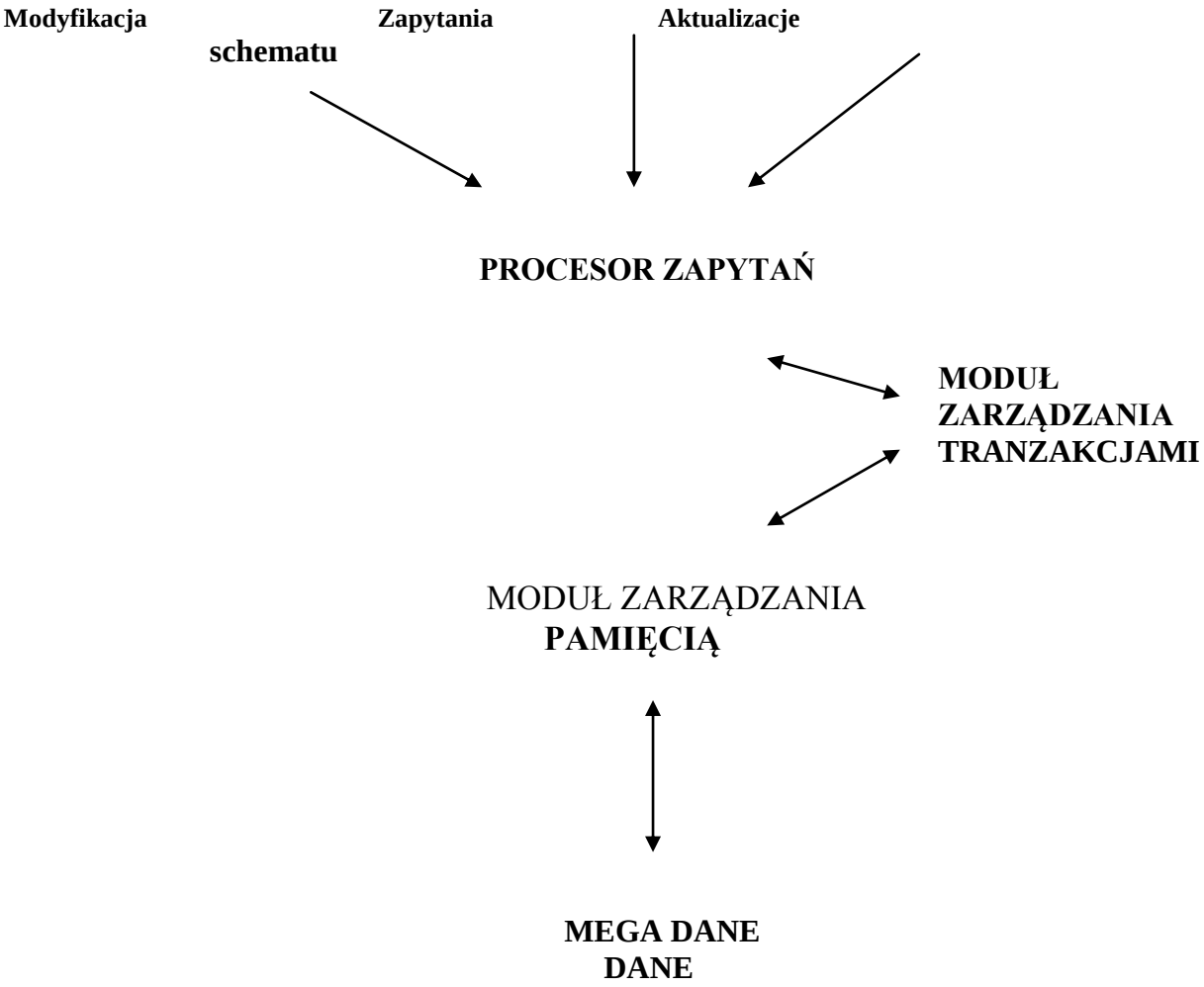
# Baza danych – zbiór danych

**SZBD – database management system**

**System zarządzania bazą danych**




**Dane** – służy do zapisu danych oraz **indeksów**, które reprezentują strukturę danych

**Moduł zarządzania pamięcią** – wybiera właściwe dane z pamięci. Składa się:

Moduł zarządzania plikami – przechowuje dane o miejscu zapisywania plików

z danymi na dysku i na polecenie modułu zarządzania buforami przesyła zawartość bloku z dysku do pamięci operacyjnej.

Moduł zarządzania buforami – obsługuje pamięć operacyjną. Wybiera z pamięci strony przydzielone do wybranych z dysku bloków.

**Moduł zarządzania zapytaniami –** obsługuje realizacje zapytań a le nie tylko również aktualizacji danych. Przetwarza zapytania i operacje wyrażone w języku wysokiego poziomu (np. SQL) na ciąg poleceń określonych danych (np. danych do aktualizacji). **Optymalizacja zapytania -** optymalne przeszukiwanie danych.

**Moduł zarządzania transakcjami –** gwarantuje określone (specjalne) wykonanie niektórych operacji.

**Własności transakcji:**

1. niepodzielność – cała transakcja musi być wykonana
2. izolacja – wykonywanie dwóch transakcji jednocześnie jest blokowane
3. trwałość – jeśli transakcja się zakończy to jej wynik nie może być utracony z powodu awarii.

Mechanizmy gwarantujące spełnienie powyższych własności:

1. blokady – blokowanie elementu, którego dotyczy transakcja (np. równoczesny dostęp do konta)
2. logi – moduł zarządzania transakcjami dokumentuje wszystkie operacje, tzn. rozpoczęcie każdej transakcji, zmiany w bazie danych dokonane przez transakcje oraz zakończenie transakcji. Zapis ten nazywa się **logiem**. Log jest przechowywany w pamięci stałej. , co zapewni przetrwanie danych w przypadku awarii.

3. zatwierdzenie transakcji – w chwili gdy transakcja zakończy działanie, jest gotowa do zatwierdzenia, zmiany są kopiowane do logu. Dopiero potem następuje aktualizacja danych.

# Relacyjny model danych

Model relacyjny dostarcza tylko jednego sposobu reprezentowania danych: jest nim dwuwymiarowa **tabela**, nazywana **relacją**.

1.Relacja (nazwa)

2.Atrybut (nazwa, typ, dziedzina)

3. Krotka (wartość atrybutów)

**Relacja Film**

| Tytuł | Rok | Długość | TypFilmu | NazwaStudia |
|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount |

**Schemat relacji** - nazwa relacji oraz zbiór atrybutów

**Film (**tytuł, rok, długość, typFilmu, NazwaStudia**)**

**Krotka relacji –** wiersz relacji

(Gwiezdne wojny, 1977, 124, kolor, Fox)

**Dziedzina –** każdy atrybut musi mieć określony typ atomowy, tzn. jego typ musi należeć do typów elementarnych, całkowity, znakowy itp. Nie może być ani tablicą, ani strukturą ani zbiorem ani inną strukturą, którą można podzielić na wiele części.

**Instancja relacji** – zbiór krotek

## Relacja Gwiazda

| Tytuł | Rok | Nazwisko Gwiazdy |
|---|---|---|
| Gwiezdne wojny | 1997 | Carrie Fischer |
| Gwiezdne wojny | 1997 | Mark Hamill |
| Gwiezdne wojny | 1997 | Harrison Ford |
| Potężne Kaczory | 1991 | Emilio Estevez |
| Swiat Wayne'a | 1992 | Dana Carvey |
| Swiat Wayne'a | 1992 | Mike Meyers |

## Relacja samochód

| Model | Rok | Kolor | Cena | Dane |
|---|---|---|---|---|
| | | | | |

## Relacja książka

| Nr | Tytuł | Autor | Rodzaj | Cena | Wydawnictwo |
|---|---|---|---|---|---|
| | | | | | |

**Zbiór identyfikującym relacji** R = { A1,A2, ..., An } nazywamy zbiór atrybutów $S \subseteq R$, który jednoznacznie identyfikuje wszystkie krotki relacji o schemacie R.

**Kluczem K** schematu relacji R nazywamy minimalny zbiór idenfikujący, tzn. taki, że nie istnieje K' $\subset$ K będące zbiorem identyfikującym schematu R. Klucze dzielą się na klucze proste i złożone.

**Klucz prosty, Klucz złożony, Klucz potencjalny, Klucz główny**

**Atrybut podstawowy, Atrybut wtórny**

Normalizacja relacji ma na celu takie przekształcenie by nie posiadała ona cech niepożądanych:

***1. Redundancja danych,***

***2. Brak możliwości wyszukiwania danych według określonego warunku***

**Przykład**

***Szkoła*** ( nazwa przedmiotu, imię, nazwisko, adres prowadzącego)
Krotka (j. Angielski, Lucyna, Nowak, ul. Cicha 8 Warszawa)

1. Adres składa się z kilku części
2. Redundancja danych jeden prowadzący może mieć zajęcia z kilku przedmiotów
3. Zmiana jednej z informacji o prowadzącym (np. adresu) powoduje konieczność zmiany wszystkich krotek zawierających te dane w celu zachowania integralności
4. Nie jest możliwe wprowadzenie informacji o prowadzącym, który w danym semestrze nie ma żadnych zajęć

5. Usuniecie przedmiotu może spowodować również usuniecie wszelkich informacji o prowadzącym.

Utrzymanie integralności takiej bazy jest bardzo trudne.

***Dekompozycja relacji Szkoła na dwie relacje Nauczyciel i Przedmiot***

***Nauczyciel*** (Id_prowadzącego , imię, nazwisko, kod pocztowy, miejscowość, ulica)

***Przedmiot (***nazwa przedmiotu, Id_prowadzącego)

***Nauczyciel***

*Klucz podstawowy*

| **Id_prowadzącego** | imię | Nazwisko | kod pocztowy | Miejscowość | ulica |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

***Przedmiot***

*Klucz obcy*

| **Id_prowadzącego** | nazwa przedmiotu |
|---|---|
| | |
| | |
| | |

## Więzy integralności

*Tabela nadrzędna* *Tabela podrzędna*

***Usuwanie - kaskada lub blokada*** ***Usuwanie - OK!***

***Modyfikacja – kaskada lub blokada*** ***Modyfikacja – kaskada lub blokada***

Dodawanie - OK.! Dodawanie - blokada

**Definicja funkcji zależności:**

**Zależność funkcyjna w relacji R** - jeśli dwie krotki relacji R są zgodne dla atrybutów A1,A2, ..., An to muszą być również zgodne w pewnym atrybucie B

***A1,A2, ..., An -> B***

Atrybut B jest **funkcjonalnie zależny** od atrybutu A tej relacji (A identyfikuje B) jeśli dla dowolnej wartości a atrybutu A odpowiada nie więcej niż jedna wartość b atrybutu B.
Jeśli zbiór atrybutów określa funkcyjnie więcej niż jeden atrybut

***A1,A2, ..., An -> B1***

***A1,A2, ..., An -> B2***

.

.A1,A2, ..., An -> Bm

***A1,A2, ..., An -> B1, B2, ..., Bm***

**Zasady podziału i łączenia**

**A1, A2, ...., An -> B1, B2,....Bm**

To

A1, A2, ...., An -> B1,

A1, A2, ...., An -> B2

***A1, A2, ...., An -> Bm***

## Zasada przechodności

W relacji R z atrybutami A, B, C zachodzą związki funkcyjne

A->B i B -> C  to A ->C

Np. (a,b1,c1)  (a, b2, c2)

## Relacja Film

| Tytuł | Rok | długość | Typ filmu | Nazwa Studia | Nazwisko Gwiazdy |
|---|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Carrie Fischer |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Mark Hamill |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Harrison Ford |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney | Emilio Estevez |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Dana Carvey |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Mike Meyers |

## Zależności funkcyjne

**1 Tytułrok -> długość**

2 Tytuł rok -> typ filmu

3 Tytuł rok -> nazwa studia

**Tytuł rok -> długość typ filmu nazwa studia**

**zła zależności funkcyjna**

**Tytułrok -> nazwisko gwiazdy**

## Klucze relacji

Zbiór atrybutów A1, A2, ...., An tworzy klucz relacji, jeśli:

1. Wszystkie pozostałe atrybuty relacji są funkcyjnie zależne od tych atrybutów tzn. nie ma dwóch różnych krotek relacji R zgodnych dla wszystkich atrybutów A1, A2, ...., An

2. Nie istnieje podzbiór właściwy zbioru A1, A2, ...., An ,od którego pozostałe atrybuty są zależne funkcyjnie, tzn. klucz musi być minimalny

Przykład:
Atrybuty (tytuł, rok, nazwiskoGwiazdy) tworzą klucz relacji Film.
Należy wykazać, że pozostałe atrybuty w relacji Film są od nich funkcyjnie zależne.
Należy wykazać, że żaden z podzbiorów właściwych zbioru (tytuł, rok, nazwiskoGwiazdy) nie wyznacza pozostałych atrybutów w sposób funkcyjny.
Przeanalizujmy pary atrybutów
(tytuł, rok) (tytuł, nazwiskoGwiazdy) (rok, nazwiskoGwiazdy)

**Klucz dla relacji Film**

**Tytułrok nazwisko gwiazdy**

błędny klucz **rok nazwisko gwiazdy**

## Nadklucze

**Zbiór atrybutów, który zawiera klucz**

**Nadklucze dla relacji film**

| | | | | |
|---|---|---|---|---|
| Tytuł | rok | nazwisko gwiazdy | | |
| Tytuł | rok | nazwisko gwiazdy | długość | |
| Tytuł | rok | nazwisko gwiazdy | długość | typ filmu |
| Tytuł | rok | nazwisko gwiazdy | typ filmu | |

## Zależności trywialne

Mówimy, że zależność funkcyjna A1, A2, ...., An -> B1, B2, ..., Bm jest trywialna, jeśli zbiór B1, B2, ..., Bm jest podzbiorem A.

**Tytuł rok -> rok**

Tytuł rok -> Tytuł

**Mówimy, że zależność funkcyjna A1, A2, ...., An -> B1, B2,....Bm jest:**

Trywialna $A1, A2, ...., An \subseteq B1, B2,....Bm$

Nietrywialna $\exists\ B \subset A$

Całkowicie nietrywialna $\neg\exists\ B \subset A$

**Tytułrok -> rok długość - nietrywialna**
**Tytułrok -> długość - całkowicie nietrywialna**

## Pierwsza postać normalna

Relacja R jest w pierwszej postaci normalnej, jeśli wartości atrybutów są elementarne tzn. są to pojedyncze wartości określonego typu, a nie zbiory wartości.

## Postać normalna Boyce'a – Codda

***Relacja R jest w postaci BCNF wtedy i tylko wtedy, gdy dla każdej nietrywialnej zależności A1, A2, ...., An ->B zbiór { A1, A2, ...., An } jest nadkluczem R.***

Oznacza to, że lewa strona każdej zależności nietrywialnej musi być nadkluczem.

Relacja R jest w postaci BCNF wtedy i tylko wtedy, gdy dla każdej nietrywialnej zależności A1, A2, ...., An -> B1, B2,....Bm zbiór { A1, A2, ...., An } jest nadkluczem R.

*Badanie postaci BCNF dla relacji Film*

**Relacja Film**

| Tytuł | Rok | Długość | Typ filmu | Nazwa Studia | Nazwisko Gwiazdy |
|---|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Carrie Fischer |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Mark Hamill |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Harrison Ford |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney | Emilio Estevez |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Dana Carvey |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Mike Meyers |

*Klucz - Tytuł rok nazwiskoGwiazdy*

**Rozważmy zależność funkcyjną**

Tytuł rok -> długość typ filmu nazwa studia

**Jest zależność funkcyjna, ale lewa strona nie jest nadkluczem**

**Dekompozycja Relacji Film**

| Tytuł | Rok | długość | Typ filmu | Nazwa Studia |
|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount |

**Relacja Film 1**

| Tytuł | Rok | Długość | Typ filmu | Nazwa Studia |
|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount |

## Relacja Film 2

| Tytuł | Rok | Nazwisko Gwiazdy |
|---|---|---|
| Gwiezdne wojny | 1997 | Carrie Fischer |
| Gwiezdne wojny | 1997 | Mark Hamill |
| Gwiezdne wojny | 1997 | Harrison Ford |
| Potężne Kaczory | 1991 | Emilio Estevez |
| Swiat Wayne'a | 1992 | Dana Carvey |
| Swiat Wayne'a | 1992 | Mike Meyers |

## Relacja Film

| Tytuł | Rok | Długość | Typ filmu | Nazwa Studia | Nazwisko Gwiazdy | Prezes | Adres studia | Adres prezesa |
|---|---|---|---|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Carrie Fischer | A ... | Holly. | a... |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Mark Hamill | A ... | Holly. | a... |
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox | Harrison Ford | A ... | Holly. | a... |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney | Emilio Estevez | B ... | Holly.1 | b... |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Dana Carvey | C ... | Holly.2 | c.... |
| Swiat Wayne'a | 1992 | 95 | Kolor | Paramount | Mike Meyers | C ... | Holly.2 | c... |

*Klucz - Tytuł rok nazwiskoGwiazdy*

## Rozważmy zależność funkcyjną

Tytuł rok -> długość typ filmu nazwa studia
Tytuł rok -> nazwaStudia
nazwaStudia -> adresStudia
Tytuł rok -> adresStudia
Tytuł rok -> długość typ filmu

**Są zależność funkcyjna, ale lewe strony nie jest nadkluczem**

*Zależność przechodnia*

## Dekompozycja Relacji Film

## Relacja Film 1

| Tytuł | Rok | Długość | Typ filmu | Nazwa Studia |
|---|---|---|---|---|
| Gwiezdne wojny | 1997 | 124 | Kolor | Fox |
| Potężne Kaczory | 1991 | 104 | Kolor | Disney |
| Swiat Wayne’a | 1992 | 95 | Kolor | Paramount |

## Relacja Film 2

| Tytuł | Rok | Nazwisko Gwiazdy |
|---|---|---|
| Gwiezdne wojny | 1997 | Carrie Fischer |
| Gwiezdne wojny | 1997 | Mark Hamill |
| Gwiezdne wojny | 1997 | Harrison Ford |
| Potężne Kaczory | 1991 | Emilio Estevez |
| Swiat Wayne’a | 1992 | Dana Carvey |
| Swiat Wayne’a | 1992 | Mike Meyers |

## Relacja Film 3

| Nazwa Studia | Adres studia |
|---|---|
| Fox | Holly. |
| Disney | Holly.1 |
| Paramount | Holly.2 |

## Relacja Film 4

| Nazwa Studia | Prezes |
|---|---|

| | |
|---|---|
| Fox | A ... |
| Disney | B ... |
| Paramount | C ... |

**Relacja Film 5**

| Prezes | Adres prezesa |
|---|---|
| A ... | a... |
| B ... | b... |
| C ... | c... |

## Odzyskiwanie danych po dekompozycji

Załóżmy, że relacja R jest zdefiniowana {A, B, C}, ale nie zachodzi w niej zależność B -> C. Wówczas w R mogą występować następujące krotki.

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 2 | 5 |

Dekompozycja

{A, B}

| A | B |
|---|---|
| 1 | 2 |
| 4 | 2 |

{B, C}

| B | C |
|---|---|
| 2 | 3 |
| 2 | 5 |

Po połączeniu

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 2 | 5 |

| | | |
|---|---|---|
| 4 | 2 | 3 |
| 4 | 2 | 5 |

**PRZYKŁAD**

SELECT tytuł AS nazwa, długość AS czasTrwania

FROM Film

WHERE nazwa STUDIA= ‘Disney’ and rok = 1990;

| *nawa* | *czasTrwcmia* |
|---|---|
| Pretty | 119 |

**PRZYKŁAD**

SELECT tytuł AS nazwa, długość * 0.016667 AS czasWGodzinach

**PRZYKŁAD**

W klauzuli SELECT można również umieszczać stałe. Stosując następujące zapytanie:

SELECT tytuł, długość * 0.016667 AS długość, 'godz.'AS wGodzinach

FROM Film

WHERE nazwaStudia = 'Disney'' AND rok = 1990;

| *Tytuł* | *długość* | *wGodzinach* |
|---|---|---|
| Pretty | l.9833 | godz. |

Do zapisu porównywania wartości w języku SQL służy sześć operatorów: =, < >, <, >, <=, oraz >=. Ich znaczenie Jest powszechnie znane, jest ono takie same jak w Pascalu (dla

przeciwników Pascala przypominamy, że tam <> oznacza „nierówne").

W wyrażeniu mogą występować stałe oraz atrybuty tych relacji, które są wymienione w klauzuli FROM. wartości numeryczne możemy łączyć w wyrażenia arytmetyczne, korzystając ze zwyczajowych operatorów +, * itp. Na przykład wartością warunku (rok - 1930) * (rok - 1930) < 100 jest prawda, jeśli wartość atrybutu rok oznacza pewien rok z lat trzydziestych. Z kolei teksty można konkatenować, stosując w tym celu operator ||, na przykład wyrażenie 'dwie' ||'belki', oznacza to samo co 'dwie-belki'.

**PRZYKŁAD**

```
SELECT tytuł
FROM Film
 WHERE rok < 1 970  AND NOT czyKolor;
```

```
SELECT tytuł
FROM Film
WHERE (rok < 1970 OR długość < 90)  AND nazwaStudia = 'MGM' ;
```

Wyrażenie:

```
s LIKE p
```

jest porównaniem, w którym jest tekstem, a *p* pewnym *wzorcem,* tzn. takim tekstem, w którym mogą wystąpić szablony, czyli w tym przypadku znaki % oraz _. Inne znaki w napisie *s* muszą być dokładnie równe znakom z wzorca *p,* natomiast szablonowi % z *p* może odpowiadać w *s* dowolny ciąg znaków, także o długości 0; z kolei znakowa _ z wzorca p odpowiada jeden dowolny znak w tekście *s.* Wartość tego porównania wynosi prawda wówczas, gdy s pasuje do wzorca p. Analogicznie jest zdefiniowane wyrażenie *s* NOT LIKE P którego wartością jest prawda wówczas, gdy tekst *s* nie pasuje do wzorca p.

**Znak wyjątku w wyrażeniu LIKE**

W języku SQL można dowolnego znaku użyć jako znaku wyjątku. Definiuje się go za pomocą słowna kluczowego ESCAPE, po którym umieszcza się ten wybrany znak otoczony apostrofami. Wówczas, jeśli we wzorcu znak % lub _ zostanie poprzedzony tym wybranym znakiem wyjątku, to będzie on traktowany dosłownie jako znak °/o lub _, a niejako szablon. Na przykład sformułowanie

```
s LIKE 'x%%x% ESCAPE  ‘x’
```

**PRZYKŁAD**

```
SELECT tytuł
FROM Film
WHERE tytuł LIKE  ‘Gwiezdn_  _ _ _ _ _’
```

**PRZYKŁAD**

Wyszukajmy teraz te angielskojęzyczne tytuły filmów, *w* których występuje apostrof '. Zapytanie przybiera wówczas następującą postać:

```
SELECT tytuł
FROM Film
WHERE tytuł LIKE ‘ %‘’s%’
```

Przyjęto konwencję, w której występujący w tekście ciąg dwóch bezpośrednio po sobie następujących apostrofów oznacza znak apostrofu, a nie nawias stałej tekstowej. Dlatego do zawartego we wzorcu ciągu znaków ‘s pasują te tytuły, w których występuje ciąg znaków '*s*.

porządkowanie wyniku

```
ORDER BY <lista atrybutów >
```

Z założenia porządek jest rosnący, ale można go odwrócić, dopisując na końcu słowo DESC *(„descending* - tj. malejący). Można także użyć słowa ASC do określenia porządku rosnącego *(ascending),* ale nie jest to konieczne.

**PRZYKŁAD**

Film (tytuł, rok, długość, czyKolor, nazwaStudia, producentC#)

```
SELECT *
FROM Film
WHERE nazwaStudia = "Disney" AND rok = 1990
ORDER BY długość, tytuł;
ORDER BY 3,1;
```

## Modyfikacje bazy danych

1. Wstawianie nowych krotek
2. Usuwanie pewnych krotek
3. Zmiany wartości pewnych składowych w określonych krotkach

### Wstawianie nowych krotek

```
INSERT INTO   R(A1, A2, ... An)   VALUES  (V1, V2,...V3)
```

***PRZYKŁAD***

```
INSERT INTO   Gwiazdy (tytuł, rok, nazwisko)   VALUES  (IT, 1990,IT)

INSERT INTO   Gwiazdy    VALUES  (IT, 1990,IT)
```

**Dołączenie szeregu krotek – podzapytanie występuje zamiast klauzuli VALUES**

Studio (nazwa, adres, prezC)
Film (tytuł, rok, długość, nazwaStudia, producentC)

```
INSERT INTO Studio(nazwa)
```

```
SELECT DISTINCT nazwaStudia
FROM Film
WHERE nazwaStudia NOT IN
(SELECT  nazwa FROM Studio)
```

**Atrybuty adres i prezC są uzupełnione wartościami NULL**

Usuwanie pewnych krotek

DELETE FROM  R WHERE <warunek>

***PRZYKŁAD***

Gwiazdy (tytuł, rok, nazwisko)

```
DELETE FROM  Gwiazdy
    WHERE tytuł = 'IT' AND
          rok = 1990 AND
          nazwisko = 'IT';
```

```
DELETE FROM  Film
    WHERE rok < 1990;
```

**AKTUALIZACJA**

UPDATE R SET <nowe przypisania> WHERE  <warunek>

```
UPDATE FilmDyr
SET nazwisko = 'Prez.' || nazwisko
WHERE  cert (IN SELECT prez FROM STUDIO)
```

## Proste deklaracje tabel

```
CREATE TABLE gwiazda (
        Nazwisko CHAR(30),
        Adres VARCHAR(255),
        Płeć    CHAR(1),
        DataUrodzenia DATE
);
```

## Usuwanie tabel

```
Drop R;
```

## Zmiany schematów relacji

Niech modyfikacja tabeli Gwiazda polega na tym, że trzeba do niej dodać atrybut telefon

```
ALTER TABLE Gwiazda ADD telefon CHAR(16);
```

Przypisano polu telefon wartości NULL

```
ALTER TABLE Gwiazda DROP dataUrodzenia
```

Przypisano wartości NULL

**Wartości domniemane**

```
CREATE TABLE gwiazda (
        Nazwisko CHAR(30),
        Adres VARCHAR(255),
        Płeć    CHAR(1) DEFAULT ‘?’ ,
        DataUrodzenia DATE DEFAULT ‘0000-00-00’
);
```

```
ALTER TABLE Gwiazda ADD telefon CHAR(16) DEFAULT ‘nieznany’;
```

**Indeksy**

```
CREATE INDEX indeksRoku ON film (rok)
```

```
CREATE INDEX indeksKlucza ON film (tytuł, rok)
```

```
DROP INDEX indeksRoku
```

**Klucze**

```
CREATE TABLE gwiazda (
        Nazwisko CHAR(30), PRIMARY KEY
        Adres VARCHAR(255),
        Płeć    CHAR(1)
        DataUrodzenia DATE
);
```

LUB

```
CREATE TABLE gwiazda (
        Nazwisko CHAR(30),
        Adres VARCHAR(255),
        Płeć    CHAR(1)
        DataUrodzenia DATE
```

```
        PRIMARY KEY(Nazwisko)
);

CREATE TABLE gwiazda (
        Nazwisko CHAR(30),
        Adres VARCHAR(255),
        Płeć    CHAR(1)
        DataUrodzenia DATE
        PRIMARY KEY(Nazwisko,adres)
);
```

nazwisko CHAR(30) UNIQUE

Adres VARCHAR(255), UNIQUE

UNIQUE (nazwisko);

**Klucz obcy**

Atrybuty klucza obcego muszą być zadeklarowane jako klucz główny w orginalnej relacji.

Jeśli klucz obcy jest pojedynczym atrybutem

REFERENCES <tabela>  (<atrybut>)

FOREIGN KEY <atrybuty> REFERENCES <tabela>  (<atrybuty>)

Przykład

Studio (nazwa, adres, prez)

Relacja zawiera klucz główny nazwa oraz klucz obcy prez, określający odwołanie do atrybutu cert w relacji

FilmDyr (nazwisko, adres, cert)

```
CREATE TABLE Studio (
Nazwa CHAR(30) PRIMARY KEY,
```

Adres VARCHAR(255),

Prez INT REFERENCES FilmyDyr(cert)
);
lub
CREATE TABLE Studio (
Nazwa CHAR(30) PRIMARY KEY,

Adres VARCHAR(255),

prez INT,

FOREIGN KEY prez REFERENCES FilmyDyr (cert)
);

## Procedura kaskadowa i procedura wstawiania wartości NULL

ON DELETE  SET NULL  usuwanie
ON DELETE  CASCADE
ON UPDATE  SET NULL
ON UPDATE  CASCADE  modyfikacja

```
CREATE TABLE Studio (
        Nazwa CHAR(30) PRIMARY KEY,
        Adres VARCHAR(255),
        Prez INT REFERENCES FilmyDyr(cert)
        ON DELETE  SET NULL
        ON UPDATE  CASCADE
);
```

### Więzy NOT-NULL

Prez INT REFERENCES FilmyDyr(cert) NOT-NULL

### Więzy CHECK

Prez INT REFERENCES FilmyDyr(cert) CHECK (prez >=100000)

Płeć CHAR(1) CHECK (płeć IN (‘K’, ‘M’))

### Więzy integralności referencyjnej za pomocą Check

Studio (nazwa, adres, prez)

Relacja zawiera klucz główny nazwa oraz klucz obcy prez, określający odwołanie do atrybutu cert w relacji

FilmDyr (nazwisko, adres, cert, cenaSieci)

```
CREATE TABLE Studio (
        Nazwa CHAR(30) PRIMARY KEY,
        Adres VARCHAR(255),
        Prez INT CHECK
             (Prez IN (Select cert From FilmDyr));
```

Zmiana relacji FilmDyr zostaje niezauważona.

Create Assertion <nazwa> Check <warunek>

Warunek zapisany w asercji musi być zawsze spełniony i każda modyfikacja, która zaburza ten stan , nie jest wykonywana.

Przykład

Załóżmy, że prezes studia musi posiadać sieć wartą nie mniej niż 10.000.000 $

Asercja dotyczy dwóch relacji
Studio (nazwa, adres, prez)

Relacja zawiera klucz główny nazwa oraz klucz obcy prez, określający odwołanie do atrybutu cert w relacji

FilmDyr (nazwisko, adres, cert, cenaSieci)

Create Assertion Bogatyprez Check

```
(Not Exists
      (select *
         from Studio, FilmDyr
         where prez=cert and
                  cenaSieci<10000000)
);
```

Warunek asercji został określony w ten sposób, że będzie spełniony, jeśli zbiór studiów filmowych, których prezesimają sieci warte mniej niż 10.000.000 $ jest pusty.

```
CREATE TABLE Studio (
        Nazwa CHAR(30) PRIMARY KEY,
        Adres VARCHAR(255),
        Prez INT References filmdyr(cert)
            CHECK Prez Not in
                (Select cert From FilmDyr
                        where cenaSieci<10000000 )
        );
```

Zauważmy, że więzy będą sprawdzane jedynie wtedy, gdy zajdzie zmiana dotyczaca relacji Studio. Aby uzyskać efekt działania asercji należy dołączyć do deklaracji tabeli FilmDyr jeszcze jedne więzy.

Przykład

Film (tytuł, rok, długość, nazwaStudia, producentC)

```
Create Assertion SumDł Check
        (10000>+ALL
                (select Sum (długość) From Film Group by
                  nazwaStudia);
        );

Check (10000>+ALL
         (select Sum (długość) From Film Group by
                nazwaStudia);
        );
```

## Definiowanie perspektyw

1. słowa kluczowego **CREATE VIEW**,
2. nazwy perspektywy,
3. słowa kluczowego **AS** oraz
4. zapytania *Q*. To zapytanie określa zakres danych perspektywy. Zawsze, gdy pewne zapytanie *K* odwołuje się do perspektywy, to SQL jakby od nowa wykonuje zapytanie *Q*, a dopiero potem wykonuje się zapytanie *K*, traktując wynik *Q* Jako swoją relację wejściową.

**CREATE VIEW** <nazwa-perspektywy> **AS** <definicja perspektywy>;

Film (tytuł, rok, długość, czyKolor, nazwaStudia, producentC#)

```
1) CREATE VIEW FilmyParamount AS
2)    SELECT tytuł, rok
3)    FROM Film
4)    WHERE nazwaStudia = 'Paramount';
```

### *Zapytania określane na perspektywach*

Przykładowa relacja **FilmyParamount** w sensie fizycznym nie zawiera krotek.
Jeśli jakieś zapytanie odwołuje się do jej nazwy, to wówczas są wyszukiwane z relacji Film krotki, które spełniają definicję perspektywy FilmyParamount i na nich operuje zapytanie. Dlatego wykonując dwa razy to samo zapytanie na jednej perspektywie, można uzyskać inne wyniki, mimo że sama relacja FilmyParamount nie uległa zmianie

- mogła jednak w międzyczasie zmienić się tabela bazowa.

**PRZYKŁAD**

Dla perspektywy **FilmyParamount** można określić zapytanie tak, jakby była ona zwykłą tabelą:

```
SELECT tytuł
FROM FilmyParamount
WHERE rok = 1979;
```

Zapytanie równoważne dotyczące relacji bazowej **Film**

```
SELECT tytuł
FROM Film
WHERE nazwaStudia = 'Paramount' AND rok = 1979;
```

**PRZYKŁAD**

Pisanie zapytań określonych zarówno na perspektywach, jak i tabelach

```
SELECT DISTINCT nazwiskoGwiazdy
FROM FilmyParamount, GwiazdyW
WHERE tytuł = tytułFilmu AND rok = rokFilmu;
```

W wyniku przedstawionego zapytania mają zostać określone nazwiska gwiazd, które wystąpiły w filmach wyprodukowanych przez wytwórnię Paramount.

**PRZYKŁAD**

Zapytanie definiujące perspektywę określone na dwóch relacjach, które zawiera tytuły filmów oraz nazwiska ich producentów.

Film (tytuł, rok, długość, czyKolor, nazwaStudia, producentC#)

FilmDyr (nazwisko, adres, cert#, cenaSieci)

```
1) CREATE VIEW FilmProd  AS
```

2) SELECT tytuł, nazwisko
3) FROM Film, FilmDyr
4) WHERE producentC# = cert#

SELECT nazwisko
FROM FilmProd
WHERE tytuł = ‘Przeminęło z wiatrem' ;

Zapytanie równoważne dotyczące relacji bazowej:

SELECT nazwisko
FROM Film, FilmDyr
WHERE producentC# = cert# AND tytuł = 'Przeminęło z wiatrem;

**Przemianowanie atrybutów**

CREATE VIEW FilmProd (tytułFilmu, nazwiskoProd) AS
SELECT tytuł nazwisko
FROM FilmDyr
WHERE producentC#= cert#;

**Modyfikowanie perspektyw**

Wykonywanie działań wstawiania, usuwania oraz zmian perspektyw.
W prostych perspektywach, tak zwanych „modyfikowalnych", można przekładać modyfikowanie perspektywy na równoważne działanie na tabeli bazowej, które zostanie dokonane na niej zamiast na perspektywie.

Modyfikacje perspektyw są dopuszczalne tylko wtedy:

- ✓ perspektywy są zdefiniowane przez selekcję (czyli SELECT lub SELECT DISTINCT) pewnych atrybutów z jednej relacji R (która sama także może być perspektywą modyfikowalną).
- ✓ w klauzuli SELECT musi być dostatecznie dużo atrybutów po to, by dla każdej krotki, którą wstawia się do perspektywy, można było wprowadzić pozostałe

atrybuty z wartościami NULL lub domyślnymi, i aby w relacji bazowej istniała krotka, która stanowi podstawę dla krotki umieszczanej w perspektywie.

**PRZYKŁAD**

Załóżmy, ze do perspektywy

```
1) CREATE VIEW FilmyParamount AS
2)    SELECT tytuł, rok
3)    FROM Film
4)    WHERE nazwaStudia = 'Paramount';
```

będzie wstawiana następująca krotka:

```
INSERT INTO FilmyParamount VALUES ('Star Trek' ,1979);
```

Perspektywa FilmyParamount nieomal spełnia warunki modyfikowalności, ponieważ wchodzą do niej dane tylko z jednej relacji

```
Film (tytuł, rok, długość, czyKolor, nazwaStudia, producentC#)
```

Problem pojawia się przy określaniu wartości nazwaStudia. ponieważ ten atrybut nie należy do perspektywy. Zatem jego wartością w krotce perspektywy nie jest ' Paramount, lecz NULL.

```
1) CREATE VIEW FilmyParamount AS
2)  SELECT nazwaStudia, tutuł, rok
3)  FROM Film
4)  WHERE nazwaStudia = ‘Paramount’;
```

Wówczas instrukcja wstawiania krotki do perspektywy przyjmie następującą postać:

```
INSERT INTO FilmyParamount
VALUES (Paramount’, 'Star Trek’, 1979) ;
```

Wartości atrybutów **długość**, **czyKolor** i **producentC#** - odpowiednie wartości domniemane: albo NULL, albo inna wartość domniemana, którą zdefiniowano dla atrybutu lub jego dziedziny. Na przykład, jeśli dla atrybutu długość określono wartość 0, a dla pozostałych dwóch wartość NULL, to krotka wstawiana do relacji Film

wygląda następująco:

| *Tytuł* | *rok* | *Długość* | *czyKolor* | *nazwaStudia* | *ProducentC#* |
|---|---|---|---|---|---|
| ‘Star | 1979 | 0 | NULL | ‘Paramount’ | null |

Z perspektywy modyfikowalnych można także usuwać krotki. Usuwanie, podobnie jak wstawianie, jest przetwarzane w relacji bazowej *R*, a w jego wyniku zostają z relacji *R* usunięte te krotki, które nie powinny już generować żadnych danych do perspektywy.

**PRZYKŁAD**

Tym razem zadanie polega na usunięciu z perspektywy modyfikowalnej FilmyParamount wszystkich filmów, w których tytułach występuje „Trek". Instrukcja usuwania ma postać następującą:

```
DELETE FROM FilmyParamount
WHERE tytuł LIKE ‘%Trek%' ;
```

Równoważne usuwanie, określone dla bazowej relacji **Film.**

```
DELETE FROM Film
WHERE tytuł LIKE ‘%Trek%' AND nazwaStudia = ‘Paramount’;
```

**PRZYKŁAD**

Modyfikowanie perspektywy polegające na wykonaniu następującej instrukcji:

```
UPDATE FilmyParamount
SET rok = 1979
WHERE tytuł - 'Film Star Trek';
```

Równoważne modyfikowanie określone dla bazowej relacji **Film.**

```
UPDATE Film
SET rok =1979
WHERE tytuł = ' Film Star Trek' AND nazwaStudia = ' Paramount’;
```

**Usuwanie perspektyw**

```
DROP VIEW FilmyParamount;
```

## Więzy i wyzwalacze w języku SQL

Elementy aktywne - wyrażenia lub instrukcje, które raz zapisane w bazie danych są automatycznie uruchamiane w odpowiedniej chwili.
Czas wykonania elementu aktywnego - zajście pewnego zdarzenia np. próba wstawienia rekordu
Więzy integralności:

Więzy kluczy - czyli więzy określone na atrybutach lub zbiorach atrybutów pełniących funkcje kluczy tabeli.
Więzy integralności referencyjnej - tzn. wymagania polegające na tym, żeby wartości atrybutu lub zbioru atrybutów jednej tabeli występowały również w innej tabeli (np. cert# w FilmDyr).
Więzy atrybutowe - które można nakładać na atrybuty, włączając w to niepowtarzalność pewnych wartości („jednoznaczność"), ograniczenia zakresu wartości atrybutów, ochronę przed występowaniem wartości pustych NULL.
Więzy krotkowe - więzy dotyczące rekordów lub tabel.
Więzy globalne (asercje) - więzy dotyczące schematu bazy danych.

Zachowanie więzów jest zawsze testowane, gdy modyfikacja dotyczy tabeli, na której są one określone.

## Klucze w języku SQL

**Klucz główny** tabeli opisującej przechowywaną relację można deklarować w instrukcji CREATE TABLE na dwa różne sposoby:

```
1) CREATE TABLE GwiazdaFilmowa (
2)     nazwisko CHAR(30) PRIMARY KEY,
3)     adres VARCHAR(255),
4)     płeć CHAR(l),
5)     dataUrodzenia DATE);
```

l) CREATE TABLE GwiazdaFilmowa (

2) nazwisko CHAR(30) ,

3) adres VARCHAR(255),

4) płeć CHAR (1),

5) dataUrodzenia DATE,

6) PRIMARY KEY(nazwisko))

7) PRIMARY KEY (tytuł, rok)

Jeszcze inaczej można określać klucz, korzystając ze słowa kluczowego **UNIQUE**. W instrukcji **CREATE TABLE** może wystąpić wiele specyfikacji **UNIQUE**, natomiast tylko jedna **PRIMARY KEY**.

2) nazwisko CHAR(30) UNIQUE

3) adres VARCHAR(255) UNIQUE,

6) UNIQUE (nazwisko)

## Integralność referencyjna i klucze obce

### Deklarowanie więzów klucza obcego

1. trybuty klucza obcego muszą być zadeklarowane jako klucz główny w oryginalnej relacji.
2. Każda wartość występująca jako atrybut klucza obcego musi wystąpić również jako wartość odpowiedniego atrybutu w oryginalnej relacji. Oznacza to istnienie więzów integralności referencyjnej łączących dwa atrybuty lub ich zbiory.

**REFERENCES** <tabela> (<atrybut>)

FOREIGN KEY <atrybuty> REFERENCES <tabela> (<atrybuty>)

Studio (nazwa, adres, prezC#)

FilmDyr (nazwisko, adres, cert#, cenaSieci)

Odpowiedniość między atrybutami **prezC#** i **certC#** można deklarować bezpośrednio w sposób następujący:

```
CREATE TABLE Studio (
nazwa CHAR( 30) PRIMARY KEY,
adres VARCHAR(255),
prezC# INT REFERENCES FilmDyr(cert#)
);
```

```
CREATE TABLE Studio (
nazwa CHAR (30) PRIMARY KEY ,
adres VARCHAR(255),
prezC# INT,
FOREIGN KEY prezC# REFERENCES FilmDyr(cert#)
);
```

Klucze główne i atrybuty różnowartościowe

Deklaracja **PRIMARY KEY** jest nieomal synonimem deklaracji **UNIQUE**. Najważniejsza różnica polega na tym, że jest jeden tylko klucz główny dla całej tabeli, a atrybutów różnowartościowych typu **UNIQUE** może być wiele.

**Klucz obcy może wskazywać wyłącznie na klucz główny relacji.**

2. **Procedura kaskadowa**

***3.*** *Procedura wstawiania wartości NULL*

Przypisaniu atrybutowi **prezC#** wartości **NULL** w powiązanej krotce relacji

Studio. Takie postępowanie nazywa się wstaw-null (*set-null*).

Tę opcję można określić oddzielnie dla usuwania i zmian, przyjmując właściwy sposób deklarowania klucza obcego. Używamy w tym celu słów kluczowych **ON DELETE** lub **ON UPDATE,** po których wybieramy opcję **SET NULL** lub **CASCADE**.

Studio(nazwa, adres, prezC#)

FilmDyr (nazwisko, adres, cert#, cenaSieci)

```
1)      CREATE TABLE Studio (
2)          nazwa CHAR(30)PRIMARY KEY,
3)          adres VARCHAR(255),
            prezC# INT REFERENCES FilmDyr(cert#),
4)          ON DELETE SET NULL,
5)      ON UPDATE CASACADE
);
```

## Więzy wartości atrybutów

l) w definicji schematu relacji jako warunek nałożony na atrybut, albo jako

2) ograniczenie dziedziny, która zostaje następnie określona jako dziedzina atrybutu.

### Więzy NOT-NULL

prezC# INT REFERENCES FilmDyr(cert#) NOT NULL

### Więzy CHECK

Nasze więzy polegają na tym, że numery certyfikatów muszą składać się co najmniej z sześciu cyfr. Deklaracja schematu relacji

Studio (nazwisko, adres, prezC#)

prezC# INT REFERENCES FilmDyr (cert#) CHECK (prezC# >= 100000)

płeć CHAR(1) CHECK (płeć IN ('K', 'M') ) ,

PRZYKŁAD

Studio (nazwisko, adres, prezC#)

FilmDyr (nazwisko, adres, cert#, cenaSieci)

prezC# INT CHECK (prezC# IN (SELECT cert# FROM FilmDyr))

Powyższa deklaracja jest prawidłowo zapisanym warunkiem typu CHECK, a w jego konsekwencji wystąpią następujące sytuacje:

- Próba wstawienia nowej krotki do relacji Studio, w której wartość atrybutu prezC# nie jest numerem certyfikatu żadnego prezesa, nie powiedzie się.
- Próba zmodyfikowania wartości atrybutu prezC# relacji Studio, w której nowa wartość atrybutu prezC# nie występuje jako wartość atrybutu cert# w relacji FilmDyr, nie powiedzie się.
- Jednakże, jeśli ulegnie zmianie relacja FilmDyr, np. w ten sposób, że zostanie z niej usunięta jakaś krotka opisująca pewnego prezesa, to pozostanie ona niezauważona przez zadeklarowane więzy CHECK.

A więc usunięcie zostanie wykonane, nawet jeśli w jego wyniku więzy CHECK zostaną naruszone.

**Więzy dziedziny**

CREATE DOMAIN DziedzinaPłci CHAR(1) CHECK (VALUE IN ('K', 'M' )) ;

Po zapisaniu takiej deklaracji wiersz 4)

4) płeć DziedzinaPłci,

CREATE DOMAIN DziedzinaCert INT CHECK (VALUE >= 100000);

4) prezC# DziedzinaCert REFERENCES FilmDyr (cert#)

## Więzy globalne

1. *Krotkowe więzy* CHECK nakładają ograniczenia na krotki pojedynczych relacji.
2. *Asercje* są to więzy nakładane jednocześnie na kilka relacji lub na zmienne krotkowe z określonej jednej relacji.

## Asercje

1) słowa kluczowego CREATE ASSERTION,
2) nazwy asercji,
3) słowa kluczowego CHECK,
4) warunku ujętego w nawiasy.

A więc postać definicji asercji można ująć w następujący schemat:

CREATE ASSERTION <nazwa> CHECK (<warunek>)

Warunek zapisany w asercji musi być zawsze spełniony i każda modyfikacja, która zaburza ten stan, nie jest wykonywana. Przypomnijmy, że inne więzy CHECK, jeśli tylko zawierały podzapytania, mogły być naruszane w pewnych szczególnych okolicznościach.

Przykład
Załóżmy, że nikt nie może zostać prezesem studia, o ile jego sieć jest warta mniej niż 10 000 000 $. Warunek asercji określimy w ten sposób, że będzie spełniony, jeśli zbiór studiów filmowych, których prezesi mają sieci warte mniej niż 10 000 000 $,jest pusty. Asercja dotyczy dwóch relacji:

FilmDyr(nazwisko, adres, cert#, cenaSieci)

Studio(nazwa, adres, prezC#)

```
CREATE ASSERTION BogatyPrez CHECK
(NOT EXISTS
(SELECT * FROM Studio, FilmDyr
WHERE prezC# = cert# AND cenaSieci < 10000000
)
```

```
) ;
```

Asercja zapewniająca, że prezesi studiów są bogaci

Mimo że asercja obejmuje dwie relacje, to można takie więzy wyrazić również za pomocą więzów krotkowych typu CHECK, zdefiniowanych osobno dla poszczególnych relacji.

```
CREATE TABLE Studio (
        nazwa CHAR (30)  PRIMARY KEY,
        adres VARCHAR(255),
4)          prezC# INT REFERENCES FilmDyr(cert#),
5)          CHECK (prezC#   NOT   IN
6)              (SELECT cert# FROM FilmDyr
7)              WHERE cenaSieci < 10000000)
)
);
```

Więzy w relacji studio, równoważne asercji

Więzy przedstawione sprawdzane jedynie wtedy, kiedy zajdzie zmiana dotycząca relacji Studio. Wobec tego nie zostanie na przykład wykryta sytuacja polegająca na tym, że wartość sieci pewnego prezesa zapisana w relacji FilmDyr spada poniżej 10 000 000 $. Aby uzyskać kompletny efekt działania asercji, trzeba by dołączyć do deklaracji tabeli FilmDyr jeszcze jedne więzy, które sprawdzają, czy jeśli dyrektor jest prezesem studia, to wartość jego sieci nie jest niższa niż 10 000 000 S. a

PRZYKŁAD

Przykład asercji, która dotyczy wyłącznie relacji

Film(tytuł, rok, długość, czyKolor, nazwaStudia, prcducentC#)

i określa, że całkowita długość wszystkich filmów w danym studiu nie przekracza 10 000 minut

CREATE ASSERTION SumDługość CHECK (10000 >= ALL

(SELSCT SUM (długość)FROM Movie GROUP BY nazwaStudia) ;

Tak się składa, że więzy dotyczą tylko relacji Film. Można wyrazić je także za pomocą więzów krotkowych typu CHECK w schemacie relacji Film. Definicja tych więzów polegałaby na dołączeniu do schematu relacji Film następującej deklaracji:

CHECK (10000 >= ALL (SELECT SUM(długość) FROM Film GROUP BY nazwaStudia)) ;

Zauważmy, że w zasadzie ten warunek stosuje się do każdej krotki relacji Film. Jednakże nie wspomina się tu o żadnym atrybucie i przetwarzanie jest skupione w podzapytaniu.

**Porównanie Więzów**

W tabeli przedstawionej poniżej zilustrowano różnice pomiędzy więzami atrybutowa więzami kotkowymi i asercjami.

| *Typ więzów* | *Gdzie są deklarowane* | *Kiedy uruchamiane* |
|---|---|---|
| Atrybutowe CHECK | Z atrybutami | Przy wstawianiu do relacji lub zmianie wartości atrybutu |
| Krotkowe CHECK | Element schematu relacji | Przy wstawianiu do relacji lub zmianie wartości w krotce |
| Asercje | Element schematu bazy danych | Przy modyfikacji w relacji |

# Wyzwalacze w języku SQL3

## Wyzwalacze a więzy

*Wyzwalacze*, czasami nazywane regułami *zdarzenie-warunek-akcja* (ECA *rules*, czyli *event-condition-action*), różnią się od omawianych poprzednio więzów na trzy sposoby:

1. Wyzwalacze są testowane tylko przy zajściu określonego przez programistę zdarzenia: dołączanie, modyfikacje lub usuwanie krotek danej relacji. W niektórych implementacjach SQL występuje również inny rodzaj zdarzeń: zakończenie transakcji.

2. Wyzwalacze testują *warunek* w chwili zajścia zdarzenia, a nie uprzedzając je. Jeśli warunek nie jest spełniony, to w odpowiedzi na zdarzenie nic się nie wykona.

3. Jeśli warunek wyzwalacza zostanie spełniony, to DBMS przetwarza *akcją* związaną z wyzwalaczem. Akcja może chronić przed zajściem zdarzenia w bazie lub może zmienić wynik zdarzenia (np. usunąć w prowadzoną krotkę). Akcja może polegać na przetworzeniu całego ciągu operacji w bazie danych, nawet takich, które nie mają żadnego związku z wyzwalanym zdarzeniem.

Podstawowe właściwości wyzwalaczy:

1. Akcja może być wykonana przed, po i w chwili zajścia zdarzenia.

2. Akcja może korzystać zarówno z wartości sprzed zajścia zdarzenia, jak i z nowych wartości powstałych w wyniku wstawienia, modyfikacji lub usunięcia krotki w trakcie zdarzenia.

3. Zdarzenia mogą wprowadzać modyfikacje do określonej kolumny lub do zbioru kolumn.

4. Warunek można określać w klauzuli WHEN i akcja jest wykonywana przy wyzwoleniu reguły oraz warunek jest spełniony, gdy zajdzie wyzwalane zdarzenie.

5. Programista może sam określić, czy akcja ma być wykonana:

a) zawsze dla każdej modyfikowanej krotki,
b) raz dla wszystkich modyfikowanych krotek w pojedynczej operacji w bazie danych.

PRZYKŁAD

FilmDyr (nazwisko, adres, cert# cenaSieci)

Akcja jest wyzwalana przy próbie modyfikacji atrybutu cenaSieci. W wyniku powinna zostać uniemożliwiona każda próba obniżenia ceny sieci prezesa studia.

Wiersz l) deklaracja złożona ze słowa kluczowego **CREATE TRIGGER** oraz nazwy Wiersz 2) określa się wyzwalane zdarzenie, którym jest modyfikacja wartości atrybutu **cenaSieci** w relacji **FilmDyr**.

Wiersze 3) - 5) - elementy warunków i akcji wyzwalacza

Wiersz 6) - warunek wyzwalania.

Wiersze 7) - 9) tworzą element działania; są to zwykłe instrukcje modyfikacji w SQL, a mają na celu odtworzenie ceny sieci sprzed modyfikacji.

Wiersz 10) precyzuje się wymaganie użycia wyzwalacza dla każdej modyfikowanej krotki. Jeśliby go nie było, to wyzwalacz byłby zastosowany tylko raz podczas przetwarzania instrukcji SQL, bez względu na to, jak wiele razy zaszłoby zdarzenie wchodzące w skład wyzwalacza.

```
1)  CREATE TRIGGER CenaSieciWyzw
2)  AFTER UPDATE OF cenaSieci  ON  FilmDyr
3)  REFERENCING
4)      OLD AS StaraKrotka,
5)      NEW AS NowaKrotka
6)  WHEN (StaraKrotka.cenaSieci > NowaKrotka.cenaSieci)
7)      UPDATE FilmDyr
8)   SET cenaSieci = StaraKrotka.cenaSieci
```

9) **Where** cert# = NowaKrotka.cert#

10) FOR EACH ROW ;

1. **BEFORE**. Warunek **WHEN** sprawdza się przed zajściem zdarzenia. Akcje wyzwalacza zostają przetworzone, o ile warunek jest spełniony. Wówczas zdarzenie, które powoduje modyfikacje, zachodzi, bez względu na to, czy warunek jest spełniony, czy nie.
2. **INSTEAD OF.** Akcja jest wykonywana (przy spełnieniu warunku **WHEN**), ale zdarzenie nigdy nie następuje.
3. **AFTER**. Akcje są przetwarzane po zajściu zdarzenia wyzwalania.

**UPDATE (OF)** **INSERT**

**DELETE**.

**OLD AS** i **NEW AS**. **NEW AS**
**OLD AS**

Jeśli zdarzenie polega na modyfikacji, to są z nim związane dwie krotki, stara i nowa, odpowiednio sprzed modyfikacji i po modyfikacji. Krotkom tym nadaje się nazwy w klauzulach **OLD AS** i **NEW AS**.

Jeśli zdarzenie polega na dopisaniu krotki, to można użyć klauzuli **NEW AS** do nazwania wstawianej krotki, klauzula **OLD AS** nie jest dopuszczalna w takiej sytuacji.

Odwrotnie postępujemy w przypadku usuwania krotek, wówczas korzystamy z klauzuli **OLD AS** do nazwania usuwanej krotki, natomiast klauzuli **NEW AS** nie stosuje się.

Jeśli opuścimy opcję **FOR EACH ROW** to wyzwalacz *poziomu-wierszy* (*row-level trigger*) stanie się *wyzwalaczem poziomu-instrukcji* (*statement-level trigger*).

**Wyzwalacz poziomu-instrukcji** wykonuje się jeden raz przy instrukcji powodującej jedno lub więcej zdarzeń wyzwalacza. Na przykład, jeśli instrukcja

SQL polega na modyfikacji całej tabeli, to wyzwalacz poziomu-instrukcji wykona się tylko jeden raz, a wyzwalacz poziomu-wierszy wykona się przy modyfikacji każdej krotki.

W przypadku wyzwalaczy poziomu-instrukcji można mówić o zbiorze starych krotek (krotkach usuniętych lub starych wersjach krotek zmodyfikowanych) i o zbiorze nowych krotek (krotek wstawianych lub nowych wersji krotek modyfikowanych) jako o dwóch relacjach. Stosujemy zatem deklaracje \

**OLD_TABLE AS** TeStare **NEW_TABLE AS TeNowe.**

gdzie TeStare oznaczają nazwę relacji zawierającej wszystkie stare krotki.
TeNowe oznacza nazwę relacji zawierającą nowe krotki.

PRZYKŁAD

Załóżmy, że należy ochronić wartość sieci prezesów od spadku poniżej 500 000 S. Takie ograniczenie może zostać naruszone przez każdą operacje: wstawianie, usuwanie oraz modyfikacji w kolumnie cenaSieci relacji:

FilmDyr (nazwisko, adres, cert#, cenaSieci)

Przypadek modyfikacji.

Wiersze 3) - 5) określono, że TeNowe i TeStare są nazwami relacji zawierających odpowiednio stare i nowe krotki uzyskane w wyniku operacji wyzwalania na bazie danych. Trzeba zauważyć, że jedna instrukcja SQL może spowodować modyfikacje szeregu krotek relacji i wówczas TeNowe i TeStare będą zawierać wiele krotek.

1) **CREATE TRIGGER** WyzwalaczŚrCenySieci
2) **INSTEAD OF UPDATE OF** cenaSieci **ON** FilmDyr

REFERENCING\

4) OLD_TABLE AS TeStare
5) **NEW_TABLE AS** TeNowe
6) **WHEN** (500000<=

```
7 )    (SELECT AVG(cenaSieci)
8)      FROM (( FilmDyr EXCEPT TeStare) UNION TeNowe))
9)          DELETE FROM FilmDyr
10)         WHERE (nazwisko, adres, cert#, cenaSieci)  IN TeStare;
11)         INSERT INTO FilmDyr
12)         (SELECT * FROM TeNowe) ;
```

Wyzwalacz dla wartości średniej ceny sieci

Gdy operacja polega na modyfikacji, to **TeStare i TeNowe** zawierają odpowiednio nowe i stare wersje modyfikowanych krotek.

Wiersze 6) - 8) warunek. Jest on spełniony, jeżeli średnia wartość ceny sieci po modyfikacji wynosi co najmniej 500 000 S.

Wiersz 8) opisuje, jaka będzie wartość relacji FilmDyr w przypadku wykonania modyfikacji.

Jednakże, ponieważ w wierszu 2) występuje klauzula INSTEAD OF, wiec żadna próba wykonania modyfikacji w kolumnie cenaSieci nie powiedzie się. Modyfikacja nie dojdzie nigdy do skutku. Zamiast tego warunek zawarty w wyzwalaczu rozstrzyga, jakie akcje zostaną przetworzone. W naszym przykładzie, jeśli modyfikacja nie powoduje obniżenia wartości sieci poniżej 500 000 $, to w wyniku działania wyzwalacza modyfikacja zostanie wprowadzona do tabeli.

Wiersze 9) i 10) zapisano instrukcje usunięcia tych wierszy, które zostałyby wprowadzone w wyniku modyfikowania relacji.

Wiersze 11) i 12) wstawiania nowych wersji tych krotek.

```
1)      CREATE TRIGGER WyzwalaczŚrCenySieci
2)      INSTEAD OF INSERT FilmDyr
3)      REFERENCING
4)          NEW_TABLE AS TeNowe
5)      WHEN (500000<=
```

```
6 )   (SELECT AVG(cenaSieci)
8)     FROM (( FilmDyr UNION TeNowe))
9)              INSERT INTO FilmDyr
10)             (SELECT * FROM TeNowe) ;
```

```
1)        CREATE TRIGGER WyzwalaczŚrCenySieci
2)        INSTEAD OF DELETE FilmDyr
3)        REFERENCING
4)              OLD_TABLE AS TeStare
5)        WHEN (500000<=
6 )   (SELECT AVG(cenaSieci)
8)     FROM (( FilmDyr EXPECT TeStare))
9)              DELETE INTO FilmDyr
10)             (SELECT * FROM TeStare) ;
```

## Asercje w języku SQL3

Zakres stosowania asercji w SQL3 jest rozszerzony w stosunku do SQL2 w następujący sposób:

1. W przypadku naruszenia więzów asercje są wyzwalane przez zdarzenia, które określa programista, a nie system.
2. Asercja może obejmować poszczególne krotki, nie tylko całą tabelę.

PRZYKŁAD

Asercje **BogatyPrezes**.
W wierszach od 2) do 6) opisano zdarzenia, które mogą wyzwalać poszczególne asercje.

```
1)        CREATE ASSERTION BogatyPrezes
2)        AFTER
```

```
3)          INSERT ON Studio,
4)          UPDATE OF prezC# ON STUDIO,
5)          UPDATE OF cenaSieci ON FilmDyr,
6)          DELETE ON FilmDyr
7)      CHECK (NOT EXISTS
8)          (SELECT *  FROM Studio, FilmDyr
          WHERE prezC# = cert# AND cenaSieci <10 000 000
)
          )
```

Zasadnicza różnica między koncepcją asercji w SQL2 i SQL3 polega na tym, ze w SQL3 jawnie wpisuje się, w jakich sytuacjach mają być sprawdzane warunki. W SQL3 jest zatem łatwiej zaimplementować asercje, ale z kolei jest trudniej z nich korzystać. Użytkownik musi bowiem:

1. przewidzieć sytuacje, w których będą wyzwalane więzy;
2. ponosić ryzyko pozostawienia bazy w stanie niespójnym, jeśli zdarzenia zostały dobrane w niewłaściwy sposób.

**Podsumowanie**

- ***Więzy klucza*** (*key constraints*): Słowa kluczowe **PRIMARY KEY** lub **UNIQUE** określają ,w deklaracji schematu bazy danych ten atrybut lub zbiór atrybutów, które stanowią klucz relacji.
- ***Więzy integralności referencyjnej*** (*referential integrity constraints*): Korzystając ze słowa kluczowego **REFERENCES** lub **FOREIGN KEY**, można zadeklarować, że pewna wartość (lub zbiór wartości) z jednej relacji występuje jako wartość (wartości) klucza głównego pewnej krotki innej relacji.
- ***Więzy typu CHECK dla wartości***: Można sprawdzać warunki określane dla wartości atrybutów przez umieszczenie w definicji schematu relacji przy tym atrybucie słowa **CHECK** oraz warunku, który ma być sprawdzany.

**Drugi sposób sprawdzana warunkowa dla wartości atrybutów polega na** wpisaniu tego warunku w definicję dziedziny atrybutu.

- ***Więzy krotkowe typu CHECK***: Sprawdzenie, że wartości składowej lub składowych krotek pewnej relacji zawsze spełniają określone warunki,

może następować w wyniku dołączenia do deklaracji relacji słowa kluczowego **CHECK** oraz warunku, który ma być przestrzegany.

- ***Asercje* (*assertions*):** Do schematu bazy danych można dołączyć asercje jako element schematu, korzystając ze słowa kluczowego **CHECK** oraz określenia warunku. W warunku może występować więcej niż jedna nazwa relacji ze schematu bazy danych, a także asercja może dotyczyć relacji jako całości, jeśli warunek dotyczy np. pewnego agregatu.
- ***Wyzwalacze* (*triggers*) *w SQL3***: W standardzie SQL3 zostały zawarte wyzwalacze, które mogą zawierać określenie zdarzeń (np. wstawień, usunięć lub zmian w określonej relacji) powodujących wykonanie sprawdzenia.
- ***Asercje w SQL3*:** Pojecie asercji w SQL3 różni się od koncepcji z SQL2. Podobnie jak wyzwalacze w SQL3, tak i asercje mogą, działać pod wpływem zajścia określonych zdarzeń, np. wstawienia krotki do relacji. Po uruchomieniu asercji jest sprawdzany warunek dotyczący albo relacji, albo krotek, i jeśli nie jest on spełniony, to nie dochodzi do wykonania modyfikacji.