Podstawowe pojęcia związane z bazami danych:

Baza danych jest zbiorem danych oraz narzędzi systemu DBMS (*Database Management System* — System Zarządzania Bazą Danych, SZBD) przeznaczonego do zarządzania bazą danych oraz gromadzenia, przekształcania i wyszukiwania danych.

Baza danych to zbiór danych, który dotyczy rzeczywistości – a konkretnie określonego jej fragmentu, który reprezentuje. Fragment ten określamy mianem obszaru analizy.

Baza danych ma takie cechy charakterystyczne, jak:

- **Trwałość danych** – oznacza możliwość przechowywania danych w pamięci masowej (trwałej) komputera. Dane tymczasowe mogą być przechowywane w pamięci komputera i tracone po jego wyłączeniu.
- **Niezależność danych** – pozwala osiągnąć większą elastyczność, ponieważ programy wymieniające informacje z bazą danych są niezależne od przechowywania danych na dysku i szczegółów reprezentacji danych na dysku. Niezależność dotyczy również posługiwania się danymi. Użytkownicy są zabezpieczeni przed logicznymi zmianami (program obsługujący bazę danych jest zabezpieczony przed modyfikacją struktury tabel bazy danych). DBMS – gwarantujący niezależność fizyczną – przejmuje na siebie zadanie określenia, w jakim formacie i jak dane będą przechowywane na dysku.
- **Ochrona (bezpieczeństwo) danych** – baza danych oferuje mechanizmy kontroli dostępu do danych w sposób umożliwiający użytkowanie danych wyłącznie przez uprawnionych do tego użytkowników.
- **Integralność danych** – zgodność z rzeczywistością. dane w bazie danych są odwzorowaniem rzeczywistości. [Omówione później]
- **Współdzielenie danych** – poszczególne fragmenty danych mogą być używane przez kilku użytkowników jednocześnie (dostęp współbieżny)
- **Abstrakcja danych** – dane opisują tylko istotne aspekty obiektów świata rzeczywistego
- **Integracja danych** gwarantująca, że dane i związki między nimi nie powtarzają się jeśli nie jest to konieczne ale wszelkie zmiany w obrębie bazy nie powodują wieloznaczności

System zarządzania bazą danych SZBD (DBMS – *DataBase Management System*) obsługuje użytkowników bazy danych, umożliwiając im eksploatację oraz tworzenie baz danych. By stworzyć i zaprojektować bazę danych, należy ją zdefiniować, a do tego konieczne jest określenie (zdefiniowanie) typów przechowywanych w niej danych. Istotną rolę odgrywa również wyznaczenie użytkowników oraz ich praw dostępu.

SZBD pełni funkcje, które określane są mianem właściwości baz danych. Zaliczamy do nich:

- tworzenie struktur baz danych,
- wykonywanie operacji CRUD (Create, Read, Update, Delete),
- obsługa zapytań (selekcjonowanie danych),
- generowanie raportów i zestawień,
- administracja bazą danych.

Tworzenie struktur baz danych

Aby utworzyć strukturę bazy danych, należy posłużyć się wcześniej sporządzonym projektem. Struktura to szkielet bazy danych, przeniesienie koncepcji tabel, powiązań na obszar systemu zarządzania bazą danych.

Kolejną właściwością bazy danych jest przeprowadzanie operacji CRUD (zapisu, odczytu, aktualizacji i usuwania). Może zajść potrzeba modyfikowania tabel, widoków oraz aktualizacji danych przechowywanych w tabelach. Baza danych powinna być tak zaprojektowana, by wykonywanie aktualizacji na danych, usuwanie danych czy wprowadzanie nowych informacji do bazy danych nie spowodowało utraty spójności. **Spójność** bazy danych to **poprawność** umieszczonych w niej informacji.

Baza danych powinna mieć mechanizmy umożliwiające uzyskanie **szybkiego dostępu** do danych i ich **selekcjonowanie**. W relacyjnych bazach danych do uzyskiwania dostępu do danych służą zapytania. **Zapytania** to instrukcje napisane przeważnie w języku SQL.

Oprócz uzyskania dostępu do informacji i danych, ich **sortowania**, **selekcjonowania** i przeszukiwania baza danych powinna oferować mechanizmy umożliwiające **drukowanie** wykazów czy zapisywanie ich poza bazą danych. Funkcje takie spełniają **raporty** i zestawienia, które mogą być generowane z baz danych.

Baza danych powinna umożliwiać administrację swoimi zasobami. Administracja może mieć charakter nie tylko projektowania i implementowania, lecz także optymalizacji i dostosowywania do potrzeb użytkowników.

Modele danych

- Definicja modelu danych
- Modele danych: model jednorodny, hierarchiczny model danych, sieciowy model danych, model relacyjno-obiektowy, obiektowy model danych, relacyjny model baz danych
- Rozproszona baza danych

Model danych to zintegrowany zbiór zasad opisujących dane, relacje, powiązania (stosunki) pomiędzy danymi, dozwolone operacje i ograniczenia nakładane na dane i operacje.

Model danych jest próbą reprezentacji świata realnego i występujących w nim obiektów, zdarzeń oraz związków zachodzących między nimi. Można go opisać jako konstrukcję składającą się z trzech komponentów:

- części strukturalnej – składającej się z reguł określających budowę bazy danych;
- części manipulacyjnej – określającej, które operacje (transakcje) aktualizacji, pobierania i zmiany struktury można wykonywać na danych;
- części zawierającej reguły integralności – gwarantującej stabilność działania systemu.

Model jednorodny – to model, w którym wszystkie dane są umieszczone w **jednej tabeli, jednym arkuszu (***pojedynczy arkusz – tabela z danymi może przechowywać duplikaty. Jeśli będzie to np. wypis sprzedanych towarów, wówczas dany towar, który został sprzedany kilkakrotnie, będzie duplikowany w kolejnych wierszach.**)*** (kostce analitycznej). Przykładem takiego modelu jest książka telefoniczna. Cechuje go łatwość i szybkość odczytywania danych. Jego wadą jest duża liczba duplikatów i tym samym zwiększyć zużycie zasobów komputera. Dane w modelu jednorodnym nie zawsze będą łatwe do odnalezienia. Gdyby jednorodny model danych prezentować na przykładzie książki telefonicznej, to łatwe byłoby odnalezienie numeru telefonu na podstawie imienia i nazwiska, jednak wyszukanie imienia i nazwiska na podstawie numeru telefonu stwarzałoby problem. Książka telefoniczna zawiera dane ułożone alfabetycznie wg kolejności nazwisk. Dlatego nie jest zoptymalizowana pod kątem przeprowadzania wyszukiwania, w którym dysponujemy numerem, a potrzebujemy odszukać imię i nazwisko. W modelu jednorodnym dane mogą być duplikowane. Dzieje się tak właśnie ze względu na strukturę tego modelu. Na przykład, gdyby w Warszawie mieszkało 50 osób o nazwisku Nowak i każda z nich miała telefon, wówczas w jednorodnym modelu danych otrzymalibyśmy pięćdziesiąt duplikujących się nazwisk w kolejnych wierszach, różniących się adresami i numerami telefonów, przy czym nazwisko byłoby wielokrotnie powtórzone.

Hierarchiczny model danych

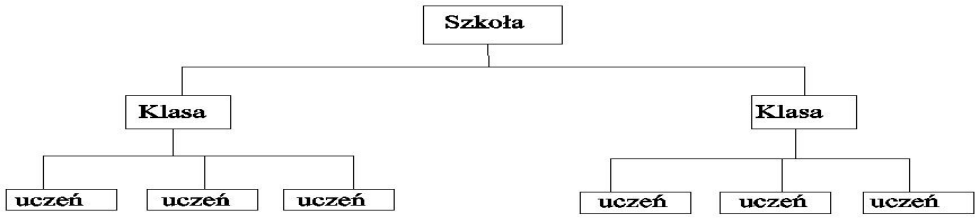
Początki sięgają 1960 r., gdy rozpoczęto prace nad projektami **IDS** (*Integrated Data Store*) oraz **MIS** (*Management Information System*). MIS rozwijany był przez IBM w ramach projektu kosmicznego Apollo. Hierarchiczny model bazy danych pod względem modelu przypomina strukturę odwróconego drzewa:

- jeden korzeń (tabela nadrzędna)

- synowie (tabele podrzędne).

Ojciec może mieć wiele dzieci, ale każde dziecko ma tylko jednego ojca.

Każdy rekord (z wyjątkiem głównego, który jest na szczycie) powiązany jest z jednym rekordem nadrzędnym. Model hierarchiczny opiera się zatem na dwóch strukturach danych – typach rekordów i związkach nadrzędny-podrzędny. Powiązanie nadrzędny-podrzędny to związek „jeden do wielu" pomiędzy dwoma typami rekordów. Model ten różni się od relacyjnego, ponieważ w modelu relacyjnym powiązania zachodzą przez klucze obce, a w hierarchicznym przez związek nadrzędny-podrzędny. W hierarchicznym modelu danych nie można wstawić rekordu podrzędnego, dopóki nie zostanie powiązany z nadrzędnym. Usunięcie rekordu nadrzędnego powoduje automatycznie usunięcie wszystkich rekordów podrzędnych.



BAZA HIERARCHICZNA

Sieciowy model

Został ustandaryzowany w 1969 r. przez **CODASYL** (*Conference on Data Systems Languages*). Jego twórcą jest Charles Bachman i mimo że nie znalazł szerszego zastosowania, to przyczynił się do powstania relacyjnego modelu baz danych. Sieciowy model przyjmuje, podobnie jak hierarchiczny, strukturę przypominającą odwrócone drzewo z tą różnicą, że gałęzie jednego drzewa mogą być wspólne z gałęziami innych drzew.

Sieciowy model oparty jest zatem na dwóch strukturach danych:

- **typ kolekcji** jest opisem związku „jeden do wielu" pomiędzy dwoma typami rekordów,
- **typ rekordów** ma swój odpowiednik w modelu hierarchicznym, jednak pola są w stanie przechowywać złożone wartości, które mogą się powtarzać

Model relacyjno-obiektowy

Jest mieszanym modelem bazodanowym, pozwala on w relacyjnych tabelach tworzyć kolumny, w których przechowywane są dane typu obiektowego, pozwala na definiowanie zmiennych oraz metod, które będą wykonywane na danych wprowadzanych do obiektu.

Obiektowy model danych

Opiera się na koncepcji obiektów (podobnie jak w projektowaniu obiektowym – obiekt jest odwzorowaniem rzeczywistości lub abstrakcji). Odwołania do określonego obiektu w tym modelu bazy danych są wykonywane za pomocą interfejsu, dzięki któremu są zachowane integralność i bezpieczeństwo danych. Obiektowe bazy danych korzystają z obiektowego języka zapytań **OQL** (*Object Query Language*). Każdy obiekt ma zaprojektowany interfejs określający metody dostępu do niego. Obecnie coraz popularniejszy staje się standard **JDO** (*Java Data Object*) stworzony przez firmę Sun Microsystems. W ramach tego standardu rozwinął się obiektowy język zapytań **JDOQL** (*Java Data Object Query Language*). Dość powszechnym niekomercyjnym i relacyjnym systemem

baz danych mającym obiektowe rozszerzenie jest **PostgreSQL**.

Relacyjny model danych

Opracował w latach osiemdziesiątych XX w. Edgar Frank Codd. Larry Ellison, współfundator korporacji Oracle, projekt systemu zarządzania bazami danych oparł na założeniu Codda. Oprócz Oracle rozwijały się w tym czasie takie bazy danych, jak Sybase lub Informix, używające języka SQL. W 1985 r. Codd przedstawił 12 zasad opisujących model relacyjny baz danych. Dane przechowuje się w tabelach, nazywanych **relacjami**, składających się z **wierszy (krotek)** i **kolumn (atrybutów)**.

Postulaty Codda

System musi być kwalifikowany jako relacyjny, jako baza danych i jako system zarządzania:

1. **Postulat informacyjny** – dane są reprezentowane jedynie poprzez wartości atrybutów wierszach tabel.
2. **Postulat dostępu** – każda wartość w bazie danych jest dostępna poprzez podanie nazwy tabeli, atrybutu i wartości klucza podstawowego.
3. **Postulat dotyczący wartości NULL** - dostępna jest specjalna wartość NULL dla reprezentacji zarówno wartości nieokreślonej, jak i nieadekwatnej.
4. **Postulat dotyczący katalogu** – wymaga się, aby system obsługiwał wbudowany katalog relacyjny z bieżącym dostępem dla uprawnionych użytkowników.
5. **Postulat języka danych** – system musi dostarczać pełny język przetwarzania danych, który może być używany zarówno w trybie interaktywnym, jak i w obrębie programów aplikacyjnych, obsługuje operacje definiowania danych, operacje manipulowania danymi, ograniczenia związane z bezpieczeństwem i integralnością oraz operacje zarządzania transakcjami.
6. **Postulat modyfikowalności perspektyw** – system musi umożliwiać modyfikowanie perspektyw, o ile jest ono semantycznie realizowalne.
7. **Postulat modyfikowalności danych** – system musi umożliwiać operacje modyfikacji danych, musi obsługiwać operatory INSERT, UPDATE oraz DELETE.
8. **Postulat fizycznej niezależności danych** – zmiany fizycznej reprezentacji danych i organizacji dostępu nie wpływają na aplikacje.
9. **Postulat logicznej niezależności danych –** zmiany wartości w tabelach nie wpływają na aplikacje.
10. **Postulat niezależności więzów spójności** – więzy spójności są definiowane w bazie i nie zależą od aplikacji.
11. **Postulat niezależności dystrybucyjnej** – działanie aplikacji nie zależy od modyfikacji i dystrybucji bazy.
12. **Postulat bezpieczeństwa względem operacji niskiego poziomu** — operacje niskiego poziomu nie mogą naruszać modelu relacyjnego i więzów spójności.

Przy użyciu zasad algebry relacyjnej opracowano również język SQL służący do komunikowania się z większością współczesnych baz danych. Obecnie spotykane bazy danych przystosowane są w większości do pracy wykorzystującej połączenia sieciowe (architektura KLIENT-SERWER).

Integralność danych, spójność danych.

jest to funkcja SZBD, która gwarantuje, że dane nie zostaną usunięte lub zmienione w nieautoryzowany sposób.

Ochrona integralności danych polega również na zapewnieniu, że dane nie ulegną zniekształceniu podczas wykonywania na nich operacji. Spójność danych związana jest z ich dokładnością – dane dokładnie odzwierciedlają modelowaną rzeczywistość. Oznacza ona również ich prawdziwość oraz aktualizowanie, gdy zmienia się rzeczywistość modelowana w bazie danych. Dane muszą być poprawne i zgodne ze schematem bazy danych. W SZBD powinny istnieć mechanizmy, które

pozwolą zabezpieczyć dane przed skutkami awarii zasilania, sprzętu lub oprogramowania. W bazie danych powinny działać mechanizmy, których zadaniem jest zabezpieczenie danych przed następstwami błędów logicznych. Zachowanie spójności danych powinny gwarantować systemy chroniące dane też przed błędami pojawiającymi się w chwilach współbieżnego dostępu do tej samej informacji. Istotną rolę odgrywa również system kontroli danych wejściowych. Proces utrzymania integralności bazy danych obejmuje również kopie zapasowe danych.

Zachowanie poprawności bazy danych opiera się na utrzymaniu poprawności w obrębie semantycznym, encji i referencyjnym.

Integralność semantyczna polega na utrzymaniu ograniczeń nakładanych na dane, min.:

- w określonej kolumnie tabeli muszą znajdować się wyłącznie dane zgodne z typem danych kolumny, np. tylko liczby całkowite;
- w kolumnie nie mogą wystąpić braki wartości – puste miejsca NULL.

Obok integralności semantycznej wyróżniamy także integralność encji.

Integralność encji wprowadza się w trakcie definiowania schematu danych. Zgodnie z regułami integralności encji każda tabela powinna mieć **klucz główny** – kolumnę, w której nie mogą wystąpić wartości NULL oraz w której dane nie mogą się powtarzać.

Integralność referencyjna polega na wprowadzeniu i utrzymaniu powiązań pomiędzy tabelami. Związki te tworzy się przez umieszczenie kolumny pełniącej rolę **klucza głównego tabeli** w innej tabeli, co nadaje kolumnie funkcję klucza obcego. Takie rozwiązanie nazwę związków: klucz **podstawowy – klucz obcy**. Reguły integralności referencyjnej determinują, czy wartości klucza obcego mają odpowiadać wartościom klucza głównego w powiązanej tabeli, czy mogą przyjmować wartości **NULL**.

Podczas omawiania zagadnienia integralności bazy danych stosuje się również podział więzów integralności na statyczne i dynamiczne.

Ograniczenia integralnościowe statyczne odnoszą się do bieżącego stanu bazy danych, np. na kolumnę wiek zostało nałożone takie ograniczenie jak **CHECK** (wiek < 200). Nałożenie tego ograniczenia podczas tworzenia tabeli spowoduje, że w kolumnie wiek wartość nie będzie większa niż 200 w trakcie nakładania ograniczenia i w przyszłości.

Więzy integralności dynamiczne to takie, które przeciwdziałają zmianom, ponieważ związane są z przejściem bazy danych z jednego stanu w drugi. Oznacza to, że odnoszą się do bazy danych w aspekcie temporalnym. Przykładem takich więzów może być wymaganie, aby wiek pracowników nigdy nie malał. Jeśli wiek pracownika po pewnym czasie wzrośnie, to wartość kolumny wiek nie może nigdy się zmniejszyć, ponieważ nikt nie staje się młodszy. Więzy dynamiczne nazywane są również więzami przejść.

SZBD – Systemy Zarządzania Bazą Danych

System Zarządzania Bazą Danych – SZBD (*DBMS – Database Management System*) to oprogramowanie umożliwiające użytkownikom definiowanie, tworzenie i zarządzanie bazą danych oraz kontrolowanie dostępu do niej. Rozszerzając tę definicję, warto wspomnieć, że oprogramowanie to umożliwia dostęp do baz danych nie tylko użytkownikom, lecz także innym programom. Wszystkie powyższe czynności są możliwe dzięki zaimplementowanej obsłudze strukturalnego języka zapytań.

Charakterystyka elementów bazodanowych

Znajomość budowy bazy danych wymaga zwykle fachowego określenia jej elementów. Twórca relacyjnego modelu danych – E.F. Codd – w pracy *Relacyjny model danych dla dużych banków* nie używa terminów tabela, kolumna, wiersz, lecz zamiast nich stosuje pojęcia: **relacja** (zamiast tabela), **atrybu** (zamiast kolumna), **krotka** (zamiast wiersz).

Encja (*enity*) jest wtedy, gdy potrzebujemy określić coś, co reprezentuje obiekt lub grupę obiektów. Pojęcia **encji** używamy, aby określić nie tylko obiekty fizyczne, lecz także niematerialne. Przykładami encji mogą być takie obiekty jak: **OSOBA,** cechami mogą być wzrost, numer buta, waga. Cechy te nazywane są atrybutami encji.

Krotka (*tuple*) może być zdefiniowana następująco: jeśli tabela spełnia wymogi relacji (jest relacją), a jej kolumny są **atrybutami,** to krotka jest wierszem (rekordem). Krotka przechowuje stałe wartości o różnych typach danych, których to typów nie można zmodyfikować w kolejnej krotce. Dlatego typy, np. tytuł, ISBN, dla następnych krotek jednej tabeli będą stałe, a ich zawartości będą się różnić. Odczyt krotki wymaga podania jej tabeli będą stałe jej indeksu (w naszym przykładzie niepowtarzalnego numeru ISBN).

Atrybut definiowany jest jako **kolumna relacji mająca identyfikator** (nazwę). W relacyjnym modelu baz danych, gdy dwuwymiarową tabelę nazwiemy relacją (gdy spełnia warunki relacji), wówczas **posiadające nazwę** kolumny tej tabeli nazywamy **atrybutami.**

Ważną cech tabeli – relacji jest to, że kolejność atrybutów nie powinna mieć znaczenia.

Przykład

Przyjmując książkę w bibliotece jako **encję,** możemy wskazać jej następujące **atrybuty:**

- ISBN,
- tytuł,
- autor,
- wydawnictwo,
- liczba książek,
- rok wydania.

Poniższy przykład pokazuje, że dla **encji *Książka*** mamy dwie **krotki,** mimo że teoretycznie mamy 3 książki (suma wartości w kolumnie ***Liczba_ksiazek***).

ISBN	Tytul	Autor	Wydawnictwo	Liczba_ksiazek	Rok_wydania
				1	1999
234-83-2623-741-0	Janko Muzykant	Henryk Sienkiewicz	Greg		
				2	2010
978-83-2623-748-0	W pustyni i w puszczy	Henryk Sienkiewicz	Zielona Sowa		

Dziedzina jest zbiorem wartości, jakie może przyjąć atrybut krotki. Jeśli kolumna tabeli przechowywać będzie numery kuli używanych do losowania Lotto, dziedzina atrybutu będą numery od 1 do 49. Gdy kolumna przechowywać będzie liczbę książek, wówczas zakładamy, że będą to wartości całkowite (ponieważ w bibliotece nie występują egzemplarze ułamkowe książek, n 0,5 książki), zatem dziedzina kolumny będą wartości całkowite **integer.** W bazi danych typ kolumny (dziedzina) może zostać zdefiniowany jako **int(11),** co oznacza, że w kolumnie książki mogą znajdować się liczby całkowite o maksymalnej długości nieprzekraczającej jedenastu znaków.

Projektowanie relacyjnej bazy danych wymaga stosowania fachowych określeń, których uzywać powinno się nie tylko w odniesieniu do budowy tabel, lecz także w stosunku do relacji – związków tworzonych pomiędzy tabelami.

Klucz główny

Dość często spotykanym problemem na etapie projektowania bazy danych jest określenie, która kolumna (kolumny) będzie pełnić funkcję klucza głównego. Ponieważ każdy wiersz w tabel musi my jednoznacznie zidentyfikować , zachodzi potrzeba wybrania atrybutów (kolumn), które spełniają to zadanie. Klucz główny odgrywa bardzo ważną rolę w tabeli (relacji), dlatego jego wybór powinien zostać poprzedzony analizą typowanych przez nas kolumn pod kątem wymienionych poniżej własności:

- **trwałość –** wartość kolumny powinna być stale obecna w wierszu, oznacza to , że kolumna taka (należąca do klucza głównego) nie może zawierać wartości NULL.
- **unikatowość –** wartość klucza dla każdego wiersza powinna być unikatowa, ponieważ w niepowtarzalny sposób powinien on identyfikować każdą krotkę (wiersz tabeli). Może się zdarzyć, że taki niepowtarzalny identyfikator otrzymamy, umieszczając w kluczu głównym więcej niż jedną kolumnę. Kombinacja wartości, trzech kolumn, które należeć będą do klucza, będzie unikatowa i jednoznacznie zidentyfikuje każdą krotkę.
- **stabilność –** wartości klucza nie powinny podlegać zmianom. Nie powinno się jako kluczy głównych używać kolumn przechowujących wartości nietrwałe, np. numer telefonu komórkowego, ponieważ mimo jego unikatowości każdy człowiek może go zmienić.

Aby jednoznacznie zidentyfikować wiersz tabeli, stosuje się **atrybut** (kolumnę), której poszczególne wartości dla kolejnych **krotek** (wierszy) będą niepowtarzalne.

ISBN	Tytul	Autor	Wydawnictwo	Liczba_ksiazek	Rok_wydania	
		Henryk			1	1999
234-83-2623-741-0	Janko Muzykant	Sienkiewicz	Greg			
		Henryk				2010
978-83-2623-748-0	W pustyni i w puszczy	Sienkiewicz	Zielona Sowa	2		

W powyższym przykładzie taką kolumna jest ISBN, ponieważ dla każdej książki jest on unikatowy. Oczywiście dany ISBN będzie identyfikował grupę książek o tym samym autorze, tytule oraz roku i wersji wydania, a także zawartości. Ponieważ każda z tych książek będzie identyczna pod względem zawartości, nie ma potrzeby rozryć w tabeli kolejnych wierszy, wystarczy wpisać, że mamy dwie książki pod tytułem *W pustyni i w puszczy*, a każda z nich jest wierna kopią poprzedniczki.

Atrybut będący **kluczem głównym** możemy stworzyć sztucznie dla przykładu wprowadzając kolejne numerowanie wierszy 1, 2, 3, 4, 5 itd., pod warunkiem że każdy wiersz ma inny numer. Możemy również posłużyć się określoną cechą (**atrybutem**) opisywanej rzeczywistości (**encji**), np. dokonując spisu ludności, możemy posłużyć się numerem PESEL. Ponieważ każdy człowiek ma inny niepowtarzalny numer PESEL, nie zachodzi obawa, że może on się powtórzyć. Taką kolumnę (atrybut) nazywamy **kluczem Głównym** *(primary key).*

Rodzaje kluczy

Zdarza się, że baza danych nie przechowuje pojedynczej tabeli, lecz wiele tabel, pomiędzy którymi występują powiązania. Aby pomiędzy tabelami można było tworzyć związki, powinny one mieć przynajmniej jeden atrybut (kolumnę), który musi spełniać dwa podstawowe warunki. Musi być unikatowy (oznacza to, że jego wartości nie mogą się powtarzać). Drugim warunkiem jest jego atomowość (musi być minimalny, tzn. powinien zawierać elementarne, niepodzielne wartości). Możemy rozróżnić następujące rodzaje kluczy:

- **klucz prosty –** to taki, który jest jednoelementowy (składa się z jednej kolumny),
- **klucz złożony –** to taki, który jest kilkuelementowy (składa się z więcej niż jednej kolumny).

Kluczem może być zatem jedna lub kilka kolumn, które wspólnie będą w stanie jednoznacznie zidentyfikować pozostałe dane w tabeli (relacji). Kolumny, które należą do kluczy (używa się ich do jednoznacznej identyfikacji wierszy w tabeli), nazywamy **atrybutami podstawowymi.** Kolumny nie należące do kluczy (zawierają dane, które w określonej relacji są przedmiotem opisu) nazywamy **atrybutami opisowymi.**

Do łączenia dwóch tabel (np. A i B) za pomocą związków używa się klucza. Klucz pochodzący z obcej tabeli B (w której jest on kluczem głównym), używany do łączenia tej tabeli z tabelą A, będzie dla tabeli A **kluczem obcym**.

Superklucz (*superkey***) –** to kolumna lub zestaw kolumn jednoznacznie identyfikujących każda krotkę tabeli. Super klucz może zawierać kolumny, które samodzielnie mogą nie identyfikować każdej z krotek. Unikatowa identyfikacja każdej z krotek może odbywać się jedynie przez zestaw np. dwóch lub trzech atrybutów. Przedmiotem zainteresowań projektantów baz danych jest taki superklucz, który zawiera minimalny zestaw atrybutów unikatowo identyfikujących krotkę.

Klucz kandydujący (nadklucz, klucz potencjalny) o super klucz zawierający minimalną liczbę kolumn unikatowo identyfikujących krotki relacji. W praktyce to kolumna lub kolumny, których użycie w charakterze klucza głównego jest rozważane przez projektanta baz danych. To właśnie twórca bazy danych decyduje, której kolumnie (kolumnom) nada funkcję klucza głównego.

Klucz główny (*primary key***)** to klucz, który został wybrany, aby unikatowo identyfikować krotki tabeli. Klucz główny jest podyktowany wyborem projektanta bazy danych.

Przykład

Rozważmy przykładową sytuację, w której projektujemy relację zawierającą np., spis lokatorów nieruchomości . Poniżej przedstawiono wycinek takiej tabeli.

Id	Imię	Nazwisko	PESEL
1	Maiej	Nowak	74092932157
2	Józef	Kowalski	65112164813

Ponieważ mogą występować osoby o tym samym imieniu i nazwisku, wybór atrybutów ***Imię*** lub ***Nazwisko*** jako klucza głównego mógłby powodować błędy w wypadku wystąpienia dwóch osób o tych samych imionach i nazwiskach. Ponieważ każdy człowiek ma niepowtarzalny unikatowy numer PESEL, atrybut ***PESEL*** spełnia wymagania kolumny jednoznacznie identyfikującej każdą krotkę. Projektant baz danych jako klucza głównego wybrał atrybut ***Id*** . Przykład ten pozwala zrozumieć różnicę pomiędzy kluczem głównym a kluczem kandydującym. W tym przypadku kluczami kandydującymi mogą być ***Id*** lub ***PESEL***. Ponieważ projektant, definiując tabelę, mianował ***Id*** kluczem głównym atrybut ***PESEL*** pozostał kluczem kandydującym.

Klucz obcy – kolumna lub zestaw kolumn w jednej tabeli, która pasuje do klucza kandydującego drugiej lub tej samej tabeli. Klucze obce wyznaczane są zwykle podczas tworzenia związków pomiędzy tabelami. Dowolne dane w kolumnie (kolumnach) klucza głównego tabeli A muszą mieć swoje odpowiedniki w odpowiadającej kolumnie (kolumnach) tabeli B (zgodność danych kolumn w dwóch tabelach). Zjawisko posiadania odpowiedników klucza głównego w obrębie różnych tabel nazywamy **integracją referencyjną.** Klucz obcy definiowany jest również jako kombinacja jednego lub kilku atrybutów (pełniących funkcję klucza głównego) wykorzystywanych do tworzenia związków pomiędzy tabelami lub w obrębie jednej tabeli.

Oprócz wymienionych powyżej definicji, istnieją również takie pojęcia jak:

- **klucz prosty** – klucz zawiera tylko jeden atrybut,
- **klucz złożony** – składa się z więcej niż jednego atrybutu,
- **klucz potencjalny** inaczej **klucz kandydujący** (*candidate key*).