- Seminar 3 -
Proceduri stocate

- exemplu de procedură care adaugă o constrângere de **valoare implicită** pt. coloana vârstă_min din tabelul Atracții:

```
CREATE PROCEDURE AdaugăConstrângereDefault
AS
BEGIN
   ALTER TABLE Atracții
   ADD CONSTRAINT df_vârstă_min DEFAULT 12 FOR vârstă_min;
END

EXEC AdaugăConstrângereDefault;
```

- următoarea procedură returnează numele, descrierea și vârsta minimă recomandată pt. toate înregistrările

```
CREATE PROCEDURE PotAtracțiiCuVârstăMin @vârstă_min INT   <- param. de intrare
AS
BEGIN
   SELECT nume, descriere, vârstă_min FROM Atracții
   WHERE vârstă_min = @vârstă_min
END;

EXEC PotAtracțiiCuVârstăMin 12;
```

```
ALTER PROCEDURE PotAtracțiiCuVârstăMin @vârstă_min INT,
@nr_atracții INT OUTPUT
AS
BEGIN
   SELECT @nr_atracții = COUNT(*) FROM Atracții WHERE vârstă_min = @vârstă_min;
END
```

- cu să o apelez, foc următorii pași:
  - o declar o variabilă locală
    `DECLARE @nratracții AS INT;`
  - o inițializează variabila locală
    `SET @nratracții = 0;`
  - o apelează procedura stocată
    `EXEC PotAtracțiiCuVârstăMin 12, @nr_atracții = @nratracții OUTPUT;`
  - o afișează pe ecran valoarea parametrului de ieșire
    `PRINT @nratracții;`

! Putem să dăm RAISERROR ca să generăm mesaj de eroare.
`RAISERROR('Nu a fost returnată nicio atracție', 16, 1);`

- ștergem procedurile cu DROP: `DROP dbo.PotAtracțiiCuVârstăMin;`

- putem avea variabile globale, care nu trebuie declarate (fiind fct. de sistem)
  ↳ numele lor începe cu @@

Exemple:
- @@ERROR → numărul celei mai recente erori
- @@IDENTITY → valoarea câmpului IDENTITY al ultimei înreg. inserate
- @@ROWCOUNT → conține nr. de înregistrări afectate de cea mai recentă instrucțiune
- @@SERVERNAME → numele instanței
- @@SPID → ID-ul de sesiune al procesului de utilizator curent
- @@VERSION → conține inf. în legătură cu sistemul și compilarea curentă a serverului

---

- SET NOCOUNT
  - SET NOCOUNT ON - oprește returnarea mesajului cu nr. de înreg. afectate de către ultima instrucțiune efectuată
  - SET NOCOUNT OFF - mesajul va fi returnat ca parte a result set-ului

---

Limbaj de control al fluxului

- BEGIN... END → delimitează un grup de instrucțiuni SQL care se execută împreună
  ↳ blocurile acestea pot fi încorporate
- RETURN iese automat din interogare / procedură stocată
  ↳ îl folosim ca să returnăm status codes → procedurile stocate ret. zero (succes) sau o val. întreagă dif. de zero (failure)

Exemplu:
```
CREATE PROCEDURE Verifică_Vârstă_Client @cod_a INT
AS
BEGIN
  IF ((SELECT vârstă_min FROM Atracții WHERE cod_a=@cod_a) =12)
      RETURN 1;
  ELSE
      RETURN 2;
END;
```

```
DECLARE @status INT;
EXEC @status = Verifică_Vârstă_Client 1;
SELECT 'Status' = @status;
```

- WHILE atașează o condiție pt. execuția repetată a unei instrucțiuni SQL sau a unui bloc de instrucțiuni;
- BREAK iese din cea mai interioară buclă WHILE sau IF...ELSE din int. buclei WHILE
- CONTINUE cauzează reîncepera buclei WHILE și ignoră instrucțiunile după CONTINUE
- THROW aruncă o excepție și transferă execuția unui bloc CATCH dintr-o construcție TRY... CATCH

Exemplu:
```
THROW 50002, N'Înreg. nu există!', 1;
BEGIN TRY
...
END TRY
BEGIN CATCH
...
END CATCH
```

– în interiorul unui bloc CATCH pot fi folosite următoarele f. sistem:

- ERROR_NUMBER() – ret. nr. erorii
- ERROR_SEVERITY() – ret. severitatea erorii
- ERROR_STATE() – ret. eroarea state number
- ERROR_PROCEDURE() – ret. numele procedurii stocate sau al trigger-ului în care a avut loc eroarea
- ERROR_LINE() – ret. nr. liniei care a cauzat eroarea
- ERROR_MESSAGE() – ret. mesajul erorii