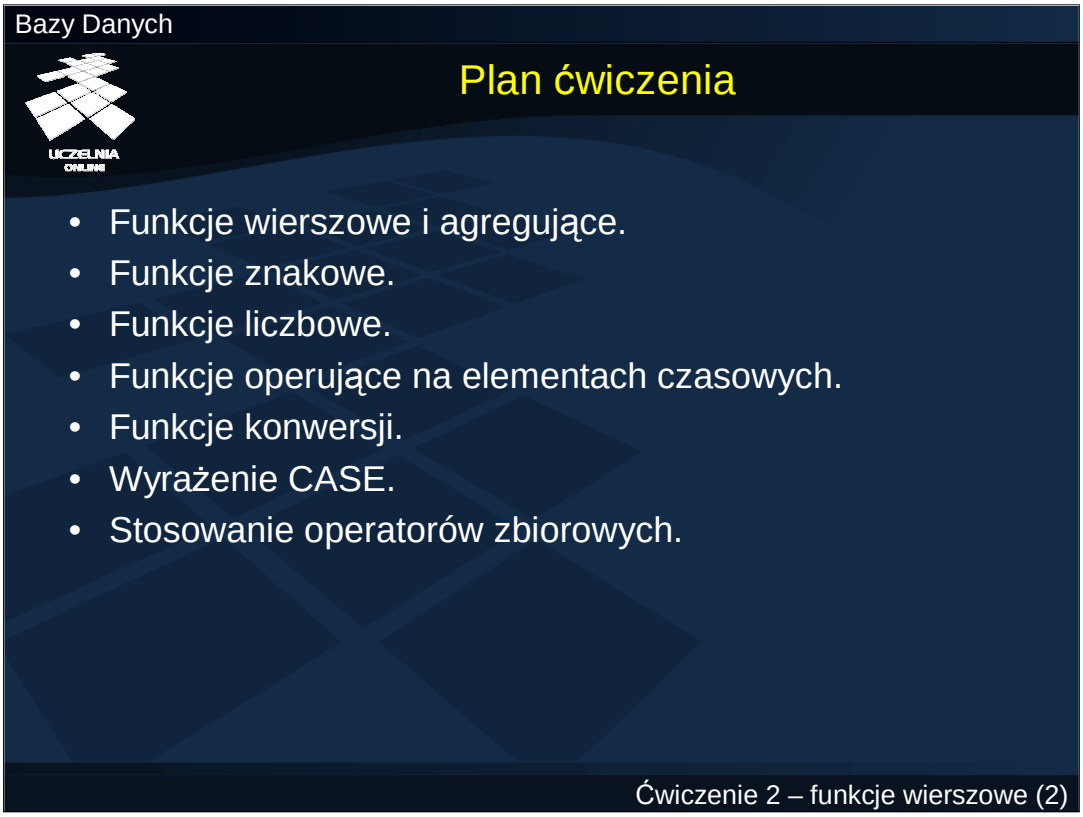
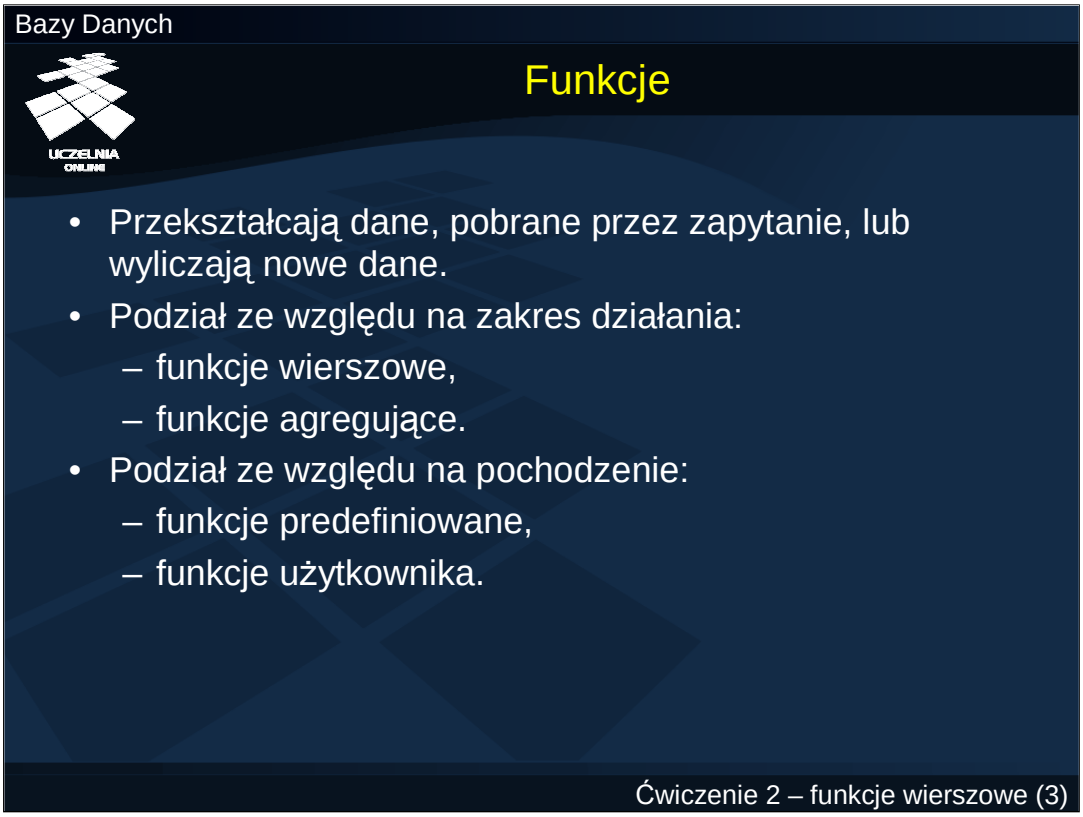
Celem ćwiczenia jest zaprezentowanie zagadnień dotyczących stosowania w zapytaniach języka SQL predefiniowanych funkcji wierszowych oraz budowy poleceń z wykorzystaniem operatorów zbiorowych.

Wymagania:

Umiejętność konstrukcji prostych zapytań w języku SQL.

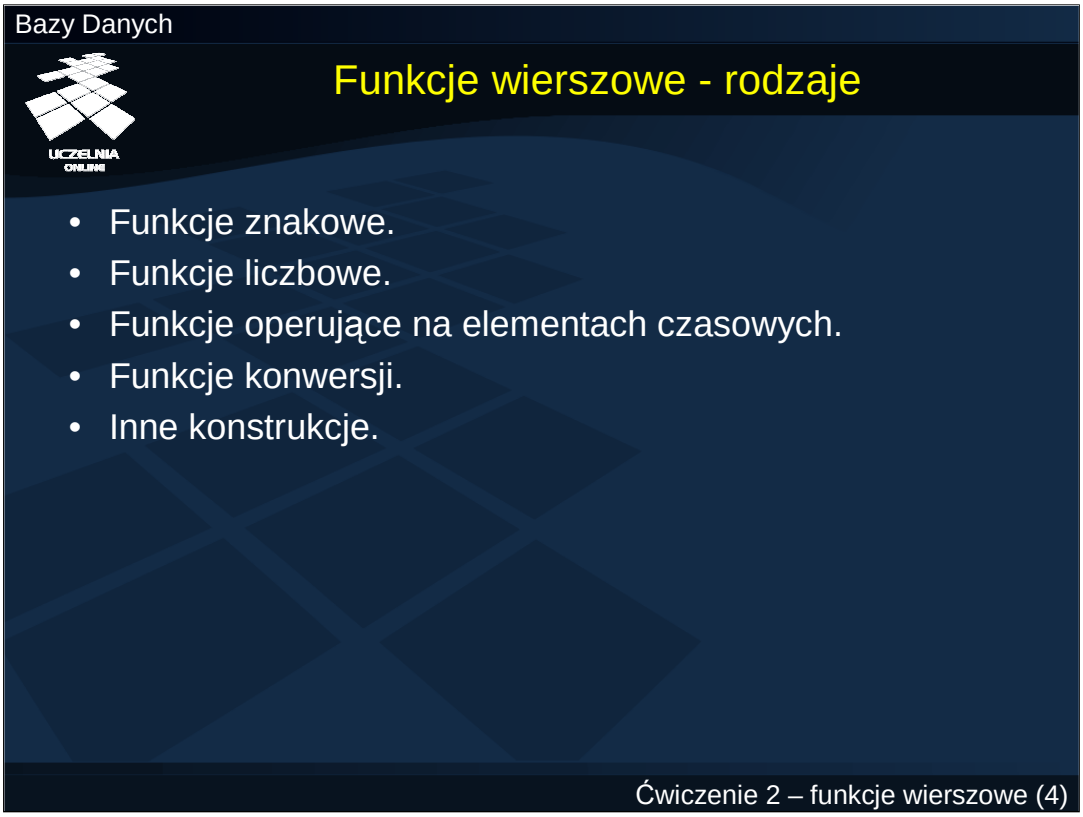


Na początku ćwiczenia zostanie omówiona koncepcja stosowania funkcji w poleceniach SQL. Kolejne zagadnienie to podział funkcji na funkcje wierszowe i agregujące. W bieżącym ćwiczeniu zajmiemy się jedynie funkcjami wierszowymi. Omówienie funkcji wierszowych zostanie przeprowadzone z zachowaniem podziału funkcji wierszowych na grupy w zależności od typów danych, na których funkcje operują. Rozpoczniemy od funkcji znakowych, następnie przedstawione zostaną funkcje liczbowe, funkcje operujące na elementach czasowych oraz funkcje konwersji. Kolejnym zagadnieniem, poruszanym w ćwiczeniu, będzie zastosowanie wyrażenia CASE w konstrukcji zapytań. Ćwiczenie zakończymy omówieniem stosowania operatorów zbiorowych.

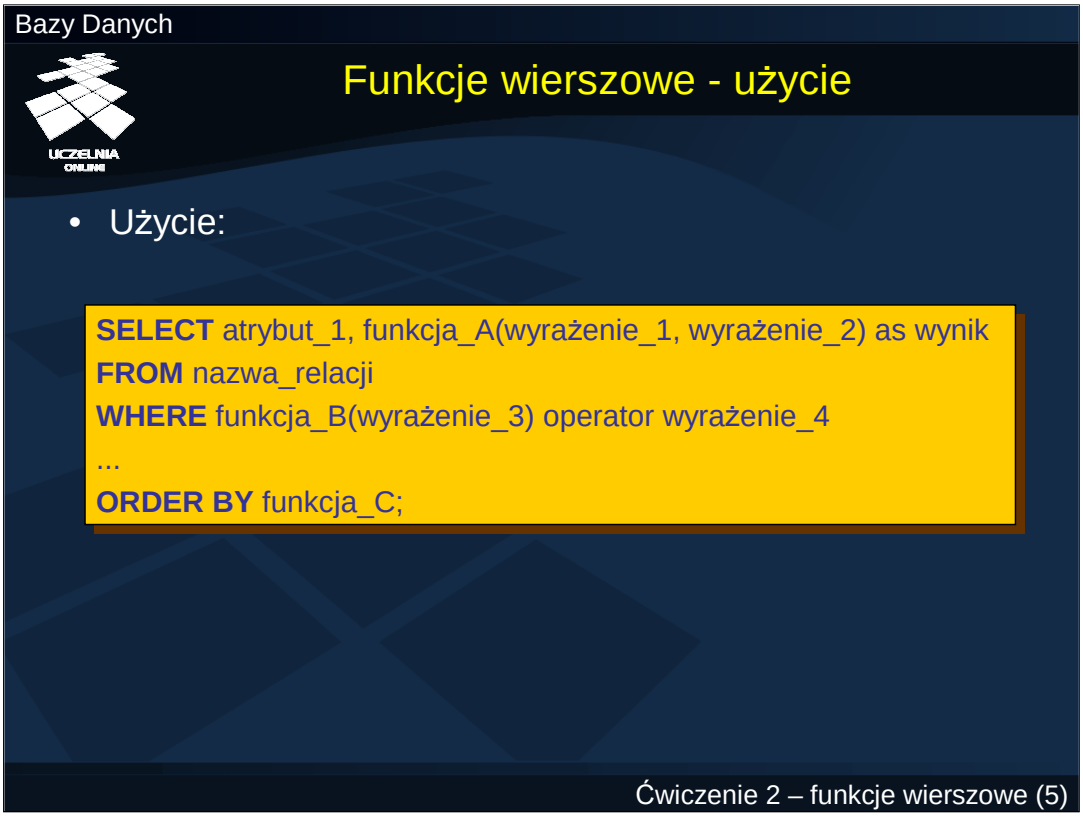


Bardzo często dane, odczytywane przez zapytania w języku SQL z bazy danych, wymagają dodatkowego przetworzenia przed prezentacją użytkownikowi. Niekiedy konieczne jest wyliczenie przez zapytanie nowych wartości, nieobecnych w bazie danych. Przykładami takich operacji są: zamiana wszystkich liter w nazwisku pracownika na wielkie, zaokrąglenie płacy podstawowej pracownika do złotych, odczytanie roku z daty zatrudnienia pracownika czy też wyliczenie sumy płac pracowników zespołu.

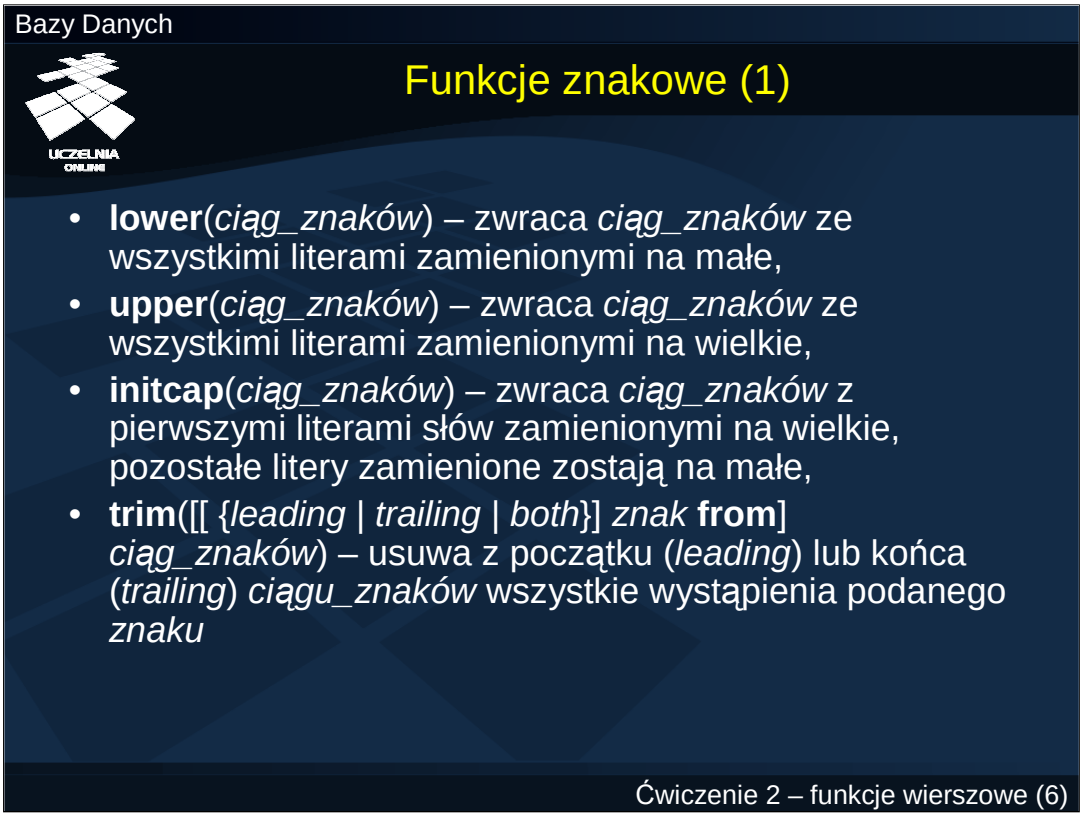
Funkcja może mieć różnych zakres działania. Jeśli funkcja operuje na wartościach atrybutów, znajdujących się w tym samym rekordzie, wówczas jest to tzw. funkcja wierszowa. Z kolei jeśli funkcja operuje na wartościach atrybutów z różnych rekordów, wyliczając na tej podstawie nową wartość, wówczas jest to tzw. funkcja agregująca. Funkcjami agregującymi zajmiemy się w ćwiczeniu 3. System zarządzania bazą danych zwykle dostarcza użytkownikowi zbiór predefiniowanych funkcji, od razu gotowych do użycia. Użytkownik może do tego zbioru dodać swoje własne funkcje, wykorzystując w tym celu język PL/SQL (dokładnie omówienie języka PL/SQL i mechanizmów tworzenia funkcji użytkownika zostanie zamieszczone w ćwiczeniach: 11, 12 i 13.). W niniejszym ćwiczeniu ograniczymy się jedynie do przedstawienia funkcji predefiniowanych.



Rozpoczniemy teraz omawianie najczęściej stosowanych funkcji wierszowych. Funkcje zostaną przedstawione z podziałem ze względu na typ danych, na jakich funkcje działają. Funkcje znakowe jako parametr pobierają ciąg (lub ciągi) znaków, a wyliczają nowe ciągi znaków (np. funkcja dokonująca zamiany liter w ciągu znaków na wielkie) bądź wartości liczbowe (np. funkcja wyliczająca długość ciągu znaków). Funkcje liczbowe jako parametry pobierają liczby i zwracają liczby po przekształceniach (np. funkcja zaokrąglająca daną liczbę do dwóch miejsc po przecinku). Kolejna grupa funkcji, funkcje operujące na elementach czasowych, bądź przekształcają datę podaną jako parametr do innej daty (np. funkcja zwracająca datę, jaka przypadnie od daty podanej jako parametr za trzy miesiące) bądź zwracają wartości liczbowe (np. funkcja wyliczająca liczbę miesięcy pomiędzy dwiema datami). Inne funkcje z tej grupy to np. funkcje operujące na przedziałach czasowych, nazywanych także interwałami czasowymi, pozwalające na uzyskanie z przedziału czasowego określonego elementu. Z kolei funkcje konwersji pozwalają na przekształcenie danych jednego typu na dane innego typu (np. funkcja przekształcająca ciąg znaków na liczbę). Zostaną również przedstawione dodatkowe konstrukcje wykorzystywane w zapytaniach.

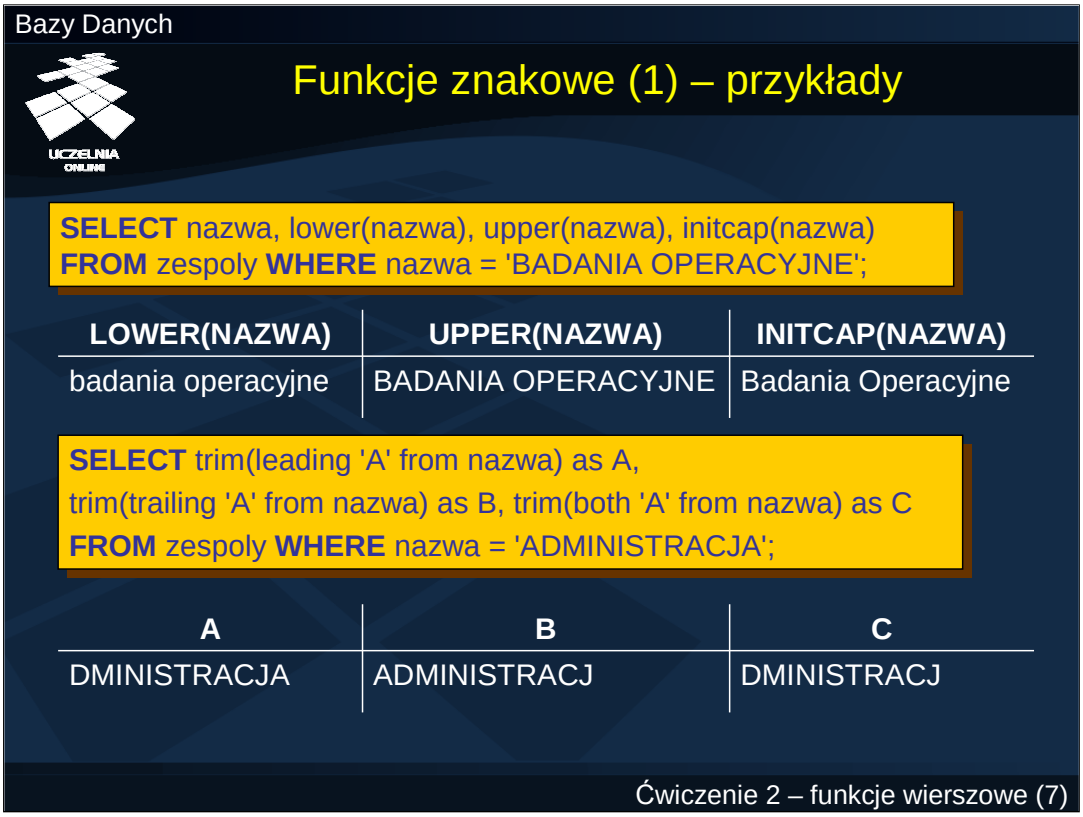


Funkcja wierszowa może zostać użyta we wszystkich klauzulach zapytania SQL. Omówimy to na zaprezentowanym przykładowym zapytaniu, w którym zastosowano trzy funkcje. Funkcja o nazwie funkcja_A, umieszczona w klauzuli SELECT, posiada dwa parametry o nazwach wyrażenie_1 oraz wyrażenie_2 i wylicza wartość, która zostanie zaprezentowana w wyniku zapytania jako dodatkowy atrybut o nazwie zdefiniowanej przez alias „wynik”. Jednoparametrowa (wyrażenie_3) funkcja o nazwie funkcja_B została użyta w warunku selekcji w klauzuli WHERE do filtrowania rekordów, odczytywanych przez zapytanie. Wynik funkcji zostanie porównany z wyrażeniem_4 przez użycia operatora. Wynik działania ostatniej, bezparametrowej funkcji o nazwie funkcja_C, zostaje użyty do posortowania rekordów w zbiorze wynikowym zapytania. Zauważmy, że w wywołaniu funkcji bezparametrowej opuszczamy nawiasy.

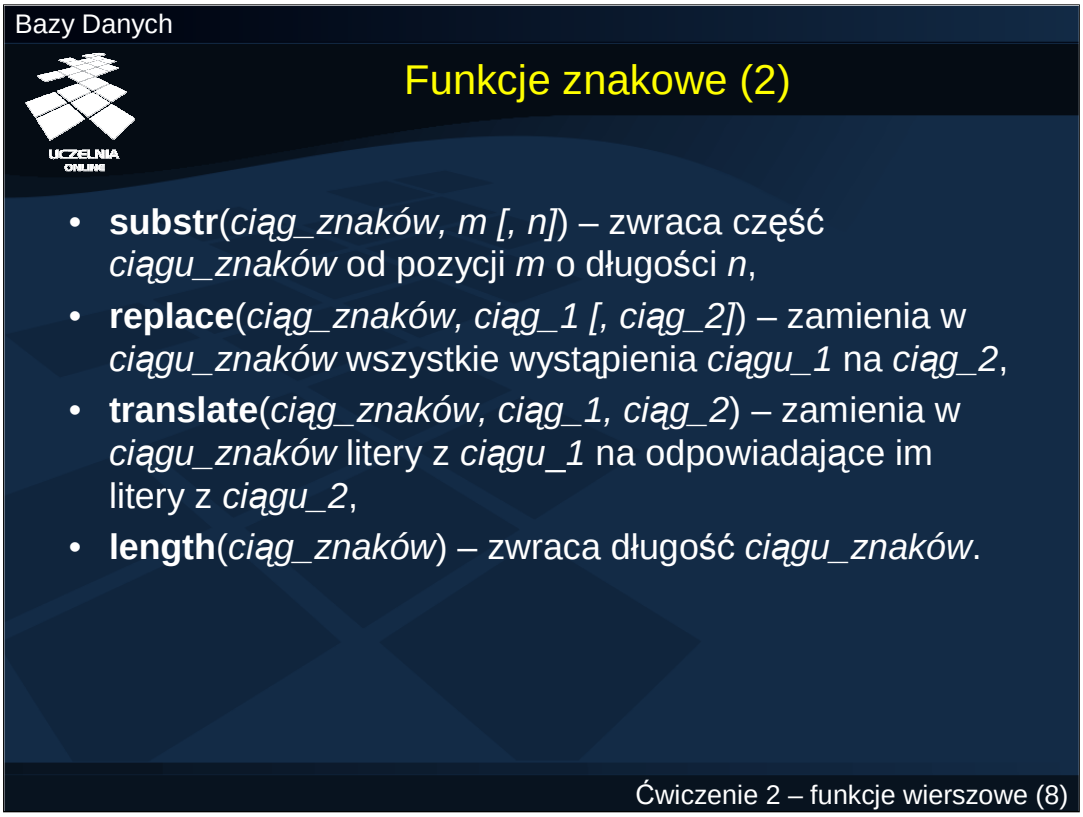


Przejdziemy teraz do omówienia predefiniowanych funkcji znakowych. Pierwsza grupa funkcji pozwala na zamianę wielkości liter w ciągu znaków. Funkcja lower(ciąg_znaków) zamienia w podanym jako parametr ciągu znaków wszystkie litery na małe, funkcja upper(ciąg_znaków) dokonuje zamiany w ciągu znaków wszystkich liter na wielkie, z kolei funkcja initcap(ciąg_znaków) przekształca ciąg znaków w ten sposób, że wszystkie litery zostają zamienione na małe, z wyjątkiem pierwszych liter wszystkich słów w ciągu, które zostają przekształcone na wielkie litery.

Kolejna funkcja, trim(leading | trailing | both znak from ciąg_znaków), pozwala na usunięcie z początku lub końca ciągu znaków wszystkich wystąpień wskazanego znaku. Słowo leading wskazuje, że mają zostać usunięte wszystkie wystąpienia znaku na początku ciągu znaków, słowo trailing – wszystkie wystąpienia z końca. Słowo both określa, że mają zostać usunięte wystąpienia znaku zarówno na początku, jak i na końcu ciągu znaków. Dopuszczalne jest wywołanie funkcji bez określenia miejsca usuwania znaków, wówczas domyślnie przyjmowane jest słowo both. Z kolei pominięcie w wywołaniu funkcji określenia znaku (parametr znak) powoduje usuniecie spacji.

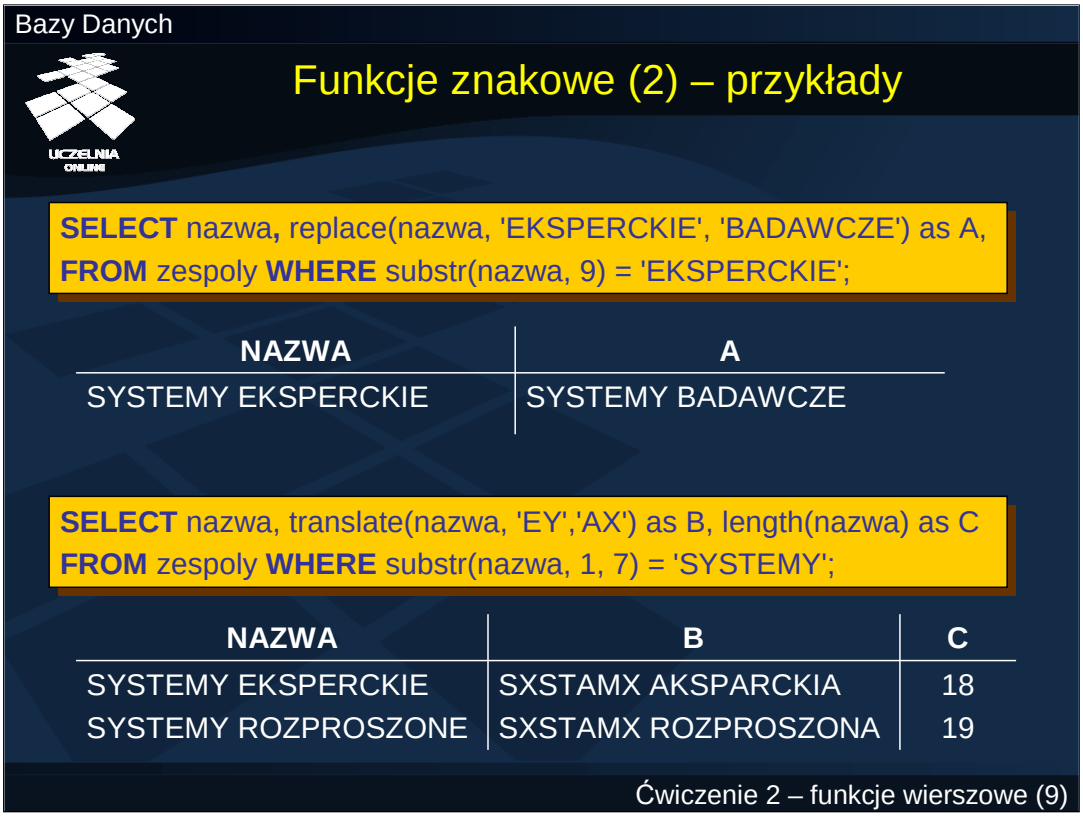


Zaprezentowane przykłady prezentują zastosowanie zdefiniowanych na poprzednim slajdzie funkcji.

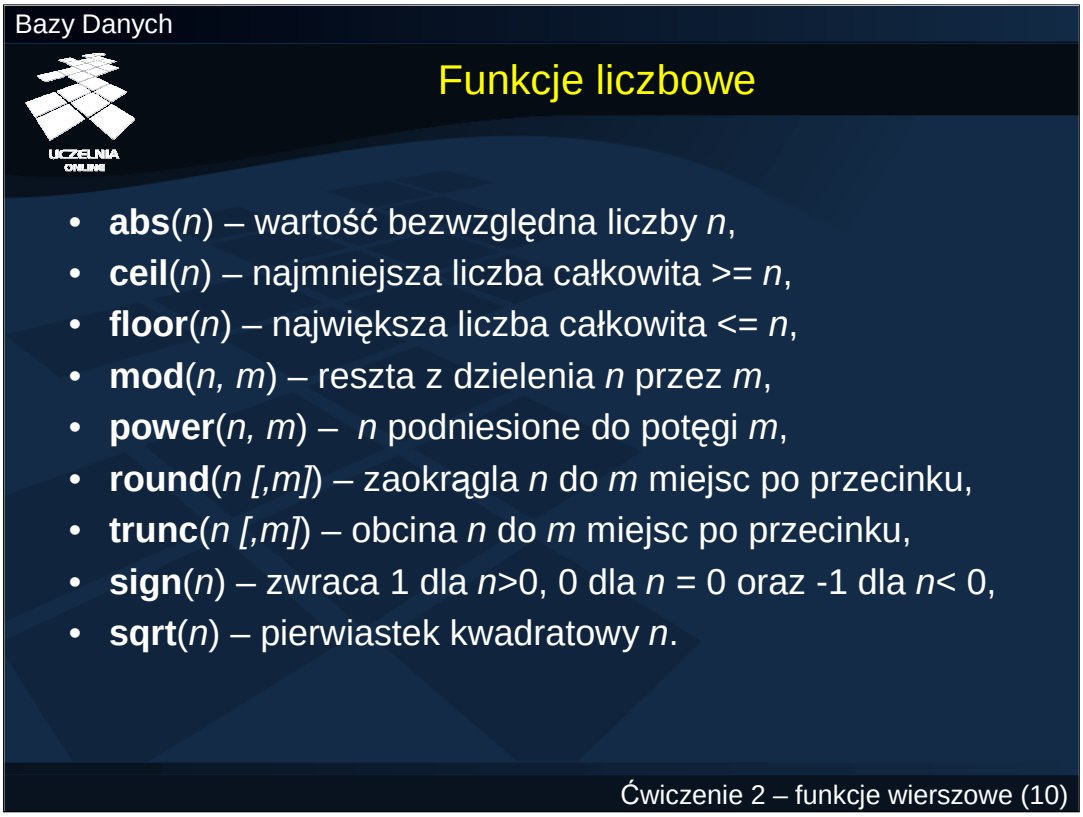


Kolejna grupa funkcji znakowych pozwala na wycinanie i zastępowanie poszczególnych podciągów w ciągu znaków. Funkcja substr posiada trzy parametry: ciąg_znaków, m i opcjonalny parametr n. Funkcja zwraca podciąg ciągu_znaków, rozpoczynający się od pozycji m i mający długość n (w przypadku opuszczenia parametru n podciąg zawiera wszystkie pozycje wyjściowego ciągu_znaków począwszy od pozycji m do końca). Funkcja replace pozwala na podmianę w ciągu znaków, będącym pierwszym parametrem funkcji (ciąg_znaków), wszystkich wystąpień podciągu, przekazanego jako drugi parametr (ciąg_1) na podciąg przekazany jako trzeci parametr (ciąg_2). W przypadku pominięcia w wywołaniu funkcji trzeciego parametru, podciąg ciąg_1 jest usuwany z wyjściowego ciągu znaków. Kolejna funkcja o nazwie translate również służy podmianie elementów wyjściowego ciągu znaków (parametr ciąg_znaków), jednak tym razem podmiana nie zachodzi dla podciągów, ale dla pojedynczych znaków w ciągu wyjściowym. Znaki, które mają zostać zamienione, zostają podane w ciągu przekazanym jako drugi parametr funkcji (ciąg_1), natomiast znaki, które mają je zastąpić, przekazuje trzeci parametr (ciąg_2). N-ty znak umieszczony w ciąg_1 zostaje zastąpiony w ciągu_znaków przez n-ty znak umieszczony w ciąg_2.

Ostatnia omawiania funkcja znakowa, funkcja length, dla ciągu znaków przekazanego jako parametr wylicza jego długość.

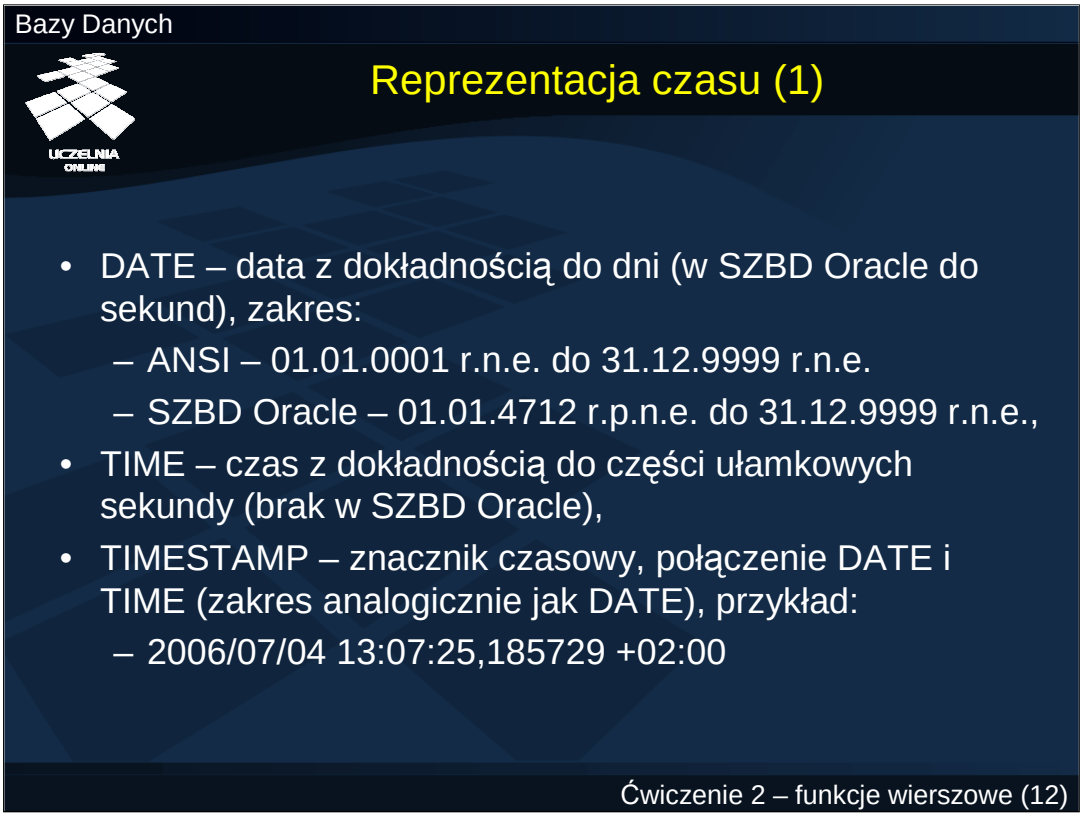


Bieżący slajd przedstawia przykłady zastosowań omawianych na poprzednim slajdzie funkcji znakowych.

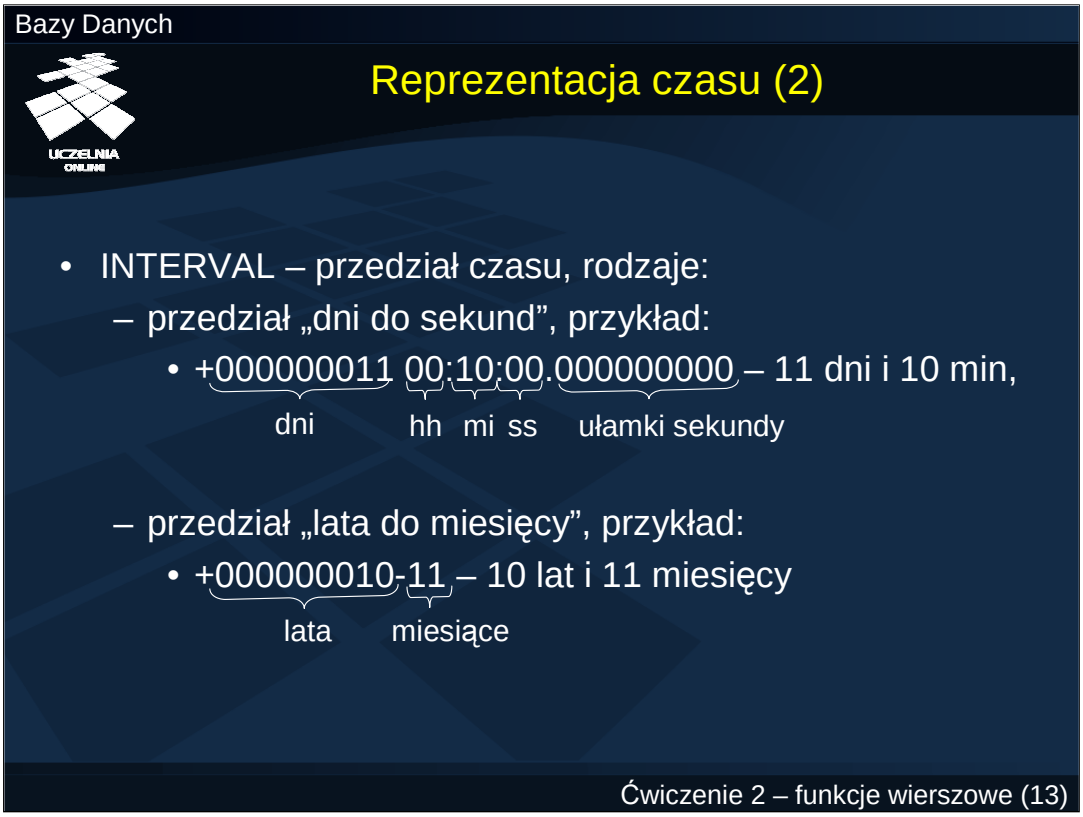


Przedstawiona teraz zostanie kolejna grupa funkcji, tym razem operujących na liczbach. Funkcja abs zwraca wartość bezwzględną liczby, przekazanej jako parametr. Funkcja ceil wylicza najmniejszą liczbę całkowitą większą bądź równą od liczby, będącej parametrem funkcji. Z kolei funkcja floor zwraca największą liczbę całkowitą mniejszą lub równą od liczby, przekazanej jako parametr funkcji. Za pomocą funkcji mod wyliczymy resztę z dzielenia liczby n (pierwszy parametr) przez liczbę m (drugi parametr). Funkcja power umożliwia podniesienie do m-tej potęgi (drugi parametr) liczby n (pierwszy parametr). Funkcja round służy do zaokrąglania wg reguł matematycznych liczby przekazanej jako pierwszy parametr. Liczbę pozycji ułamkowych zaokrąglanej liczby określa drugi, opcjonalny parametr funkcji. Pominięcie tego parametru powoduje zaokrąglenie n do liczby całkowitej. Analogicznie działa funkcji trunc, z tym że nie zaokrągla liczby, ale obcina do żądanej ilości pozycji ułamkowych. Kolejna funkcja, funkcja sign, zwraca wartość –1 jeśli liczba, przekazana jako parametr, jest liczbą ujemną. W przypadku liczby dodatniej funkcja zwraca wartość 1, natomiast dla liczby równej 0 zwraca wartość 0. Ostatnia z omawianych funkcji matematycznych, funkcja sqrt, pozwala wyliczyć pierwiastek kwadratowy liczby przekazanej jako parametr.

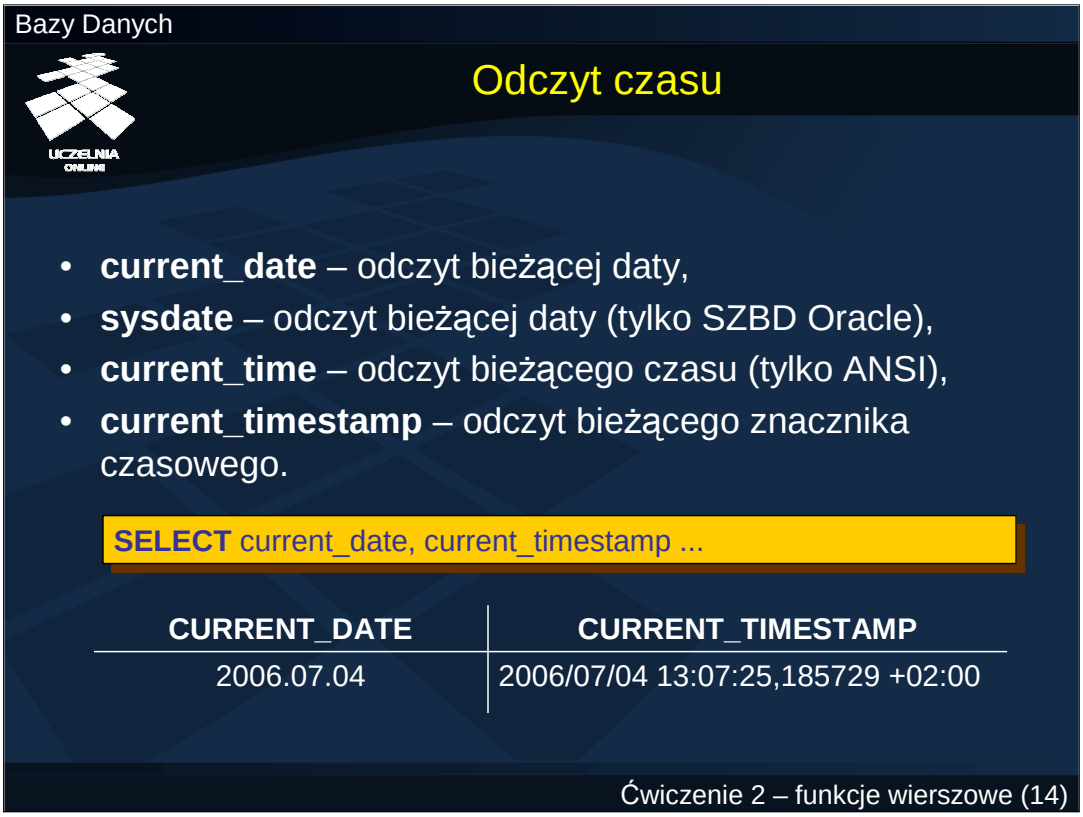
Przykłady omówionych wcześniej funkcji liczbowych przedstawia niniejszy slajd.



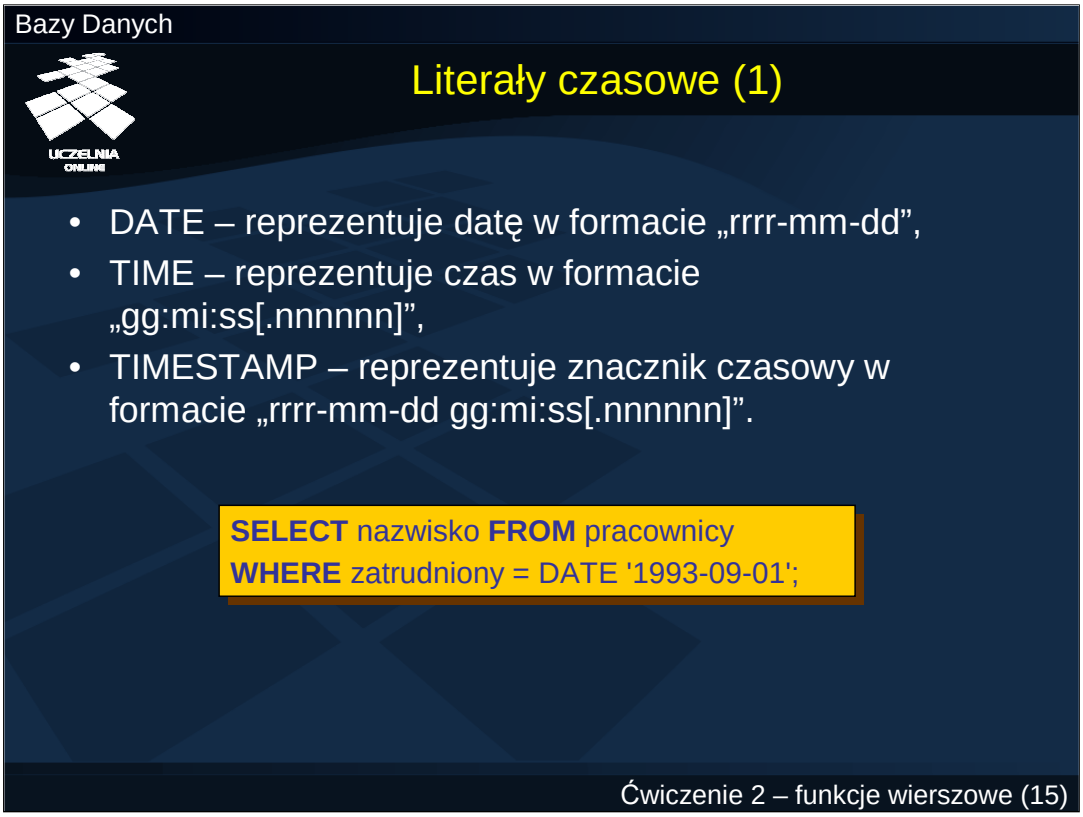
Reprezentacja czasu w systemie zarządzania bazą danych jest złożonym problemem. Wielu producentów dostarcza różne typy danych, służących do przechowywania elementów czasowych. Podstawowym typem danych jest typ DATE. W standardzie SQL-99 typ ten umożliwia przechowywanie dat z zakresu 1 stycznia 1 r.n.e. do 31 grudnia 9999 r.n.e. Z kolei w SZBD Oracle typ DATE przechowuje nie tylko datę, ale również określenie momentu czasowego z dokładnością do pełnych sekund. Inny jest również zakres dopuszczalnych wartości typu DATE – w SZBD Oracle zakres ten rozpoczyna się od 1 stycznia 4712 r.p.n.e. i trwa do 31 grudnia 9999 r.n.e. Standard SQL-99 definiuje typ TIME, służący do przechowywania czasu z dokładnością do części ułamkowych sekundy. Brak implementacji tego typu w SZBD Oracle. Kolejny typ danych, TIMESTAMP, służy do przechowywania tzw. znaczników czasowych. Znacznik czasowy przechowuje dokładne określenie momentu w czasie, a więc datę i czas z dokładnością do ułamkowych części sekundy, dodatkowo dla czasu składowane jest również przesunięcie strefy czasowej.



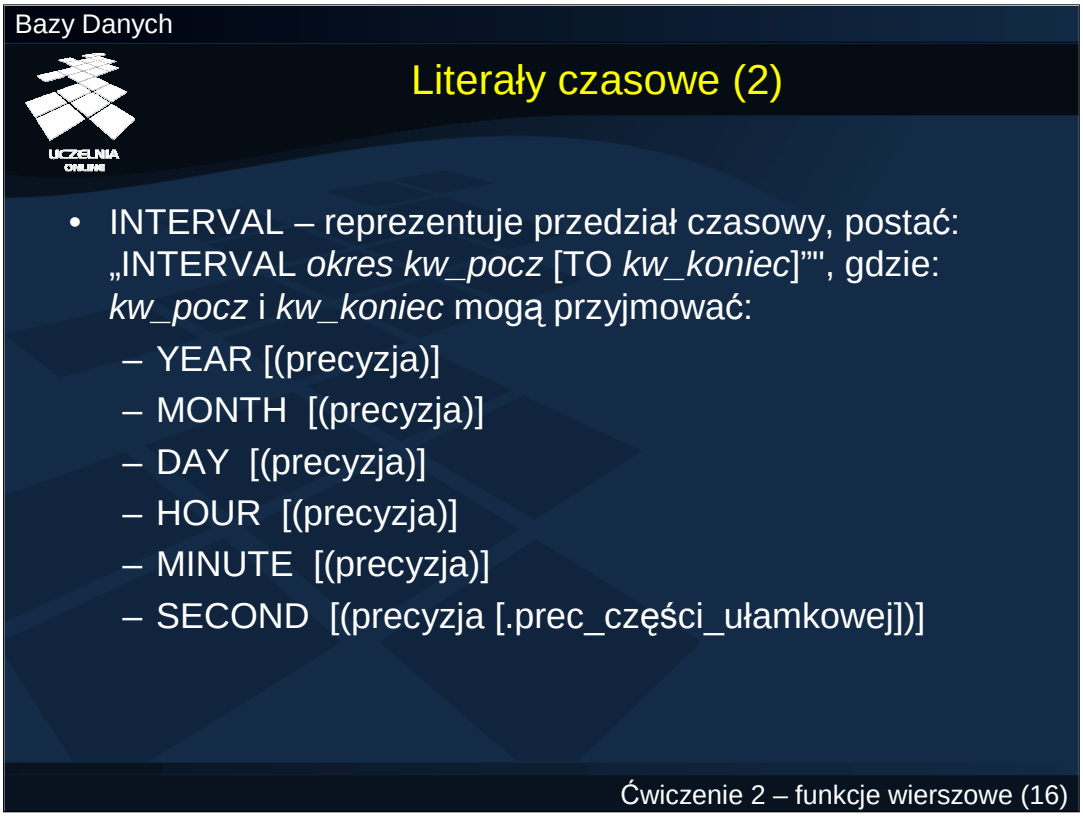
Kolejny typ danych, INTERVAL, pozwala na przechowywanie w bazie danych przedziałów czasowych (okresów). Możliwa jest definicja dwóch rodzajów przedziałów czasowych. Pierwszy z nich, przedział „dni do sekund”, pozwala na składowanie przedziału wyrażonego w dniach, godzinach, minutach, sekundach i częściach ułamkowych sekund. Należy zwrócić uwagę na predefiniowane separatory poszczególnych pozycji w przedziale tego rodzaju (patrz slajd). Drugi rodzaj przedziału czasowego, przedział „lata do miesięcy”, składuje okres wyrażony w latach i miesiącach. Tutaj domyślnym separatorem pozycji jest znak „-” (myślnik). Przedział czasowy nie musi wykorzystywać wszystkich elementów – np. przedział „dni do sekund” może korzystać jedynie z godzin i minut, jeśli przechowywany okres nie jest długi (nie zawiera dni) i nie wymaga większej precyzji (sekund i części ułamkowych sekundy).



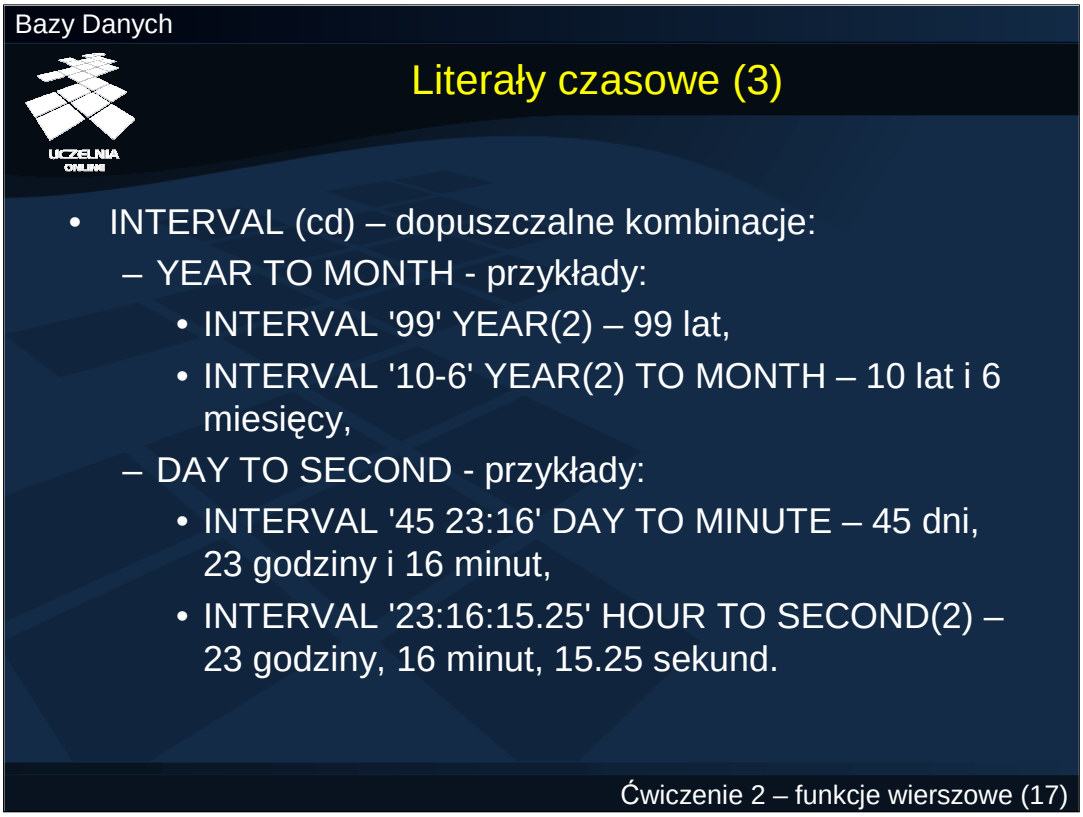
Bieżący slajd przedstawia zestaw predefiniowanych funkcji, pozwalających na odczyt bieżącego czasu z SZBD. Funkcja current_date zwraca bieżącą datę systemową (wartość typu DATE). W SZBD Oracle ten sam efekt można uzyskać stosując również funkcję sysdate (pamiętajmy, że DATE w SZBD Oracle zawiera również czas). Funkcja current_time odczytuje bieżący czas systemowy. Ta funkcja nie jest zaimplementowana w SZBD Oracle. Funkcja current_timestamp pozwala na odczytanie wartości bieżącego systemowego znacznika czasowego.



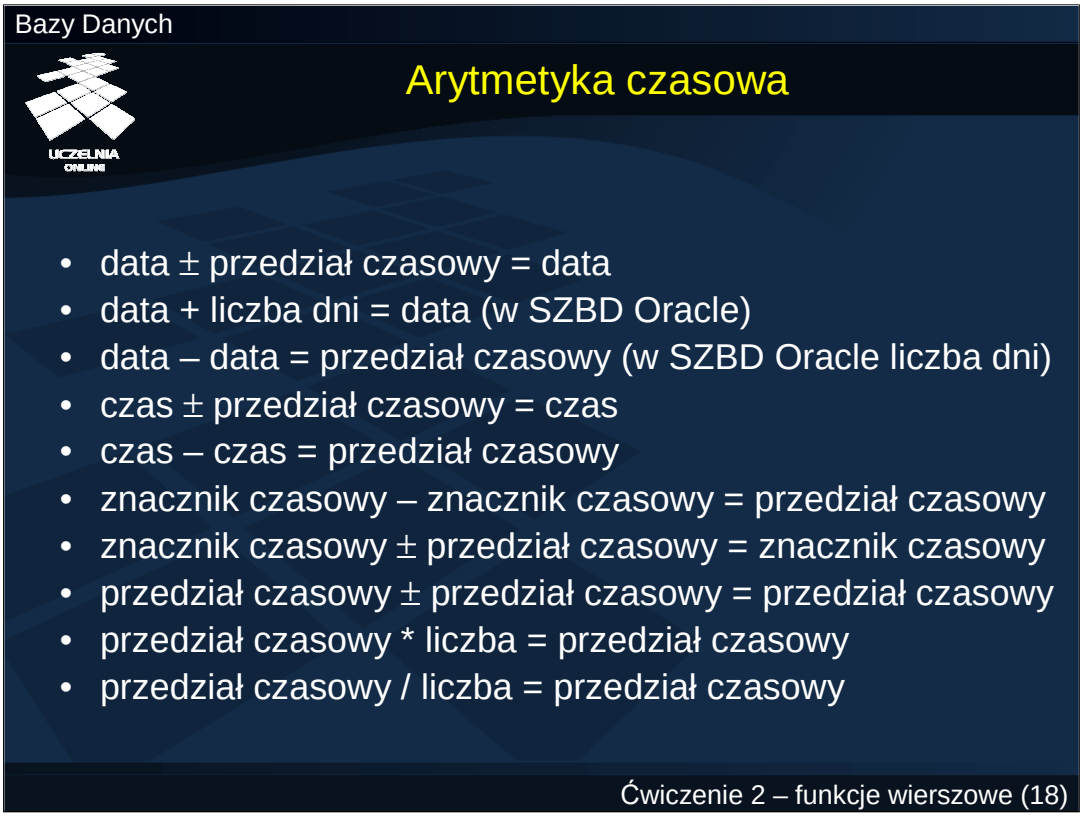
Elementy czasowe w systemie bazy danych składowane są w różnych formatach w zależności od rozwiązań zastosowanych przez producenta. Z kolei narzędzia do definiowania i wykonywania zapytań do bazy danych mogą stosować różne formaty prezentacji elementów czasowych (np. data w formacie dd.mm.rrrr albo w formacie rr-nazwa_miesiąca-dd). Powstaje problem – jak skonstruować zapytanie, które będzie niezależne od stosowanego przez narzędzie formatu prezentacji elementów czasowych. Rozwiązaniem jest zastosowanie w zapytaniu odpowiednich literałów czasowych. Literał DATE służy do wskazania, że ciąg znaków, wymieniony bezpośrednio za słowem DATE, prezentuje datę w formacie rrrr-mm-dd. Z kolei literał TIME pozwala na zdefiniowanie momentu czasowego z dokładnością do ułamkowych części sekundy. Stosowany format to gg:mi:ss[.nnnnnn] (części ułamkowe sekund są opcjonalne). Do definicji znacznika czasowego służy literał TIMESTAMP, tutaj format to rrrr-mm-dd gg:mi:ss[.nnnnnn]. Prezentowane na slajdzie zapytanie pozwala na wyszukanie w zbiorze pracowników osób, zatrudnionych 1 września 1993 r. Zapytanie to, dzięki zastosowaniu literału DATE, jest niewrażliwe na stosowany przez narzędzie domyślny format daty.



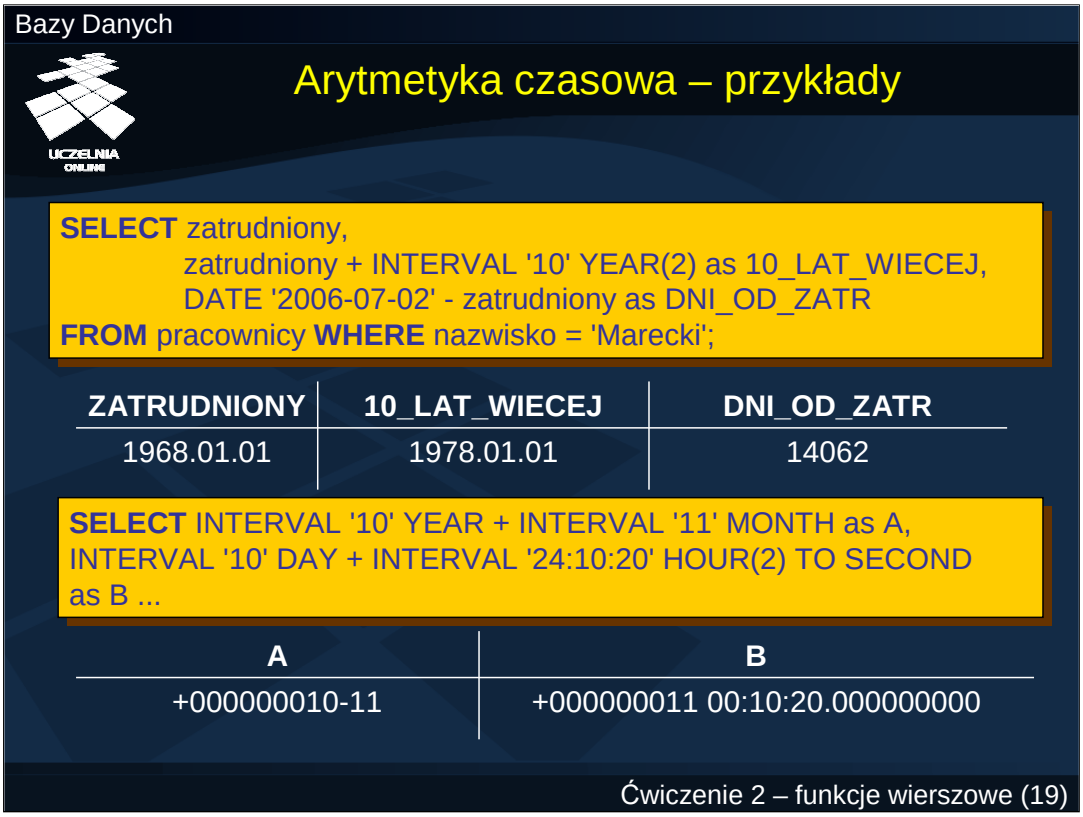
Kolejny literał, INTERVAL, służy do definicji przedziału czasowego. Za słowem INTERVAL należy podać kwalifikator początkowy przedziału (największy element reprezentowany w przedziale), po słowie TO kwalifikator końcowy (najmniejszy element reprezentowany w przedziale), przy czym kwalifikator końcowy jest opcjonalny. Oba kwalifikatory mogą przyjmować wartości z następującego zbioru: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, dla każdego kwalifikatora można określić precyzję wartości, dodatkowo dla kwalifikatora SECOND istnieje możliwość określenia precyzji części ułamkowej sekundy.



Dopuszczalne kombinacje kwalifikatorów zależą od rodzaju przedziału czasowego, który ma zostać utworzony. I tak dla przedziału „lata do miesięcy” można użyć kwalifikatorów YEAR i MONTH, z kolei dla przedziału „dni do sekund” używa się kwalifikatorów DAY, HOUR, MINUTE i SECOND. Przykładowe przedziały czasowe, zdefiniowane z użyciem literału INTERVAL, przedstawiono na bieżącym slajdzie.

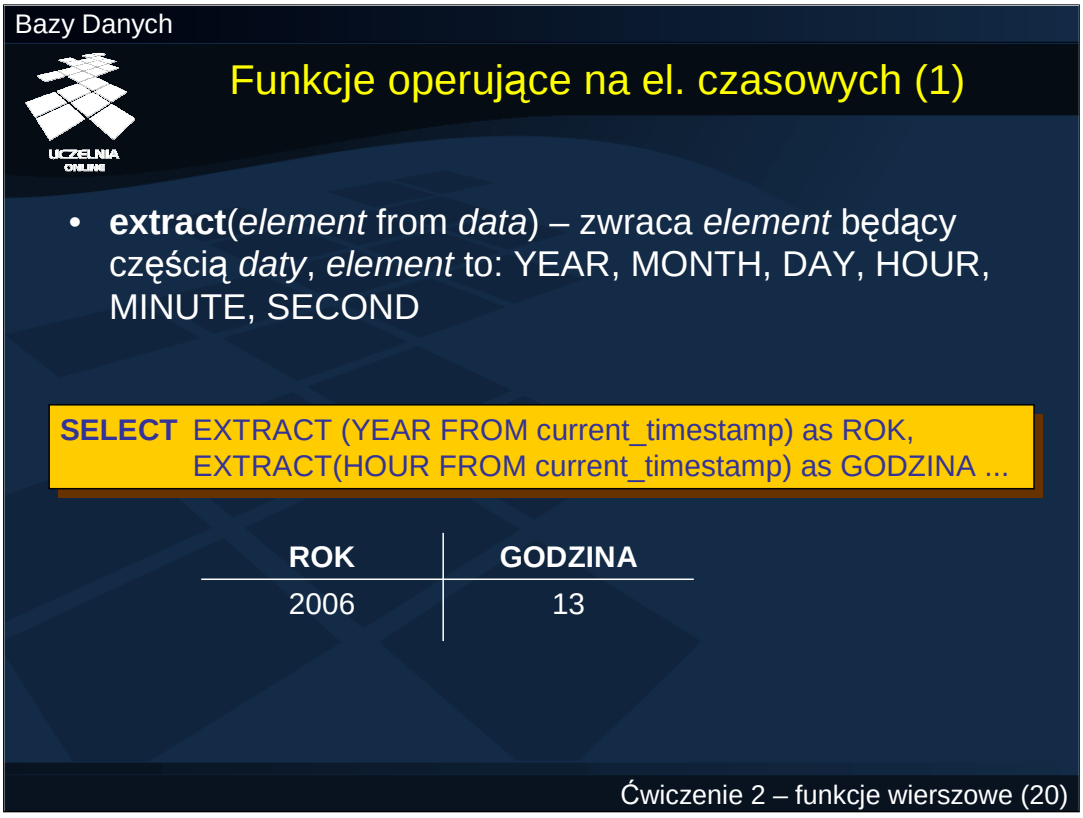


Kolejnym zagadnieniem są wyniki operacji arytmetycznych, realizowanych na elementach określających czas. Jeśli do daty dodamy przedział czasowy lub od daty odejmiemy przedział czasowy, otrzymamy w wyniku datę. W SZBD Oracle do daty możemy dodać liczbę, tutaj interpretowaną jako liczbę dni, w wyniku takiej operacji otrzymamy nową datę. Różnica dwóch dat daje w wyniku przedział czasowy, określający czas, jaki upłynął między datami (w SZBD Oracle wynikiem różnicy dat jest liczba dni pomiędzy datami). Jeśli do elementu reprezentującego czas dodamy przedział czasowy lub odejmiemy od niego przedział czasowy, otrzymamy w wyniku element reprezentujący czas. Różnica dwóch elementów reprezentujących czas da nam w wyniku przedział czasowy. Różnica dwóch znaczników czasowych daje w rezultacie przedział czasowy. Z kolei jeśli do znacznika czasowego dodamy przedział czasowy lub odejmiemy od niego przedział czasowy, otrzymamy znacznik czasowy. Suma lub różnica dwóch przedziałów czasowych da nam w wyniku nowy przedział czasowy. Jeśli przedział czasowy pomnożymy lub podzielimy przez liczbę, w wyniku otrzymamy również przedział czasowy.

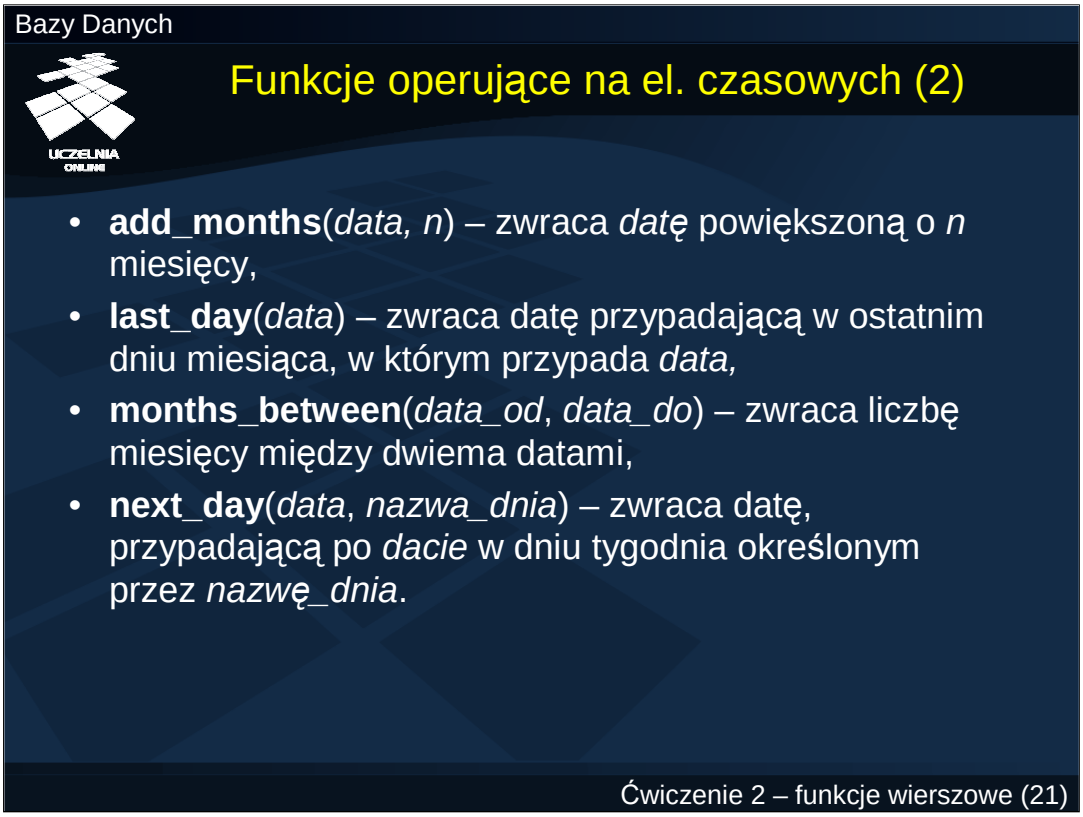


W pierwszym przykładzie do daty zatrudnienia pracownika Mareckiego dodajemy przedział czasowy o długości 10 lat, w wyniku otrzymujemy nową datę, przypadającą 10 lat po zatrudnieniu pracownika. W tym samym przykładzie od daty 2 lipca 2006 r., zdefiniowanej za pomocą literału DATE, odjęto datę zatrudnienia pracownika, w wyniku otrzymano liczbę dni, jaka upłynęła od zatrudnienia pracownika o nazwisku Marecki, do dnia 2 lipca 2006 r. (zapytanie wykonano w SZBD Oracle).

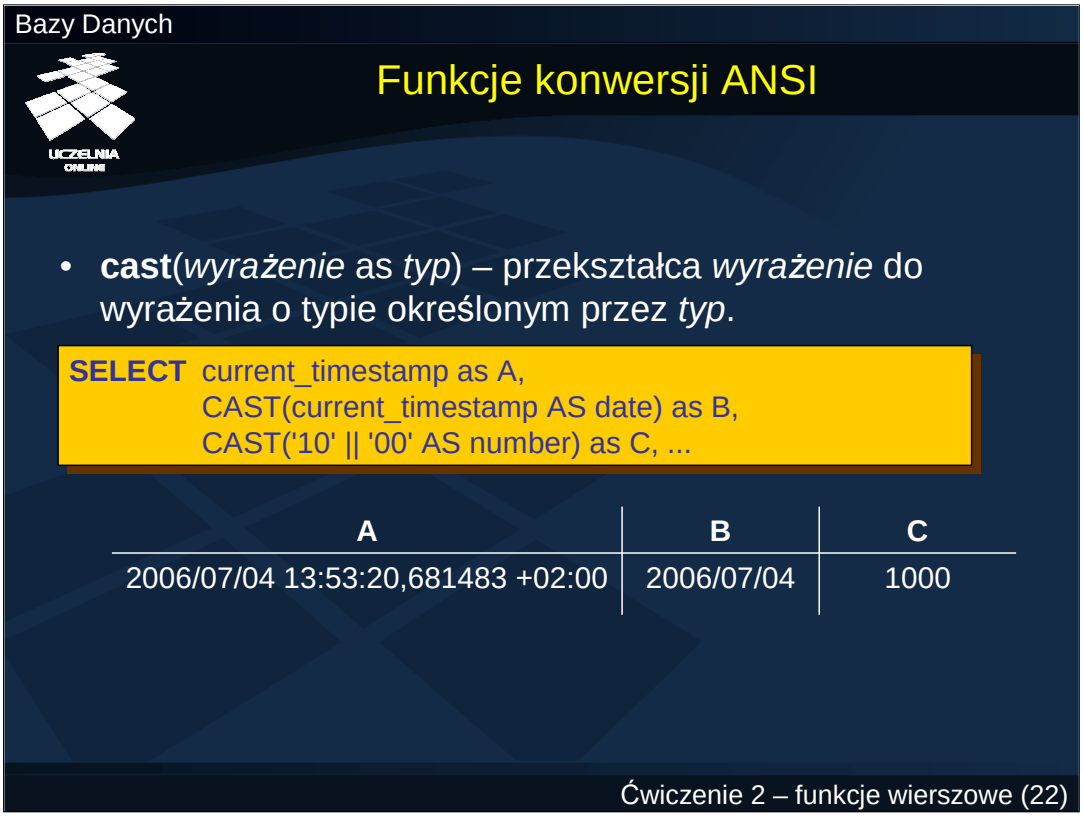
W drugim przykładzie dodajemy do siebie dwa przedziały czasowe: do 10 lat dodajemy 11 miesięcy, w wyniku otrzymujemy nowy przedział czasowy, 10 lat i 11 miesięcy. Drugie wyrażenie do przedziału czasowego 10 dni dodano przedział 24 godziny, 10 minut o 20 sekund, w wyniku otrzymano przedział 11 dni, 10 minut i 20 sekund.



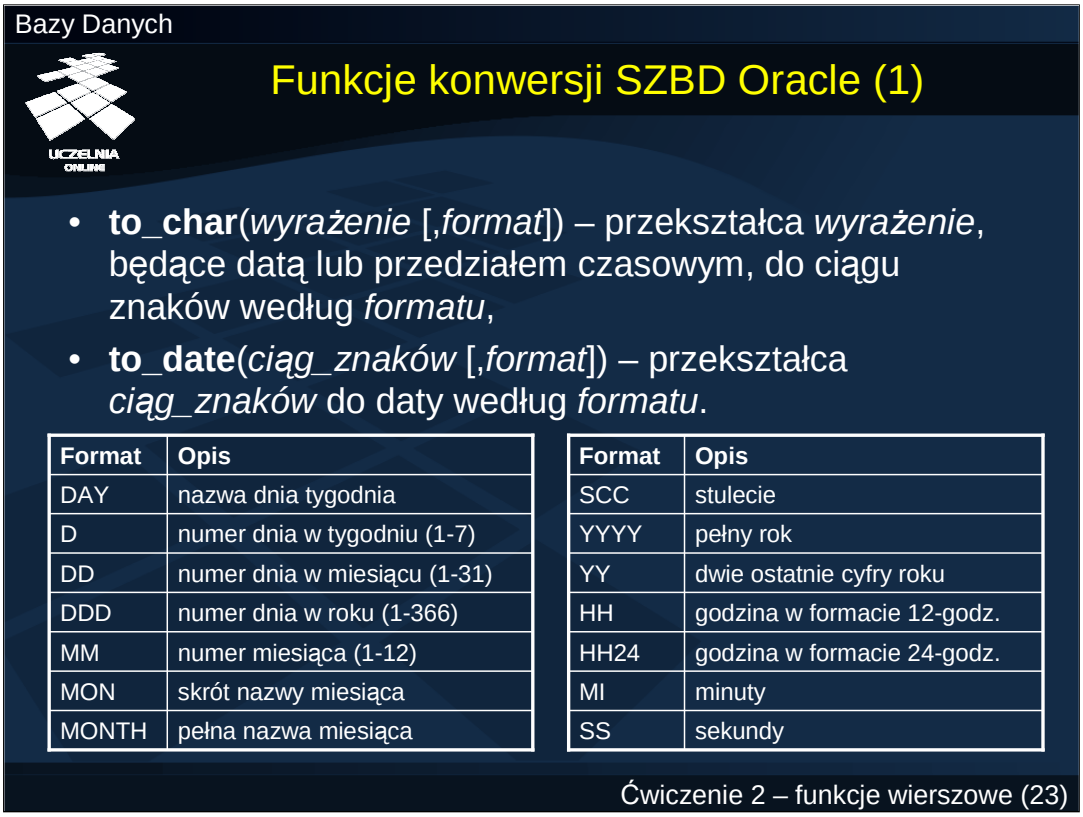
Rozpoczniemy teraz przegląd funkcji operujących na elementach czasowych. Pierwsza z nich, funkcja extract, umożliwia wydobycie z elementu czasowego określonego składnika: roku, miesiąca, dnia, godziny, minuty lub sekundy. W wywołaniu funkcji jako pierwszy podaje się element czasowy, po słowie from określa się wydobywany składnik. Słowa określające wydobywany składnik to odpowiednio: year, month, day, hour, minute i second. Wynikiem funkcji jest liczba, będąca wartością wydobywanego składnika. W przykładzie pierwsze wyrażenie wydobywa z bieżącego znacznika czasowego wartość roku, natomiast drugie wyrażenie odczytuje z bieżącego znacznika czasowego wartość godziny.



Bieżący slajd przedstawia przykłady innych funkcji, operujących na datach, a zaimplementowanych w SZBD Oracle. Funkcja add_months do daty, podanej jako pierwszy parametr, dodaje liczbę miesięcy, przekazaną jako drugi parametr. Wynikiem działania funkcji jest nowa data. Kolejna funkcja, last_day, wylicza datę, przypadającą w ostatnim dniu miesiąca, w którym znajduje się data przekazana do funkcji jako parametr. Funkcja months_between zwraca liczbę określającą, ile miesięcy upłynęło miedzy dwiema datami, przekazanymi jako parametry funkcji. Funkcja next_day wylicza datę, przypadającą po dacie, podanej jako pierwszy parametr, w dniu tygodnia, którego nazwę przekazano w postaci drugiego parametru.



Bieżący slajd rozpoczyna omawianie zestawu funkcji, dokonujących konwersji pomiędzy wartościami różnych typów. Pierwsza funkcja o nazwie cast, wchodząca w skład standardu SQL-99, umożliwia konwersję wartości pomiędzy zdefiniowanymi przez standard typami danych. Pierwszym parametrem funkcji jest wyrażenie wyliczające wartość, która ma być poddana konwersji, drugi parametr, podany po słowie „AS”, określa docelowy typ danych. W zaprezentowanym przykładzie zapytanie odczytuje wartość bieżącego systemowego znacznika czasowego, następnie realizowana jest konwersja wartości tego znacznika do wartości typu data. Trzecie wyrażenie pokazuje konwersję ciągu znaków, powstałego przez sklejenie operatorem konkatenacji dwóch innych ciągów znaków, do wartości liczbowej. Oczywiście wartość poddawana konwersji musi być poprawną wartością w docelowym typie danych (jak w przykładzie – ciąg znaków '1000' określa poprawną liczbę, tak więc konwersja zakończy się sukcesem).



Omówione teraz zostaną funkcje konwersji, zaimplementowane w SZBD Oracle. Najpierw przedstawimy funkcje, umożliwiające konwersję pomiędzy ciągami znaków a datami lub przedziałami czasowymi. Pierwsza z nich, funkcja to_char, realizuje konwersję wyrażenia, będącego datą lub przedziałem czasowym (pierwszy parametr), do ciągu tekstowego według formatu, przekazanego jako drugi parametr. W przypadku pominięcia w wywołaniu funkcji parametru określającego format, konwersja dokonywana jest zgodnie z domyślnym formatem dla danego systemu bazy danych. Druga funkcja, to_date, realizuje operację odwrotną – konwertuje ciąg znaków, podany jako pierwszy parametr, do daty. Określenie formatu daty, jaki przedstawia ciąg znaków, jest realizowane przez drugi parametr, format. Jeśli w wywołaniu funkcji format zostanie pominięty, ciąg znaków powinien przechowywać datę w domyślnym formacie danego systemu bazy danych.

Na slajdzie przedstawiono elementy, z których można skonstruować format, wykorzystywany w obu funkcjach. Należy dodać, że przy elementach, które w dacie określają nazwę dnia (DAY), trzyliterowy skrót nazwy miesiąca (MON) lub pełną nazwę miesiąca (MONTH), znaczenie ma wielkość liter, którymi te elementy zostaną zapisane. W przypadku zapisu wymienionych elementów wielkimi literami w dacie otrzymamy nazwę danego składnika podaną wielkimi literami (np. „PONIEDZIAŁEK”, „MARZEC”), natomiast przy zapisie elementów małymi literami, nazwy składników również będą zapisane małymi literami (a więc „poniedziałek”, „marzec”).

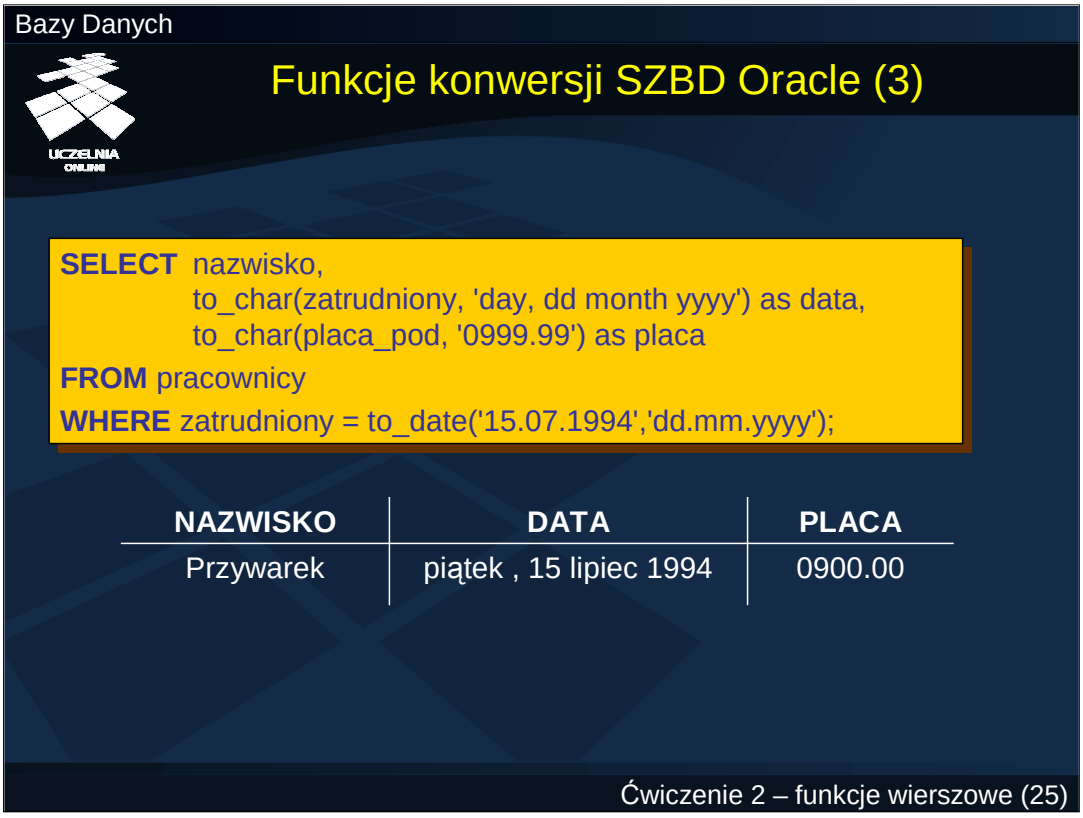
Funkcje konwersji SZBD Oracle (2)

- **to_char**(*liczba* [,*format*]) – przekształca *liczbę* do ciągu znaków według *formatu*,
- **to_number**(*ciąg_znaków* [,*format*]) – przekształca *ciąg_znaków* do liczby według *formatu*.

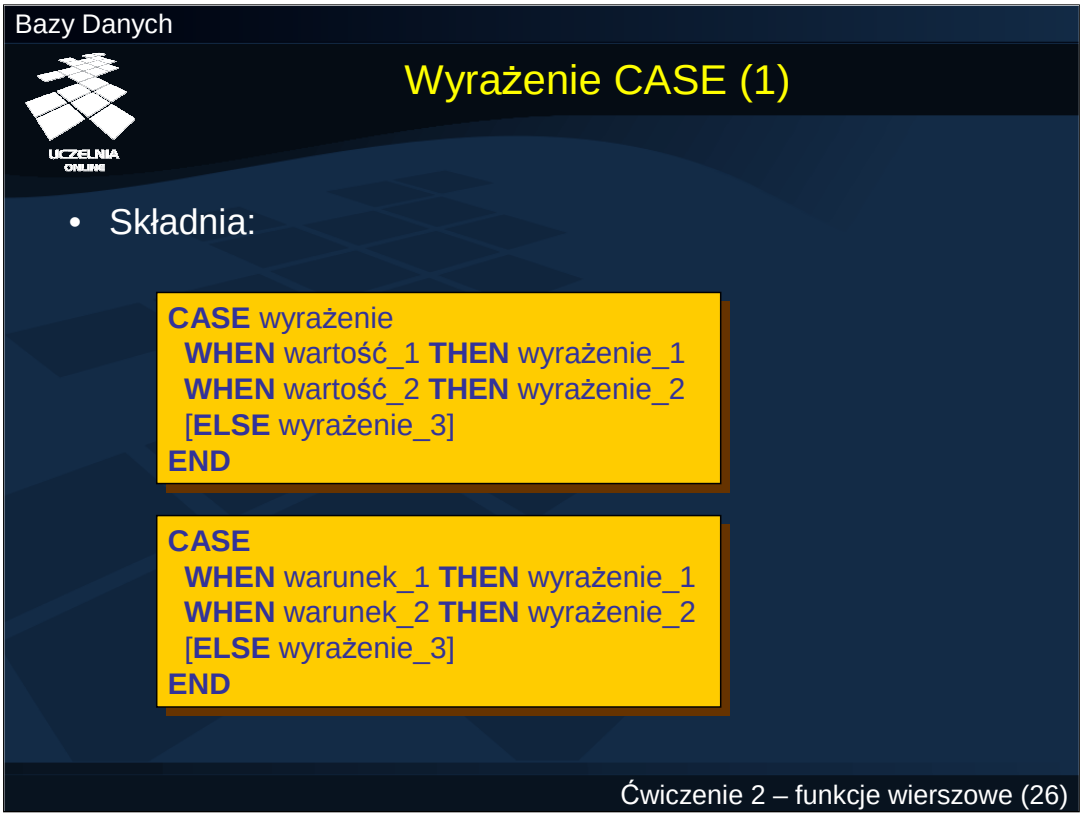
Format	Opis
.	kropka (oddziela część całkowitą od ułamkowej)
,	przecinek (oddziela elementy liczby, np. tysiące od milionów)
9	określa pozycję w liczbie, zera z lewej strony są pomijane
0	określa drukowanie zera z lewej lub prawej strony liczby

Funkcja to_char posiada odmianę, wykorzystywaną do konwersji liczby do ciągu znaków. W tym wypadku pierwszym parametrem jest konwertowana liczba, natomiast drugi element określa format konwersji. Format można pominąć, wówczas do konwersji zostanie wykorzystywany domyślny format wykorzystywany przez dany system bazy danych. Operację odwrotną realizuje funkcja to_number, które umożliwia konwersję liczby, zapisanej w postaci ciągu znaków, do właściwej liczby. I tutaj drugim, opcjonalnym parametrem funkcji, jest format konwersji.

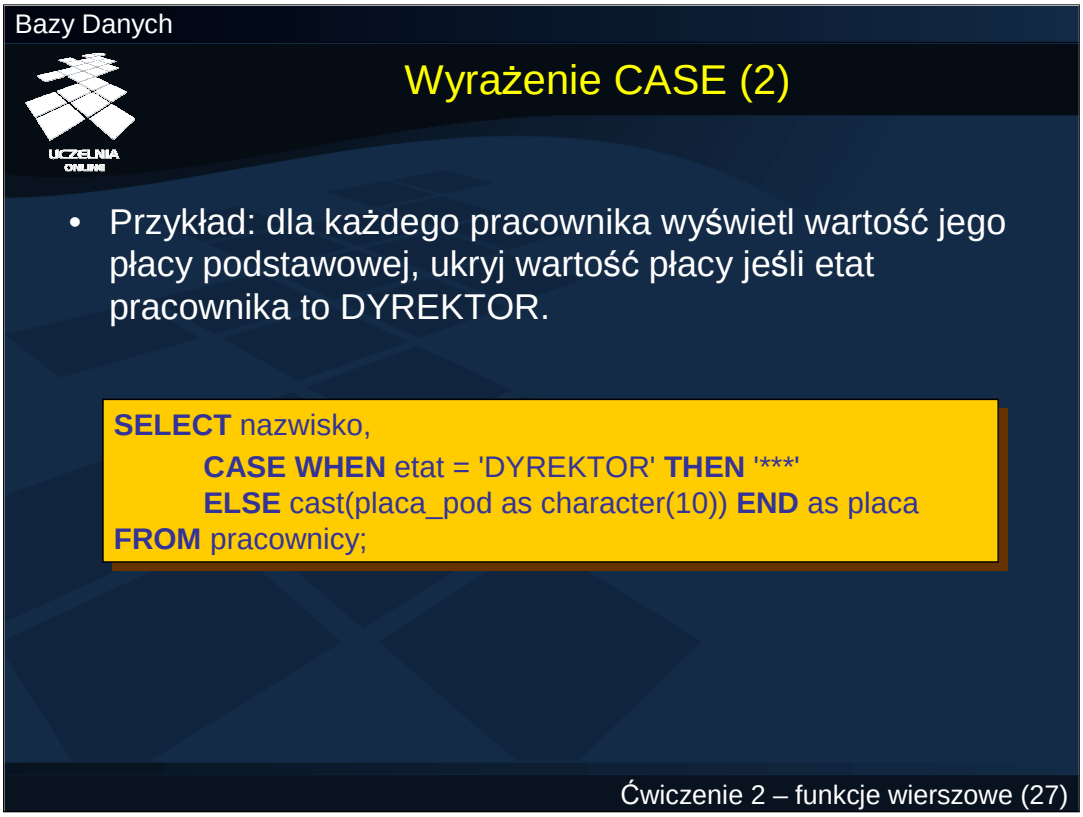
Na powyższym slajdzie przedstawiono wykorzystywane przy konstruowaniu formatu elementy. Element „9” określa pozycję liczby, element „0” dodaje się na początku bądź końcu formatu celem określenia, czy liczba ma być uzupełniana o zera (np. 900,5 ma być drukowane jako 0900,50). Dwa pozostałe elementy to separator części całkowitej od ułamkowej („.”) oraz separatory poszczególnych części liczby („,”), np. milionów od tysięcy, tysięcy od setek, itd.



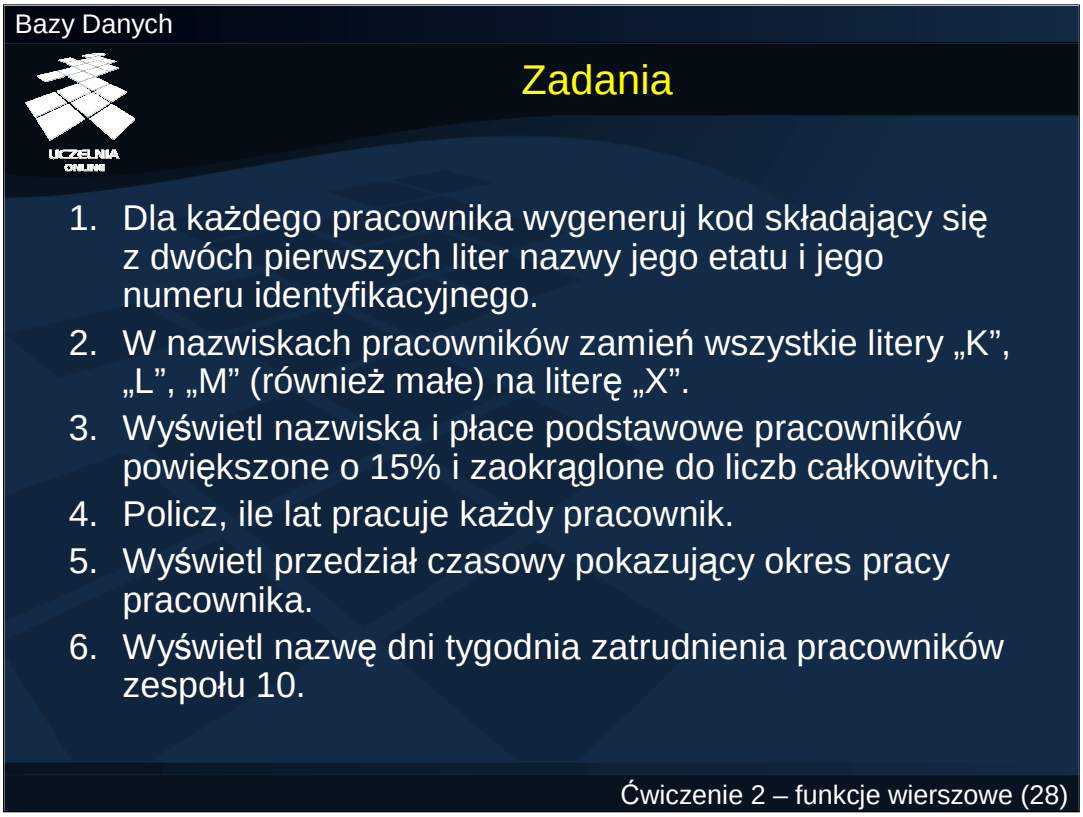
W przykładzie pokazano konwersję daty zatrudnienia pracownika do ciągu tekstowego w formacie „nazwa dnia, numer dnia w miesiącu nazwa miesiąca czterocyfrowy numer roku”. Drugie wyrażenie przekształca wartość płacy podstawowej pracownika do ciągu znaków, format wymusza dodanie przed zera przed liczbą i drukowanie dwóch pozycji ułamkowych. Wyrażenie w warunku zapytania konstruuje datę z ciągu tekstowego, data w ciągu zapisana jest w formacie „numer dnia w miesiącu.numer miesiąca w roku.czterocyfrowy numer roku”.



Wyrażenie CASE umożliwia zbudowanie konstrukcji, której wynik będzie uzależniony od wyniku wartościowania zdefiniowanego wyrażenia. Konstrukcji CASE można używać w dwóch postaciach. W pierwszej postaci po słowie kluczowym CASE umieszcza się wyrażenie (np. atrybut), natomiast spodziewane wartości tego wyrażenia umieszcza się w kolejnych sekcjach po słowie kluczowym WHEN (na slajdzie wartości te oznaczono jako wartość_1 i wartość_2). W trakcie wykonania zapytania wyrażenie zwraca pewną wartość, realizowane jest wówczas dopasowanie tej wartości do jednej z wartości w sekcjach WHEN (dopasowanie zachodzi tylko dla pierwszej pasującej wartości). Wynikiem całej konstrukcji CASE jest wynik wyrażenia umieszczonego po słowie THEN sekcji, dla której zaszło dopasowanie (na slajdzie wyrażenia te oznaczono przez wyrażenie_1 i wyrażenie_2). Opcjonalna klauzula ELSE pozwala na zdefiniowanie wyrażenia, którego wartość zostanie zwrócona jeśli nie zajdzie żadne dopasowanie. Dość poważnym ograniczeniem tej postaci konstrukcji CASE jest możliwość jedynie równościowego porównania wyrażenia po słowie CASE z wartościami w sekcjach WHEN. Ograniczenia tego nie ma druga postać konstrukcji CASE, w której w kolejnych sekcjach po słowie WHEN umieszcza się warunek logiczny. Wartością konstrukcji CASE będzie wartość wyrażenia umieszczonego po słowie THEN w tej sekcji, dla której warunek logiczny jest prawdziwy. Ta postać konstrukcji CASE jest bardziej elastyczna.



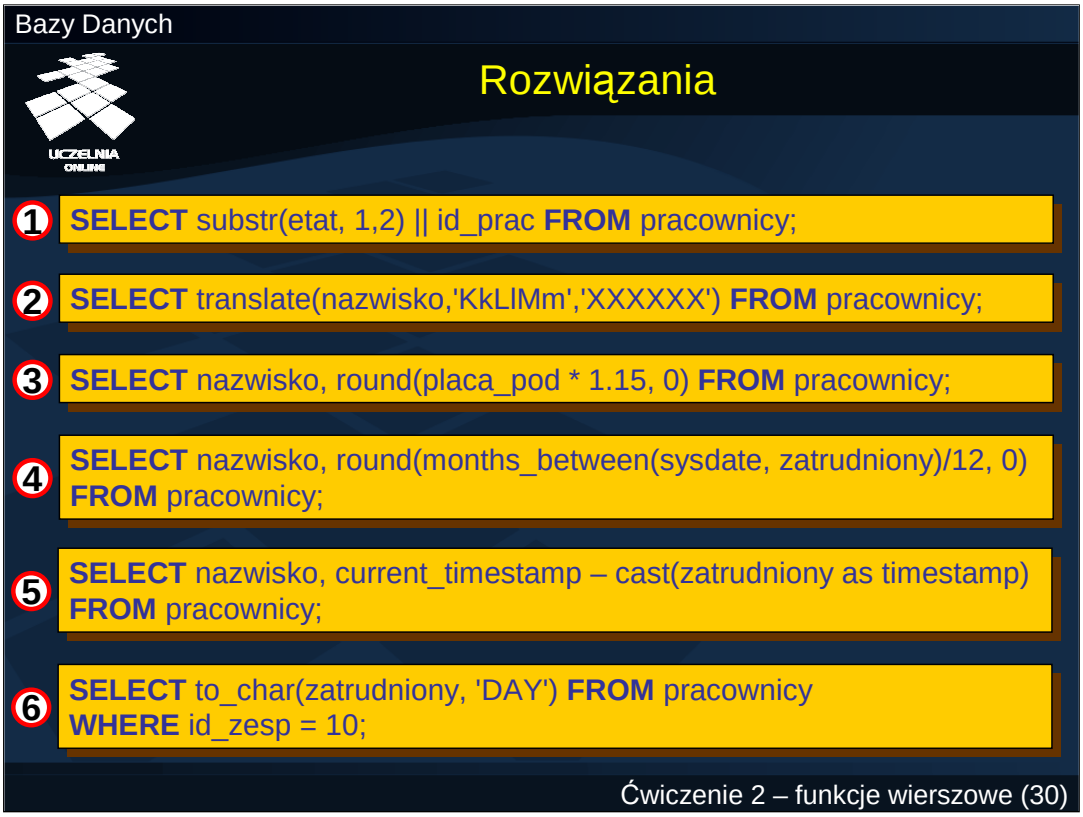
Zaprezentowany przykład pokazuje wykorzystanie CASE do ukrycia pensji pracowników na etacie DYREKTOR. SZBD za typ wartości zwracanej przez wyrażenie CASE w przykładzie przyjmuje ciąg znaków – jest to spowodowane umieszczeniem ciągu „***” w pierwszej sekcji THEN. Wszystkie pozostałe wartości w pozostałych sekcjach THEN i sekcji ELSE w konsekwencji też muszą być ciągami znaków. Stąd zastosowanie funkcji CAST do konwersji płacy podstawowej do ciągu znaków.



Bieżący slajd zawiera zestaw zadań, pozwalających na utrwalenie wiadomości z zastosowania funkcji wierszowych w zapytaniach SQL.

Zadania

7. Wyświetl informacje o wszystkich zespołach wraz z nazwami dzielnic, w których zlokalizowane są zespoły. Przyjmij, że Mielżyńskiego i Strzelecka należą do dzielnicy Stare Miasto, Piotrowo należy do dzielnicy Nowe Miasto a Wieniawskiego należy do dzielnicy Grunwald. Skorzystaj z wyrażenia CASE.
8. Dla każdego pracownika wyświetl jego nazwisko, płacę podstawową i informację o tym, czy jego pensja jest mniejsza, równa lub większa od 1850 złotych. Skorzystaj z wyrażenia CASE.



Bieżący slajd przedstawia rozwiązania zadań (1), (2), (3), (4), (5) i (6), których treść zacytowano poniżej.

(1) Dla każdego pracownika wygeneruj kod składający się z dwóch pierwszych liter nazwy jego etatu i jego numeru identyfikacyjnego.

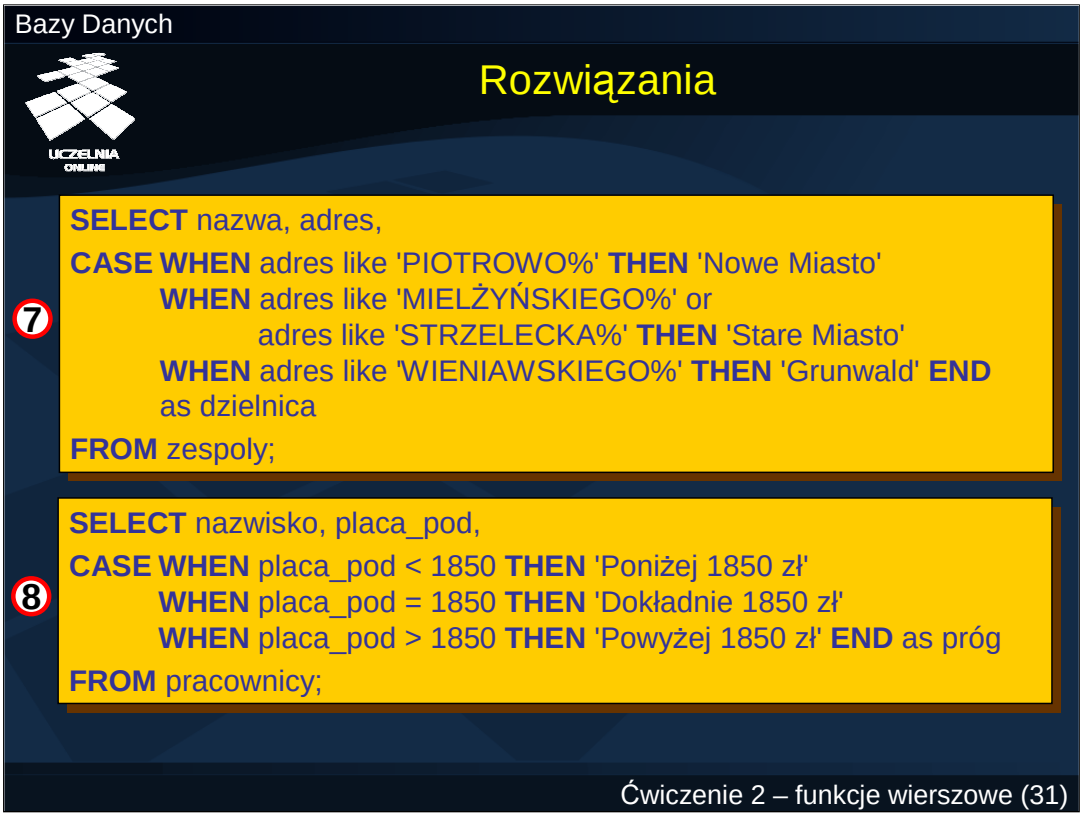
(2) W nazwiskach pracowników zamień wszystkie litery „K”, „L”, „M” (również małe) na literę „X”.

(3) Wyświetl nazwiska i płace podstawowe pracowników powiększone o 15% i zaokrąglone do liczb całkowitych.

(4) Policz, ile lat pracuje każdy pracownik.

(5) Wyświetl przedział czasowy pokazujący okres pracy pracownika.

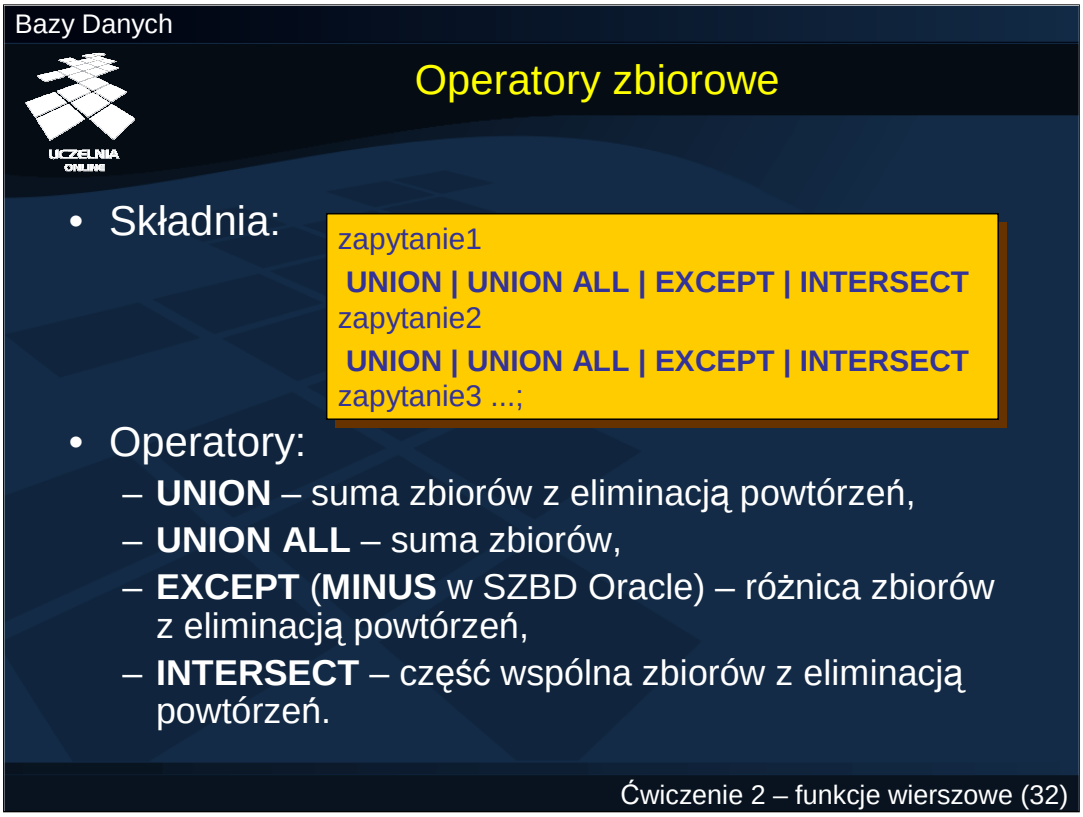
(6) Wyświetl nazwę dni tygodnia zatrudnienia pracowników zespołu 10.



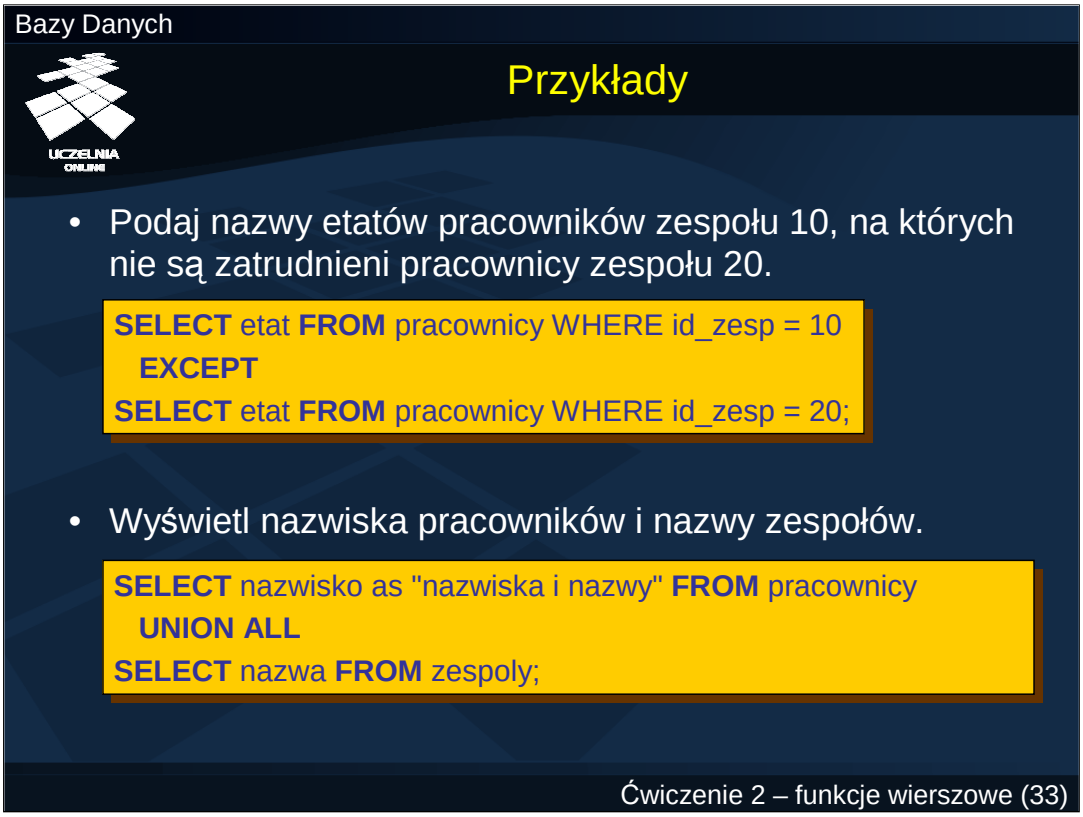
Bieżący slajd przedstawia rozwiązania zadań (7) i (8), których treść zacytowano poniżej.

(7) Wyświetl informacje o wszystkich zespołach wraz z nazwami dzielnic, w których zlokalizowane są zespoły. Przyjmij, że Mielżynskiego i Strzelecka należą do dzielnicy Stare Miasto, Piotrowo należy do dzielnicy Nowe Miasto a Wieniawskiego należy do dzielnicy Grunwald. Skorzystaj z wyrażenia CASE.

(8) Dla każdego pracownika wyświetl jego nazwisko, płacę podstawową i informację o tym, czy jego pensja jest mniejsza, równa lub większa od 1850 złotych. Skorzystaj z wyrażenia CASE.

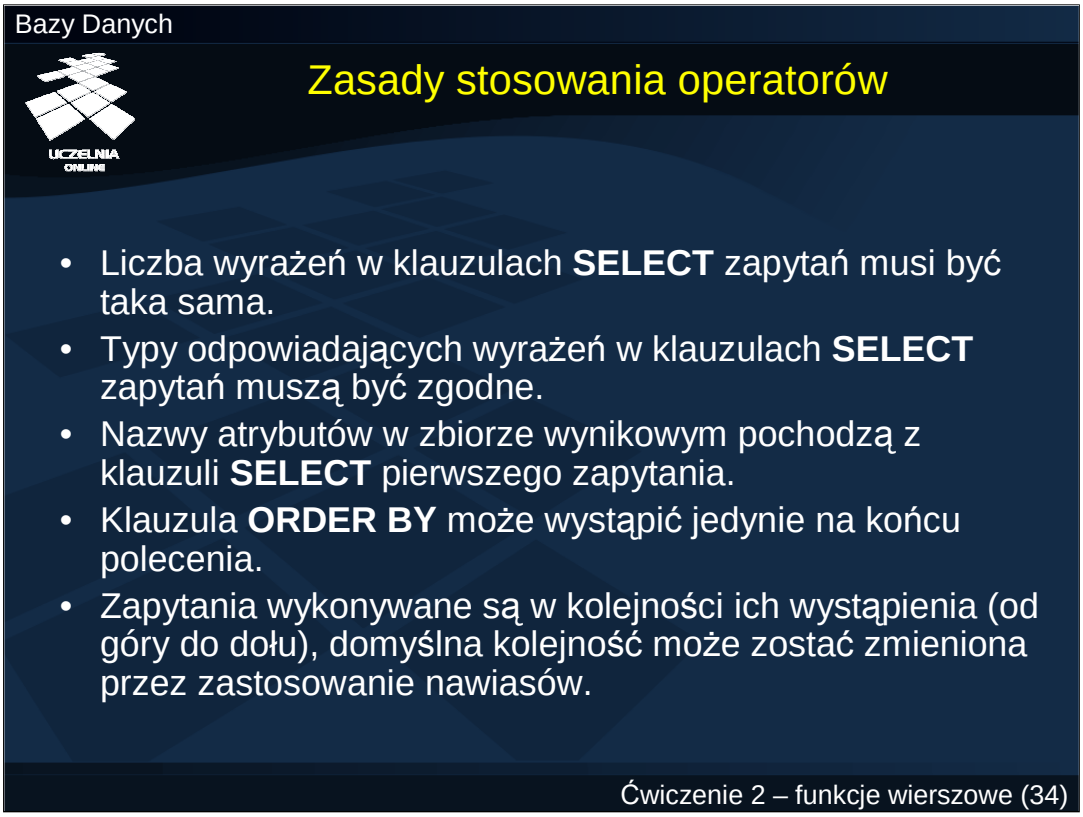


Ostatnim zagadnieniem, jakie zostanie poruszone w bieżącym rozdziale, są zapytania wykorzystujące operatory zbiorowe. Ogólny schemat takiego zapytania przedstawia bieżący slajd. Zapytanie złożone jest z kilku zapytań składowych. Każde z zapytań wylicza zbiór rekordów. Następnie zbiory łączone są z wykorzystaniem operatorów zbiorowych. Dostępne operatory to: operator UNION, wyliczający sumę dwóch zbiorów i eliminujący powtórzenia ze zbioru wynikowego, operator UNION ALL wyliczający sumę dwóch zbiorów jednak bez eliminacji powtórzeń, operator EXCEPT wyliczający różnicę dwóch zbiorów i eliminujący powtórzenia ze zbioru wynikowego oraz operator INTERSECT, znajdujący część wspólną dwóch zbiorów i eliminujący powtórzenia. W SZBD Oracle operator EXCEPT jest zastąpiony przez identycznie działający operator MINUS.



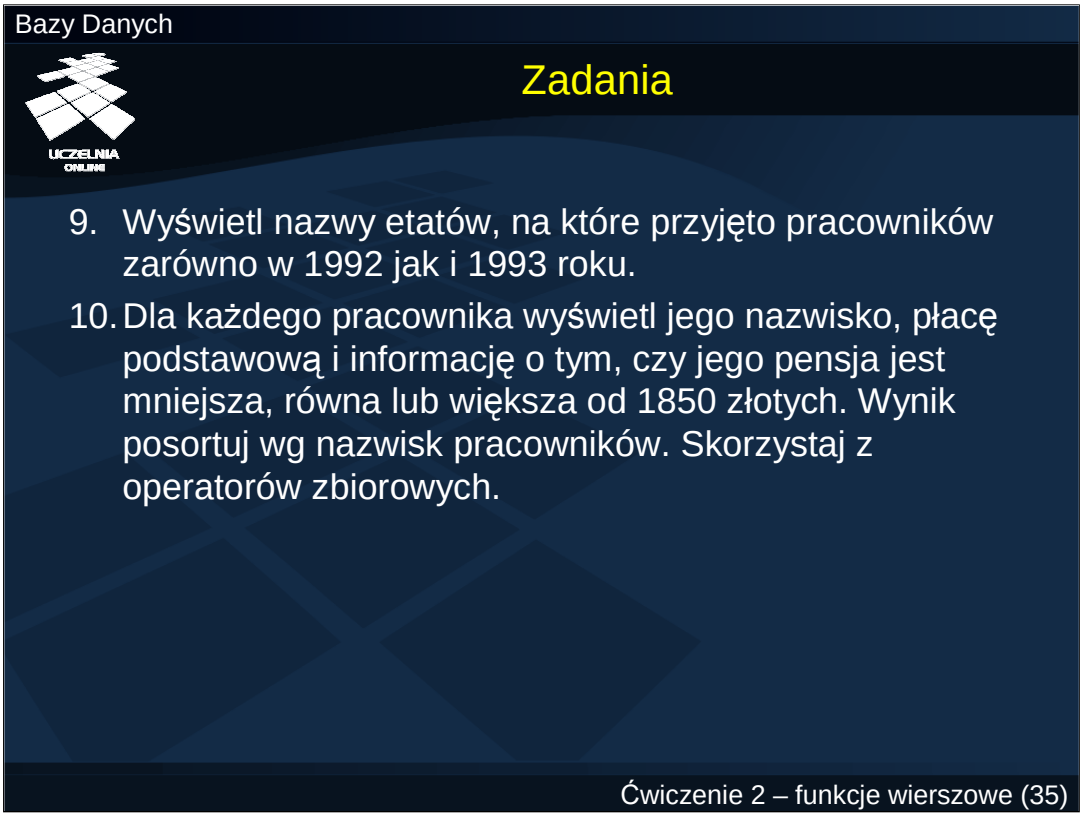
Bieżący slajd przedstawia przykład zapytań z operatorami zbiorowymi. W pierwszym przykładzie pierwsze zapytanie znajduje zbiór wartości etatów, na jakich są zatrudnieni pracownicy zespołu o numerze 10. Z kolei drugie zapytanie znajduje zbiór wartości etatów, na jakich są zatrudnieni pracownicy zespołu o numerze 20. Oba zbiory zostają połączone operatorem EXCEPT a więc wyliczającym różnicę zbiorów. W konsekwencji otrzymujemy zbiór wartości etatów, na jakich zatrudnieni są pracownicy zespołu o numerze 10, a na których nie pracują żadni pracownicy zespołu o numerze 20. Dodatkowo zostają wyeliminowane powtórzenia wartości nazw etatów.

Drugi przykład tworzy sumę dwóch zbiorów: pierwszego, zawierającego nazwiska pracowników, z drugim, zawierającym nazwy zespołów. Zbiór wynikowy zawiera zatem nazwiska pracowników „pomieszane” z nazwami zespołów.



Zasady konstruowania zapytań z operatorami zbiorowymi podlegają ścisłym rygorom. Po pierwsze, liczba wyrażeń w klauzulach SELECT wszystkich zapytań, wchodzących w skład konstrukcji, musi być taka sama. A więc jeśli w klauzuli SELECT pierwszego zapytania są trzy wyrażenia, w klauzulach SELECT wszystkich następnych zapytań też muszą być zdefiniowane trzy wyrażenia. Po wtóre, typy wartości odpowiadających sobie wyrażeń w klauzulach SELECT poszczególnych zapytań muszą być zgodne (lub istnieje możliwość realizacji domyślnej konwersji). Jeśli np. typ pierwszego wyrażenia pierwszego zapytania to liczba, pierwsze wyrażenia w klauzulach SELECT pozostałych zapytań też muszą być liczbami (lub muszą takich typów, dla których będzie możliwe przeprowadzenie konwersji do liczby).

Nagłówki atrybutów wyniku zapytania z operatorami zbiorowymi są tworzone z nazw wyrażeń lub aliasów, jakie zostały zdefiniowane w klauzuli SELECT pierwszego zapytania. Klauzula ORDER BY może wystąpić jedynie na końcu całej konstrukcji. W przypadku konstrukcji zawierającej więcej niż dwa zapytania, wykonanie następuje od góry: wynik pierwszego zapytania zostaje połączony operatorem zbiorowym z wynikiem drugiego zapytania, wynik tej operacji zostaje połączony z wynikiem trzeciego zapytania, itd. Jeśli konieczna jest zmiana kolejności wykonywania zapytań, należy użyć nawiasów, otaczając nimi te pary zapytań, które mają być zrealizowane jako pierwsze, np. w konstrukcji `zapytanie_1 union (zapytanie_2 except zapytanie_3)` najpierw połączone zostaną wyniki zapytań 2 i 3 a następnie wynik zapytania 1.

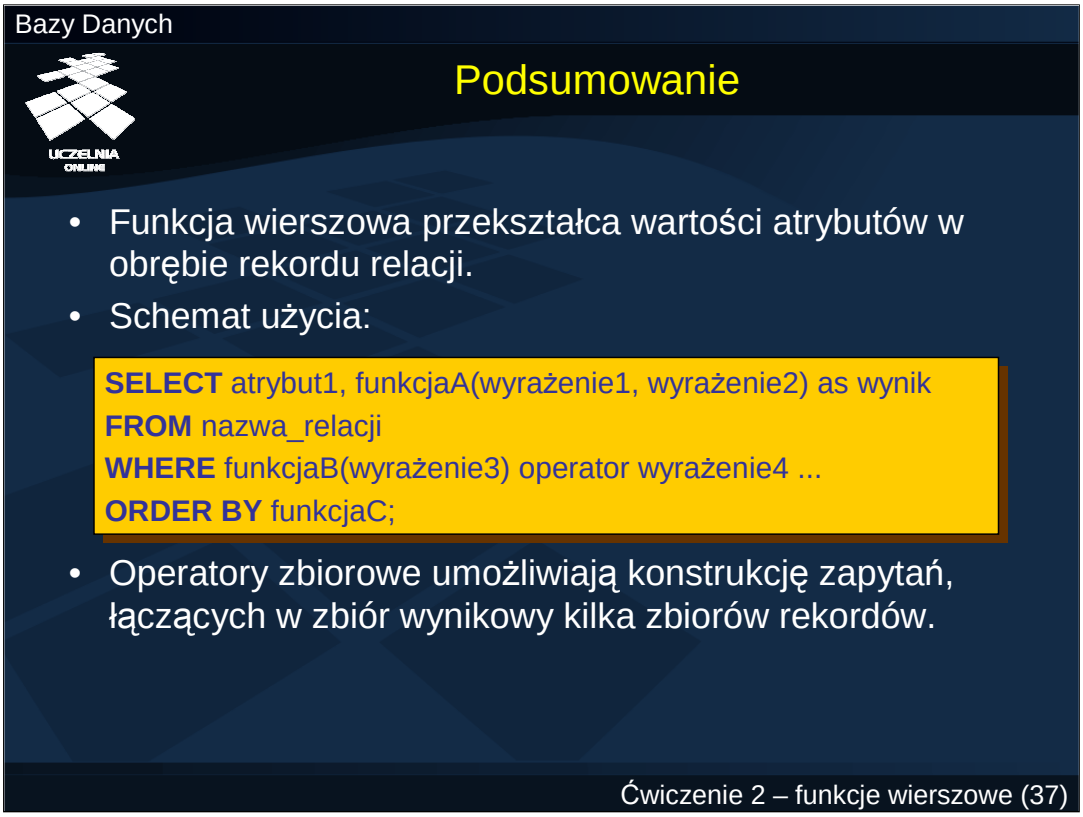


Bieżący slajd przedstawia zadania, których celem jest utrwalenie wiadomości ze stosowania operatorów zbiorowych w zapytaniach SQL.

Bieżący slajd przedstawia rozwiązania zadań (9) i (10), których treść zacytowano poniżej.

(9) Wyświetl nazwy etatów, na które przyjęto pracowników zarówno w 1992 jak i 1993 roku.

(10) Dla każdego pracownika wyświetl jego nazwisko, płacę podstawową i informację o tym, czy jego pensja jest mniejsza, równa lub większa od 1850 złotych. Wynik posortuj wg nazwisk pracowników. Skorzystaj z operatorów zbiorowych.



W zakończonym ćwiczeniu została zaprezentowana koncepcja funkcji języka SQL. W ćwiczeniu omówiono stosowanie funkcji wierszowych, przetwarzających wartości atrybutów w obrębie rekordu relacji. Przedstawiono sposób konstrukcji zapytań z funkcjami wierszowymi, a następnie omówiono poszczególne rodzaje funkcji wierszowych. Następnie zaprezentowano konstrukcje umożliwiające budowanie zapytań z wykorzystaniem operatorów zbiorowych.

Każde z omówionych zagadnień zostało utrwalone przez serię zadań.