

1 Spis treści:

1.1 Spis tabel:

1	SPIS TREŚCI:	1
1.1	SPIS TABEL:	1
1.2	SPIS TABEL	3
1.3	SPIS RYSUNKÓW:	4
2	MODELE BAZ DANYCH:	5
2.1	MODEL HIERARCHICZNY	5
2.2	MODEL SIECIOWY	5
2.3	MODEL RELACYJNY	6
3	SQL I ZARZĄDZANIE RELACYJNĄ BAZĄ DANYCH	7
3.1	PODSUMOWANIE:	8
4	PROJEKT:	10
4.1	DOBRY PROJEKT:	10
4.2	ZŁY PROJEKT MOŻE:	10
5	WPROWADZENIE DO BAZY	11
5.1	UŻYTKOWNICY MOGĄ STAWIAĆ PYTANIA:	11
5.2	POLITYKA WYDAWCY, DLA KTÓREGO ZOSTAŁA ZAPROJEKTOWANA BAZA JEST NASTĘPUJĄCA	11
6	PROJEKTOWANIE BAZY	12
6.1	ENCJA DANYCH I ICH ZWIĄZKI	12
6.1.1	<i>Wstępna lista encji</i>	12
6.2	KLUCZE GŁÓWNE	12
6.3	DEFINIOWANIE KLUCZA GŁÓWNEGO	13
6.4	ZWIĄZEK JEDEN DO WIELU	13
6.5	KLUCZE OBCE	14
6.5.1	<i>Związek wiele do wiele</i>	15
6.5.2	<i>Związek jeden do jeden</i>	15
6.6	PODSUMOWANIE	16
6.7	TABELA UAKTUALNIONA	17
7	PIERWSZA POSTAĆ NORMALNA	18
8	DRUGA POSTAĆ NORMALNA	19
8.1.1	<i>Przykład</i>	19
8.1.2	<i>Podsumowanie</i>	19
9	TRZECIA POSTAĆ NORMALNA	21
10	TABELE	22
10.1	MASKA WPROWADZANIA	22
10.2	RELACJE MIĘDZY TABELAMI	22
10.3	POŁĄCZENIA MIĘDZY TABELAMI:	23
11	ZASADY SKŁADNI SQL-A (HELP)	24
11.1	PRZYKŁADY:	24
12	TWORZENIE TABEL	25
12.1	PODSTAWOWA SKŁADNIA SQL	25
12.2	TWORZENIE TABELI – PROCES TWORZENIA	25
13	POLECENIE CREATE	26
13.1.1	<i>Uproszczona składnia polecenia CREATE TABLE</i>	26
13.1.2	<i>Rozszerzona składnia CREATE TABLE</i>	26
13.1.3	<i>Ograniczenia tabel</i>	26
13.1.4	<i>rozszerzona składnia SQL-92</i>	27
13.1.5	<i>Wstawianie - liczby całkowite</i>	27
13.1.6	<i>Wstawianie - Liczby zmiennoprzecinkowe</i>	27
13.1.7	<i>Wstawianie - kwotowe</i>	28

13.1.8	Wstawianie - znakowe typy danych.....	29
13.1.9	Wstawianie - Data	29
13.1.10	Wstawianie - logiczne.....	30
13.1.11	Tworzenie danych i wstawianie.....	30
14	MODYFIKACJA DANYCH	32
14.1	WSTAWIANIE DANYCH – INSERT INTO	32
14.1.1	Wstawianie danych do wszystkich kolumn.....	32
14.1.2	Wstawianie danych do wybranych kolumn.....	32
14.1.3	SELECT w poleceniu INSERT.....	34
14.2	ZMIANA ISTNIEJĄCYCH DANYCH – UPDATE.....	35
14.3	USUWANIE DANYCH Z TABELI	36
14.3.1	Usuwanie wszystkich danych – całej tabeli	36
14.3.2	Usuwanie wybranych danych	36
14.4	ZMIANA DEFINICJI TABEL	36
14.4.1	Wstawianie dodatkowej kolumny.....	36
14.4.2	Wstawianie INDEX-u.....	37
14.4.3	Usuwanie tabeli i indexu.....	37
14.4.4	Zmiana nazwy tabeli i kolumny.....	38
14.4.5	Podsumowanie.....	39
15	INSTRUKCJA SELECT	41
15.1	POBIERANIE DANYCH	41
15.1.1	Wszystkich wierszy i kolumn	41
15.1.2	Wszystkich wierszy i wybranej kolumny.....	41
15.1.3	Wszystkich wierszy i zestawu wybranych kolumn	41
15.1.4	Zmiana kolejności kolumn	41
15.1.5	Listy przecinkowe.....	41
15.1.6	Słowo kluczowe ALL.....	42
15.1.7	Słowo kluczowe DISTINCT.....	42
15.1.8	Zamiana nazwy kolumn	42
15.2	SORTOWANIE	43
15.2.1	Wyznaczanie kolumny sortującej przez nazwę.....	43
15.2.2	Wyznaczanie kolumny sortującej przez kolejność	43
15.2.3	Sposób sortowania ASC (rosnąco – ascending – A>Z).....	44
15.2.4	Sposób sortowania oraz DESC (malejąco – descending – Z<A).....	44
15.3	PODSUMOWANIE.....	45
15.3.1	Sortowanie na podstawie kilku kolumn.....	45
15.3.2	Sortowanie według kolumn nie występujących na liście SELECT.....	45
15.4	WYBIERANIE TABEL	46
15.5	WYBIERANIE WIERSZY : KLAUZULA WHERE	46
15.5.1	Operator porównania	47
15.5.2	Kombinacje lub logiczne negacje warunków.....	47
15.5.3	Przedziały	48
15.5.4	Listy (IN i NOT IN)	49
15.5.5	Dopasowanie napisów: LIKE.....	50
15.5.6	Porównywanie bez rozróżniania wielkości liter	50
15.6	OBLICZENIA ZE STAŁYMI.....	51
15.6.1	W klauzuli SELECT	52
15.6.2	W klauzuli WHERE.....	52
15.6.3	W klauzuli ORDER BY.....	53
16	GRUPOWANIE DANYCH	54
16.1	SKŁADNIA GROUP BY.....	54
17	FUNKCJE AGREGUJĄCE	55
17.1	SKŁADNIA FUNKCJI AGREGUJĄCEJ	55
17.1.2	Klauzula count(*) i count.....	56
17.1.3	Klauzula max i min	57
17.1.4	Klauzula AVG	58
18	PODZAPYTANIA.....	59
18.1	DZIAŁANIE PODZAPYTAŃ	59
18.1.1	Podział podzapytań.....	59
18.2	REGUŁY – PODZAPYTANIA	60
18.2.1	Podzapytanie ... IN.....	61
18.2.2	Podzapytanie – operator porównania ANY bądź ALL.....	66
18.3	PORÓWNANIE IN, ANY, ALL	69

18.3.1	Podzapytanie EXISTS	69
18.3.2	Definicja EXISTS	70
18.3.3	Zastosowanie EXISTS	71
19	ZŁĄCZENIA	73
19.1	SKŁADNIA	73
19.1.1	Złączenia a model relacyjny	73
19.2	WARUNKI WSTĘPNE ZŁĄCZENIA	73
19.2.1	UWAGI:	73
20	UNION.....	74
20.1	SKŁADNIA	74
21	ROZWIĄZANIE ZADAŃ ZALICZENIE NR 1.....	76
22	ROZWIĄZANIE ZADAŃ KOŃCOWYCH	84
22.1	PYTANIE I ROZWIĄZANIA.....	84
23	ADMINISTRACJA	86
23.1	TWORZENIE KOPII BEZPIECZEŃSTWA.....	86
23.1.1	PostgreSQL - Pg_dump	86
23.1.2	Linux - tar	86
23.1.3	Podsumowanie.....	86
23.2	TRANSAKCJE	87
23.3	TWORZENIE BAZY I UŻYTKOWNIKÓW.....	87
24	VISUAL BASIC.....	88
24.1	FORMULARZ W TRYBIE „TYLKO DO ODCZYTU”	88
24.2	WYSZUKIWANIE I FILTROWANIE REKORDÓW	88
24.2.1	Pole kombinowane do wyszukiwania rekordów.....	88
24.3	FILTROWANIE DANYCH	94
24.3.1	Utwórz grupę opcji	94
24.3.2	Utwórz procedurę zdarzenia uruchamiającą grupę opcji.....	97
25	PODSUMOWANIE.....	99
25.1	POSTACIE NORMALNE – OGÓLNE	99
25.1.1	Pierwsza.....	99
25.1.2	Druga.....	99
25.1.3	Trzecia	99
26	ELEMENTY TEORII RELACYJNYCH BAZ DANYCH.....	100
26.1	MODEL RELACYJNYCH BAZ DANYCH.....	100
26.1.1	Definicja 1. Schemat relacji.....	100
26.1.2	Definicja 3. Relacje.....	100
26.2	ZASADY DOTYCZĄCE STRUKTURY DANYCH	100
26.2.1	Postulat informacyjny.....	100
26.2.2	Postulat dostępu.....	100
26.2.3	Postula fizycznej niezależności danych.....	100
26.2.4	Postulat logicznej niezależności danych.....	100
26.2.5	Postulat niezależności dystrybucyjnej.....	100
26.2.6	Postulat zabezpieczania przed operacjami na niższym poziomie abstrakcji.....	101
26.3	ZASADY DOTYCZĄCE POBIERANIA I MODYFIKOWANIA DANYCH	101
26.4	PIERWSZA POSTAĆ NORMALNA	101
26.5	DRUGA POSTAĆ NORMALNA.....	102
26.6	TRZECIA POSTAĆ NORMALNA.....	102
26.7	TABELA.....	102
26.7.1	Tworzenie.....	102
26.7.2	Wstawianie.....	102
26.7.3	Modyfikacja	102
26.7.4	Usuwanie wszystkich danych – całej tabeli	102
26.7.5	Usuwanie wybranych danych	103
26.7.6	Wstawianie dodatkowej kolumny.....	103

1.2 Spis tabel

TABELA 1 TYTUŁY – ROZWIĄZANIE WERSJA 1.....	14
TABELA 2 WYDAWCY – ROZWIĄZANIE WERSJA 1.....	14

TABELA 3 TYTUŁY – ROZWIĄZANIE NR 2	14
TABELA 4 WYDAWCY – ROZWIĄZANIE NR 2.....	14
TABELA 5 TABEL ŁĄCZĄCA ZWIĄZKI WIELE DO WIELU	15
TABELA 6 SUMA RELACJI	101
TABELA 7 PRZECIĘCIE RELACJI	101
TABELA 8 DOPEŁNIENIE RELACJI	101

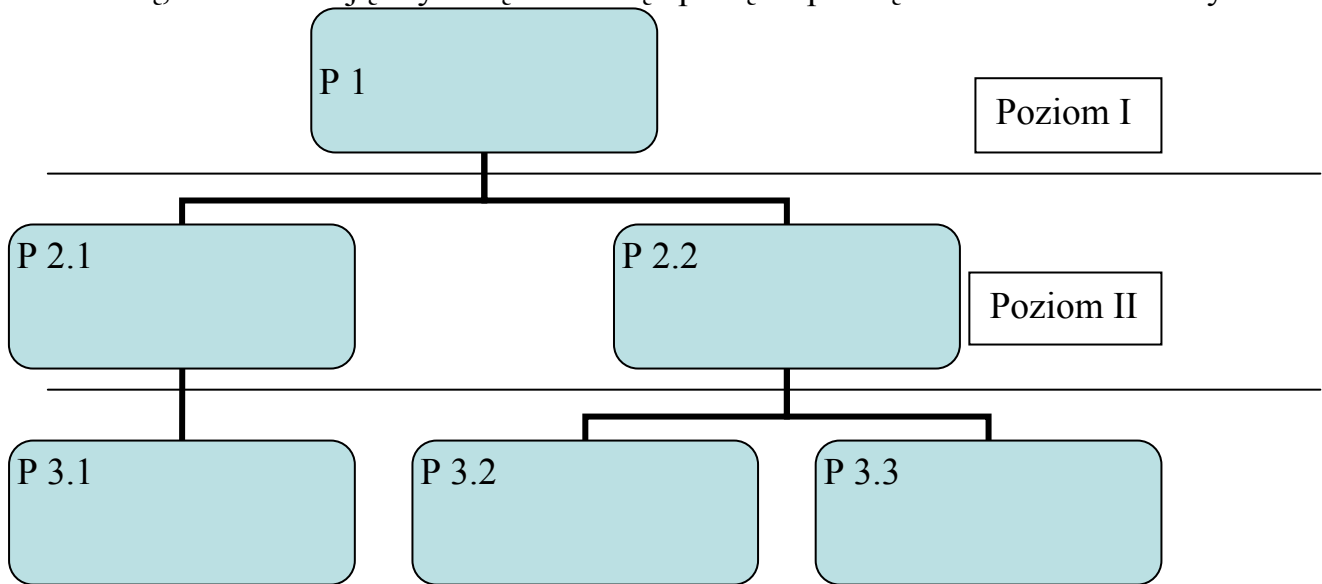
1.3 Spis rysunków:

RYSUNEK 1 SCHEMAT MODELU HIERARCHICZNEGO	5
RYSUNEK 2 SCHEMAT MODEL SIECIOWY	5
RYSUNEK 3 SZKIC TABEL BAZY WYDAWNICTWO	12
RYSUNEK 4 WSTĘPNY SZKIC BAZY DANYCH KSIĘGARNIA	12
RYSUNEK 5 ROZBICIE NAZWISKO_AUTORA NA NAZWISKO_AUTORA I IMIE_AUTORA	13
RYSUNEK 6 ROZBICIE NAZWISK NA NAZWISKO I IMIĘ ORAZ UTWORZENIE KLUCZY GŁÓWNYCH.....	13
RYSUNEK 7 ROZWIĄZANIE NR 1 – BEZ KLUCZA OBCEGO	13
RYSUNEK 8 ROZWIĄZANIE NR 2 - DODANIE KLUCZA OBCEGO ID_WYDAWCY DO TABELI TYTUŁY	14
RYSUNEK 9 DRUGA POSTAĆ NORMALNA- KONIEC	20

2 Modele baz danych

2.1 Model hierarchiczny

Zdarza się, że dane mają czytelną strukturę opartą na powiązaniach hierarchicznych



Rysunek 1 Schemat modelu hierarchicznego

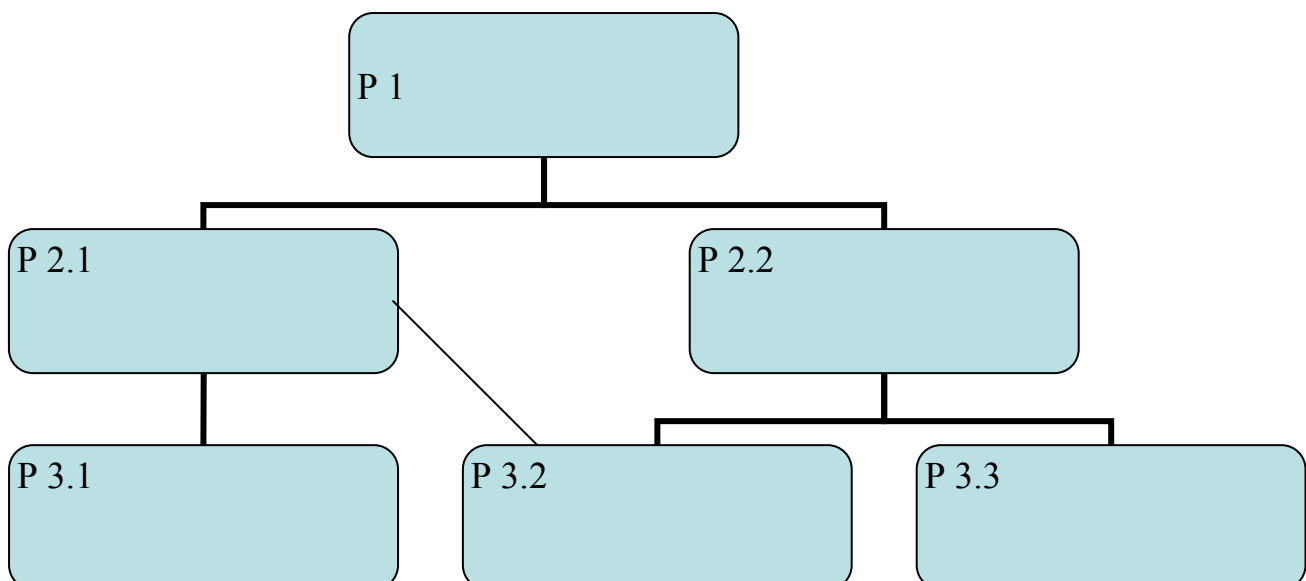
Na czym polega istota takiego podziału. Dane z określonego poziomu mają odniesienie do jednego zbioru nadrzędnego. Mogą natomiast generować odniesienia do wielu zbiorów niższego rzędu. Przykładem takiej struktury jest lokalizacja plików w pamięci komputera.

Można powiedzieć, że jedną z ważnych funkcji systemu operacyjnego jest zarządzanie hierarchicznym modelem bazy danych, w których danymi są zasoby komputera.

Podaj przykłady : Szkoła -> klasa -> uczeń struktura firmy dyrektor...

2.2 Model sieciowy

Okazuje się, że zachowanie hierarchicznej zależności jest poważną przeszkodą, ponieważ rzeczywistość z reguły jest bardziej skomplikowana.. Jeśli zbudujesz choćby jedno dodatkowe powiązanie, które zakłuci strukturę hierarchiczną, otrzymasz model sieciowy



Rysunek 2 Schemat model sieciowy

Może się zdarzyć, jeśli zdecydujesz na przykład, że jeden z pracowników podlega w różnych sprawach kilku innym pracownikom, także z różnych poziomów hierarchii. Ła-

two zauważyć, że model sieciowy może przyjmować różne postaci. Trudno zatem zbudować wokół niego przejrzystą i w miarę prostą teorię.

2.3 Model relacyjny

Model ten jest najczęściej wykorzystywany.

Jego popularność wynika z wielu przyczyn:

Model ten ma bardzo dobrą podbudowę teoretyczną opartą na teorii mnogości, którą zajmuje się matematyka

Bardzo dobrze opisuje praktyczne problemy, które musi rozwiązać SZBD

Dane w relacyjnym modelu bazy danych przechowuje w tabelach, które są bardzo popularną strukturą, znaną z np. arkusza kalkulacyjnego

Dane dotyczące tabel:

Pierwszy, wyróżniony wiersz tabeli nazywa się nagłówkiem – identyfikuje on dane zapisane w kolumnach

Zapisy kolejnych wierszy nazywamy rekordami (krotkami); rekord odpowiada na pytanie, w jakiej relacji z nagłówkiem jest opisywany fragment rzeczywistości

Dane zapisane w jednej kolumnie nazywamy polami rekordów – wszystkie mają te same właściwości

Każda kolumna (pole rekordu) opisuje inny zestaw właściwości

3 SQL i zarządzanie relacyjną bazą danych

SQL to język dla relacyjnych baz danych.

Wszystkie systemy zarządzania bazami danych zapamiętują i operują nimi. Podejście relacyjne do zarządzania bazami danych jest oparte na modelu matematycznym, który zawiera tak groźnie brzmiące pojęcia, jak algebra relacyjna i rachunek relacyjny.

C.J Date podaje nieformalną definicję systemu zarządzania relacyjną bazą danych (DBSM; database management system):

System reprezentuje całą zawartość w bazie danych informację w postaci tabel

Wspiera trzy operacje relacyjne, zwane jako

wybór (selekcja; selection),

rzutowanie (projekcja; projection) i

złączanie (join), w celu umożliwienia dokładnego określenia danych, jakie chciałoby się

obejrzeć (przy czym operacje te mogą być wykonywane bez konieczności fizycznego przechowywania danych przez system w jakiś określony sposób)

E. F. Codd, twórca modelu relacyjnego, podał szczegółową listę warunków, jakie musi spełniać model relacyjny. Reasumując, aby można było powiedzieć o systemie zarządzania relacyjną bazą danych, że jest w pełni relacyjna, musi:

Reprezentować całą zawartość w bazie danych informację w postaci tabel

Utrzymywać niezależność logicznej reprezentacji danych od fizycznego sposobu ich przechowywania

Używać języka wysokiego poziomu w celu strukturalizacja, wyszukiwania i zmiany informacji w bazie danych (teoretycznie powinno być wiele języków spełniających ten wymóg, w praktyce takim językiem jest SQL)

Wspierać podstawowe operacje relacyjne (wybór, rzutowanie, złączanie) i takie operacje teorii zbiorów, jak suma, przecięcie, różnica i dzielenie

Wspierać perspektywy, które umożliwiają użytkownikom określenie alternatywnych sposobów oglądania danych tabel

Dostarczać metodę pozwalającą odróżnić wartości nieznane (null) od zera lub od miejsca niewypełnionego

Wspierać mechanizmy zapewniające spójność, prawa dostępu, transakcje i odzyskiwanie danych

Zarówno algebra jak i rachunek posługują się kilkoma operacjami podstawowymi:

łączenie (union) – w operacji tej porównywane są tylko te relacje, które zawierają identyczną grupę atrybutów, zaś rezultatem działania jest powstanie nowej relacji, w której znajduje się każde wystąpienie bytu z jednej lub drugiej relacji

przecięcie (intersect) – w operacji tej, w wyniku porównania powstaje nowa relacja, zawierająca wystąpienie bytu w obu porównywanych relacjach

różnica (difference) – operacja ta nie jest przemienne, zaś w jej wyniku powstaje nowa relacja, zawierająca wszystkie wystąpienia bytu z pierwszej relacji, które nie wystąpiły w drugiej relacji

iloczyn kartezjański (Cartesian product) – w wyniku takiej operacji powstaje relacja zawierająca wszystkie możliwe połączenia atrybutów i wystąpienia bytów z obu pierwotnych relacji

ograniczenie (restriction) – operacja ta jest możliwa wyłącznie na relacji i jest to wybór wszystkich atrybutów danej relacji oraz tych wystąpień bytu, które spełniają warunki restrykcji

projekt (project) – jest to wybór wszystkich wystąpień bytu i tylko tych jego atrybutów, które spełniają wymogi projektu

scalanie (join) jest to operacja tworzenia nowej relacji z kilku już istniejących, przy użyciu jednego lub kilku atrybutów i wspólnych wystąpień bytów

podział (divide) operacja ta ma na celu wydzielenie z podstawowych relacji (dzielnej) takiej relacji, w której zawarte są wszystkie atrybuty podstawowej relacji oraz takie wystąpienia bytu, które dla atrybutów relacji dzielnika również występowały

porównanie (compare) – operacja porównania występowania jednego bytu z drugim lub z określoną wartością liczbową, znakową bądź wyrażeniem.

3.1 Podsumowanie:

Pierwotnym pojęciem teorii mnogości jest zbiór i przynależność elementu do zbioru. Pojęcia te są transportowane na model byt-relacja. Taki model jest często stosowany przy przenoszeniu zdarzeń rzeczywistych i systemów działających w realnym świecie na system oparty na relacyjnej bazie danych.

W modelu relacyjnym, na którym oparta jest również baza danych Postgres, wszystkie elementy zbioru są klasyfikowane jako **byty**, którymi mogą być osoby lub rzeczy, lub jako relacje, czyli powiązania pomiędzy obiektami. *Zarówno byt, jak i relacje są reprezentowane w bazie danych przez strukturę nazywaną tablicą.*

Tablica służy do przechowywania danych i jest strukturą, która może przedstawiać pojedynczy byt. Cały system jest złożony z pojedynczych bytów. W tablicy może być ukazany związek pomiędzy dwoma bytami, przedstawiany przez klucze obce. Poprawnie zbudowane tablice powinny przedstawiać albo byt, albo relacje, chociaż możliwe jest przedstawienie bytu i relacji w jednej tablicy. Każda tablica zawiera kolumny i wiersze.

Kolumna tablicy przedstawia jeden i tylko jeden atrybut bytu. Z reguły nazwa kolumny sygnalizuje jej znaczenie.

Klucz główny jest konstrukcją używaną do jednoznacznego określenia każdego wiersza w tablicy i zawiera jedną lub więcej kolumn. Zwykle klucz główny jest określany w momencie tworzenia tabeli i chociaż jest to zabronione, to regułą jest tworzenie jednego klucza głównego dla jednej tablicy.

Klucze obce przedstawiają związki pomiędzy tabelami. Są to kolumny lub grupy kolumn, których wartości pochodzą z klucza głównego innej tablicy. Jedna tablica może zawierać wiele kluczy obcych.

Oprócz kluczy głównych i obcych, w bazie relacyjnej mogą istnieć **klucze unikatowe**, które mają wszystkie cechy klucza głównego, jednakże pełnią inną funkcję. Zasadniczym zadaniem klucza unikatowego jest zapewnienie niepowtarzalnych wierszy w tablicy. Klucz unikatowy nie może zawierać wartości pustych. W efekcie stosowania kluczy w bazie relacyjnej osiąga się spójność danych. Spójność relacyjna zapewnia utrzymanie we właściwej postaci związków relacyjnych reprezentowanych przez klucze główne i obce.

System zarządzania bazą danych to zestaw programów umożliwiający skonstruowanie bazy danych i aplikacji, które z nich korzystają czyli obejmuje:

Tworzenie baz danych

Funkcje tworzenia zapytań i aktualizacji danych

Wielozadaniowość

Utrzymanie dzienników systemowych

Zarządzanie bezpieczeństwem bazy danych
Utrzymanie integralności odwołań

4 Projekt:

4.1 Dobry projekt:

Umożliwia lepsze zrozumienie współpracy z bazą danych

Gwarantuje spójność bazy danych

Toruje drogę do najwyższej efektywności, jaką system może osiągnąć

4.2 Zły projekt może:

Ułatwić niewłaściwą interpretację wyników zapytań

Zwiększać ryzyko wprowadzania niespójności do danych

Wymuszać wprowadzanie nadmiarowych danych

Utrudniać życie, jeśli jest konieczna zmiana struktury zbudowanych baz i wypełnionych już danych tabel

5 Wprowadzenie do bazy

5.1 Użytkownicy mogą stawiać pytania:

Którzy autorzy mieszkają w Warszawie

Które książki dotyczące biznesu kosztują powyżej 50 zł

Kto napisał największą liczbę książek

Jaka jest średnica zaliczka wypłacana za wszystkie książki informatyczne

Jaką sumę jesteśmy winni autorowi książki Turbo C++

Jak kształtuje się sprzedaż w komputerowej filii wydawcy

5.2 Polityka wydawcy, dla którego została zaprojektowana baza jest następująca

Jeden autor może napisać więcej niż jedną książkę

Książka może być wynikiem współpracy większej liczby autorów

Porządek nazwisk autorów na stronie tytułowej jest istotną informacją, gdyż wyznacza procentową wielkość honorarium autorskiego, jakie każdy z nich otrzymuje

Redaktor może opracować więcej niż jedną książkę, a jedna książka może być przypisana do wielu redaktorów

Zamówienie może być wystawione na jeden lub wiele tytułów

6 Projektowanie bazy

6.1 Encja danych i ich związki

Na poziomie podstawowym modelowanie związków encji (zwane czasem modelowaniem encji) oznacza zidentyfikowanie :

rzeczy – encji –o których będą gromadzone dane w systemie bazy danych

Właściwości tych rzeczy

Związków między nimi

6.1.1 Wstępna lista encji

Autorzy, którzy napisali książki wydane przez firmę

Same książki

Redaktorzy, którzy pracują dla firmy

Filie będące własnością firmy

Każdy z tych typów ma następujące właściwości:

Tytuł książki

Cena książki

Data publikacji książki

Całe nazwisko autora

Adres autora

Numer telefonu autora

Całe nazwisko redaktora

Adres redaktora

Numer telefonu redaktora

Nazwa wydawcy

Adres wydawcy



Rysunek 3 Szkic tabel bazy wydawnictwo

Każdy wiersz w tabeli reprezentuje występowanie (lub inaczej instancję) encji- pojedynczą książkę, autora, redaktora lub wydawcę.

Jednym z zadań przy projektowaniu bazy danych jest przewidzenie sposobu rozróżniania wystąpień encji, tak aby system mógł przetwarzać pojedynczy wiersz.

6.2 Klucze główne

Co jest kluczem głównym dla każdej z tabel?

Kandydat na klucz główny: **Nazwisko_autora**

Dlaczego zły:

kombinacja imienia i nazwiska

Możliwość powtórzenia Nazwiska i imienia przy dużej bazie i brak możliwości zidentyfikowania autora

Wnioski: podział encji Nazwisko_autora na Nazwisko i imię



Rysunek 4 Wstępny szkic bazy danych księgarnia

Mogło by się wydawać, że wybór dwóch kolumn Imie i nazwisko jako kandydaci na klucz główny jest najbardziej odpowiedni, a nawet świetny...aż do czasu, kiedy tabela będzie tak duża, że całe nazwiska będą się powtarzać, czyli nie będzie można jednoznacznie zidentyfikować odpowiednią encją np. danego autora..



Rysunek 5 Rozbicie nazwisko_autora na Nazwisko_autora i Imie_autora

6.3 Definiowanie klucza głównego

Wnioski: ustaliliśmy strukturę czterech tabel: Autorzy, Redaktorzy, Tytuły, Wydawcy oraz klucze główne.



Rysunek 6 Rozbicie nazwisk na nazwisko i imię oraz utworzenie kluczy głównych

W naszym życiu codziennym, czyli świecie rzeczywistym też tak postępujemy. Przykładem tego są numery ubezpieczenia społecznego, numery identyfikacyjne pracowników, numery na tablicach rejestracyjnych pojazdów, numery faktur, numery lotów ... Należy zaznaczyć że wybór i przyjęcie kolumny (kolumn), która ma służyć jako klucz główny tabeli, jest jednym z najważniejszych kroków w projektowaniu bazy danych. Mimo tak istotnego znaczenia kluczy głównych, pierwsze wersje SQL-a nie miały składni umożliwiającej ich zaplanowanie.. W standardzie z 1992 roku dla SQL-a dopiero jest klauzula Primary Key w instrukcji CREATE TABLE.

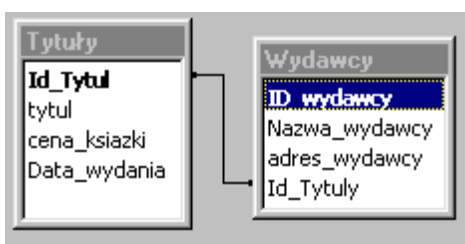
Mamy już więc w bazie danych strukturę czterech tabel i określiliśmy też pene własności każdej encji opisanej tabelą, a także ustaliliśmy klucze główne.

6.4 Związek jeden do wielu

Problem do rozwiązania: związki między danymi. Żadna tabela nie informuje, jaki jest związek np. między określonym wydawcą a wydawanymi przez niego książkami

Problem „Jeden do wiele”: Każda książka ma jednego wydawcę, podczas gdy jeden wydawca może wydać wiele książek.

Rysunek 7 Rozwiązanie nr 1 – bez klucza obcego



dodanie kolumny Id_tytuly w tabeli Wydawcy
ID_Tytuly w tabeli Wydawcy jest kluczem obcym

Niestety proponowane rozwiązanie prowadzi w złym kierunku, ponieważ ma być jeden wydawca wiele książek. Rozważmy co się stanie, gdy zostanie opublikowana nowa książka:

Tabela 1 Tytuły – rozwiązanie wersja 1

Id_tytul	Tytul	Cena_ksiazki	Data_wydania
0001	C++	120	2001-09-28
0002	TP	100	2001-09-29

Tabela 2 Wydawcy – rozwiązanie wersja 1

Id_wydawcy	Nazwa_wydawcy	Adres_wydawcy	ID_tytul
0001a	PWN	Warszawa	0001
0001a	PWN	Warszawa	0002

Uwagi: jednym z celów przy projektowaniu bazy danych jest unikanie redundancji, gdyż może być źródłem błędów.

Rysunek 8 Rozwiązanie nr 2 - Dodanie klucza obcego Id_wydawcy do tabeli Tytuły

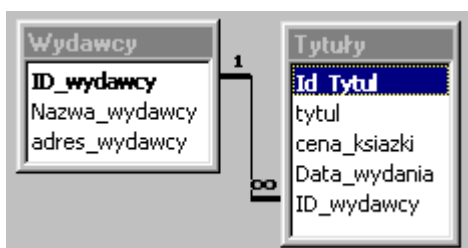


Tabela 3 Tytuły – rozwiązanie nr 2

Id_tytul	Tytul	Cena_ksiazki	Data_wydania	ID_wydawcy
0001	C++	120	2001-09-28	0001a
0002	TP	100	2001-09-29	0001a

Tabela 4 Wydawcy – rozwiązanie nr 2

Id_wydawcy	Nazwa_wydawcy	Adres_wydawcy
0001a	PWN	Warszawa
0001b	HELION	Kraków

Wraz z tymi zmianami wyłania się nowa struktura:

Tabela **Wydawcy** ma jeden wiersz dla każdego wydawcy

Tabela **Tytuły** ma jeden wiersz dla każdej książki

Identyfikatory **Wydawców** powtarzają się w tabeli **Tytuły**, gdyż każdy wydawca wydaje wiele książek: jest to jednak dużo mniejsza redundancja niż jakakolwiek inna, z którą mielibyśmy do czynienia przy każdym innym rozwiązaniu.

6.5 Klucze obce

Pojęcie klucza obcego w projektowaniu baz danych, podobnie jak pojęcie klucza głównego, ma zasadnicze znaczenie. Nieformalnie klucz obcy jest kolumną (lub kombinacją kolumn) w jednej tabeli, której wartości są zgodne z wartościami klucza głównego w jakiejś innej tabeli.

Należy zaznaczyć, że baza danych powinna obejmować zaplanowanie zgodności między kluczem głównym i kluczem obcym (zapewnić spójność odwołań).

Pełne projektowanie w bazie danych powinno obejmować zaplanowanie zgodności między kluczem głównym i kluczem obcym

Oto przykład:

Kiedy identyfikator wydawcy zostaje zaktualizowany lub usunięty z tabeli wydawcy, wtedy system powinien automatycznie powtórzyć tę zmianę w tabeli tytuły – albo przez aktualizację wszystkich odpowiednich wartości w kolumnie id_wydawcy tabeli tytuły albo przez kaskadowe (począwszy od tabeli wydawcy) usuwanie wierszy tabeli tytuły z odpowiadającymi im wartościami id_wydawcy

Kiedy zostanie dodany nowy tytuł, wtedy system powinien mieć możliwość sprawdzenia, czy powiązany z nim numer Id_wydawcy jest poprawny (tzn. czy istnieje w tabeli Wydawcy)

6.5.1 Związek wiele do wiele

Następnym krokiem, po zidentyfikowaniu w bazie danych wszystkich związków jeden do wielu i po związaniu z nimi par klucz główny – klucz obcy, jest rozważenie innych rodzajów związków danych.

Niektóre książki są napisane przez kilku autorów, a niektórzy autorzy napisali kilka książek. Innymi słowy, między autorami i książkami zachodzi związek **wiele do wiele** czasami zwany **asocjacja**.

Zgodnie z teorią związków encji asocjacje w relacyjnej bazie danych są reprezentowane przez tabele same w sobie. Mówiąc inaczej, baza danych potrzebuje tabeli dla autorów, tabeli dla tytułów oraz tabeli reprezentującej związek między nimi..

Tabela 5 Tabel łącząca związki wiele do wielu

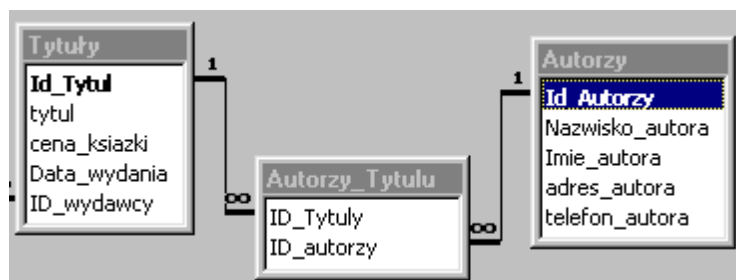
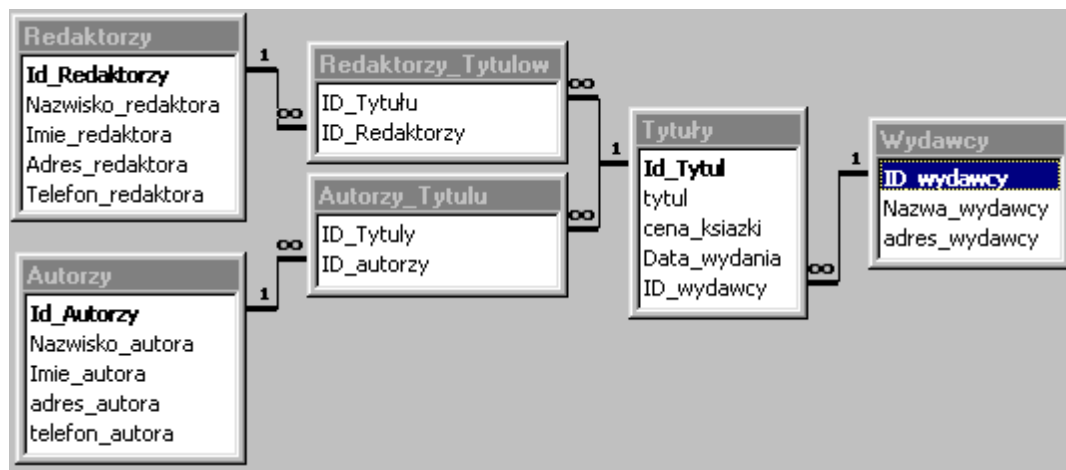


Tabela **Autorzy_Tytulu** jest tabelą bazową związku wiele do wiele czyli obrazuje asocjację, a nie niezależną encję.

Ogólna zasada sformułowana przez **C. J. Date’a** mówi, że „w modelu relacyjnym elementy asocjacji są identyfikowane przez klucze obce w tabeli reprezentującej tę asocjację”.



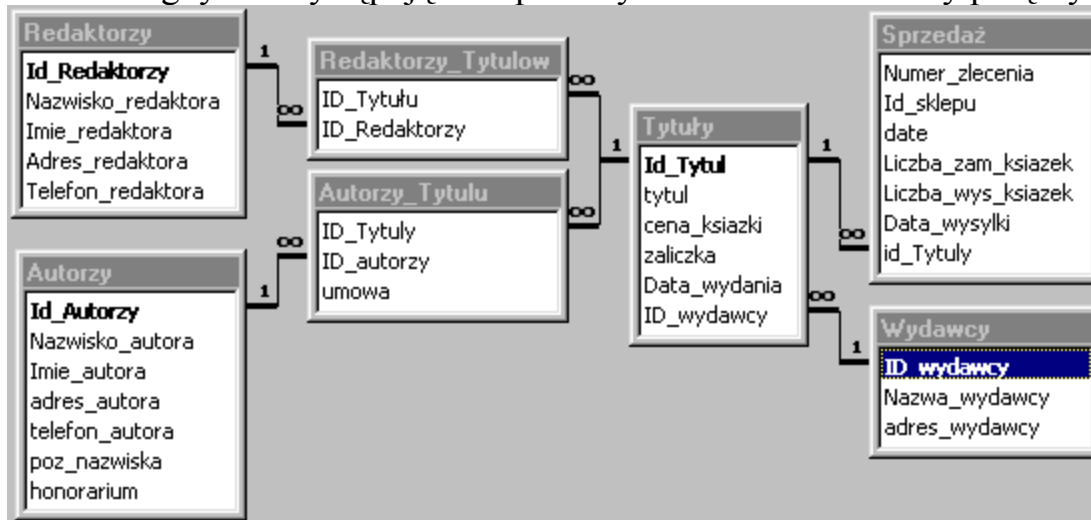
6.5.2 Związek jeden do jeden

Powodem ww. związku są:

Szybkość wykonywania zapytań

Ukrycie danych bardzo ważnych

Wnioski: gdy nie występują ww. powody to dwie tabele należy połączyć w jedną.



6.6 Podsumowanie

Przedstawienie każdej niezależnej encji (książki, autora, wydawcy, redaktora, pracownika, wydziału, studenta, wykładu itp.) jako tabeli bazowe.

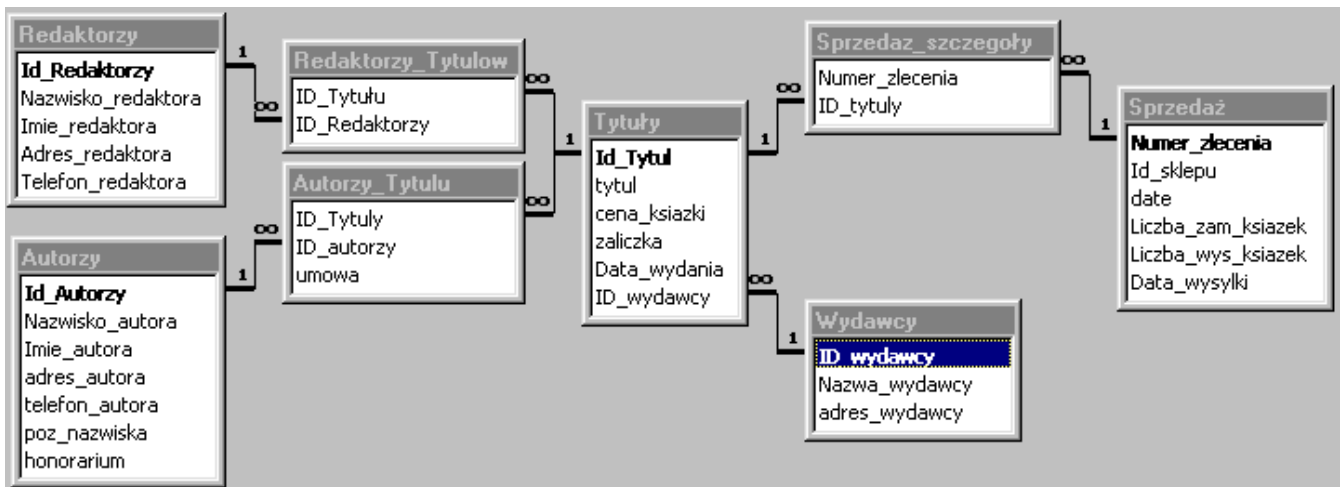
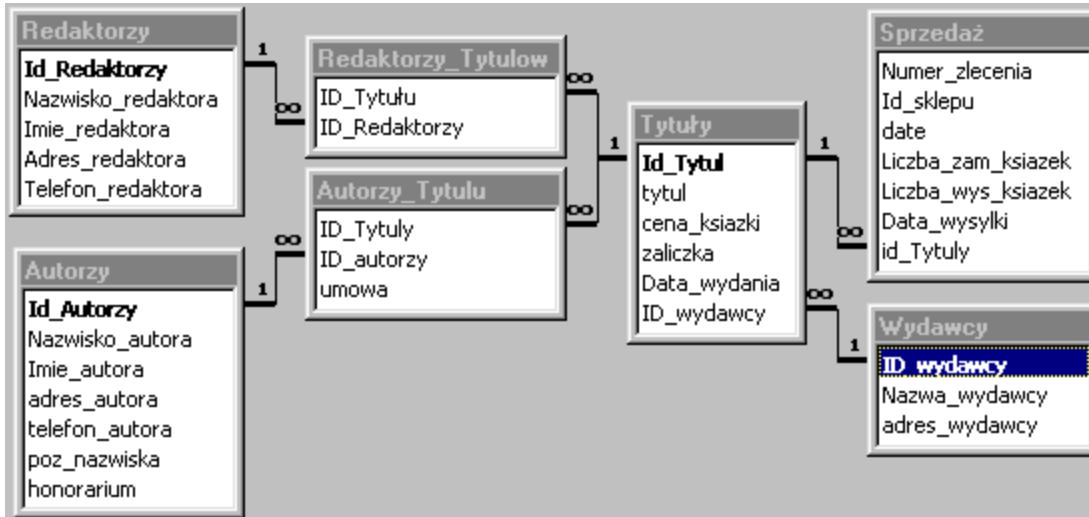
Przedstawienie każdej właściwości encji (adresu autora, ceny książki itp.) jako kolumny w tabeli encji.

Sprawdzenie czy każda tabela ma klucz główny. Kluczem może być istniejąca właściwość (nazwisko) lub dodana przez Ciebie cecha sztuczna (np. numer ubezpieczenia społecznego czy numer porządkowy), lub kombinacja dwóch bądź więcej właściwości. Zawsze musisz każdy wiersz opisać jednoznacznie.

Zlokalizowanie związku jeden do wiele między tabelami. Sprawdzenie, czy istnieje kolumna (kolumny) z kluczem obcym po stronie „wiele”, wskazującym na kolumnę (kolumny) klucza głównego w tabeli po stronie „jeden”. Przeanalizowanie więzów spójności odwołań skojarzonych z każdym kluczem.

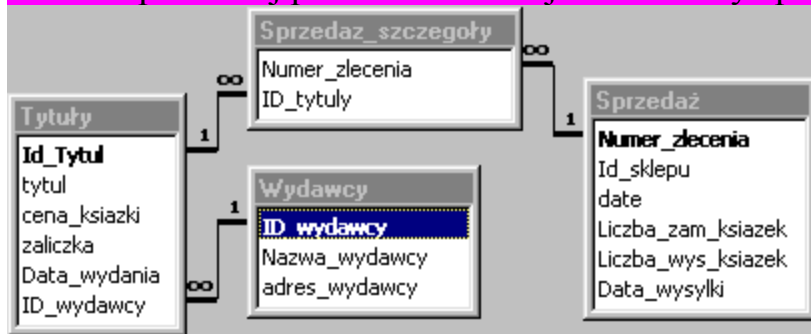
Przedstawienie każdego związku wiele do wiele (asocjacji) jako „tabeli łączącej” dwie tabele występujące w asocjacji. Umieszczenie w tej tabeli łączącej kluczy obcych wskazujących na tabele encji. Klucz główny tabeli łączącej jest często kombinacją tych kluczy obcych.

6.7 Tabela uaktualniona



7 Pierwsza postać normalna

Pierwsza postać normalna wymaga, aby na każdym przecięciu wiersza i kolumny występowała tylko jedna wartość i aby wartość ta była atomowa: w tabeli spełniającej warunki pierwszej postaci normalnej nie może być powtarzających się grup danych.



Jak potraktować pojedyncze zlecenie sprzedaży, uwzględniające wiele książek. Nie możesz zapamiętać wielu identyfikatorów id_tytul w jednym polu. Naruszałoby to pierwszą postać normalną. Dodanie kolumn Id_tytul1, Id_tytul2... to po prostu ukrycie powtarzających się grup.

Wniosek: Tam gdzie występują powtarzające się kolumny, poprawny projekt wymaga wprowadzenia **tabeli nadrzędnej (Sprzedaż)** dla zleceń sprzedaży jako całości i **tabeli szczegółowej (Sprzedaz_szczegóły)** do przechowywania informacji związanych z poszczególnymi wierszami w zleceniu sprzedaży.

Zauważ, że zasady związków encji doprowadziły by do tych samych wniosków, gdyż jest to struktura jeden do wiele (jedno zlecenie sprzedaży, wiele wierszy)



Gdy rozpatrujemy powtarzające pola, rozważyć należy podzielenie wszystkich złożonych kolumn na ich składowe: np. adres powinien mieć osobne kolumny

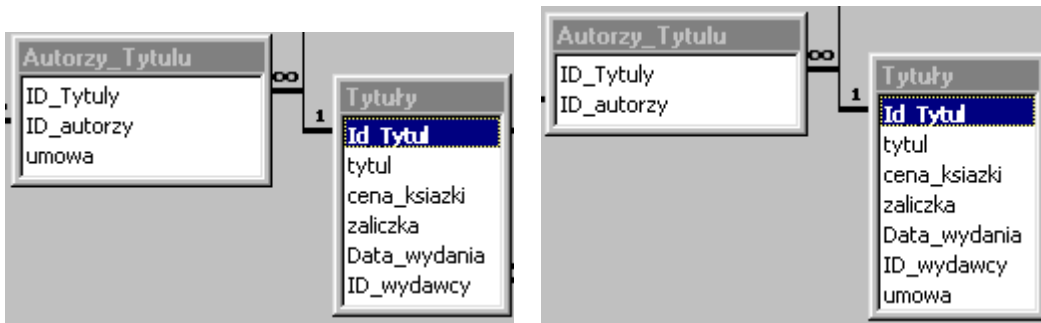
Miasto

Kod pocztowy

Ul.

województwo

8 Druga postać normalna



Druga reguła normalizacji mówi, że każda kolumna nie należąca do klucza (niekluczowa) musi zależeć od całego klucza głównego. Wynika stąd, że tabela nie może zawierać kolumny niekluczowej odnoszącej się tylko do jakiejś części złożonego klucza głównego.

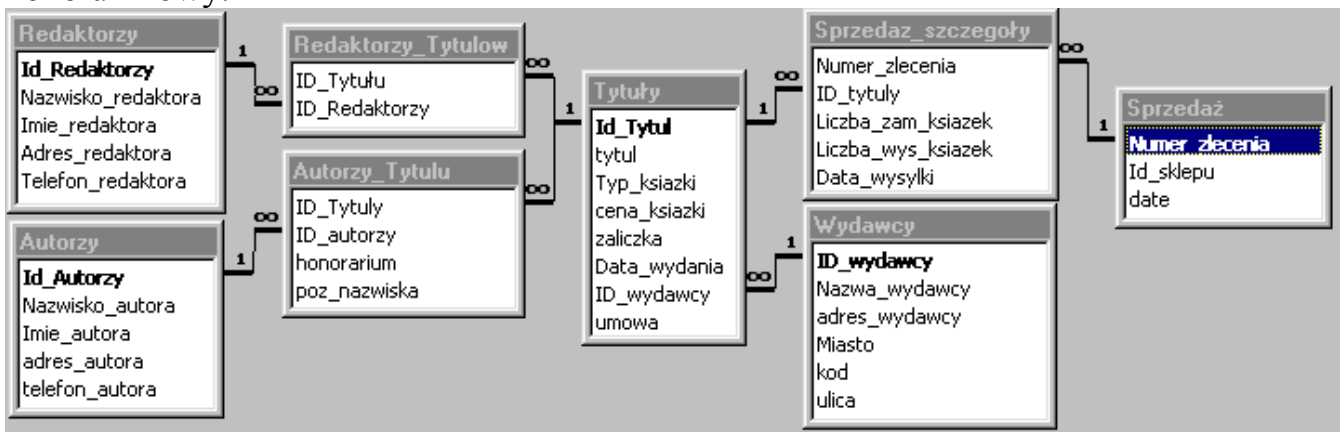
Sprowadzenie tabeli do drugiej postaci normalnej wymaga upewnienia się, że wszystkie kolumny nie będące częściami składowymi klucza głównego (kolumny informujące o przedmiocie, ale nie definiujące go jeszcze jednoznacznie) odnoszą się do całego klucza głównego, a nie tylko jednej z jego składowych.

8.1.1 Przykład

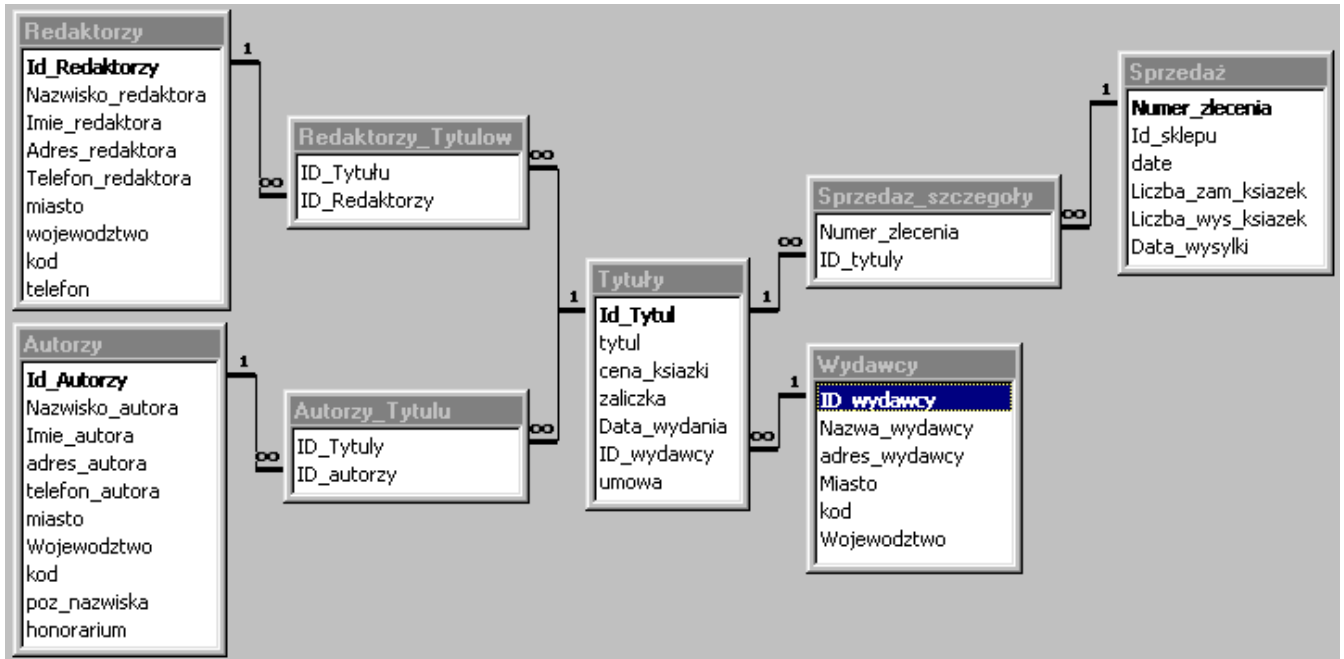
Rozpatrzmy kolumnę **umowa** z tabeli **Autorzy_tytulu**. Czy odnosi się ona do każdej kombinacji **autor_tytulu**? Jeśli każdy autor książki ma oddzielną umowę, to tak; jeśli jednak umowę w imieniu autorów podpisała firma, to nie. W tym wypadku dział prawny poinformuje Cię, że umowa jest związana z książką, a nie z każdym indywidualnym autorem, a Ty przeniesiesz tę kolumnę do tabeli **Tytuły**.

8.1.2 Podsumowanie

Druga postać normalna wymaga, aby kolumna nie będąca kluczem (nie kluczowa) nie była podzbiorem klucza głównego. Odnosi się to do sytuacji, kiedy klucz główny jest utworzony z więcej niż jednej kolumny, a nie do sytuacji, kiedy klucz główny jest jednokolumnowy.



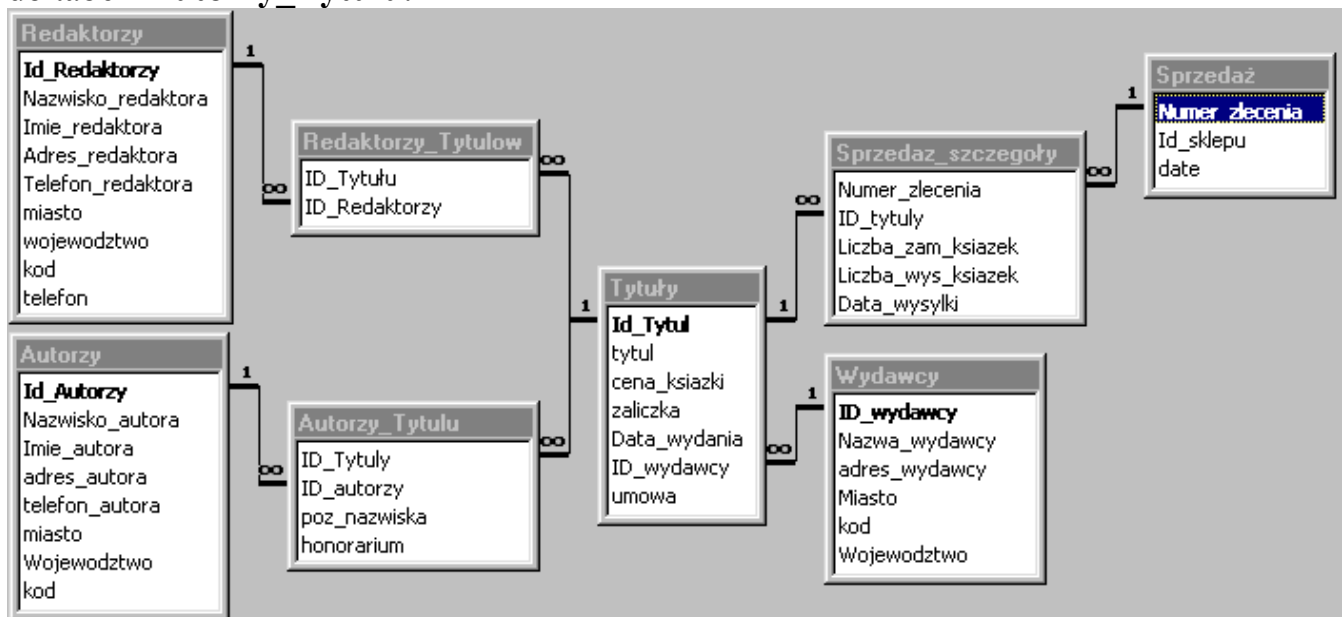
Rysunek 9 Druga postać normalna- koniec



9 Trzecia postać normalna

Trzecia postać normalna jest związana z zasadą wyrażoną przez drugą postać normalną w bardziej ogólny sposób: nie jest ona ograniczona tylko do złożonych kluczy głównych. Trzecia postać normalna wymaga, **żeby żadna kolumna niekluczowa nie zależała od innej kolumny niekluczowej**. Każda kolumna nie będąca kluczem musi być związana z kolumną klucza głównego.

W tabeli **Autorzy** kluczem głównym jest **Id_Autorzy**. Gdy zaznaczysz każdą kolumnę, stwierdzisz, że **poz_nazwisko** (pozycja nazwiska autora wśród nazwisk innych współautorów dane książki) nie dotyczy określonego autora (Id_Autorzy), gdyż autor może być współautorem wielu książek i w każdym przypadku może zajmować inną pozycję (może być pierwszym, drugim, trzecim autorem). Informacja o autorze w rzeczywistości dotyczy **autor-tytuł**. Podobne jest z honorarium autora. (obie te kolumny należą do tabeli **Autorzy_Tytułu**).



Kolumny **Liczba_zam_ksiazek**, **Liczba_wys_ksiazek**, w tabeli **Sprzedaż** również ilustrują tę zasadę. Dotyczą one pojedynczych pozycji, a nie całego zlecenia sprzedaży, i powinny zostać przeniesione do tabeli **Sprzedaż_Szczegóły**. Bardzo zagadkowa jest kolumna **Data_wysylki**.

Jeśli zlecenia są realizowane tylko wtedy, gdy wszystkie tytuły ze zlecenia są dostępne, to **data_wysylki** odnosi się do zlecenia jako całości i wówczas ta kolumna powinna przejść do tabeli **sprzedaż**

Jeśli zlecenie jest realizowane w miarę dostępności zamówionych tytułów, to kolumna powinna należeć do tabeli **Sprzedaż_Szczegolow**

10 Tabele

10.1 Maska wprowadzania

Maska wprowadzania ułatwia wprowadzanie danych do pól tekstowych. Maska wprowadzania składa się z trzech sekcji, oddzielonych średnikami, przy czym:

Sekcja	Dotyczy	
Pierwsza	Służy do określania samej maski wprowadzania	
druga	Wartość	Efekt
	0	Wszystkie znaki używane w masce wprowadzania będą przechowywane razem z wprowadzoną wartością
	1	Przechowywane dane będą jedynie dane wpisane do pola (ten sam efekt osiągniemy pozostawiając tą sekcję pustą)
Trzecia	Określa znak, jaki będzie wyświetlany w miejscu, w którym wstawiono spację zamiast żadanego znaku	

Przy tworzeniu maski wprowadzania można zdecydować, które znaki będą wymagane, a które będą jedynie opcjonalne.

Znak	Opis
0	Cyfra (0 ... 9, pozycja wymagana)
9	Cyfra lub spacja (pozycja nie wymagana)
#	Cyfra lub spacja (pozycja nie wymagana, dozwolone są również znaki + i -
L	Litera (A .. Z, pozycja wymagana)
?	Litera (A .. Z, pozycja nie wymagana)
A	Litera lub cyfra (pozycja wymagana)
a	Litera lub cyfra (pozycja nie wymagana)
&	Dowolny znak lub spacja (pozycja wymagana)
C	Dowolny znak lub spacja (pozycja nie wymagana)
<	Powoduje, że wszystkie litery zostaną zmienione na małe
>	Powoduje, że wszystkie litery zostaną zmienione na duże
!	Powoduje, że wszystkie dane są wyświetlane od strony prawej do lewej zamiast od lewej do prawej
\	Powoduje, że znak, który po nim występuje, zostanie wyświetlony jako zwykły literał
Hasło	Dowolny wprowadzony znak będzie wyświetlany w postaci gwiazdki

Właściwość format wpływa jedynie na sposób wyświetlania wartości, a nie na sposób zapamiętywania jej w tabeli. Format nie kontroluje też sposobu wprowadzania danych. Jeżeli wymagana jest taka kontrola danych już podczas prowadzenia danych, to należy wykorzystać maskę wprowadzania.

10.2 Relacje między tabelami

Wymuszaj więzy integralności – włączenie tej opcji wymusza wzajemną zgodność danych w połączonych tabelach

Kaskadowe aktualizacje powiązane pola – włączenie tej opcji powoduje automatyczną aktualizację wartości połączonych pól od strony „wiele”, a w przypadku zmiany wartości połączonego pola od strony „jeden”

Kaskadowo usuwaj powiązane rekordy – włączenie tej opcji pozwala na usuwanie rekordów z tabeli połączonej od strony „jednej” z automatycznym usuwaniem wszystkich powiązanych rekordów z tabeli od strony „wiele”.

10.3 Połączenia między tabelami:

Inner join – odpowiada tzw. połączeniu zawężającemu – wyświetlające są tylko te rekordy, w których sprzęgane pola obu tabel są równe

Left join – odpowiada tzw. lewemu połączeniu rozszerzającemu – wyświetlane są wszystkie rekordy z pierwszej tabeli i tylko te rekordy z drugiej tabeli, gdy sprzężone pola są równe

Right join - odpowiada tzw. prawemu połączeniu rozszerzającemu – wyświetlane są wszystkie rekordy z drugiej tabeli i tylko te rekordy z pierwszej tabeli, gdy sprzężone pola są równe

11 Zasady składni SQL-a (HELP)

Słowa kluczowe SQL w instrukcjach składni są zawsze pisane wielkimi literami, chociaż w większości wersji SQL możesz je pisać (w odróżnieniu od identyfikatorów) w dowolny sposób

Nawiasy klamrowe {} otaczają słowa lub wyrażenia oznaczają, że musisz wybrać, co najmniej jedną z zaznaczonych opcji. Jeżeli opcje są rozdzielone pionowymi kreskami (|), to musisz wybrać tylko jedną; lecz jeśli są rozdzielone przecinkami (,), to musisz wybrać jedną lub więcej.

Nawiasy kwadratowe ([]) oznaczają, że stosowana opcja .. opcjonalna. Jeśli opcje są rozdzielone pionowymi kreskami (|), to możesz wybrać jedną z nich lub nie wybrać żadnej, ale jeśli są rozdzielone przecinkami (,), to możesz wybrać jedną lub więcej bądź nie wybierać żadnej.

Instrukcja SQL wymaga znaku końca, który powoduje, że polecenie SQL jest wysyłane do systemu zarządzania bazą danych w celu wykonania go. W różnych dialektach SQL są stosowane różne znaki końca; na ogół jest to średnik (;) lub słowo go.

11.1 Przykłady:

{opcja_1 opcja_2}	musisz wybrać jedno
{opcja_1, opcja_2, opcja_3}	musisz wybrać jedno lub więcej
[opcja_1]	nie musisz tego wybierać
[opcja_1 opcja_2 opcja_3]	możesz wybierać jedno lub nie wybierać niczego
[opcja_1 , opcja_2 , opcja_3]	możesz wybrać jedno lub więcej lub nie wybierać niczego

12 Tworzenie tabel

12.1 podstawowa składnia SQL

Gdy tworzysz tabelę w SQL, wtedy co najmniej

nadajesz jej **nazwę tabeli**

nadajesz **nazwy** wchodzącym w jej skład **kolumnom**

określasz **typ danych** dla każdej kolumny

określasz status **null** (lub akceptujesz wartość domyślną) dla każdej kolumny

12.2 tworzenie tabeli – proces tworzenia

doprowadź tabele min do trzeciej postaci normalnej

Określ typ danych (długość, precyzję i skalę, jeśli to wymagane) każdej kolumny

Określ, które kolumny powinny akceptować wartość null, a które nie

Określ, które kolumny muszą być jednoznaczne.

Zrób notatkę o parach klucz główny – klucz obcy.

Utwórz tabelę (i koniecznie index-y) za pomocą poleceń CREATE TABLE i CREATE INDEX

13 Polecenie CREATE

```
CREATE TABLE nazwa_tabeli
    (nazwa_pola1 typ_danych1 ograniczenie1,
    nazwa_pola2 typ_danych2 ograniczenie2,
nazwa_pola3 typ_danych3 [NULL|NOT NULL] [DEFAULT wartosc_domyslna),
    PRIMARY KEY (nazwa_pola1)
    FOREIGN KEY nazwa_klucza_obcego (nazwa_pola_kolumny_obcej)
    REFERENCES tabela_klucza_podstawowego (nazwa_pola1)
    ON UPDATE czynność_przy_modyfikacji
    ON DELETE czynność_przy_kasowaniu;
```

```
CREATE TABLE distributors (
    did DECIMAL(3),
    name VARCHAR(40),
    CONSTRAINT if_film_exists
    FOREIGN KEY(did)
    REFERENCES films
        ON UPDATE CASCADE
        ON DELETE CASCADE );
```

13.1.1 Uproszczona składnia polecenia CREATE TABLE

```
CREATE TABLE nazwa_tabeli
(nazwa_kolumny typ_danych [NULL|NOT NULL]
[, nazwa_kolumny typ_danych [NULL|NOT NULL] ]....);
```

13.1.2 Rozszerzona składnia CREATE TABLE

```
CREATE TABLE nazwa_tabeli
    (nazwa_pola1 typ_danych1 ograniczenie1,
    nazwa_pola2 typ_danych2 ograniczenie2,
nazwa_pola3 typ_danych3 [NULL|NOT NULL] [DEFAULT wartosc_domyslna),
    PRIMARY KEY (nazwa_pola1)
    FOREIGN KEY nazwa_klucza_obcego (nazwa_pola_kolumny_obcej)
    REFERENCES tabela_klucza_podstawowego (nazwa_pola1)
    ON UPDATE czynność_przy_modyfikacji
    ON DELETE czynność_przy_kasowaniu;
```

13.1.3 Ograniczenia tabel

PRIMARY KEY zaznacza kolumnę (nie dopuszczającą wartości null) jako klucz główny tabeli. Wewnętrzna obsługa tego ograniczenia jest różna w różnych systemach, lecz często jest równoważna tworzeniu indeksu

UNIQUE również gwarantuje jednoznaczność każdej wartości w kolumnie, lecz dopuszcza, żeby kolumna mogła być zdefiniowana jako null (zauważ, że kolumnę zadeklarowaną jako UNIQUE może przypaść tylko jedna wartość null).

DEFAULT definiuje wartość, jaką system nadaje automatycznie, gdy użytkownik wprowadzając dane nie poda jej w sposób jawny

CHECK określa, jakie dane mogą zostać wprowadzone do określonej kolumny: jest to sposób na zdefiniowanie dziedziny kolumny. Ograniczenie nakładane przez CHECK

są czasem nazywane regułami poprawności, gdyż pozwalają one systemowi sprawdzić, czy wprowadzana do kolumny wartość jest elementem dziedziny kolumny.

REFERENCES i **FOREIGN KEY** wiąże klucz główny z kluczem obcym.

13.1.4 rozszerzona składnia SQL-92

Rozszerzona składnia SQL dopuszcza:

wartości domyślne (wartości przypisywane automatycznie) **DEFAULT**

więzy ograniczające dopuszczalne wartości nadawane poszczególnym elementom kolumny

więzy definiujące kolumnę jako klucz główny, klucz jednoznaczny lub jako oba klucze jednocześnie **PRIMARY KEY**, **FOREIGN KEY**, **REFERENCES**

więzy ustawiające sprawdzenie zależności między kluczami głównymi a obcymi

13.1.5 Wstawianie - liczby całkowite

Całkowitoliczbowe typy danych służą do przechowywania liczb całkowitych (bez ułamków i liczb dziesiętnych)

Number, integer, int, smailint, tinyint ...

Tworzenie tabeli	Wstawianie danych
<pre>CREATE TABLE liczby (id_liczby serial, liczba_4 int4, liczba_8 int8);</pre>	<pre>INSERT INTO liczby VALUES (1, 2147483647, 12345678901234567890);</pre>
<p>NOTICE: CREATE TABLE will create implicit sequence 'liczby_id_liczby_seq' for SERIAL column 'liczby.id_liczby'</p> <p>NOTICE: CREATE TABLE/UNIQUE will create implicit index 'liczby_id_liczby_key' for table 'liczby'</p> <p>CREATE</p>	<pre>INSERT 56166 1</pre>

\d liczby

table = liczby

Field	Type	Length
id_liczby	int4 not null default nextval	4
liczba_4	int4	4
liczba_8	int8	8

Index: liczby_id_liczby_key

Składnia	efekt
<pre>SELECT * FROM liczby;</pre>	<pre>id_liczby liczba_4 liczba_8 -----+-----+----- 1 2147483647 1234567801234567890 (1 row)</pre>

13.1.6 Wstawianie - Liczby zmiennoprzecinkowe

Dziesiętne typy danych służą do przechowywania liczb z częściami ułamkowymi. Dokładnie liczby dziesiętne są znane pod nazwami decimal lub numeric. Zazwyczaj na-

leży określić ich precyzję (całkowitą liczbę cyfr po obu stronach kropki dziesiętnej) i skalę (liczbę cyfr po kropce dziesiętnej)

Real, double, double precision, float, smallfloat

Porządkowe typy danych służą do przechowywania liczb rosnących sekwencyjnie.

W niektórych przypadkach taka cecha jest traktowana jako oddzielny typ danych

Serial

Tworzenie tabeli	Wstawianie danych
<pre>CREATE TABLE zmienne (liczba10 decimal(10,2), liczba11 float4, liczba12 float8, liczba13 numeric,);</pre>	<pre>INSERT INTO liczby VALUES (12.34, 1234567890.1234567890, 1234567890.1234567890, 1234567890.123456);</pre>
CREATE	INSERT 56236 1

\d zmienne

Table = zmienne

Field	Type	Length
Liczba10	numeric	5.2
Liczba11	float4	4
Liczba12	float8	8
Liczba	numeric	30.2

Składnia	efekt
SELECT * FROM zmienne;	<pre>liczba10 liczba11 liczba12 liczba -----+-----+-----+----- 12.34 1.23457e+09 1234567890.12346 1234567890.123456 (1 row)</pre>

13.1.7 Wstawianie - kwotowe

Walutowy typ danych służy do przechowywania wartości walutowych. Jeśli twój system nie ma specjalnego typu walutowego, używaj typu dokładnych liczb dziesiętnych

Tworzenie tabeli	Wstawianie danych
<pre>CREATE TABLE place (placa_zasadnicza money);</pre>	<pre>INSERT INTO place VALUES ('12345678,12'</pre>
CREATE	INSERT 56265 1

\d place

Table = place

Field	Type	Length
Placa zasadnicza	money	4

Składnia	efekt
SELECT * FROM place;	<pre>Placa_zasadnicza ----- Zł12.345.678,12 (1 row)</pre>

13.1.8 Wstawianie - znakowe typy danych

Znakowe typy danych służą do przechowywania liter, liczb i znaków specjalnych

Char (character), varchar (variable character)

Mogą występować też inne

Text , long, nchar

Tworzenie tabeli	Wstawianie danych
<pre>CREATE TABLE tekst (jeden_zn char, wiele_znakow char(5), zmienny_zn varchar(5), opis text);</pre>	<pre>INSERT INTO place VALUES ('T', 'Marek', 'Marco', 'Opis tekstowy'</pre>
CREATE	INSERT 56314

\d tekst

Table = tekst

Field	Type	Length
jeden_zn	char()	1
wiele_zn	char()	5
zmienny_zn	varchar()	5
opis	text	var

Składnia	efekt
SELECT * FROM place;	<pre>jeden_zn wiele_zn zmienny_zn opis -----+-----+-----+----- T Marek Marco Opis techniczny (1 row)</pre>

13.1.9 Wstawianie - Data

Datowy i czasowy typ danych służy do przechowywania daty, czasu oraz kombinacji daty i czasu

Tworzenie tabeli	Wstawianie danych
<pre>CREATE TABLE daty (data_ur date, data_sys date DEFAULT 'today');</pre>	<pre>INSERT INTO daty (data_ur) VALUES ('10/03/1961'</pre>
CREATE	INSERT 56326 1

\d daty

Table = daty

Field	Type	Length
data_ur	date	4
data_sys	date default 'today'	4

Składnia	efekt
----------	-------

<code>SELECT * FROM daty;</code>	Data_ur data_sys -----+----- 10-03-1961 12-12-2001 (1 row)
----------------------------------	---

13.1.10 Wstawianie - logiczne

Tworzenie tabeli	Wstawianie danych
<code>CREATE TABLE logika (Tak Boolean);</code>	<code>INSERT INTO logika VALUES ('T'); INSERT INTO logika VALUES ('Y');</code>
<code>CREATE</code>	<code>INSERT 56337 1 INSERT 56338 1</code>

\d logika

Table = logika

Field	Type	length
tak	bool	1

Składnia	efekt
<code>SELECT * FROM logika;</code>	Tak ----- t t (2 rows)

13.1.11 Tworzenie danych i wstawianie

<code>CREATE TABLE mozliwosci (id_mozliw int NOT NULL, data_wprow date DEFAULT 'today', Nazwisko char(10) NOT NULL, Imie varchar(8) NOT NULL, Miasto varchar(10) DEFAULT 'Ostroleka', Data_ur date, dzieci char CHECK (dzieci = 'T' OR dzieci = 'N'), zarobki money, zonaty boolean NOT NULL, PRIMARY KEY (id_mozliwosci));</code>
<code>CREATE TABLE mozliwosci (nr_mozl int CONSTRAINT ID_m PRIMARY KEY, data_wprow date DEFAULT 'today', Nazwisko char(10) NOT NULL, Imie varchar(8) NOT NULL, Miasto varchar(10) DEFAULT 'Ostroleka', Data_ur date, dzieci char CHECK (dzieci = 'T' OR dzieci = 'N'),</code>

```
zarobki    money,  
zonaty     boolean NOT NULL
```

```
);
```

```
INSERT INTO mozliwosci ( zonaty, dzieci, imie, nazwisko, nr_mozl)  
VALUES
```

```
(  
  'T', 'N', 'Marek', 'RADECKI', 1  
);
```

14 Modyfikacja danych

14.1 Wstawianie danych – INSERT INTO

14.1.1 Wstawianie danych do wszystkich kolumn

INSERT INTO nazwa_tabeli
VALUES (stala1, stala2 ...);

```
CREATE TABLE mozliwosci
(
  id_m int NOT NULL,
  data_wp date DEFAULT 'today',
  Nazwisko char(10) NOT NULL,
  Imie varchar(8) NOT NULL,
  Miasto varchar(10) DEFAULT 'Ostroleka',
  Data_ur date,
  dzieci char CHECK (dzieci = 'T' OR dzieci = 'N'),
  zarobki money,
  zonaty boolean NOT NULL,
  PRIMARY KEY (id_m)
);
```

NOTICE: CREATE TABLE/PRIMARY KEY will create implicit index 'mozliwosci_pkey' for table 'm-
mozliwosci'

```
CREATE
EOF
```

uczen2000=> \d mozliwosci

Table = mozliwosci

Field	Type	Length
id_m	int4 not null	4
data_wp	date default 'today'	4
nazwisko	char() not null	10
imie	varchar() not null	8
miasto	varchar() default 'Ostroleka'	10
data_ur	date	4
dzieci	char()	1
zarobki	money	4
zonaty	bool not null	1

Index: mozliwosci_pkey

```
INSERT INTO mozliwosci
```

```
VALUES
```

```
(1, '2002-02-26', 'Radecki', 'Marek', 'Ostroleka', '1961-03-10', 'T', '2000.00', 'T');
```

```
INSERT 66784 1
```

```
EOF
```

uczen2000=> select * from mozliwosci;

id_m	data_wp	nazwisko	imie	miasto	data_ur	dzieci	zarobki	zonaty
1	02-26-2002	Radecki	Marek	Ostroleka	03-10-1961	T	Z1200.000,00	t

(1 row)

14.1.2 Wstawianie danych do wybranych kolumn

```
INSERT INTO nazwa_tabeli (nazwa_kol1, nazwa_kol2 ....)  
VALUES (stala1, stala2 ...);
```

```
CREATE TABLE mozliwosci
(
  id_m int NOT NULL,
  data_wp      date DEFAULT 'today',
  Nazwisko    char(10) NOT NULL,
  Imie        varchar(8) NOT NULL,
  Miasto      varchar(10) DEFAULT 'Ostroleka',
  Data_ur     date,
  dzieci      char CHECK (dzieci = 'T' OR dzieci = 'N'),
  zarobki     money,
  zonaty      boolean NOT NULL, PRIMARY KEY (id_m)
);
```

NOTICE: CREATE TABLE/PRIMARY KEY will create implicit index 'mozliwosci_pkey' for table 'mozliwosci'

```
CREATE
EOF
```

uczen2000=> \d mozliwosci

Table = mozliwosci

Field	Type	Length
id_m	int4 not null	4
data_wp	date default 'today'	4
nazwisko	char() not null	10
imie	varchar() not null	8
miasto	varchar() default 'Ostroleka'	10
data_ur	date	4
dzieci	char()	1
zarobki	money	4
zonaty	bool not null	1

Index: mozliwosci_pkey

```
INSERT INTO mozliwosci
```

```
  (id_m, nazwisko, imie, zonaty, dzieci)
VALUES
  (2, 'Beńka', 'Benia', 'Y', 'T');
```

```
INSERT 66848 1
```

```
EOF
```

uczen2000=> select * from mozliwosci;

id_m	data_wp	nazwisko	imie	miasto	data_ur	dzieci	zarobki	zonaty
1	02-26-2002	Radecki	Marek	Ostrołęka	03-10-1961	T	zł200.000,00	t
2	02-26-2002	Beńka	Benia	Ostrołęka		T		T

(2 row)

14.1.3 SELECT w poleceniu INSERT

INSERT INTO *nazwa_tabeli* (lista_kolumn)

SELECT lista_kolumn

FROM lista_tabel

WHERE warunki_wyszukiwania

```
CREATE TABLE mozliwosci
(
  nr_mozl          int          CONSTRAINT ID_m PRIMARY KEY,
  data_wprow      date          DEFAULT 'today',
  Nazwisko        char(10)      NOT NULL,
  Imie            varchar(8),
  Miasto         varchar(10)    DEFAULT 'Ostroleka',
  Data_ur        date,
  zarobki        money,
);
```

NOTICE: CREATE TABLE/PRIMARY KEY will create implicit index 'id_m' for table 'm

ozliwosci'

CREATE

EOF

INSERT INTO *mozliwosci* (nr_mozl, nazwisko)

SELECT nump, nazwp

FROM *prac*

WHERE stanowisko='SPRZEDAWCA';

INSERT 0 4

EOF

uczen2000=> select * from *mozliwosci*;

nr_mozl	data_wprow	nazwisko	imie	miasto	data_ur	zarobki
7499	02-26-2002	ALLEN		Ostroleka		
7521	02-26-2002	WARD		Ostroleka		
7654	02-26-2002	MARTIN		Ostroleka		
7844	02-26-2002	TURNER		Ostroleka		

(4 rows)

14.2 Zmiana istniejących danych – UPDATE

UPDATE nazwa_tabeli

SET nazwa_kolumny = wyrażenie

WHERE warunki_wyszukiwania

Po słowie kluczowym UPDATE podajemy nazwę tabeli lub perspektywy

Klauzula SET określa kolumny i zmieniane wartości

Uczen2000=> select * from mozliwosci;

nr_mozl	data_wprow	nazwisko	imie	miasto	data_ur	zarobki
7499	02-26-2002	ALLEN		Ostroleka		
7521	02-26-2002	WARD		Ostroleka		
7654	02-26-2002	MARTIN		Ostroleka		
7844	02-26-2002	TURNER		Ostroleka		

(4 rows)

UPDATE mozliwosci

SET zarobki='1000'

WHERE nazwisko='WARD';

UPDATE 1

EOF

Uczen2000=> select * from mozliwosci;

nr_mozl	data_wprow	nazwisko	imie	miasto	data_ur	zarobki
7499	02-26-2002	ALLEN		Ostroleka		
7654	02-26-2002	MARTIN		Ostroleka		
7844	02-26-2002	TURNER		Ostroleka		
7521	02-26-2002	WARD		Ostroleka		Z11.000,00

(4 rows)

14.3 Usuwanie danych z tabeli

14.3.1 Usuwanie wszystkich danych – całej tabeli

DROP TABLE nazwa_tabeli;

14.3.2 Usuwanie wybranych danych

DROP TABLE nazwa_tabeli
WHERE warunek_wyszukiwan;

14.4 Zmiana definicji tabel

14.4.1 Wstawianie dodatkowej kolumny

ALTER TABLE nazwa_tabeli
ADD nazwa_kolumny typ ograniczenie;

```
CREATE TABLE premia_01_2002
```

```
( nazwp          int,  
  stanowisko     varchar(10) not null,  
  zarob          numeric(10,2),  
  prow          numeric(10,2)  
);
```

```
CREATE
```

```
ALTER TABLE premia_01_2002
```

```
ADD zatrud date;
```

```
ADD
```

```
EOF
```

```
uczen2000=> \d premia_01_2002
```

```
Table = premia_01_2002
```

Field	Type	Length
nazwp	int4	4
stanowisko	varchar() not null	10
zarob	numeric	10.2
prow	Numeri	10.2
zatrud	date	4

14.4.2 Wstawianie INDEX-u

```
CREATE INDEX nazwa_indeksu  
ON nazwa_tabeli (nazwa_kolumny)
```

```
CREATE TABLE premia_01_2002  
( nazwp int,  
  stanowisko varchar(10) not null,  
  zarob numeric(10,2),  
  prow numeric(10,2)  
);  
CREATE
```

```
CREATE INDEX id_nazwp  
ON premia_01_2002 (nazwp);  
CREATE  
EOF
```

```
uczen2000=> \d premia_01_2002  
Table = premia_01_2002
```

Field	Type	Length
nazwp	int4	4
stanowisko	varchar() not null	10
zarob	numeric	10.2
prow	numeric	10.2

```
Index: id_nazwp
```

14.4.3 Usuwanie tabeli i indexu

```
DELETE TABLE nazwa_tabeli;  
DELETE INDEX nazwa_indexu;
```

```
uczen2000=> DROP INDEX id_nazwp;  
DROP
```

```
uczen2000=> DROP TABLE premia_01_2002;  
DROP
```

```
uczen2000=> DROP TABLE premia_01_2002;  
DROP
```

```
uczen2000=> DROP INDEX id_nazwp;  
ERROR: index "id_nazwp" nonexistent
```

14.4.4 Zmiana nazwy tabeli i kolumny

ALTER TABLE nazwa_tabeli

RENAME TO nowa_nazwa_tabeli;

ALTER TABLE nazwa_tabeli

RENAME nazwa_kolumny **TO** nowa_nazwa_kolumny

uczen2000=> \d premia_01_2002

Table = premia_01_2002

Field	Type	Length
nazwp	varchar()	10
stanowisko	varchar() not null	10
zarob	numeric	10,2
prow	numeric	10,2
zatrud	date	4
nump	int4	4
zarob_01	int4	4

Index: id_nazwp

ALTER TABLE premia_01_2002

RENAME TO premia_02_2002;

RENAME

EOF

ALTER TABLE premia_02_2002

RENAME zarob **TO** zarobki;

RENAME

ALTER TABLE premia_02_2002

RENAME prow **TO** prowizja;

RENAME

EOF

uczen2000=> \d premia_02_2002

Table = **premia_02_2002**

Field	Type	Length
nazwp	varchar()	10
stanowisko	varchar() not null	10
zarobki	numeric	10,2
prowizja	numeric	10,2
zatrud	date	4
nump	int4	4
zarob_01	int4	4

Index: id_nazwp

14.4.5 Podsumowanie

<pre>CREATE TABLE premia_01_2002 (nazwp varchar(10), stanowisko varchar(10) not null, zarob numeric(10,2), prow numeric(10,2)); CREATE ALTER TABLE premia_01_2002 ADD zatrud date; ADD</pre>	<pre>ALTER TABLE premia_01_2002 ADD nump int; ADD CREATE INDEX id_nazwp ON premia_01_2002 (nump); CREATE EOF uczen2000=> select * from premia_01_2002; nazwp stanowisko zarob prow zatrud nump -----+-----+-----+-----+-----+----- (0 rows)</pre>
<pre>uczen2000=> \d premia_01_2002 Table = premia_01_2002 +-----+-----+-----+-----+ Field Type Length +-----+-----+-----+-----+ nazwp varchar() 10 stanowisko varchar() not null 10 zarob numeric 10.2 prow numeric 10.2 zatrud date 4 nump int4 4 +-----+-----+-----+-----+ Index: id_nazwp</pre>	
<pre>INSERT INTO premia_01_2002 (nazwp, stanowisko, zarob, prow, zatrud, nump) SELECT nazwp, stanowisko, zarob, prow, zatrud, nump FROM prac WHERE stanowisko='URZEDNIK' OR stanowisko='SPRZEDAWCA'; INSERT 0 8 EOF</pre>	
<pre>uczen2000=> select * from premia_01_2002; nazwp stanowisko zarob prow zatrud nump -----+-----+-----+-----+-----+----- SMITKO URZEDNIK 800.00 12-17-1980 7369 ALLEN SPRZEDAWCA 1600.00 300.00 02-20-1981 7499 WARD SPRZEDAWCA 1250.00 500.00 02-22-1981 7521 MARTIN SPRZEDAWCA 1250.00 1400.00 09-28-1981 7654 TURNER SPRZEDAWCA 1500.00 200.00 08-09-1981 7844 ADAMCZYK URZEDNIK 1100.00 12-01-1983 7876 JAMSKI URZEDNIK 950.00 03-12-1981 7900 MILLER URZEDNIK 1300.00 01-23-1982 7934 (8 rows)</pre>	
<pre>ALTER TABLE premia_01_2002 ADD zarob_01 int; ADD UPDATE premia_01_2002 SET zarob_01=2000 WHERE stanowisko='SPRZEDAWCA';</pre>	<pre>UPDATE premia_01_2002 SET zarob_01=3500 WHERE stanowisko='URZEDNIK'; UPDATE 4 UPDATE premia_01_2002 SET zarob_01=zarob_01*1.3</pre>

UPDATE 4

WHERE nazwp='MILLER';

UPDATE 1

EOF

uczen2000=> select * from premia_01_2002;

nazwp	stanowisko	zarob	prow	zatrud	nump	zarob_01
ALLEN	SPRZEDAWCA	1600.00	300.00	02-20-1981	7499	2000
WARD	SPRZEDAWCA	1250.00	500.00	02-22-1981	7521	2000
MARTIN	SPRZEDAWCA	1250.00	1400.00	09-28-1981	7654	2000
TURNER	SPRZEDAWCA	1500.00	200.00	08-09-1981	7844	2000
SMITKO	URZEDNIK	800.00		12-17-1980	7369	3500
ADAMCZYK	URZEDNIK	1100.00		12-01-1983	7876	3500
JAMSKI	URZEDNIK	950.00		03-12-1981	7900	3500
MILLER	URZEDNIK	1300.00		01-23-1982	7934	4550

(8 rows)

uczen2000=> \d premia_01_2002

Table = premia_01_2002

Field	Type	Length
nazwp	varchar()	10
stanowisko	varchar() not null	10
zarob	numeric	10,2
prow	numeric	10,2
zatrud	date	4
nump	int4	4
zarob_01	int4	4

Index: id_nazwp

15 Instrukcja SELECT

15.1 Pobieranie danych

15.1.1 Wszystkich wierszy i kolumn

Składnia	wynik
SELECT * FROM DZIAL;	Numdz nazwdz lok -----+-----+----- 10 KSIEGOWOSC NOWY YORK 20 BADANIA DALLAS 30 SPRZEDAZ CHICAGO 40 OPERACJE BOSTON (4 rows)

15.1.2 Wszystkich wierszy i wybranej kolumny

Składnia	wynik
SELECT numdz FROM DZIAL;	numdz ----- 10 20 30 40 (4 rows)

15.1.3 Wszystkich wierszy i zestawu wybranych kolumn

Składnia	wynik
SELECT numdz, lok FROM DZIAL;	Numdz lok -----+----- 10 NOWY YORK 20 DALLAS 30 CHICAGO 40 BOSTON (4 rows)

15.1.4 Zmiana kolejności kolumn

Składnia	wynik
SELECT lok, numdz FROM DZIAL;	lok numdz -----+----- NOWY YORK 10 DALLAS 20 CHICAGO 30 BOSTON 40 (4 rows)

15.1.5 Listy przecinkowe

Lista przecinkowa frazy SELECT określa, jakie kolumny mają być wybrane w zapytaniu

Lista przecinkowa frazy FROM wylicza tabele, z których dane mają być wybrane

Lista przecinkowa frazy ORDER BY pozwala ustalić kryteria sortowania danych

Lista przecinkowa frazy GROUP BY jest używana do agregowania danych

Lista przecinkowa jest używana w instrukcji IN

15.1.6 Słowo kluczowe ALL

Składnia	wynik
SELECT ALL stanowisko FROM <i>premia</i> ;	stanowisko ----- URZEDNIK SPRZEDAWCA SPRZEDAWCA KIEROWNIK SPRZEDAWCA (5 rows)
SELECT stanowisko FROM <i>premia</i> ;	

Słowo ALL wybiera wszystkie wiersze tabeli i jest to ustawienie domyślne.

15.1.7 Słowo kluczowe DISTINCT

Składnia	wynik
SELECT DISTINCT stanowisko FROM <i>premia</i> ;	stanowisko ----- KIEROWNIK SPRZEDAWCA URZEDNIK (3 rows)
SELECT DISTINCT stanowisko, nazwp FROM <i>premia</i> ;	Stanowisko nazwp -----+----- KIEROWNIK JONAS SPRZEDAWCA ALLEN SPRZEDAWCA MARTIN SPRZEDAWCA WARD URZEDNIK SMITKO (5 rows)

Słowo kluczowe DISTINCT zwraca tylko różne wiersze

Niezależnie od tego, czy tabela ma dwa wiersze, czy milion wierszy, to jeśli dana kolumna zawiera tylko dwie różne wartości a w instrukcji SELECT DISTINCT użyjesz tylko tej kolumny, to zawsze otrzymasz dokładnie dwa wiersze

15.1.8 Zamiana nazwy kolumn

Gdy wyświetlane są zapytania, każda kolumna ma domyślny nagłówek z nazwą nadaną przez jej w bazie danych. Nazwy kolumn w bazie danych są często zlepkiem różnych wyrazów (aby można je było łatwo zapisać) bądź nic nie mówią użytkownikom nie obznajomionych z wydziałowymi akronimami, pseudonimami lub żargonem projektu. Problem ten można rozwiązać przez takie określenie nagłówków kolumn, żeby wyniki zapytania dało się łatwiej odczytywać i zrozumieć. Aby otrzymać pożądaną nagłówek, trzeba po prostu na liście wyboru w miejscu nazwy kolumny napisać:

Nazwa_kolumny naglowek_kolumny

lub

Nazwa_kolumny **AS** naglowek_kolumny

Składnia	Wynik
SELECT nazwp AS nazwisko, stanowisko FROM <i>premia</i> ;	nazwisko stanowisko -----+----- ALLEN SPRZEDAWCA

	JONAS	KIEROWNIK
	MARTIN	SPRZEDAWCA
	SMITKO	URZEDNIK
	WARD	SPRZEDAWCA
	(5 rows)	

15.2 Sortowanie

15.2.1 Wyznaczanie kolumny sortującej przez nazwę

Składnia	wynik
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY nazwp;</pre>	<pre>nazwp stanowisko -----+----- ALLEN SPRZEDAWCA JONAS KIEROWNIK MARTIN SPRZEDAWCA SMITKO URZEDNIK WARD SPRZEDAWCA (5 rows)</pre>
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY stanowisko;</pre>	<pre>nazwp stanowisko -----+----- JONAS KIEROWNIK ALLEN SPRZEDAWCA WARD SPRZEDAWCA MARTIN SPRZEDAWCA SMITKO URZEDNIK (5 rows)</pre>

15.2.2 Wyznaczanie kolumny sortującej przez kolejność

Składnia	wynik
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 1;</pre>	<pre>nazwp stanowisko -----+----- ALLEN SPRZEDAWCA JONAS KIEROWNIK MARTIN SPRZEDAWCA SMITKO URZEDNIK WARD SPRZEDAWCA (5 rows)</pre>
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 2;</pre>	<pre>nazwp stanowisko -----+----- JONAS KIEROWNIK ALLEN SPRZEDAWCA WARD SPRZEDAWCA MARTIN SPRZEDAWCA SMITKO URZEDNIK (5 rows)</pre>

15.2.3 Sposób sortowania ASC (rosnąco – ascending – A>Z)

Składnia	wynik
SELECT nazwp, stanowisko FROM premia ORDER BY 2 ASC ; SELECT nazwp, stanowisko FROM premia ORDER BY 2;	nazwp stanowisko -----+----- JONAS KIEROWNIK ALLEN SPRZEDAWCA WARD SPRZEDAWCA MARTIN SPRZEDAWCA SMITKO URZEDNIK (5 rows)

15.2.4 Sposób sortowania oraz DESC (malejąco – descending – Z<A)

Składnia	wynik
SELECT nazwp, stanowisko FROM premia ORDER BY 2 DESC ;	nazwp stanowisko -----+----- SMITKO URZEDNIK ALLEN SPRZEDAWCA WARD SPRZEDAWCA MARTIN SPRZEDAWCA JONAS KIEROWNIK (5 rows)

15.3 podsumowanie

Jeśli nie jest określony sposób sortowania, wynik będzie uporządkowany w porządku rosnącym kolumn użytych do sortowania. Ten sam efekt można uzyskać określając ten porządek wprost, używając słowa kluczowego ASC po nazwie kolumny
Słowo kluczowe DESC ma przeciwne znaczenie: nakazuje sortować w porządku malejącym

15.3.1 Sortowanie na podstawie kilku kolumn

Składnia	Wynik
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 1,2;</pre>	<pre>nazwp stanowisko -----+----- ALLEN SPRZEDAWCA JONAS KIEROWNIK MARTIN SPRZEDAWCA SMITKO URZEDNIK WARD SPRZEDAWCA (5 rows)</pre>
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 2,1;</pre>	<pre>nazwp stanowisko -----+----- JONAS KIEROWNIK ALLEN SPRZEDAWCA MARTIN SPRZEDAWCA WARD SPRZEDAWCA SMITKO URZEDNIK (5 rows)</pre>
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 1,2 DESC;</pre>	<pre>nazwp stanowisko -----+----- ALLEN SPRZEDAWCA JONAS KIEROWNIK MARTIN SPRZEDAWCA SMITKO URZEDNIK WARD SPRZEDAWCA (5 rows)</pre>
<pre>SELECT nazwp, stanowisko FROM premia ORDER BY 2,1 DESC;</pre>	<pre>nazwp stanowisko -----+----- JONAS KIEROWNIK WARD SPRZEDAWCA MARTIN SPRZEDAWCA ALLEN SPRZEDAWCA SMITKO URZEDNIK (5 rows)</pre>

15.3.2 Sortowanie według kolumn nie występujących na liście SELECT

Składnia	Wynik
<pre>SELECT nazwp, stanowisko, zarob FROM premia ORDER BY 3;</pre>	<pre>nazwp stanowisko zarob -----+-----+----- SMITKO URZEDNIK 800.00 WARD SPRZEDAWCA 1250.00 MARTIN SPRZEDAWCA 1250.00</pre>

Składnia	Wynik
	ALLEN SPRZEDAWCA 1600.00 JONAS KIEROWNIK 2975.00 (5 rows)
SELECT nazwp, stanowisko FROM premia ORDER BY zarob;	nazwp stanowisko -----+----- SMITKO URZEDNIK WARD SPRZEDAWCA MARTIN SPRZEDAWCA ALLEN SPRZEDAWCA JONAS KIEROWNIK (5 rows)
SELECT nazwp, stanowisko FROM premia ORDER BY 3;	ERROR: ORDER BY position 3 is not in target list

Niezależnie od tego czy kolumna istnieje na liście SELECT (jest pokazywana czy nie) czy nie, porządek jest taki sam

15.4 Wybieranie tabel

Składnia	Wynik
SELECT numdz, nazwdz, lok FROM dzial;	Numdz nazwdz lok -----+----- 10 KSIEGOWOSC NOWY YORK 20 BADANIA DALLAS 30 SPRZEDAZ CHICAGO 40 OPERACJE BOSTON (4 rows)
SELECT D.numdz, D.nazwdz, D.lok FROM dzial D;	Numdz nazwdz lok -----+----- 10 KSIEGOWOSC NOWY YORK 20 BADANIA DALLAS 30 SPRZEDAZ CHICAGO 40 OPERACJE BOSTON (4 rows)
SELECT P.numdz, P.nazwp, P.Stanowisko, D.lok FROM prac P, dzial D WHERE stanowisko='KIEROWNIK' AND P.numdz=D.numdz;	Numdz nazwp stanowisko lok -----+----- 20 JONAS KIEROWNIK DALLAS 30 BLACKI KIEROWNIK CHICAGO 10 CELAREK KIEROWNIK NOWY YORK (3 rows)

15.5 Wybieranie wierszy : klauzula WHERE

SQL dostarcza różnorodnych operatorów i słów kluczowych służących do formułowania warunków wyszukiwania:

Operatory porównania (=, <, >, <=, >=, <>, !=)

Kombinacje lub logiczne negacje warunków (AND, OR, NOT)

Przedziały (BETWEEN, NOT BETWEEN)

Listy (IN, NOT IN)

Wartości nieznanne (IS NULL, IS NOT NULL)

Zgodność znakowa (LIKE, NOT LIKE)

15.5.1 Operator porównania

Często potrzebujesz porównania ze sobą różnych wielkości. Chcesz stwierdzić, co jest „większe” lub „mniejsze, lub „leży niżej” w alfabetycznym sortowaniu, lub jest „równe” innej wielkości bazodanowej bądź stałej. W SQL-u są udostępnione w tym celu następujące operatory.

=	Równe	>=	Większe niż lub równe	!=	Nie równe
>	Większe niż	<=	Mniejsze niż lub równe	<>	Nie równe
<	Mniejsze niż				

Operatory te są używane w następujący sposób:

WHERE **wyrażenie** **Operator** **wyrażenie**

Wyrażenie może być stałą, nazwą kolumny, funkcją, podzapytaniem lub dowolną kombinacją tych elementów połączonych operatorami arytmetycznymi lub porównania. Porównywanie najczęściej nam się kojarzy z wartościami liczbowymi. W SQL-u są one używane również z danymi typu char i varchar czyli sortowanie alfabetyczne. Pamiętaj, że wartości typu znakowego należy umieszczać w cudzysłowach oraz kolejność, w jakiej są porządkowane wielkie i małe litery oraz znaki specjalne, zależy od kolejności znaków określonych przez system bazodanowy lub od używanej maszyny.

Składnia	Wynik																								
SELECT P.numdz as nr, P.nazwp as nazwisko, P.Stanowisko, D.zarob as placa FROM <i>prac</i> P, <i>premia</i> D WHERE P.nazwp=D.nazwp;	<table border="1"> <tr> <th>Nr</th> <th> nazwisko</th> <th> stanowisko</th> <th> placa</th> </tr> <tr> <td>20</td> <td> SMITKO</td> <td> URZEDNIK</td> <td> 800.00</td> </tr> <tr> <td>30</td> <td> ALLEN</td> <td> SPRZEDAWCA</td> <td> 1600.00</td> </tr> <tr> <td>30</td> <td> WARD</td> <td> SPRZEDAWCA</td> <td> 1250.00</td> </tr> <tr> <td>20</td> <td> JONAS</td> <td> KIEROWNIK</td> <td> 2975.00</td> </tr> <tr> <td>30</td> <td> MARTIN</td> <td> SPRZEDAWCA</td> <td> 1250.00</td> </tr> </table> (5 rows)	Nr	nazwisko	stanowisko	placa	20	SMITKO	URZEDNIK	800.00	30	ALLEN	SPRZEDAWCA	1600.00	30	WARD	SPRZEDAWCA	1250.00	20	JONAS	KIEROWNIK	2975.00	30	MARTIN	SPRZEDAWCA	1250.00
Nr	nazwisko	stanowisko	placa																						
20	SMITKO	URZEDNIK	800.00																						
30	ALLEN	SPRZEDAWCA	1600.00																						
30	WARD	SPRZEDAWCA	1250.00																						
20	JONAS	KIEROWNIK	2975.00																						
30	MARTIN	SPRZEDAWCA	1250.00																						
SELECT nazwp AS nazwisko, stanowisko, zarob AS placa FROM <i>prac</i> WHERE zarob > 2000;	<table border="1"> <tr> <th>nazwisko</th> <th> stanowisko</th> <th> placa</th> </tr> <tr> <td>JONAS</td> <td> KIEROWNIK</td> <td> 2975.00</td> </tr> <tr> <td>BLACKI</td> <td> KIEROWNIK</td> <td> 2850.00</td> </tr> <tr> <td>CELAREK</td> <td> KIEROWNIK</td> <td> 2450.00</td> </tr> <tr> <td>SKOTNIK</td> <td> ANALITYK</td> <td> 3000.00</td> </tr> <tr> <td>KING</td> <td> PREZES</td> <td> 5000.00</td> </tr> <tr> <td>FORD</td> <td> ANALITYK</td> <td> 3000.00</td> </tr> </table> (6 rows)	nazwisko	stanowisko	placa	JONAS	KIEROWNIK	2975.00	BLACKI	KIEROWNIK	2850.00	CELAREK	KIEROWNIK	2450.00	SKOTNIK	ANALITYK	3000.00	KING	PREZES	5000.00	FORD	ANALITYK	3000.00			
nazwisko	stanowisko	placa																							
JONAS	KIEROWNIK	2975.00																							
BLACKI	KIEROWNIK	2850.00																							
CELAREK	KIEROWNIK	2450.00																							
SKOTNIK	ANALITYK	3000.00																							
KING	PREZES	5000.00																							
FORD	ANALITYK	3000.00																							
SELECT nazwp as nazwisko, Stanowisko, zarob AS placa FROM <i>prac</i> WHERE zarob > 2000 AND stanowisko != 'PREZES';	<table border="1"> <tr> <th>nazwisko</th> <th> stanowisko</th> <th> placa</th> </tr> <tr> <td>JONAS</td> <td> KIEROWNIK</td> <td> 2975.00</td> </tr> <tr> <td>BLACKI</td> <td> KIEROWNIK</td> <td> 2850.00</td> </tr> <tr> <td>CELAREK</td> <td> KIEROWNIK</td> <td> 2450.00</td> </tr> <tr> <td>SKOTNIK</td> <td> ANALITYK</td> <td> 3000.00</td> </tr> <tr> <td>FORD</td> <td> ANALITYK</td> <td> 3000.00</td> </tr> </table> (5 rows)	nazwisko	stanowisko	placa	JONAS	KIEROWNIK	2975.00	BLACKI	KIEROWNIK	2850.00	CELAREK	KIEROWNIK	2450.00	SKOTNIK	ANALITYK	3000.00	FORD	ANALITYK	3000.00						
nazwisko	stanowisko	placa																							
JONAS	KIEROWNIK	2975.00																							
BLACKI	KIEROWNIK	2850.00																							
CELAREK	KIEROWNIK	2450.00																							
SKOTNIK	ANALITYK	3000.00																							
FORD	ANALITYK	3000.00																							

15.5.2 Kombinacje lub logiczne negacje warunków

Jeżeli w klauzuli WHERE podajesz więcej niż jeden warunek, to musisz zastosować operatory logiczne:

Opera- tor	nazwa	Wyjaśnienie
1	2	4
NOT	negacja (nie)	1 gdy x jest równy 0, w przeciwnym wypadku 0
AND	koniunkcja (i)	0 jeśli albo x albo y jest równy 0
OR	alternatywa (lub)	0 gdy x i y są równe 0

NOT	1	0
-----	---	---

AND	1	0
-----	---	---

OR	1	0
----	---	---

Negacja		
	0	1

koniunkcja		
1	1	0
0	0	0

Alternatywa		
1	1	1
0	1	0

Uwaga:

AND łączy dwa lub więcej warunków i zwraca wyniki tylko wtedy, gdy wszystkie warunki są prawdziwe

OR również łączy dwa lub więcej warunków, ale wyniki zwraca wtedy, gdy którykolwiek z warunków jest prawdziwy

NOT zaprzecza danemu wyrażeniu. Gdy używasz go wraz z operatorami porównania, wówczas umieszczaj go przed wyrażeniami, a nie przed operatorami porównania.

Składnia		Wynik			
SELECT	nazwp, stanowisko, zatrud, zarob	nazwp	stanowisko	zatrud	zarob
FROM	<i>prac</i>	SMITKO	URZEDNIK	12-17-1980	800.00
WHERE	NOT stanowisko='PREZES';	ALLEN	SPRZEDAWCA	02-20-1981	1600.00
		WARD	SPRZEDAWCA	02-22-1981	1250.00
		JONAS	KIEROWNIK	02-04-1981	2975.00
		MARTIN	SPRZEDAWCA	09-28-1981	1250.00
		BLACKI	KIEROWNIK	01-05-1981	2850.00
		CELAREK	KIEROWNIK	09-06-1981	2450.00
		SKOTNIK	ANALITYK	09-12-1981	3000.00
		TURNER	SPRZEDAWCA	08-09-1981	1500.00
		ADAMCZYK	URZEDNIK	12-01-1983	1100.00
		JAMSKI	URZEDNIK	03-12-1981	950.00
		FORD	ANALITYK	03-12-1981	3000.00
		MILLER	URZEDNIK	01-23-1982	1300.00
		(13 rows)			

15.5.3 Przedziały

Innymi, popularnymi warunkami wyszukiwania jest podanie przedziału. Są dwa różne sposoby określenia przedziałów za pomocą:

Operatorów porównania > i <

Słowa kluczowego BETWEEN

BETWEEN należy używać do określenia przedziału domkniętego, w którym poszukujemy wartości najmniejszej i największej, a także wszystkich wartości zawartych między nimi.

Wyrażenie **NOT BETWEEN** powoduje wyszukiwanie wszystkich wierszy nie należących do przedziału.

15.5.3.1 Przykład – operator porównania

Składnia		Wynik		
SELECT	nazwp AS Nazwisko, Stanowisko, zarob AS Place	Nazwisko	stanowisko	placa
FROM	<i>prac</i>	ALLEN	SPRZEDAWCA	1600.00
WHERE	zarob > 1000 AND zarob < 2500;	WARD	SPRZEDAWCA	1250.00
		MARTIN	SPRZEDAWCA	1250.00
		CELAREK	KIEROWNIK	2450.00
		TURNER	SPRZEDAWCA	1500.00
		ADAMCZYK	URZEDNIK	1100.00
		MILLER	URZEDNIK	1300.00
		(7 rows)		
SELECT	nazwp AS Nazwisko, Stanowisko, zarob AS Place	Nazwisko	stanowisko	placa
FROM	<i>prac</i>	ALLEN	SPRZEDAWCA	1600.00
WHERE	zarob BETWEEN 1000 AND 2500;	WARD	SPRZEDAWCA	1250.00
		MARTIN	SPRZEDAWCA	1250.00
		CELAREK	KIEROWNIK	2450.00
		TURNER	SPRZEDAWCA	1500.00

Składnia	Wynik
	ADAMCZYK URZEDNIK 1100.00 MILLER URZEDNIK 1300.00 (7 rows)

15.5.3.2 Przykład – operator BETWEEN

Składnia	Wynik
SELECT nazwp AS Nazwisko, Stanowisko, zarob AS Place FROM <i>prac</i> WHERE zarob BETWEEN 1000 AND 2500;	Nazwisko stanowisko placa -----+-----+----- ALLEN SPRZEDAWCA 1600.00 WARD SPRZEDAWCA 1250.00 MARTIN SPRZEDAWCA 1250.00 CELAREK KIEROWNIK 2450.00 TURNER SPRZEDAWCA 1500.00 ADAMCZYK URZEDNIK 1100.00 MILLER URZEDNIK 1300.00 (7 rows)
SELECT nazwp AS Nazwisko, Stanowisko, zarob AS Place FROM <i>prac</i> WHERE zarob > 1000 AND zarob < 2500;	Nazwisko stanowisko placa -----+-----+----- ALLEN SPRZEDAWCA 1600.00 WARD SPRZEDAWCA 1250.00 MARTIN SPRZEDAWCA 1250.00 CELAREK KIEROWNIK 2450.00 TURNER SPRZEDAWCA 1500.00 ADAMCZYK URZEDNIK 1100.00 MILLER URZEDNIK 1300.00 (7 rows)

15.5.3.3 Przykład – operator NOT BETWEEN

Składnia	Wynik
SELECT nazwp AS Nazwisko, Stanowisko, zarob AS Place FROM <i>prac</i> WHERE zarob NOT BETWEEN 1000 AND 2500;	Nazwisko stanowisko placa -----+-----+----- SMITKO URZEDNIK 800.00 JONAS KIEROWNIK 2975.00 BLACKI KIEROWNIK 2850.00 SKOTNIK ANALITYK 3000.00 KING PREZES 5000.00 JAMSKI URZEDNIK 950.00 FORD ANALITYK 3000.00 (7 rows)
SELECT nazwp AS Nazwisko, Stanowisko, zarob AS Place FROM <i>prac</i> WHERE zarob < 1000 OR zarob > 2500;	Nazwisko stanowisko placa -----+-----+----- SMITKO URZEDNIK 800.00 JONAS KIEROWNIK 2975.00 BLACKI KIEROWNIK 2850.00 SKOTNIK ANALITYK 3000.00 KING PREZES 5000.00 JAMSKI URZEDNIK 950.00 FORD ANALITYK 3000.00 (7 rows)

15.5.4 Listy (IN i NOT IN)

Słowo kluczowe IN umożliwia wybranie wartości pokrywających się z którąkolwiek wymienioną na liście wartości.

NOT IN powoduje wyszukuje wiersze, które nie spełniają warunku

15.5.4.1 Przykłady IN i NOT IN

Składnia	Wynik
SELECT numdz AS ID, nazwdz AS dzial, lok AS miasto FROM <i>dzial</i> WHERE lok IN ('DALLAS', 'BOSTON');	Id dzial miasto -----+-----+----- 20 BADANIA DALLAS 40 OPERACJE BOSTON (2 rows)
SELECT numdz AS ID,	Id dzial miasto

Składnia	Wynik
<pre> nazwdz AS dzial, lok AS miasto FROM dzial WHERE lok='DALLAS' OR lok='BOSTON'; </pre>	<pre> -----+-----+----- 20 BADANIA DALLAS 40 OPERACJE BOSTON (2 rows) </pre>
<pre> SELECT numdz AS ID, nazwdz AS dzial, lok AS miasto FROM dzial WHERE lok NOT IN ('DALLAS', 'BOSTON'); </pre>	<pre> Id dzial miasto -----+-----+----- 10 KSIEGOWOSC NOWY YORK 30 SPRZEDAZ CHICAGO (2 rows) </pre>

15.5.5 Dopasowanie napisów: LIKE

Niektóre problemy nie mogą być rozwiązywane na drodze porównania. Np.

Jego nazwisko rozpoczyna się na „Rade” lub „Radw” – dalej nie pamiętam

Potrzebujemy spisu wszystkich telefonów o numerze kierunkowym 029

W każdym z tych przypadków znasz wzorzec zanurzony gdzieś w kolumnę i musisz go używać, żeby odnaleźć cały wiersz lub jego część. Do rozwiązania tych problemów jest przeznaczone słowo kluczowe LIKE. Możesz go używać z polami znakowymi (a w niektórych systemach z polami danych). Z polami numerycznymi zdefiniowanymi jako liczby całkowite, waluta, liczby dziesiętne lub zmiennopozycyjne nie działa.

15.5.5.1 Składnia

WHERE nazwa_kolumny **[NOT] LIKE** ‘wzorzec’

ANSI SQL dostarcza dwa znaki uniwersalne, które możesz stosować z LIKE:

% zastępuje dowolny złożony z zera lub większej liczby znaków

- zastępuje jeden dowolny znak

Składnia	Wynik
<pre> SELECT nazwp AS nazwisko, Stanowisko, zatrud AS Data_Zatrudnienia FROM prac WHERE nazwp LIKE 'J%'; </pre>	<pre> nazwisko stanowisko data_zatrudnienia -----+-----+----- JONAS KIEROWNIK 02-04-1981 JAMSKI URZEDNIK 03-12-1981 (2 rows) </pre>
<pre> SELECT nump as nr, nazwp AS nazwisko, Stanowisko, zatrud AS Data_Zatrudnienia FROM prac WHERE nump LIKE '77%'; </pre>	<pre> nr nazwisko stanowisko data_zatrudnienia -----+-----+-----+----- 7782 CELAREK KIEROWNIK 09-06-1981 7788 SKOTNIK ANALITYK 09-12-1981 (2 rows) </pre>

15.5.6 Porównywanie bez rozróżniania wielkości liter

Podczas porównywania napisów, czy o zmiennej długości, zawsze uwzględniana jest wielkość liter. Wobec tego dla SQL „Radecki” to nie to samo co „RADECKI”. Kiepskim rozwiązaniem jest przechowywanie w bazie wszystkich nazwisk zapisanych wielkimi literami. Lepszym rozwiązaniem jest przechowywanie nazwisk w możliwie poprawnej formie, a następnie używanie funkcji konwersji wielkości liter do robienia porównań.

SQL przedstawia następujące funkcje

Upper zamienia cały ciąg znaków na duże litery

Lower zamienia cały ciąg znaków na małe litery

15.5.6.1 Składnia

upper(kolumna) = 'ciąg_znakow_duzymi_literami'

przekształca ona małe litery na wielkie litery

lower(kolumna) = 'ciąg_znakow_malymi_literami'

przekształca ona duże litery na małe litery

Składnia	Wynik
SELECT nazwp, stanowisko FROM <i>prac</i> WHERE upper (stanowisko)='URZEDNIK';	nazwp stanowisko -----+----- SMITKO URZEDNIK ADAMCZYK URZEDNIK JAMSKI URZEDNIK MILLER URZEDNIK (4 rows)
SELECT nazwp, stanowisko FROM <i>prac</i> WHERE lower (stanowisko)='urzednik';	nazwp stanowisko -----+----- SMITKO URZEDNIK ADAMCZYK URZEDNIK JAMSKI URZEDNIK MILLER URZEDNIK (4 rows)

15.6 Obliczenia ze stałymi

Lista wyboru jest miejscem, gdzie umieszcza się informację, jakie obliczenia zostaną wykonane na danych liczbowych lub stałych. Dostępne są operatory arytmetyczne

+	Dodawanie	/	Dzielenie
-	Odejmowanie	*	mnożenie

15.6.1 W klauzuli SELECT

Składnia	Wynik																		
SELECT nazwp AS Nazwisko, Stanowisko, z arob*12 AS Kwota_roczna FROM <i>prac</i> WHERE stanowisko='URZEDNIK';	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>kwota_roczna</th> </tr> </thead> <tbody> <tr><td>SMITKO</td><td>URZEDNIK</td><td>9600.00</td></tr> <tr><td>ADAMCZYK</td><td>URZEDNIK</td><td>13200.00</td></tr> <tr><td>JAMSKI</td><td>URZEDNIK</td><td>11400.00</td></tr> <tr><td>MILLER</td><td>URZEDNIK</td><td>15600.00</td></tr> </tbody> </table> <p>(4 rows)</p>	Nazwisko	stanowisko	kwota_roczna	SMITKO	URZEDNIK	9600.00	ADAMCZYK	URZEDNIK	13200.00	JAMSKI	URZEDNIK	11400.00	MILLER	URZEDNIK	15600.00			
Nazwisko	stanowisko	kwota_roczna																	
SMITKO	URZEDNIK	9600.00																	
ADAMCZYK	URZEDNIK	13200.00																	
JAMSKI	URZEDNIK	11400.00																	
MILLER	URZEDNIK	15600.00																	
SELECT nazwp AS Nazwisko, Stanowisko, (z arob+prow)*12 AS wypłata_z_prowizja FROM <i>prac</i> WHERE stanowisko='URZEDNIK';	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>wypłata_z_prowizja</th> </tr> </thead> <tbody> <tr><td>SMITKO</td><td>URZEDNIK</td><td>13440.00</td></tr> <tr><td>ADAMCZYK</td><td>URZEDNIK</td><td></td></tr> <tr><td>JAMSKI</td><td>URZEDNIK</td><td></td></tr> <tr><td>MILLER</td><td>URZEDNIK</td><td></td></tr> </tbody> </table> <p>(4 rows)</p>	Nazwisko	stanowisko	wypłata_z_prowizja	SMITKO	URZEDNIK	13440.00	ADAMCZYK	URZEDNIK		JAMSKI	URZEDNIK		MILLER	URZEDNIK				
Nazwisko	stanowisko	wypłata_z_prowizja																	
SMITKO	URZEDNIK	13440.00																	
ADAMCZYK	URZEDNIK																		
JAMSKI	URZEDNIK																		
MILLER	URZEDNIK																		
SELECT nazwp AS Nazwisko, Stanowisko, (z arob+prow) AS wypłata_z_prowizja FROM <i>prac</i> WHERE stanowisko='SPRZEDAWCA';	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>wypłata_z_prowizja</th> </tr> </thead> <tbody> <tr><td>ALLEN</td><td>SPRZEDAWCA</td><td>1900.00</td></tr> <tr><td>WARD</td><td>SPRZEDAWCA</td><td>1750.00</td></tr> <tr><td>MARTIN</td><td>SPRZEDAWCA</td><td>2650.00</td></tr> <tr><td>TURNER</td><td>SPRZEDAWCA</td><td>1500.00</td></tr> </tbody> </table> <p>(4 rows)</p>	Nazwisko	stanowisko	wypłata_z_prowizja	ALLEN	SPRZEDAWCA	1900.00	WARD	SPRZEDAWCA	1750.00	MARTIN	SPRZEDAWCA	2650.00	TURNER	SPRZEDAWCA	1500.00			
Nazwisko	stanowisko	wypłata_z_prowizja																	
ALLEN	SPRZEDAWCA	1900.00																	
WARD	SPRZEDAWCA	1750.00																	
MARTIN	SPRZEDAWCA	2650.00																	
TURNER	SPRZEDAWCA	1500.00																	
SELECT nazwp AS Nazwisko, Stanowisko, Zarob + prow AS placa_brutto FROM <i>prac</i> WHERE (zarob + prow) > 0;	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>placa_brutto</th> </tr> </thead> <tbody> <tr><td>ALLEN</td><td>SPRZEDAWCA</td><td>1900.00</td></tr> <tr><td>WARD</td><td>SPRZEDAWCA</td><td>1750.00</td></tr> <tr><td>MARTIN</td><td>SPRZEDAWCA</td><td>2650.00</td></tr> <tr><td>TURNER</td><td>SPRZEDAWCA</td><td>1500.00</td></tr> <tr><td>ADAMCZYK</td><td>URZEDNIK</td><td>1120.00</td></tr> </tbody> </table> <p>(5 rows)</p>	Nazwisko	stanowisko	placa_brutto	ALLEN	SPRZEDAWCA	1900.00	WARD	SPRZEDAWCA	1750.00	MARTIN	SPRZEDAWCA	2650.00	TURNER	SPRZEDAWCA	1500.00	ADAMCZYK	URZEDNIK	1120.00
Nazwisko	stanowisko	placa_brutto																	
ALLEN	SPRZEDAWCA	1900.00																	
WARD	SPRZEDAWCA	1750.00																	
MARTIN	SPRZEDAWCA	2650.00																	
TURNER	SPRZEDAWCA	1500.00																	
ADAMCZYK	URZEDNIK	1120.00																	
SELECT nazwp AS Nazwisko, Stanowisko, Zarob + prow AS placa_brutto FROM <i>prac</i> WHERE prow < (0.25 * zarob);	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>placa_brutto</th> </tr> </thead> <tbody> <tr><td>ALLEN</td><td>SPRZEDAWCA</td><td>1900</td></tr> <tr><td>TURNER</td><td>SPRZEDAWCA</td><td>1500</td></tr> <tr><td>ADAMCZYK</td><td>URZEDNIK</td><td>1120</td></tr> </tbody> </table> <p>(3 rows)</p>	Nazwisko	stanowisko	placa_brutto	ALLEN	SPRZEDAWCA	1900	TURNER	SPRZEDAWCA	1500	ADAMCZYK	URZEDNIK	1120						
Nazwisko	stanowisko	placa_brutto																	
ALLEN	SPRZEDAWCA	1900																	
TURNER	SPRZEDAWCA	1500																	
ADAMCZYK	URZEDNIK	1120																	
SELECT nazwp AS Nazwisko, Stanowisko, Zarob + prow AS placa_brutto FROM <i>prac</i> WHERE prow < (0.25 * zarob);	ERROR: Unable to identify an operator '*' fro types 'float8' and 'numeric' You will have to retype this query using an explicit cast																		

15.6.2 W klauzuli WHERE

Składnia	Wynik																		
SELECT nazwp AS Nazwisko, Stanowisko, Zarob + prow AS placa_brutto FROM <i>prac</i> WHERE (zarob + prow) > 0;	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>placa_brutto</th> </tr> </thead> <tbody> <tr><td>ALLEN</td><td>SPRZEDAWCA</td><td>1900.00</td></tr> <tr><td>WARD</td><td>SPRZEDAWCA</td><td>1750.00</td></tr> <tr><td>MARTIN</td><td>SPRZEDAWCA</td><td>2650.00</td></tr> <tr><td>TURNER</td><td>SPRZEDAWCA</td><td>1500.00</td></tr> <tr><td>ADAMCZYK</td><td>URZEDNIK</td><td>1120.00</td></tr> </tbody> </table> <p>(5 rows)</p>	Nazwisko	stanowisko	placa_brutto	ALLEN	SPRZEDAWCA	1900.00	WARD	SPRZEDAWCA	1750.00	MARTIN	SPRZEDAWCA	2650.00	TURNER	SPRZEDAWCA	1500.00	ADAMCZYK	URZEDNIK	1120.00
Nazwisko	stanowisko	placa_brutto																	
ALLEN	SPRZEDAWCA	1900.00																	
WARD	SPRZEDAWCA	1750.00																	
MARTIN	SPRZEDAWCA	2650.00																	
TURNER	SPRZEDAWCA	1500.00																	
ADAMCZYK	URZEDNIK	1120.00																	
SELECT nazwp AS Nazwisko, Stanowisko, Zarob + prow AS placa_brutto	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>placa_brutto</th> </tr> </thead> <tbody> <tr><td>ALLEN</td><td>SPRZEDAWCA</td><td>1900</td></tr> <tr><td>TURNER</td><td>SPRZEDAWCA</td><td>1500</td></tr> </tbody> </table>	Nazwisko	stanowisko	placa_brutto	ALLEN	SPRZEDAWCA	1900	TURNER	SPRZEDAWCA	1500									
Nazwisko	stanowisko	placa_brutto																	
ALLEN	SPRZEDAWCA	1900																	
TURNER	SPRZEDAWCA	1500																	

Składnia	Wynik
FROM <i>prac</i> WHERE <i>prow</i> < (0.25 * <i>zarob</i>);	ADAMCZYK URZEDNIK 1120 (3 rows)
SELECT <i>nazwp</i> AS Nazwisko, <i>Stanowisko</i> , <i>Zarob</i> + <i>prow</i> AS <i>placa_brutto</i> FROM <i>prac</i> WHERE <i>prow</i> < (0.25 * <i>zarob</i>);	ERROR: Unable to identify an operator ‘*’ fro types ‘float8’ and ‘numeric’ You will have to retype this query using an explicit cast

15.6.3 W klauzuli ORDER BY

Składnia	Wynik															
SELECT <i>nazwp</i> AS Nazwisko, <i>Zarob</i> AS Placa, <i>Zarob</i> - <i>prow</i> FROM <i>prac</i> WHERE <i>stanowisko</i> = 'SPRZEDAWCA' ORDER BY <i>zarob</i> - <i>prow</i> ;	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>placa</th> <th>?column?</th> </tr> </thead> <tbody> <tr> <td>MARTIN</td> <td> 1250.00 </td> <td>-150.00</td> </tr> <tr> <td>WARD</td> <td> 1250.00 </td> <td>750.00</td> </tr> <tr> <td>ALLEN</td> <td> 1600.00 </td> <td>1300.00</td> </tr> <tr> <td>TURNER</td> <td> 1500.00 </td> <td>1500.00</td> </tr> </tbody> </table> (4 rows)	Nazwisko	placa	?column?	MARTIN	1250.00	-150.00	WARD	1250.00	750.00	ALLEN	1600.00	1300.00	TURNER	1500.00	1500.00
Nazwisko	placa	?column?														
MARTIN	1250.00	-150.00														
WARD	1250.00	750.00														
ALLEN	1600.00	1300.00														
TURNER	1500.00	1500.00														

16 Grupowanie danych

Klauzula GROUP BY jest blisko związana z agregowaniem. Istotnie, z GROUP BY nie ma zbyt wiele pożytku bez zastosowania funkcji agregujących. GROUP BY dzieli tabele na zbiory, a następnie funkcje agregujące produkują summaryczne wyniki dla każdego z tych zbiorów. Otrzymane wartości nazywamy agregatami wektorowymi.

Uwaga: Agregat skalarny to pojedyncza wartość wytworzona przez funkcję agregującą.

16.1 Składnia GROUP BY

SELECT lista_wyboru

FROM lista_tabel

[**WHERE** warunki]

[**GROUP BY** lista_grupowania]

[**HAVING** warunek_dla_grupy]

[**ORDER BY** lista_porzadkowa]

Gdy używasz **GROUP BY**, klauzula **SELECT** może zawierać tylko trzy rodzaje wyrażeń:

Nazwy kolumny występująca na liście GROUP BY

Funkcje summaryczne (SUM, AVG, MIN, MAX i COUNT0 dotyczące innych kolumn wtabelach

Stałe wyrażenia, takie jak zł

Użyteczną klauzulą towarzyszącą klauzuli **GROUP BY** jest **HAVING**. Podczas gdy klauzula **WHERE** jest do wybierania wierszy, klauzula **HAVING** służy do wybierania grup, które mają się pojawiać w wyniku.

17 Funkcje agregujące

Agregaty to funkcje umożliwiające otrzymywanie wartości sumarycznych.

Stosowanie się je do:

zbiorów wierszy:

wszystkich wierszy tabeli

wierszy określonych przez klauzulę WHERE (nie należy stosować w klauzule WHERE funkcji agregujących)

grup wierszy w tabeli utworzonych przez klauzulę GROUP BY

Bez względu na to, jaką strukturę nada się tym zbiorom, dla każdego zbioru wierszy otrzymuje się pojedynczą wartość

17.1 Składnia funkcji agregującej

Ponieważ agregaty to funkcje, więc zawsze pobierają argument, którym jest wyrażenie zawarte w nawiasach.

Składnia jest następująca:

Funkcja_agregująca ([DISTINCT] wyrażenie)

Wyrażenie w definicji składni funkcji agregującej często jest :

nazwą kolumny,

ale może być również stałą, funkcją lub

dowolną kombinacją nazw kolumn, stałych i funkcji połączonych operatorami arytmetycznymi (a w niektórych systemach operatorami bitowymi).

Należy zaznaczyć, że **SUM** i **AVG** można tylko używać dla danych liczbowych. **MIN**, **MAX**, **COUNT** i **COUNT(*)** działają ze wszystkimi typami danych.

Uwaga: **DISTINCT** można używać z dowolną funkcją agregującą oprócz **COUNT(*)** oraz zastosowanie przy **MIN** i **MAX** nie zmienia wyniku.

Funkcja agregująca	Wynik
SUM ([DISTINCT] wyrażenie)	Suma (różniących się) wartości w wyrażeniu numerycznym)
AVG ([DISTINCT] wyrażenie)	Wartość średnia (różniących się) wartości w wyrażeniu numerycznym
COUNT ([DISTINCT] wyrażenie)	Liczba (różniących się) wartości (różnych od NULL) w wyrażeniu
COUNT (*)	Liczba wybranych wierszy
MAX (wyrażenie)	Największa wartość wyrażenia
MIN (wyrażenie)	Najmniejsza wartość wyrażenia

17.1.1.1 Funkcje agregujące i WHERE

Funkcje agregujące można stosować na liście wyboru, lub w klauzuli **HAVING** instrukcji **SELECT**. W klauzuli **WHERE** funkcji agregujących używać nie wolno.

Jeśli jednak tak się zdarzy, to zostanie zgłoszony błąd. Można jednakże zastosować klauzulę **WHERE** do ograniczenia wierszy rozpatrywanych w obliczeniach agregujących.

17.1.1.2 Wartość null i funkcje agregujące

Jeśli w kolumnie, na której operuje funkcja agregująca, występują jakiegokolwiek wartości null, to są one ignorowane.

Uwaga: wyjątkiem od tej zasady jest **COUNT(*)**, które zlicza każdy wiersz niezależnie od tego, czy wartością kolumny jest **NULL**, czy nie.

17.1.2 Klauzula count(*) i count

Zadanie Nr 1 Ilu pracowników jest na poszczególnych stanowiska. Ilu otrzymuje zarobki i prowizję

<pre>SELECT stanowisko, count(*) AS il_calosc, count(zarob) AS il_zarobki, count(prow) AS il_prowizja FROM prac GROUP BY stanowisko;</pre>	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>il_calosc</th> <th>il_zarobki</th> <th>il_prowizja</th> </tr> </thead> <tbody> <tr> <td>ANALITYK</td> <td>2</td> <td>2</td> <td>0</td> </tr> <tr> <td>KIEROWNIK</td> <td>3</td> <td>3</td> <td>0</td> </tr> <tr> <td>PREZES</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>SPRZEDAWCA</td> <td>4</td> <td>4</td> <td>4</td> </tr> <tr> <td>URZEDNIK</td> <td>4</td> <td>4</td> <td>1</td> </tr> </tbody> </table> <p>(5 rows)</p>	Stanowisko	il_calosc	il_zarobki	il_prowizja	ANALITYK	2	2	0	KIEROWNIK	3	3	0	PREZES	1	1	0	SPRZEDAWCA	4	4	4	URZEDNIK	4	4	1
Stanowisko	il_calosc	il_zarobki	il_prowizja																						
ANALITYK	2	2	0																						
KIEROWNIK	3	3	0																						
PREZES	1	1	0																						
SPRZEDAWCA	4	4	4																						
URZEDNIK	4	4	1																						

Zadanie Nr 2 Ilu jest pracowników na stanowisku KIEROWNIK

Składnia	Wynik
<pre>SELECT COUNT(*) AS ilosc FROM prac WHERE stanowisko = 'KIEROWNIK';</pre>	<p>Ilość</p> <p>-----</p> <p>3</p> <p>(1 row)</p>

Zadanie Nr 3 Ilu jest wszystkich pracowników zatrudnionych w Naszym zakładzie pracy

Składnia	Wynik
<pre>SELECT COUNT(*) AS ilosc FROM prac;</pre>	<p>Ilość</p> <p>-----</p> <p>14</p> <p>(1 row)</p>

Zadanie Nr 4 Ilu jest pracowników na poszczególnych stanowiskach

Składnia	Wynik												
<pre>SELECT stanowisko, COUNT(*) AS ilosc FROM prac GROUP BY stanowisko ORDER BY 2;</pre>	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>ilosc</th> </tr> </thead> <tbody> <tr> <td>PREZES</td> <td>1</td> </tr> <tr> <td>ANALITYK</td> <td>2</td> </tr> <tr> <td>KIEROWNIK</td> <td>3</td> </tr> <tr> <td>SPRZEDAWCA</td> <td>4</td> </tr> <tr> <td>URZEDNIK</td> <td>4</td> </tr> </tbody> </table> <p>(5 rows)</p>	Stanowisko	ilosc	PREZES	1	ANALITYK	2	KIEROWNIK	3	SPRZEDAWCA	4	URZEDNIK	4
Stanowisko	ilosc												
PREZES	1												
ANALITYK	2												
KIEROWNIK	3												
SPRZEDAWCA	4												
URZEDNIK	4												

Zadanie Nr 5 Dla każdego działu wypisz wszystkie stanowiska oraz liczbę pracowników na tych stanowiskach (cout_01.sql)

Składnia	Wynik																														
<pre>SELECT stanowisko, numdz, count(*) FROM prac GROUP BY stanowisko, numdz;</pre>	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>numdz</th> <th>Mount</th> </tr> </thead> <tbody> <tr> <td>ANALITYK</td> <td>20</td> <td>2</td> </tr> <tr> <td>KIEROWNIK</td> <td>10</td> <td>1</td> </tr> <tr> <td>KIEROWNIK</td> <td>20</td> <td>1</td> </tr> <tr> <td>KIEROWNIK</td> <td>30</td> <td>1</td> </tr> <tr> <td>PREZES</td> <td>10</td> <td>1</td> </tr> <tr> <td>SPRZEDAWCA</td> <td>30</td> <td>4</td> </tr> <tr> <td>URZEDNIK</td> <td>10</td> <td>1</td> </tr> <tr> <td>URZEDNIK</td> <td>20</td> <td>1</td> </tr> <tr> <td>URZEDNIK</td> <td>30</td> <td>2</td> </tr> </tbody> </table> <p>(9 rows)</p>	Stanowisko	numdz	Mount	ANALITYK	20	2	KIEROWNIK	10	1	KIEROWNIK	20	1	KIEROWNIK	30	1	PREZES	10	1	SPRZEDAWCA	30	4	URZEDNIK	10	1	URZEDNIK	20	1	URZEDNIK	30	2
Stanowisko	numdz	Mount																													
ANALITYK	20	2																													
KIEROWNIK	10	1																													
KIEROWNIK	20	1																													
KIEROWNIK	30	1																													
PREZES	10	1																													
SPRZEDAWCA	30	4																													
URZEDNIK	10	1																													
URZEDNIK	20	1																													
URZEDNIK	30	2																													

W tym wypadku głównym zadaniem klauzuli **GROUP BY** jest wykonywanie operacji sumarycznych (operacjami sumarycznymi są funkcje na kolumnach, takie jak **SUM**, **AVG**, **MIN**, **MAX**, i **COUNT**) Kiedy używa się **GROUP BY**, najpierw tworzą się grupy, a później są wykonywane operacje sumaryczne w odniesieniu do grup.

Do utworzenia grup wierszy mające te same wartości **STANOWISKO** i **NUMDZ** system SQL wykonuje wewnętrzny proces sortowania (podobny do **ORDER BY, STANOWISKO, NUMDZ**). W wyniku sortowania wiersze z identycznymi wartościami **STANOWISKO** i **NUMDZ** są zaliczone do jednej grupy, po czym dla każdej grupy jest obliczona funkcja kolumnowa **COUNT(*)**.

Można tworzyć grupy ze wszystkich wierszy w tabeli (lub złączenia tabel) albo można ograniczyć się do podzbioru wierszy, korzystając z klauzuli **WHERE**.

Klauzula **SELECT** może zawierać niektóre lub nawet wszystkie z podanych tu pozycji:

Kolumny o nazwach występujących na liście kolumn **GROUP BY**

Funkcje kolumnowe dotyczące innych kolumn w tabeli

Wyrażenia stałe

Zadanie Nr 6 Sprawdź, które stanowiska w których działach są obsadzone przez dwóch lub więcej pracowników (count_02.sql)

Składnia	Wynik												
SELECT stanowisko, numdz, count(*) FROM prac GROUP BY stanowisko, numdz HAVING count(*) >=2;	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>numdz</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>ANALITYK</td> <td>20</td> <td>2</td> </tr> <tr> <td>SPRZEDAWCA</td> <td>30</td> <td>4</td> </tr> <tr> <td>URZEDNIK</td> <td>30</td> <td>2</td> </tr> </tbody> </table> (3 rows)	Stanowisko	numdz	count	ANALITYK	20	2	SPRZEDAWCA	30	4	URZEDNIK	30	2
Stanowisko	numdz	count											
ANALITYK	20	2											
SPRZEDAWCA	30	4											
URZEDNIK	30	2											

Bez ostatniej klauzuli (**HAVING**) otrzymalibyśmy wyniki dla wszystkich grup. Chcemy jednak ograniczyć grupy, które są dołączone do wyniku. Klauzula **HAVING** używana jest do wybierania potrzebnych grup po ich utworzeniu (robi się to podobnie jak w przypadku klauzuli **WHERE**, za pomocą której wybiera się wiersze).

Zadanie Nr 7 Przedstaw sumę pensji dla tych wydziałów, których wartość przekracza 2000 zł (count_03.sql)

Składnia	Wynik												
SELECT d.numdz, d.nazwdz , sum(p.zarob) FROM prac P, dzial D WHERE d.numdz = p.numdz AND p.zarob > 2000 GROUP BY d.numdz, d.nazwdz;	<table border="1"> <thead> <tr> <th>Numdz</th> <th>nazwdz</th> <th>sum</th> </tr> </thead> <tbody> <tr> <td>10</td> <td> KSIEGOWOSC</td> <td> 7450.00</td> </tr> <tr> <td>20</td> <td> BADANIA</td> <td> 8975.00</td> </tr> <tr> <td>30</td> <td> SPRZEDAZ</td> <td> 2850.00</td> </tr> </tbody> </table> (3 rows)	Numdz	nazwdz	sum	10	KSIEGOWOSC	7450.00	20	BADANIA	8975.00	30	SPRZEDAZ	2850.00
Numdz	nazwdz	sum											
10	KSIEGOWOSC	7450.00											
20	BADANIA	8975.00											
30	SPRZEDAZ	2850.00											

17.1.3 Klauzula max i min

Składnia	Wynik												
SELECT stanowisko, MIN (zarob) AS min FROM prac GROUP BY stanowisko;	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>min</th> </tr> </thead> <tbody> <tr> <td>ANALITYK</td> <td>3000.00</td> </tr> <tr> <td>KIEROWNIK</td> <td>2450.00</td> </tr> <tr> <td>PREZES</td> <td>5000.00</td> </tr> <tr> <td>SPRZEDAWCA</td> <td>1250.00</td> </tr> <tr> <td>URZEDNIK</td> <td>800.00</td> </tr> </tbody> </table> (5 rows)	Stanowisko	min	ANALITYK	3000.00	KIEROWNIK	2450.00	PREZES	5000.00	SPRZEDAWCA	1250.00	URZEDNIK	800.00
Stanowisko	min												
ANALITYK	3000.00												
KIEROWNIK	2450.00												
PREZES	5000.00												
SPRZEDAWCA	1250.00												
URZEDNIK	800.00												

SELECT stanowisko, MAX (zarob) AS min FROM <i>prac</i> GROUP BY stanowisko;	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>max</th> </tr> </thead> <tbody> <tr><td>ANALITYK</td><td>3000.00</td></tr> <tr><td>KIEROWNIK</td><td>2975.00</td></tr> <tr><td>PREZES</td><td>5000.00</td></tr> <tr><td>SPRZEDAWCA</td><td>1600.00</td></tr> <tr><td>URZEDNIK</td><td>1300.00</td></tr> </tbody> </table> (5 rows)	Stanowisko	max	ANALITYK	3000.00	KIEROWNIK	2975.00	PREZES	5000.00	SPRZEDAWCA	1600.00	URZEDNIK	1300.00												
Stanowisko	max																								
ANALITYK	3000.00																								
KIEROWNIK	2975.00																								
PREZES	5000.00																								
SPRZEDAWCA	1600.00																								
URZEDNIK	1300.00																								
SELECT nazwdz, stanowisko, zarob FROM <i>prac</i> P , <i>dzial</i> D WHERE p.numdz = d.numdz AND nazwdz = 'SPRZEDAZ' ;	<table border="1"> <thead> <tr> <th>nazwdz</th> <th>stanowisko</th> <th>zarob</th> </tr> </thead> <tbody> <tr><td>SPRZEDAZ</td><td>SPRZEDAWCA</td><td>1600.00</td></tr> <tr><td>SPRZEDAZ</td><td>SPRZEDAWCA</td><td>1250.00</td></tr> <tr><td>SPRZEDAZ</td><td>SPRZEDAWCA</td><td>1250.00</td></tr> <tr><td>SPRZEDAZ</td><td>KIEROWNIK</td><td>2850.00</td></tr> <tr><td>SPRZEDAZ</td><td>SPRZEDAWCA</td><td>1500.00</td></tr> <tr><td>SPRZEDAZ</td><td>URZEDNIK</td><td>1100.00</td></tr> <tr><td>SPRZEDAZ</td><td>URZEDNIK</td><td>950.00</td></tr> </tbody> </table> (7 rows)	nazwdz	stanowisko	zarob	SPRZEDAZ	SPRZEDAWCA	1600.00	SPRZEDAZ	SPRZEDAWCA	1250.00	SPRZEDAZ	SPRZEDAWCA	1250.00	SPRZEDAZ	KIEROWNIK	2850.00	SPRZEDAZ	SPRZEDAWCA	1500.00	SPRZEDAZ	URZEDNIK	1100.00	SPRZEDAZ	URZEDNIK	950.00
nazwdz	stanowisko	zarob																							
SPRZEDAZ	SPRZEDAWCA	1600.00																							
SPRZEDAZ	SPRZEDAWCA	1250.00																							
SPRZEDAZ	SPRZEDAWCA	1250.00																							
SPRZEDAZ	KIEROWNIK	2850.00																							
SPRZEDAZ	SPRZEDAWCA	1500.00																							
SPRZEDAZ	URZEDNIK	1100.00																							
SPRZEDAZ	URZEDNIK	950.00																							
SELECT nazwdz, Stanowisko , MAX (zarob) FROM <i>prac</i> P , <i>dzial</i> D WHERE p.numdz = d.numdz AND nazwdz = 'SPRZEDAZ' GROUP BY nazwdz, stanowisko ;	<table border="1"> <thead> <tr> <th>nazwdz</th> <th>stanowisko</th> <th>max</th> </tr> </thead> <tbody> <tr><td>SPRZEDAZ</td><td>KIEROWNIK</td><td>2850.00</td></tr> <tr><td>SPRZEDAZ</td><td>SPRZEDAWCA</td><td>1600.00</td></tr> <tr><td>SPRZEDAZ</td><td>URZEDNIK</td><td>1100.00</td></tr> </tbody> </table> (3 rows)	nazwdz	stanowisko	max	SPRZEDAZ	KIEROWNIK	2850.00	SPRZEDAZ	SPRZEDAWCA	1600.00	SPRZEDAZ	URZEDNIK	1100.00												
nazwdz	stanowisko	max																							
SPRZEDAZ	KIEROWNIK	2850.00																							
SPRZEDAZ	SPRZEDAWCA	1600.00																							
SPRZEDAZ	URZEDNIK	1100.00																							
SELECT stanowisko, MAX (zarob) – MIN (zarob) AS min_max FROM <i>prac</i> GROUP BY stanowisko ORDER BY 2;	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>min_max</th> </tr> </thead> <tbody> <tr><td>ANALITYK</td><td>0.00</td></tr> <tr><td>PREZES</td><td>0.00</td></tr> <tr><td>SPRZEDAWCA</td><td>350.00</td></tr> <tr><td>URZEDNIK</td><td>500.00</td></tr> <tr><td>KIEROWNIK</td><td>525.00</td></tr> </tbody> </table> (5 rows)	Stanowisko	min_max	ANALITYK	0.00	PREZES	0.00	SPRZEDAWCA	350.00	URZEDNIK	500.00	KIEROWNIK	525.00												
Stanowisko	min_max																								
ANALITYK	0.00																								
PREZES	0.00																								
SPRZEDAWCA	350.00																								
URZEDNIK	500.00																								
KIEROWNIK	525.00																								

17.1.4 Klauzula AVG

Składnia	Wynik												
SELECT stanowisko, AVG (zarob) AS srednie_zarobki FROM <i>prac</i> WHERE stanowisko != 'PREZES' GROUP BY stanowisko HAVING AVG (zarob) < 2300;	<table border="1"> <thead> <tr> <th>Stanowisko</th> <th>srednie_zarobki</th> </tr> </thead> <tbody> <tr><td>SPRZEDAWCA</td><td>1400.0000000000</td></tr> <tr><td>URZEDNIK</td><td>1037.5000000000</td></tr> </tbody> </table> (2 ROWS)	Stanowisko	srednie_zarobki	SPRZEDAWCA	1400.0000000000	URZEDNIK	1037.5000000000						
Stanowisko	srednie_zarobki												
SPRZEDAWCA	1400.0000000000												
URZEDNIK	1037.5000000000												
SELECT stanowisko, AVG (zarob) AS srednia FROM <i>prac</i> GROUP BY stanowisko;	<table border="1"> <thead> <tr> <th>stanowisko</th> <th>srednia</th> </tr> </thead> <tbody> <tr><td>ANALITYK</td><td>3000.0000000000</td></tr> <tr><td>KIEROWNIK</td><td>2758.3333333333</td></tr> <tr><td>PREZES</td><td>5000.0000000000</td></tr> <tr><td>SPRZEDAWCA</td><td>1400.0000000000</td></tr> <tr><td>URZEDNIK</td><td>1037.5000000000</td></tr> </tbody> </table> (5 rows)	stanowisko	srednia	ANALITYK	3000.0000000000	KIEROWNIK	2758.3333333333	PREZES	5000.0000000000	SPRZEDAWCA	1400.0000000000	URZEDNIK	1037.5000000000
stanowisko	srednia												
ANALITYK	3000.0000000000												
KIEROWNIK	2758.3333333333												
PREZES	5000.0000000000												
SPRZEDAWCA	1400.0000000000												
URZEDNIK	1037.5000000000												

18 PODZAPYTANIA

Stosowanie podzapytań to dodatkowa metoda służąca do jednoczesnego operowania na wielu tabelach.

Instrukcja SELECT może być zagnieżdżona:

W klauzulach WHERE, HAVING lub SELECT innej instrukcji SELECT

W klauzulach INSERT, UPDATE lub DELETE

W innych podzapytaniach

Możliwość zagnieżdżania instrukcji SQL jest powodem, dla którego SQL nazwano strukturalnym językiem zapytań (Structured Query Language)

18.1 Działanie podzapytań

Podzapytania przekazują wyniki z zapytania wewnętrznego do klauzuli zewnętrznej, występujące w dwóch rodzajach, jako podzapytania: nieskorelowane i skorelowane.

Podzapytanie nieskorelowane rozpoczyna pracę (pojęciowo) od wewnętrznej instrukcji SQL. Oznacza to, że zapytanie zewnętrzne podejmuje działanie na podstawie wyników zapytania wewnętrznego

O podzapytaniu skorelowanym można myśleć jako o podzapytaniu działającym w przeciwną stronę: zewnętrzna instrukcja SQL dostarcza wartości dla zapytania wewnętrznego, aby mogło ono ich używać w swoich obliczeniach

18.1.1 Podział podzapytań

Oba podzapytania, skorelowane i nieskorelowane, dzielą się na trzy typy w zależności od elementów w klauzuli WHERE zapytania zewnętrznego:

Podzapytania zwracające zero lub więcej pozycji (wprowadzone za pomocą IN lub operatora porównania zmodyfikowanego przez ANY bądź ALL).

Podzapytania zwracające pojedynczą wartość (wprowadzone za pomocą niezmodyfikowanego operatora porównania)

Podzapytania będące testem na istnienie (wprowadzone za pomocą EXISTS)

SQL: nieskorelowane SELECT pub_name FROM publisher WHERE pub_id IN (SELECT pub_id FROM titles WHERE type = 'business')	SQL : skorelowane SELECT pub_name FROM publisher p WHERE 'business' IN (SELECT type FROM titles t WHERE t.pub_id = p.pub_id)
SQL:nieskorelowane SELECT nazwp, stanowisko FROM prac WHERE numdz IN (SELECT numdz FROM dzial WHERE lok = 'CHICAGO');	SQL : skorelowane SELECT nazwp, stanowisko FROM prac P WHERE 'CHICAGO' IN (SELECT lok FROM dzial D WHERE P.numdz = D.numdz);

18.2 Reguły – podzapytania

Lista wyboru podzapytania wewnętrznego wprowadzona przez operatorów porównania lub **IN** może zawierać tylko jedno wyrażenie lub jedną nazwę kolumny. Nazwa kolumny w klauzuli WHERE instrukcji zewnętrznej musi być złączeniowo zgodna z nazwą kolumny na liście wyboru podzapytania ... **WHERE** **pub_id** **IN** (**SELECT** **pub_id**

Lista wyboru podzapytania wprowadzonego przez **EXISTS** prawie zawsze składa się z gwiazdki (*). Nie ma potrzeby określania nazw kolumn, gdyż sprawdzane jest tylko istnienie (lub nieistnienie) wierszy spełniających nałożone kryteria. (wiersze można zakwalifikować w klauzuli WHERE podzapytania). Reguły rządzące listą wyboru podzapytania wprowadzonego przez **EXISTS** są pod każdym względem identyczne z tymi dla standardowej listy wyboru.

Podzapytania wprowadzone przez niezmodyfikowany operator porównania (czyli operator porównania bez następującego po nim słowa kluczowego **ANY** lub **ALL**) nie może zawierać klauzul **GROUP BY** i **HAVING**, chyba że z góry zrobi się założenie, iż grupowanie zwraca pojedynczą wartość.

Podzapytania nie mogą wewnątrz operować na swoich wynikach. Oznacza to, że podzapytanie nie może zawierać klauzuli **ORDER BY** lub słowa kluczowego **INTO**. Opcjonalnie słowo **DISTINCT** może efektywnie porządkować wyniki podzapytania, gdyż niektóre systemy eliminują powtórzenia przy pierwszym porządkowaniu wyników.

Przegląd trzech głównych typów połączeń podzapytań wyjaśni te ograniczenia i powody ich istnienia.

Jak było powiedziane wcześniej, połączenia te dotyczą:

Podzapytań zwracających zero lub więcej elementów (wprowadzających za pomocą **IN** lub operatora porównania zmodyfikowanego przez **ANY** bądź **ALL**)

Podzapytań zwracających pojedynczą wartość (wprowadzonych za pomocą niezmodyfikowanego operatora porównania)

Podzapytań będących testem na istnienie (wprowadzonych za pomocą **EXISTS**)

Używaj nawiasów, aby zmusić system do robienia dokładnie tego co chcesz.

Możesz również postawić **NOT** przed warunkiem wyszukiwania ujętym w nawiasy np.

... **AND NOT** (predykat_1 **OR** predykat_2)

... **WHERE NOT** (predykat_1 **AND** predykat_2)

Jest ważna zasada, o której musisz pamiętać podczas używania podzapytań. Otóż lewa strona predykatu wymaga zgodności wyniku z podzapytaniem po prawej stronie. Jeśli na przykład w predykacie jest operator logiczny, to podzapytanie musi zwrócić pojedynczą wartość.

Liczba \bowtie (podzapytanie zwraca pojedynczą liczbę)

Napis = (podzapytanie zwracające pojedynczy napis)

Inne zasady dotyczą operatora **IN**, gdy używa się go z podzapytaniem. Obowiązują one zarówno wtedy, kiedy podzapytanie daje jedną wartość, jak i wtedy, kiedy daje ono więcej wartości. Dopuszczalny jest nawet brak wartości (rezultat **NULL**). Sytuacja taka oznacza, że warunek logiczny jest fałszywy i że dany wiersz nie zostanie zaliczony do wyniku.

Liczba **IN** $>$ (podzapytanie zwraca listę liczb)

Napis **IN** = (podzapytanie zwraca listę napisów)

Należy zaznaczyć że

liczba > (lista liczb)

nie jest poprawna. Możliwe jest jednak uniknięcie niepoprawnych zapytań tego typu. Możesz używać słowa kluczowego ANY (służy do porównania wartości z każdą wartością zwracaną przez podzapytanie)

Liczba > ANY (lista liczb)

18.2.1 Podzapytanie ... IN

18.2.1.1 Składnia

Początek instrukcji

SELECT, INSERT, UPDATE, DELETE;

lub podzapytanie

WHERE wyrażenie [NOT] **IN** (*podzapytanie*)

[koniec instrukcji **SELECT**, INSERT, UPDATE, DELETE; lub podzapytanie]

WHERE wyrażenia [NOT] IN (podzapytanie)

Wynikiem podzapytania wewnętrznego jest lista zawierająca zero lub więcej wartości. Jeśli podzapytanie zwraca już wyniki, to wykorzystuje je zapytanie zewnętrzne.

Składnia	Wynik
SELECT numdz AS ID, nazwdz AS dzial, lok AS miasto FROM <i>dzial</i> WHERE lok IN ('DALLAS', 'BOSTON');	Id dzial miasto -----+-----+----- 20 BADANIA DALLAS 40 OPERACJE BOSTON (2 rows)
SELECT numdz AS ID, nazwdz AS dzial, lok AS miasto FROM <i>dzial</i> WHERE lok='DALLAS' OR lok='BOSTON';	Id dzial miasto -----+-----+----- 20 BADANIA DALLAS 40 OPERACJE BOSTON (2 rows)
SELECT numdz AS ID, nazwdz AS dzial, lok AS miasto FROM <i>dzial</i> WHERE lok NOT IN ('DALLAS', 'BOSTON');	Id dzial miasto -----+-----+----- 10 KSIEGOWOSC NOWY YORK 30 SPRZEDAZ CHICAGO (2 rows)

Zadanie Nr 8 Wyświetl wszystkie dane personalne pracowników pracujących w DALLAS

Składnia podzapytania		Składnia złączenia	
<pre>SELECT * FROM prac WHERE numdz IN (SELECT numdz FROM dzial WHERE lok='DALLAS');</pre>	<pre>SELECT * FROM prac P, dzial D WHERE P.numdz = D.numdz AND D.lok='DALLAS';</pre>		
Wynik			
Nump	nazwp	stanowisko	kier zatrud zarob prow numdz
7369	SMITKO	URZEDNIK	7902 12-17-1980 800.00 20 20
7566	JONAS	KIEROWNIK	7839 02-04-1981 2975.00 20 20
7788	SKOTNIK	ANALITYK	7566 09-12-1981 3000.00 20 20
7902	FORD	ANALITYK	7566 03-12-1981 3000.00 20 20
(4 rows)			
Nump	nazwp	stanowisko	kier zatrud zarob prow numdz numdz nazwdz lok
7369	SMITKO	URZEDNIK	7902 12-17-1980 800.00 20 20 BADANIA DALLAS
7566	JONAS	KIEROWNIK	7839 02-04-1981 2975.00 20 20 BADANIA DALLAS
7788	SKOTNIK	ANALITYK	7566 09-12-1981 3000.00 20 20 BADANIA DALLAS
7902	FORD	ANALITYK	7566 03-12-1981 3000.00 20 20 BADANIA DALLAS
(4 rows)			

Jeśli w wyniku działania podzapytania dochodzi do przekazywania predykatowi w głównym zapytaniu wartości kolumny lub listy wartości, to główne zapytanie może być przekształcone w złączenie bez użycia podzapytania.

Zadanie Nr 9 Czy jest pracownik pracujący w DALLAS, który otrzymał maksymalną premię uznaniową przydzielaną w Zakładzie pracy

Składnia (podzapytanie)	
<pre>SELECT numdz, nazwp, stanowisko, zatrud, zarob FROM prac WHERE numdz IN (SELECT numdz FROM dzial WHERE lok='DALLAS') AND zarob =(SELECT MAX(zarob) FROM premia);</pre>	
Wynik	
<pre>Numdz Nazwp stanowisko zatrud zarob -----+-----+-----+-----+----- 20 JONAS KIEROWNIK 02-04-1981 2975.00 (1 row)</pre>	
Składnia	Wynik
<pre>SELECT MAX(zarob) FROM premia;</pre>	<pre>Max ----- 2975.00 (1 row)</pre>
<pre>SELECT numdz FROM dzial WHERE lok='DALLAS'</pre>	<pre>numdz ----- 20 (1 row)</pre>
<pre>SELECT numdz, Nazwp, Stanowisko, Zatrud, Zarob FROM prac</pre>	<pre>Numdz nazwp stanowisko zatrud zarob -----+-----+-----+-----+----- 20 SMITKO URZEDNIK 12-17-1980 800.00 30 ALLEN SPRZEDAWCA 02-20-1981 1600.00 30 WARD SPRZEDAWCA 02-22-1981 1250.00 20 JONAS KIEROWNIK 02-04-1981 2975.00 30 MARTIN SPRZEDAWCA 09-28-1981 1250.00 30 BLACKI KIEROWNIK 01-05-1981 2850.00 10 CELAREK KIEROWNIK 09-06-1981 2450.00 20 SKOTNIK ANALITYK 09-12-1981 3000.00 10 KING PREZES 11-17-1981 5000.00 30 TURNER SPRZEDAWCA 08-09-1981 1500.00 30 ADAMCZYK URZEDNIK 12-01-1983 1100.00 30 JAMSKI URZEDNIK 03-12-1981 950.00 20 FORD ANALITYK 03-12-1981 3000.00 10 MILLER URZEDNIK 01-23-1982 1300.00 (14 rows)</pre>
<pre>SELECT pa.zarob from prac pc, dzial d, premia pa WHERE pc.numdz = d.numdz and pc.nazwp = pa.nazwp and d.lok = 'DALLAS';</pre>	<pre>max ----- 2975.00 (1 row)</pre>
<pre>SELECT numdz, nazwp, stanowisko, zatrud, zarob FROM prac WHERE numdz IN (SELECT numdz FROM dzial WHERE lok='DALLAS') AND zarob =(SELECT MAX(zarob) FROM premia);</pre>	
<pre>numdz nazwp stanowisko zatrud zarob -----+-----+-----+-----+----- (0 rows)</pre>	
<pre>SELECT max(pa.zarob)</pre>	<pre>max</pre>

Składnia (podzapytanie)

<p>FROM <i>prac</i> <i>pc</i>, dzial <i>d</i>, premia <i>pa</i></p> <p>WHERE <i>pc.numdz</i> = <i>d.numdz</i> and <i>pc.nazwp</i> = <i>pa.nazwp</i> and <i>d.lok</i> = 'CHICAGO';</p>	<p>----- 1600.00 (1 row)</p>
--	--

Zadanie Nr 10 Przedstaw podstawowe dane pracowników którzy otrzymali premię oraz tych którzy jej nie otrzymali

Składnia	Wynik																																								
<p>SELECT <i>nazwp AS nazwisko</i>, <i>stanowisko</i>, <i>zarob AS zarobki</i>, <i>prow AS prowizja_zasadnicza</i></p> <p>FROM <i>prac</i></p> <p>WHERE <i>nazwp IN</i> (SELECT <i>nazwp</i> FROM <i>premia</i>);</p>	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>zarobki</th> <th>prowizja</th> </tr> </thead> <tbody> <tr> <td>SMITKO</td> <td>URZEDNIK</td> <td>800.00</td> <td></td> </tr> <tr> <td>ALLEN</td> <td>SPRZEDAWCA</td> <td>1600.00</td> <td>300.00</td> </tr> <tr> <td>WARD</td> <td>SPRZEDAWCA</td> <td>1250.00</td> <td>500.00</td> </tr> <tr> <td>JONAS</td> <td>KIEROWNIK</td> <td>2975.00</td> <td></td> </tr> <tr> <td>MARTIN</td> <td>SPRZEDAWCA</td> <td>1250.00</td> <td>1400.00</td> </tr> </tbody> </table> <p>(5 rows)</p>	Nazwisko	stanowisko	zarobki	prowizja	SMITKO	URZEDNIK	800.00		ALLEN	SPRZEDAWCA	1600.00	300.00	WARD	SPRZEDAWCA	1250.00	500.00	JONAS	KIEROWNIK	2975.00		MARTIN	SPRZEDAWCA	1250.00	1400.00																
Nazwisko	stanowisko	zarobki	prowizja																																						
SMITKO	URZEDNIK	800.00																																							
ALLEN	SPRZEDAWCA	1600.00	300.00																																						
WARD	SPRZEDAWCA	1250.00	500.00																																						
JONAS	KIEROWNIK	2975.00																																							
MARTIN	SPRZEDAWCA	1250.00	1400.00																																						
<p>SELECT <i>nazwp AS nazwisko</i>, <i>stanowisko</i>, <i>zarob AS zarobki</i>, <i>prow AS prowizja</i></p> <p>FROM <i>prac</i></p> <p>WHERE <i>nazwp NOT IN</i> (SELECT <i>nazwp</i> FROM <i>premia</i>);</p>	<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>zarobki</th> <th>prowizja</th> </tr> </thead> <tbody> <tr> <td>BLACKI</td> <td>KIEROWNIK</td> <td>2850.00</td> <td></td> </tr> <tr> <td>CELAREK</td> <td>KIEROWNIK</td> <td>2450.00</td> <td></td> </tr> <tr> <td>SKOTNIK</td> <td>ANALITYK</td> <td>3000.00</td> <td></td> </tr> <tr> <td>KING</td> <td>PREZES</td> <td>5000.00</td> <td></td> </tr> <tr> <td>TURNER</td> <td>SPRZEDAWCA</td> <td>1500.00</td> <td>0.00</td> </tr> <tr> <td>ADAMCZYK</td> <td>URZEDNIK</td> <td>1100.00</td> <td>20.00</td> </tr> <tr> <td>JAMSKI</td> <td>URZEDNIK</td> <td>950.00</td> <td></td> </tr> <tr> <td>FORD</td> <td>ANALITYK</td> <td>3000.00</td> <td></td> </tr> <tr> <td>MILLER</td> <td>URZEDNIK</td> <td>1300.00</td> <td></td> </tr> </tbody> </table> <p>(9 ROWS)</p>	Nazwisko	stanowisko	zarobki	prowizja	BLACKI	KIEROWNIK	2850.00		CELAREK	KIEROWNIK	2450.00		SKOTNIK	ANALITYK	3000.00		KING	PREZES	5000.00		TURNER	SPRZEDAWCA	1500.00	0.00	ADAMCZYK	URZEDNIK	1100.00	20.00	JAMSKI	URZEDNIK	950.00		FORD	ANALITYK	3000.00		MILLER	URZEDNIK	1300.00	
Nazwisko	stanowisko	zarobki	prowizja																																						
BLACKI	KIEROWNIK	2850.00																																							
CELAREK	KIEROWNIK	2450.00																																							
SKOTNIK	ANALITYK	3000.00																																							
KING	PREZES	5000.00																																							
TURNER	SPRZEDAWCA	1500.00	0.00																																						
ADAMCZYK	URZEDNIK	1100.00	20.00																																						
JAMSKI	URZEDNIK	950.00																																							
FORD	ANALITYK	3000.00																																							
MILLER	URZEDNIK	1300.00																																							
<p>SELECT <i>nazwp AS nazwisko</i>, <i>stanowisko</i>, <i>zarob AS zarobki</i>, <i>prow AS prowizja</i></p> <p>FROM <i>prac</i></p> <p>WHERE <i>nazwp IN</i> (SELECT <i>nazwp</i> FROM <i>premia</i>);</p>	<p>SELECT <i>pc.nazwp AS nazwisko</i>, <i>pc.stanowisko</i>, <i>pc.zatrud AS placa</i>, <i>pc.prow AS prowizja</i>, <i>pa.zarob AS premia</i>, <i>pa.prow AS prowizja</i></p> <p>FROM <i>prac pc</i>, <i>premia pa</i></p> <p>WHERE <i>pc.nazwp = pa.nazwp</i>;</p>																																								
<table border="1"> <thead> <tr> <th>Nazwisko</th> <th>stanowisko</th> <th>placa</th> <th>prowizja</th> <th>premia</th> <th>prowizja</th> </tr> </thead> <tbody> <tr> <td>SMITKO</td> <td>URZEDNIK</td> <td>12-17-1980</td> <td>800.00</td> <td></td> <td></td> </tr> <tr> <td>ALLEN</td> <td>SPRZEDAWCA</td> <td>02-20-1981</td> <td>300.00</td> <td>1600.00</td> <td>300.00</td> </tr> <tr> <td>WARD</td> <td>SPRZEDAWCA</td> <td>02-22-1981</td> <td>500.00</td> <td>1250.00</td> <td>500.00</td> </tr> <tr> <td>JONAS</td> <td>KIEROWNIK</td> <td>02-04-1981</td> <td>2975.00</td> <td></td> <td></td> </tr> <tr> <td>MARTIN</td> <td>SPRZEDAWCA</td> <td>09-28-1981</td> <td>1400.00</td> <td>1250.00</td> <td>1400.00</td> </tr> </tbody> </table> <p>(5 rows)</p>		Nazwisko	stanowisko	placa	prowizja	premia	prowizja	SMITKO	URZEDNIK	12-17-1980	800.00			ALLEN	SPRZEDAWCA	02-20-1981	300.00	1600.00	300.00	WARD	SPRZEDAWCA	02-22-1981	500.00	1250.00	500.00	JONAS	KIEROWNIK	02-04-1981	2975.00			MARTIN	SPRZEDAWCA	09-28-1981	1400.00	1250.00	1400.00				
Nazwisko	stanowisko	placa	prowizja	premia	prowizja																																				
SMITKO	URZEDNIK	12-17-1980	800.00																																						
ALLEN	SPRZEDAWCA	02-20-1981	300.00	1600.00	300.00																																				
WARD	SPRZEDAWCA	02-22-1981	500.00	1250.00	500.00																																				
JONAS	KIEROWNIK	02-04-1981	2975.00																																						
MARTIN	SPRZEDAWCA	09-28-1981	1400.00	1250.00	1400.00																																				

Zadanie Nr 11 Którzy pracownicy z tego samego miejsca pracy co BLACKI zarabiają mniej niż wynosi średnia pensja

Składnia	Wynik																																																																								
<pre>SELECT * FROM prac WHERE zarob < (SELECT AVG(zarob) FROM prac) AND numdz IN (SELECT numdz FROM dzial WHERE lok = (SELECT lok FROM dzial WHERE numdz = (SELECT numdz FROM prac WHERE nazwp = 'BLACKI')));</pre>																																																																									
<table border="1"> <thead> <tr> <th>Nump</th> <th>nazwp</th> <th>stanowisko</th> <th>kier</th> <th>zatrud</th> <th>zarob</th> <th>prow</th> <th>numdz</th> </tr> </thead> <tbody> <tr><td>7499</td><td>ALLEN</td><td>SPRZEDAWCA</td><td>7698</td><td>02-20-1981</td><td>1600.00</td><td>300.00</td><td>30</td></tr> <tr><td>7521</td><td>WARD</td><td>SPRZEDAWCA</td><td>7698</td><td>02-22-1981</td><td>1250.00</td><td>500.00</td><td>30</td></tr> <tr><td>7654</td><td>MARTIN</td><td>SPRZEDAWCA</td><td>7698</td><td>09-28-1981</td><td>1250.00</td><td>1400.00</td><td>30</td></tr> <tr><td>7844</td><td>TURNER</td><td>SPRZEDAWCA</td><td>7698</td><td>08-09-1981</td><td>1500.00</td><td>0.00</td><td>30</td></tr> <tr><td>7876</td><td>ADAMCZYK</td><td>URZEDNIK</td><td>7788</td><td>12-01-1983</td><td>1100.00</td><td>20.00</td><td>30</td></tr> <tr><td>7900</td><td>JAMSKI</td><td>URZEDNIK</td><td>7698</td><td>03-12-1981</td><td>950.00</td><td></td><td>30</td></tr> </tbody> </table> <p>(6 rows)</p>	Nump	nazwp	stanowisko	kier	zatrud	zarob	prow	numdz	7499	ALLEN	SPRZEDAWCA	7698	02-20-1981	1600.00	300.00	30	7521	WARD	SPRZEDAWCA	7698	02-22-1981	1250.00	500.00	30	7654	MARTIN	SPRZEDAWCA	7698	09-28-1981	1250.00	1400.00	30	7844	TURNER	SPRZEDAWCA	7698	08-09-1981	1500.00	0.00	30	7876	ADAMCZYK	URZEDNIK	7788	12-01-1983	1100.00	20.00	30	7900	JAMSKI	URZEDNIK	7698	03-12-1981	950.00		30																	
Nump	nazwp	stanowisko	kier	zatrud	zarob	prow	numdz																																																																		
7499	ALLEN	SPRZEDAWCA	7698	02-20-1981	1600.00	300.00	30																																																																		
7521	WARD	SPRZEDAWCA	7698	02-22-1981	1250.00	500.00	30																																																																		
7654	MARTIN	SPRZEDAWCA	7698	09-28-1981	1250.00	1400.00	30																																																																		
7844	TURNER	SPRZEDAWCA	7698	08-09-1981	1500.00	0.00	30																																																																		
7876	ADAMCZYK	URZEDNIK	7788	12-01-1983	1100.00	20.00	30																																																																		
7900	JAMSKI	URZEDNIK	7698	03-12-1981	950.00		30																																																																		
<pre>SELECT * FROM prac WHERE zarob < (SELECT AVG(zarob) FROM prac);</pre>																																																																									
<table border="1"> <thead> <tr> <th>Nump</th> <th>nazwp</th> <th>stanowisko</th> <th>kier</th> <th>zatrud</th> <th>zarob</th> <th>prow</th> <th>numdz</th> </tr> </thead> <tbody> <tr><td>7369</td><td>SMITKO</td><td>URZEDNIK</td><td>7902</td><td>12-17-1980</td><td>800.00</td><td></td><td>20</td></tr> <tr><td>7499</td><td>ALLEN</td><td>SPRZEDAWCA</td><td>7698</td><td>02-20-1981</td><td>1600.00</td><td>300.00</td><td>30</td></tr> <tr><td>7521</td><td>WARD</td><td>SPRZEDAWCA</td><td>7698</td><td>02-22-1981</td><td>1250.00</td><td>500.00</td><td>30</td></tr> <tr><td>7654</td><td>MARTIN</td><td>SPRZEDAWCA</td><td>7698</td><td>09-28-1981</td><td>1250.00</td><td>400.00</td><td>30</td></tr> <tr><td>7844</td><td>TURNER</td><td>SPRZEDAWCA</td><td>7698</td><td>08-09-1981</td><td>1500.00</td><td>0.00</td><td>30</td></tr> <tr><td>7876</td><td>ADAMCZYK</td><td>URZEDNIK</td><td>7788</td><td>12-01-1983</td><td>1100.00</td><td>20.00</td><td>30</td></tr> <tr><td>7900</td><td>JAMSKI</td><td>URZEDNIK</td><td>7698</td><td>03-12-1981</td><td>950.00</td><td></td><td>30</td></tr> <tr><td>7934</td><td>MILLER</td><td>URZEDNIK</td><td>7782</td><td>01-23-1982</td><td>1300.00</td><td></td><td>10</td></tr> </tbody> </table> <p>(8 rows)</p>	Nump	nazwp	stanowisko	kier	zatrud	zarob	prow	numdz	7369	SMITKO	URZEDNIK	7902	12-17-1980	800.00		20	7499	ALLEN	SPRZEDAWCA	7698	02-20-1981	1600.00	300.00	30	7521	WARD	SPRZEDAWCA	7698	02-22-1981	1250.00	500.00	30	7654	MARTIN	SPRZEDAWCA	7698	09-28-1981	1250.00	400.00	30	7844	TURNER	SPRZEDAWCA	7698	08-09-1981	1500.00	0.00	30	7876	ADAMCZYK	URZEDNIK	7788	12-01-1983	1100.00	20.00	30	7900	JAMSKI	URZEDNIK	7698	03-12-1981	950.00		30	7934	MILLER	URZEDNIK	7782	01-23-1982	1300.00		10	
Nump	nazwp	stanowisko	kier	zatrud	zarob	prow	numdz																																																																		
7369	SMITKO	URZEDNIK	7902	12-17-1980	800.00		20																																																																		
7499	ALLEN	SPRZEDAWCA	7698	02-20-1981	1600.00	300.00	30																																																																		
7521	WARD	SPRZEDAWCA	7698	02-22-1981	1250.00	500.00	30																																																																		
7654	MARTIN	SPRZEDAWCA	7698	09-28-1981	1250.00	400.00	30																																																																		
7844	TURNER	SPRZEDAWCA	7698	08-09-1981	1500.00	0.00	30																																																																		
7876	ADAMCZYK	URZEDNIK	7788	12-01-1983	1100.00	20.00	30																																																																		
7900	JAMSKI	URZEDNIK	7698	03-12-1981	950.00		30																																																																		
7934	MILLER	URZEDNIK	7782	01-23-1982	1300.00		10																																																																		
<pre>SELECT numdz FROM prac WHERE nazwp = 'BLACKI';</pre>	<pre>numdz ----- 30 (1 row)</pre>																																																																								
<pre>(SELECT lok FROM dzial WHERE numdz = 30;</pre>	<pre>lok ----- CHICAGO (1 row)</pre>																																																																								
<pre>(SELECT numdz FROM dzial WHERE lok ='CHICAGO';</pre>	<pre>numdz ----- 30 (1 row)</pre>																																																																								
<pre>SELECT AVG(zarob) FROM prac;</pre>	<pre>avg ----- 2073.2142857143 (1 row)</pre>																																																																								

18.2.2 Podzapytanie – operator porównania ANY bądź ALL

18.2.2.1 składnia

Kolejny rodzaj podzapytania, zwracającego zero lub więcej wierszy, wykorzystuje operator porównania zmodyfikowany przez słowa kluczowe ALL bądź ANY.

Początek instrukcji **SELECT**, INSERT, UPDATE, DELETE; lub podzapytanie
WHERE wyrażenie operator_porównania [ANY | ALL] (podzapytanie)
[koniec instrukcji **SELECT**, INSERT, UPDATE, DELETE; lub podzapytanie]

18.2.2.2 znaczenie ALL i ANY

Jeśli jako przykładu użyjemy operatora porównania „>”, to:

> **ALL** oznacza **większe niż każda inna wartość** - innymi słowy, większe od największej wartości. W ten sposób > ALL (1,2,3) oznacza większe niż 3.

> **ANY** oznacza **większe od co najmniej jednej wielkości** – innymi słowy, większe niż minimum. Tak więc > ANY (1,2,3) oznacza większe niż 1.

ALL	Wynik	ANY	Wynik
> ALL(1,2,3)	> 3	> ANY (1,2,3)	> 1
< ALL (1,2,3)	< 1	< ANY (1,2,3)	< 3
= ALL (1,2,3)	= 1 i =2 i =3 (jednocześnie)	= ANY (1,2,3)	= 1 lub =2 lub =3

Jest ważna zasada, o której musisz pamiętać podczas używania podzapytań. Otóż lewa strona predykatu wymaga zgodności wyniku z podzapytaniem po prawej stronie. Jeśli na przykład w predykacie jest operator logiczny, to podzapytanie musi zwrócić pojedynczą wartość.

Liczba > (podzapytanie **zwraca pojedynczą liczbę**)

Napis = (podzapytanie **zwracające pojedynczy napis**)

Inne zasady dotyczą operatora IN, gdy używa się go z podzapytaniem. Obowiązują one zarówno wtedy, kiedy podzapytanie daje jedną wartość, jak i wtedy, kiedy daje ono więcej wartości. Dopuszczalny jest nawet brak wartości (rezultat NULL). Sytuacja taka oznacza, że warunek logiczny jest fałszywy i że dany wiersz nie zostanie zaliczony do wyniku.

Liczba IN > (podzapytanie **zwraca listę liczb**)

Napis IN = (podzapytanie **zwraca listę napisów**)

Należy zaznaczyć że

liczba > (lista liczb)

nie jest poprawna.

Możliwe jest jednak uniknięcie niepoprawnych zapytań tego typu. Możesz używać słowa kluczowego ANY (służy do porównania wartości z każdą wartością zwracaną przez podzapytanie)

Liczba > ANY (lista liczb)

18.2.2.3 Podzapytania z ALL

Które książki uzyskały zaliczkę większą niż zaliczka na dowolną książkę wydaną przez New Age Books?

Możemy to zapytanie sformułować, aby tłumaczenie na SQL było jaśniejsze:

Które książki uzyskały zaliczkę większą niż największa zaliczka wypłacona przez New Age Books?

SELECT title

FROM titles

```
WHERE advence > ALL
      (SELECT advence
       FROM publishers, titles
       WHERE titles.pub_id = publishers.pub_id AND pub_name = 'New Age Boks');
```

Uwagi:

Zapytanie wewnętrzne znajduje dla każdego tytułu listę zaliczek wypłacanych przez New Age Books. Zapytanie zewnętrzne wyszukuje największą wartość z listy i określa, czy rozważany tytuł nie uzyskał większej zaliczki. Jeśli wprowadzone przez ALL i operator porównania zapytanie wewnętrzne jako jedną ze swych wartości zwraca NULL, to całe zapytanie zawodzi.

Zadanie Nr 12 Wskaż wszystkich pracowników, których pensja jest wyższa niż pensja wszystkich kierowników.

```
SELECT nump, nazwp, stanowisko, zarob
FROM   prac
WHERE  zarob > ALL
      (SELECT zarob
       FROM   prac
       WHERE  stanowisko = 'KIEROWNIK');
```

Nump	nazwp	stanowisko	zarob
7788	SKOTNIK	ANALITYK	3000.00
7839	KING	PREZES	5000.00
7902	FORD	ANALITYK	3000.00

(3 rows)

```
SELECT nump, nazwp, stanowisko, zarob
FROM   prac
WHERE  zarob >
      (SELECT MAX(zarob)
       FROM   prac
       WHERE  stanowisko = 'KIEROWNIK');
```

Uwaga: w tym wypadku sformułowanie pytania nie wprowadza w błąd. Pytamy się o pracowników, których pensja (zarob) jest większa niż pensja wszystkich kierowników. To odpowiada definicji operatora ALL. Alternatywą w SQL dla tego zapytania jest ... ZAROB > (SELECT MAX(ZAROB) ...)

18.2.2.4 Podzapytania z ANY

SQL umożliwia przetworzenie wielu wartości przekazywanych przez podzapytanie w sytuacji gdy mamy do czynienia z predykatem wymagającym tylko pojedynczej wartości. Przypuśćmy, że chcesz wiedzieć, kto zarabia więcej od „Smitka” .

```
SELECT *
FROM   prac
WHERE  zarob >
      (SELECT zarob
       FROM   prac
       WHERE  nazwp = 'SMITKO');
```

Jeśli jest tylko jeden SMITKO w tabeli PRAC, to podane zapytanie zadziała. Podzapytanie zwróci pojedynczą wartość, która z kolei zostanie użyta przez predykat akceptujący pojedynczą wartość.

Jeśli natomiast w tabeli PRAC jest dwóch lub kilku SMITKÓW to podzapytanie zwróci zbiór numerów – po jednym dla każdego wiersza PRAC, w którym NAZWP ma wartość SMITKO. Zapytanie nie zadziała, ponieważ składnia

Liczba > (lista liczb)

Nie jest poprawna.

Możliwe jest jednak uniknięcie niepoprawnych zapytań tego typu. Możesz użyć słowa ANY (służy do porównania wartości z każdą wartością zwracaną przez podzapytanie)

Liczba > ANY (lista liczb)

Powyższe wyrażenie jest prawdziwe, jeśli liczba po lewej stronie jest większa niż jedna z liczb na liście. Zapis >ANY należy tłumaczyć jako „większe niż przynajmniej jeden element listy”

```
SELECT *
FROM prac
WHERE zarob > ANY
(SELECT zarob
FROM prac
WHERE nazwp = 'SMITKO');
```

Zapytanie z ANY wyszukuje wartości większe niż „pewna” wartość podzapytania.

Następujące zapytanie znajduje tytuły, na które wypłacono zaliczkę większą od minimalnej zaliczki (5000 USD) wypłaconej przez Algodata Infosystems.

```
SELECT title, advence
```

```
FROM titles
```

```
WHERE advence > ANY
```

```
(SELECT advence
```

```
FROM publishers, titles
```

```
WHERE titles.pub_id = publishers.pub_id
```

```
AND pub_name = 'Algodata Infosystems');
```

Zapytanie wewnętrzne znajduje dla każdego tytułu listę wielkości zaliczek wypłaconych przez Algodata Infosystems. Zapytanie zewnętrzne przegląda te wszystkie wartości z listy i określa, czy rozpatrywany tytuł uzyskał zaliczkę większą od jakiegokolwiek z nich. Jeśli podzapytanie nie zwraca żadnej wartości, to całe zapytanie zawodzi.

Zadanie Nr 13 W tabeli PRAC wskaż wszystkich urzędników, których pensja jest większa niż pensja sprzedawców. Podaj NUMP, NAZWP, STANOWISKO I ZAROB

```
SELECT nump, nazwp, stanowisko, zarob
FROM prac
WHERE stanowisko = 'URZEDNIK' AND zarob > ANY
(SELECT zarob
FROM prac
WHERE stanowisko = 'SPRZEDAWCA');
```

Nump	nazwp	stanowisko	zarob
7934	MILLER	URZEDNIK	1300.00

(1 row)

```
SELECT nump, nazwp, stanowisko, zarob
FROM prac
WHERE stanowisko = 'URZEDNIK' AND zarob >
(SELECT min(zarob)
FROM prac
WHERE stanowisko = 'SPRZEDAWCA');
```

Jest to podchwytliwe pytanie, użyte do pogłębienia znajomości definicji konstrukcji ANY. Gdybyśmy poprosili o „urzędników, których pensja jest większa niż jakakolwiek pensja sprzedawcy”, moglibyśmy wprowadzić Cię w błąd i mógłbyś użyć ANY. Sformułowanie „większa niż jakakolwiek” znaczy zazwyczaj „wyższa niż wszystkie możliwe wartości”. Pamiętaj, że większa niż jakakolwiek” tłumaczy się w SQL na ALL. Bądź więc ostrożny. W naszej odpowiedzi użyliśmy ANY, aby wybrać tych urzędników, których pensja jest większa niż jakaś jedna pensja sprzedawcy. Kiedy używasz ANY, predykat ma zachodzić „co najmniej jedna wartość na liście”. Można to jeszcze napisać inaczej, a mianowicie ... ZAROB > (SELECT MIN(zarob))

18.3 Porównanie IN, ANY, ALL

Operatory IN i NOT IN są szczególnymi przypadkami bardziej podstawowym fraz ANY, ALL i EXISTS, stosowanych w podzapytaniach.

Fraza ANY zwraca wartość logiczną *prawda*, jeżeli porównanie jest prawdziwe dla jednej wartości. Np. porównanie **pole**= ANY(5,7,9) zwraca wartość logiczną *prawda*, jeżeli **pole** jest równa jednej z tych trzech wartości.

Fraza ALL zwraca wartość logiczną *prawda*, jeżeli wszystkie porównania dają wartość logiczną *prawda*, zatem col != ALL(5,7,9) zwraca wartość logiczną *prawda*, jeżeli **pole** nie jest równa żadnej z tych wartości.

Operator IN() działa tak samo jak fraza ANY ()

Operator NOT IN() działa tak samo jak fraza <> ALL()

Normalnie operatory takie jak „równa się” czy „większy niż” stosuje się jedynie w podzapytaniach zwracających jeden wiersz. Jednakże w przypadku fraz ANY i ALL można dokonywać porównania z podzapytaniem zwracającym więcej niż jeden wiersz. Frazy te pozwalają odpowiednio określić, czy wszystkie, chociaż jedno porównanie musi dać w wyniku wartość logiczną *prawda*, aby cała fraza miała wartość logiczną *prawda*.

Pole IN (podzapytanie)	Pole = ANY (podzapytanie)
Pole NOT IN (podzapytanie)	Pole <> ALL (podzapytanie)
Pole > ANY (podzapytanie)	Pole > (SELECT MIN(pole) ..
Pole > ALL (podzapytanie)	Pole > (SELECT MAX(pole)

Aby wyszukać autorów mieszkających w tym samym mieście, w którym znajduje się jakiś wydawca, można użyć:

<pre>SELECT au_lname, au_fname, city FROM authors WHERE city IN (SELECT city FROM publishers);</pre>	<pre>SELECT au_lname, au_fname, city FROM authors WHERE city = ANY (SELECT city FROM publishers);</pre>
---	--

Strona 217

18.3.1 Podzapytanie EXISTS

Kiedy podzapytanie jest wprowadzone za pomocą słowa kluczowego EXISTS, działa wtedy jak „test na istnienie”. Słowo kluczowe w klauzuli WHERE sprawdza istnienie bądź nieistnienie danych spełniających kryteria podzapytania.

Logiczne wyrażenie EXISTS (podzapytanie) jest prawdziwe wtedy, kiedy w wyniku działania podzapytania zostanie zwrócony co najmniej jeden wiersz. W przeciwnym razie jest ono fałszywe.

18.3.2 Definicja EXISTS


Początek instrukcji **SELECT**, **INSERT**, **UPDATE**, **DELETE**; lub podzapytanie
WHERE [NOT] EXISTS (podzapytanie)
[koniec instrukcji **SELECT**, **INSERT**, **UPDATE**, **DELETE**; lub podzapytanie]

Przypuśćmy, że chciałbyś uzyskać listę działów z tabeli DZIAL, w których są zatrudnieni pracownicy otrzymujący prowizję.

```
SELECT *  
FROM dzial D  
WHERE EXISTS  
    (SELECT *  
     FROM prac P  
     WHERE P.numdz=D.numdz  
     AND prow IS NOT NULL);
```

Główne zapytanie z etykietą tabelową (lub inaczej nazwą korelacyjną) D prowadzi do wyszukania wierszy w tabeli DZIAL. W wyniku działania podzapytania zostaną wybrane te wiersze z tabeli PRAC, w których wartość prowizji jest różna od NULL i kiedy mają tę samą wartość NUMDZ co wiersz tabeli DZIAL w głównym zapytaniu.

```
SELECT *  
FROM dzial D  
WHERE EXISTS – prawda  
    (SELECT * FROM prac P  
     WHERE P.numdz=D.numdz  
     AND prow IS NOT NULL);
```



Fakt, iż doszło do wybrania wiersza, dowodzi, że predykat EXISTS (podzapytanie) jest prawdziwy. Zostanie wyświetlony wiersz w tabeli DZIAL, z tą samą wartością NUMDZ co wiersz z tabeli PRAC, wybrany w wyniku działania podzapytania.

Konstrukcja **SELECT *** jest naturalna dla podzapytania, ponieważ podzapytanie skorelowane nie przekazuje żadnego wyniku tymczasowego do głównego podzapytania.

Aby wyszukać nazwy wszystkich wydawców publikujących książki o biznesie należy:

```
SELECT distinct pub_name  
FROM publishers  
WHERE EXISTS  
    (SELECT *  
     FROM titles  
     WHERE pub_id = publishers.pub_id AND type = 'biznes')
```

Uwagi:

EXISTS sprawdza obecność lub brak „pustego zbioru” wierszy. Jeśli podzapytanie zwraca co najmniej jeden wiersz, to wynik podzapytania określa się jako „prawdziwy”

(true). Oznacza to, że wyrażenie EXISTS jest prawdziwe, a wyrażenie NOT EXISTS jest fałszywe. Jeśli podzapytanie zwraca zbiór pusty (brak wierszy), to wynik podzapytania określa się jako „fałszywy” (false). Oznacza to, że wyrażenie NOT EXISTS jest prawdziwe, a wyrażenie EXISTS jest fałszywe.

Składnia podzapytania wprowadzonych za pomocą EXISTS różni się nieco od składni innych zapytań:

Słowo EXISTS nie jest poprzedzone nazwą kolumny, stałą ani innym wyrażeniem

Lista wyboru podzapytania wprowadzanego za pomocą EXISTS prawie zawsze składa się z gwiazdki (*). Nazwa kolumny nie ma znaczenia, gdyż po prostu sprawdza się istnienie wierszy spełniających podzapytania, a te są wyrażone w klauzuli WHERE podzapytania, a nie w klauzuli SELECT podzapytania.

Wyszukiwanie tytułu książek wydanych przez dowolnego wydawcę mającego siedzibę w mieście o nazwie rozpoczynającej się literą B.

<pre>SELECT title FROM titles WHERE pub_id IN (SELECT pub_id FROM publishers WHERE city like 'B%');</pre>	<pre>SELECT title FROM titles WHERE EXISTS (SELECT * FROM publishers WHERE pub_id = titles.pub_id AND city like 'B%');</pre>
---	--

18.3.2.1 Not EXISTS

Zapytania z NOT EXISTS kończą się powodzeniem, gdy podzapytanie nie zwraca wartości.

Wyszukaj nazwy wydawców nie publikujących książek o biznesie

```
SELECT pub_name
FROM publishers
WHERE NOT EXISTS
  (SELECT *
   FROM titles
   WHERE pub_id = publishers.pub_id AND type = 'bussiness');
```

Wyszukaj tytuły książek których nie wydano

```
SELECT titles
FROM titles
WHERE NOT EXISTS
  (SELECT title_id
   FROM salesdetails
   WHERE title_id = titles.title_id);
```

18.3.3 Zastosowanie EXISTS

Podzapytania wprowadzane za pomocą EXISTS i NOT EXISTS mogą być stosowane do dwóch operacji teorii zbiorów - znajdowania :

części wspólnej (przecięcia). Część wspólna dwóch zbiorów zawiera wszystkie elementy należące do obu początkowych zbiorów

różnicy. Różnica zawiera elementy, które należą tylko do pierwszego z dwóch zbiorów.

Część wspólna kolumn city w tabelach authors i publishers jest zbiorem miast, w których są ulokowani zarówno autor, jak i wydawca.

Różnica kolumn city z tabeli authors i publishers jest zbiorem miast, w których mieszkają autorzy, ale w których nie ma siedziby wydawcy (czyli zbiorem wszystkich miast oprócz Berkeley).

Część wspólna (I)	Różnica (II)
SELECT DISTINCT city FROM authors WHERE EXISTS (SELECT * FROM publishers WHERE authors.city = publishers.city);	SELECT DISTINCT city FROM authors WHERE NOT EXISTS (SELECT * FROM publishers WHERE authors.city = publishers.city);

19 Złączenia

Operacje złączenia umożliwiają wyszukiwanie i operowanie danymi, pochodzącymi z więcej niż jednej tabeli, za pomocą pojedynczej instrukcji SELECT

19.1 Składnia

```
SELECT lista_wyboru  
FROM tabela_1, tabela_1  
WHERE tabela_1.kolumna operator_złączenia tabela_2.kolumna
```

Lista tabel klauzuli FROM musi zawierać **co najmniej dwie tabele**, a kolumny określone w klauzuli WHERE muszą być **złączonowe zgodnie**. Jeśli kolumny łączące mają identyczne nazwy, to koniecznie trzeba te kolumny określić, podając na liście wyboru i w klauzuli WHERE nazwy tabel, z których te kolumny pochodzą.

19.1.1 Złączenia a model relacyjny

Złączenia są możliwe, ponieważ występują w modelu relacyjnym, a są konieczne dlatego, że występują w modelu relacyjnym. (typy postaci normalnych)

19.2 Warunki wstępne złączenia

W składni instrukcji SELECT wstępne warunki złączenia to:

Podanie na liście FROM więcej niż jednej nazwy tabeli

Dodanie w klauzuli WHERE warunków zapewniających utworzenie złączenia za pomocą kolumn łączących.

19.2.1 UWAGI:

Jeśli w kolumnach łączących występują wartości null, to nie zostaną one nigdy złączone, gdyż reprezentują wartości nieznane bądź nie znajdujące zastosowania.

Kolejność tabel (lub perspektyw) na liście wpływa na wynik wyświetlany tylko wtedy, gdy do określenia listy wyboru użyto SELECT *

Grupowanie danych

20 UNION

Union nie oznacza złączenia, ale zapewnia możliwość połączenia danych z wielu zapytań w jeden wynik.

20.1 Składnia

Instrukcja_select

UNION

Instrukcja_select;

Operacja **UNION** jest użyteczna, gdy chcemy obejrzeć podobne dane z dwóch lub więcej tabel w czasie jednego wyświetlenia.

UNION domyślnie usuwa powtarzające się wiersze z wyświetlanych wyników. Może to być mylące, gdy w zapytaniu występuje tylko jedna kolumna.

Oto reguły obowiązujące przy stosowaniu **UNION**:

Liczba wyrażeń w każdym zapytaniu musi być taka sama

Typy danych odpowiadających sobie wyrażeń w każdym zapytaniu muszą być zgodne.

Zgodność oznacza, że jeśli pierwsze wyrażenie jest liczbowego typu danych, to wszystkie odpowiadające pierwsze wyrażenia również muszą być liczbowego typu danych

Nazwy odpowiadających sobie kolumn w dwóch zapytaniach nie muszą być takie same.

Poprawna składnia SQL:

```
SELECT ZAROB
```

```
FROM PRAC
```

```
UNION
```

```
SELECT PROW
```

```
FROM PREMIA;
```

Ponieważ istnieje taka możliwość, że nazwy odpowiadające sobie kolumn w zapytaniach będą różne, sortowanie wartości za pomocą opcjonalnej klauzuli **ORDER BY**, występującej po ostatnim zapytaniu, musi się odbywać według pozycji, a nie nazwy. Klauzula **ORDER BY 1** jest poprawna a **ORDER BY p.NUMP** nie

W wyniku użycia **UNION** dochodzi do wyeliminowania powtarzających się wierszy. Oznacza to, że jeśli wszystkie wartości kolumn w dwóch wierszach wynikowych są równe, to jeden z tych wierszy zostaje usunięty z tabeli wynikowej.

Przykład:

```
Select nazwp, zarob
```

```
From premia
```

```
UNION
```

```
Select nazwp, zarob
```

```
From PRAC
```

```
WHERE stanowisko = 'KIEROWNIK';
```

Wyniki zapytania przedstawiały listę pracowników, którzy dostają premię lub którzy mają pensję powyżej średniej. Przypuśćmy, że chcesz wiedzieć, które wiersze pochodzą z pierwszego zapytania (dotyczącego przysługujących premii), a które z drugiego (dotyczącego pensji powyżej średniej)

```
SELECT p.NUMP, P.NAZWP, 'DOSTAJACEY PREMIE'
```

```
FROM PRAC P, PREMIA B
WHERE P.NAZWP=B.NAZWP
UNION
SELECT NUMP, NAZWP, 'PENSJA POWYZEJ SREDNIEJ'
FROM PRAC WHERE ZAROB >
(select avg (zarob) from prac
order by 2;
```

21 Rozwiązanie zadań zaliczenie nr 1

Zadanie Nr 14 (1) grupa 1

Utwórz tabelę **lekarz** z następującymi atrybutami:
- **identyfikator** lekarza, który jest kluczem głównym
- data utworzenia rekordu z wykorzystaniem wpisywania daty systemowej
- **imię** lekarza: 10 znaków, nie puste
- **nazwisko** lekarza: max 10 znaków, nie puste
- **identyfikator** specjalizacji – atrybut, w którym mogą się pojawiać tylko znaki: (p) lekarz przychodni (s) lekarz szpitalny, (o) lekarz przyjmujący w przychodni i szpitalu
- **adres** zamieszkania, ale pamiętaj, że 90 % pracowników mieszka w **Kadzidło**

Składnia

```
CREATE TABLE lekarz
```

```
(  
  ID_lek      int CONSTRAINT ID_lekarz PRIMARY KEY,  
  data        date      DEFAULT 'today',  
  imie        varchar(10) NOT NULL,  
  nazwisko    varchar(10) NOT NULL,  
  spec        char(1)    CHECK ( spec='p'  
                                or spec='s'  
                                or spec='o'),  
  miasto      varchar(9) DEFAULT 'Kadzidło',  
  kod         char(5),  
  ul          varchar(15)  
);
```

NOTICE: CREATE TABLE/PRIMARY KEY
will create implicit index 'id_lekarz' for table 'lekarz'

```
CREATE  
EOF
```

uczen2000=> \d lekarz

Table = lekarz

Field	Type	Length
id_lek	int4 not null	4
data	date default 'today'	4
imie	varchar() not null	10
nazwisko	varchar() not null	10
spec	char()	1
miasto	varchar() default 'Kadzidło'	9
kod	char()	5
ul	varchar()	15

Index: id_lekarz

uczen2000=>

Wczytaj do tabeli lekarz następujące dane:

- **Pan** Benio Beniowski zatrudniony w szpitalu zam. 07-450 Myszyniec, ulica Warszawska 10
- **Pan** Czesio Czesiowski zamieszkały w Kadzidle, kod pocztowy 07-400 ul. Zacisze 12 i zatrudniony w przychodni
- **Pan** Genio Geniowski, zamieszkały w Kadzidle przy ul. Wesołka 45, który jest zatrudniony w szpitalu i przychodni

Składnia

```
INSERT INTO lekarz (ID_lek, imie, nazwisko, spec, miasto, kod, ul)
VALUES (1,
        'BENIO',
        'BENIOWSKI',
        's',
        'Myszyniec',
        '07-450',
        'Warszawska 10');
```

```
INSERT 64662 1
EOF
```

```
INSERT INTO lekarz
        (ID_lek , imie , nazwisko ,kod ,ul ,spec )
VALUES (2 , 'CZESIO' , 'CZESIOWSKI' , '07-400' , 'Zacisze 12' , 'p' );
```

```
INSERT INTO lekarz
        (ID_lek , imie , nazwisko , spec , kod , ul )
VALUES (3 , 'GENIO' , 'GENIOWSKI' , 'o' , '07-400' , 'Wesołka 14' );
```

Sprawdzenie wczytania danych

```
uczen2000=> select * from lekarz;
```

id_lek	data	imie	nazwisko	spec	miasto	kod	ul
1	02-16-2002	BENIO	BENIOWSKI	s	Myszyniec	07-45	Warszawska 10
2	02-16-2002	CZESIO	CZESIOWSKI	p	Kadzidło	07-40	Zacisze 12
3	02-16-2002	GENIO	GENIOWSKI	o	Kadzidło	07-40	Wesołka 14

```
(3 rows)
```

```
uczen2000=>
```

CREATE TABLE lekarz

```
(
    ID_lek      int CONSTRAINT ID_lekarz PRIMARY KEY,
    data        date      DEFAULT 'today',
    imie        varchar(10) NOT NULL,
    nazwisko    varchar(10) NOT NULL,
    spec        char(1)    CHECK ( spec='p'
                                or spec='s'
                                or spec='o'),
    miasto      varchar(9)  DEFAULT 'Kadzidło',
    kod         char(5),
    ul          varchar(15)
);
```

Wczytaj z katalogu `/home/2000/uczen2000/kadry` skrypt SQL-a o nazwie `kadry_zal.sql`

`uczen2000=> \i /home/2000/uczen2000/kadry/kadry_zal.sql`

Zadanie Nr 17 (4) grupa I

Wyświetl wartość średnich zarobków wszystkich stanowisk z wyłączeniem stanowiska ANALITYK

i tych stanowisk których średnia przekracza 5000 zł.

Dane posortuj malejąco wg ich średnich zarobków a następnie wg stanowisk

Składnia	
SELECT	stanowisko, AVG (zarob)
FROM	<i>prac</i>
WHERE	stanowisko NOT IN ('ANALITYK')
GROUP BY	stanowisko
HAVING	AVG (zarob) < 5000
ORDER BY	2, 1;
Wynik	
Stanowisko	avg
-----+-----	
URZEDNIK	1037.5000000000
SPRZEDAWCA	1400.0000000000
KIEROWNIK	2758.3333333333
(3 rows)	

Zadanie Nr 18 (5) grupal

Przedstaw listę pracowników z uwzględnieniem następujących danych:

- nazwisko pracownika; stanowisko; płaca zasadnicza; premia; prowizja.

Należy zadbać żeby stały uwzględnione tylko te osoby które otrzymują prowizję

Składnia					
SELECT	<i>pc.nazwp</i>	AS nazwisko,			
	<i>pc.stanowisko</i> ,				
	<i>pc.zarob</i>	AS placa,			
	<i>pc.prow</i>	AS prowizja,			
	<i>pa.zarob</i>	AS premia,			
	<i>pa.prow</i>	AS dodatek			
FROM	<i>prac pc</i> ,				
	premia <i>pa</i>				
WHERE	<i>pc.nazwp</i> = <i>pa.nazwp</i>				
	AND <i>pa.prow</i> > 0;				
Wynik					
Nazwisko	stanowisko	placa	prowizja	premia	dodatek
-----+-----+-----+-----+-----					
ALLEN	SPRZEDAWCA	1600.00	300.00	1600.00	300.00
WARD	SPRZEDAWCA	1250.00	500.00	1250.00	500.00
MARTIN	SPRZEDAWCA	1250.00	1400.00	1250.00	1400.00
(3 rows)					

Zadanie Nr 19 (6) grupa I

Przedstaw **różnicę** między maksymalnymi a minimalnymi zarobkami pracowników wydziału SPRZEDAWCA

Składnia	
SELECT	stanowisko, MAX (zarob) - MIN (zarob) AS Max_MIN
FROM	<i>prac</i>
WHERE	stanowisko = 'SPRZEDAWCA'
GROUP BY	stanowisko;
Wynik	
stanowisko	max_min
-----+-----	
SPRZEDAWCA	350.00
(1 row)	

Zadanie Nr 20 (7) grupa I

Przedstaw listę tych pracowników których premia przekroczyła 1000 zł.
Na liście powinny się znaleźć także podstawowe dane pracownika
czyli : Nazwisko, stanowisko, data zatrudnienia oraz wysokość premii

Składnia	
SELECT	pc.nazwp AS Nazwisko, pc.stanowisko, pc.zatrud AS Data_zatr, pa.zarob AS premia
FROM	<i>prac pc,</i> <i>premia pa</i>
WHERE	pa.nazwp = pc.nazwp AND pa.zarob > 1000;
Wynik	
Nazwisko	stanowisko data_zatr premia
-----+-----+-----+-----	
ALLEN	SPRZEDAWCA 02-20-1981 1600.00
WARD	SPRZEDAWCA 02-22-1981 1250.00
JONAS	KIEROWNIK 02-04-1981 2975.00
MARTIN	SPRZEDAWCA 09-28-1981 1250.00
(4 rows)	

Zadanie Nr 21 (8) grupa I

W dziale Księgowości potrzebna jest lista, która uwzględni następujące dane:

- nazwisko;
- stanowisko;
- datę zatrudnienia,
- płacę zasadniczą
- wydział
- jego lokalizację

Składnia

```

SELECT  pc.nazwp  AS Nazwisko,
          pc.stanowisko,
          pc.zatrud  AS data_zatr,
          pc.zarob   AS placa,
          dz.nazwdz  AS Wydzial,
          dz.lok     AS Lokalizacja
FROM    prac pc,
          dzial dz
WHERE   dz.numdz = pc.numdz;
    
```

Wynik

Nazwisko	stanowisko	data_zatr	placa	wydzial	lokalizacja
SMITKO	URZEDNIK	12-17-1980	800.00	BADANIA	DALLAS
ALLEN	SPRZEDAWCA	02-20-1981	1600.00	SPRZEDAZ	CHICAGO
WARD	SPRZEDAWCA	02-22-1981	1250.00	SPRZEDAZ	CHICAGO
JONAS	KIEROWNIK	02-04-1981	2975.00	BADANIA	DALLAS
MARTIN	SPRZEDAWCA	09-28-1981	1250.00	SPRZEDAZ	CHICAGO
BLACKI	KIEROWNIK	01-05-1981	2850.00	SPRZEDAZ	CHICAGO
CELAREK	KIEROWNIK	09-06-1981	2450.00	KSIEGOWOSC	NOWY YORK
SKOTNIK	ANALITYK	09-12-1981	3000.00	BADANIA	DALLAS
KING	PREZES	11-17-1981	5000.00	KSIEGOWOSC	NOWY YORK
TURNER	SPRZEDAWCA	08-09-1981	1500.00	SPRZEDAZ	CHICAGO
ADAMCZYK	URZEDNIK	12-01-1983	1100.00	SPRZEDAZ	CHICAGO
JAMSKI	URZEDNIK	03-12-1981	950.00	SPRZEDAZ	CHICAGO
FORD	ANALITYK	03-12-1981	3000.00	BADANIA	DALLAS
MILLER	URZEDNIK	01-23-1982	1300.00	KSIEGOWOSC	NOWY YORK

(14 rows)

Zadanie Nr 22 (9) grupa I

Oblicz:
 - średni dochód miesięczny każdego sprzedawcy czyli uwzględniając jego zarobki przez 12 miesięcy
 - prowizję roczną
 - dodatkowo podaj jego wydział wraz z lokalizacją.
 Nie zapomnij posortować dane wg zarobków

Składnia

```
SELECT pc.nazwp AS nazwisko,
       pc.zarob*12+prow AS dochody,
       dz.nazwdz AS wydział,
       dz.lok AS miasto
FROM prac pc,
     dzial dz
WHERE pc.numdz = dz.numdz
      AND stanowisko = 'SPRZEDAWCA'
ORDER BY 2;
```

Wynik

Nazwisko	dochody	wydział	miasto
WARD	15500.00	SPRZEDAZ	CHICAGO
MARTIN	16400.00	SPRZEDAZ	CHICAGO
TURNER	18000.00	SPRZEDAZ	CHICAGO
ALLEN	19500.00	SPRZEDAZ	CHICAGO

(4 rows)

Zadanie Nr 23 (10) grupa I

Poszukujemy pracownika, który został zatrudniony w roku **1981** oraz nazwisko zawiera wyraz **(LA)**

Składnia

```
SELECT nazwp AS nazwisko,
       Stanowisko,
       zatrud
FROM prac
WHERE (zatrud > '80-12-31' OR zatrud < '82-01-01')
      AND nazwp LIKE '%LA%';
```

Wynik

Nazwisko	stanowisko	zatrud
BLACKI	KIEROWNIK	01-05-1981
CELAREK	KIEROWNIK	09-06-1981

(2 rows)

22 Rozwiązanie zadań końcowych

22.1 Pytanie i rozwiązania

Zadanie 1

Baza KADRY: Wybierz wszystkich pracowników, których pensja jest większa niż pensja kierowników oraz posortuj dane wg stanowiska

<pre>SELECT nump, nazwp, stanowisko, zarob FROM prac WHERE zarob > ALL (SELECT zarob FROM prac WHERE stanowisko = 'KI-EROWNIK') GROUP BY stanowisko;</pre>	<pre>SELECT nump, nazwp, stanowisko, zarob FROM prac WHERE zarob > (SELECT MAX(zarob) FROM prac WHERE stanowisko = 'KI-EROWNIK') GROUP BY stanowisko;</pre>
---	--

Zadanie 2

Wskaż działy, które mają co najmniej 4 pracowników

<pre>SELECT stanowisko, numdz, count(*) FROM prac GROUP BY stanowisko, numdz HAVING count(*) > 4 ORDER BY 3, 2</pre>	
--	--

Zadanie 3

Wyświetl dane osobowe pacjentów, którzy byli w szpitalu po 1999 roku

<pre>SELECT P.id_pacjenta, P.nazwisko_p FROM pacjenci P, szpital S WHERE P.id_pacjenta = S.id_pacjenta AND S.data_od > '1998-12-31'</pre>	
--	--

Zadanie 4

Wyświetl dane osobowe pacjentów, którzy nigdy nie byli w szpitalu

<pre>SELECT P.id_pacjenta, P.nazwisko_p FROM pacjenci P WHERE P.id_pacjenta NOT IN (SELECT S.id_pacjenta FROM szpital S WHERE P.id_pacjenta=S.id_pacjenta);</pre>	
---	--

23 Administracja

23.1 Tworzenie kopii bezpieczeństwa

Standardowa instalacja Postgresa zawiera dwa programy do wykonywania zrzutów zawartości (kopii) baz danych:

Pg_dump i pgdumpall

23.1.1 PostgreSQL - Pg_dump

Polecenie **pg_dump** ma składnię:

pg_dump [dbname]

pg_dump [-h host] [-p port] [-t table] [-f outputfile] [-a] [-c] [-d] [-D] [-n] [-N] [-o] [-s] [-u] [-v] [-x] [dbname]

gdzie:

- a** zrzut tylko danych
- f** wyspecyfikowanie nazwy pliku, do którego będą zrzucane dane
- N** domyślna, eksportuje również identyfikatory
- o** zrzut identyfikatorów do wszystkich obiektów w bazie danych
- s** dokonuje wyłącznie zrzutu definicji bazy danych
- t nazwa_tablicy** dokonuje zrzutu wyłącznie danych z tej tablicy
- x** zrzuca informacje o właścicielach tablic i uprawnieniach do nich
- u** pozwala na zalogowanie się do bazy danych poprzez podanie identyfikatora i hasła

23.1.2 Linux - tar

tar

kopiuje plik do lub z archiwum

tar c|r|t|u|x [l] [o] [v] [w] [0-9] [f pliktar] nazwa_pliku

tar -opcje nazwa_archiwum.tar.gz nazwa_katalogu_i_plikow

opcje

- c** tworzy archiwum
- x** odtworzenie archiwum
- f** używa z nazwą urządzenia lub pliku, do którego archiwizujemy
- r** dodaje plik do istniejącego archiwum
- u** aktualizuje pliki w archiwum
- t** pozwala na obejrzenie pliku
- z** wywołuje program gzip by dokonał kompresji

Uwagi:

- | | |
|---|--------------------------------------|
| tar -xzvf archiwum /katalog | wydobycie plików z archiwum |
| tar -czvf backup.tar.gz /katalog | tworzenie archiwum.tar.gz z /katalog |
| tar -tvf archiwum.tar more | wyświetlenie zawartości archiwum.tar |

23.1.3 Podsumowanie

23.1.3.1 Tworzenie kopii

pg_dump nazwa_bazy > nazwa_kopii

np. **pg_dump** uczen2000 > uczen2000.kopia

tar -cvz nazwa_kopii sciezka_dostepu/nazwa_pliku

np. **tar -cvzf** kopia_zapasowa.tar \$PGDATA/base/*
compress kopia_zapasowa.tar

23.1.3.2 Odzyskanie kopii

psql -e nazwa_bazy < nazwa_kopii

```
Np. psql -e uczen2000 <uczen2000.kopia
```

```
uncompress kopia_zapasowa.tar.Z  
tar -xvzf kopia_zapasowa.tar $PGDATA/base/*
```

23.2 Transakcje

BEGIN [WORK| TRANSACTION]

służy do zaznaczenia początku pojedynczej transakcji lub wielu transakcji.

COMMIT [WORK | TRANSACTION]

Służy do zatwierdzania jednej lub wielu transakcji wykonywanych na bazie danych. Transakcja jest to sekwencja instrukcji języka SQL traktowana przez bazę danych postgres jako wyodrębniona całość. Transakcja rozpoczyna się z chwilą, gdy wykonywana jest pierwsza instrukcja SQL po instrukcji commit lub rollback lub też bezpośrednio po połączeniu się z bazą danych.

ROLLBACK [WORK | TRANSACTION]

Polecenie służy do cofania prac wykonywanych w bieżącej transakcji na bazie danych. Użycie instrukcji rollback powoduje zakończenie transakcji, unieważnia wszystkie zmiany wprowadzone podczas bieżącej transakcji zacierając wszystkie punkty zabezpieczeń oraz zwalnia blokady nałożone na obiekty bazy danych a związane z bieżącą transakcją.

BEGIN WORK; rozpoczyna transakcję w trybie związanym

COMMIT WORK ; zatwierdzenie transakcji - zakończenie bez możliwości cofnięcia

ROLLBACK WORK; cofa prace wykonywane w bieżącej transakcji na bazie danych

23.3 Tworzenie bazy i użytkowników

```
su - postgres  
createdb marco  
createuser marco
```

24 Visual Basic

24.1 Formularz w trybie „tylko do odczytu”

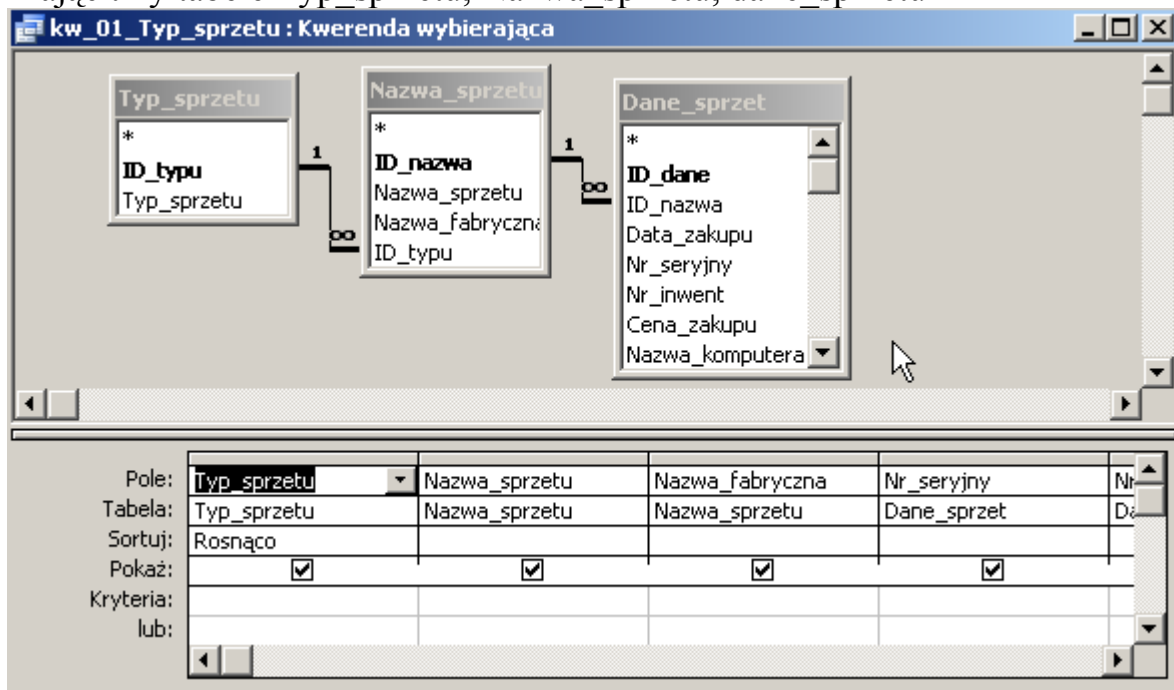
2x myszką kliknąć na danym formularzu. Widok -> Widok projektu

Lewy przycisk myszki w elemencie dla całego formularza i wyświetlić właściwości. Zakładka Dane Edycja dozwolona zmienić na Nie

24.2 Wyszukiwanie i filtrowanie rekordów

24.2.1 Pole kombinowane do wyszukiwania rekordów

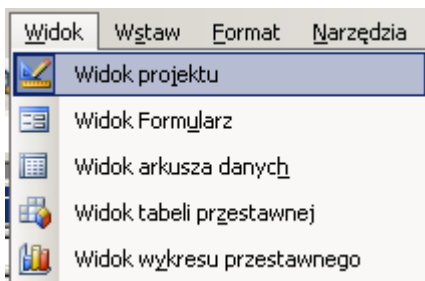
Mając trzy tabele Typ_sprzetu, Nazwa_sprzetu, dane_sprzetu



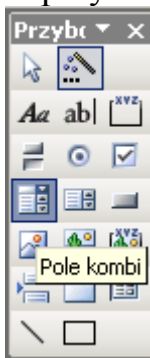
tworzymy formularz, którego zadaniem będzie wyświetlenie typu sprzętu i jego danych. W istniejącym formularzu utworzymy pole kombinowane w polu typ sprzętu. Uwaga **typ sprzętu** z napisem **Cluster** to już efekt końcowy

Nr seryjny	Nr inwentarzewy	Nazwa komputera	K	N	T	Pokój	Cena zakupu	Miasto	Gniazdo
77-C9027	IV-491-341		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	12	4 514,00 zł	Ostrów Maz.	
77-C2235	IV-491-295		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	314	5 612,00 zł	Ostrołęka	
77-C2236	IV-491-296		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	12	4 514,00 zł	Przasnysz	
77-D9195	IV-491-265		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	5	6 100,00 zł	Maków Maz.	
77-D3781	IV-491-340		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	12	5 311,88 zł	Wyszaków	
77-E8132	IV-491-339		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	314	5 612,00 zł	Ostrołęka	

Otwórz formularz i przejdź do widoku projekt – Widok -> Widok projektu



Utwórz pole kombinowane wyszukujące rekordy wykorzystujący **Kreator formantów** i w przyborniku zaznaczamy narzędzie **pole kombi**

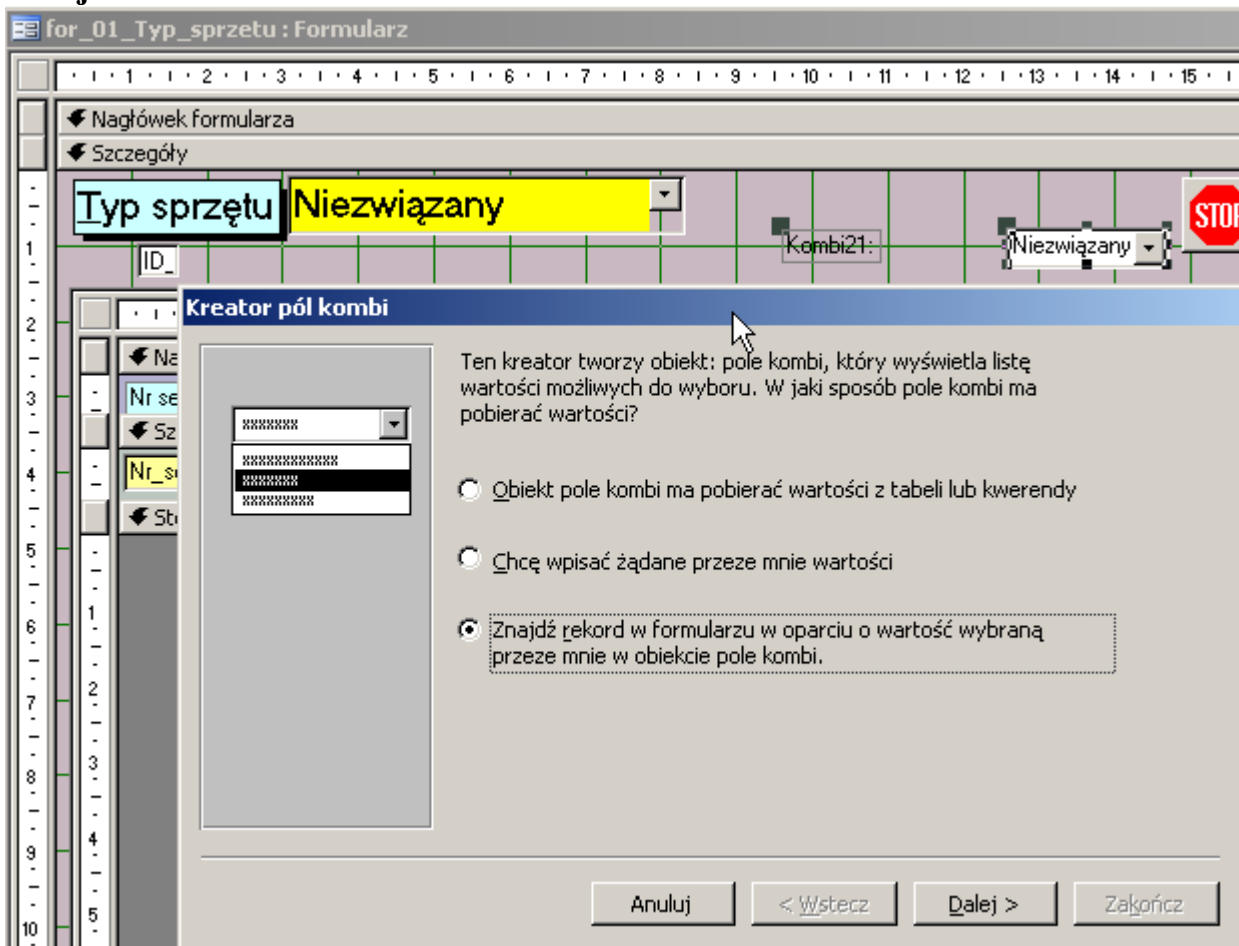


i trzymając lewy przycisk myszki zaznaczamy w wybranym miejscu w formularzu i zaznaczamy dany obszar

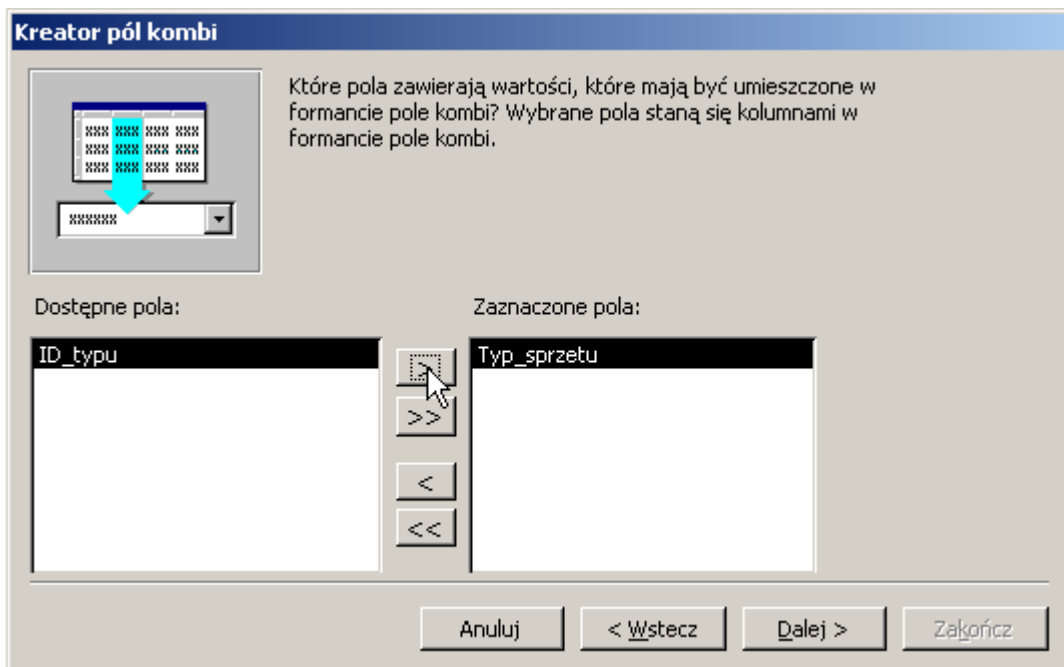


Zapamiętaj nazwę **Kombi21** (uwaga: do tej nazwy będziemy się odwoływać) ale będziesz mógł ją zmienić...

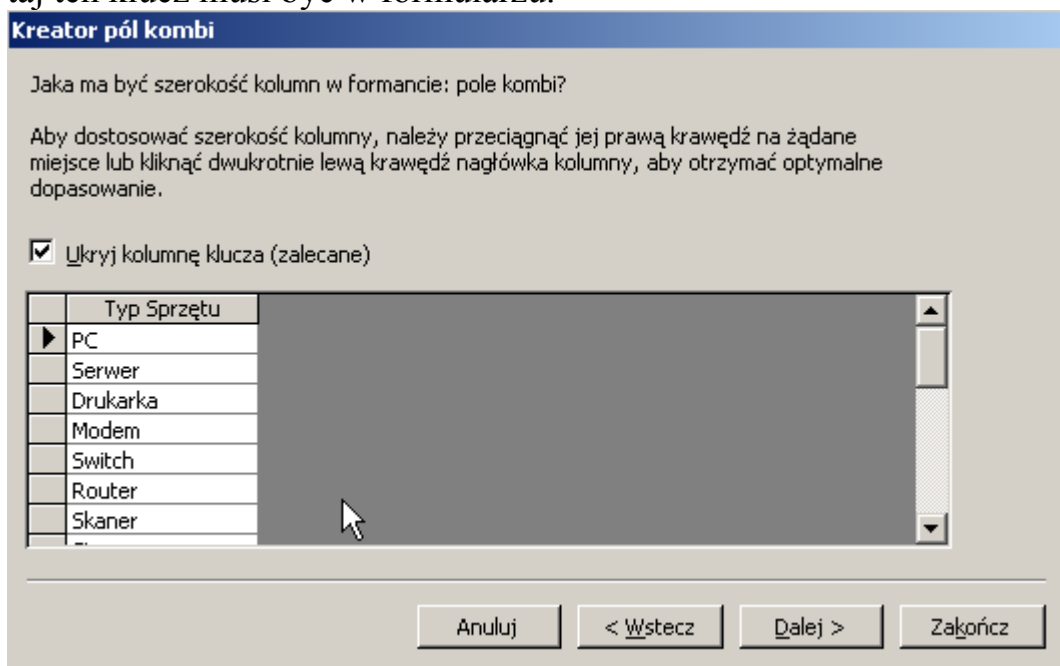
Następnie zaznaczamy trzecią opcję – Znajdź rekord w formularzu i kliknij przycisk **Dalej**



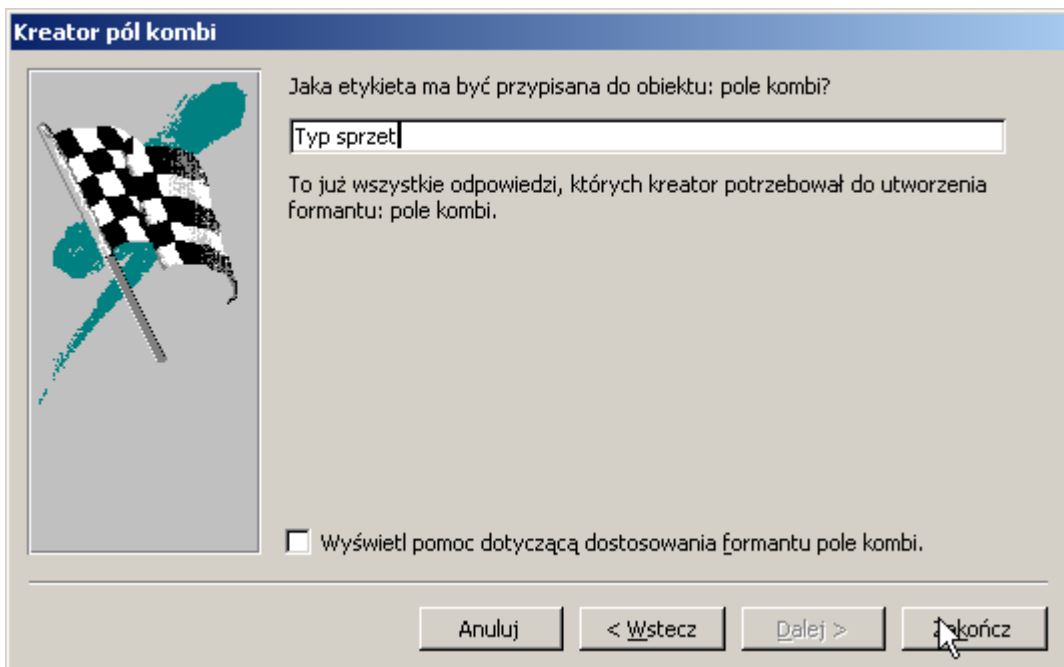
Klikamy dwukrotnie na pole **typ_sprzetu** aby przenieść je do listy **Zaznaczone pola**:



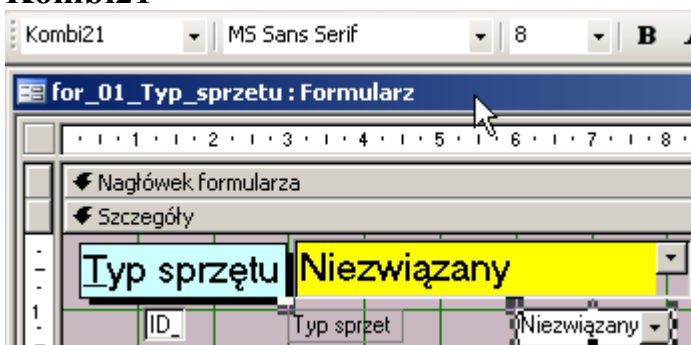
Kreator Wyświetli dane z tabeli **Typ_Sprzetu** i masz szansę ustalenia odpowiedniej szerokości pola z wyświetlanymi danymi oraz zaznacz **Ukryj kolumnę klucza** (ale pamiętaj ten klucz musi być w formularzu).



Na koniec możesz nadać etykietę danemu polu kombinowanemu np. **typ sprzętu**. Jak chcesz utworzyć np. skrót klawiszowy ALT+T to nadaj mu nazwę **&Typ sprzet**.

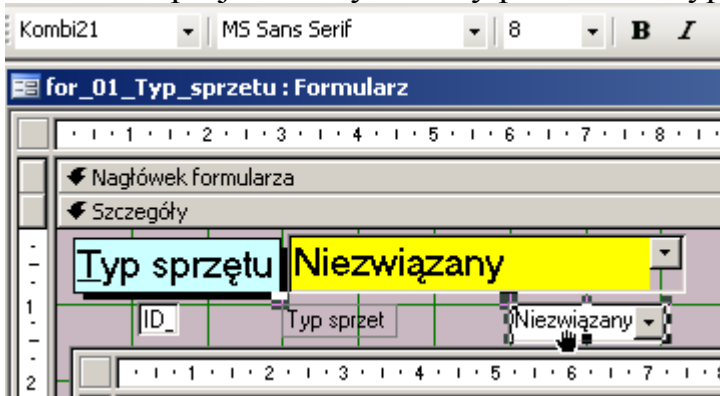


Efekt końcowy w formularzu jak na rysunku poniżej.. zapamiętaj nazwę pola kombi **Kombi21**

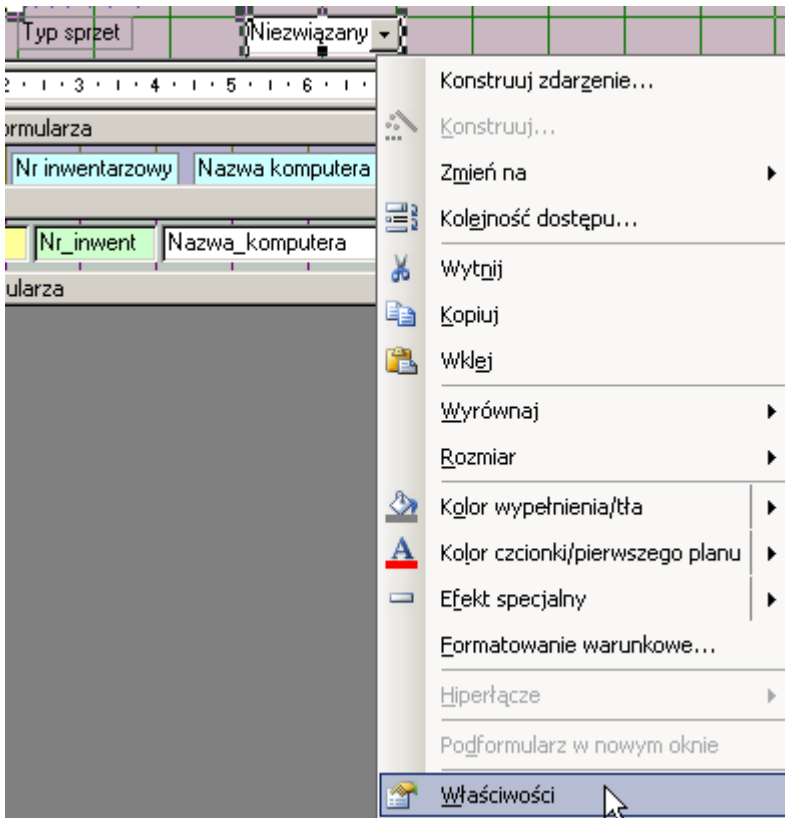


Lecz efekt nas nie zadawała. Po pierwsze dane w polu kombi jest nie posortowane i pole nie jest automatycznie na wstępie aktualizowane.

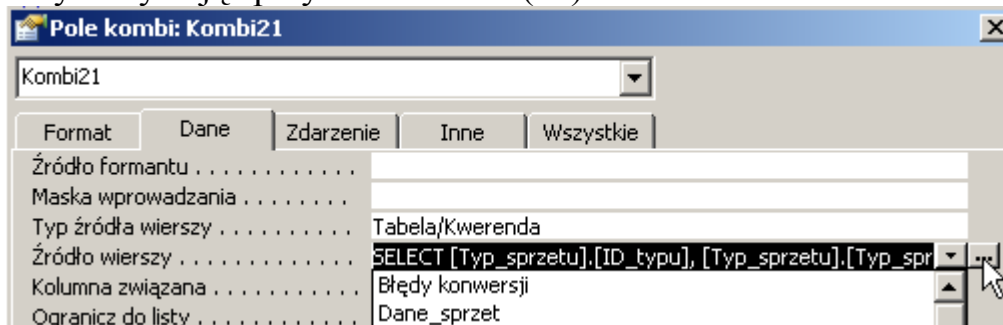
W widoku projekt uaktywniamy pole kombi typ sprzęt (Kombi21)



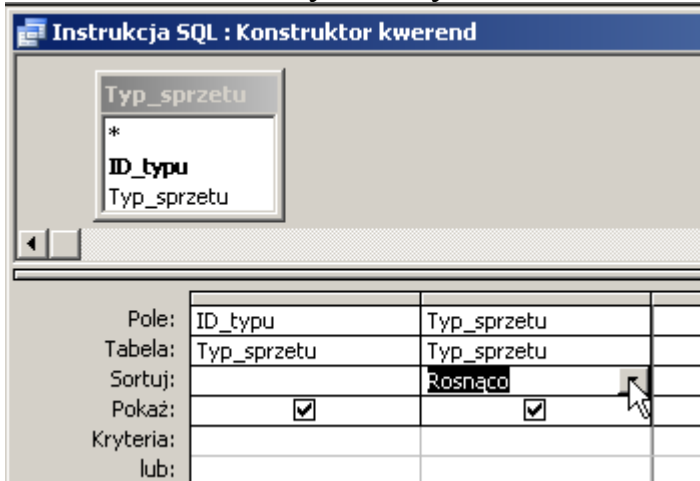
Po naciśnięciu prawego przycisku myszki uaktywniamy właściwości



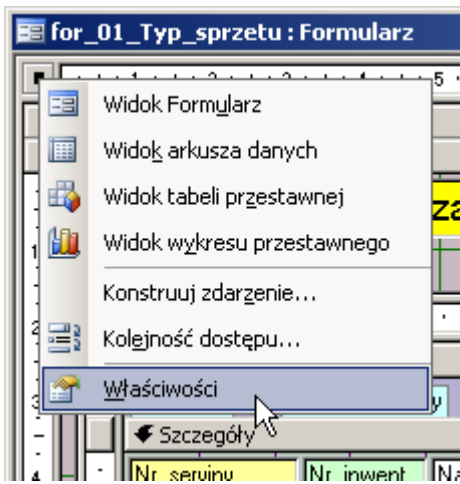
Uaktywniamy zakładkę Dane i poszukujemy Źródło wierszy ..
I wykorzystując przycisk kreatora (...)



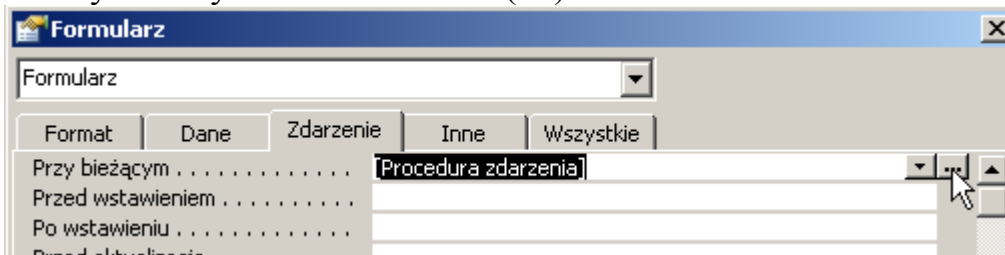
W siatce kwerend wybieramy sortowanie danego pola.



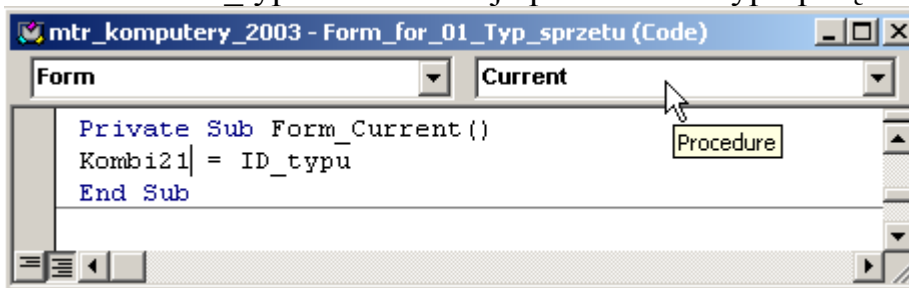
Teraz pozostało nam tylko wpisanie kodu umożliwiające synchronizację pola kombi
W widoku projekt formularza uaktywniamy właściwości ale dla całego formularza w polu jak poniżej (lewy górny róg w postaci).



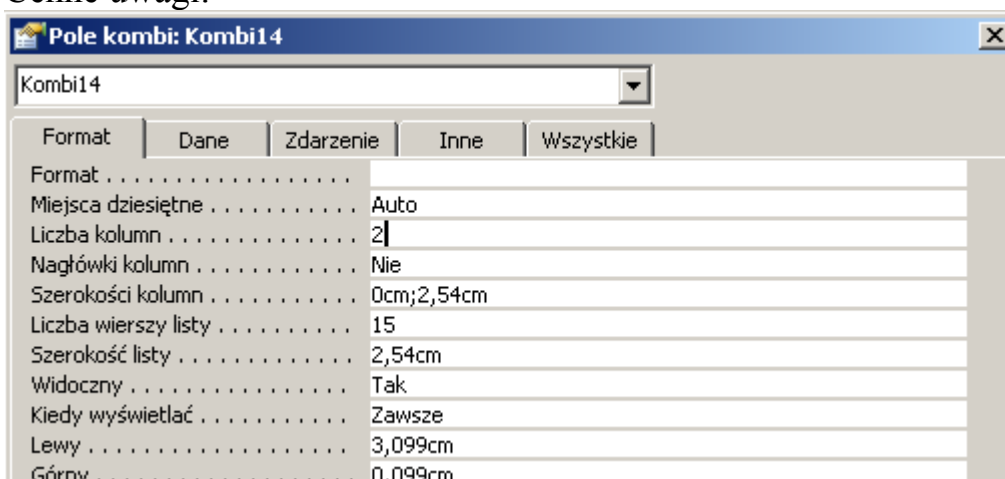
Przechodzimy do zakładki **Zdarzenia** i poszukujemy **Przy bieżącym..**
I uaktywniamy konstruktor kodu (...)



Pojawia się moduł formularza for_01_Typ_sprzetu:Formularz, w którym automatycznie zostaje wstawiona procedura zdarzenia Form_Current. I wpisujemy Kombi21 = ID_typu 'Aktualizacja pola kombi Typ Sprzętu



Zachowaj zmiany i to już koniec.
Cenne uwagi:



Pole Liczba kolumn ... (2) oznacza że w polu kombi będą widoczne dwie kolumny .. ale pamiętasz że jedno pole zaznaczyłeś jako ukryte ..

Szerokość kolumny... (0cm;2,54cm) możesz ręcznie zmienić współrzędne odpowiedzialne za szerokość pola

Liczba wierszy listy (15) oznacza że po uaktywnieniu listy będzie widoczne 15 elementów..

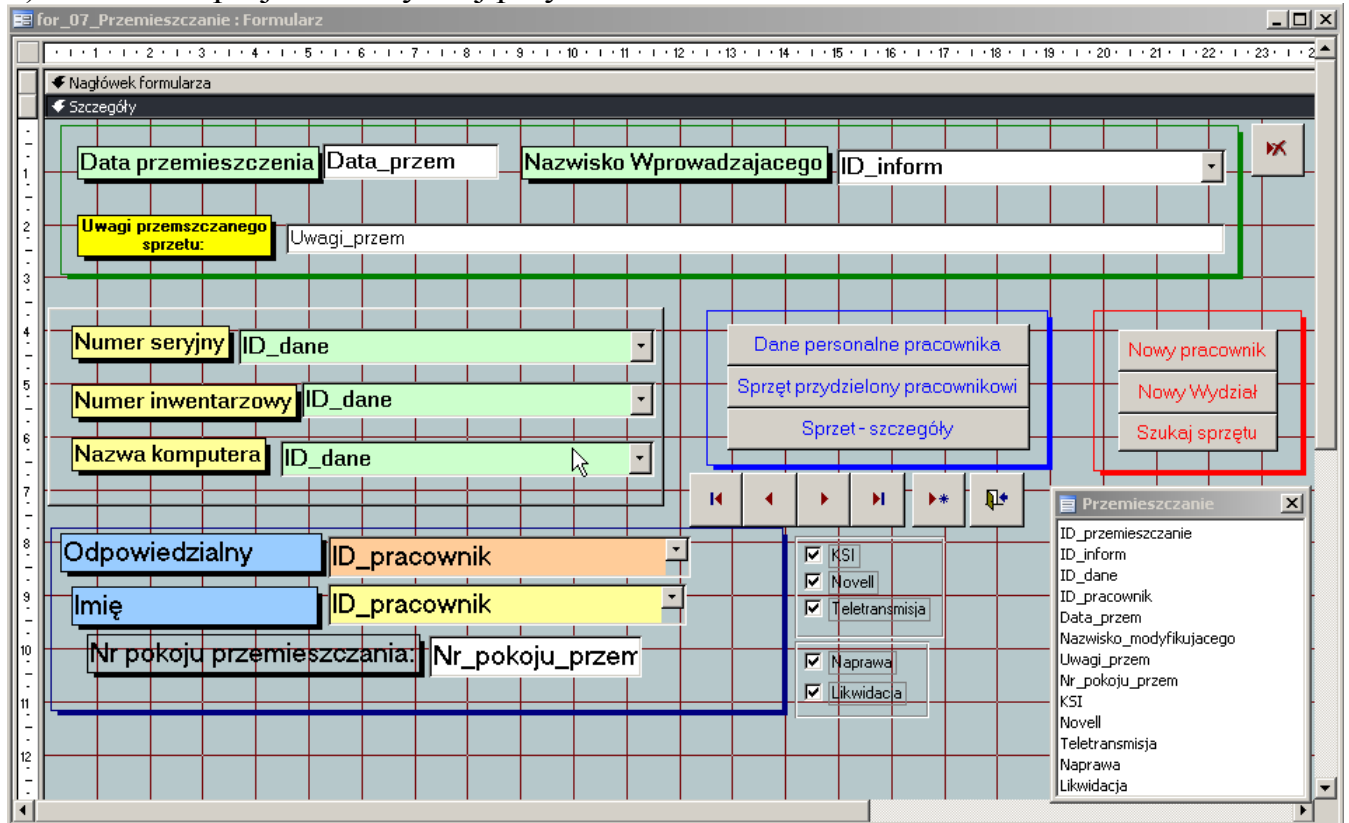
24.3 Filtrowanie danych

24.3.1 Utwórz grupę opcji

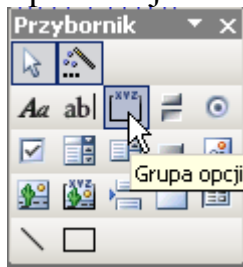
Utworzenie pola kombi umożliwia przejście do określonego rekordu w zestawie danych, jednakże pozostałe dane są wciąż dostępne. Często jednak chcesz ukryć pewne dane lub w celu selekcji i szybszego znajdowania zastosować filtr.

Zakładamy że mamy formularz do wprowadzania danych oraz użytkownik podejmuje decyzję czy sprzęt jest w sieci KSI, Novell czy teletransmisji. Naszym zadaniem będzie utworzenie filtru sprzętu tylko w KSI lub Novell-u lub teletransmisji lub nie przydzielony lub bez filtru całość

1) W widoku projekt uaktywnij przybornik

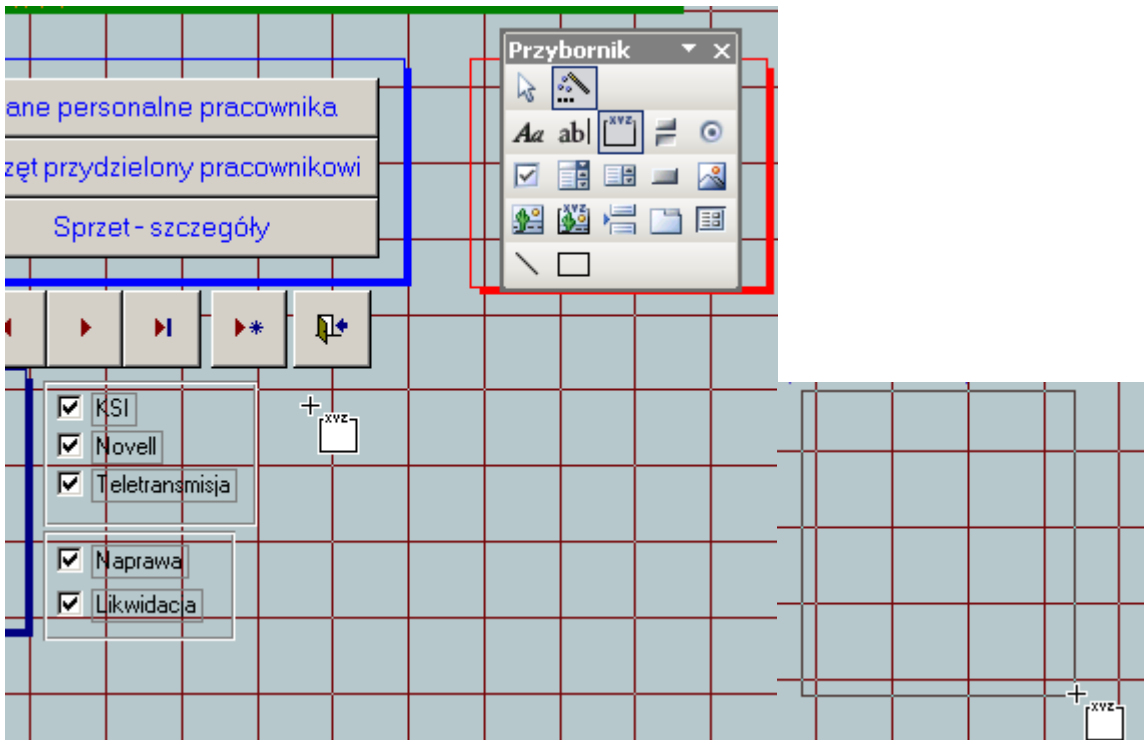


I poszukaj narzędzia grupa opcji

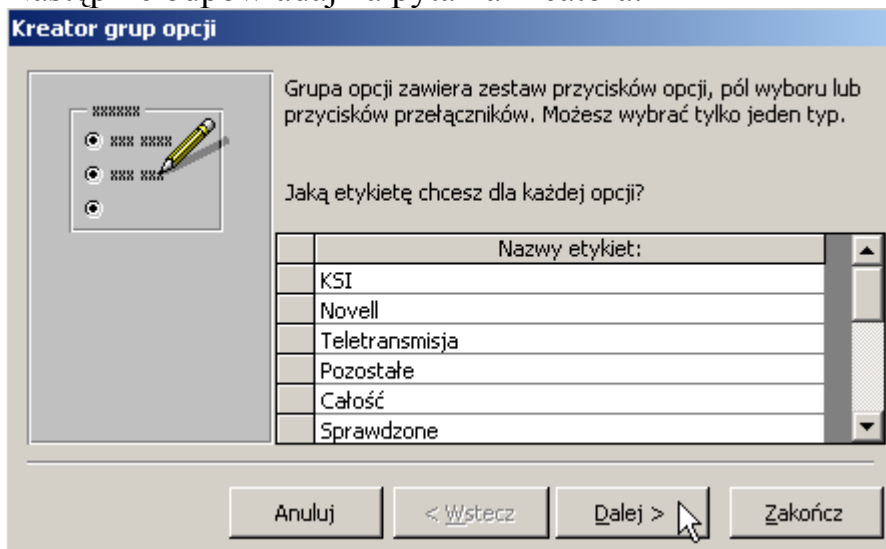


Znajdź wolne miejsce na formularzu. W naszym przypadku będzie wolne w miejscu Lista pól. Jest uaktywniona ze względu na fakt, że jest widać jakie nazwy są dostępne w tym formularzu. Zwróć uwagę na pola KSI, Novell, Teletransmisja, Naprawa.

Trzymając lewy przycisk myszki zaznacz pole na grupę opcji.



Następnie odpowiadaj na pytania kreatora:



I nadaj etykiety widoczne w grupie opcji czyli :

KSI, - sprzęt w KSI

Novell – sprzęt podłączony do sieci Novell

Teletransmisja – sprzęt podłączony do mainframe

Pozostałe – sprzęt nie przydzielony do żadnej grupy,

Całość – cały sprzęt,

Sprawdzone – sprzęt sprawdzany np. w czasie inwentaryzacji z natury

Nie sprawdzone – sprzęt nie odznaczony czyli ten który należy szukać

Następnie zaznaczasz – Nie, nie chcę wartości domyślnej

Kreator grup opcji

Czy chcesz, aby jedna z opcji była wyborem domyślnym?

Tak, domyślny wybór to:

Nie, nie chcę wartości domyślnej

Anuluj < Wstecz Dalej > Zakończ

Zwróć uwagę na wartości dla poszczególnych etykiet

Kreator grup opcji

Kliknięcie opcji w grupie opcji ustawia wartość grupy opcji na wartość wybranej opcji.

Jaką wartość chcesz przypisać każdej z opcji?

Nazwy etykiet:	Wartości:
KSI	1
Novell	2
Teletransmisja	3
Pozostałe	4
Całość	5
Sprawdzone	6
Nie sprawdzone	7

Anuluj < Wstecz Dalej > Zakończ

Na pytanie Co chcesz zrobić z wartością wybranej opcji? Wybierz Zapisz wartość ...

Kreator grup opcji

Możesz przechować wartość zaznaczonej opcji w polu lub użyć tej wartości, aby później wykonać zadania takie, jak drukowanie raportu.

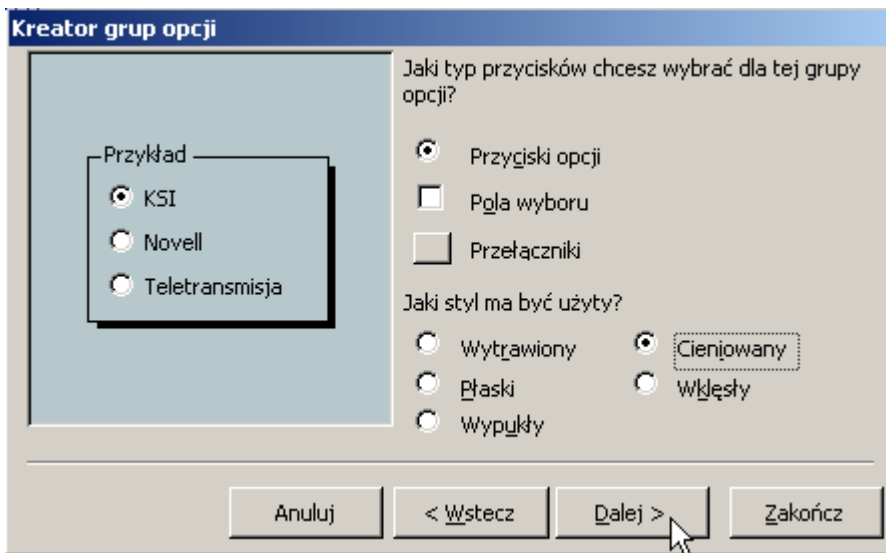
Co chcesz zrobić z wartością wybranej opcji?

Zapisz wartość do późniejszego wykorzystania

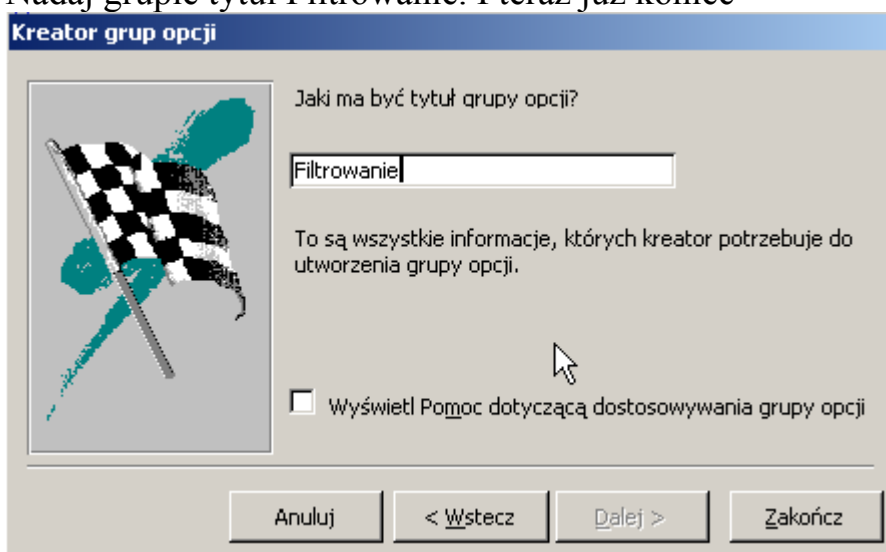
Przechowaj wartość w tym polu:

Anuluj < Wstecz Dalej > Zakończ

Typ przycisku musisz wybrać jako przycisk opcji pozostałe elementy zależą od twojej wyobraźni.



Nadaj grupie tytuł Filtrowanie. I teraz już koniec

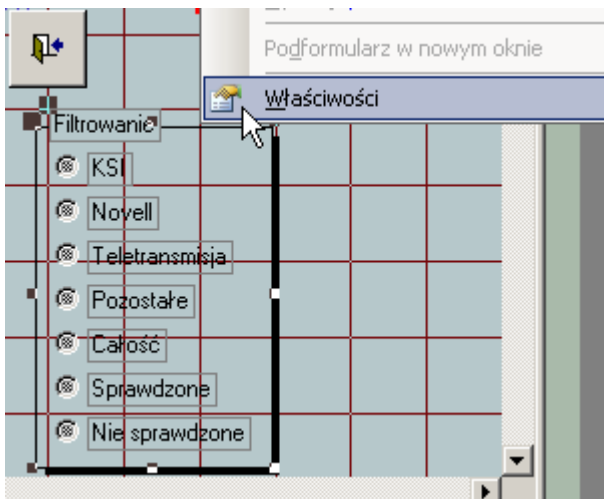


I efekt to:

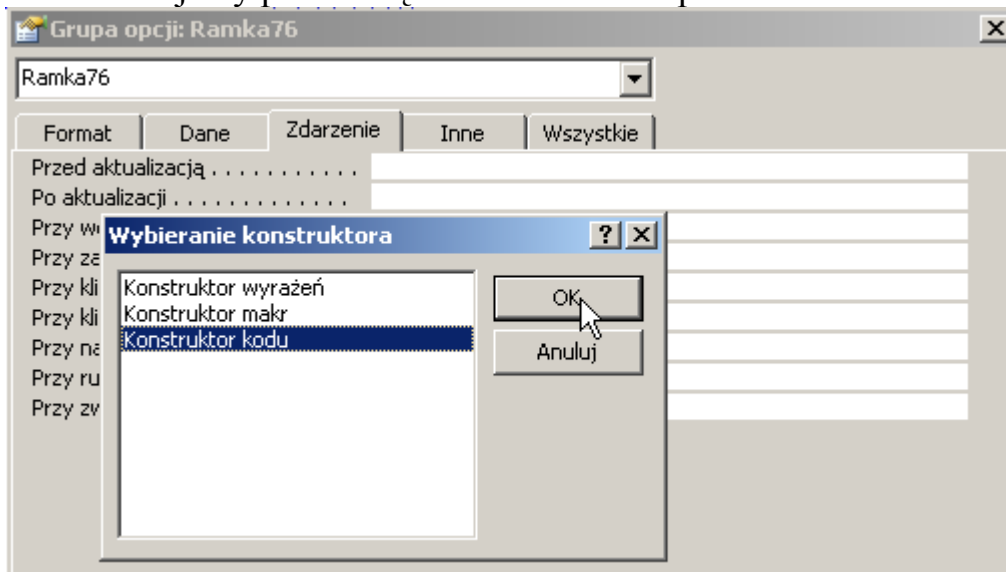


24.3.2 Utwórz procedurę zdarzenia uruchamiającą grupę opcji.

Uaktywniając grupę opcji prawym przyciskiem myszki zaznaczamy ich właściwości:



Następnie klikając na zakładkę Zdarzenie i uaktywniając Po aktualizacji Konstruktor kodu i definiujemy procedurę zdarzenia AfterUpdate



Następująco:

25 Podsumowanie

25.1 Postacie normalne – ogólnie

25.1.1 Pierwsza

Pierwsza postać normalna wymaga, aby na każdym przecięciu wiersza i kolumny występowała tylko jedna wartość i aby wartość ta była atomowa. w tabeli spełniającej warunki pierwszej postaci normalnej nie może być powtarzających się grup danych. Tam gdzie występują powtarzające się kolumny, poprawny projekt wymaga wprowadzenia tabeli nadrzędnej.

25.1.2 Druga

Druga reguła normalizacji mówi, że każda kolumna nie należąca do klucza (niekluczowa) musi zależeć od całego klucza głównego. Wynika stąd, że tabela nie może zawierać kolumny niekluczowej odnoszącej się tylko do jakiejś części złożonego klucza głównego.

25.1.3 Trzecia

Trzecia postać normalna jest związana z zasadą wyrażoną przez drugą postać normalną w bardziej ogólny sposób: nie jest ona ograniczona tylko do złożonych kluczy głównych. Trzecia postać normalna wymaga, żeby żadna kolumna niekluczowa nie zależała od innej kolumny niekluczowej. Każda kolumna nie będąca kluczem musi być związana z kolumną klucza głównego.

26 Elementy teorii relacyjnych baz danych

26.1 Model relacyjnych baz danych

Relacyjna baza danych oznacza bazę zbudowaną z relacji. Podstawowy obiekt takiej bazy danych, tabela, jest konkretną reprezentacją relacji – technicznego pojęcia matematyki.

26.1.1 Definicja 1. Schemat relacji

Schemat relacji nazywamy zbiór $R = \{A_1, A_2, A_3, A_4, A_5, A_6, \dots, A_n\}$, gdzie $A_1, A_2, A_3, A_4, A_5, A_6, \dots, A_n$ są atrybutami reprezentowanymi w tabeli poprzez nazwy kolumn. Na przykład, schemat relacji DANE jest zbiór $\{IDDANE, NRSER, NRINW, DATAP, WARTOSC\}$

Definicja 2. Dziedzina relacji

Każdemu atrybutowi ($A_1, A_2, A_3, A_4, A_5, A_6, \dots, A_n$) przyporządkowana jest **dziedzina** (zakres dopuszczalnych wartości atrybutu) reprezentowana przez typ danych. W przypadku tabeli DANE poszczególne kolumny są następujących typów: IDDANE – auto-numerowanie, liczba całkowita długa, NRSER – tekst(30), NRINW- liczba całkowita długa, DATAP – data, Wartość – waluta.

Dziedziną relacji o schemacie $R = \{A_1, A_2, A_3, A_4, A_5, A_6, \dots, A_n\}$ nazywamy sumę dziedzin wszystkich jej atrybutów $DOM(R) = DOM(A_1) \cup DOM(A_1) \cup DOM(A_2) \cup \dots \cup DOM(A_n)$

Dziedzina relacji DANE jest suma dziedzin wymienionych wyżej kolumn.

26.1.2 Definicja 3. Relacje

Relacją o schemacie $R = \{A_1, A_2, A_3, A_4, A_5, A_6, \dots, A_n\}$ nazywamy skończony zbiór $r = \{t_1, t_2, t_3, \dots, t_m\}$ odwzorowań $t_i: R \rightarrow DOM(R)$ takich, że dla każdego j z zakresu $1 \leq j \leq n$ zachodzi zależność: $t_i(A_j) \in DOM(A_j)$

Tak zdefiniowane pojedyncze odwzorowanie nosi nazwę krotki i odpowiada mu pojedynczy wiersz tabeli. Wartością krotki jest suwa wartości poszczególnych atrybutów obiektu. Na przykład, w tabeli DANE zapisane są następujące dane 1, YBGV128745, 706, 2006-02-12, 12500 zł.

26.2 Zasady dotyczące struktury danych

26.2.1 Postulat informacyjny.

Na poziomie logicznym dane są reprezentowane wyłącznie przez tabele.

26.2.2 Postulat dostępu.

Każdą pojedynczą informację możemy odczytać, odwołując się do tabel, kolumn i wartości kluczy podstawowych

26.2.3 Postula fizycznej niezależności danych.

Zmiana w sposobie przechowywania danych i dostępu do nich nie wpływają na aplikacje kliencką.

26.2.4 Postulat logicznej niezależności danych.

Zmiany w tabelach, zachowujące informację i dopuszczalne semantycznie, nie mają wpływu na aplikację kliencką.

26.2.5 Postulat niezależności dystrybucyjnej

System i jego język umożliwiają dostęp do danych zapisanych w różnych miejscach, np. na wielu komputerach w sieci.

26.2.6 Postulat zabezpieczenia przed operacjami na niższym poziomie abstrakcji.

Jeśli system zarządzania bazą danych umożliwia bezpośrednio operacje na niższych poziomach abstrakcji, nie mogą one naruszać reguł relacyjnego modelu baz danych, w szczególności nie mogą pomijać ograniczeń określonych przez więzy spójności.

26.3 Zasady dotyczące pobierania i modyfikowania danych

Dane przechowywane w bazach danych nie są niezmiennie. Wręcz przeciwnie – aby baza była przydatna musi być cały czas aktualizowana. Operacje modyfikowania danych nie mogą jednak naruszać struktury danych. Dlatego operacje odczytywania i modyfikowania danych muszą przebiegać z zachowaniem poniższych zasad.

Pierwszy typ przekształceń, które można wykonywać na wartościach atrybutów wynika z definicji relacji jako zbioru.

Operatorom algebry zbiorów, czyli :

\cup (suma relacji) odpowiada operator algebry relacji **UNION**

Tabela 6 Suma relacji

A[0]		\cup	B[0]		=	C[0]	
A[1]			B[1]			C[1]	
A[2]			B[2]			C[2]	
A[3]			B[3]			C[3]	
A[4]			B[4]			C[4]	

Na sumę dwóch relacji $r \cup s$ składają się wszystkie elementy relacji r i wszystkie elementy relacji s

\cap (część wspólna, przecięcie, przekrój relacji) odpowiada operator **INTERSEC**

Tabela 7 Przecięcie relacji

A[0]		\cap	B[0]		=	C[0]	
A[1]			B[1]			C[1]	
A[2]			B[2]			C[2]	
A[3]			B[3]			C[3]	
A[4]			B[4]			C[4]	

Wynikiem przecięcia relacji $r \cap s$ jest relacja składająca się z elementów wspólnych relacji r i s

$-$ (dopełnienie relacji, różnica relacji) odpowiada operator **EXCEPT**

Tabela 8 dopełnienie relacji

A[0]		\setminus	B[0]		=	C[0]	
A[1]			B[1]			C[1]	
A[2]			B[2]			C[2]	
A[3]			B[3]			C[3]	
A[4]			B[4]			C[4]	

Wynikiem odejmowania relacji $r \setminus s$ jest relacja powstała na skutek usunięcia z relacji r wszystkich elementów wchodzących w skład relacji s

26.4 Pierwsza postać normalna

Encja znajduje się w pierwszej postaci normalnej, jeśli wszystkie jej atrybuty mają pojedynczą, niepodzielną wartość. Jeśli jakkolwiek atrybut składa się z kilku wartości, encja nie spełnia wymogów pierwszej postaci normalnej. Innymi słowy, encja spełnia za-

łożenie pierwszej postaci normalnej, jeżeli posiada tylko pola zawierające wartości atomowe (niepodzielne).

Gdyby wszystkie tabele bazy danych spełniały pierwszą postać normalną, to związki między tabelami byłyby tylko jeden do wielu (nigdy w obrębie jednej tabeli)

26.5 Druga postać normalna

Encja odpowiada wymogom drugiej postaci normalnej, jeżeli spełnia dwa założenia spełnia wymogi 1PN

kolumny nie wchodzące w skład klucza głównego są niezależne od całego klucza.

Tak więc encja jest w drugiej postaci normalnej, jeśli znajduje się w pierwszej postaci normalnej i wszystkie zależne funkcyjnie atrybuty są zależne od całego неповtarzalnego identyfikatora (klucza podstawowego głównego) Jeśli jakkolwiek atrybut nie jest całkowicie zależny od klucza głównego, należy zmodyfikować projekt bazy danych. Atrybuty takie normalizuje się albo poprzez odnalezienie encji, którą bezpośrednio opisują te atrybuty, albo przez utworzenie dodatkowej encji, do której atrybut powinien zostać dodany

26.6 Trzecia postać normalna

Encja jest w 3PN, jeśli wszystkie atrybuty spełniają założenia 2 PN, a ponadto każdy z atrybutów nie należących do klucza głównego nie zależy funkcjonalnie od jakiegokolwiek nadklucza. Innymi słowy, encja znajduje się w 3PN, kiedy jest już w 2PN i żadne nieidentyfikujące atrybuty nie są zależne od innych nieidentyfikujących atrybutów (nie występują zależności przechodnie)

W praktyce przekształcenie modelu bazy danych do 3PN często sprowadza się do wyodrębnienia encji do tej pory traktowanej jako zbiór atrybutów opisujących encje nadrzędne i utworzenia dla nich odrębnych tabel słownikowych.

26.7 Tabela

26.7.1 Tworzenie

```
CREATE TABLE nazwa_tabeli
    (nazwa_pola1 typ_danych1 ograniczenie1,
    nazwa_pola2 typ_danych2 ograniczenie2,
    nazwa_pola3 typ_danych3 [NULL|NOT NULL] [DEFAULT wartosc_domyslna),
    PRIMARY KEY (nazwa_pola1)
    FOREIGN KEY nazwa_klucza_obcego (nazwa_pola_kolumny_obcej)
    REFERENCES tabela_klucza_podstawowego (nazwa_pola1)
    ON UPDATE czynność_przy_modyfikacji
    ON DELETE czynność_przy_kasowaniu;
```

26.7.2 Wstawianie

```
INSERT INTO nazwa_tabeli (nazwa_kol1, nazwa_kol2 ....)
VALUES (stala1, stala2 ...);
```

26.7.3 Modyfikacja

```
UPDATE nazwa_tabeli
SET nazwa_kolumny = wyrażenie
WHERE warunki_wyszukiwania;
```

26.7.4 Usuwanie wszystkich danych – całej tabeli

DROP TABLE nazwa_tabeli;

26.7.5 Usuwanie wybranych danych

DROP TABLE nazwa_tabeli

WHERE warunek_wyszukiwan;

26.7.6 Wstawianie dodatkowej kolumny

ALTER TABLE nazwa_tabeli

ADD nazwa_kolumny typ ograniczenie;