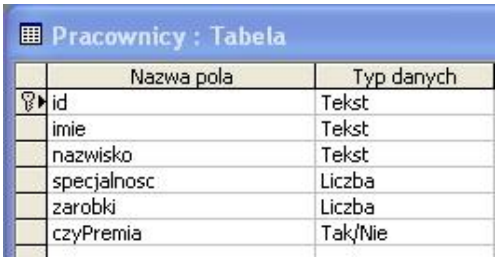
**KÓŁKO INFORMATYCZNE - ACCESS**

## 1. UTWÓRZ BAZĘ DANYCH ZAWIERAJĄCĄ TABELE:

Pacjenci : Tabela

| Nazwa pola | Typ danych |
|---|---|
| idPacjenta | Tekst |
| nazwiskoPacjenta | Tekst |
| imiePacjenta | Tekst |
| lekarzRodzinny | Tekst |
| ubezpieczenie | Tekst |
| dataUrodzenia | Data/Godzina |
| pesel | Tekst |
| adres | Tekst |
| telefon | Liczba |

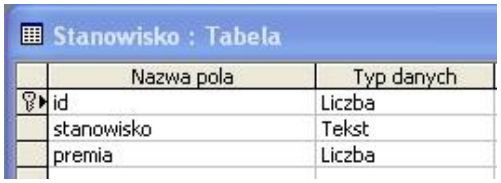


Pracownicy : Tabela

| Nazwa pola | Typ danych |
|---|---|
| id | Tekst |
| imie | Tekst |
| nazwisko | Tekst |
| specjalnosc | Liczba |
| zarobki | Liczba |
| czyPremia | Tak/Nie |

Wizyty : Tabela

| Nazwa pola | Typ danych |
|---|---|
| ID | Autonumerowai |
| idPacjenta | Tekst |
| idLekarza | Tekst |
| dataWizyty | Data/Godzina |



Stanowisko : Tabela

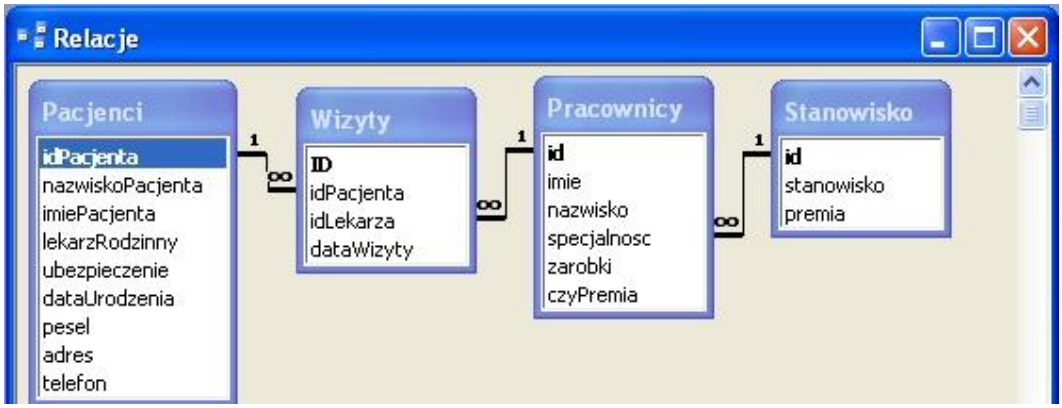
| Nazwa pola | Typ danych |
|---|---|
| id | Liczba |
| stanowisko | Tekst |
| premia | Liczba |

**Wskazówki dotyczące importowania tabel:**

- W Access 2003 importujemy pliki tekstowe: Plik / Pobierz dane zewnętrzne / Importuj / Wskaż plik tekstowy / Ograniczony / średnik / w istniejącej tabeli
- PESEL - nie da się zaimportować pola PESEL jako liczbę(nie mieści się w typie danych) dlatego importujemy jako tekst
- liczbę zmiennoprzecinkową (np. stopa procentowa 0.05) importować jako podwójna precyzja
- Importowanie pola TAK/NIE nie spowoduje ładnego widoku checkboxa. Pole te wynosi -1, gdy mamy wartość TAK, natomiast NIE to 0. W warunkach kwerend możemy się odwoływać do wartości yes lub no. Problem można obejść importując pole jako tekst. W naszej bazie mamy pole Pacjenci.ubezpieczenie (jako tekst) oraz pole Pracownicy.czyPremia (jako TAK/NIE)
- Jeśli podczas importowania pojawi się dodatkowa niepotrzebna kolumna to usuń w Widoku projekt lub na poziomie importowania pominąć
- Jeśli przy importowaniu zauważymy, że jest problem z wyświetlaniem polskich liter to podczas importu wybieramy przycisk „Zaawansowane” dla danego pola i możemy zmienić kodowanie
- Każda tabela powinna mieć zdefiniowany klucz (key), którego wartość jednoznacznie identyfikuje dany wiersz. To on reprezentuje całą tabelę w bazie danych. Dzięki niemu możliwe jest tworzenie relacji pomiędzy tabelami.
- Jeśli zaimportujemy pola o nazwie zawierającej minimum dwa wyrazy musimy później używać tych nazw zapisując je w nawiasach kwadratowych

## 2. UTWÓRZ RELACJE ODPOWIEDNIEGO TYPU, WŁAŚCIWE DLA STRUKTURY PRZECHOWYWANYCH INFORMACJI.

- Narzędzia / Relacje / Dodaj wszystkie
- Powiązania twórz od tabel, gdzie danej wartości jest więcej (w tabeli zawodnicy więcej zawodników niż w konkretnym konkursie)
- Pamiętaj o włączeniu więzów integralności




## 3. PODSTAWOWE FRAZY INSTRUKCJI SELECT

- Pozwala obejrzeć dane z bazy danych - nie modyfikuje ich
- Niezbędne jest użycie frazy select i from

### 3.1. Fraza select ... from – wybierz z

#### 3.1.1. wszystko – użycie znaku *

```
SELECT *
FROM pacjenci;
```

#### 3.1.2. wybrane elementy – oddzielamy przecinkiem

```
SELECT nazwiskoPacjenta, imiePacjenta FROM pacjenci;
```

#### 3.1.3. Distinct - eliminuje powtarzające się wiersze

```
SELECT DISTINCT nazwiskoPacjenta, imiePacjenta FROM pacjenci;
```

!!! pamiętajmy, że DISTINCT jest dosyć nieczuły i nie obchodzi go, że mamy dwie różne Augustowskie Justyny

### 3.2. Fraza where – gdzie

```
SELECT nazwiskoPacjenta, imiePacjenta FROM pacjenci
WHERE ubezpieczenie='TAK';
```

### 3.3. Fraza order by – porządkuj według

Sortowanie danych:

- dla danych tekstowych od A do Z
- dla liczb od najmniejszej do największej
- dla dat zgodnie z chronologią

#### 3.3.1. od najstarszego do najmłodszego

```
SELECT nazwiskoPacjenta, imiePacjenta, dataUrodzenia FROM pacjenci
WHERE ubezpieczenie='TAK' ORDER BY dataUrodzenia;
```

### 3.3.2.od najmłodszego do najstarszego: DESC (zmiana kolejności uporządkowania)

SELECT nazwiskoPacjenta, imiePacjenta, dataUrodzenia FROM pacjenci
WHERE ubezpieczenie='TAK' ORDER BY dataUrodzenia DESC;

## 3.4. Top

- Zwraca określoną liczbę rekordów, które mieszczą się w górnej lub dolnej części zakresu określonego przez klauzulę ORDER BY
- TOP nie dokonuje wyboru między równymi wartościami. Jeśli miejsce np. 3 będzie miało taką samą wartość jak 4 rekord, kwerenda zwróci 4 rekordów
- Wartość umieszczona po orzeczeniu TOP musi być liczbą całkowitą
- Można dodać słówko PERCENT i zwróci określony procent rekordów

Wyświetl 3 najstarszych pacjentów
SELECT TOP 3 nazwiskoPacjenta, imiePacjenta, dataUrodzenia FROM pacjenci
WHERE ubezpieczenie='TAK' ORDER BY dataUrodzenia;

Wyświetl 10 % najstarszych pacjentów
SELECT TOP 10 PERCENT nazwiskoPacjenta, imiePacjenta, dataUrodzenia FROM pacjenci
WHERE ubezpieczenie='TAK' ORDER BY dataUrodzenia;

## 3.5. Warunki

- Aby skorzysta z warunków porównania można użyć:
  = <> > < >= <=
- dla znaków „>" oznacza „po" i korzysta z porządku znaków w kodzie ASCII
- aby skonstruować warunki złożone (kilka w jednym zapytaniu) używamy warunków logicznych: AND, OR, NOT

Warunki złożone
- zrób zestawienie zarabiających minimum 2000, ale nie więcej niż 4000

SELECT nazwisko, imie, zarobki FROM pracownicy
WHERE zarobki>=2000 and zarobki<=4000;

Zastosowanie OR i AND jednocześnie:
- należy pamiętać o priorytetach i kolejności wykonywania warunków
- przy zastosowaniu warunku OR najlepiej ustalać kolejność ich wykonywania stosując nawiasy

Zrób zestawienie zarabiających minimum 2000, ale nie więcej niż 4000 o specjalnościach 1 lub 2:
SELECT nazwisko, imie, zarobki FROM pracownicy
WHERE zarobki<4000 and **(**specjalnosc=1 or specjalnosc=2**)** and zarobki>=2000;

DOWÓD: W powyższym zadaniu opuść nawiasy i sprawdź różnice w wynikach

Operator between ... and …
Zastępuje podwójny warunek (patrz poprzednie zapytanie)
SELECT nazwisko, imie, zarobki FROM pracownicy
WHERE zarobki between 2000 and 4000;

Operator IN lub NOT IN

- pozwala na przeszukanie wartości zgodnych ze zbiorem znajdującym się po IN lub NOT IN
- wyświetl wszystkich internistów(1), pediatrów(2) i laryngologów(3) spośród pracowników

```
SELECT * FROM pracownicy WHERE specjalnosc in (1,2,3);
```

- wyświetl wszystkich nie internistów(1), pediatrów(2) i laryngologów(3) spośród pracowników

```
SELECT * FROM pracownicy WHERE specjalnosc not in (1,2,3);
```

### 3.6. Warunki dla tekstów – wieloznaczne przeszukiwania

- w tym celu wykorzystujemy konstrukcję LIKE lub NOT LIKE
- łańcuchy mogą zawierać metaznaki:
  *- zastępuje wszystkie znaki lub zero znaków (w normalnych bazach to %)
  ? - zastępuje pojedynczy dowolny znak (w normalnych bazach to _)

wyszukaj wszystkie kobiety wśród pracowników (imiona kończą się na ‘a’)

```
SELECT * FROM pracownicy WHERE imie like '*a';
```

wyszukaj pacjentów, którzy urodzili się w latach 70’

```
SELECT * FROM pacjenci WHERE pesel like '7?*';
```

### 3.7. Wyodrębnianie tekstu

- Left(pole, ile) – zwraca z danego pola ilość znaków od lewej podanych w drugim parametrze
- Right(pole, ile) – zwraca z danego pola ilość znaków od prawej
- Mid(pole, poczatek, ile) – z danego pola podaną ilość znaków od numeru znaku podanego w drugim parametrze

Wyświetl pacjentów urodzonych w maju na podstawie pola pesel:

```
SELECT nazwiskoPacjenta, imiePacjenta, pesel FROM pacjenci
WHERE mid(pesel,3,2)='05' ;
```

### 3.8. Funkcje łańcuchowe – znak &

- Łączenie pól i tekstów odbywa się znakiem &

```
SELECT nazwiskoPacjenta & ' ' &  imiePacjenta AS kto, pesel
FROM pacjenci WHERE mid(pesel,3,2)='05';
```

### 3.9. Funkcje agregujące

- MIN() – oblicza minimum ze wszystkich wartości w polu
- MAX() – oblicza maksimum ze wszystkich wartości w polu
- SUM() – sumowanie wszystkich wartości w polu
- COUNT() – zlicza i pokazuje ogólną liczbę wierszy
- AVG() – obliczanie średniej arytmetycznej.

AS - możemy nadać etykietę dla wyświetlanej w wyniku kolumny (w Oraclu po spacji nazwa)

Dokonaj statystyk zarobków pracowników w firmie:

```
SELECT FormatNumber(avg(zarobki),2) AS srednia, min(zarobki) AS minimum, max(zarobki) AS
maksimum, sum(zarobki) AS suma, count(*) AS ile FROM pracownicy;
```

### 3.10. Zaokrąglenia w Accessie

Wynik da się zaokrąglić na dwa sposoby:

round
- round(co_zaokrąglamy, ile_miejsc_po_przecinku)
- Algorytm bankiera zastosowany w round polega na tym, że „połówki" zaokrąglane są raz w górę, a raz w dół, zawsze do najbliższej liczby parzystej. Dlatego 1,5 zaokrąglone zostanie do 2, jednak 2,5 zaokrąglona ona zostanie do 2, a nie do 3 !
- Po zastosowaniu round do dalszych obliczeń brana jest wartość zaokrąglona.

FormatNumber
- FormatNumber(co_zaokrąglamy, ile_miejsc_po_przecinku)
- Algorytm wyświetla dla liczby 5 wynik, który jest zaokrąglany w górę
- warto pamiętać o tym, że FormatNumber tylko zmienia widok na ekranie, ale do ewentualnych dalszych obliczeń pobierana jest liczba rzeczywista, bez zaokrągleń.

Sprawdź zaokrąglenia:
Obie funkcje dają takie same wyniki:
SELECT Round(1.5,0) AS round, FormatNumber(1.5,0) AS FormatNumber;
lub różnica w wynikach:
SELECT Round(2.5,0) AS round, FormatNumber(2.5,0) AS FormatNumber;

Podsumowując: Używajmy FormatNumber

### 3.11. Fraza Group by – grupuj według

- łączy ze sobą podobne wiersze dając pojedynczy wiersz wynikowy dla każdej z grup
- bez funkcji agregujących podobne do DISTINCT w Select

Ile było wizyt w danych dniach:
SELECT dataWizyty, count(*) as ile FROM wizyty GROUP BY dataWizyty ;

### 3.12. Fraza Having – porządkuj w grupach

- ustanawia warunki dla frazy GROUP BY
- tu piszemy warunki, które nie dają się zapisać w WHERE

Ile było wizyt w danych dniach, przy czym liczba wizyt nie może być mniejsza niż 10 danego dnia:
SELECT dataWizyty, count(*) AS ile FROM wizyty
GROUP BY dataWizyty HAVING count(*)>=10;

### 3.13. Wyrażenia arytmetyczne

- Dodawanie to znak: +
- Odejmowanie to znak: -
- Mnożenie to znak: *
- Dzielenie to znak: /

Jaka jest różnica między najlepiej i najsłabiej zarabiajacym pracownikiem w firmie:
SELECT max(zarobki)-min(zarobki) as roznica FROM pracownicy;

## 4. WIELOTABLICOWE INSTRUKCJE SELECT

- można wybierać dane z kilku tablic
- tablice muszą być powiązane w klauzuli WHERE (patrz relacje)

### 4.1. Wyświetlić pracowników z pełną nazwą specjalności(stanowisko)

SELECT pracownicy.id, imie, nazwisko, stanowisko, zarobki, czyPremia
FROM pracownicy, stanowisko
WHERE Pracownicy.specjalnosc=stanowisko.id;

UWAGA! Jeśli dane pole nie jest jednoznacznie identyfikowalne w kilku tabelach należy uściślić, z której tabeli brane są dane tzn. tabela.pole (tu pracownicy.id, gdyż pole id jest w obu tabelach)

### 4.2. Wyświetl pełne dane (jaki pacjent, jaki lekarz, data) dotyczące wizyt pacjentów u lekarzy

SELECT Pacjenci.idPacjenta, nazwiskoPacjenta, imiePacjenta, Pracownicy.id, imie, nazwisko, dataWizyty FROM pacjenci, wizyty, pracownicy
WHERE pacjenci.idPacjenta=wizyty.idPacjenta AND wizyty.idLekarza=Pracownicy.id;

UWAGA! Jeśli wykorzystujemy więcej niż dwie tabele w WHERE łączymy je warunkami logicznymi (AND). Jeśli do powiązań chcemy dodać warunek obowiązują te same zasady.

### 4.3. Wyświetl pełne dane (jaki pacjent, jaki lekarz, data) dotyczące wizyt pacjentów u lekarzy, lecz tylko tych nieubezpieczonych

SELECT Pacjenci.idPacjenta, nazwiskoPacjenta, imiePacjenta, Pracownicy.id, imie, nazwisko, dataWizyty FROM pacjenci, wizyty, pracownicy
WHERE pacjenci.idPacjenta=wizyty.idPacjenta AND wizyty.idLekarza=Pracownicy.id and ubezpieczenie='NIE';

### 4.4. Aliasy – zmień zapytanie z 4.3 aby zastosować aliasy

- definiujemy po FROM stosując słówko AS (tylko w Accessie) po nazwie tablicy
- przydatne przy skomplikowanym zapytaniu lub skomplikowanych nazwach tabel

SELECT p.idPacjenta, nazwiskoPacjenta, imiePacjenta, Pracownicy.id, imie, nazwisko, dataWizyty
FROM pacjenci as p, wizyty as w, pracownicy
WHERE p.idPacjenta=w.idPacjenta AND w.idLekarza=Pracownicy.id and ubezpieczenie='NIE';

### 4.5. Samozłączenia

- można tworzyć różne aliasy dla tej samej tabeli – przydatne w warunkach, gdzie chcemy robić warunki odwołujące się z obu stron równania do tej samej tabeli

<u>wybierz pracowników i podaj ich współpracowników, którzy mają o 1000 zł więcej zarobków</u>
SELECT p1.imie, p1.nazwisko, p1.zarobki, p2.imie,p2.nazwisko, p2.zarobki
FROM pracownicy as p1, pracownicy as p2
WHERE p1.zarobki+1000<p2.zarobki;

Ile osób w firmie zarabia więcej od każdego z pracowników

```
SELECT p1.imie, p1.nazwisko, count(*) as ile
FROM pracownicy AS p1, pracownicy AS p2
WHERE p1.zarobki<p2.zarobki
GROUPBY p1.imie,p1.nazwisko;
```

## 5. FUNKCJE DATY

- daty zapisują więcej informacji niż to jest widoczne
- data – druga_data = liczbę oznaczającą czas pomiędzy tymi datami.
- Przydatne funkcje dla daty i czasu
- now() – wyświetlenie daty i czasu jednocześnie
- date() – wyświetlenie daty
- time() – wyświetlenie godziny

Funkcja DateValue

- Access wymaga, żeby każda data, bez względu na językową wersję Accessa i ustawienia systemu, zawsze była wpisana w formacie Amerykańskim i ujęta w znaki # np. #5/16/96#
- funkcja DateValue, która jako argument przyjmuje datę zapisaną w formacie 'lokalnym' a zwraca odpowiednią datę „uniwersalną” np. DateValue('98-12-20')

Funkcja DateDiff

- Funkcja **DateDiff** zwraca różnicę dat wybranych przedziałów czasowych określonych typów
- Trzy parametry: sposób obliczeń(patrz tabelki poniżej), data, data obecna)

| Typ dla wyliczenia różnicy | Opis |
|---|---|
| Yyyy | Rok |
| Q | Kwartał |
| M | Miesiąc |
| Y | Dzień_roku |
| D | Dzień |

| Typ dla wyliczenia różnicy | Opis |
|---|---|
| W | Dzień_tygodnia |
| Ww | Tydzień |
| H | Godzina |
| N | Minuta |
| S | Sekunda |

### 5.1. Sprawdź działanie funkcji czasowych

```
SELECT now() AS funkcja_now, date() AS funkcja_date, time() AS funkcja_time;
```

### 5.2. Ile dni przeżył dany pacjent

```
SELECT nazwiskoPacjenta, imiePacjenta, date()-dataUrodzenia AS dni
FROM pacjenci;
```

| nazwiskoPacjenta | imiePacjenta | dni |
|---|---|---|
| Augustowska | Justyna | 12374 |
| Sosnowiecka | Dorota | 14031 |

### 5.3. Liczenie wieku dla danych pacjentów rocznikiem (niedokładne)

```
SELECT nazwiskoPacjenta, imiePacjenta, DateDiff("yyyy",dataUrodzenia,date()) AS wiek
FROM pacjenci;
```

| nazwiskoPacjenta | imiePacjenta | wiek |
|---|---|---|
| Augustowska | Justyna | 32 |
| Sosnowiecka | Dorota | 36 |

### 5.4. Liczenie co do dnia (dokładne)

SELECT nazwiskoPacjenta, imiePacjenta,DateDiff("**d**",dataUrodzenia,date())/365.25 AS wiek FROM pacjenci;

| nazwiskoPacjenta | imiePacjenta | Wiek |
|---|---|---|
| Augustowska | Justyna | 31,8795345653661875427789 |
| Sosnowiecka | Dorota | 36,4161533196440793976728 |

SELECT [numer], int(DateDiff("d",[urodziny],date())/365.25) AS wiek FROM lata;

| nazwiskoPacjenta | imiePacjenta | wiek |
|---|---|---|
| Augustowska | Justyna | 31 |
| Sosnowiecka | Dorota | 36 |

### 5.5. Odwołanie się do konkretnej daty

Ile było wizyt w 2012 roku po 16 maja 2012 r.:

SELECT count(*) AS ile FROM wizyty WHERE dataWizyty>#5/16/12#;
lub
SELECT count(*) AS ile FROM wizyty WHERE dataWizyty>DateValue('2012-05-16')

### 5.6. Funkcje daty - dodatkowe

- Day(data) – wyznaczenie z daty dnia
- Month(data) – wyznaczenie z daty miesiąca
- Year(data) – wyznaczenie z daty roku
- Weekday(data) – wyznaczanie dnia tygodnia z daty (wyświetlane jako liczby od 1 do 7)
- DatePart("typ", Data) – odwołanie się do konkretnej części w roku wg. parametrów z tabelek (tych samych, z których korzystaliśmy przy DateDiff)

Ile wizyt było w 2012 roku:
SELECT count(*) as ile FROM wizyty WHERE year(dataWizyty)=2012;

Jak rozkładają się ilości wizyt w kwartałach:
SELECT DatePart("q",dataWizyty) as kwartał, count(*) as ile FROM wizyty
GROUP BY DatePart("q",dataWizyty);

## 6. PARAMETR

- Parametr organizujemy poprzez podanie nazwy niezdefiniowanej w tabeli – pojawi się monit o tekscie podanym jako nieznany
- Podawany parametr jest tekstem, więc przy parametrze liczbowym należy zastosować funkcję int(n) konwertującą tekst na liczbę

Wyszukaj pracowników, którzy zarabiają mniej od podanego parametru
SELECT imie, nazwisko, zarobki FROM pracownicy
WHERE zarobki<int([Podaj zarobki]);

## 7. PODKWERENDY (PODZAPYTANIA), WIRTUALNE TABELE

- Podkwerenda to instrukcja SELECT (w nawiasach) lub [w nawiasach zakończonych kropką]. zagnieżdżona wewnątrz innej kwerendy lub podkwerendy.
- W podkwerendzie instrukcja SELECT służy do generowania zbioru konkretnych wartości, które zostaną poddane ocenie w wyrażeniu klauzuli WHERE lub HAVING.
- W celu utworzenia podkwerendy można użyć kilku rodzajów składni:

- *porównanie* [ANY | ALL | SOME] (SELECT...) np. where punkty > any (select ...)
- *wyrażenie* [NOT] IN (SELECT...) np. where punkty not in (select ...)
- [NOT] EXISTS (SELECT...)

| Składnia | Opis |
|---|---|
| ANY<br>SOME | pobierają rekordy spełniające kryteria porównania z dowolnymi rekordami pobranymi przy użyciu podkwerendy |
| ALL | umożliwia pobranie w głównej kwerendzie tylko tych rekordów, które spełniają kryteria porównania ze wszystkimi rekordami pobranymi za pośrednictwem podkwerendy (bardziej restrykcyjne) |
| IN | spowoduje pobieranie przy użyciu głównej kwerendy tylko tych rekordów, dla których pewien rekord w podkwerendzie zawiera taką samą wartość |
| NOT IN | W głównej kwerendzie można uwzględnić tylko te rekordy, dla których żaden rekord w podkwerendzie nie zawiera identycznej wartości |
| EXISTS | (z opcjonalnym słowem zarezerwowanym NOT) użyte w porównaniach prawda/fałsz pozwala określić, czy kwerenda będzie zwracała jakiekolwiek rekordy |

### 7.1. Suma – oblicz ile mamy wśród pacjentów mężczyzn, a ile kobiet

- w celu połączenia dwóch zapytań w jedno
- UNION – wybiera wszystkie wiersze z obu zapytań, ale usuwa powtarzające się wiersze
- Oba zapytania muszą zwracać tyle samo kolumn i muszą one być takiego samego typu
- Jeśli jedna tablica mimo zgodności kolumn będzie pozwalała na wartości puste a druga nie to zapytanie nie będzie działać !!!
- Jeśli chcemy ujrzeć powtarzające się wyniki stosujemy UNION ALL

Ile wśród pacjentów mamy kobiet i mężczyzn:

```
SELECT count(*) as ile FROM pacjenci WHERE imiePacjenta like '*a'
UNION
SELECT count(*) as ile FROM pacjenci WHERE imiePacjenta not like '*a';
```

Ile wśród pacjentów mamy kobiet i mężczyzn – żeby było widać dobrze wyniki

- Możemy w zapytania we frazie SELECT wyświetlić teksty, które chcemy zobaczyć
- Teksty zawarte są jak zwykle w apostrofy
- Ponieważ tekst, który chcemy wyświetlić nie zawiera się w jakiejkolwiek tabeli, należy go opatrzyć etykietą za pomocą frazy AS

```
SELECT 'kobiety' as plec, count(*) as ile FROM pacjenci WHERE imiePacjenta like '*a'
UNION
SELECT 'faceci' as plec, count(*) as ile FROM pacjenci WHERE imiePacjenta not like '*a';
```

!!! jeśli w UNION w kolejnych, połączonych zapytaniach będziemy chcieli użyć polecenia ORDER BY wystąpią błędy. Objawiają się brakiem sortowania w zapytaniach (poza pierwszym w UNION) – nie mamy żadnych komunikatów o błędach! – rozwiązania szukaj w punkcie 7.4

### 7.2. Różnica

- zwraca rekordy które występują w pierwszej tablicy, a nie ma nich w drugiej
- uzyskujemy stosując NOT IN z podzapytaniem

Wyświetl pacjentów, którzy jeszcze nigdy nie byli na wizycie w przychodni
SELECT idPacjenta
FROM pacjenci
WHERE idPacjenta not in **(SELECT idPacjenta FROM wizyty)**;

UWAGA! Od wszystkich pacjentów w tabeli odejmij tych co już byli w przychodni. Istotne jest to, by pole z WHERE było te samo (lub umożliwiało powiązanie) co z podzapytania w SELECT (patrz podkreślenie)

### 7.3. Iloczyn

- daje wynik w postaci rekordów będących częścią wspólna w obu tablicach
- uzyskujemy stosując IN lub EXISTS z podzapytaniem
- należy pamiętać o uporządkowaniu wyników frazą ORDER BY

Wyświetl pacjentów, którzy skorzystali z usług przychodni
SELECT idPacjenta
FROM pacjenci
WHERE idPacjenta in (SELECT idPacjenta FROM wizyty);

### 7.4. UNION a ORDER BY

- Gdy stosujemy zapytanie Union, order by może być użyte tylko w pierwszym z tych zapytań
- Zapytanie nie wyrzuci żadnych błędów, tylko poda błędne wyniki – szczególnie widoczne w połączeniu ORDER by z TOP
- w przypadku potrzeby stosowania kilku, obchodzimy problem na dwa sposoby.

Wyświetl najlepiej zarabiającego pediatrę(2) i laryngologa(3):

#### 7.4.1. wirtualna tabelka

SELECT top 1 imie, nazwisko, specjalnosc, zarobki FROM pracownicy
WHERE specjalnosc=2 ORDER BY zarobki desc
UNION
SELECT imie, nazwisko, specjalnosc, zarobki
FROM **(SELECT top 1 imie, nazwisko, specjalnosc, zarobki FROM pracownicy
WHERE specjalnosc=3 ORDER BY zarobki desc)**

#### 7.4.2. Nawiasy okalające kolejne zapytania w UNION

SELECT top 1 imie, nazwisko, specjalnosc, zarobki FROM pracownicy
WHERE specjalnosc=2 ORDER BY zarobki desc
UNION
**(**SELECT top 1 imie, nazwisko, specjalnosc, zarobki FROM pracownicy
WHERE specjalnosc=3 ORDER BY zarobki desc**)**;

## 8. UMIEJSCOWIENIE PODZAPYTAŃ SELECT

### 8.1. po SELECT

- rzadko stosowane - może być przydatne przy tworzeniu wszelkiego rodzaju zestawień
- SELECT (zapytanie)

Wyświetl liczbę kobiet i mężczyzn obok siebie w poziomie:

```
SELECT DISTINCT
(SELECT count(*) FROM pacjenci WHERE imiePacjenta like '*a') AS kobiety,
(SELECT count(*) FROM pacjenci WHERE imiePacjenta not like '*a') AS faceci
FROM pacjenci;
```

UWAGA! Użycie w zapytaniu pól bazy powoduje konieczność zastosowania minimalnie klauzul SELECT ... FROM ... . Ponieważ nie mamy jak ograniczyć w WHERE wyników do jednego rekordu wyniku stosujemy DISTINCT

### 8.2. w warunku WHERE

- kiedy warunek umożliwia dokładniejsze precyzowanie wartości np. where dana=(select...) lub dana in(select...)
- przykłady w 7.2 i 7.3

### 8.3. w HAVING

Wyświetl stanowiska, na których średnia zarobków jest większa, niż średnia zarobków w firmie

```
SELECT stanowisko, avg(zarobki) FROM pracownicy, stanowisko
WHERE pracownicy.specjalnosc=stanowisko.id GROUP BY stanowisko
HAVING avg(zarobki)>(select avg(zarobki) from pracownicy)
```

### 8.4. po FROM

tymczasowa (wirtualna) tabela, z której uzyskamy dane

Na podstawie wcześniejszego zapytania (8.3) wyświetl ile jest takich specjalności

```
SELECT count(*) AS ile
FROM (SELECT stanowisko, avg(zarobki)
FROM pracownicy, stanowisko
WHERE pracownicy.specjalnosc=stanowisko.id
GROUP BY stanowisko
HAVING avg(zarobki)>(select avg(zarobki) from pracownicy));
```

## 9. iif

Jeśli pacjent jest niepełnoletni wyświetl dziecko, w przeciwnym wypadku wyświetl dorosły

```
SELECT imiePacjenta, nazwiskoPacjenta, dataUrodzenia,
IIf(int(DateDiff("d",dataUrodzenia,date())/365.25)<18,'dziecko','dorosły')
FROM pacjenci;
```

## 10. POZOSTAŁE KWERENDY

- należy pamiętać, że operacje na bazie to nie tylko wyświetlanie jakiś informacji
- każdą z uprzednio nauczonych czynności (np. podzapytania, działania matematyczne, parametry itp. itd.) można wykorzystać przy pisaniu wymienionych w tym rozdziale zapytań

### 10.1. Instrukcja INSERT - Wstawianie wierszy

- pozwala utworzyć nowy wiersz używając podanych wartości dla Np.
  insert into tabela values(wartość,'wartość');
  lub insert into tabela(pole1) values(wartość);
- w nawiasie musimy zadeklarować wartości dla pól (teksty w apostrofach) odpowiednio dla typu danych w tabeli
- jeśli zamiast konkretnej wartości wpiszemy NULL umożliwiamy sobie pominięcie podania prawidłowych danych na etapie dodawania nowego rekordu

Wpisz nowo zatrudnioną pielęgniarkę(specjalność: 12) wiedząc, że jej idendtyfikator to o6, pensja to 1900 zł. Nazywa się ona Anna Nowak
```
INSERT INTO pracownicy VALUES ('o5', 'Anna', 'Nowak', 12, 1900, NULL);
```

UWAGA! Ostatnim polem w tabeli pracownicy jest pole TAK/NIE, więc możemy zamiast NULL podać wartość 0 lub -1

Wpisz nową wizytę w tabeli wizyta
```
INSERT INTO wizyty ( idPacjenta, idLekarza, dataWizyty )
VALUES ([Podaj id pacjenta], [Podaj id lekarza], DATE());
```

### 10.2. Instrukcja UPDATE - Aktualizacja wierszy

- do zmiany zawartości jednej lub wielu kolumn w jednym lub wielu wierszach tablicy
  UPDATE tabela SET pole1 = wartość, pole2 = 'wartość' WHERE pole=warunek;
- gdzie: set – miejsce gdzie nastąpi aktualizacja.
- Warunek – wartości dla których pól będzie dokonywana aktualizacja (może występować OR, AND itp)

Ustaw, aby żaden z pracowników nie miał premii
```
UPDATE pracownicy SET czyPremia = no;
```

Premia dla 5 najbardziej obleganych lekarzy
```
UPDATE pracownicy SET czyPremia = yes WHERE id in
(SELECT top 5 pracownicy.id FROM pracownicy, wizyty
WHERE pracownicy.id=Wizyty.idLekarza
GROUP BY pracownicy.id
ORDER BY Count(*) DESC;);
```

Premia dla pracowniów mających więcej wizyt od liczby podanej parametrem(dla 70 - 11 lekarzy)
```
UPDATE pracownicy SET czyPremia = yes WHERE id in
(SELECT idlekarza FROM pracownicy, wizyty
WHERE pracownicy.id=wizyty.idlekarza
GROUP BY idlekarza
HAVING count(*)>=[ile wizyt aby premia była uznana]);
```

### 10.3. Instrukcja DELETE - Usuwanie wierszy

- usuwa dowolny wiersz lub kombinację wierszy z tablicy
- DELETE może zawierać frazę WHERE, której postać jest taka sama jak dla instrukcji SELECT

Usuń pracownika z bazy(np. o5)

```
DELETE * FROM Pracownicy
WHERE id=[Podaj identyfikator pracownika];
```

### 10.4. Instrukcja CREATE TABLE

- Tworzy nową tabelę w naszej bazie
- Polom należy nadać typy pól tzn.:

- counter – autonumerowanie
- char – tekst
- datetime – data
- integer – liczba

Stwórz tabelę wizytyRok

```
CREATE TABLE wizytyRok
(
id counter PRIMARY KEY,
idPacjenta CHAR,
idLekarza CHAR,
dataWizyty DATETIME
);
```

Wypełnij ją polami z wybranego roku

```
INSERT INTO WizytyRok ( idPacjenta, idLekarza, dataWizyty )
SELECT idPacjenta, idLekarza, dataWizyty
FROM Wizyty
WHERE year(dataWizyty)=[Podaj rok];
```

Zmiksuj dwa polecenia wyżej

```
SELECT idPacjenta, idLekarza, dataWizyty INTO wizytyRok
FROM wizyty
WHERE year(dataWizyty)=[Podaj rok];
```

UWAGA! W tym zapytaniu trzeba sobie samemu w Widoku projektu dodać pole id i klucz główny

## 11. KWERNDY DODATKOWE

### 11.1. Odwołanie się do pola logicznego

Wyświetl pracowników, którzy zasłużyli na premię:

```
SELECT * FROM pracownicy WHERE czyPremia=yes;
```

### 11.2. Odwołanie przez tekst

Wyświetl pracowników, których nazwiska są z przedziału literowego A do D

SELECT * FROM pracownicy WHERE nazwisko >= 'A' and nazwisko < 'E';

### 11.3. Odwołanie się do pustego pola

- abyśmy mogli zaobserwować sytuację użyjemy kwerendy dodającej pracownika bez nazwiska
  INSERT INTO pracownicy VALUES ('o5', 'Anna', NULL, 12, 1900, NULL);
- znajdujemy osobę bez nazwiska stosując is null
  SELECT * FROM pracownicy WHERE nazwisko is null;
- kasujemy niepotrzebny rekord
  DELETE * FROM pracownicy WHERE nazwisko is null;
- sens tego zapytania w tej bazie jest żaden, ale jeśli wyobrazimy sobie hurtownię z polem wysyłka, gdzie wpisujemy np. daty zrealizowania zamówienia - wtedy polecenie is null może nam wyświetlić zamówienia do zrealizowania

## 12. TROCHĘ FUNKCJI DODATKOWYCH

Funkcje pracują pod Oraclem. Czy działają w Access – sprawdź sam.

- sqr(liczba) – oblicza pierwiastek z liczby
- sin(liczba), cos(liczba), tan(liczba) – funkcje trygonometryczne z liczby
- log(liczba) – logarytm naturalny z liczby
- abs(liczba) – wartość bezwzględna z liczby
- chr(liczba) – zamienia wartosc liczby na odpowiadajacy znak ASCII
- asc(‘litera’) – zwraca kod ASCII litery
- UCase(pole) – zamienia litery łańcucha zawartego w pole na wielkie
- LCase(pole) – zamienia litery łańcucha zawartego w pole na małe
- Len(pole) – zwraca długość łancucha tekstu zawartego w polu
- InStr(start; lancuch1; lancuch2; wielkość_liter) – wyznacza pozycje podłańcucha lancuch2 w lancuch1 począwszy od znaku start; parametr wielkość_liter ustala, czy uwzględniania jest wielkość liter (0 - tak, 1 – nie, 2 – zgodnie z ustawieniami systemowymi bazy danych)
- LTrim(lancuch), RTrim(lancuch), Trim(lancuch) – usuwa spacje wiodące, spacje z prawej strony lub obustronne

## 13. REAKCJE NA STANDARDOWE KOMUNIKATY W ACCESIE

### 13.1. Wyskakujące okienko parametru, gdy go nie używamy

PRZYPADEK 1:
- Umieściliśmy wyraz niezrozumiały dla bazy np. popełniliśmy błąd literówki.
- Popraw wyraz na prawidłowy zawarty w naszej bazie

PRZYPADEK 2:
- Możliwe, że we frazie FROM nie masz wpisanej tabeli, z której chcesz wyświetlić pole

### 13.2. Próba wykonania kwerendy, która nie zawiera podanego wyrażenia ‘tu_jest_podane_pole’ jako elementu funkcji agregującej

- Wykonujesz zapytanie, które zawiera w sobie funkcje agregujące.
- Nie wykonałeś poprawnego grupowania pól dla tych funkcji.
- Dodaj wskazane pole do frazy GROUP BY w zapytaniu

### 13.3. Nie można umieścić funkcji agregującej w klauzuli where

- Warunki dla funkcji agregujących nie mogą być zawarte we frazie WHERE
- Zrób ten warunek we frazie HAVING

### 13.4. W wynikach kwerendy pojawia się dużo nadmiarowych wyników, a one same w części wyglądają jak powielone rekordy

- Brak powiązania międzytablicowego we frazie WHERE

## 14. INFORMACJE O AUTORZE

Nazywam się Sebastian Ostrowski. Jestem nauczycielem informatyki w Liceum Ogólnokształcącym im. Marsz. St. Małachowskiego w Płocku. Baza i skrypt zostały przeze mnie stworzone na potrzeby koła informatycznego oraz z myślą o maturzystach z Informatyki. Wszelkie uwagi, propozycje zmian lub chęci wykorzystania skryptu w celach innych niż samokształcenie proszę zgłaszać na maila: szarpman@poczta.onet.pl

Pozdrawiam i życzę przyjemnej nauki.