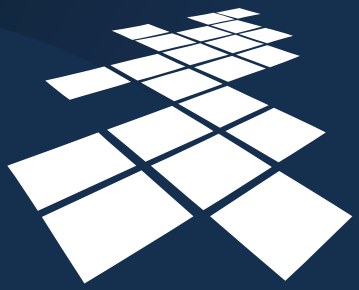
Bazy Danych

Ćwiczenie 4 - połączenia

Połączenia relacji.



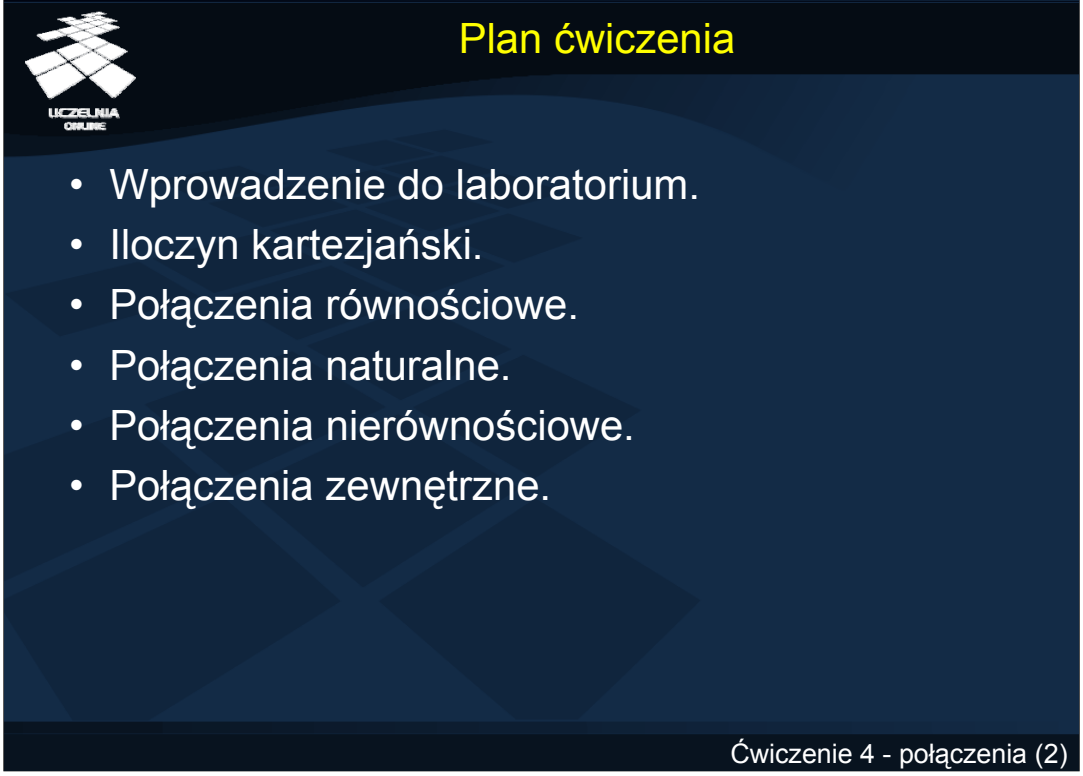
UCZELNIA
ONLINE

Ćwiczenie 4 – połączenia

Dotychczas omawiane zapytania zawsze dotyczyły jednej relacji. Możliwe jest jednak pisanie zapytań, które odczytują i łączą dane z wielu relacji. Celem tego ćwiczenia jest zapoznanie państwa z mechanizmem połączeń, oraz notacją polecenia SELECT pozwalającą na ich wykonywanie.

Wymagania:

Znajomość tematyki omawianej na poprzednich zajęciach i umiejętność jej praktycznego wykorzystania.



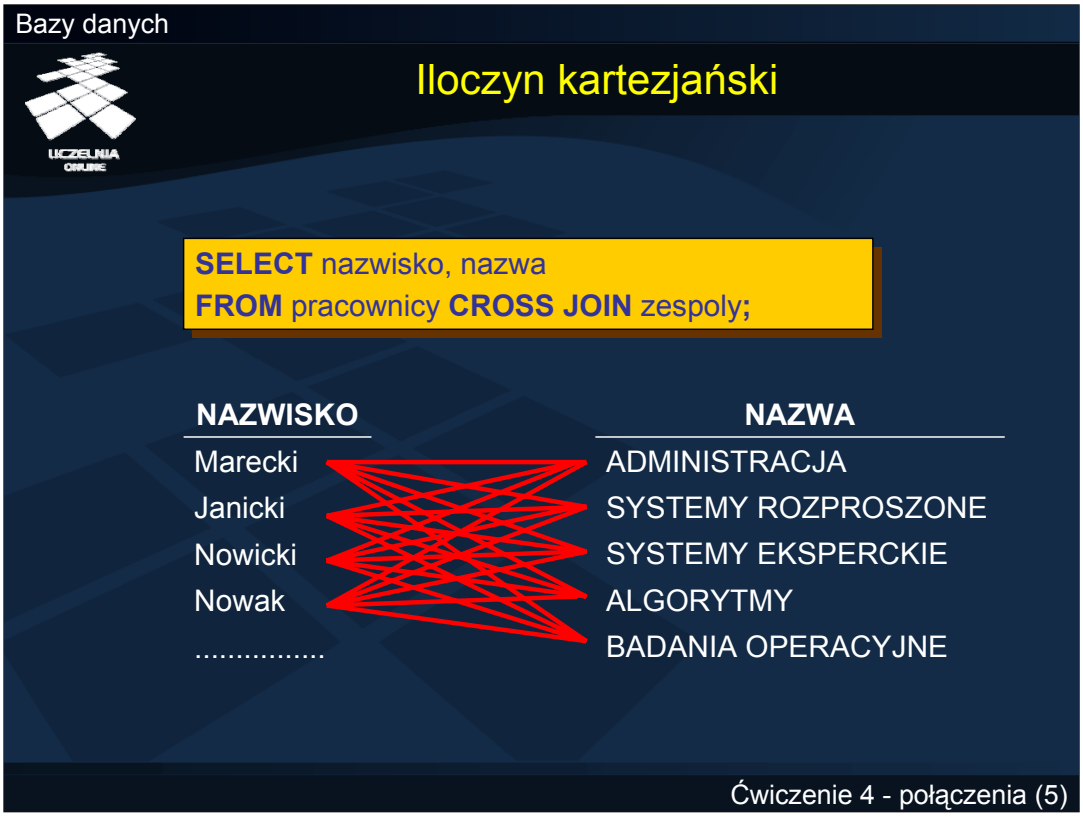
Ćwiczenie rozpoczniemy od wprowadzenia do laboratorium, na którym przedstawimy motywację stającą za mechanizmem połączeń. Następnie omówimy kolejne, coraz bardziej skomplikowane rodzaje połączeń. Rozpoczniemy od iloczynu kartezjańskiego, następnie omówimy połączenia równościowe i naturalne, oraz połączenia nierównościowe. Po omówieniu wymienionych wcześniej połączeń pokażemy, jak można w oparciu o nie definiować tzw. połączenia zewnętrzne, oraz ....

Plan ćwiczenia – cd.

- Połączenia zwrotne.
- Połączenia wielu tabel.
- Stara notacja połączeń.
- Zadania.
- Podsumowanie.

... połączenia zwrotne. Na końcu ćwiczenia pokażemy państwu jak można wykonać połączenia wielu tabel, oraz przedstawimy inną, zgodną ze starszą wersją standardu, składnię polecenia SELECT pozwalającą na wykonywanie połączeń. Każdy z wymienionych wyżej tematów zostanie zakończony krótkim zadaniem ilustrującym jego zastosowanie. Na końcu ćwiczenia przedstawimy państwu kilka dodatkowych zadań, które powinniście państwo wykonać w celu nabrania wprawy w posługiwaniu się poleceniami przedstawionymi na ćwiczeniu. Ćwiczenie zakończymy slajdem podsumowującym omówioną tematykę.

Ćwiczenie 3 poświęcone jest bardzo ważnemu mechanizmowi wykorzystywanemu przy realizacji zapytań. Jest to mechanizm tzw. „połączeń". Co to są połączenia i jaka jest motywacja stojąca za stworzeniem tego mechanizmu? Przyjrzyjmy się następującemu problemowi. Korzystając z bazy danych, poznanej na poprzednich zajęciach, chcemy odnaleźć dla każdego pracownika nazwę jego zespołu. Jak zapewne państwo pamiętacie, w relacji PRACOWNICY, z każdym pracownikiem związany jest jedynie identyfikator zespołu, w którym pracownik jest zatrudniony. Nazwy zespołów są zapisane w osobnej relacji – ZESPOLY. Pierwszym odruchem przy rozwiązywaniu tego problemu byłoby najpierw odczytać nazwiska i identyfikatory zespołów z relacji pracownicy, a potem odczytać nazwy zespołów o odczytanych wcześniej identyfikatorach. Wykorzystując wartości identyfikatorów zespołów w obu relacjach wynikowych można skojarzyć nazwisko z nazwą zespołu. Problemy z tym podejściem są dwa. Po pierwsze konieczne jest wykonanie dwóch zapytań, a po drugie należy zaimplementować własnoręcznie połączenie tych informacji. Mechanizm połączeń w języku SQL pozwala uniknąć tych problemów, gdyż pozwala na nakazanie SZBD aby połączył dane z dwóch, lub więcej, tabel. Jeżeli zapytanie SQL zostanie odpowiednio skonstruowane, to system zarządzania bazą danych sam dobierze najbardziej wydajny algorytm połączenia danych z kilku tabel, a wyniki zwróci w postaci jednej relacji wynikowej. Istnieje wiele rodzajów połączeń danych z dwóch tabel: iloczyn kartezjański oraz połączenia: naturalne, równościowe, nierównościowe, zewnętrzne i zwrotne. Każdy z tych rodzajów zostanie na niniejszych ćwiczeniach omówiony.



Najprostszym typem połączenia jest tzw. „iloczyn kartezjański” (albo *cross-join*). W wyniku iloczynu kartezjańskiego powstaje relacja, która zawiera wszystkie atrybuty z obu relacji. Krotki w tej relacji powstają jako każda możliwa kombinacja krotki z pierwszej łączonej relacji, z krotką z drugiej łączonej relacji. Jak łatwo zauważyć, liczba krotek w relacji stanowiącej wynik połączenia poprzez iloczyn kartezjański jest równa iloczynowi rozmiarów oryginalnych relacji (o ile nie wprowadzi się dodatkowych warunków selekcji). Wobec olbrzymich rozmiarów, jakie potrafią przyjmować relacje w zastosowaniach praktycznych, w większości wypadków wystąpienie iloczynu kartezjańskiego sygnalizowane jest błędem w zapytaniu. Iloczyn kartezjański w czystej postaci rzadko bywa przydatny.

W języku SQL, według standardu ANSI, sposób połączenia dwóch lub więcej tabel definiowany jest w klauzuli FROM. Połączenie poprzez iloczyn kartezjański definiowane jest za pomocą operatora połączenia CROSS JOIN umieszczanego pomiędzy nazwami łączonych relacji:

```
SELECT.....
FROM relacja1 CROSS JOIN relacja2
WHERE ....
ORDER BY .....
```

Przeanalizujmy przykład przedstawiony na slajdzie:

```
SELECT nazwisko, nazwa
FROM pracownicy CROSS JOIN zespoly;
```

Klauzula FROM zawiera połączenie, poprzez iloczyn kartezjański, dwóch relacji: PRACOWNICY i ZESPOLY. Zapytanie zatem przetwarza relację, która powstała w wyniku połączenia każdej krotki z relacji PRACOWNICY z każdą krotką z relacji ZESPOLY. Tutaj, przetwarzanie polega na projekcji, czyli wybraniu atrybutów NAZWISKO i NAZWA z relacji powstałej w wyniku połączenia i zwrócenie ich w relacji wynikowej.

Zadanie (1)

- Wyświetl wszystkie kombinacje nazw etatów zaczynających się na literę A i nazwisk pracowników na literę N.

NAZWA	NAZWISKO
ADIUNKT	Nowicki
ADIUNKT	Nowak
ASYSTENT	Nowicki
ASYSTENT	Nowak

Rozwiązanie (1)

- Wyświetl wszystkie kombinacje nazw etatów zaczynających się na literę A i nazwisk pracowników na literę N.

```
SELECT nazwa, nazwisko
FROM etaty CROSS JOIN pracownicy
WHERE nazwa LIKE 'A%' AND nazwisko LIKE 'N%';
```

Problem, o którym wspomniano na początku niniejszej prezentacji, polegający na znalezieniu dla każdego pracownika nazwy jego zespołu można rozwiązać za pomocą tzw. „połączenia równościowego” *(equi join*). W wyniku połączenia równościowego powstaje relacja, która zawiera wszystkie atrybuty z obu łączonych relacji, jednak, w przeciwieństwie do iloczynu kartezjańskiego, krotki w takiej relacji są konstruowane w inny sposób. Powstają one poprzez znalezienie wszystkich par krotek, z których jedna pochodzi z pierwszej łączonej relacji, a druga z drugiej i spełniają one tzw. „warunek połączenia”. Każda taka para jest łączona i tworzy nową krotkę w relacji wynikowej. Ważne jest, aby warunki połączeniowe porównywały jedynie wartości atrybutów pochodzących z łączonych relacji. W połączeniach równościowych warunki te muszą być oparte o operator równości (‘=‘). Podobnie jak w przypadku iloczynów kartezjańskich, połączenie równościowe jest również definiowane w klauzuli FROM:

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja1 [alias1] JOIN relacja2 [alias2] ON warunek_połączenia

WHERE ....

ORDER BY .....

W celu dokładniejszego zilustrowania ogólnej składni przedstawionej powyżej omówmy przykład pokazany na slajdzie:

SELECT pracownicy.nazwisko, z.nazwa, z.id_zesp

FROM pracownicy JOIN zespoly z ON (pracownicy.id_zesp=z.id_zesp);

Powyższe zapytanie jest rozwiązaniem problemu zdefiniowanego na początku tej prezentacji, który polegał na odnalezieniu dla każdego pracownika,nazwy jego zespołu. Nazwę zespołu pracownika można zidentyfikować korzystając z numeru zespołu (atrybut ID_ZESP), który jest każdemu pracownikowi przypisany. W relacji ZESPOLY również, dla każdego zespołu, zdefiniowano jego numer (również atrybut ID_ZESP). Oczywistym wydaje się zatem, że warunek połączeniowy tych relacji powinien być oparty o równość tych dwóch atrybutów.

Przeanalizujmy najpierw klauzulę FROM. Relacje PRACOWNICY i ZESPOLY są łączone za pomocą operatora JOIN, a warunek połączenia podawany jest za słowem kluczowym ON. Przy definicji warunku połączeniowego można napotkać na pewien problem. Otóż nazwy atrybutów, według których łączone są relacje, są takie same w obu relacjach. Konieczny jest zatem jakiś mechanizm pozwalający na rozróżnienie z których relacji pochodzą atrybuty wykorzystane w warunku. Jeżeli przyjrzymy się warunkowi połączeniowemu na przykładzie, możemy zobaczyć, że nazwę atrybutu pochodzącego z relacji PRACOWNICY poprzedzono nazwą relacji oddzieloną od nazwy atrybutu kropką. Z kolei atrybut pochodzący z drugiej relacji poprzedzono literą Z i kropką. Jeżeli przyjrzymy się nazwie relacji ZESPOLY, wymienionej po słowie kluczowym JOIN, możemy zauważyć, że za tą nazwą podano tą samą literę. Litera ta stanowi tzw. „alias", czyli alternatywną nazwę dla relacji wykorzystywaną w zapytaniu. Stąd Z.ID_ZESP jest równoważne ZESPOLY.ID_ZESP i oznacza atrybut ID_ZESP z relacji ZESPOLY. Aliasy są opcjonalne i najczęściej składają się z jednej do dwóch liter. Stosuje się je dla skrócenia zapisu zapytania oraz do zapobiegania niejednoznaczności w bardziej skomplikowanych zapytaniach. W podobny sposób należy poprzedzać nazwy atrybutów (aliasami albo nazwami relacji) w wyrażeniach w klauzulach: SELECT, WHERE, albo ORDER BY. Stąd też, w klauzuli SELECT, w przykładowym zapytaniu, nazwy atrybutów, które miały się znaleźć w relacji wynikowej, są poprzedzone aliasami, bądź nazwami relacji. Należy tutaj jeszcze zaznaczyć, że poprzedzanie nazw atrybutów aliasami, bądź nazwami relacji jest obowiązkowe jedynie w sytuacji, gdy nie zrobienie tego prowadzi do niejednoznaczności. Należy również pamiętać, że jeżeli zdefiniowano alias, to nie wolno już korzystać z oryginalnej nazwy relacji.

Zadanie (2)

- Dla każdego pracownika zatrudnionego na etacie DYREKTOR albo SEKRETARKA wyświetl jego płacę podstawową i widełki płacowe.

NAZWISKO	PLACA_POD	PLACA_OD	PLACA_DO
Marecki	4730	4280	5100
Krakowska	1590	1470	1650

Rozwiązanie (2)

- Dla każdego pracownika zatrudnionego na etacie DYREKTOR albo SEKRETARKA wyświetl jego płacę podstawową i widełki płacowe.

```
SELECT
 p.nazwisko, p.placa_pod, e.placa_od, e.placa_do
FROM etaty e JOIN pracownicy p ON (p.etat=e.nazwa)
WHERE p.etat IN ('DYREKTOR','SEKRETARKA');
```

Połączenia naturalne są specjalnym rodzajem połączeń równościowych. Połączenie naturalne dwóch relacji to połączenie równościowe relacji, w którym warunki równości dotyczą wszystkich par atrybutów o takich samych nazwach. Podstawową różnicą, pomiędzy zapytaniami równościowymi, a naturalnymi, jest lista atrybutów relacji powstającej w wyniku połączenia. W wyniku połączenia naturalnego atrybut (albo atrybuty) połączeniowe występują tylko raz, podczas gdy w wyniku połączenia równościowego występują oba atrybuty połączeniowe z obu łączonych relacji. Istnieją dwie notacje dla połączeń naturalnych:

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja1 [alias1] NATURAL JOIN relacja2 [alias2]

WHERE ....

ORDER BY .....

lub:

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja1 [alias1] JOIN relacja2 USING (atrybut1,atrybut2,.....) [alias2]

WHERE ....

ORDER BY .....

Różnica pomiędzy tymi notacjami jest taka, że pierwsza notacja automatycznie wymaga, aby wszystkie pary atrybutów o takich samych nazwach w obu łączonych relacjach były równe, a druga pozwala określić, które z par atrybutów, o takich samych nazwach, powinny być równe.

W celu lepszej ilustracji działania połączeń naturalnych, na slajdzie przedstawiono trzy równoważne zapytania. Zapytanie (1) jest identyczne z zapytaniem omawianym przy okazji połączeń równościowych. Zapytania (2) i (3) wykorzystują połączenia naturalne do realizacji tego samego zadania, co zapytanie (1).

Przeanalizujmy zapytanie (2). Relacje PRACOWNICY i ZESPOLY są łączone (w klauzuli FROM) za pomocą operatora NATURAL JOIN. Ponieważ ta odmiana połączenia naturalnego wymaga, aby wszystkie pary atrybutów o takich samych nazwach były równe, a jedynymi takimi atrybutami w obu tych relacjach są atrybuty o nazwie ID_ZESP, to relacje zostaną połączone równościowo zgodnie z warunkiem PRACOWNICY.ID_ZESP=ZESPOLY.ID_ZESP. W zapytaniu (3) użyto drugiej notacji stosowanej w połączeniach naturalnych. Relacje są łączone, tak jak w przypadku połączeń równościowych, za pomocą operatora JOIN. W przeciwieństwie jednak do połączeń równościowych, za nazwą drugiej relacji użyto słowa kluczowego USING, a nie ON, i podano wspólną nazwę atrybutów z obu łączonych relacji, które mają zostać wykorzystane do połączenia. Podobnie jak poprzednio, warunek użyty do połączenia relacji będzie następujący: PRACOWNICY.ID_ZESP=ZESPOLY.ID_ZESP. Jak zatem łatwo zauważyć, połączenia we wszystkich 3 zapytaniach przedstawionych na slajdzie są równoważne. Porównajmy obecnie klauzulę SELECT zapytania (1) z klauzulami SELECT zapytań (2) i (3). Jedyną różnicą pomiędzy tymi klauzulami jest to, iż w zapytaniu (1) nazwę atrybutu ID_ZESP poprzedzono aliasem relacji ZESPOLY, podczas gdy w zapytaniach (2) i (3) tego nie zrobiono. Przyczyną nie podania aliasu, bądź nazwy relacji, przed nazwą atrybutu ID_ZESP jest fakt, że jest to atrybut połączeniowy, a, jak wspominano na początku omawiania niniejszego slajdu, atrybuty połączeniowe występują w wyniku połączenia jedynie raz. Ponieważ atrybut ID_ZESP nie należy już do żadnej konkretnej relacji nie może być poprzedzany nazwą relacji, bądź jej aliasem.

Zadanie (3)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników.

NAZWA	COUNT(*)
ALGORYTMY	1
SYSTEMY EKSPERCKIE	3
SYSTEMY ROZPROSZONE	7
ADMINISTRACJA	2

Rozwiązanie (3)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników.

```
SELECT nazwa, count(*)
FROM pracownicy p NATURAL JOIN  zespoly z
GROUP BY nazwa;
```

Połączenia nierównościowe są połączeniami, w których warunek połączeniowy nie używa operatora równości, ale dowolny inny operator. Podobnie jak w przypadku połączenia równościowego, w wyniku połączenia nierównościowego powstaje relacja, która zawiera wszystkie atrybuty z obu relacji. Krotki są również tworzone w podobny sposób. Znajdowane są wszystkie pary krotek, z których jedna pochodzi z pierwszej łączonej relacji, a druga z drugiej i spełniają one warunki połączenia. Każda taka para jest łączona i tworzy nową krotkę w relacji powstającej w wyniku połączenia. Ogólna notacja połączeń jest taka sama jak dla połączeń równościowych (zmieniają się tylko warunki połączeniowe):

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja1 [alias1] JOIN relacja2 [alias2] ON warunek_połączenia

WHERE ....

ORDER BY .....

Przeanalizujmy zapytanie przykładowe pokazane na slajdzie.

SELECT nazwisko, nazwa, placa_pod, placa_od, placa_do

FROM pracownicy JOIN etaty

ON placa_pod BETWEEN placa_od AND placa_do;

Zapytanie wykonuje połączenie nierównościowe relacji PRACOWNICY i ETATY. Warunkiem połączeniowym jest tutaj to, iż płaca podstawowa (atrybut PLACA_POD) powinna się mieścić w widełkach płacowych dla konkretnego etatu. Z otrzymanej w wyniku połączenia relacji wyciągane są atrybuty NAZWISKO, NAZWA, PLACA_POD, PLACA_OD i PLACA_DO i zwracane w relacji wynikowej. W zapytaniu nie poprzedzono żadnego atrybutu nazwą relacji, bądź aliasem. Jest tak dlatego, iż wszystkie atrybuty w obu relacjach mają różne nazwy, a zatem podanie samej nazwy atrybutu jest jednoznaczne.

Zadanie (4)

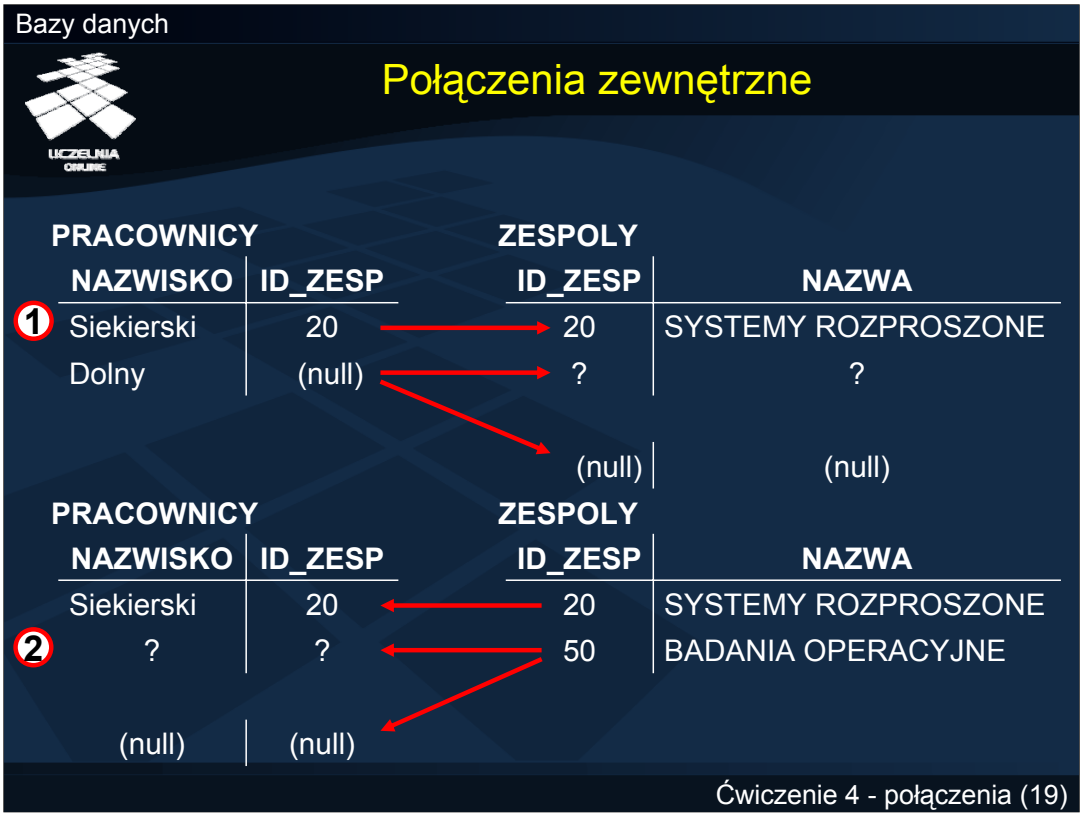
- Wyświetl nazwiska i etaty pracowników, których rzeczywiste zarobki odpowiadają widełkom płacowym przewidzianym dla sekretarek.

NAZWISKO	ETAT
Krakowska	SEKRETARKA

Rozwiązanie (4)

- Wyświetl nazwiska i etaty pracowników, których rzeczywiste zarobki odpowiadają widełkom płacowym przewidzianym dla sekretarek.

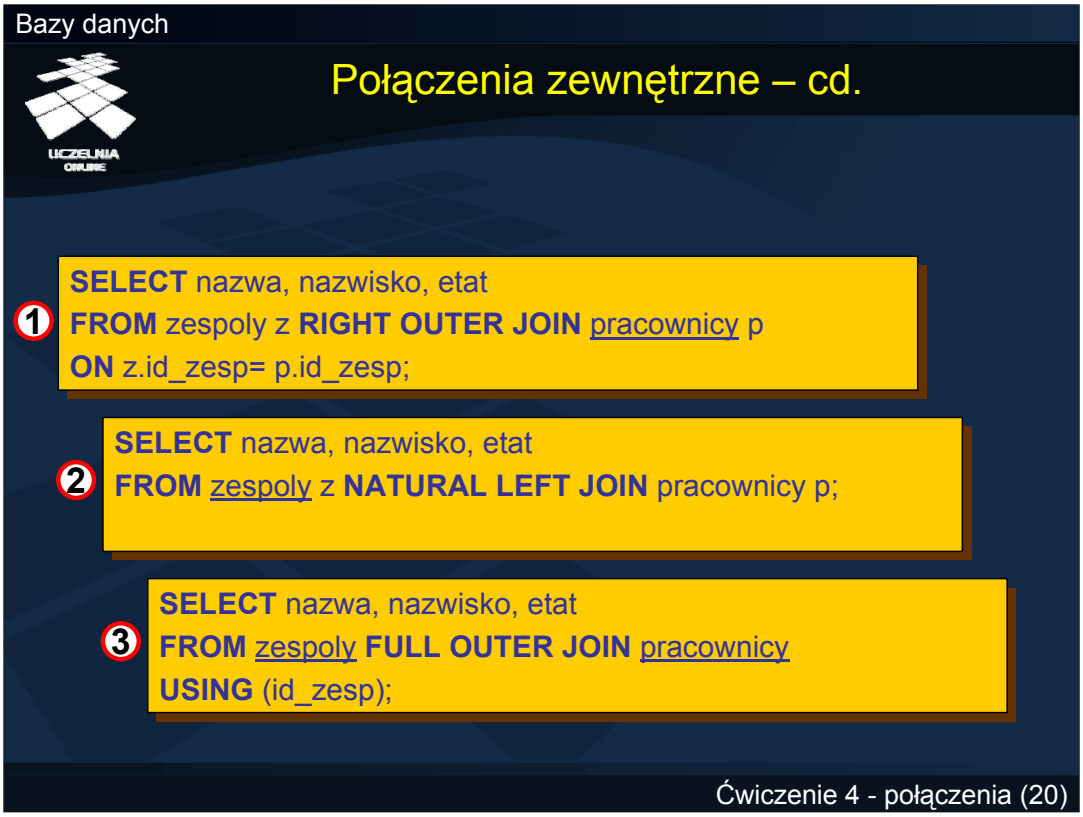
```
SELECT nazwisko, etat
FROM pracownicy p JOIN  etaty e ON
  placa_pod BETWEEN placa_od AND placa_do
WHERE nazwa = 'SEKRETARKA';
```



We wszystkich opisanych dotychczas rodzajach połączeń, w relacji powstającej w wyniku połączenia, znajdują się jedynie krotki, które spełniają warunki połączenia. Taki typ połączeń nazywany jest „połączeniem wewnętrznym” (*inner join*). Istnieją również „połączenia zewnętrzne” (*outer join*), w których można zażądać, aby wszystkie krotki z jednej, albo z obydwu łączonych relacji znalazły się w wyniku połączenia, nawet takie, które nie spełniają warunków połączenia (nie znalazły pary). Aby móc zachować wszystkie krotki z jednej relacji, do drugiej relacji wprowadzana jest „wirtualna” krotka, która wypełniona jest wartościami pustymi. Wszystkie krotki z relacji, które nie mogą znaleźć swojej pary, łączone są z "wirtualną" krotką w drugiej relacji. Koncepcję połączeń zewnętrznych ilustrują przykłady pokazane na slajdzie.

Zacznijmy od przykładu (1). W relacji PRACOWNICY znajduje się pracownik o nazwisku „Dolny”, który nie jest przydzielony do żadnego zespołu (atrybut ID_ZESP ma wartość NULL). Dane o tym pracowniku nie znalazłyby się w wyniku normalnego połączenia równościowego, gdyż nie zostałaby znaleziona żadna odpowiadająca mu krotka w relacji ZESPOLY. W wyniku połączenia zewnętrznego, w którym zażądalibyśmy, aby wszystkie krotki z relacji PRACOWNICY znalazły się w wyniku połączenia, krotka dotycząca pracownika „Dolnego” zostałaby połączona z wirtualną krotką umieszczoną w relacji ZESPOLY i znalazłaby się w wyniku.

Na przykładzie (2) pokazana jest podobna sytuacja. W relacji ZESPOLY zdefiniowano zespół BADANIA OPERACYJNE (o ID_ZESP równym 50), w którym nikt nie jest zatrudniony. Żaden pracownik nie ma atrybutu ID_ZESP równego ID_ZESP zespołu BADANIA OPERACYJNE. Dane o tym zespole nie znalazłyby się w wyniku normalnego połączenia równościowego, gdyż nie zostałaby znaleziona żadna odpowiadająca mu krotka w relacji PRACOWNICY. W wyniku połączenia zewnętrznego, w którym zażądalibyśmy, aby wszystkie krotki z relacji ZESPOLY znalazły się w wyniku połączenia, krotka dotycząca zespołu „BADANIA OPERACYJNE” zostałaby połączona z wirtualną krotką i znalazłaby się w wyniku.



Ogólna składnia połączeń zewnętrznych wygląda następująco:

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja1 [alias1] [NATURAL] {LEFT|RIGHT|FULL} [OUTER] JOIN relacja2 [alias2] {ON (warunek_połączenia1) | USING (atrybut) | ø}

WHERE ....

ORDER BY .....

Aby zilustrować sposób tworzenia zapytań zewnętrznych, przedstawiono na slajdzie kilka przykładowych zapytań.

1. Zapytanie (1)

SELECT nazwa, nazwisko, etat

FROM zespoly z RIGHT OUTER JOIN pracownicy p

ON z.id_zesp= p.id_zesp;

W powyższym zapytaniu, relacje łączone są za pomocą operatora RIGHT OUTER JOIN. Znaczy to, że relacje ZESPOLY i PRACOWNICY będą łączone za pomocą połączenia zewnętrznego, a relacja, z której wszystkie krotki mają się znaleźć w wyniku połączenia, to relacja po prawej stronie operatora (RIGHT OUTER JOIN), czyli relacja PRACOWNICY. Takie połączenia nazywane są „połączeniami zewnętrznymi prawostronnymi”. Prócz tego, że jest to połączenie zewnętrzne, jest to typowe połączenie równościowe, gdyż warunek połączeniowy korzysta z operatora równości. W ogólności jednak może to być dowolny operator, a zatem ta składnia nadaje się również do definiowania „zewnętrznych połączeń nierównościowych”.

2. Zapytanie (2)

```
SELECT nazwa, nazwisko, etat
FROM zespoly z NATURAL LEFT JOIN pracownicy p;
```

W powyższym zapytaniu, relacje łączone są za pomocą operatora NATURAL LEFT JOIN. Znaczy to, że relacje ZESPOLY i PRACOWNICY będą łączone za pomocą naturalnego połączenia zewnętrznego, a relacja, z której wszystkie krotki mają się znaleźć w wyniku połączenia, to relacja po lewej stronie operatora (LEFT JOIN), czyli relacja ZESPOLY. Takie połączenia nazywane są „połączeniami zewnętrznymi lewostronnymi". Ponieważ mamy do czynienia z połączeniem naturalnym, warunkiem połączeniowym jest tutaj równość wartości na atrybutach o takich samych nazwach. Dodatkowo, ponieważ jest to połączenie zewnętrzne lewostronne, wszystkie krotki z relacji ZESPOLY znajdą się w wyniku połączenia. Należy tutaj zwrócić uwagę na jeszcze dwie rzeczy. Otóż, jak łatwo zauważyć, pominięto w zapytaniu słowo kluczowe OUTER. Słowo to jest nieobowiązkowe, a o tym, czy połączenie jest zewnętrzne, czy nie, decyduje obecność słowa kluczowego RIGHT, LEFT albo FULL (patrz poniżej). Drugą rzeczą jest fakt, iż pod względem funkcjonalnym połączenia lewostronne i prawostronne się niczym nie różnią, gdyż można zamienić kolejność nazw relacji w zapytaniu, zmieniając tym samym wynik połączenia w taki sam sposób w jaki zmieniłaby go zamiana operatora połączenia lewostronnego na operator połączenia prawostronnego.

3. Zapytanie (3)

```
SELECT nazwa, nazwisko, etat
FROM zespoly FULL OUTER JOIN pracownicy
USING (id_zesp);
```

W powyższym zapytaniu, relacje łączone są za pomocą operatora FULL OUTER JOIN, a za nazwą drugiej relacji użyto słowa kluczowego USING i podano atrybut ID_ZESP. Znaczy to, że relacje ZESPOLY i PRACOWNICY będą łączone za pomocą naturalnego połączenia zewnętrznego. Ten przykład pokazuje specjalny typ połączenia zewnętrznego, w którym żądamy, aby wszystkie krotki z obu relacji pojawiły się w wyniku połączenia przynajmniej raz (FULL OUTER JOIN). Takie połączenia nazywane są „połączeniami zewnętrznymi pełnymi". Wynik takiego połączenia można najłatwiej zrozumieć jako sumę wyników połączenia lewostronnego i prawostronnego:

```
SELECT nazwa, nazwisko, etat
FROM zespoly LEFT OUTER JOIN pracownicy
USING (id_zesp)
UNION
SELECT nazwa, nazwisko, etat
FROM zespoly RIGHT OUTER JOIN pracownicy
USING (id_zesp);
```

Ponieważ mamy do czynienia z połączeniem naturalnym, warunkiem połączeniowym jest tutaj równość wartości na atrybucie ID_ZESP w obu łączonych relacjach.

W ogólności podział typów połączeń ze względu na to które krotki trafiają do relacji wynikowej (wewnętrzne, zewnętrzne - lewostronne, prawostronne, pełne) jest ortogonalny względem podziału połączeń ze względu na warunek połączenia (równościowe, naturalne, nierównościowe). Każdą z kombinacji tych typów połączeń można skonstruować (za wyjątkiem połączenia typu iloczyn kartezjański, który jest zupełnie osobnym typem połączenia).

Zadanie (5)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników. W wyniku ma zostać uwzględniony zespół BADANIA OPERACYJNE, na którym nie zatrudniono żadnego pracownika.

NAZWA	COUNT(NAZWISKO)
SYSTEMY EKSPERCKIE	3
ALGORYTMY	1
SYSTEMY ROZPROSZONE	7
ADMINISTRACJA	2
BADANIA OPERACYJNE	0

Rozwiązanie (5)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników. W wyniku ma zostać uwzględniony zespół BADANIA OPERACYJNE, na którym nie zatrudniono żadnego pracownika.

```
SELECT nazwa, count(nazwisko)
FROM pracownicy p NATURAL RIGHT JOIN  zespolu z
GROUP BY nazwa;
```

„Połączenia zwrotne" (*self join*) są specjalnym przypadkiem połączeń, w których łączymy tabelę z samą sobą. Połączeniem zwrotnym może być dowolny typ połączenia (wewnętrzne, zewnętrzne, równościowe i nierównościowe), za wyjątkiem połączenia naturalnego, co wynika z faktu, że łączenie równościowe relacji z samą sobą według atrybutów o tej samej nazwie nic nie daje (co najwyżej oryginalną relację). Przykładowym zastosowaniem połączeń zwrotnych może być znajdowanie nazwiska szefa dla każdego pracownika. W relacji PRACOWNICY, dla każdego pracownika pamiętany jest identyfikator pracownika, który jest jego szefem. Aby odnaleźć nazwisko szefa należy połączyć relację PRACOWNICY z samą sobą, stosując warunek połączeniowy ID_PRAC=ID_SZEFA. Ilustruje to rysunek na slajdzie.

Ogólna składnia połączenia zwrotnego jest taka sama, jak każdego innego typu połączenia omawianego poprzednio. Jedyną różnicą jest tutaj podanie tej samej nazwy relacji po obu stronach operatora definiującego połączenie. Dodatkowo, przy pisaniu zapisań z połączeniem zwrotnym należy pamiętać, żeby nadać różne aliasy obu wystąpieniom nazwy relacji w zapytaniu. Jest to konieczne aby możliwe było rozróżnienie z którego wystąpienia relacji pochodzi atrybut. Rozważmy przykładowe zapytanie na slajdzie:

```
SELECT p.nazwisko AS pracownik,
  s.nazwisko AS szef
FROM pracownicy p JOIN pracownicy s
ON p.id_szefa = s.id_prac;
```

W zapytaniu tym, relacja PRACOWNICY jest łączona sama z sobą za pomocą operatora połączenia JOIN. Każde z wystąpień nazwy tej relacji w zapytaniu ma nadany inny alias. Można zatem traktować obydwa wystąpienia relacji PRACOWNICY jako dwie relacje: jedną, która przechowuje dane o pracownikach i drugą, która przechowuje dane o szefach. W powyższym zapytaniu relację PRACOWNICY z aliasem P traktujemy jako relację z pracownikami, a relację PRACOWNICY z aliasem S jako relację z szefami. Aby zatem znaleźć nazwiska szefów musimy połączyć relację pracowników z relacją szefów stosując warunek połączeniowy P.ID_SZEFA=S.ID_PRAC (identyfikator szefa pracownika musi być równy identyfikatorowi pracownika będącego szefem). W wyniku połączenia równościowego, przy wykorzystaniu tego warunku, otrzymujemy relację z krotkami powstałymi w wyniku sklejenia krotek pracowników z krotkami ich szefów. Za pomocą klauzuli SELECT, z relacji powstałej w wyniku połączenia wybierane są atrybuty reprezentujące nazwiska pracownika i jego szefa, i zwracane w relacji wynikowej.

Zadanie (6)

- Wyświetl nazwiska wszystkich pracowników, którzy zarabiają więcej od Nowickiego.

NAZWISKO
Marecki
Janicki
Nowak
Kowalski

Rozwiązanie (6)

- Wyświetl nazwiska wszystkich pracowników, którzy zarabiają więcej od Nowickiego.

```
SELECT p.nazwisko
FROM pracownicy p JOIN  pracownicy r
  ON p.placa_pod > r.placa_pod
WHERE r.nazwisko='Nowicki';
```

Jak wspomniano wcześniej, w wyniku połączenia powstaje relacja, która jest następnie dalej przetwarzana w celu realizacji zapytania (selekcja, projekcja, grupowanie itp.). Ponieważ wynik połączenia jest relacją, to nic nie stoi na przeszkodzie, aby nie można jej było połączyć z kolejną relacją. W ten sposób można wykonywać dowolną liczbę połączeń. Ostateczna składnia polecenia SELECT z uwzględnieniem możliwości definicji dowolnej liczby połączeń wygląda następująco:

SELECT relacja1.atrybut, alias2.atrybut.....

FROM relacja

WHERE ....

ORDER BY .....

Gdzie „relację” można, w sposób rekursywny, zdefiniować następująco:

- nazwa relacji [alias]

- (relacja)

- relacja1 CROSS JOIN relacja2

- relacja1 [NATURAL] [{LEFT|RIGHT|FULL} [OUTER]] JOIN relacja2 {ON (warunek_połączenia1) | USING (atrybut) | ø}

Jak łatwo zauważyć, dla każdego połączenia definiowany jest warunek połączenia (za wyjątkiem iloczynu kartezjańskiego). Ponieważ połączeń jest o jedno mniej niż łączonych relacji, tyle też należy w zapytaniu zdefiniować warunków połączeniowych. Dodatkową ważną uwagą jest to, iż operator połączenia jest łączny lewostronnie, chociaż priorytet połączeń można zmieniać za pomocą nawiasów (stąd nawiasy w rekursywnej definicji przedstawionej powyżej).

W celu demonstracji składni poleceń SQL z wieloma połączeniami, przedstawiono na slajdzie dwa przykładowe zapytania. Zacznijmy od analizy zapytania (1).

```
SELECT
 P.NAZWISKO, S.NAZWISKO, E.NAZWA,
 PLACA_OD, PLACA_DO, Z. NAZWA
FROM ZESPOLY Z
 NATURAL RIGHT OUTER JOIN PRACOWNICY P
 JOIN ETATY E ON P.ETAT=E.NAZWA
 JOIN PRACOWNICY S ON P.ID_SZEFA=S.ID_PRAC;
```

W zapytaniu tym mamy do czynienia z trzema połączeniami. Ponieważ w klauzuli FROM nie występują nawiasy, należy przyjąć porządek wykonywania połączeń zgodny z łącznością operatora połączenia (łączność lewostronna). Pierwszym połączeniem, które się wykona jest zatem połączenie tabel ZESPOLY (alias Z) i PRACOWNICY (alias P) za pomocą połączenia naturalnego, zewnętrznego prawostronnego. Ponieważ jedynymi atrybutami o takich samych nazwach w obu tych relacjach są atrybuty o nazwie ID_ZESP, relacje te są łączone równościowo według warunku Z.ID_ZESP=P.ID_ZESP. Ponieważ jest to połączenie zewnętrzne prawostronne, to wszystkie krotki z relacji PRACOWNICY znajdą się w wyniku połączenia. W wyniku połączenia otrzymujemy relację, w której każdemu pracownikowi przypisano dane związane z jego zespołem. Pracownicy, którzy nie zostali przypisani do zespołu mają w tych miejscach wartości puste (NULL). Wynik pierwszego połączenia jest następnie łączony z tabelą ETATY (alias E) za pomocą wewnętrznego połączenia równościowego według warunku P.ETAT=E.NAZWA. W wyniku tego połączenia każdemu pracownikowi przypisano dodatkowo dane dotyczące minimalnej i maksymalnej płacy, jaką może otrzymywać ze względu na swój etat. Ostatecznie, wynik poprzednich połączeń jest łączony z relacją PRACOWNICY (alias S) za pomocą równościowego, lewostronnego połączenia zewnętrznego z warunkiem połączenia P.ID_SZEFA=S.ID_PRAC. Z warunku połączenia wynika, że każdemu z pracowników, znajdujących się w wyniku poprzednich połączeń, zostaną przypisane dodatkowo dane dotyczące jego szefa. Połączenie zewnętrzne zastosowano, aby zachować w wyniku wszystkich pracowników (również tych, którzy szefa nie posiadają). Z relacji otrzymanej w wyniku wszystkich połączeń wybierane są atrybuty reprezentujące: nazwisko pracownika (P.NAZWISKO), nazwisko szefa (S.NAZWISKO), nazwę etatu pracownika (E.NAZWA), minimalną i maksymalną płacę na etacie (PLACA_OD i PLACA_DO) oraz nazwę zespołu pracownika, które ostatecznie są zwracane w postaci relacji wynikowej.

Przejdźmy obecnie do drugiego przykładowego zapytania (2):

```
SELECT NAZWISKO, PLACA_OD, PLACA_DO, Z. NAZWA
FROM ZESPOLY Z FULL OUTER JOIN
 (PRACOWNICY P JOIN ETATY E ON P.ETAT=E.NAZWA) ON
 P.ID_ZESP = Z.ID_ZESP;
```

W zapytaniu tym priorytet połączeń został zmodyfikowany za pomocą nawiasów. Pierwszym wykonywanym połączeniem jest równościowe połączenie wewnętrzne pomiędzy relacjami PRACOWNICY i ETATY. W wyniku takiego połączenia powstaje relacja, w której każdemu pracownikowi przypisane są widełki jego płacy wynikające z jego etatu. Wynik tego połączenia jest następnie łączony za pomocą pełnego zewnętrznego połączenia równościowego z relacją ZESPOLY, w wyniku czego otrzymujemy relację, w której każdemu pracownikowi przypisano dane dotyczące zespołu, w którym jest zatrudniony. Jeżeli pracownik nie jest zatrudniony w żadnym zespole, ma w tym miejscu wartości puste. Również każdy zespół znajduje się w wyniku przynajmniej raz, i jeżeli żaden pracownik nie jest do niego przypisany, to w atrybutach dotyczących pracowników zapisane są wartości puste.

Zadanie (7)

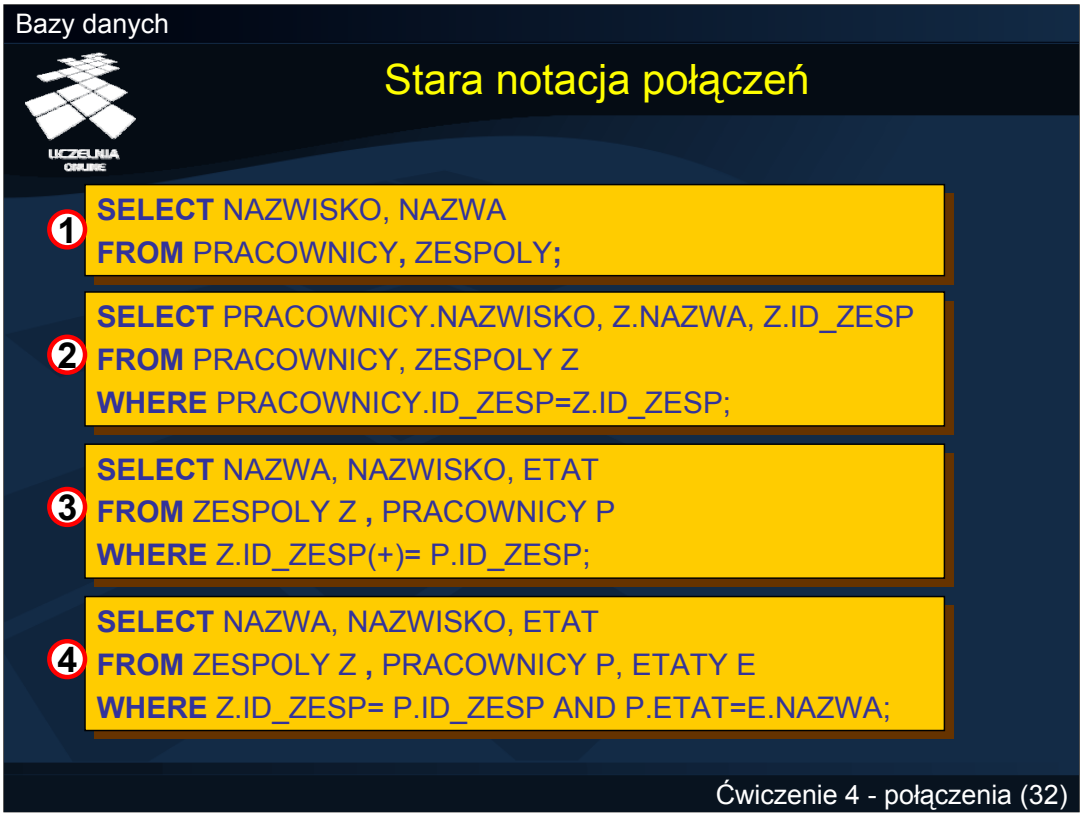
- Wyświetl dla każdego pracownika jego nazwisko, nazwisko jego szefa, adres zespołu pracownika i adres zespołu szefa. Dobierz odpowiednio typy połączeń tak, aby wszyscy pracownicy znaleźli się w rozwiązaniu (zarówno Ci nie przydzieleniu do zespołów, jak i ci bez szefów).

NAZWISKO	NAZWISKO	NAZWA	NAZWA
Makowski	Marecki	ADMINISTRACJA	ADMINISTRACJA
Dolny	Nowicki		SYSTEMY EKSPERCKIE
Marecki		ADMINISTRACJA	
.....	.....	.....	.....

Rozwiązanie (7)

- Wyświetl dla każdego pracownika jego nazwisko, nazwisko jego szefa, adres zespołu pracownika i adres zespołu szefa. Dobierz odpowiednio typy połączeń tak, aby wszyscy pracownicy znaleźli się w rozwiązaniu (zarówno Ci nie przydzieleniu do zespołów, jak i ci bez szefów).

```
SELECT P.NAZWISKO, S.NAZWISKO, PZ.NAZWA, SZ. NAZWA
FROM (PRACOWNICY P NATURAL LEFT JOIN ZESPOLY PZ)
 LEFT JOIN (PRACOWNICY S NATURAL LEFT JOIN ZESPOLY SZ)
 ON P.ID_SZEFA=S.ID_PRAC;
```



Dotychczas opisano sposób łączenia tabel zdefiniowany w późniejszych wersjach standardu SQL. W starszych wersjach stosowano inny zapis, który teraz zostanie pokrótce przedstawiony. Starsze połączenia były wszystkie definiowane w oparciu o pomysł filtrowania wyniku iloczynu kartezjańskiego za pomocą standardowej klauzuli służącej do selekcji (WHERE). W klauzuli FROM definiowano zatem jedynie iloczyn kartezjański poprzez wymienienie po przecinku wszystkich relacji wchodzących w jego skład. Zapytanie (1) przedstawione na slajdzie definiuje właśnie iloczyn kartezjański relacji PRACOWNICY i ZESPOŁY. W sytuacji, gdy konieczne było wykonanie połączenia równościowego, z wyniku takiego iloczynu wybierano, za pomocą klauzuli WHERE, jedynie krotki spełniające warunki połączenia (zapytanie (2)). W podobny sposób wykonywano połączenia nierównościowe. Takie podejście na pierwszy rzut oka wydaje się być bardzo niewydajne, jednak większość SZBD jest w stanie wykryć typ połączenia na podstawie warunków w klauzuli WHERE i zastosować najbardziej wydajny algorytm. Ten sposób łączenia tabel nie uwzględniał połączeń zewnętrznych. Stało się to przyczyną powstania rozwiązań specyficznych dla SZBD, np. takich jak przedstawione na zapytaniu (3). Rozwiązanie przedstawione na tym zapytaniu jest charakterystyczne dla SZBD firmy ORACLE. W zapytaniu (3) umieszczono w klauzuli WHERE, przy jednym z atrybutów w warunku połączeniowym, operator (+). Znaczenie tego operatora jest następujące: „dla tego połączenia, umieść wirtualną krotkę (krotkę z pustymi wartościami) w relacji, z której pochodzi atrybut, przy którym umieszczono niniejszy operator”. W konsekwencji, w wyniku połączenia, wszystkie krotki z drugiej relacji, która uczestniczyła w połączeniu (nie tej przy której umieszczono operator) znajdowały się w wyniku. Przykładowo, w zapytaniu (3) operator (+) umieszczono przy atrybucie ID_ZESP pochodzącym z relacji ZESPOLY, a zatem w tej relacji pojawiła się wirtualna krotka. W związku z tym, wszystkie krotki z relacji PRACOWNICY znajdą się w rozwiązaniu. Niestety, w tej notacji nie jest możliwe zdefiniowanie pełnego połączenia zewnętrznego i jeżeli zachodzi potrzeba wykonania takiego połączenia, to należy zapytanie rozbić na dwa, a wynik połączyć za pomocą operatora UNION.

Zapytanie (4) pokazuje sposób wykonania połączenia kilku tabel. Podobnie jak w poprzednich przykładach wykonywany jest tutaj iloczyn kartezjański wszystkich tabel, a następnie, za pomocą warunków umieszczonych w klauzuli WHERE, wybierane są jedynie te krotki, o które chodzi. Stara notacja nie pozwala również na tworzenie połączeń naturalnych.

Zadanie (8)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników. Ćwiczenie wykonaj korzystając ze starej notacji połączeń.

NAZWA	COUNT(*)
ALGORYTMY	1
SYSTEMY EKSPERCKIE	3
SYSTEMY ROZPROSZONE	7
ADMINISTRACJA	2

Rozwiązanie (8)

- Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników. Ćwiczenie wykonaj korzystając ze starej notacji połączeń.

```
SELECT NAZWA, COUNT(NAZWISKO)
FROM PRACOWNICY P, ZESPOLY Z
WHERE P.ID_ZESP=Z.ID_ZESP
GROUP BY NAZWA
```

Zadania

9. Wyświetl nazwiska, etaty, numery zespołów i nazwy zespołów wszystkich pracowników.
10. Wyświetl wszystkich pracowników z ul. PIOTROWO 3A. Uporządkuj wyniki według nazwisk pracowników.
11. Wyświetl nazwiska, miejsca pracy oraz nazwy zespołów tych pracowników, których miesięczna pensja przekracza 1000.
12. Dla każdego pracownika wyświetl jego kategorię płacową i widełki płacowe w jakich mieści się pensja pracownika.

Wykonaj zadania przedstawione na tym i na kilku kolejnych slajdach.

Zadania – cd.

13. Wyświetl nazwiska, etaty, wynagrodzenia, kategorie płacowe i nazwy zespołów pracowników nie będących asystentami. Wyniki uszereguj zgodnie z malejącym wynagrodzeniem.
14. Wyświetl nazwisko, etat, dochody (placa z uwzględnieniem płacy dodatkowej), nazwa zespołu i etat wynikający z przynależności do kategorii płacowej, dla tych pracowników, którzy są asystentami lub adiunktami i których dochody przekraczają 2000.

Zadania – cd.

15. Wyświetl nazwiska i numery pracowników wraz z numerami i nazwiskami ich szefów.
16. Zmodyfikuj powyższe zlecenie w ten sposób, aby było możliwe wyświetlenie pracownika o nazwisku Marecki (który nie ma szefa).
17. Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników i ich średnią płacę (z uwzględnieniem zespołów, na których nie zatrudniono żadnych pracowników).

Zadania – cd.

18. Dla każdego pracownika posiadającego podwładnych wyświetl ich liczbę. Wyniki posortuj zgodnie z malejącą liczbą podwładnych.
19. Wyświetl nazwiska i daty zatrudnienia pracowników, którzy zostali zatrudnieni nie później niż 10 lat (3650 dni) po swoich przełożonych.

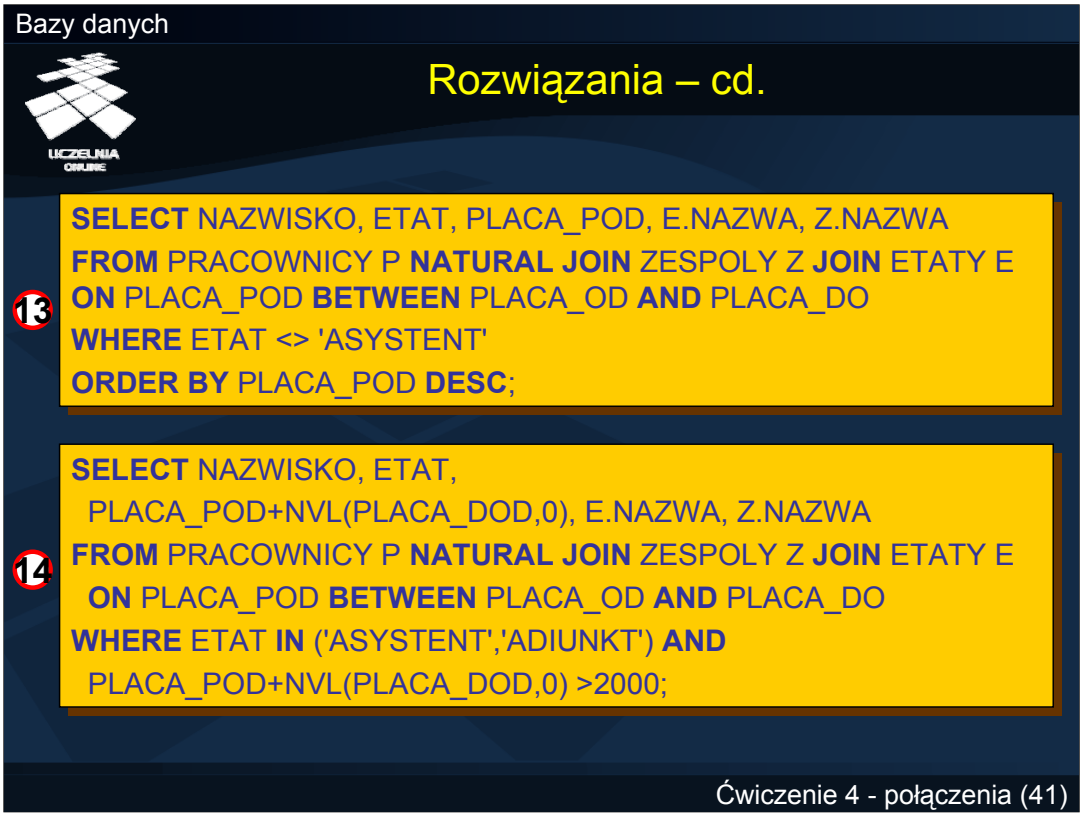
Slajd przedstawia rozwiązania zadań: (9), (10), (11) i (12), których treść zacytowano poniżej.

(9) Wyświetl nazwiska, etaty, numery zespołów i nazwy zespołów wszystkich pracowników.

(10) Wyświetl wszystkich pracowników z ul. PIOTROWO 3A. Uporządkuj wyniki według nazwisk pracowników.

(11) Wyświetl nazwiska, miejsca pracy oraz nazwy zespołów tych pracowników, których miesięczna pensja przekracza 1000.

(12) Dla każdego pracownika wyświetl jego kategorię płacową i widełki płacowe w jakich mieści się pensja pracownika.



Slajd przedstawia rozwiązania zadań: (13) i (14), których treść zacytowano poniżej.

(13) Wyświetl nazwiska, etaty, wynagrodzenia, kategorie płacowe i nazwy zespołów pracowników nie będących asystentami. Wyniki uszereguj zgodnie z malejącym wynagrodzeniem.

(14) Wyświetl nazwisko, etat, dochody (płaca z uwzględnieniem płacy dodatkowej), nazwa zespołu i etat wynikający z przynależności do kategorii płacowej, dla tych pracowników, którzy są asystentami lub adiunktami i których dochody przekraczają 2000.

Slajd przedstawia rozwiązania zadań: (15), (16) i (17) których treść zacytowano poniżej.

(15) Wyświetl nazwiska i numery pracowników wraz z numerami i nazwiskami ich szefów.

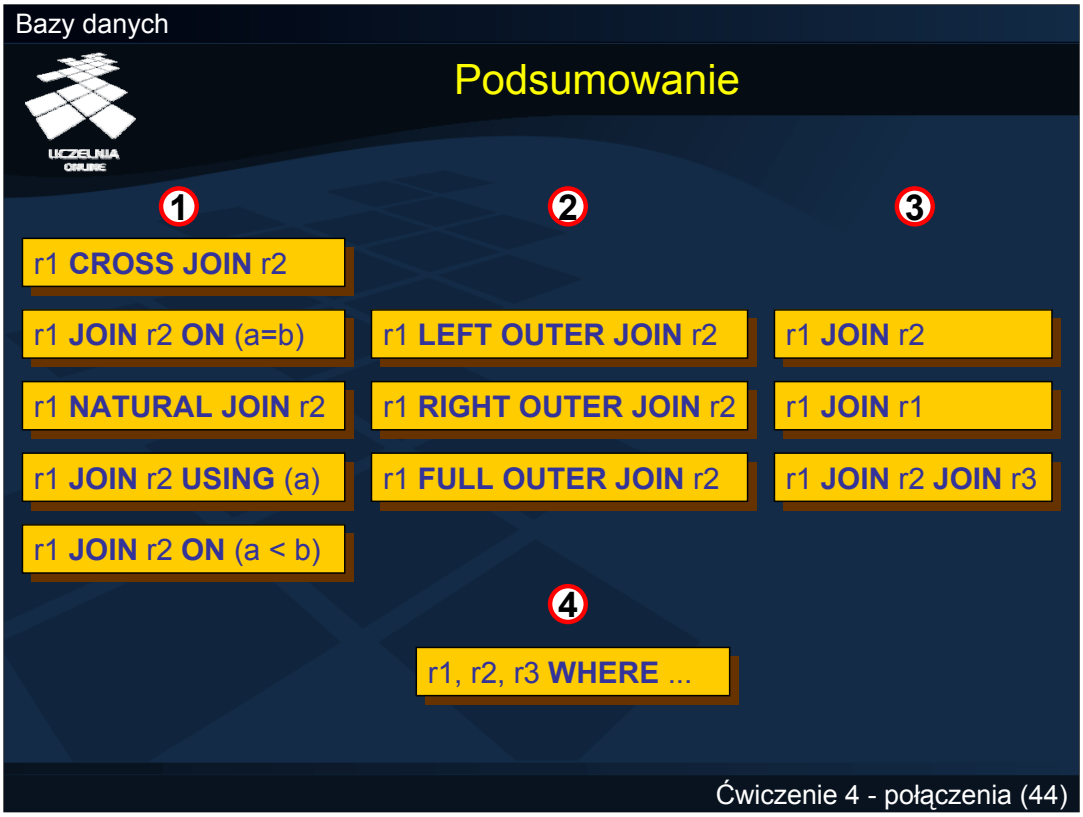
(16) Zmodyfikuj powyższe zlecenie w ten sposób, aby było możliwe wyświetlenie pracownika o nazwisku Marecki (który nie ma szefa).

(17) Dla każdego zespołu wyświetl liczbę zatrudnionych w nim pracowników i ich średnią płacę (z uwzględnieniem zespołów, na których nie zatrudniono żadnych pracowników).

Slajd przedstawia rozwiązania zadań: (18) i (19), których treść zacytowano poniżej.

(18) Dla każdego pracownika posiadającego podwładnych wyświetl ich liczbę. Wyniki posortuj zgodnie z malejącą liczbą podwładnych.

(19) Wyświetl nazwiska i daty zatrudnienia pracowników, którzy zostali zatrudnieni nie później niż 10 lat (3650 dni) po swoich przełożonych.



Na tym ćwiczeniu poznaliście państwo wiele różnych rodzajów połączeń. Poznane przez państwa połączenia można podzielić, ze względu na warunek połączeniowy, na: iloczyn kartezjański, połączenia równościowe (w tym naturalne) i nierównościowe (1). Każde z tych połączeń może być wewnętrzne, albo zewnętrzne, przy czym istnieją trzy rodzaje połączeń zewnętrznych (lewostronne, prawostronne i pełne) (2). Połączeniu mogą ulec dwie różne relacje, relacja sama z sobą (połączenie zwrotne), jak i dowolna kombinacja wielu relacji (3). Poznaliście również państwo starą notację połączeń, opartą o filtrowanie wyniku iloczynu kartezjańskiego (4).