy indeks pokrywający i rozpoznaje go w EXPLAIN („Using index”)
- przepisuje warunki z funkcją na kolumnie i z LIKE '%...' tak, by mogły użyć indeksu
- wyszukuje tekst przez MATCH ... AGAINST w trybie naturalnym i logicznym (BOOLEAN MODE)
- zna ograniczenia FULLTEXT: minimalna długość słowa, brak odmiany wyrazów

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza `demo` z zadania 15 (plik `r2_15_demo.sql`) i plik `r2_16_poradnik.sql` z folderu Pliki_do_zadan

Oprogramowanie

- **XAMPP** — konsola `mysql` — pomiar czasu zapytań
apacheFriends.org/download.html
- **phpMyAdmin** — import pliku `r2_16_poradnik.sql`, podgląd indeksów (zakładka Struktura)
localhost/phpmyadmin

Przygotowanie przed zajęciami

- Jeśli nie masz bazy `demo` z zadania 15, zaimportuj w konsoli plik `r2_15_demo.sql` (`SOURCE ścieżka/r2_15_demo.sql`).
- Przypomnij sobie z prezentacji 2.09 zasadę lewego prefiksu i działanie MATCH ... AGAINST.

Bezpieczeństwo

- Plik `r2_16_poradnik.sql` usuwa i tworzy od nowa bazę `poradnik`. Ćwicz tylko w bazach `demo` i `poradnik`, nie w `sklep`.
- Tworzenie indeksu na milionie wierszy trwa kilka sekund i obciąża dysk — nie przerywaj polecenia w trakcie.

Przebieg zadania

1 Indeks złożony i zasada lewego prefiksu

Baza `demo`. Analizowane zapytanie: odwiedziny strony `'/koszyk'` w marcu 2026 (`czas >= '2026-03-01' AND czas < '2026-04-01'`). Każdy pomiar czasu wykonaj trzy razy.

1. Z1. Wyświetl indeksy tabeli `odwiedziny`. Usuń wszystkie oprócz PRIMARY (np. `idx_klient` i `idx_czas` z zadania 15).
2. Z2. Policz odwiedziny strony `'/koszyk'` w marcu i ich średni `czas_ms`. Zapisz czas wykonania.
3. Z3. Utwórz indeks `idx_strona_czas` na kolumnach (strona, czas). Powtórz liczenie z Z2 (samo COUNT) i wyświetl EXPLAIN. Zapisz type, key_len i rows.
4. Z4. Wyświetl EXPLAIN dla dwóch zapytań: tylko `strona = '/koszyk'` oraz tylko odwiedziny z 1 marca. Zmierz czas drugiego. Którą część indeksu (key_len) wykorzystał serwer i dlaczego drugie zapytanie jest wolne?
5. Z5. Wyświetl EXPLAIN dla 5 najnowszych odwiedzin strony `'/koszyk'` (`ORDER BY czas DESC LIMIT 5`). Czy pojawiło się „Using filesort”?

2 Indeks pokrywający i kolejność kolumn

1. Z6. Wyświetl EXPLAIN i zmierz czas średniego `czas_ms` dla `'/koszyk'` w marcu. Utwórz indeks `idx_pokrywajacy` (strona, czas, czas_ms), powtórz pomiar i EXPLAIN. Porównaj kolumnę Extra. Usuń `idx_pokrywajacy`.
2. Z7. Utwórz indeks `idx_czas_strona` (czas, strona). Porównaj plany zapytania COUNT z Z3 z dopiskiem `FORCE INDEX (idx_czas_strona)` i `FORCE INDEX (idx_strona_czas)` po nazwie tabeli. Który indeks czyta mniej wierszy (rows) i który wybiera serwer sam?

3 Kiedy indeks nie działa

1. Z8. Policz odwiedziny z 1 marca warunkiem `DATE(czas) = '2026-03-01'`. Wyświetl EXPLAIN i czas. Przepisz warunek na zakres bez funkcji i porównaj.
2. Z9. Porównaj EXPLAIN dla `strona LIKE '/kosz%'` i `strona LIKE '%szyk'` (wyświetlane kolumny: id, czas). Porównaj kolumny type, possible_keys i rows.
3. Z10. Wyświetl EXPLAIN dla warunku `strona = '/koszyk' OR czas_ms = 100`. Dlaczego indeks nie został użyty?

4 Wyszukiwanie pełnotekstowe

Zaimportuj `r2_16_poradnik.sql` i przejdź do bazy `poradnik` (tabela `artykuły`: tytuł, treść, kategoria).

1. Z11. Znajdź artykuły, w których tytule lub treści występuje słowo „dysk”, operatorem LIKE. Wyświetl EXPLAIN.
2. Z12. Utwórz indeks `FULLTEXT ft_artykuły` na kolumnach (tytuł, treść). Wyszukaj w trybie naturalnym „dysk SSD” i wyświetl trafność (`MATCH ... AGAINST` także w liście kolumn) posortowaną malejąco. Pokaż EXPLAIN.
3. Z13. Spróbuj wyszukać „dysk” tylko w kolumnie `treść` (`MATCH(treść)`). Zanotuj błąd i wyjaśnij go.
4. Z14. W trybie BOOLEAN MODE znajdź: artykuły o dyskach, ale bez słowa SSD; artykuły zawierające jednocześnie SSD i NVMe; dokładną frazę „pasta termoprzewodząca”. Porównaj wynik frazy z wyszukiwaniem tych samych słów w trybie naturalnym.
5. Z15. Porównaj wyniki: tryb naturalny „procesora” i tryb logiczny `procesor*`; słowo „PC” przez MATCH i przez LIKE. Sprawdź `@innodb_ft_min_token_size` i wyjaśnij różnice.
6. Zapisz polecenia w pliku `r2_16_indeksy.sql`. Na koniec usuń indeks `idx_czas_strona` (indeks `idx_strona_czas` zostaw do zadania 17).

W sprawozdaniu umieść

- ☐ tabela zapytań Z1–Z15 z wynikami
- ☐ tabela planów EXPLAIN (key, key_len, rows, Extra) dla Z3–Z7
- ☐ zrzuty ekranu EXPLAIN z Z6 przed i po utworzeniu indeksu pokrywającego
- ☐ przepisane warunki z Z8 i wyjaśnienie, dlaczego działają szybciej
- ☐ wyniki wyszukiwania pełnotekstowego Z12–Z15 z wnioskami
- ☐ plik `r2_16_indeksy.sql` i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Na czym polega zasada lewego prefiksu? Dla jakich warunków WHERE przyda się indeks (nazwisko, imie)?
2. Kiedy indeks jest pokrywający i dlaczego przyspiesza zapytanie?
3. Dlaczego warunek `YEAR(czas) = 2026` nie korzysta ze zwykłego indeksu na `czas`? Jak go przepisać?
4. Czym różni się wyszukiwanie przez FULLTEXT od `LIKE '%słowo%'`? Podaj jedną zaletę i jedno ograniczenie FULLTEXT.

Kryteria oceny

Kryterium	Punkty
Indeks złożony i zasada lewego prefiksu (Z1–Z5)	5
Indeks pokrywający i kolejność kolumn (Z6–Z7)	4
Przypadki bez indeksu i poprawione zapytania (Z8–Z10)	3
Wyszukiwanie pełnotekstowe (Z11–Z15)	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Optymalizacja zapytań

Czas: 2 × 45 min

praca indywidualna

Cel: Znalezienie wolnych zapytań za pomocą dziennika i planu wykonania oraz przepisanie ich na szybsze odpowiedniki: stronicowanie bez OFFSET, złączenie zamiast podzapytania skorelowanego, jedno zapytanie zamiast N+1.

Po zajęciach potrafisz:

- włącza dziennik wolnych zapytań i odczytuje z niego czas oraz liczbę przeczytanych wierszy
- porównuje plan szacowany (EXPLAIN) z rzeczywistym (ANALYZE / EXPLAIN ANALYZE)
- zastępuje stronicowanie przez OFFSET stronicowaniem po kluczu (keyset)
- przepisuje podzapytanie skorelowane na złączenie i uzasadnia różnicę czasu
- rozpoznaje problem N+1 w kodzie PHP i usuwa go jednym zapytaniem z JOIN

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4, PHP 8) lub MySQL 8.4
- baza `demo` z zadania 15–16 i baza `sklep`
- plik `r2_17_zamowienia_n1.php` z folderu `Pliki_do_zadan`

Oprogramowanie

- **XAMPP — konsola mysql, Apache i PHP** — dziennik wolnych zapytań w `C:\xampp\mysql\data`
apachefriends.org/download.html
- **VS Code** — edycja skryptu PHP i podgląd pliku dziennika
code.visualstudio.com

Przygotowanie przed zajęciami

- Sprawdź, że baza `demo` ma indeks `idx_strona_czas` (strona, czas) z zadania 16. Jeśli nie — zaimportuj `r2_15_demo.sql` i utwórz go.
- Przypomnij sobie z prezentacji 2.10 kolumny EXPLAIN i pojęcie problemu N+1.

Bezpieczeństwo

- Zmiany `SET GLOBAL` dotyczą całego serwera — wykonuj je tylko na swoim XAMPP i na koniec przywróć wartości domyślne (dziennik wyłączony, `long_query_time = 10`).
- Dziennik wolnych zapytań zapisuje pełną treść zapytań (także dane klientów). Na serwerze produkcyjnym chroń ten plik i nie wysyłaj go nikomu.
- Skrypt PHP uruchamiaj tylko lokalnie (`localhost`) — zawiera dane logowania do bazy.

Przebieg zadania

1 Dziennik wolnych zapytań

W MariaDB nowa wartość `slow_query_log` obowiązuje dopiero w nowych połączeniach — po `SET GLOBAL` wyjdź z konsoli (`exit`) i połącz się ponownie. W MySQL 8.4 zmiana działa od razu.

1. Z1. Wyświetl zmienne `slow_query_log`, `long_query_time`, `slow_query_log_file` i `log_output` (`SHOW VARIABLES WHERE Variable_name IN (...)`).
2. Z2. Włącz dziennik (`slow_query_log = 1`), ustaw zapis do pliku i tabeli (`log_output = 'FILE, TABLE'`) oraz próg `long_query_time = 0.1` s. Połącz się ponownie i sprawdź wartości.
3. Z3. Wykonaj w bazie `demo` trzy zapytania: `SELECT SLEEP(0.3);`, liczenie odwiedzin z 1 marca warunkiem `DATE(czas) = '2026-03-01'` oraz to samo zapytanie z zakresem czasu. Odczytaj tabelę `mysql.slow_log` (`start_time`, `query_time`, `rows_examined`, `sql_text`) i otwórz plik dziennika w edytorze. Które zapytania trafiły do dziennika?

2 Plan szacowany i rzeczywisty

MariaDB: `ANALYZE SELECT ...` (kolumny `r_rows`, `r_filtered`). MySQL 8.4: `EXPLAIN ANALYZE SELECT ...` (actual rows, actual time).

1. Z4. Wykonaj `ANALYZE` (lub `EXPLAIN ANALYZE`) dla zapytania liczącego odwiedzin strony `'/koszyk'` z 1 marca. Porównaj `rows` z `r_rows`.
2. Z5. Wykonaj `ANALYZE` dla liczenia odwiedzin z `czas_ms BETWEEN 100 AND 110`. Ile wierszy serwer przeczytał (`r_rows`), a jaki procent przeszedł przez warunek (`r_filtered`)? Co to mówi o potrzebie indeksu?
3. Z6. Porównaj czas i `EXPLAIN` zapytań `SELECT *` oraz `SELECT id, czas` dla strony `'/koszyk'` w marcu (po trzy pomiary). Wyjaśnij różnicę w kolumnie Extra.

3 Stronicowanie i podzapytania

1. Z7. Wyświetl „stronę” 10 odwiedzin od pozycji 900 001 w kolejności id przez `LIMIT 10 OFFSET 900000`. Zmierz czas i pokaż `EXPLAIN (rows)`.
2. Z8. Zapisz to samo jako stronicowanie po kluczu: warunek `id > ostatnie_id_z_poprzedniej_strony` i `LIMIT 10`. Porównaj czas i plan.
3. Z9. Stronicowanie po czasie: kolejne 10 odwiedzin strony `'/koszyk'` po chwili `2026-06-01 12:00:00` (`ORDER BY czas`). Pokaż `EXPLAIN` — który indeks i jaki typ dostępu?
4. Z10. Podzapytanie skorelowane: dla odwiedzin strony `'/koszyk'` z 1 marca między 0:00 a 1:00 wyświetl id, id_klienta i liczbę wszystkich odwiedzin tego klienta (podzapytanie w liście `SELECT`). Zmierz czas i pokaż `EXPLAIN (DEPENDENT SUBQUERY)`.
5. Z11. Przepisz Z10 na złączenie z tabelą pochodną (`SELECT id_klienta, COUNT(*) ... GROUP BY id_klienta`). Sprawdź, że wynik jest taki sam, i porównaj czas.
6. Z12. W bazie `sklep` napisz dwiema metodami (podzapytanie w `SELECT` oraz `LEFT JOIN` z `GROUP BY`) liczbę zamówień każdego klienta, także klientów bez zamówień. Porównaj wyniki i plany.

4 Problem N+1 w aplikacji

1. Z13. Uruchom `r2_17_zamowienia_n1.php` (w przeglądarce z `htdocs` albo w konsoli `php r2_17_zamowienia_n1.php`). Zapisz liczbę zapytań i czas. Wskaż w kodzie, skąd bierze się ta liczba zapytań.
2. Z14. Zapisz kopię jako `r2_17_zamowienia_join.php` i zastąp pętlę z zapytaniami jednym zapytaniem z `JOIN` i `GROUP BY` (klient i wartość zamówienia). Porównaj wynik, liczbę zapytań i czas.
3. Z15. Wyłącz dziennik wolnych zapytań i przywróć `long_query_time = 10` oraz `log_output = 'FILE'`. Zapisz polecenia SQL w pliku `r2_17_optymalizacja.sql`.

W sprawozdaniu umieść

- ☐ tabela zapytań Z1–Z15 z wynikami i czasami
- ☐ zrzut ekranu tabeli `mysql.slow_log` lub fragment pliku dziennika z Z3
- ☐ porównanie OFFSET i keyset (czas, rows) oraz podzapytania i złączenia (czas)
- ☐ kod zapytania z `r2_17_zamowienia_join.php` i porównanie liczby zapytań
- ☐ pliki `r2_17_optymalizacja.sql` i `r2_17_zamowienia_join.php`
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego `LIMIT 10 OFFSET 900000` jest wolne, choć zwraca tylko 10 wierszy?
2. Czym różnią się kolumny rows i r_rows? Kiedy duża różnica jest sygnałem problemu?
3. Dlaczego podzapytanie skorelowane bez indeksu potrafi być tysiące razy wolniejsze od złączenia?
4. Co to jest problem N+1 i dlaczego jest groźny, gdy baza działa na innym serwerze niż PHP?

Kryteria oceny

Kryterium	Punkty
Dziennik wolnych zapytań — włączenie, odczyt, przywrócenie (Z1–Z3, Z15)	4
Analiza planów rzeczywistych i SELECT * (Z4–Z6)	3
Stronicowanie keyset i przepisanie podzapytań (Z7–Z12)	6
Usunięcie problemu N+1 w PHP (Z13–Z14)	4
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Silniki składowania oraz MySQL i MariaDB

Czas: 2 × 45 min

praca indywidualna

Cel: Porównanie w praktyce najważniejszych silników składowania (InnoDB, MyISAM, Aria, MEMORY, CSV, ARCHIVE) i poznanie różnic między MySQL a MariaDB.

Po zajęciach potrafisz:

- wyświetla dostępne silniki i zmienia silnik tabeli (SHOW ENGINES, ALTER TABLE ... ENGINE)
- sprawdza doświadczalnie obsługę transakcji i kluczy obcych w różnych silnikach
- porównuje rozmiar danych i plików tabel w katalogu danych serwera
- dobiera silnik do zastosowania (dane trwałe, tymczasowe, archiwum, eksport)
- wymienia różnice między MySQL i MariaDB istotne dla programisty

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- baza `demo` z zadania 15 (plik `r2_15_demo.sql`) i baza `sklep`

Oprogramowanie

- **XAMPP — konsola mysql i XAMPP Control Panel** — restart serwera: Stop/Start przy module MySQL apachefriends.org/download.html
- **Eksplorator plików** — katalog danych: `C:\xampp\mysql\data\silniki`
- **Dokumentacja MariaDB i MySQL** — MySQL: <https://dev.mysql.com/doc/refman/8.4/en/storage-engines.html>
mariadb.com/kb/en/storage-engines

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.11 cechy InnoDB, MyISAM i MEMORY oraz z zadania 13 wynik ROLLBACK na tabeli MyISAM.

Bezpieczeństwo

- Wszystkie tabele twórz w nowej bazie `silniki`. Nie zmieniaj silnika tabel w bazach `sklep` i `demo`.
- Nie usuwaj ani nie przenoś ręcznie plików z katalogu `mysql\data` — tylko je oglądaj. Usunięcie pliku `.ibd` lub `.frm` uszkadza bazę.
- Przed restartem serwera w XAMPP zakończ otwarte transakcje i zamknij połączenia (phpMyAdmin, konsola).

Przebieg zadania

1 Silniki dostępne na serwerze

1. Z1. Wyświetl `SHOW ENGINES` oraz zmienne `@@version`, `@@default_storage_engine` i `@@datadir`. Zapisz, które silniki obsługują transakcje.
2. Z2. Utwórz bazę `silniki` (utf8mb4_polish_ci) i tabelę `t_innodb` o strukturze tabeli `demo.odwiedziny` (id PRIMARY KEY, id_klienta z indeksem, strona, czas, czas_ms). Utwórz jej kopie `t_myisam`, `t_aria` (tylko MariaDB) i `t_memory` przez `CREATE TABLE ... LIKE t_innodb` i `ALTER TABLE ... ENGINE=...`.
3. Z3. Wstaw do każdej tabeli 100 000 pierwszych wierszy z `demo.odwiedziny` i zapisz czasy. Zanotuj błąd przy tabeli MEMORY i sprawdź `@@max_heap_table_size`. Usuń `t_memory`, zwiększ limit dla sesji (`SET SESSION max_heap_table_size = 64*1024*1024;`), utwórz ją ponownie i wstaw dane.

2 Rozmiar i pliki

1. Z4. Wykonaj `ANALYZE TABLE` dla tabel i wyświetl z `information_schema.TABLES` silnik, liczbę wierszy, rozmiar danych i indeksów w MB.
2. Z5. Otwórz katalog `C:\xampp\mysql\data\silniki` i zapisz, jakie pliki ma każda tabela i ile zajmują. Dlaczego tabela MEMORY ma tylko plik .frm?

3 Transakcje, klucze obce i ograniczenia

1. Z6. W jednej transakcji usuń z każdej z czterech tabel wiersze o id ≤ 1000 i wykonaj ROLLBACK. Wyświetl ostrzeżenie i liczbę wierszy w każdej tabeli.
2. Z7. Utwórz w MyISAM tabele `klienci_m` (id INT PRIMARY KEY) i `zam_m` (id INT PRIMARY KEY, id_klienta INT, FOREIGN KEY (id_klienta) REFERENCES klienci_m(id)). Wstaw zamówienie klienta 999 i wyświetl `SHOW CREATE TABLE zam_m`. Powtórz to samo w InnoDB (`klienci_i`, `zam_i`).
3. Z8. Spróbuj utworzyć tabelę MEMORY z kolumną typu TEXT. Zanotuj błąd.
4. Z9. Utwórz tabelę `eksport` (id INT NOT NULL, nazwa VARCHAR(100) NOT NULL, cena DECIMAL(10,2) NOT NULL) ENGINE=CSV, wstaw do niej monitory z `sklep.produkty` i otwórz plik `eksport.CSV` w Notatniku. Sprawdź, co się stanie, gdy dodasz PRIMARY KEY albo kolumnę dopuszczającą NULL.
5. Z10. ARCHIVE: jeśli `SHOW ENGINES` go nie pokazuje, w MariaDB wykonaj `INSTALL SONAME 'ha_archive'`. Utwórz `t_archive` (id, strona, czas — bez indeksów), wstaw 100 000 wierszy z `demo.odwiedziny`, porównaj rozmiar z InnoDB i spróbuj zmienić oraz usunąć jeden wiersz. (Jeśli wtyczki nie ma w Twojej instalacji, opisz ARCHIVE na podstawie dokumentacji.)

4 Restart serwera, konwersja i porównanie systemów

1. Z11. Zrestartuj MySQL w XAMPP Control Panel (Stop → Start) i policz wiersze w `t_memory`, `t_innodb` i `t_archive`. Wyświetl definicję `t_memory`.
2. Z12. Zmień silnik `t_myisam` na InnoDB (`ALTER TABLE`), zmierz czas i sprawdź silnik w `information_schema.TABLES`. Zastanów się, kiedy taka konwersja jest potrzebna po imporcie starej bazy.
3. Z13. Sprawdź funkcję specyficzną dla MariaDB: `SELECT * FROM seq_1_to_5;` (w MySQL zakończy się błędem). Na stronach mariadb.org i dev.mysql.com odszukaj aktualne wersje LTS obu systemów i datę końca wsparcia wersji z Twojego XAMPP.
4. Z14. Na koniec usuń bazę `silniki` (`DROP DATABASE silniki;`). Polecenia zapisz w pliku `r2_18_silniki.sql`.

W sprawozdaniu umieść

- ☐ tabela zapytań Z1–Z14 z wynikami
- ☐ tabela porównania silników (transakcje, FK, rozmiar, pliki, po restarcie)

- ☐ zrzut ekranu katalogu `data\silniki` z plikami tabel
- ☐ zawartość pliku `eksport.CSV`
- ☐ krótkie porównanie MySQL i MariaDB z aktualnymi wersjami LTS i źródłem
- ☐ plik `r2_18_silniki.sql` i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego InnoDB jest domyślnym silnikiem i do czego dziś używa się MyISAM lub Aria?
2. Do czego nadaje się MEMORY, a dlaczego nie wolno w nim trzymać zamówień klientów?
3. Co dzieje się z kluczem obcym w tabeli MyISAM i jakie problemy to powoduje po imporcie starej bazy?
4. Podaj dwie różnice między MySQL a MariaDB, które mogą przeszkodzić w przeniesieniu bazy z jednego systemu do drugiego.

Kryteria oceny

Kryterium	Punkty
Tabele w różnych silnikach, dane i rozmiary (Z1–Z5)	5
Doświadczenia z transakcjami, FK i ograniczeniami (Z6–Z10)	6
Restart, konwersja i porównanie MySQL/MariaDB (Z11–Z13)	4
Tabela porównania silników i wnioski	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Bezpieczne zapytania w aplikacji

Czas: 2 × 45 min

praca indywidualna

Cel: Poprawienie aplikacji PHP tak, aby dane od użytkownika nigdy nie stawały się częścią polecenia SQL, a aplikacja łączyła się z bazą kontem o minimalnych uprawnieniach i nie ujawniała szczegółów błędów.

Po zajęciach potrafisz:

- wyjaśnia, dlaczego sklepanie zapytania z danymi z formularza jest błędem
- stosuje zapytania przygotowane PDO (prepare, execute, parametry ? i :nazwa)
- zabezpiecza nazwy kolumn w ORDER BY białą listą
- tworzy konto aplikacji z minimalnymi uprawnieniami i sprawdza ich działanie
- obsługuje błędy bazy bez pokazywania ich użytkownikowi i stosuje htmlspecialchars

Sprzęt i materiały

- komputer z XAMPP (Apache, PHP 8, MariaDB 10.4)
- plik `r2_19_klienci_start.php` z folderu Pliki_do_zadan
- świeżo zaimportowana baza `sklep`

Oprogramowanie

- **XAMPP** — pliki PHP w `C:\xampp\htdocs`, dziennik błędów `C:\xampp\apache\logs\error.log`
apachefriends.org/download.html
- **VS Code** — edycja PHP
code.visualstudio.com
- **Dokumentacja PHP — PDO** — prepare, execute, PDOException
php.net/manual/pl/book.pdo.php
- **OWASP — SQL Injection Prevention Cheat Sheet** — lista zasad obrony
cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.12 zasadę „dane to nie kod” i budowę zapytania przygotowanego w PDO.

Bezpieczeństwo

- Testuj wyłącznie własną aplikację na `localhost`. Sprawdzanie zabezpieczeń cudzych stron i serwerów bez pisemnej zgody właściciela jest niezgodne z prawem.
- W zadaniu używamy tylko zwykłych danych (nazwiska z apostrofem, polskie znaki). Nie szukaj i nie wpisuj „ataków” — celem jest poprawny kod, a nie łamanie zabezpieczeń.
- Hasło konta aplikacji wymyśl sam (min. 12 znaków) i nie wpisuj go do sprawozdania — w sprawozdaniu zastąp je gwiazdkami.
- Nie wystawiaj XAMPP do sieci (domyślnie nasłuchuje lokalnie) i na koniec usuń konto testowe, jeśli nauczyciel tak poleci.

Przebieg zadania

1 Diagnostyka kodu startowego

Skopiuj `r2_19_klienci_start.php` do `C:\xampp\htdocs` i otwórz `http://localhost/r2_19_klienci_start.php`.

1. Z1. Wyszukaj klientów, których nazwisko zaczyna się od „Now”. Działa?
2. Z2. Dodaj klienta: Liam O'Connor, liam.oconnor@example.ie, Kraków. Zapisz komunikat. Wyszukaj nazwisko zaczynające się od „O”.
3. Z3. W kodzie zaznacz (komentarzem `// PROBLEM`) każde miejsce, w którym dane od użytkownika trafiają do SQL lub do HTML bez zabezpieczenia, oraz miejsca ujawniające szczegóły błędów. Wypisz je w sprawozdaniu (powinno być co najmniej 6).

2 Konto aplikacji z minimalnymi uprawnieniami

Wykonaj w konsoli `mysql -u root`.

1. Z4. Utwórz użytkownika `sklep_www@localhost` z własnym hasłem i nadaj mu tylko SELECT i INSERT na tabeli `sklep.klienci`. Wyświetl `SHOW GRANTS`.
2. Z5. Zaloguj się jako `sklep_www` (`mysql -u sklep_www -p sklep`) i sprawdź: SELECT z `klienci`, DELETE z `klienci`, SELECT z `zamowienia`, DROP TABLE `klienci`, SHOW DATABASES. Zapisz, które polecenia zostały odrzucone i jakim błędem.

3 Nowa wersja: PDO i zapytania przygotowane

Utwórz plik `klienci.php` (nie zmieniaj pliku startowego — porównasz oba).

```
$pdo = new PDO('mysql:host=localhost;dbname=sklep;charset=utf8mb4', 'sklep_www', '...', [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
]);
// dalej: prepare(...) z ? lub :nazwa, execute([...]) – bez sklejanego danych
```

1. Z6. Połącz się kontem `sklep_www` w bloku try/catch. Przy błędzie połączenia zapisz szczegóły funkcją `error_log()`, a użytkownikowi pokaż tylko „Serwis chwilowo niedostępny”. Ustaw `display_errors` na 0.
2. Z7. Napisz wyszukiwanie przez `prepare` z warunkiem `nazwisko LIKE ?` — znak `%` dodaj do wartości parametru, nie do treści SQL.
3. Z8. Napisz dodawanie klienta przez `prepare` z parametrami nazwanymi. Przed zapisem sprawdź: imię i nazwisko niepuste, e-mail poprawny (`filter_var(..., FILTER_VALIDATE_EMAIL)`). Puste miasto zapisz jako NULL.
4. Z9. Obsłuż powtórzony e-mail (błąd 1062, `$e->errorInfo[1]`) komunikatem „Ten e-mail jest już zarejestrowany.”, a inne błędy — ogólnym komunikatem i wpisem w dzienniku.
5. Z10. Sortowanie: nazwy kolumn nie da się przekazać parametrem — zastosuj białą listę (tablica dozwolonych kolumn, domyślnie `nazwisko`) i kierunek tylko ASC albo DESC.
6. Z11. Wszystkie dane wypisywane w HTML (wyniki i wartość pola wyszukiwania) przepuść przez `htmlspecialchars(..., ENT_QUOTES, 'UTF-8')`.

4 Testy i lista kontrolna

1. Z12. Przetestuj nową wersję: dodaj Liama O'Connora i Giulie D'Angelo, wyszukaj „O'”, dodaj drugi raz ten sam e-mail, wpisz niepoprawny e-mail, podaj w adresie `?sort=cena` i `?sort=miasto&kier=desc`. Zapisz wyniki w tabeli testów.
2. Z13. Sprawdź w `C:\xampp\apache\logs\error.log`, że szczegóły błędu 1062 trafiły do dziennika, a nie na stronę.
3. Z14. Wypełnij w sprawozdaniu listę kontrolną bezpiecznej aplikacji i porównaj obie wersje kodu.

W sprawozdaniu umieść

- ☐ lista problemów znalezionych w kodzie startowym (Z3)
- ☐ polecenia tworzenia konta aplikacji i wyniki testu uprawnień (Z4–Z5, hasło zamaskowane)
- ☐ kod `klienci.php` z komentarzami
- ☐ tabela testów Z12 i fragment dziennika błędów z Z13
- ☐ wypełniona lista kontrolna
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego nazwisko O'Connor zepsuło zapytanie w wersji startowej i dlaczego w wersji z PDO działa bez żadnego „escapowania”?
2. Dlaczego nazwy kolumny w ORDER BY nie można przekazać jako parametru zapytania przygotowanego i jak sobie z tym radzimy?
3. Jakie korzyści daje konto aplikacji z uprawnieniami tylko SELECT i INSERT zamiast konta root?
4. Dlaczego pokazywanie użytkownikowi treści błędu SQL jest złą praktyką? Gdzie powinny trafiać szczegóły błędów?

Kryteria oceny

Kryterium	Punkty
Diagnoza kodu startowego (Z1–Z3)	3
Konto aplikacji i test uprawnień (Z4–Z5)	3
Zapytania przygotowane, walidacja i obsługa błędów (Z6–Z9)	6
Biała lista sortowania i htmlspecialchars (Z10–Z11)	3
Testy, dziennik i lista kontrolna (Z12–Z14)	3
Odpowiedzi na pytania kontrolne	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

PHP i MySQLi — strona z danymi z bazy

Czas: 2 × 45 min

praca indywidualna (warunki jak na egzaminie: bez internetu poza dokumentacją)

Cel: Zbudowanie strony internetowej, która pobiera dane z bazy `sklep` funkcjami `mysqli`, wyświetla je w tabeli HTML i filtruje po kategorii wybranej w formularzu.

Po zajęciach potrafisz:

- łączy się z bazą funkcją `mysqli_connect` i ustawia kodowanie `utf8mb4`
- wykonuje zapytanie (`mysqli_query`) i wypisuje wiersze w pętli `mysqli_fetch_assoc` / `mysqli_fetch_row`
- buduje listę rozwijaną formularza z danymi z bazy
- filtruje dane według wartości z formularza zapytaniem przygotowanym `mysqli_prepare`
- tworzy układ strony z plikiem CSS zgodnie z opisem

Sprzęt i materiały

- komputer z XAMPP (Apache, PHP 8, MariaDB 10.4)
- świeżo zaimportowana baza `sklep` (plik `sklep.sql`)

Oprogramowanie

- **XAMPP** — pliki strony w `C:\xampp\htdocs\sklep_www`
apachefriends.org/download.html
- **phpMyAdmin** — testowanie zapytań Z1–Z6 i zrzuty
localhost/phpmyadmin
- **VS Code lub Notepad++** — edytor HTML, CSS i PHP
code.visualstudio.com
- **Dokumentacja PHP — MySQLi** — dozwolona w trakcie pracy
php.net/manual/pl/book.mysqli.php

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.13 kolejność: połączenie → zapytanie → pętla po wierszach → zamknięcie połączenia.

Bezpieczeństwo

- Pracuj na koncie root tylko lokalnie w XAMPP — na prawdziwym serwerze aplikacja używa osobnego konta z minimalnymi uprawnieniami (zadanie 19).
- Wartość z formularza (`$_GET['kat']`) przekazuj do bazy tylko jako parametr zapytania przygotowanego. Nazwy produktów wypisuj przez `htmlspecialchars` — zawierają cudzysłowy (np. Monitor 27").
- Nie zmieniaj danych w bazie — zadanie wymaga tylko odczytu.

Przebieg zadania

1 Zapytania w phpMyAdmin

Każde zapytanie wykonaj w zakładce SQL bazy `sklep`, zrób zrzut ekranu wyniku (nazwy plików `z1.png` ... `z6.png`) i zapisz treść w pliku `kwerendy.txt`.

1. Z1. Lista kategorii: identyfikator i nazwa, posortowane alfabetycznie po nazwie.
2. Z2. Wszystkie produkty: nazwa produktu, nazwa kategorii, cena i stan — także produkty bez kategorii; od najdroższego.
3. Z3. Produkty kategorii o identyfikatorze 2: nazwa, kategoria, cena, stan; od najdroższego.
4. Z4. Dla każdej kategorii: nazwa, liczba produktów i wartość magazynu (suma cena × stan); także kategorie bez produktów.
5. Z5. Produkty, których nie ma w magazynie (stan 0).
6. Z6. Liczba produktów i średnia cena (zaokrąglona do 2 miejsc) w każdej kategorii, tylko kategorie z co najmniej 3 produktami.

2 Układ strony i CSS

Utwórz folder `C:\xampp\htdocs\sklep_www` z plikami `produkty.php` i `styl.css`. Stronę oglądaj pod adresem `http://localhost/sklep_www/produkty.php`.

1. Strona ma tytuł karty „Sklep komputerowy — produkty”, język polski i kodowanie UTF-8.
2. Baner (header): tło `#1F4E79`, biały tekst, wyśrodkowany, nagłówek `h1` „Sklep komputerowy”.
3. Panel lewy (aside): 25% szerokości, wysokość 600 px, tło `#DDEBF7`; panel prawy (main): 75% szerokości, wysokość 600 px. Panele obok siebie.
4. Stopka: jak baner, z tekstem „Stronę wykonał: ” i Twoim imieniem i nazwiskiem.
5. Tabela: obramowanie 1 px ciągłe `#1F4E79`, scalone krawędzie, nagłówki z tłem `#9BC2E6`, podświetlenie wiersza po najechaniu kursorem (`#F2F2F2`). Klasa `brak` — czerwony tekst.

3 Skrypty PHP z mysqli

Używaj funkcji w stylu proceduralnym: `mysqli_connect`, `mysqli_set_charset`, `mysqli_query`, `mysqli_fetch_assoc` / `mysqli_fetch_row`, `mysqli_num_rows`, `mysqli_prepare`, `mysqli_close`.

1. Skrypt 1 (panel lewy): formularz metodą GET z listą rozwijaną `kat`. Pierwsza opcja „wszystkie” (wartość 0), kolejne — kategorie z zapytania Z1 (wartość = id, tekst = nazwa) i przycisk „Pokaż”.
2. Skrypt 2 (panel lewy, pod formularzem): nagłówek „Podsumowanie” i lista punktowana z wyników zapytania Z4 w formacie „Laptopy: produktów 3, wartość 61 584,00 zł” (`number_format`).
3. Skrypt 3 (panel prawy): jeśli nie wybrano kategorii — wynik zapytania Z2, jeśli wybrano — Z3 z parametrem z formularza (`mysqli_prepare`, `mysqli_stmt_bind_param` z typem `i`, `mysqli_stmt_execute`, `mysqli_stmt_get_result`). Nad tabelą wypisz „Liczba produktów: N” (`mysqli_num_rows`).
4. W tabeli: produkt, kategoria (dla NULL — „bez kategorii”), cena w formacie „1 149,00 zł”, stan. Produkty ze stanem 0 dostają klasę `brak`, a w kolumnie stan słowo „brak”.
5. Na końcu skryptu zamknij połączenie. Sprawdź stronę dla „wszystkie”, „Monitory” i „Drukarki” oraz po ręcznym wpisaniu w adresie `?kat=abc` (strona ma pokazać wszystkie produkty bez błędu).

4 Oddanie pracy

1. Zrób zrzut ekranu całej strony dla kategorii „Monitory” (`strona.png`) i sprawdź poprawność HTML w walidatorze `https://validator.w3.org/` (wklej kod źródłowy z przeglądarki).
2. Spakuj folder `sklep_www` razem z plikami `kwerendy.txt` i zrzutami do archiwum `nazwisko_r2_20.zip`.

W sprawozdaniu umieść

- ☐ plik `kwerendy.txt` z zapytaniami Z1–Z6 i zrzuty wyników
- ☐ pliki `produkty.php` i `styl.css`
- ☐ zrzut ekranu strony dla kategorii „Monitory”
- ☐ wynik walidacji HTML (zrzut lub informacja o błędach)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różnią się `mysqli_fetch_assoc`, `mysqli_fetch_row` i `mysqli_fetch_array`?
2. Dlaczego w zapytaniu Z2 trzeba użyć LEFT JOIN, a w Z3 wystarczy JOIN?
3. Po co rzutowanie `(int)$GET['kat']` i zapytanie przygotowane, skoro w formularzu są tylko liczby?
4. Co się stanie, jeśli pominiesz `mysqli_set_charset($db, 'utf8mb4')`?

Kryteria oceny

Kryterium	Punkty
Zapytania Z1–Z6 ze zrzutami	6
Układ strony i CSS zgodne z opisem	4
Skrypt 1 i 2 (formularz z bazy, podsumowanie)	4
Skrypt 3 (tabela, filtr przez <code>mysqli_prepare</code> , klasa brak)	4
Odpowiedzi na pytania kontrolne	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

PHP i PDO — CRUD produktów

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zbudowanie w PHP i PDO kompletnego panelu do zarządzania produktami sklepu: odczyt, dodawanie, edycja i usuwanie (CRUD) z zapytaniami przygotowanymi.

Po zajęciach potrafisz:

- łączy się z bazą przez PDO z trybem `ERRMODE_EXCEPTION` i kodowaniem `utf8mb4`
- wykonuje `SELECT`, `INSERT`, `UPDATE` i `DELETE` zapytaniami przygotowanymi (`prepare/execute`)
- stosuje wzorzec `Post/Redirect/Get` i usuwa dane tylko metodą `POST`
- waliduje dane formularza po stronie serwera i wypisuje je przez `htmlspecialchars`
- obsługuje błąd naruszenia klucza obcego (1451) przy usuwaniu

Sprzęt i materiały

- komputer z XAMPP (Apache, PHP 8, MariaDB 10.4)
- świeżo zaimportowana baza `sklep`

Oprogramowanie

- **XAMPP** — folder projektu
`C:\xampp\htdocs\panel`
[apachefriends.org/download.html](https://www.apachefriends.org/download.html)
- **VS Code** — edycja PHP
code.visualstudio.com
- **Dokumentacja PHP — PDO** —
`PDO::prepare`, `PDOStatement::execute`,
`fetch`, `fetchAll`, `rowCount`
php.net/manual/pl/book.pdo.php

Przygotowanie przed zajęciami

- Zaimportuj ponownie `sklep.sql` (zadanie zmienia tabelę `produkty`).
- Przypomnij sobie z prezentacji 2.12 i 2.13 budowę połączenia PDO i zapytań przygotowanych.

Bezpieczeństwo

- Każda wartość z `$_GET` i `$_POST` trafia do SQL wyłącznie jako parametr `execute()`. Nie sklejaj zapytań z danymi.
- Usuwanie wykonuj tylko metodą `POST` (formularz z przyciskiem) — link `GET` mógłby zostać „kliknięty” przez przeglądarkę lub robota.
- Panel nie ma logowania, więc działa tylko lokalnie. Nie publikuj go w internecie w tej postaci.
- Przed testami usuwania zrób kopię tabeli: `CREATE TABLE produkty_kopia AS SELECT * FROM produkty;`

Przebieg zadania

1 Połączenie i lista (Read)

Struktura projektu: `połaczenie.php`, `index.php`, `formularz.php`, `dodaj.php`, `edytuj.php`, `usun.php`.

```
<?php
// polaczenie.php
$pdo = new PDO('mysql:host=localhost;dbname=sklep;charset=utf8mb4', 'root', '', [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES  => false,
]);
function h($s) { return htmlspecialchars((string)$s, ENT_QUOTES, 'UTF-8'); }
```

1. Z1. Utwórz `połaczenie.php` jak wyżej i dołączaj go w pozostałych plikach przez `require`.
2. Z2. W `index.php` wyświetl tabelę produktów: id, nazwa, nazwa kategorii (LEFT JOIN, brak — „—”), cena, stan i kolumna akcji: link „edytuj” (`edytuj.php?id=...`) i przycisk „usuń” w formularzu POST do `usun.php` z ukrytym polem `id` i potwierdzeniem `confirm()`.
3. Z3. Nad tabelą wyświetl komunikat zależny od parametru `?msg=` (dodano, zapisano, usunięto, w_zamowieniach). Tekst komunikatu wybieraj z tablicy w PHP — nie wypisuj wartości z adresu.

2 Dodawanie (Create)

1. Z4. W `formularz.php` napisz funkcję `waliduj(array $d): array` zwracającą listę błędów: nazwa niepusta i do 100 znaków, cena liczbą > 0 , stan liczbą całkowitą ≥ 0 (`filter_var` z `FILTER_VALIDATE_INT`).
2. Z5. Napisz funkcję `formularz()` wypisującą pola: nazwa, lista kategorii z bazy (pierwsza opcja „— brak —” z pustą wartością), cena, stan. Pola mają zachować wpisane wartości po błędzie walidacji (`value` przez `h()`).
3. Z6. W `dodaj.php` po poprawnej walidacji wykonaj INSERT z parametrami nazwanymi (pusta kategoria $\rightarrow$ NULL, data dodania `CURDATE()`) i przekieruj `header('Location: index.php?msg=dodano');` `exit;`.
4. Z7. Dodaj produkt `Kamera internetowa 4K "Pro"` (kategoria Akcesoria, 349.99 zł, stan 8). Sprawdź, że cudzysłowy w nazwie wyświetlają się poprawnie, i odśwież stronę po dodaniu (F5) — czy produkt dodał się drugi raz?

3 Edycja (Update) i usuwanie (Delete)

1. Z8. W `edytuj.php` dla GET pobierz produkt po id zapytaniem przygotowanym; gdy go nie ma — kod 404 i komunikat „Nie ma takiego produktu.”. Dla POST zwaliduj dane, wykonaj UPDATE i przekieruj z `msg=zapisano`.
2. Z9. Zmień kamerze nazwę na `Kamera internetowa 4K`, cenę na 299 zł i stan na 10. Sprawdź zmiany w phpMyAdmin.
3. Z10. W `usun.php` przyjmij tylko POST (dla innej metody kod 405). Wykonaj DELETE w bloku try/catch; przy błędzie 1451 (`$e->errorInfo[1]`) przekieruj z `msg=w_zamowieniach`.
4. Z11. Usuń kamerę, a potem spróbuj usunąć „Laptop Vertex 14” (id 1). Otwórz w przeglądarce `usun.php?id=15` — co się stało?
5. Z12. Sprawdź walidację: pusta nazwa, cena -5, stan 1.5. Wszystkie trzy błędy mają się wyświetlić naraz, a wpisane wartości pozostać w polach.

4 Sprawdzenie i oddanie

1. Z13. Wyświetl w phpMyAdmin wynik `SELECT COUNT(*), MAX(id_produktu) FROM produkty;` i porównaj ze stanem początkowym. Wyjaśnij, dlaczego nowy produkt dostałby id 17, a nie 16.
2. Spakuj folder `panel` do archiwum `nazwisko_r2_21.zip` i dołącz zrzuty ekranu listy, formularza z błędami i komunikatu o produkcie w zamówieniach.

W sprawozdaniu umieść

- ☐ pliki projektu (6 plików PHP) w archiwum zip
- ☐ zrzuty ekranu: lista produktów, formularz z błędami walidacji, komunikat po próbie usunięcia produktu z zamówień
- ☐ tabela testów Z7–Z13 z wynikami
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Na czym polega wzorzec Post/Redirect/Get i jaki problem rozwiązuje (Z7)?
2. Dlaczego usuwanie nie powinno być wykonywane linkiem GET?
3. Co oznacza błąd 1451 i jak inaczej można by rozwiązać usuwanie produktów, które były już zamawiane?
4. Dlaczego walidacja w JavaScript lub atrybut `required` w HTML nie wystarczy i trzeba sprawdzać dane w PHP?

Kryteria oceny

Kryterium	Punkty
Połączenie PDO i lista z komunikatami (Z1–Z3)	3
Dodawanie z walidacją i PRG (Z4–Z7)	5
Edycja z obsługą braku produktu (Z8–Z9)	4
Usuwanie POST z obsługą błędu 1451 (Z10–Z12)	4
Testy, zrzuty i odpowiedzi na pytania	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

PHP — transakcje, błędy i wzorzec repozytorium

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zaprogramowanie w PHP składania zamówienia jako jednej transakcji, która przy każdym błędzie wycofuje wszystkie zmiany, oraz uporządkowanie kodu dostępu do bazy w klasach.

Po zajęciach potrafisz:

- wykonuje kilka poleceń SQL w transakcji PDO (`beginTransaction`, `commit`, `rollBack`)
- obsługuje wyjątki `PDOException` i własne wyjątki, przekazując błąd dalej po wycofaniu transakcji
- przechowuje dane logowania do bazy w pliku konfiguracyjnym poza katalogiem publicznym
- wydziela zapytania SQL do klasy repozytorium i używa jej w kodzie aplikacji
- blokuje odczytany wiersz (`SELECT ... FOR UPDATE`), aby uniknąć sprzedaży nieistniejącego towaru

Sprzęt i materiały

- komputer z XAMPP (Apache, PHP 8, MariaDB 10.4)
- świeżo zaimportowana baza `sklep`

Oprogramowanie

- **XAMPP** — PHP w konsoli:
`C:\xampp\php\php.exe` (przycisk „Shell”, polecenie `php`)
apachefriends.org/download.html
- **VS Code** — projekt z podkatalogiem `src`
code.visualstudio.com
- **Dokumentacja PHP — transakcje PDO** — `beginTransaction`, `commit`, `rollBack`, `inTransaction`
php.net/manual/pl/pdo.transactions.php

Przygotowanie przed zajęciami

- Zaimportuj ponownie `sklep.sql` — testy zmieniają stany magazynowe i dodają zamówienia.
- Przypomnij sobie procedurę `zloz_zamowienie` z prezentacji 2.07 oraz klasy i konstruktory w PHP.

Bezpieczeństwo

- Plik `config.php` z hasłem trzymaj poza `htdocs` (np. `C:\xampp\sklep_config`) — pliki w `htdocs` może pobrać każdy, kto zna adres. W projekcie z Git dopisz go do `.gitignore`.
- Komunikaty dla użytkownika nie mogą zawierać treści błędów SQL — szczegóły zapisuj funkcją `error_log()`.
- Przed testami zrób kopię: `CREATE TABLE produkty_kopia AS SELECT * FROM produkty;` albo zaimportuj `sklep.sql` po zajęciach.

Przebieg zadania

1 Konfiguracja i połączenie

Struktura plików:

```
C:\xampp\sklep_config\config.php      <- dane logowania (poza htdocs)
C:\xampp\htdocs\sklep22\zamow.php     <- skrypt testowy
C:\xampp\htdocs\sklep22\src\Baza.php
C:\xampp\htdocs\sklep22\src\ProduktRepozytorium.php
C:\xampp\htdocs\sklep22\src\ZamowienieSerwis.php
```

1. Z1. Utwórz `config.php`, który zwraca (`return [...]`) tablicę z kluczami `dsn`, `uzytkownik`, `haslo`.
2. Z2. Napisz klasę `Baza` z metodą statyczną `polacz(): PDO`, która wczytuje konfigurację (`require __DIR__ . '/.././.././sklep_config/config.php'`) i tworzy PDO z `ERRMODE_EXCEPTION`, `FETCH_ASSOC` i bez emulacji zapytań przygotowanych.
3. Z3. Sprawdź w przeglądarce, że adres `http://localhost/sklep_config/config.php` nie istnieje (404), a `http://localhost/sklep22/zamow.php` działa (na razie może wypisać tylko „OK”).

2 Repozytorium produktów

Repozytorium to klasa, która zbiera w jednym miejscu wszystkie zapytania dotyczące jednej tabeli. Reszta aplikacji nie pisze SQL — wywołuje metody.

1. Z4. Napisz klasę `ProduktRepozytorium` z konstruktorem przyjmującym `PDO` i metodami: `znajdz(int $id): ?array`, `wKategorii(int $idKategorii): array`.
2. Z5. Dodaj metodę `znajdzDoSprzedazy(int $id): ?array` z `SELECT ... FOR UPDATE` (do użycia w transakcji) oraz `zmniejszStan(int $id, int $ile): void` z warunkiem `stan >= ?` — gdy `rowCount()` nie jest równe 1, rzuć `RuntimeException`.
3. Z6. Przetestuj w `zamow.php`: wypisz nazwę i cenę produktu 5 oraz liczbę produktów w kategorii 4.

3 Zamówienie w transakcji

1. Z7. Napisz klasę `ZamowienieSerwis` z metodą `zloz(int $idKlienta, array $pozycje): int` (`$pozycje = [id_produkta => ilość]`). Pusta tablica — `InvalidArgumentException` jeszcze przed transakcją.
2. Z8. W transakcji: INSERT do `zamowienia` (status 'nowe', `NOW()`), `lastInsertId()`, a dla każdej pozycji: odczyt produktu z blokadą, sprawdzenie, czy istnieje i czy stan wystarcza (inaczej `RuntimeException` z czytelnym komunikatem), INSERT pozycji z aktualną ceną i zmniejszenie stanu. Na końcu `commit()` i zwrot numeru zamówienia.
3. Z9. Cały kod transakcji umieść w `try`; w `catch (Throwable $e)` wywołaj `rollback()` i rzuć wyjątek dalej (`throw $e;`).
4. Z10. W `zamow.php` wykonaj pięć testów, każdy w osobnym try/catch (PDOException: zapis do dziennika i numer błędu, Exception: komunikat): T1 klient 3: produkt 7 × 2 i 11 × 1; T2 klient 3: produkt 8 × 1 i 12 × 10; T3 klient 3: produkt 7 × 1 i 999 × 1; T4 klient 99: produkt 7 × 1; T5 klient 3 bez pozycji. Na koniec wypisz stany produktów 7, 8, 11, 12 i liczbę zamówień.

4 Sprawdzenie w bazie

1. Z11. W phpMyAdmin wyświetl zamówienia o numerze większym niż 12 i ich pozycje. Czy po nieudanych testach zostały „puste” zamówienia albo zmienione stany?
2. Z12. Dodaj jedno zamówienie ręcznie (`INSERT INTO zamowienia (id_klienta, data_zamowienia) VALUES (1, NOW());`) i sprawdź `LAST_INSERT_ID()`. Wyjaśnij lukę w numeracji.
3. Z13. Sprawdź, co by się stało bez transakcji: zakomentuj `beginTransaction`, `commit` i `rollback`, zaimportuj bazę ponownie, uruchom testy i porównaj wynik Z11. Przywróć transakcję.

W sprawozdaniu umieść

- ☐ kod czterech plików PHP i `config.php` (z zamaskowanym hasłem)
- ☐ wynik działania `zamow.php` (zrzut lub tekst z konsoli)
- ☐ wyniki zapytań Z11 i Z12 oraz wyjaśnienie luki w numeracji
- ☐ porównanie działania z transakcją i bez niej (Z13)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

- Dlaczego po `rollback()` wyjątek jest rzucany dalej, zamiast od razu wypisać komunikat w klasie serwisu?
- Po co `SELECT ... FOR UPDATE` w `znajdzDoSprzedazy` i dlaczego w `zmniejszStan` jest jeszcze warunek `stan >= ??`?
- Jakie korzyści daje repozytorium w porównaniu z zapytaniami SQL rozrzuconymi po plikach strony? Jak ten pomysł rozwijają biblioteki ORM (np. Doctrine, Eloquent)?
- Dlaczego plik z hasłem do bazy nie powinien leżeć w katalogu `htdocs`?

Kryteria oceny

Kryterium	Punkty
Konfiguracja poza <code>htdocs</code> i klasa Baza (Z1–Z3)	3
ProduktRepozytorium z metodami i testem (Z4–Z6)	4
Transakcja w ZamowienieSerwis z obsługą błędów (Z7–Z9)	6
Testy T1–T5 i sprawdzenie w bazie (Z10–Z13)	4
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Replikacja źródło–replika

Czas: 2 × 45 min

w parach: wariant A — dwa komputery w sieci pracowni; wariant B — dwie instancje na jednym komputerze

Cel: Skonfigurowanie replikacji asynchronicznej źródło–replika, sprawdzenie, że zmiany ze źródła trafiają na replikę, oraz przeciwczenie zatrzymania, wznowienia i naprawy przerwanej replikacji.

Po zajęciach potrafisz:

- włącza dziennik binarny i ustawia unikalne `server_id`
- tworzy konto replikacji i kopię startową z pozycją dziennika (`mysqldump --master-data`)
- podłącza replikę poleceniem `CHANGE MASTER TO` (MariaDB) / `CHANGE REPLICATION SOURCE TO` (MySQL 8.4)
- odczytuje `SHOW SLAVE STATUS` / `SHOW REPLICA STATUS` i rozpoznaje błędy replikacji
- wyjaśnia rolę `read_only` na replice i różnicę między replikacją a kopią zapasową

Sprzęt i materiały

- dwa komputery z XAMPP w jednej sieci (wariant A) albo jeden komputer z XAMPP i ok. 1 GB wolnego miejsca (wariant B)
- baza `sklep` na źródle

Oprogramowanie

- **XAMPP (MariaDB 10.4)** — plik konfiguracyjny `C:\xampp\mysql\bin\my.ini`
[apachefriends.org/download.html](https://www.apachefriends.org/download.html)
- **Wiersz polecenia (cmd)** — `mysql`, `mysqldump` i `mysqld` z katalogu `C:\xampp\mysql\bin`
- **Dokumentacja MariaDB — Setting Up Replication** — MySQL 8.4:
<https://dev.mysql.com/doc/refman/8.4/en/replication-configuration.html>
mariadb.com/kb/en/setting-up-replication

Przygotowanie przed zajęciami

- Przypomnij sobie z prezentacji 2.14 pojęcia: dziennik binarny, pozycja w dzienniku, wątek I/O i wątek SQL repliki.
- Wariant A: ustal z nauczycielem adresy IP obu komputerów (`ipconfig`). Wariant B: sprawdź, że port 3308 jest wolny.

Bezpieczeństwo

- Przed zmianą `my.ini` zrób jego kopię (`my.ini.bak`). Pliki danych kopiuj tylko przy zatrzymanym serwerze MySQL.
- Konto replikacji ma tylko uprawnienie `REPLICATION SLAVE` i silne hasło. W wariantcie A zezwól w Zaporze Windows na port 3306 tylko z adresu komputera partnera i usuń tę regułę po zajęciach.
- Nie wystawiaj serwera do internetu i nie używaj konta root do replikacji.
- Na replice nie zmieniaj danych kontem root — omija ono `read_only` i rozspójnia replikę (patrz Z12).

Przebieg zadania

1 Konfiguracja serwerów

Źródło = komputer (lub instancja) z oryginalną bazą sklep. W MariaDB 10.4 używamy poleceń z MASTER/SLAVE; MariaDB 10.5+ przyjmuje też START REPLICA i SHOW REPLICA STATUS.

```
# źródło: my.ini, sekcja [mysqld] (zatrzymaj MySQL, popraw, uruchom)
server-id      = 1
log-bin        = mysql-bin
binlog_format  = ROW

# wariant B: plik C:\xampp\mysql\my2.ini dla repliki
[mysqld]
datadir        = C:/xampp/mysql/data2
port           = 3308
server-id      = 2
read_only      = ON
relay_log      = relay-bin
```

1. Z1. Na źródle zatrzymaj MySQL w XAMPP Control Panel, w my.ini odkomentuj lub dopisz log-bin, sprawdź server-id = 1, dopisz binlog_format = ROW i uruchom serwer. Sprawdź @@server_id, @@log_bin, @@binlog_format.
2. Wariant A: na replicie w my.ini ustaw server-id = 2 i read_only = ON (bez log-bin). Na źródle sprawdź, czy w my.ini nie ma bind-address = 127.0.0.1 — jeśli jest, zakomentuj go.
3. Wariant B: przy zatrzymanym MySQL skopiuj folder C:\xampp\mysql\data do C:\xampp\mysql\data2 (zrób to przed włączeniem log-bin albo usuń z kopii pliki mysql-bin.*), utwórz my2.ini jak wyżej i uruchom replikę w osobnym oknie cmd:
C:\xampp\mysql\bin\mysqld.exe --defaults-file=C:\xampp\mysql\my2.ini --console. Łącz się z nią: mysql -u root -h 127.0.0.1 -P 3308.
4. Z2. Na replicie sprawdź @@server_id i @@read_only. Oba serwery muszą mieć różne server_id.

2 Konto replikacji i kopia startowa

1. Z3. Na źródle sprawdź, czy istnieją konta anonimowe (SELECT user, host FROM mysql.user WHERE user = '';) — jeśli tak, usuń je (przechwytyują logowanie i replika dostaje błąd 1045). Utwórz użytkownika replika @% z silnym hasłem i nadaj mu REPLICATION SLAVE ON *.* . Pokaż SHOW GRANTS.
2. Z4. Wykonaj na źródle kopię z zapisaną pozycją dziennika: mysqldump -u root --single-transaction --master-data=2 --databases sklep > sklep_repl.sql (MySQL 8.4: --source-data=2). Odszukaj w pliku komentarz -- CHANGE MASTER TO MASTER_LOG_FILE=..., MASTER_LOG_POS=... i porównaj z SHOW MASTER STATUS.
3. Z5. Zaimportuj sklep_repl.sql na replicie (wariant A: skopiuj plik przez udział sieciowy lub pendrive; wariant B: mysql -u root -h 127.0.0.1 -P 3308 < sklep_repl.sql).

3 Uruchomienie i test replikacji

```
-- replika, MariaDB (XAMPP):
CHANGE MASTER TO
  MASTER_HOST = '...',          -- wariant A: IP źródła, wariant B: '127.0.0.1'
  MASTER_PORT = 3306,
  MASTER_USER = 'replika', MASTER_PASSWORD = '...',
  MASTER_LOG_FILE = '...', MASTER_LOG_POS = ...;  -- wartości z pliku kopii
START SLAVE;

-- replika, MySQL 8.4:
CHANGE REPLICATION SOURCE TO SOURCE_HOST = '...', SOURCE_PORT = 3306,
  SOURCE_USER = 'replika', SOURCE_PASSWORD = '...', GET_SOURCE_PUBLIC_KEY = 1,
  SOURCE_LOG_FILE = '...', SOURCE_LOG_POS = ...;
START REPLICATION;
```

1. Z6. Podłącz replikę i uruchom replikację. Wyświetl `SHOW SLAVE STATUS\G` i zapisz: `Slave_IO_State`, `Master_Log_File`, `Slave_IO_Running`, `Slave_SQL_Running`, `Seconds_Behind_Master`, `Last_IO_Error`, `Last_SQL_Error`.
2. Z7. Na źródle dodaj klientkę Ola Replikowska (ola.r@example.pl, Lublin, dzisiejsza data). Na replice wyszukaj ją w tabeli `klienci`. Na źródle zmień cenę produktu 1 na 3199 zł i sprawdź ją na replice.
3. Z8. Na replice utwórz użytkownika `czytelnik@localhost` z uprawnieniami `SELECT` i `INSERT` na `sklep.*`, zaloguj się nim i spróbuj dodać kategorię. Zanotuj błąd.

4 Przerwanie, wznowienie i naprawa

1. Z9. Na replice zatrzymaj replikację (`STOP SLAVE;`). Na źródle zmień stan produktu 2 na 10. Sprawdź na replice stan produktu i pola `Slave_IO_Running`, `Read_Master_Log_Pos`, `Seconds_Behind_Master`.
2. Z10. Wznów replikację (`START SLAVE;`) i sprawdź, czy zmiana dotarła. Czy zmiany wykonane podczas przerwy zginęły?
3. Z11. Konflikt: na replice jako root dodaj kategorię `(7, 'Gry')`. Na źródle dodaj kategorię 'Smartfony' (dostanie id 7). Sprawdź stan repliki: `Slave_SQL_Running`, `Last_SQL_Errno`, `Last_SQL_Error`. Dodaj na źródle jeszcze kategorię 'Tablety' — czy dotarła na replikę?
4. Z12. Napraw replikę: usuń na niej kategorię 7 dodaną ręcznie i wykonaj `START SLAVE;`. Sprawdź, że kategorie 7 i 8 są takie jak na źródle i oba wątki działają. Zapisz wniosek: dlaczego na replice nie wolno pisać.
5. Z13. Na koniec zatrzymaj replikację (`STOP SLAVE; RESET SLAVE ALL;`), wyłącz replikę (wariant B: Ctrl+C w oknie mysqld), przywróć `my.ini` z kopii (lub zostaw log-bin, jeśli nauczyciel zaleci) i usuń reguły zapory.

W sprawozdaniu umieść

- ☐ fragmenty `my.ini` (i `my2.ini`) z ustawieniami replikacji
- ☐ tabela zapytań Z1–Z13 z wynikami
- ☐ zrzut ekranu `SHOW SLAVE STATUS` po uruchomieniu (Z6) i w czasie konfliktu (Z11)
- ☐ opis naprawy z Z12 i wniosek o `read_only`
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Za co odpowiadają wątki I/O i SQL repliki? Który z nich zatrzymał się w Z11 i dlaczego?
2. Po co w kopii startowej zapisuje się nazwę pliku i pozycję dziennika binarnego?
3. Dlaczego replikacja nie zastępuje kopii zapasowej? Co stanie się na replice po przypadkowym `DELETE FROM klienci;` na źródle?

4. Podaj dwa zastosowania replikacji w sklepie internetowym.

Kryteria oceny

Kryterium	Punkty
Konfiguracja serwerów i konto replikacji (Z1–Z3)	4
Kopia startowa i uruchomienie replikacji (Z4–Z6)	5
Testy zapisu i read_only (Z7–Z8)	3
Przerwanie, konflikt i naprawa (Z9–Z12), porządki Z13	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Partycjonowanie tabel

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Podzielenie dużej tabeli pomiarów na partycje metodami RANGE COLUMNS, LIST COLUMNS i HASH, sprawdzenie przycinania partycji (pruning) w EXPLAIN oraz porównanie usuwania starych danych przez DELETE i DROP PARTITION.

Po zajęciach potrafisz:

- tworzy tabelę partycjonowaną RANGE COLUMNS według daty oraz LIST COLUMNS i HASH
- sprawdza w EXPLAIN i information_schema.PARTITIONS, które partycje czyta zapytanie i ile mają wierszy
- porównuje czas usuwania starych danych przez DELETE i DROP PARTITION
- dodaje nowe partycje poleceniem REORGANIZE PARTITION
- wyjaśnia ograniczenia partycjonowania (klucz główny, klucze obce)

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- plik `r2_24_pomiary.sql` z folderu `Pliki_do_zadan`

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — konsola: XAMPP Control Panel → Shell, polecenie `mysql -u root` apacheFriends.org
- **Edytor tekstu (VS Code lub Notatnik++)** — plik `zadanie24.sql` z zapytaniami code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z lekcji 2.09 polecenie EXPLAIN i znaczenie kolumn type, key i rows.

Bezpieczeństwo

- Pracuj wyłącznie w bazie `pomiary`. Nie zmieniaj bazy `sklep` ani baz innych osób.
- DROP PARTITION i TRUNCATE PARTITION usuwają dane bez możliwości cofnięcia (nie działa ROLLBACK). Przed każdym takim poleceniem sprawdź nazwę tabeli i partycji.
- Import i kopiowanie 500 tys. wierszy obciążają dysk — nie przerywaj importu w połowie; jeśli się nie powiodł, zaimportuj plik ponownie (sam usuwa starą bazę `pomiary`).

Przebieg zadania

1 Import danych i ograniczenia partycjonowania

Baza pomiary ma tabelę czujniki (40 czujników w halach A–D) i tabelę odczyty (temperatura co ok. 47 s od 1 stycznia do 29 września 2026).

1. Zimportuj plik `r2_24_pomiary.sql` (phpMyAdmin → Import albo w konsoli `mysql -u root < r2_24_pomiary.sql` uruchomione w folderze z plikiem). Wszystkie polecenia zapisuj w pliku `zadanie24.sql`.
2. Z1. Napisz zapytanie zwracające liczbę wierszy oraz najwcześniejszą i najpóźniejszą datę w tabeli odczyty.
3. Z2. Odczytaj z `information_schema.TABLES` liczbę wierszy (`TABLE_ROWS`) i rozmiar w MB (`DATA_LENGTH + INDEX_LENGTH`) tabel bazy pomiary.
4. Z3. Spróbuj podzielić istniejącą tabelę odczyty poleceniem `ALTER TABLE odczyty PARTITION BY RANGE COLUMNS (czas) (...)` z dwiema partycjami: do 1 kwietnia 2026 i `MAXVALUE`. Zapisz numer i treść błędu.
5. Z4. Utwórz testową tabelę `t_test` z kolumnami `id` (klucz główny `AUTO_INCREMENT`) i `czas` `DATETIME`, partycjonowaną jak w Z3 (bez klucza obcego). Zapisz numer błędu i wyjaśnij go, a potem popraw klucz główny tak, by tabela powstała.

2 Partycje według zakresu dat (RANGE COLUMNS)

Utwórz kopię tabeli odczyty podzieloną na kwartały. Kolumna `czas` musi należeć do klucza głównego, a tabela partycjonowana nie może mieć kluczy obcych.

```
CREATE TABLE odczyty_p (  
  id          INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  id_czujnika SMALLINT UNSIGNED NOT NULL,  
  hala        VARCHAR(10) NOT NULL,  
  czas        DATETIME NOT NULL,  
  temperatura DECIMAL(5,2) NOT NULL,  
  PRIMARY KEY (id, czas)  
)  
PARTITION BY RANGE COLUMNS (czas) (  
  PARTITION p2026q1 VALUES LESS THAN ('2026-04-01'),  
  -- uzupełnij: p2026q2, p2026q3  
  PARTITION pmax  VALUES LESS THAN (MAXVALUE)  
);
```

1. Z5. Uzupełnij i wykonaj polecenie z ramki, a następnie skopiuj do `odczyty_p` wszystkie wiersze z odczyty (`INSERT ... SELECT`). Wykonaj `ANALYZE TABLE odczyty_p`.
2. Z6. Wyświetl z `information_schema.PARTITIONS` nazwę każdej partycji, jej granicę (`PARTITION_DESCRIPTION`) i szacowaną liczbę wierszy, w kolejności partycji.
3. Z7. Policz odczyty z sierpnia 2026 w obu tabelach. Sprawdź plan obu zapytań: w MariaDB `EXPLAIN PARTITIONS SELECT ...`, w MySQL 8.4 zwykłe `EXPLAIN SELECT ...` (kolumna `partitions` jest w nim zawsze). Zapisz kolumny `partitions` i `rows`.
4. Z8. Sprawdź w `EXPLAIN`, które partycje czyta zapytanie o odczyty z okresu 25 marca – 5 kwietnia 2026, a które zapytanie o średnią temperaturę czujnika nr 7 (bez warunku na dacie).
5. Z9. Policz wiersze tylko w partycji `p2026q1`, używając składni `FROM odczyty_p PARTITION (p2026q1)`.

3 Zarządzanie partycjami: usuwanie i dodawanie

Typowe zadanie administratora: usunąć dane starsze niż pół roku i przygotować miejsce na następny kwartał.

1. Z10. Usuń z tabeli odczyty (bez partycji) wszystkie odczyty z I kwartału poleceniem DELETE. Zanotuj czas wykonania podany przez konsolę lub phpMyAdmin.
2. Z11. Usuń te same dane z odczyty_p poleceniem `ALTER TABLE ... DROP PARTITION`. Zanotuj czas i porównaj liczbę wierszy w obu tabelach.
3. Z12. Wstaw do odczyty_p jeden odczyt z 15 listopada 2026 i sprawdź (jak w Z6), do której partycji trafił.
4. Z13. Podziel partycję pmax poleceniem `REORGANIZE PARTITION` na p2026q4 (do 1 stycznia 2027) i nową pmax. Sprawdź, w której partycji jest teraz listopadowy odczyt, a potem opróżnij p2026q4 poleceniem `TRUNCATE PARTITION`.

4 Partycje LIST i HASH

Te same dane można dzielić także według listy wartości albo równomiernie według funkcji skrótu.

1. Z14. Utwórz tabelę odczyty_hala o tych samych kolumnach, partycjonowaną `LIST COLUMNS (hala)`: p_produkcja — hala A i B, p_magazyn — hala C, p_biuro — hala D. Dobierz klucz główny. Skopiuj dane z odczyty_p i sprawdź liczbę wierszy w partycjach oraz EXPLAIN zapytania o maksymalną temperaturę w hali C.
2. Z15. Spróbuj wstawić do odczyty_hala odczyt z hali E. Zapisz numer błędu i zaproponuj, jak przyjąć dane z nowej hali.
3. Z16. Utwórz tabelę odczyty_hash partycjonowaną `HASH (id_czujnika) PARTITIONS 4`, skopiuj dane i porównaj liczbę wierszy w partycjach p0–p3 z wynikiem zapytania grupującego według `id_czujnika % 4`. Sprawdź w EXPLAIN, którą partycję czyta zapytanie o czujnik nr 7.

W sprawozdaniu umieść

- ☐ plik `zadanie24.sql` ze wszystkimi poleceniami Z1–Z16 opisanymi komentarzami `-- Z1` itd.
- ☐ zrzuty ekranu wyników Z6, Z7 (oba plany) i Z14
- ☐ tabela czasów DELETE i DROP PARTITION
- ☐ numery i treść błędów z Z3, Z4 i Z15 z wyjaśnieniem
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego kolumna czas musi być częścią klucza głównego tabeli odczyty_p?
2. Na czym polega przycinanie partycji (partition pruning) i kiedy nie działa? Odwołaj się do wyników Z7 i Z8.
3. Dlaczego DROP PARTITION jest znacznie szybsze od DELETE i czego nie da się po nim zrobić?
4. Do jakiej tabeli w bazie sklep partycjonowanie by się nie nadawało i dlaczego? Kiedy lepiej wystarczy zwykły indeks?

Kryteria oceny

Kryterium	Punkty
Z1–Z4: import, rozmiar tabeli, poprawnie opisane błędy ograniczeń	3
Z5–Z9: tabela RANGE COLUMNS, odczyt partycji, poprawnie odczytany pruning w EXPLAIN	6
Z10–Z13: pomiar DELETE vs DROP PARTITION, REORGANIZE i TRUNCATE PARTITION	4
Z14–Z16: tabele LIST COLUMNS i HASH z analizą rozkładu wierszy	4
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Dane JSON w MySQL i MariaDB

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zapisywanie, odczytywanie, wyszukiwanie i modyfikowanie danych o zmiennej strukturze w kolumnie JSON oraz przyspieszanie wyszukiwania po polu JSON indeksem na kolumnie generowanej.

Po zajęciach potrafisz:

- odczytuje pola i elementy tablic JSON funkcjami `JSON_VALUE`, `JSON_EXTRACT`, `JSON_LENGTH` i `JSON_KEYS`
- wyszukuje wiersze po zawartości dokumentu JSON (`JSON_CONTAINS`, `JSON_CONTAINS_PATH`)
- modyfikuje dokumenty funkcjami `JSON_SET`, `JSON_ARRAY_APPEND` i `JSON_REMOVE`
- buduje odpowiedź JSON z danych tabeli (`JSON_OBJECT`, `JSON_ARRAYAGG` lub `GROUP_CONCAT`)
- tworzy kolumnę generowaną z indeksem i sprawdza jej użycie w `EXPLAIN`

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- plik `r2_25_katalog.sql` z folderu Pliki_do_zadan

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — phpMyAdmin → zakładka SQL albo konsola `mysql -u root`
apachefriends.org
- **VS Code** — plik `zadanie25.sql` z zapytaniami
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie zapis JSON z lekcji programowania: obiekt `{ }`, tablica `[ ]`, cudzysłowy wokół kluczy i tekstów.

Bezpieczeństwo

- Pracuj w bazie `katalog`. Przed poleceniami `UPDATE` sprawdź warunek `WHERE` zapytaniem `SELECT` — `UPDATE` bez `WHERE` zmieni parametry wszystkich produktów.
- Dane klientów w tabeli `zamowienia_api` są fikcyjne. W prawdziwej aplikacji nie zapisuj w dokumentach JSON haseł ani numerów kart.

Przebieg zadania

1 Odczyt pól JSON

Zaimportuj plik `r2_25_katalog.sql`. Tabela produkty ma kolumnę parametry — każda kategoria ma w niej inne pola. Ścieżka `$.poLe` wskazuje pole obiektu, `$.porty[0]` pierwszy element tablicy, `$.klient.miasto` pole zagnieżdżone.

1. Obejrzyj strukturę tabeli (`SHOW CREATE TABLE produkty`). Zapisz, jakiego typu jest kolumna parametry na twoim serwerze i czy ma ograniczenie CHECK.
2. Z1. Wyświetl nazwę, ilość RAM i kolor wszystkich laptopów (`JSON_VALUE`).
3. Z2. Dla produktów Laptop Vertex 14 i Mysz bezprzewodowa wyświetl kolor trzema sposobami: `JSON_EXTRACT`, `JSON_UNQUOTE(JSON_EXTRACT(...))` i `JSON_VALUE`, a także całą tablicę porty przez `JSON_EXTRACT` i `JSON_VALUE`. Opisz różnice.
4. Z3. Wyświetl nazwę, listę kluczy dokumentu (`JSON_KEYS`) i liczbę portów (`JSON_LENGTH`) każdego produktu, od największej liczby portów.
5. Z4. Znajdź produkty, które mają port HDMI (`JSON_CONTAINS` na ścieżce `$.porty`).
6. Z5. Wyświetl laptopy z co najmniej 32 GB RAM, posortowane malejąco według RAM (zamień wartość na liczbę przez `CAST`).
7. Z6. Znajdź produkty, które nie mają pola kolor albo nie mają parametrów wcale (`JSON_CONTAINS_PATH`, `IS NULL`).

2 Zmiana dokumentów i kontrola poprawności

Funkcje `JSON_SET`, `JSON_ARRAY_APPEND` i `JSON_REMOVE` zwracają zmieniony dokument — wynik trzeba zapisać poleceniem `UPDATE`.

1. Z7. Spróbuj dodać produkt z niepoprawnym JSON-em w kolumnie parametry, np. z kluczem bez cudzysłowu. Zapisz numer i treść błędu.
2. Z8. Laptopowi Nimbus Air zmień kolor na „niebieski” i dodaj pole gwarancja_mies o wartości 24 — jednym poleceniem `UPDATE` z `JSON_SET`.
3. Z9. Laptopowi Nimbus Studio dopisz do tablicy porty wartość „Thunderbolt”, a klawiaturze mechanicznej usuń pole ukłlad. Wyświetl oba dokumenty po zmianie.
4. Z10. Pakietowi biurowemu (parametry = `NULL`) zapisz dokument zbudowany funkcjami `JSON_OBJECT` i `JSON_ARRAY`: wersja „2026”, stanowiska 5, platformy Windows i macOS.

3 JSON jako odpowiedź dla aplikacji i indeks

Aplikacja w JavaScriptcie oczekuje danych w formacie JSON. Funkcja `JSON_ARRAYAGG` jest w MySQL 8 i MariaDB od 10.5 — w XAMPP (MariaDB 10.4) użyj zamiennika z `GROUP_CONCAT`.

```
-- MariaDB 10.4 (XAMPP): tablica JSON sklejona z obiektów
SELECT CONCAT('[', GROUP_CONCAT(JSON_OBJECT(...) ORDER BY ... SEPARATOR ','), ']')
-- MySQL 8.4 / MariaDB 10.5+:
SELECT JSON_ARRAYAGG(JSON_OBJECT(...))
```

1. Z11. Dla każdego monitora zbuduj obiekt JSON z polami id, nazwa i cena (`JSON_OBJECT`).
2. Z12. Dla każdej kategorii zwróć jedną tablicę JSON obiektów {nazwa, cena} posortowanych według ceny. Sprawdź funkcją `JSON_VALID`, że wynik jest poprawnym JSON-em.
3. Z13. Dodaj do tabeli produkty kolumnę generowaną `ram_gb` typu `INT`, wyliczaną z `$.ram_gb` (`VIRTUAL`), i indeks na niej. Porównaj `EXPLAIN` zapytania o laptopy z ≥ 32 GB RAM: z warunkiem na `ram_gb` i z warunkiem na `JSON_VALUE(...)`.

4 Dokumenty zamówień z API

Tabela zamówienia_api przechowuje całe zamówienia tak, jak przyszły z zewnętrznego systemu (klient i pozycje zagnieżdżone w dokumencie).

1. Z14. Wyświetl numer zamówienia, nazwisko i miasto klienta, liczbę pozycji i nazwę pierwszego produktu dla zamówień opłaconych BLIK-iem.
2. Z15. Znajdź zamówienia z kodem rabatowym (wyświetl kod) oraz zamówienia zawierające pozycję z produktem „Mysz bezprzewodowa” (JSON_CONTAINS z obiektem na ścieżce `$.pozycje`).
3. Z16. Policz zamówienia według miasta klienta. W sprawozdaniu oceń: które dane z zamówienia_api warto przenieść do zwykłych kolumn i tabel, a które mogą zostać w JSON-ie?

W sprawozdaniu umieść

- ☐ plik `zadanie25.sql` z zapytaniami Z1–Z16 opisanymi komentarzami
- ☐ typ kolumny parametry na twoim serwerze (wynik `SHOW CREATE TABLE`)
- ☐ zrzuty ekranu wyników Z2, Z7, Z12 i obu planów `EXPLAIN` z Z13
- ☐ ocena projektu tabeli zamówienia_api (Z16)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się wynik `JSON_EXTRACT` od `JSON_VALUE` dla pola tekstowego i dla tablicy?
2. Jak MariaDB pilnuje poprawności danych w kolumnie JSON, a jak MySQL? Odwołaj się do Z7 i `SHOW CREATE TABLE`.
3. Dlaczego warunek `WHERE JSON_VALUE(parametry, '$.ram_gb') >= 32` nie korzysta z indeksu, a warunek na kolumnie generowanej tak?
4. Podaj dwa przykłady danych, które dobrze jest trzymać w JSON-ie, i dwa, które powinny być zwykłymi kolumnami. Uzasadnij.

Kryteria oceny

Kryterium	Punkty
Z1–Z6: poprawny odczyt i wyszukiwanie po polach JSON	6
Z7–Z10: kontrola poprawności i modyfikacja dokumentów	4
Z11–Z13: budowanie JSON, kolumna generowana i analiza <code>EXPLAIN</code>	4
Z14–Z16: zapytania do dokumentów zagnieżdżonych i ocena projektu	3
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik `.docx` o tym samym numerze).

Dane przestrzenne (GIS) — punkty odbioru i strefy dostaw

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zapisywanie położenia w kolumnach POINT i POLYGON, obliczanie odległości na kuli ziemskiej, wyszukiwanie najbliższego punktu odbioru i sprawdzanie, czy adres leży w strefie dostawy, oraz użycie indeksu SPATIAL.

Po zajęciach potrafisz:

- zapisuje i odczytuje punkty i wielokąty (ST_GeomFromText, POINT, ST_AsText, ST_X, ST_Y)
- oblicza odległości funkcją ST_Distance_Sphere i znajduje najbliższy punkt dla każdego klienta
- sprawdza przynależność punktu do obszaru funkcją ST_Contains
- oblicza długość trasy z kolejnych przystanków (LAG + ST_Distance_Sphere)
- sprawdza w EXPLAIN użycie indeksu SPATIAL i eksportuje dane do GeoJSON

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4, dostęp do internetu (podgląd mapy)
- plik `r2_26_dostawy.sql` z folderu Pliki_do_zadan

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — phpMyAdmin pokazuje kolumny geometryczne jako dane binarne — używaj ST_AsText
apacheFriends.org
- **geojson.io** — podgląd GeoJSON na mapie w przeglądarce, bez rejestracji
geojson.io
- **Mapa internetowa (OpenStreetMap)** — odczyt współrzędnych: prawy przycisk myszy → Pokaż adres
openstreetmap.org

Przygotowanie przed zajęciami

- Przypomnij sobie z geografii: długość geograficzna (wschód–zachód) i szerokość geograficzna (północ–południe). Kraków leży ok. 20° E i 50° N.

Bezpieczeństwo

- Pracuj w bazie `dostawy`. Położenie klientów to dane osobowe — w zadaniu są fikcyjne; w prawdziwej aplikacji nie publikuj ich na mapie ani w GeoJSON bez potrzeby.
- Do geojson.io wklejaj tylko dane z zadania (punkty odbioru, strefy), nie dane klientów.

Przebieg zadania

1 Punkty i odległości

Zaimportuj plik `r2_26_dostawy.sql`. Współrzędne zapisano jako POINT(długość szerokość) bez układu odniesienia (SRID 0). Funkcja ST_Distance_Sphere(p1, p2) zwraca odległość w metrach po powierzchni Ziemi.

```
-- dwa sposoby zapisu tego samego punktu
ST_GeomFromText('POINT(19.9373 50.0617)')
POINT(19.9373, 50.0617)
```

1. Z1. Wyświetl nazwę, zapis tekstowy położenia (ST_AsText), długość (ST_X) i szerokość (ST_Y) punktów odbioru w Krakowie.
2. Z2. Odczytaj na mapie OpenStreetMap współrzędne dowolnego miejsca w Krakowie (np. szkoły lub dworca) i dodaj punkt odbioru o nazwie „Punkt Dworzec” lub własnej. Uwaga: OpenStreetMap podaje najpierw szerokość, potem długość.
3. Z3. Wyświetl odległość w km (2 miejsca po przecinku) każdego punktu w Krakowie od punktu Stare Miasto (19.9373, 50.0617), od najbliższego.
4. Z4. Znajdź jeden najbliższy punkt odbioru dla klienta Zielińskiego (odległość w metrach).
5. Z5. Dla każdego klienta znajdź najbliższy punkt odbioru (ROW_NUMBER() OVER (PARTITION BY ... ORDER BY odległość)).
6. Z6. Wyświetl punkty odbioru w promieniu 2 km od klientki Nowak.
7. Z7. Oblicz odległości w linii prostej między punktami w Krakowie (Stare Miasto), Warszawie, Gdańsku i Wrocławiu — każdą parę miast raz (złączenie tabeli z samą sobą). Porównaj jedną z odległości z trasą samochodową na mapie.

2 Strefy dostawy (wielokąty)

Tabela strefy zawiera dwa prostokątne obszary (POLYGON). Pierwszy i ostatni punkt wielokąta muszą być takie same.

1. Z8. Dla każdego klienta wyświetl strefę dostawy i opłatę, a dla klientów spoza stref napis „poza strefami” (LEFT JOIN ... ON ST_Contains(obszar, polozenie)).
2. Z9. Policz punkty odbioru w każdej strefie.
3. Z10. Wyeksportuj punkty odbioru w Krakowie i strefę „Centrum” funkcją ST_AsGeoJSON. Wklej wynik dla strefy do geojson.io (zamień zawartość okna na obiekt Polygon) i zrób zrzut mapy. Dodaj wielokąt własnej strefy dostawy (np. wokół szkoły) poleceniem INSERT i sprawdź go tak samo.

3 Trasa kuriera i indeks SPATIAL

Tabela kurs_przystanki zawiera kolejne przystanki kursu nr 1, a tabela automaty — 5000 fikcyjnych automatów paczkowych z indeksem SPATIAL.

1. Z11. Wyświetl przystanki kursu 1 z długością odcinka od poprzedniego przystanku w km (funkcja LAG), a następnie oblicz długość całej trasy.
2. Z12. Policz automaty leżące w prostokącie 19,5–20,5° E i 49,8–50,3° N (MBRContains z wielokątem). Porównaj EXPLAIN tego zapytania bez wskazówek i z `IGNORE INDEX (sp_automat)`.
3. Z13. Policz automaty w promieniu 20 km od Starego Miasta warunkiem na ST_Distance_Sphere i sprawdź EXPLAIN. Następnie napisz zapytanie, które najpierw zawęży wyszukiwanie prostokątem (MBRContains — korzysta z indeksu), a potem sortuje po odległości i zwraca 3 najbliższe automaty.

W sprawozdaniu umieść

- ☐ plik `zadanie26.sql` z zapytaniami Z1–Z13 opisanymi komentarzami

- ☐ zrzuty ekranu wyników Z5, Z8 i Z11
- ☐ zrzut mapy z geojson.io ze strefą „Centrum” i własną strefą (Z10)
- ☐ porównanie planów EXPLAIN z Z12 i Z13
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego odległość z ST_Distance_Sphere jest mniejsza niż trasa samochodowa i do czego mimo to się przydaje?
2. Czym różni się MBRContains od ST_Contains? Kiedy wynik może być inny?
3. Dlaczego warunek ST_Distance_Sphere(...) < 20000 nie korzysta z indeksu SPATIAL i jak to obejść?
4. Podaj trzy pytania, na które odpowiada aplikacja z danymi przestrzennymi w sklepie internetowym lub firmie kurierskiej.

Kryteria oceny

Kryterium	Punkty
Z1–Z4: zapis punktów i obliczanie odległości	4
Z5–Z7: najbliższy punkt dla każdego klienta, promień, odległości między miastami	4
Z8–Z10: strefy (ST_Contains), GeoJSON i własny wielokąt na mapie	4
Z11–Z13: długość trasy i analiza indeksu SPATIAL	5
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Hierarchie i modele danych w bazie relacyjnej

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Porównanie w praktyce sposobów zapisu hierarchii (lista sąsiedztwa, ścieżka materializowana, dokument JSON) i grafu (tabela krawędzi) w bazie relacyjnej oraz dobór modelu danych do zastosowania.

Po zajęciach potrafisz:

- odczytuje poddrzewo, przodków i liście hierarchii zapisanej jako lista sąsiedztwa (WITH RECURSIVE)
- wyszukuje w ścieżce materializowanej i porównuje koszt przeniesienia gałęzi w obu modelach
- odnajduje węzły w drzewie zapisanym jako dokument JSON
- wykonuje zapytania grafowe: znajomi, propozycje znajomych, stopień oddalenia i najkrótsza ścieżka
- uzasadnia wybór modelu danych dla konkretnego zastosowania

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4
- plik `r2_27_modele.sql` z folderu `Pliki_do_zadan`

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — zapytania WITH RECURSIVE wygodniej pisać w konsoli lub w zakładce SQL [apachefriends.org](https://www.apachefriends.org)
- **draw.io (diagrams.net)** — szkic drzewa kategorii i grafu znajomości app.diagrams.net

Przygotowanie przed zajęciami

- Przypomnij sobie z lekcji 2.03 budowę zapytania WITH RECURSIVE: część początkowa, UNION ALL, część rekurencyjna z warunkiem zakończenia.

Bezpieczeństwo

- Pracuj w bazie `modele`. Polecenia zmieniające drzewo (Z8) wykonuj w transakcji i zakończ ROLLBACK, żeby dalsze zapytania dawały wyniki zgodne z treścią.
- Zapytanie rekurencyjne bez warunku zakończenia może działać bardzo długo przy cyklu w danych — zawsze ograniczaj głębokość i sprawdzaj, czy węzeł nie był już odwiedzony.

Przebieg zadania

1 Lista sąsiedztwa i WITH RECURSIVE

Zaimportuj plik `r2_27_modele.sql`. Narysuj w draw.io drzewo z tabeli `kategorie` (16 węzłów, korzeń „Sklep”). Każdy wiersz zna tylko swojego rodzica (`id_rodzica`).

```
-- W MySQL kolumna tekstowa CTE ma długość z części początkowej --
-- dlatego ścieżkę zaczynaj od CAST(... AS CHAR(255)).
WITH RECURSIVE poddrzewo AS (
  SELECT id_kategorii, nazwa, 0 AS poziom, CAST(nazwa AS CHAR(255)) AS sciezka
  FROM kategorie WHERE nazwa = 'Peryferia'
  UNION ALL
  SELECT ...    -- uzupełnij
)
SELECT ...
```

1. Z1. Wyświetl bezpośrednie podkategorie kategorii „Peryferia” (złączenie tabeli z samą sobą).
2. Z2. Uzupełnij zapytanie z ramki: wyświetl całe poddrzewo „Peryferia” z poziomem i ścieżką „Peryferia > Myszy > ...”. Nazwy wcięte według poziomu (REPEAT), sortowanie według ścieżki.
3. Z3. Zbuduj „okruszki” (breadcrumb) dla kategorii „Myszy pionowe”: jedna wartość w postaci „Sklep > Peryferia > ...” (rekurencja w górę drzewa i GROUP_CONCAT).
4. Z4. Wyświetl liście drzewa — kategorie bez podkategorii (NOT EXISTS).
5. Z5. Dla każdej kategorii policz wszystkich jej potomków na wszystkich poziomach; pomір kategorii bez potomków. Wskazówka: rekurencyjnie zbuduj pary (przodek, potomek).

2 Ścieżka materializowana i dokument JSON

Tabela `kategorie_sciezka` przechowuje to samo drzewo jako tekst ścieżki, np. „/1/3/10/” (Sklep → Peryferia → Myszy). Tabela `katalog_json` zawiera gałąź Peryferia jako jeden zagnieżdżony dokument.

1. Z6. Wyświetl poddrzewo „Peryferia” z tabeli `kategorie_sciezka` jednym zapytaniem bez rekurencji (LIKE). Oblicz poziom węzła z liczby znaków „/” w ścieżce.
2. Z7. Wyświetl przodków kategorii „Myszy pionowe” z tabeli `kategorie_sciezka` (złączenie tabeli z samą sobą: ścieżka potomka zaczyna się od ścieżki przodka).
3. Z8. W transakcji przenieś kategorię „Myszy biurowe” (z jej podkategorią) bezpośrednio pod „Peryferia” w obu modelach. Zapisz, ile wierszy zmieniło każde polecenie UPDATE (ROW_COUNT()), sprawdź wynik i wykonaj ROLLBACK.
4. Z9. Z dokumentu w `katalog_json` odczytaj nazwę drugiej podkategorii, liczbę podkategorii pierwszego poziomu oraz nazwę najgłębiej położonego węzła. Funkcją JSON_SEARCH znajdź ścieżkę do „Myszy pionowe” i wszystkie ścieżki nazw zaczynających się od „Myszy”.

3 Graf znajomości w tabeli krawędzi

Tabela znajomosci zapisuje każdą znajomość raz ($id_a < id_b$). Narysuj graf w draw.io: 10 osób, 10 krawędzi.

1. Z10. Utwórz widok znajomi (osoba, znajomy, od_kiedy) zawierający każdą znajomość w obu kierunkach (UNION ALL).
2. Z11. Wyświetl znajomych Celiny oraz liczbę znajomych każdej osoby, od największej.
3. Z12. Zaproponuj Ali nowych znajomych: znajomi jej znajomych, którzy nie są jej znajomymi ani nią samą, z liczbą wspólnych znajomych. Osobno wyświetl wspólnych znajomych Ali i Darka.
4. Z13. Zapytaniem WITH RECURSIVE oblicz stopień oddalenia każdej osoby od Ali (Ala = 0, jej znajomi = 1 itd.). Zapobiegaj cyklom: przechowuj listę odwiedzonych id i sprawdzaj ją funkcją FIND_IN_SET; ogranicz głębokość do 6.
5. Z14. Wyświetl najkrótszą ścieżkę znajomości od Ali do Henryka w postaci „Ala → ... → Henryk”. Wyjaśnij, dlaczego w wyniku Z13 nie ma Igi i Jana.

4 Wybór modelu danych

Na podstawie wyników i lekcji 2.17 dobierz model do zastosowania.

1. Uzupełnij w sprawozdaniu tabelę „Wybór modelu”: dla każdego zastosowania wskaż model (relacyjny, hierarchiczny — lista sąsiedztwa lub ścieżka, dokumentowy, grafowy, klucz–wartość) i jedno zdanie uzasadnienia.

W sprawozdaniu umieść

- ☐ plik `zadanie27.sql` z zapytaniami Z1–Z14 opisanymi komentarzami
- ☐ rysunki drzewa kategorii i grafu znajomości z draw.io
- ☐ zrzuty ekranu wyników Z2, Z8 (przed ROLLBACK), Z13 i Z14
- ☐ tabela porównania modeli hierarchii i tabela „Wybór modelu”
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Który zapis hierarchii (lista sąsiedztwa czy ścieżka materializowana) jest lepszy, gdy drzewo często się zmienia, a który — gdy częściej się je odczytuje? Odwołaj się do Z8.
2. Dlaczego znajomość zapisano raz z warunkiem $id_a < id_b$ i po co powstał widok znajomi?
3. Jak zapytanie z Z13 unika nieskończonej pętli w grafie z cyklem (Ala–Bartek–Celina)?
4. Kiedy warto użyć osobnej bazy grafowej (np. Neo4j) zamiast tabeli krawędzi w MySQL?

Kryteria oceny

Kryterium	Punkty
Z1–Z5: zapytania rekurencyjne na liście sąsiedztwa	5
Z6–Z9: ścieżka materializowana, przeniesienie gałęzi, dokument JSON	4
Z10–Z14: widok i zapytania grafowe	5
Rysunki, tabele porównania i wyboru modelu	3
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

MongoDB — dokumenty i operacje CRUD

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Zapisywanie danych jako dokumentów w kolekcjach MongoDB oraz wykonywanie operacji CRUD: wstawianie, wyszukiwanie z operatorami i projekcją, sortowanie, aktualizacja i usuwanie, także w polach zagnieżdżonych i tablicach.

Po zajęciach potrafisz:

- importuje dokumenty do kolekcji i wstawia je poleceniem insertMany
- wyszukuje dokumenty z operatorami porównania (\$lt, \$gte, \$in), logicznymi (\$or) i tablicowymi (\$all, \$size, \$exists)
- stosuje projekcję, sortowanie, limit i skip
- aktualizuje dokumenty operatorami \$set, \$inc, \$mul, \$push i \$unset
- porównuje polecenia MongoDB z odpowiednikami w SQL

Sprzęt i materiały

- komputer z Windows 11 i uprawnieniami do instalacji programów (albo dostęp do internetu — wariant Atlas)
- plik `r2_28_produkty.json` z folderu Pliki_do_zadan

Oprogramowanie

- **MongoDB Community Server** — instalator MSI dla Windows; usługa nasłuchuje domyślnie tylko na localhost:27017 mongodb.com/try/download/community
- **MongoDB Compass** — graficzny klient z wbudowaną konsolą mongosh (pasek na dole okna) mongodb.com/try/download/compass
- **MongoDB Shell (mongosh)** — opcjonalnie — samodzielna konsola mongodb.com/try/download/shell
- **MongoDB Atlas (wariant bez instalacji)** — darmowy klaster w chmurze; wymaga konta — tylko za zgodą nauczyciela mongodb.com/cloud/atlas/register

Przygotowanie przed zajęciami

- Sprawdź z nauczycielem, czy w pracowni jest zainstalowany MongoDB Community Server i Compass. Jeśli nie — zainstaluj je z oficjalnych stron (instalator serwera może od razu zainstalować Compass).
- Przypomnij sobie z lekcji 2.16 zapis JSON i różnicę między obiektem a tablicą.

Bezpieczeństwo

- Nie zmieniaj ustawienia `bindIp` w pliku konfiguracyjnym serwera i nie otwieraj portu 27017 w zaporze — serwer bez użytkowników i hasła nie może być dostępny z sieci.
- Przed każdym `updateMany` i `deleteMany` wykonaj `find` (albo `countDocuments`) z tym samym filtrem i sprawdź, ile dokumentów zostanie zmienionych. Pusty filtr `{}` oznacza WSZYSTKIE dokumenty.
- W wariancie Atlas nie zapisuj hasła do klastra w sprawozdaniu ani na zrzutach ekranu, a w Network Access dodaj tylko swój adres IP.

Przebieg zadania

1 Połączenie, import i insertMany

Uruchom Compass i połącz się z `mongodb://localhost:27017`. Wszystkie polecenia wpisuj w konsoli mongosh (w Compass: pasek „MONGOSH” na dole okna) i zapisuj w pliku `zadanie28.js`.

```
use sklep
db.klienci.insertMany([
  { imie: "Anna", nazwisko: "Nowak", email: "anna.nowak@example.pl",
    adres: { miasto: "Kraków", kod: "31-001" }, zainteresowania: ["laptopy", "gaming" ] },
  { imie: "Piotr", nazwisko: "Kowalski", email: "p.kowalski@example.pl",
    adres: { miasto: "Warszawa", kod: "00-001" }, zainteresowania: ["monitory" ] },
  // dopisz: Zofia Wiśniewska (Kraków, 30-002, bez zainteresowań – pusta tablica),
  //          Jakub Wójcik (Gdańsk, 80-001, gaming i podzespoły),
  //          Julia Zielińska (bez pola adres, zainteresowania: drukarki)
])
```

1. Z1. W Compass utwórz bazę `sklep` z kolekcją `produkty` (Create database) i zaimportuj do niej plik `r2_28_produkty.json` (kolekcja produkty → ADD DATA → Import JSON or CSV file). Policz dokumenty poleceniem `db.produkty.countDocuments({})`.
2. Z2. Uzupełnij i wykonaj polecenie z ramki. Zapisz, co zwróciło (acknowledged, insertedIds) i skąd wzięły się wartości `_id`. Sprawdź, że w kolekcji klienci jest 5 dokumentów.

2 Wyszukiwanie dokumentów

`find(filtr, projekcja)`: filtr odpowiada WHERE, projekcja — liście kolumn w SELECT (1 — pokaż, 0 — ukryj). Pola zagnieżdżone zapisuj w cudzysłowie z kropką, np. `"parametry.ram_gb"`.

1. Z3. Wyświetl nazwę i cenę wszystkich laptopów bez pola `_id`.
2. Z4. Wyświetl produkty tańsze niż 500 zł posortowane rosnąco według ceny.
3. Z5. Wyświetl nazwę i stan produktów z kategorii Monitory lub Akcesoria (`$in`), które są dostępne (stan większy od 0).
4. Z6. Znajdź produkty, które w parametrach mają co najmniej 32 GB RAM (pole zagnieżdżone).
5. Z7. Znajdź produkty z tagiem „gaming”, produkty mające jednocześnie tagi „gaming” i „grafika” (`$all`) oraz produkty z pustą tablicą tagów (`$size`).
6. Z8. Policz i wyświetl produkty, których nie ma w magazynie albo które kosztują ponad 5000 zł (`$or`).
7. Z9. Wyświetl produkty, które nie mają pola parametry (`$exists`).
8. Z10. Wyświetl 3 najdroższe produkty, a następnie „drugą stronę” wyników — produkty z miejsc 4–6 (sort, skip, limit).

3 Aktualizacja i usuwanie

updateOne i updateMany przyjmują filtr i opis zmian z operatorami. Wynik zawiera matchedCount (pasujące) i modifiedCount (zmienione).

1. Z11. Myszy bezprzewodowej (_id: 7) ustaw pole promocja na true i zmniejsz stan o 5 jednym poleceniem (\$set, \$inc). Wyświetl dokument po zmianie.
2. Z12. Wszystkim akcesoriom obniż cenę o 10% (\$mul) i dopisz tag „wyprzedaż” (\$push). Sprawdź ceny po zmianie — co zauważasz? Wykonaj też updateOne z filtrem { _id: 99 } i porównaj matchedCount.
3. Z13. W kolekcji klienci wyświetl nazwiska klientów z Krakowa, dopisz Zofii Wiśniewskiej zainteresowanie „monitory” i znajdź klientów bez pola adres. Usuń wszystkim klientom pole email (\$unset) — najpierw sprawdź, ilu dokumentów dotyczy filtr.
4. Z14. Policz produkty o stanie 0, a następnie usuń je jednym poleceniem (deleteMany). Usuń klientkę Zielińską (deleteOne). Podaj liczbę dokumentów w obu kolekcjach po usunięciu.
5. Z15. Zapisz w sprawozdaniu odpowiedniki w SQL (baza sklep) dla Z4, Z8 i Z11 oraz zaznacz, które z poleceń MongoDB nie mają prostego odpowiednika w SQL (np. Z7).

W sprawozdaniu umieść

- ☐ plik zadanie28.js z poleceniami Z1–Z15 opisanymi komentarzami // Z1 itd.
- ☐ zrzuty ekranu: zaimportowana kolekcja produkty w Compass, wyniki Z6, Z7, Z12 i Z14
- ☐ tabela porównania poleceń MongoDB i SQL (Z15)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego w jednej kolekcji produkty mogą być dokumenty z polem parametry i bez niego? Jakie to ma zalety i wady?
2. Czym różnią się matchedCount i modifiedCount? Kiedy oba są równe 0?
3. Skąd wzięła się wartość 116.10000000000001 w Z12 i jak przechowywać ceny, żeby tego uniknąć?
4. Dlaczego serwera MongoDB bez hasła nie wolno udostępniać w sieci i jakie domyślne ustawienie chroni przed tym nową instalację?

Kryteria oceny

Kryterium	Punkty
Z1–Z2: import i wstawienie dokumentów (w tym zagnieżdżonych)	3
Z3–Z10: wyszukiwanie z operatorami, projekcją, sortowaniem i stronicowaniem	7
Z11–Z14: aktualizacja i usuwanie z kontrolą filtrów	5
Z15: porównanie z SQL	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

MongoDB — agregacje i porównanie z SQL

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Budowanie potoków agregacji w MongoDB (\$match, \$group, \$sort, \$project, \$unwind, \$lookup), porównanie ich z zapytaniami SQL na bazie sklep, użycie indeksu w MongoDB i ocena, kiedy wybrać model dokumentowy, a kiedy relacyjny.

Po zajęciach potrafisz:

- buduje potok agregacji z etapami \$match, \$group, \$sort, \$limit i \$project
- rozpakowuje tablice etapem \$unwind i oblicza wartości wyrażeniami (\$multiply, \$sum, \$size)
- łączy kolekcje etapem \$lookup
- zakłada indeks i porównuje plan zapytania (COLLSCAN / IXSCAN) w explain
- porównuje te same raporty w SQL i MongoDB i dobiera model danych do zastosowania

Sprzęt i materiały

- komputer z MongoDB Community Server i Compass (jak w zadaniu 28) oraz XAMPP z bazą sklep
- pliki `r2_28_produkty.json` i `r2_29_zamowienia.json` z folderu Pliki_do_zadan

Oprogramowanie

- **MongoDB Compass** — zakładka Aggregations (kreator potoku) i konsola mongosh
mongodb.com/try/download/compass
- **MongoDB Community Server** — lokalnie, tylko localhost
mongodb.com/try/download/community
- **XAMPP (MariaDB, phpMyAdmin)** — zapytania SQL na bazie sklep
[apachefriends.org](https://www.apachefriends.org)

Przygotowanie przed zajęciami

- Sprawdź, że baza sklep jest zaimportowana w XAMPP (plik sklep.sql z lekcji).
- Przypomnij sobie z lekcji 1.13–1.14 GROUP BY, HAVING i złączenia.

Bezpieczeństwo

- Serwer MongoDB zostaw z domyślnym `bindIp: 127.0.0.1`. Nie włączaj dostępu z sieci, nawet po utworzeniu użytkownika.
- Nie włączaj w pracowni `security.authorization` w pliku `mongod.cfg` bez zgody nauczyciela — po restarcie usługi bez znanego hasła stracisz dostęp do bazy. Hasło użytkownika z Z12 wymyśl, nie wpisuj go na zrzutach ekranu.
- Przed zaimportowaniem kolekcji usuń jej starą wersję (z zadania 28) tylko w swojej bazie `sklep`.

Przebieg zadania

1 Przygotowanie danych

W MongoDB jedno zamówienie to jeden dokument: dane klienta i tablica pozycji są w środku. W SQL te same dane są w czterech tabelach.

```
// fragment dokumentu z kolekcji zamowienia
{ _id: 1,
  klient: { id: 1, imie: "Anna", nazwisko: "Nowak", miasto: "Kraków" },
  data: ISODate("2026-02-10T09:15:00Z"), status: "wyslane",
  pozycje: [ { id_produktu: 1, produkt: "Laptop Vertex 14", ilosc: 1, cena: 3299 },
             { id_produktu: 7, produkt: "Mysz bezprzewodowa", ilosc: 1, cena: 129 } ] }
```

1. Z1. W bazie sklep (MongoDB) usuń kolekcję produkty z zadania 28 (`db.produkty.drop()`) i zaimportuj ją od nowa z `r2_28_produkty.json`. Utwórz kolekcję zamowienia i zaimportuj `r2_29_zamowienia.json`. Policz dokumenty w obu kolekcjach. Sprawdź w Compass, że pole data ma typ Date.

2 Potoki agregacji i ich odpowiedniki w SQL

`db.kolekcja.aggregate([ etap1, etap2, ... ])` — każdy etap przetwarza wynik poprzedniego. Do każdego potoku napisz zapytanie SQL na bazie sklep i porównaj wyniki. Wartość pozycji: `{ $multiply: ["$pozycje.ilosc", "$pozycje.cena"] }`.

1. Z2. Policz zamówienia o statusie „wyslane” (`$match`, `$count`).
2. Z3. Policz zamówienia w każdym statusie, od najliczniejszego (`$group`, `$sum: 1`, `$sort`).
3. Z4. Oblicz wartość każdego zamówienia i wyświetl 5 najdroższych (`$unwind`, `$group`, `$sort`, `$limit`).
4. Z5. Wyświetl 3 klientów (nazwisko) z największą łączną wartością zamówień, pomijając zamówienia anulowane.
5. Z6. Wyświetl 5 najczęściej kupowanych produktów (suma sztuk). Przy równej liczbie sztuk sortuj alfabetycznie.
6. Z7. Policz zamówienia w każdym miesiącu (`$dateToString` z formatem `"%Y-%m"`).
7. Z8. Dla każdego zamówienia wyświetl numer, nazwisko klienta, liczbę pozycji (`$size`) i numer miesiąca (`$month`) — bez `$unwind` (`$project`).
8. Z9. Oblicz przychód z każdej kategorii produktów (bez zamówień anulowanych). Kategoria jest w kolekcji produkty — połącz kolekcje etapem `$lookup` (`localField: "pozycje.id_produktu"`, `foreignField: "_id"`). Porównaj z SQL i wyjaśnij różnicę dla produktu Hub USB-C 7w1.
9. Z10. Znajdź produkty, które nie wystąpiły w żadnym zamówieniu (`$lookup` z kolekcji produkty do zamowienia i warunek na pustą tablicę).

3 Indeks i użytkownik

Plan zapytania w MongoDB wyświetla metoda `explain("executionStats")`: COLLSCAN oznacza przegląd całej kolekcji, IXSCAN — użycie indeksu.

1. Z11. Sprawdź plan zapytania o zamówienia klientów z Krakowa (pole klient.miasto): stage i totalDocsExamined. Załóż indeks na klient.miasto (`createIndex`), powtórz `explain` i porównaj. Wyświetl listę indeksów (`getIndexes`).
2. Z12. Utwórz użytkownika aplikacji `sklep_app` z rolą `readWrite` tylko do bazy sklep (`db.createUser`) i wyświetl użytkowników (`db.getUsers()`). Opisz w sprawozdaniu, co trzeba jeszcze zmienić w pliku `mongod.cfg`, żeby serwer wymagał logowania, i jak wtedy wygląda adres połączenia (`connection string`) — nie wykonuj tej zmiany.

4 Kiedy SQL, a kiedy dokumenty

1. Z13. Uzupełnij w sprawozdaniu tabelę porównania: liczba poleceń/etapów potrzebnych do raportów Z4 i Z9 w SQL i MongoDB, oraz tabelę wyboru bazy dla podanych zastosowań z uzasadnieniem.

W sprawozdaniu umieść

- ☐ plik `zadanie29.js` z potokami Z1–Z12 i plik `zadanie29.sql` z odpowiednikami SQL (komentarze `// Z4, -- Z4`)
- ☐ zrzuty ekranu wyników Z4, Z9 (MongoDB i SQL) oraz obu planów z Z11
- ☐ wyjaśnienie różnicy wyników w Z9
- ☐ tabele porównania i wyboru bazy (Z13)
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Do czego służy etap \$unwind i co by się stało w Z4 bez niego?
2. Który etap potoku odpowiada w SQL klauzuli WHERE, który GROUP BY, a który LEFT JOIN? Czy \$match po \$group odpowiada WHERE czy HAVING?
3. Dlaczego w MongoDB nazwisko klienta zapisano w dokumencie zamówienia, a w SQL tylko id_klienta? Co się stanie po zmianie nazwiska klienta w obu modelach?
4. Podaj dwa zastosowania, w których wybierzesz MongoDB, i dwa, w których wybierzesz bazę relacyjną. Uzasadnij.

Kryteria oceny

Kryterium	Punkty
Z1–Z3: import danych i proste agregacje	2
Z4–Z8: potoki z \$unwind, wyrażeniami i \$project wraz z odpowiednikami SQL	7
Z9–Z10: \$lookup i wyjaśnienie różnic z SQL	3
Z11–Z12: indeks z analizą explain, użytkownik z minimalną rolą	3
Z13 i odpowiedzi na pytania kontrolne	5
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Administracja serwerem bazy danych

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Poznanie podstawowych czynności administratora serwera MySQL/MariaDB: odczyt i zmiana konfiguracji, praca z dziennikami, nadzór nad połączeniami, odczyt metadanych z `information_schema`, konserwacja tabel i zabezpieczenie kont.

Po zajęciach potrafisz:

- odnajduje plik konfiguracyjny i odczytuje zmienne serwera, odróżniając zmianę `SESSION`, `GLOBAL` i w pliku
- włącza i odczytuje dziennik ogólny oraz odnajduje dziennik błędów
- wyświetla aktywne połączenia i przerywa zapytanie lub połączenie poleceniem `KILL`
- odczytuje z `information_schema` rozmiary tabel i baz oraz klucze obce
- sprawdza i konserwuje tabele (`CHECK`, `ANALYZE`, `OPTIMIZE`, `REPAIR`) oraz wykrywa konta bez hasła

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4 — własny serwer lokalny ucznia
- baza sklep (plik `sklep.sql` z lekcji)

Oprogramowanie

- **XAMPP Control Panel** — przyciski Config (`my.ini`), Logs, Shell; Stop/Start modułu MySQL
apachefriends.org
- **Konsola mysql** — dwa okna Shell potrzebne w kroku 3
- **Notatnik++ lub VS Code** — edycja `my.ini`
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z lekcji 1.16 polecenia `CREATE USER`, `GRANT` i `SHOW GRANTS`.

Bezpieczeństwo

- Pracuj tylko na własnym serwerze lokalnym. Zmienne `GLOBAL` i plik `my.ini` dotyczą całego serwera — na wspólnym serwerze szkolnym ich nie zmieniasz.
- Przed edycją `my.ini` zrób kopię pliku (`my.ini.bak`). Błąd w pliku może uniemożliwić uruchomienie MySQL — wtedy przywróć kopię.
- Nie zmieniasz hasła konta `root` w XAMPP bez zgody nauczyciela: `phpMyAdmin` loguje się z hasłem zapisanym w `config.inc.php` i po zmianie przestanie działać.
- `KILL` wykonuj tylko na połączeniach, które sam otworzyłeś. Dziennik ogólny zapisuje wszystkie zapytania (także z danymi) — wyłącz go po ćwiczeniu.

Przebieg zadania

1 Konfiguracja i zmienne serwera

Plik konfiguracyjny XAMPP otworzysz z Panelu sterowania: wiersz MySQL → Config → my.ini. Ustawienia serwera są w sekcji [mysqld].

1. Z1. Wyświetl wersję serwera, katalog danych, port, katalog instalacji, domyślne kodowanie i porównywanie znaków, max_connections i domyślny silnik (SELECT VERSION(), @@datadir, ...). Porównaj port i katalog danych z wpisami w my.ini.
2. Z2. Wyświetl wartość innodb_buffer_pool_size poleceniem SHOW VARIABLES LIKE i przelicz ją na MB. Znajdź odpowiadający wpis w my.ini.
3. Z3. Ustaw wait_timeout na 120 tylko dla swojej sesji (SET SESSION) i porównaj SHOW SESSION VARIABLES z SHOW GLOBAL VARIABLES. Ustaw max_connections na 200 poleceniem SET GLOBAL, sprawdź wartość, zrestartuj MySQL w Panelu (Stop, Start) i sprawdź ponownie.
4. Z4. Zrób kopię my.ini, dopisz w sekcji [mysqld] wiersz `max_connections = 200`, zrestartuj MySQL i sprawdź wartość. Wyświetl też Uptime, Threads_connected, Questions i Max_used_connections (SHOW GLOBAL STATUS).

2 Dzienniki serwera

Dziennik błędów zapisuje uruchomienia, zatrzymania i awarie serwera. Dziennik ogólny (general log) — każde polecenie klientów.

1. Z5. Odczytaj ścieżkę dziennika błędów (@@log_error) i otwórz go (Panel → Logs albo w edytorze). Znajdź wpisy z restartu z Z3 i zapisz dwa z nich.
2. Z6. Ustaw zapis dziennika ogólnego do tabeli (log_output = 'TABLE'), włącz go (general_log = ON), wykonaj dwa dowolne zapytania do bazy sklep i odczytaj 5 ostatnich wpisów z tabeli mysql.general_log. Wyłącz dziennik i przywróć log_output = 'FILE'.

3 Połączenia: PROCESSLIST i KILL

Otwórz dwa okna konsoli (Shell → `mysql -u root`). Okno A — „użytkownik”, okno B — „administrator”.

1. Z7. W oknie A uruchom `SELECT SLEEP(120);`. W oknie B wyświetl połączenia (SHOW PROCESSLIST oraz zapytanie do information_schema.PROCESSLIST z kolumnami ID, USER, COMMAND, TIME, STATE, INFO) i przerwij zapytanie z okna A poleceniem KILL QUERY. Zapisz komunikat w oknie A.
2. Z8. W oknie A rozpocznij transakcję i zmień stan produktu o id 1 w bazie sklep, nie zatwierdzając jej. W drugiej sesji (np. phpMyAdmin) spróbuj zmienić ten sam wiersz. W oknie B odszukaj obie sesje w PROCESSLIST, zakończ połączenie A poleceniem KILL i sprawdź, co stało się z obiema zmianami.

4 Metadane i konserwacja tabel

information_schema to wbudowana baza z opisem wszystkich obiektów serwera.

performance_schema zbiera statystyki wydajności — w MariaDB jest domyślnie wyłączona, w MySQL 8.4 włączona.

1. Z9. Wyświetl tabele bazy sklep z silnikiem, liczbą wierszy i rozmiarem danych i indeksów w KB, od największej. Następnie podaj liczbę tabel i rozmiar w MB każdej bazy użytkownika (bez baz systemowych).
2. Z10. Wyświetl wszystkie klucze obce bazy sklep (tabela, kolumna, tabela i kolumna wskazywana) z information_schema.KEY_COLUMN_USAGE. Sprawdź wartość @@performance_schema i zapisz, czy jest włączona na twoim serwerze.
3. Z11. Wykonaj CHECK TABLE i ANALYZE TABLE dla tabel produkty i zamowienia oraz OPTIMIZE TABLE dla produkty. Zapisz komunikat OPTIMIZE dla InnoDB. Utwórz testową tabelę z silnikiem Aria (w MySQL: MyISAM), wstaw dwa wiersze, wykonaj CHECK TABLE i REPAIR TABLE, a potem ją usuń.

5 Bezpieczeństwo kont

1. Z12. Wyświetl wszystkie konta (user, host, plugin) z mysql.user i oznacz konta bez hasła (puste authentication_string) oraz konta anonimowe (pusta nazwa użytkownika).
2. Z13. Utwórz konto `admin_<twoje_inicjały>@localhost` z silnym hasłem i uprawnieniami ALL do bazy sklep, sprawdź SHOW GRANTS, zmień mu hasło poleceniem ALTER USER i sprawdź logowanie nowym i starym hasłem. Na koniec usuń konto.
3. Z14. Zapisz w sprawozdaniu listę kontrolną zabezpieczenia instalacji XAMPP (co najmniej 5 punktów) na podstawie wyników Z12 i lekcji, w tym: hasło root i zmiana logowania phpMyAdmin na `auth_type = 'cookie'` w config.inc.php, usunięcie kont anonimowych, bind-address, konta aplikacji z minimalnymi uprawnieniami.

W sprawozdaniu umieść

- ☐ plik `zadanie30.sql` z poleceniami Z1–Z13 opisanymi komentarzami
- ☐ zrzut fragmentu my.ini z dopisaną zmienną i wyniki Z3 przed i po restarcie
- ☐ dwa wpisy z dziennika błędów i zrzut wyniku z mysql.general_log
- ☐ zrzuty z obu okien konsoli z Z7 i opis przebiegu Z8
- ☐ lista kontrolna zabezpieczenia (Z14) i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się SET SESSION, SET GLOBAL i wpis w my.ini? Która zmiana przetrwa restart serwera?
2. Co stało się z niezatwierdzoną zmianą z okna A po poleceniu KILL w Z8 i dlaczego?
3. Dlaczego OPTIMIZE TABLE dla InnoDB wyświetla komunikat „doing recreate + analyze instead” i kiedy warto go użyć?
4. Dlaczego konto bez hasła jest groźne, nawet jeśli serwer nasłuchuje tylko na localhost?

Kryteria oceny

Kryterium	Punkty
Z1-Z4: konfiguracja i zmienne (SESSION/GLOBAL/my.ini)	4
Z5-Z6: dziennik błędów i dziennik ogólny	3
Z7-Z8: PROCESSLIST i KILL z opisem skutków	3
Z9-Z11: information_schema i konserwacja tabel	4
Z12-Z14: konta i lista kontrolna zabezpieczeń	3
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Kopie zaawansowane i odtwarzanie do punktu w czasie

Czas: 2 × 45 min

praca indywidualna lub w parach

Cel: Wykonanie pełnej kopii bazy spójnej w czasie, włączenie i odczyt dziennika binarnego oraz odtworzenie bazy po błędzie użytkownika do chwili tuż przed błędem (point-in-time recovery) z zachowaniem późniejszych poprawnych zmian.

Po zajęciach potrafisz:

- włącza dziennik binarny w pliku konfiguracyjnym i sprawdza jego stan
- wykonuje kopię mysqldump --single-transaction --flush-logs z zapisaną pozycją binlogu
- odczytuje zdarzenia z dziennika binarnego (SHOW BINLOG EVENTS, mysqlbinlog) i znajduje pozycję błędu
- odtwarza bazę z kopii i dogrywa zmiany z binlogu do pozycji lub czasu (--stop-position, --stop-datetime) oraz za błędem (--start-position)
- weryfikuje odtworzone dane i opisuje procedurę kopii dla firmy

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4 — własny serwer ucznia
- baza sklep (plik sklep.sql — pozwala w każdej chwili zacząć od nowa)
- folder roboczy, np. `C:\kopie`

Oprogramowanie

- **XAMPP (MariaDB, konsola)** — XAMPP Control Panel → Shell; programy mysqldump i mysqlbinlog są w katalogu mysql\bin
apachefriends.org
- **Wiersz polecenia (cmd)** — używaj cmd/Shell XAMPP, nie PowerShell — PowerShell nie obsługuje przekierowania `<plik.sql`
- **Notatnik++ lub VS Code** — edycja my.ini i podgląd kopii
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z lekcji 1.17 polecenie mysqldump i odtwarzanie kopii, a z lekcji 2.14 — do czego służy dziennik binarny (binlog).

Bezpieczeństwo

- Pracuj tylko na własnym serwerze. Zrób kopię pliku my.ini przed edycją.
- Nie usuwaj ręcznie plików mysql-bin.* z katalogu danych serwera — do czyszczenia służy polecenie PURGE BINARY LOGS. Przed odtwarzaniem skopiuj potrzebny plik binlogu do folderu z kopiami.
- Kopie zawierają wszystkie dane bazy (także osobowe) — przechowuj je w bezpiecznym miejscu i usuń z komputera w pracowni po zajęciach.
- Błąd w kroku 3 jest celowy i dotyczy tylko bazy sklep na twoim serwerze; w razie pomyłki zaimportuj sklep.sql i zacznij od kroku 2.

Przebieg zadania

1 Włączenie dziennika binarnego

W MySQL 8.4 dziennik binarny jest włączony domyślnie (pliki binlog.000001...). W XAMPP (MariaDB) trzeba go włączyć.

```
[mysqld]
log_bin      = mysql-bin
server_id    = 1
binlog_format = ROW
```

1. Zaimportuj od nowa bazę sklep (plik sklep.sql), żeby zacząć od znanego stanu.
2. Z1. Sprawdź `SHOW VARIABLES LIKE 'log_bin';`. Jeśli wynik to OFF: zrób kopię my.ini (Panel XAMPP → MySQL → Config), dopisz w sekcji [mysqld] wiersze z ramki (jeśli w pliku jest zakomentowany wiersz log-bin, nie dubluj go), zapisz i zrestartuj MySQL. Sprawdź ponownie log_bin, binlog_format i listę plików (SHOW BINARY LOGS).

2 Pełna kopia i normalna praca

Opcja --single-transaction robi spójną kopię tabel InnoDB bez blokowania pracy, --flush-logs zaczyna po kopii nowy plik binlogu, a --master-data=2 (w MySQL 8.4: --source-data=2) zapisuje w kopii komentarz z pozycją binlogu.

```
cd C:\kopie
mysqldump -u root --single-transaction --flush-logs --master-data=2 --routines --databases
sklep > pelna.sql
rem MySQL 8.4: zamiast --master-data=2 użyj --source-data=2
```

1. Z2. Wykonaj kopię poleceniem z ramki. Odszukaj w pliku pelna.sql wiersz „CHANGE MASTER TO MASTER_LOG_FILE=...” i zapisz nazwę pliku i pozycję. Porównaj z wynikiem `SHOW MASTER STATUS` (MariaDB) / `SHOW BINARY LOG STATUS` (MySQL 8.4).
2. Z3. Wykonaj „normalną pracę sklepu”: dodaj nowe zamówienie klientki o id 3 z jedną pozycją (karta graficzna, id 12, cena 2899) i obniż cenę produktu o id 1 do 3199 zł. Zapisz w sprawozdaniu wynik `SELECT NOW();` po tych zmianach.
3. Oczekaj co najmniej 2 sekundy (żeby czasy zdarzeń się różniły).

3 „Przypadkowy” błąd i dalsza praca

Symulujemy pomyłkę pracownika: polecenie bez warunku WHERE.

1. Z4. Wykonaj `DELETE FROM pozycje_zamowienia;` i sprawdź, ile pozycji zostało. Oczekaj 2 sekundy.
2. Z5. Wykonaj jeszcze jedną poprawną zmianę po błędzie: zarejestruj nowego klienta (dowolne dane, miasto Lublin). Zapisz liczbę klientów i pozycji zamówień.

4 Analiza dziennika binarnego

Format ROW zapisuje zmienione wiersze; MariaDB dołącza też treść polecenia (Annotate_rows).

```
SHOW BINLOG EVENTS IN 'mysql-bin.000002';

rem w cmd, w katalogu danych serwera (ścieżka z SELECT @@datadir):
mysqlbinlog -v --base64-output=DECODE-ROWS mysql-bin.000002 > C:\kopie\binlog.txt
```

1. Z6. Wyświetl zdarzenia z pliku binlogu utworzonego po kopii (nazwa z Z2). Znajdź transakcję z DELETE: zapisz pozycję jej początku (kolumna Pos wiersza Gtid / Query BEGIN przed DELETE) i pozycję końca (End_log_pos wiersza Xid/COMMIT).
2. Z7. Zapisz dziennik do pliku tekstowego mysqlbinlog z ramki i odszukaj w nim DELETE oraz godzinę zdarzenia (wiersze zaczynające się od #rrmdd gg:mm:ss). Policz, ile wierszy „### DELETE FROM” zawiera.

5 Odtwarzanie do punktu w czasie

Skopiuj plik binlogu z katalogu danych do C:\kopie (np. jako kopia-bin.000002). Wszystkie polecenia wykonuj w cmd w folderze C:\kopie.

```
mysql -u root < pelna.sql
mysqlbinlog --stop-position=POZ_POCZĄTKU_DELETE kopia-bin.000002 | mysql -u root
mysqlbinlog --start-position=POZ_KOŃCA_DELETE kopia-bin.000002 | mysql -u root
```

1. Z8. Odtwórz pełną kopię i sprawdź stan: liczba zamówień, pozycji, klientów i cena produktu 1. Którą chwilę odpowiada ten stan?
2. Z9. Dograj zmiany z binlogu do pozycji tuż przed DELETE (--stop-position) i sprawdź te same wartości.
3. Z10. Dograj zmiany zapisane po błędzie (--start-position za transakcją DELETE) i sprawdź, czy klient z Lublina wrócił, a pozycje zamówień są na miejscu.
4. Z11. Powtórz odtworzenie w wariantcie czasowym: pelna.sql, a potem `mysqlbinlog --stop-datetime="rrrr-mm-dd gg:mm:ss"` z czasem z Z3. Porównaj wynik z Z9 i Z10 — czego brakuje i dlaczego?

W sprawozdaniu umieść

- ☐ zrzut fragmentu my.ini i wyników Z1
- ☐ wiersz CHANGE MASTER z pelna.sql i wynik SHOW MASTER STATUS (Z2)
- ☐ fragment SHOW BINLOG EVENTS z zaznaczoną transakcją DELETE i fragment binlog.txt (Z6–Z7)
- ☐ tabela stanów bazy po Z5, Z8, Z9, Z10 i Z11
- ☐ procedura kopii zapasowych dla małego sklepu (harmonogram pełnych kopii, przechowywanie binlogów, test odtwarzania) i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego sama kopia mysqldump z poprzedniej nocy nie wystarcza, gdy błąd zdarzył się po południu?
2. Do czego służą opcje --single-transaction i --flush-logs w poleceniu kopii?
3. Czym różni się odtwarzanie do pozycji od odtwarzania do czasu? Kiedy --stop-datetime może dać zły wynik?
4. Dlaczego plików binlogu nie wolno przechowywać tylko na tym samym dysku co baza i jak długo trzeba je trzymać?

Kryteria oceny

Kryterium	Punkty
Z1–Z2: włączenie binlogu i spójna kopia z pozycją	4
Z3–Z7: scenariusz i analiza dziennika binarnego	5
Z8–Z11: odtworzenie do pozycji, za błędem i do czasu z weryfikacją	6
Procedura kopii dla sklepu	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Próbnny egzamin INF.03 — baza danych i strona PHP (kino „Kadr”)

Czas: 2 × 45 min

praca samodzielna, warunki egzaminacyjne

Cel: Wykonanie w 90 minut kompletnego zadania egzaminacyjnego: utworzenie bazy z pliku, zapisanie czterech zapytań ze zrzutami oraz zbudowanie strony internetowej z arkuszem stylów i skryptem PHP pobierającym dane z bazy przez mysqli.

Po zajęciach potrafisz:

- importuje bazę danych i wykonuje zapytania SELECT (z WHERE, ORDER BY, JOIN, GROUP BY) oraz UPDATE
- dokumentuje zapytania zrzutami ekranu i plikiem kwerendy.txt
- buduje stronę z sekcjami header, main (dwa panele) i footer zgodnie z opisem wyglądu
- pisze skrypt PHP: mysqli_connect, mysqli_query, pętla mysqli_fetch_assoc, mysqli_close
- sprawdza pracę z kryteriami oceny w stylu egzaminacyjnym

Sprzęt i materiały

- komputer z XAMPP (Apache, MariaDB, PHP 8) i przeglądarką
- pliki `r2_32_kino.sql` i `r2_32_kadr.png` z folderu Pliki_do_zadan
- bez dostępu do internetu i materiałów (jak na egzaminie)

Oprogramowanie

- **XAMPP** — uruchom moduły Apache i MySQL; pliki strony w `C:\xampp\htdocs\kino`
[apachefriends.org](http://www.apachefriends.org)
- **phpMyAdmin** — `http://localhost/phpmyadmin`
- **Edytor kodu (VS Code, Notepad++)** — bez wtyczek generujących kod
code.visualstudio.com

Przygotowanie przed zajęciami

- Powtórz z lekcji 1.18 i 2.13 budowę skryptu PHP z mysqli oraz typowe układy stron z egzaminu (baner, panele, stopka).

Bezpieczeństwo

- Konto root bez hasła stosujemy tylko na lokalnym serwerze ćwiczeniowym, tak jak na egzaminie. W prawdziwej aplikacji używa się osobnego konta z hasłem (zadanie 2.19).
- Przed zapytaniem UPDATE (kw4) upewnij się, że ma poprawny warunek WHERE — po pomyłce zaimportuj plik `r2_32_kino.sql` ponownie.

Przebieg zadania

1 Przygotowanie stanowiska (5 min)

Utwórz na pulpicie folder `nazwisko_imie` (bez polskich znaków). Zapisuj w nim wszystkie wyniki pracy.

1. Uruchom Apache i MySQL w XAMPP. W phpMyAdmin zaimportuj plik `r2_32_kino.sql` — sam tworzy bazę `kino` z tabelami `filmy` i `seanse` (relacja: `seanse.id_filmu` → `filmy.id`).
2. Skopiuj plik `r2_32_kadr.png` do folderu `C:\xampp\htdocs\kino` pod nazwą `kadr.png`.

2 Operacje na bazie danych (25 min)

Każde zapytanie wykonaj w phpMyAdmin, zrób zrzut ekranu wyniku (cały wynik i treść zapytania) i zapisz jako `kw1.png` ... `kw4.png`. Treść zapytań zapisz w pliku `kwerendy.txt`, każde zakończone średnikiem.

1. Z1 (kw1). Wybierz tytuł, rok produkcji i ocenę filmów z gatunku komedia. Posortuj malejąco według roku.
2. Z2 (kw2). Wybierz godzinę seansu, tytuł filmu i numer sali dla seansów z 14 listopada 2026 r. Posortuj rosnąco według godziny.
3. Z3 (kw3). Dla każdego gatunku wybierz jego nazwę i liczbę filmów. Kolumnę z liczbą nazwij `liczba_filmow`.
4. Z4 (kw4). Obniż o 4 zł cenę wszystkich seansów rozpoczynających się przed godziną 14:00. Na zrzucie ma być widoczna liczba zmienionych wierszy.

3 Strona internetowa (50 min)

Utwórz w folderze `C:\xampp\htdocs\kino` pliki `kino.php` i `styl.css`. Strona ma być zgodna ze standardem HTML5, z językiem polskim i kodowaniem UTF-8; tytuł karty: „Kino Kadr — repertuar”.

1. Z5. Struktura: baner (header) z nagłówkiem pierwszego stopnia „Kino studyjne KADR”; część główna z dwoma panelami — lewym i prawym; stopka (footer) z akapitem „Stronę wykonał: ” i twoim imieniem i nazwiskiem.
2. Z6. Panel lewy: obraz `kadr.png` z tekstem alternatywnym „Logo kina Kadr”, nagłówek drugiego stopnia „Cennik” i lista nienumerowana z trzema punktami: „bilet normalny — według seansu”, „bilet ulgowy — 20% taniej”, „seanse przed 14:00 — taniej o 4 zł”.
3. Z7. Panel prawy: nagłówek drugiego stopnia „Repertuar na 14 listopada 2026” i skrypt PHP, który łączy się z bazą `kino` (serwer `localhost`, użytkownik `root`, bez hasła), wysyła zapytanie z `kw2` rozszerzone o cenę biletu i dla każdego seansu wypisuje akapit w postaci: godzina pogrubiona (tylko gg:mm), spacja, tytuł, „ — sala ”, numer sali, „, „, cena, „ zł”. Na końcu skrypt zamyka połączenie.
4. Z8. Styl CSS: wszystkie elementy krojem Verdana; tło strony `#F4F1EA`; baner i stopka — tło `#1B2A41`, biały kolor czcionki, wyrównanie baneru do środka, stopki do prawej, marginesy wewnętrzne 10 px; panel lewy — szerokość 30%, wysokość 500 px, tło `#E9C46A`; panel prawy — szerokość 70%, wysokość 500 px, tło białe (panele obok siebie, margines wewnętrzny 10 px nie może psuć układu); obraz — szerokość 90%, wyśrodkowany, zaokrąglone rogi 10 px; nagłówki drugiego stopnia — kolor `#1B2A41` i dolne obramowanie 2 px ciągłe w tym kolorze; akapit w panelu prawym po najechaniu kursorem — tło `#E9C46A`.
5. Z9. Zrób zrzut całej strony w przeglądarce (`http://localhost/kino/kino.php`) i zapisz jako `strona.png`. Skopiuj `kino.php`, `styl.css` i `kadr.png` do folderu `nazwisko_imie`.

4 Samoocena (10 min)

1. Porównaj pracę z kryteriami w sprawozdaniu (nazwy plików, średniki w kwerendy.txt, kompletność zrzutów, poprawność kodu w walidatorze — jeśli nauczyciel na to pozwoli po zakończeniu części egzaminacyjnej). Wypełnij tabelę samooceny.

W sprawozdaniu umieść

- ☐ folder `nazwisko_imie` z plikami kw1.png–kw4.png, kwerendy.txt, kino.php, styl.css, kadr.png i strona.png
- ☐ tabela samooceny z punktami za każdy element
- ☐ lista błędów, które popełniłeś, i jak je poprawisz na egzaminie

Pytania kontrolne

1. Dlaczego w zapytaniu kw2 trzeba złączyć tabele i jak zapisać warunek złączenia?
2. Co zwraca `mysqli_fetch_assoc` i dlaczego pętla `while` kończy się po ostatnim wierszu?
3. Jak w CSS ustawić dwa panele obok siebie, aby stopka znalazła się pod nimi?
4. Jak w PHP wyświetlić tylko gg:mm z wartości 11:00:00 i jak to samo zrobić w SQL?

Kryteria oceny

Kryterium	Punkty
Zapytania kw1–kw4: poprawne wyniki, zrzuty, plik kwerendy.txt	6
Struktura strony HTML (Z5–Z6): sekcje, nagłówki, obraz z alt, lista, stopka	4
Skrypt PHP (Z7): połączenie, zapytanie, pętla z formatem wyniku, zamknięcie połączenia	5
Styl CSS (Z8) zgodny z opisem	4
Samoocena i komplet plików w folderze	1
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Projekt końcowy cz. 1 — baza danych z logiką

Czas: 2 × 45 min

praca w zespołach 2–3-osobowych

Cel: Zaprojektowanie i wykonanie w zespole bazy danych dla wybranej dziedziny wraz z logiką po stronie serwera (widoki, procedura z transakcją, wyzwalacz audytu), indeksami i kontem aplikacji z minimalnymi uprawnieniami — jako podstawy aplikacji PHP w części 2.

Po zajęciach potrafisz:

- analizuje wymagania dziedziny i projektuje schemat ERD w 3NF z kluczami i ograniczeniami
- pisze skrypt tworzący bazę z danymi, który wykonuje się bez błędów na czystym serwerze
- tworzy widoki, procedurę składowaną z transakcją i walidacją oraz wyzwalacz audytu
- dobiera indeksy na podstawie zapytań i dokumentuje ich działanie w EXPLAIN
- tworzy konto aplikacji z minimalnymi uprawnieniami i sprawdza odmowę dostępu

Sprzęt i materiały

- komputer z XAMPP (MariaDB 10.4) lub MySQL 8.4 dla każdego członka zespołu
- wspólny folder zespołu (dysk sieciowy, pendrive lub repozytorium Git)

Oprogramowanie

- **XAMPP (MariaDB, phpMyAdmin)** — konsola mysql do testów kont i procedur apachefriends.org
- **draw.io (diagrams.net)** — diagram ERD (biblioteka Entity Relation) app.diagrams.net
- **VS Code** — plik projekt.sql code.visualstudio.com
- **Git (opcjonalnie)** — wspólna praca nad plikami zespołu git-scm.com

Przygotowanie przed zajęciami

- Zespół wybiera dziedzinę z listy w kroku 1 (lub własną po akceptacji nauczyciela) i zapisuje 5 typowych czynności użytkownika systemu (np. „recepcja przyjmuje zlecenie”, „mechanik dodaje części”).
- Przejrzyj swoje rozwiązania z zadań 2.02 (widoki), 2.08–2.12 (procedury, wyzwalacze, transakcje, indeksy) i 1.16 (uprawnienia).

Bezpieczeństwo

- Baza zawiera wyłącznie dane fikcyjne — nie wpisujcie prawdziwych danych osobowych kolegów ani klientów.
- Hasło konta aplikacji wymyślcie na potrzeby projektu; nie używajcie haseł z innych serwisów. W dokumentacji zapiszcie je tylko w pliku konfiguracyjnym, nie w sprawozdaniu.
- Skrypt projekt.sql zaczyna się od DROP DATABASE IF EXISTS tylko dla bazy projektu — sprawdźcie nazwę, żeby nie usunąć innej bazy.
- Przed testem „na czystym serwerze” upewnijcie się, że macie aktualną kopię pliku projekt.sql we wspólnym folderze.

Przebieg zadania

1 Dziedzina i wymagania

Przykładowe dziedziny: warsztat samochodowy, przychodnia weterynaryjna, siłownia z karnetami, szkoła językowa (kursy i zapisy), hotel (pokoje i rezerwacje), biblioteka gier planszowych, firma cateringowa (menu i zamówienia).

1. Zapiszcie w sprawozdaniu nazwę dziedziny, podział ról w zespole i 5 czynności użytkowników.
2. Do każdej czynności dopiszcie, jakie dane są potrzebne i które z nich wymagają kontroli (np. brak towaru, termin zajęty, limit miejsc).

2 Projekt ERD i schemat

Wymagania minimalne: co najmniej 5 tabel (w tym jedna łącząca relację wiele-do-wielu) i tabela dziennika zmian; 3NF; klucze główne i obce z przemyślanym ON DELETE; NOT NULL, UNIQUE i co najmniej jedno ograniczenie CHECK; kodowanie utf8mb4_polish_ci.

1. Narysujcie diagram ERD w draw.io (encje, atrybuty, klucze, krotności) i zapiszcie jako `erd.png`.
2. Napiszcie w pliku `projekt.sql` polecenia CREATE DATABASE i CREATE TABLE. Przy każdym kluczu obcym dopiszcie komentarz, dlaczego wybraliście RESTRICT, CASCADE albo SET NULL.
3. Dodajcie dane testowe: co najmniej 8–10 wierszy w głównych tabelach, tak aby dało się przetestować wszystkie zapytania i błędy.

3 Logika w bazie

Każdy obiekt opatrzcie komentarzem: do czego służy i która czynność z kroku 1 go używa.

1. Utwórzcie co najmniej dwa widoki: raport z JOIN i GROUP BY (np. koszt zleceń, obłożenie zajęć) oraz widok dla aplikacji bez kolumn wrażliwych.
2. Napiszcie procedurę wykonującą główną operację biznesową (np. dodanie części do zlecenia, rezerwacja) z parametrami IN, transakcją, blokadą wierszy (SELECT ... FOR UPDATE) i walidacją przez SIGNAL SQLSTATE '45000' z czytelnym komunikatem; w razie błędu ROLLBACK.
3. Utwórzcie wyzwalacz audytu zapisujący do tabeli dziennika zmianę ważnej kolumny (OLD i NEW, użytkownik — funkcja USER(), czas).
4. Przetestujcie procedurę: jedno poprawne wywołanie i co najmniej dwa wywołania kończące się waszymi komunikatami błędów. Zapiszcie wyniki.

4 Wydajność i bezpieczeństwo

1. Wybierzcie dwa zapytania, które aplikacja będzie wykonywać najczęściej. Pokażcie EXPLAIN przed i po dodaniu indeksów (co najmniej jeden indeks złożony) i uzasadnijcie kolejność kolumn.
2. Utwórzcie konto aplikacji `<baza>_app @ localhost` z minimalnymi uprawnieniami: SELECT na widokach, tylko potrzebne SELECT/INSERT/UPDATE na tabelach, EXECUTE na procedurze; bez DELETE i bez dostępu do tabeli dziennika. Sprawdźcie, że zalogowany tym kontem użytkownik może wywołać procedurę, a DELETE kończy się błędem 1142.
3. Sprawdźcie, czy wyzwalacz zapisuje w dzienniku nazwę konta aplikacji, gdy zmiana pochodzi z tego konta.

5 Test na czystym serwerze i oddanie

1. Wykonajcie cały plik `projekt.sql` na komputerze innego członka zespołu (albo po usunięciu bazy) poleceniem `mysql -u root < projekt.sql` i poprawcie wszystkie błędy, aż plik wykona się w całości.
2. Dopiszcie na końcu pliku zakomentowane polecenia testowe (CALL, UPDATE wyzwalający audyt, EXPLAIN), żeby nauczyciel mógł je uruchomić.
3. Oddajcie folder zespołu: projekt.sql, erd.png, plik z poleceniami tworzącymi konto (bez hasła w treści sprawozdania) i sprawozdanie.

W sprawozdaniu umieść

- ☐ opis dziedziny, podział ról i 5 czynności użytkowników z kontrolami
- ☐ diagram ERD (erd.png) i uzasadnienie wyboru ON DELETE
- ☐ wyniki testów procedury (poprawne wywołanie i dwa błędy) oraz zawartość tabeli dziennika
- ☐ plany EXPLAIN przed i po indeksach z uzasadnieniem
- ☐ wynik SHOW GRANTS konta aplikacji i zrzut odmowy DELETE (błąd 1142)

Pytania kontrolne

1. Dlaczego główną operację biznesową umieściliście w procedurze z transakcją, a nie jako kilka osobnych zapytań w aplikacji?
2. Po co w procedurze SELECT ... FOR UPDATE? Co mogłoby się stać bez blokady przy dwóch równoczesnych użytkownikach?
3. Dlaczego konto aplikacji nie ma uprawnień DELETE ani dostępu do tabeli dziennika, a mimo to wyzwalacz może do niej pisać?
4. Czym różni się w wyzwalaczu wynik funkcji USER() od CURRENT_USER()?

Kryteria oceny

Kryterium	Punkty
Wymagania i ERD: kompletność, 3NF, poprawne krotności	3
Schemat i dane: klucze, ograniczenia, uzasadnione ON DELETE, plik wykonuje się na czystym serwerze	4
Widoki (2) i procedura z transakcją, blokadą i walidacją SIGNAL, przetestowana	5
Wyzwalacz audytu działający także dla konta aplikacji	2
Indeksy z EXPLAIN przed/po i uzasadnieniem	2
Konto aplikacji z minimalnymi uprawnieniami i test odmowy	2
Odpowiedzi na pytania i udział każdego członka zespołu (ocena nauczyciela)	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).

Projekt końcowy cz. 2 — aplikacja PHP, kopia i prezentacja

Czas: 2 × 45 min

praca w zespołach 2–3-osobowych (te same co w zadaniu 33)

Cel: Zbudowanie w zespole prostej aplikacji PHP (PDO, zapytania przygotowane) korzystającej z bazy z części 1 przez konto z minimalnymi uprawnieniami, wykonanie i sprawdzenie kopii zapasowej, przygotowanie dokumentacji i 5-minutowej prezentacji projektu.

Po zajęciach potrafisz:

- łączy się z bazą przez PDO z trybem wyjątków, korzystając z pliku konfiguracyjnego poza katalogiem publicznym
- wyświetla i zapisuje dane zapytaniami przygotowanymi oraz wywołuje procedurę składowaną z obsługą błędów SIGNAL
- zabezpiecza wypisywane dane (`htmlspecialchars`) i parametry (biała lista, walidacja)
- wykonuje kopię bazy z procedurami i wyzwalaczami i sprawdza jej odtworzenie
- dokumentuje projekt (ERD, opis tabel, instrukcja) i prezentuje go w 5 minut

Sprzęt i materiały

- komputer z XAMPP (Apache, MariaDB, PHP 8) dla każdego członka zespołu
- baza z zadania 33 (projekt.sql) albo — gdy nie działa — wzorcowa baza `r2_34_warsztat_wzor.sql` z folderu Pliki_do_zadan
- rzutnik do prezentacji

Oprogramowanie

- **XAMPP** — aplikacja w `C:\xampp\htdocs\<projekt>`, konfiguracja poza `htdocs` apachefriends.org
- **VS Code** — kod PHP code.visualstudio.com
- **draw.io (diagrams.net)** — ERD do dokumentacji app.diagrams.net
- **Edytor prezentacji (PowerPoint, Impress lub Google Slides)** — maks. 6 slajdów

Przygotowanie przed zajęciami

- Zaimportujcie aktualny projekt.sql z zadania 33 i sprawdźcie, że konto aplikacji działa (logowanie w konsoli `mysql -u <konto> -p`).
- Przypomnijcie sobie z zadań 2.19–2.22 połączenie PDO, `prepare/execute`, `htmlspecialchars` i obsługę wyjątków.

Bezpieczeństwo

- Aplikacja łączy się z bazą wyłącznie kontem `<baza>_app` z części 1, nigdy kontem root.
- Plik z hasłem (config.php) trzymajcie poza katalogiem htdocs i nie umieszczajcie go w repozytorium Git (dodajcie do .gitignore); w dokumentacji opiszcie tylko jego budowę.
- Każdą wartość od użytkownika przekazujcie jako parametr zapytania przygotowanego, a nazwy kolumn do sortowania lub filtry — przez białą listę. Każdą wartość wypisywaną na stronie przepuszczajcie przez htmlspecialchars.
- Użytkownikowi pokazujcie tylko komunikaty z waszych SIGNAL; inne błędy zapisujcie w dzienniku (error_log) i wyświetlajcie ogólny komunikat.
- Kopia bazy zawiera całe dane — przechowujcie ją w folderze zespołu, nie w htdocs.

Przebieg zadania

1 Struktura aplikacji i połączenie

Wymagania minimalne: lista danych z widoku z filtrem, formularz dodawania rekordu, wywołanie procedury z części 1 przez formularz, czytelne komunikaty błędów.

```
C:\xampp\<projekt>\_config\config.php  <- dsn, użytkownik, hasło (return [...])
C:\xampp\htdocs\<projekt>\db.php      <- funkcja połącz(): PDO i funkcja e() =
htmlspecialchars
C:\xampp\htdocs\<projekt>\index.php    <- lista z widoku + filtr
C:\xampp\htdocs\<projekt>\dodaj.php    <- formularz INSERT
C:\xampp\htdocs\<projekt>\operacja.php <- formularz wywołujący procedurę (CALL)
```

1. Utwórzcie pliki według struktury z ramki. W db.php połączenie PDO z charset=utf8mb4, PDO::ERRMODE_EXCEPTION i domyślnym trybem FETCH_ASSOC; plik konfiguracyjny wczytujcie przez require ze ścieżki poza htdocs.
2. Sprawdźcie, że adres `http://localhost/<projekt>/config.php` nie istnieje (plik jest poza katalogiem publicznym).

2 Funkcje aplikacji

1. Strona index.php: tabela danych z widoku z części 1 i filtr (lista select) przekazywany metodą GET. Dozwolone wartości filtra sprawdźcie białą listą (in_array z true), a wartość przekażcie do zapytania jako parametr.
2. Strona dodaj.php: formularz POST dodający rekord do głównej tabeli (zapytanie przygotowane INSERT), walidacja pól (filter_input, wymagane pola), po zapisie przekierowanie (header Location) i komunikat.
3. Strona operacja.php: formularz wywołujący procedurę z części 1 (`CALL nazwa(?, ?, ?)` przez prepare/execute). Błąd z SIGNAL (SQLSTATE 45000 w `$e->errorInfo[0]`) pokażcie użytkownikowi, pozostałe zapiszcie przez error_log i pokażcie ogólny komunikat.
4. Przetestujcie: poprawne dodanie, błędne dane w formularzu, wywołanie procedury kończące się każdym z waszych komunikatów SIGNAL, wartość filtra spoza listy, tekst z apostrofem i znakami < > w polu formularza. Sprawdźcie w tabeli dziennika, że zmiany z aplikacji zapisują nazwę konta aplikacji.

3 Kopia zapasowa i odtworzenie

mysqldump domyślnie zapisuje wyzwalacze, ale NIE zapisuje procedur i funkcji — potrzebna jest opcja --routines.

```
mysqldump -u root --single-transaction --routines --triggers --databases <baza> >
kopia_<baza>.sql
```

1. Wykonajcie kopię poleceniem z ramki (w Shell XAMPP) i sprawdźcie w pliku, że zawiera CREATE PROCEDURE i CREATE TRIGGER. Porównajcie z kopią bez --routines.
2. Odtwórzcie kopię pod inną nazwą bazy albo na komputerze innego członka zespołu i sprawdźcie, że aplikacja działa na odtworzonej bazie (po utworzeniu konta aplikacji).

4 Dokumentacja i prezentacja

Prezentacja: 5 minut, maks. 6 slajdów, każdy członek zespołu mówi; po niej 2–3 pytania nauczyciela lub klasy.

1. Dokumentacja (plik PDF lub DOCX): opis dziedziny, ERD, opis każdej tabeli (kolumny, klucze, ograniczenia), lista widoków, procedur i wyzwalaczy z celem, uprawnienia konta aplikacji, instrukcja instalacji (import bazy, utworzenie konta, plik config.php, adres strony) i procedura kopii.
2. Prezentacja: problem i dziedzina, ERD, najciekawszy element logiki w bazie (procedura lub wyzwalacz), pokaz działania aplikacji z jednym błędem SIGNAL, zabezpieczenia, wnioski i podział pracy.
3. Oddajcie folder zespołu: kod aplikacji (bez config.php z hasłem — tylko config.przyklad.php), projekt.sql, kopia bazy, dokumentacja, prezentacja.

W sprawozdaniu umieść

- ☐ dokumentacja projektu (ERD, opis tabel i obiektów, uprawnienia, instrukcja instalacji, procedura kopii)
- ☐ tabela testów aplikacji (dane wejściowe, oczekiwany i otrzymany wynik)
- ☐ zrzuty: lista z filtrem, poprawne dodanie, komunikat błędu z SIGNAL, wpis w dzienniku z kontem aplikacji
- ☐ plik kopii bazy i opis testu odtworzenia
- ☐ prezentacja i krótka samoocena pracy zespołu

Pytania kontrolne

1. Dlaczego plik config.php leży poza katalogiem htdocs i co mogłoby się stać, gdyby serwer przestał interpretować PHP?
2. Dlaczego filtra statusu nie wystarczy przekazać jako parametr, tylko sprawdza się go też białą listą?
3. Po co przekierowanie po zapisie formularza (POST → przekierowanie → GET)?
4. Co by się stało przy odtwarzaniu kopii wykonanej bez --routines i jak szybko to wykryć?

Kryteria oceny

Kryterium	Punkty
Połączenie PDO, konfiguracja poza httdocs, konto aplikacji (nie root)	3
Lista z filtrem (biała lista + parametr) i formularz INSERT z walidacją i przekierowaniem	4
Wywołanie procedury z obsługą komunikatów SIGNAL i ukrywaniem innych błędów; htmlspecialchars	4
Kopia z procedurami i wyzwalaczami oraz test odtworzenia	2
Dokumentacja (ERD, opis tabel i obiektów, instrukcja)	3
Prezentacja (czas, pokaz działania, udział całego zespołu, odpowiedzi na pytania)	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).