**Utworzenie nowej tabeli**

CREATE TABLE nazwa\_tabeli( Deptno INT PRIMARY KEY, DNAME varchar(14) NOT NULL)

**CONSTRAINT** wprowadza nazwę dla więzów spójności

**Wpisywanie do tabeli**

INSERT INTO nazwatabeli (at1, at2,at3) VALUES( 1,2,’trzy’);

**Dodanie Kolumny**

ALTER TABLE nazwatabeli ADD nazwakolumny typ;

**Modyfikacja Kolumny**

ALTER TABLE nazwatabeli ALTER COLUMN probaa varchar(10);

**Usuniecie kolumny**

ALTER TABLE nazwatabeli DROP COLUMN nazwa kolumny;

**Usuniecie tabeli**

DROP TABLE nazwatabeli;

**Modyfikacja Tabeli**

UPDATE nazwatabeli

SET probne=’okej’

WHERE probne=’tak’

**Usuniecie Wiersza**

DELETE nazwatabeli

WHERE probne=’okej’;

**@@Version** - zwraca informację o wersji używanego serwera bazy danych,

**@@Identity** – ostatnio wygenerowana wartość w autonumerowanej kolumnie,

**@@Error –** numer ostatniego błędu,

**@@Rowcount** – liczba rekordów, na których operowała ostatnia instrukcja SQL; w przypadku SELECT – liczba zwracanych przez instrukcję rekordów,

**@@Fetch\_status** – zwraca informację o podstawieniu rekordu w ostatniej instrukcji FETCH kursora (0 sukces, rekord podstawiony, -1 porażka, rekord niepodstawiony

**Deklarowanie zmiennej:**

DECLARE @zmienna int= (SELECT COUNT(\*) FROM EMP) lub DECLARE @nazwisko Varchar(30), @imie Varchar(20), @Data\_urodzenia Date;

DECLARE @Cena Money = 1500 ,@Kupujacy Varchar(30) = 'PJWSTK' ,@Data\_sprzedazy Date = **Getdate();**

**Getdate()- weźmie dzisiejszą datę, ale musi być przypisane w DECLARE**

**Wypisywanie informacji:**

PRINT ‘W Tabeli jest’ + ‘ cast(@zmienna’ AS varchar)+ ‘liczba osób’;

Należy pamiętać o konieczności dokonywania jawnej konwersji danych na typ napisowy przy konkatenacji **(funkcje Cast i Convert).**

**SET pozwala podstawić wartość pod zmienną :**

SET @DataRekrutacji = GETDATE();

**PRIMARY KEY**- klucz główny

**FOREIGN KEY**- klucz obcy

**IDENTITY(1,1**) – auto numeracja co 1 od 1

**NOT NULL**- wartość nie może być nigdy null

**UNIQUE** – Unikatowa wartość w kolumnie.

**CONSTRAINTS** (ograniczenia) to reguły które określają jakie dane mogą się znajdować w tabelach (kolumnach).

CREATE TABLE NAZWA\_TABELI

(NAZWA\_KOLUMNY TYP\_DANYCH CONSTRAINT TWOJA\_NAZWA\_CONSTRAINTU WYBRANY\_CONSTRAINT WŁAŚCIWOŚCI/OPCJE CONSTRAINTU);

**REFERENCES**- Określa ona kolumnę jako klucz obcy. Klucz obcy zawiera odwołanie do innej tabeli przez umieszczenie w deklarowanej kolumnie wartości z klucza głównego do tamtej tabeli. Oznacza to, że dozwolone są tylko takie wartości, które występują jako klucze w tabeli, do której się odwołujemy. Takie ograniczenie jest czasem nazywane więzami integralności referencyjnej

CREATE TABLE EGZAMIN (idEgzamin int not null IDENTITY (1,1) PRIMARY KEY ,

idPrzedmiot int FOREIGN KEY **REFERENCES** PRZEDMIOT,

DataEgzaminu DATETIME)

CREATE TABLE PRZEDMIOT(idPrzedmiot int IDENTITY (1,1) PRIMARY KEY ,

Nazwa VARCHAR(10))

CREATE TABLE OSOBA(IdOsoba int IDENTITY (1,1),

Imie VARCHAR(15) not null,

Nazwisko VARCHAR(15) not null,

**CONSTRAINT** Osoba\_pk PRIMARY KEY (IdOsoba)

CREATE TABLE ZAPISY(IdEgzamin int NOT NULL ,

IdOsoba int NOT NULL,

CONSTRAINT zapisy\_pk PRIMARY KEY(IdEgzamin,IdOsoba))

ALTER TABLE Stypendium ADD CONSTRAINT Stypendium\_Student

FOREIGN KEY (IdOsoba)

REFERENCES Student (IdOsoba);

**Możemy używać operatorów**

MIN/ SUM/ MAX/ AVG

**SYSDATETIME() –** ukazanie całej daty wraz z godziną.

MOTH(data)/ DAY(data) / YEAR(data)/ MINUTE(data)

**Jeśli chcemy aby zamiast nulla ( data zmienić na stringa) COALESCE(CONVERT(varchar),dataod), ‘brak’) lub COALESCE(data, 0) (czyli jak data będzie null wstaw 0)**

**ISNULL(Max(empno), 0)**

---PROCEDURY

*DECLARE @zmienna int ; SELECT @int = MIN(sal) FROM EMP WHERE deptno=@deptno;*

**CREATE PROCEDURE** – jeśli chcemy stworzyć

**ALTER PROCEDURE** – jeśli chcemy zmodyfikować, tą która już jest

**EXEC/EXECUTE** nazwa\_procedury {zmienne}; - uruchomienie naszej procedury.

**CREATE PROCEDURE** nazwa\_procedury @nrdzialu int {…zmienne}

**AS BEGIN** – rozpoczecie procedury

Możemy użyć **IF/ELSE** po których zawsze jest **BEGIN/END**

**EXISTS/NOT EXISTS** czyli prawda/fałsz

**IF [NOT] EXISTS** (**dowolna instrukcja SELECT**) Jeżeli instrukcja SELECT zwraca cokolwiek (Nawet NULL), cała instrukcja zwraca TRUE.

IF EXISTS (SELECT 1 FROM EMP WHERE Deptno IS NULL)

PRINT 'Istnieje pracownik nie przydzielony do działu.’

**\*Przykład\***

**CREATE PROCEDURE nauk2 @nazwisko varchar(20), @nrdzialu int**

AS BEGIN

IF EXISTS ( SELECT \* FROM DEPT WHERE DEPTNO=@nrdzialu)

**BEGIN**

DECLARE @pensja int = (SELECT MIN(SAL) FROM EMP WHERE DEPTNO=@nrdzialu GROUP BY deptno)

INSERT INTO EMP ( empno, ename, sal, deptno)

SELECT MAX(empno)+1, @nazwisko, @pensja, @nrdzialu FROM EMP;

PRINT ‘Wstawiono ‘+@nazwisko + ‘ do ‘+@nrdzialu + ‘ dzialu ‘;

**END**

ELSE

**BEGIN**

PRINT ‘Niestety nie ma takiego nrdzialu’;

**END**

END

\*CREATE PROCEDURE Ile\_pracownikow

\*AS BEGIN

\*DECLARE @ile = COUNT(1)

\*FROM Emp;

**\*RETURN @ile;**

\*END;

Pętla **WHILE** posiada następującą składnię**: WHILE warunek** Ciąg instrukcji Podobnie jak w przypadku instrukcji IF ciąg instrukcji powinien być poprzedzony słowem **BEGIN** i zakończony słowem **END.** Wymóg ten może zostać pominięty, jeśli w pętli ma być wykonywana tylko jedna instrukcja.

----KURSORY

**@@Cursor\_rows** - przechowuje liczbę rekordów odczytanych z bazy przez ostatnio otwarty kursor

**DECLARE** nazwa\_kursora **CURSOR FOR** instrukcja\_SELECT

**OPEN** nazwa\_kursora / **CLOSE** nazwa\_kursora – zawsze musimy zamknąć kursor

**DEALLOCATE**- usunięcie na koniec kursora

Instrukcja która odwołuje się do kolejnego wiersza i dokonuje podstawienia odczytanych wartości na wcześniej zadeklarowane zmienne ma postać **FETCH NEXT FROM nazwa\_kursora INTO zmienne**

WHILE @@Fetch\_status = 0 będzie wykonywana do momentu odczytania ostatniego wiersza. Wtedy @@Fetch\_status przyjmie wartość – 1 i pętla się zakończy.

Przy każdym przejściu pętli trzeba uwzględnić FETCH STATUS

**FETCH NEXT FROM nazwakursora INTO @empno, reszta zmiennych**

FETCH NEXT – następny wiersz

FETCH PRIOR – poprzedni wiersz

FETCH FIRST – pierwszy wiersz

FETCH LAST – ostatni wiersz

FETCH ABSOLUTE(n) – wiersz o numerze n

FETCH RELATIVE(n) – wiersz o n wierszy dalej niż bieżący, lub bliżej, jeśli n < 0)

**\*Przykład\***

DECLARE kursor

CURSOR FOR SELECT empno, ename, sal FROM EMP WHERE sal<1000 OR sal>1500

DECLARE @empno int, @ename varchar(20), @sal DECIMAL(8,2)

OPEN kursor

FETCH NEXT FROM kursor INTO @empno, @ename, @sal

WHILE @@FETCH\_STATUS=0

BEGIN

IF @sal <1000

BEGIN

SET @sal=@sal1.1

UPDATE emp SET sal=@sal WHERE empno=@empno

PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);

END

ELSE IF @sal>1500

BEGIN

SET @sal=@sal0.9

UPDATE emp SET sal=@sal WHERE empno=@empno

PRINT 'Zwiększono pensje o 10% pracownikowi '+ @ename+' o '+ CAST(@sal as varchar);

END

FETCH NEXT FROM kursor INTO @empno,@ename,@sal

END

CLOSE kursor

DEALLOCATE kursor;

----WYZWALACZE

**Tworzymy go:**

CREATE TRIGGER NazwaWyzwalacza

ON tabela

FOR tutaj może być INSERT/UPDATE/DELETE

AS

ROLLBACK; czyli co ma wykonać w tym przypadku ROLLBACK- wycofa całą transakcję.

**ENABLE / DISABLE** TRIGGER nazwa\_wyzwalacza ON nazwa\_tabeli; --Włączenie lub wyłączenie

Wiersze, które zostały zmienione przez instrukcję umieszczane są w tabelach tymczasowych o nazwach **INSERTED i DELETED**. W przypadku instrukcji **INSERT** nowe wiersze znajdują się w tabeli **INSERTED**, w przypadku instrukcji **DELETE** usunięte wiersze znajdują się w tabeli **DELETED**, w przypadku instrukcji **UPDATE** wiersze w postaci sprzed zmiany umieszczane są w tabeli **DELETED**, w postaci po zmianie w tabeli **INSERTED**.

**\*Przykład\* Zablokowania operacji UPDATE/INSERT jeśli sal jest mniejsza od 100**

CREATE TRIGGER TR3

ON EMP

FOR INSERT, UPDATE

**AS BEGIN**

IF EXISTS (SELECT 1 FROM inserted WHERE Sal < 100)

BEGIN

ROLLBACK;

Raiserror ('Niedopuszczalna wartość SAL!', 1, 2);

END;

**END;**

