











## Les Triggers et les procédures stockées :

La base de données dispose de deux trigger permettant de gérer certaines actions automatiquement de façon à rendre la base de données dynamique. Elle dispose aussi de procédures stockées qui vont principalement permettre au logiciel d'interagir avec la base de données.

Les procédures stockées permettant au logiciel d'interagir avec la base de données :

- +  dbo.prc\_ajout\_medicament
- +  dbo.prc\_getDesisions
- +  dbo.prc\_getEtapas
- +  dbo.prc\_getFamilles
- +  dbo.prc\_getHistoModifEtpNormee
- +  dbo.prc\_getMedicaments
- +  dbo.prc\_getUtilisateurs
- +  dbo.prc\_getWorkflows
- +  dbo.prc\_maj\_etapenormee
- +  dbo.prc\_setDecisionEtape

Ces procédures permettent principalement de récupérer les données de chaque table (get), d'insérer un médicament (ajout\_medicament) et de mettre à jour des données (set et maj).

### - prc\_ajout\_medicament

```
USE [GSB_gesAMM]
GO
/***** Object:  StoredProcedure [dbo].[prc_ajout_medicament]    Script Date: 06/01/2023 17:14:03 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_ajout_medicament] (@depotlegale as varchar(100), @nomcommercial as varchar(100), @famcode as varchar(100), @composition as varchar(100), @effets as varchar(100), @contreindication as varchar(100), @prixchantillon as decimal(10,2))
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO MEDICAMENT
    VALUES(@depotlegale,
    @nomcommercial,
    @famcode,
    @composition,
    @effets,
    @contreindication,
    @prixchantillon)
END
```

- prc\_getDecisions :

```
USE [GSB_gesAMM]
GO
/***** Object:  StoredProcedure [dbo].[prc_getDecisions]    Script Date: 06/01/2023 17:15:20 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getDecisions]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT DCS_ID, DCS_LIBELLE
    FROM [dbo].[DESISION]
END
```

- prc\_getEtapas :

```
USE [GSB_gesAMM]
GO
/***** Object:  StoredProcedure [dbo].[prc_getEtapas]    Script Date: 06/01/2023 17:16:28 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getEtapas]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT ETP_NUM, ETP_LIBELLE, ETP_NORME, ETP_DATE_NORME
    FROM [dbo].[ETAPE]
END
```

- prc\_getFamilles :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_getFamilles]    Script Date: 06/01/2023 17:17:38 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getFamilles]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT FAM_CODE, FAM_LIBELLE, nbMedAutorisé
    FROM [dbo].[FAMILLE]
END
```

- prc\_getHistoModifEtpNormee :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_getHistoModifEtpNormee]    Script Date: 06/01/2023 17:18:23 ***/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getHistoModifEtpNormee]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    IF OBJECT_ID('ARCHIVE_ETAPENORME') IS NOT NULL
    BEGIN
        SELECT idModif, dateModif, etpLibelle, etpNorme, etpDateNorme
        FROM [dbo].[ARCHIVE_ETAPENORME]
    END
END
```

- prc\_getMedicaments :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_getMedicaments]    Script Date: 06/01/2023 17:19:53 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getMedicaments]
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

-- Insert statements for procedure here
SELECT MED_DEPOTLEGAL, MED_NOMCOMMERCIAL, FAM_CODE, MED_COMPOSITION, MED_EFFETS, MED_CONTREINDIC, MED_PRIXECHANTILLON
FROM [dbo].[MEDICAMENT]
END
```

- prc\_getUtilisateurs :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_getUtilisateurs]    Script Date: 06/01/2023 17:20:35 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getUtilisateurs]
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

-- Insert statements for procedure here
SELECT UTI_NOM_UTILISATEUR, UTI_MDP_UTILISATEUR, UTI_DRO_ID
FROM [dbo].[UTILISATEURS]
END
```

- prc\_getWorkflows :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_getWorkflows]    Script Date: 06/01/2023 17:21:34 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_getWorkflows]
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT idDecision, numEtape, dateDecision, MED_DL
    FROM [dbo].[WORKFLOW]
END
```

- prc\_maj\_etapenormee :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_maj_etapenormee]    Script Date: 06/01/2023 17:22:29 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_maj_etapenormee]
    @numEtape INTEGER,
    @etpNorme VARCHAR(20),
    @etpNormeDate DATETIME
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    IF (@etpNorme <> (SELECT ETP_NORME FROM [dbo].[ETAPE] WHERE ETP_NUM = @numEtape)) AND (@etpNormeDate <> (SELECT ETP_DATE_NORME FROM [dbo].[ETAPE] WHERE ETP_NUM = @numEtape))
    BEGIN
        UPDATE [dbo].[ETAPE]
        SET ETP_NORME = @etpNorme,
            ETP_DATE_NORME = @etpNormeDate
        WHERE ETP_NUM = @numEtape

        RETURN
    END

    IF @etpNorme <> (SELECT ETP_NORME FROM [dbo].[ETAPE] WHERE ETP_NUM = @numEtape)
    BEGIN
        UPDATE [dbo].[ETAPE]
        SET ETP_NORME = @etpNorme
        WHERE ETP_NUM = @numEtape

        RETURN
    END

    IF @etpNormeDate <> (SELECT ETP_DATE_NORME FROM [dbo].[ETAPE] WHERE ETP_NUM = @numEtape)
    BEGIN
        UPDATE [dbo].[ETAPE]
        SET ETP_DATE_NORME = @etpNormeDate
        WHERE ETP_NUM = @numEtape

        RETURN
    END
END
```

- prc\_setDecisionEtape :

```
USE [GSB_gesAMM]
GO
/***** Object: StoredProcedure [dbo].[prc_setDecisionEtape]    Script Date: 06/01/2023 17:23:02 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
ALTER PROCEDURE [dbo].[prc_setDecisionEtape]
    @numEtape integer,
    @medDL varchar (100),
    @idDec integer,
    @dateDec date
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    UPDATE [dbo].[WORKFLOW]
    SET idDecision = @idDec,
        dateDecision = @dateDec
    WHERE numEtape = @numEtape
        AND MED_DL = @medDL
END
```

- trg\_archv\_etapenormee :

```

ALTER TRIGGER [dbo].[trg_archv_etapenormee]
ON [dbo].[ETAPE]
INSTEAD OF UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    DECLARE @numEtape AS int
    DECLARE @etpNorme AS VARCHAR(20)
    DECLARE @etpNormeDate AS DATETIME

    DECLARE @etpLibelleAvantModif AS VARCHAR(50)
    DECLARE @etpNormeAvantModif AS VARCHAR(20)
    DECLARE @etpNormeDateAvantModif AS DATETIME

    SELECT @numEtape = ETP_NUM, @etpNorme = ETP_NORME, @etpNormeDate = ETP_DATE_NORME FROM inserted

    SELECT @etpLibelleAvantModif = ETP_LIBELLE, @etpNormeAvantModif = ETP_NORME, @etpNormeDateAvantModif = ETP_DATE_NORME
    FROM ETAPE
    WHERE ETP_NUM = @numEtape

    IF OBJECT_ID('ARCHIVE_ETAPENORME') IS NULL
    BEGIN
        CREATE TABLE ARCHIVE_ETAPENORME
        (
            idModif INTEGER NOT NULL IDENTITY(1,1),
            dateModif DATETIME NOT NULL,
            etpLibelle VARCHAR(50) NOT NULL,
            etpNorme VARCHAR(20) NOT NULL,
            etpDateNorme DATETIME NOT NULL,

            CONSTRAINT PK7 PRIMARY KEY (idModif)
        )
    END

    INSERT INTO ARCHIVE_ETAPENORME
    (dateModif,etpLibelle,etpNorme,etpDateNorme)
    VALUES (GETDATE(),@etpLibelleAvantModif,@etpNormeAvantModif,@etpNormeDateAvantModif)

    IF @etpNormeDateAvantModif <> @etpNormeDate
    BEGIN
        UPDATE ETAPE
        SET ETP_DATE_NORME = @etpNormeDate
        WHERE ETP_NUM = @numEtape
    END

    IF (@etpNormeAvantModif <> @etpNorme) AND (@etpNorme <> '')
    BEGIN
        UPDATE ETAPE
        SET ETP_NORME = @etpNorme
        WHERE ETP_NUM = @numEtape
    END
END

```

Ce trigger permet, lors de la mise à jour de la date ou du libellé de la norme, de réaliser un historique des modifications dans une table qui sera créée si elle n'existe pas déjà et de réaliser la mise à jour de la date ou du libellé de la norme (ou les deux) si les données saisies sont différentes de celles enregistrées en base de données.

Evènement : à la place/après une mise à jour dans la table « ETAPE » (Instead of).

Condition : Si la table « archive\_etapenomme » n'existe pas (historique). Si la nouvelle date est différente de celle enregistré en base de données (mise à jour de la date de la norme). Si la norme est différente de celle enregistrée en base de données (mise à jour de la norme).

Action : Insérer les étapes avant modification dans la table « archive\_etapenomme » afin de réaliser un historique. Mettre à jour la date ou le libellé (ou les deux) de la norme.

- trg\_maj\_nbMedValidFam :

```

ALTER TRIGGER [dbo].[trg_maj_nbMedValidFam]
ON [dbo].[WORKFLOW]
AFTER UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here
    DECLARE @medDL AS VARCHAR(100)
    DECLARE @codeFam AS VARCHAR(100)
    DECLARE @nbMedValid AS INTEGER
    DECLARE @numEtape AS INTEGER
    DECLARE @decision AS INTEGER
    DECLARE @estDernier AS INTEGER

    SELECT @numEtape = numEtape, @medDL = MED_DL, @decision = idDecision FROM inserted

    SELECT @estDernier = 0

    SELECT @codeFam = FAM_CODE
    FROM [dbo].[MEDICAMENT]
    WHERE MED_DEPOTLEGAL = @medDL

    SELECT @nbMedValid = nbMedAutorisé
    FROM [dbo].[FAMILLE]
    WHERE FAM_CODE = @codeFam

    IF @numEtape = (SELECT MAX(numEtape) FROM WORKFLOW WHERE MED_DL = @medDL)
    BEGIN
        SELECT @estDernier = 1
    END

    IF (@estDernier = 1)
    BEGIN
        IF (@decision = 0)
        BEGIN
            IF (@nbMedValid IS NULL)
            BEGIN
                UPDATE [dbo].[FAMILLE]
                SET nbMedAutorisé = 1
                WHERE FAM_CODE = @codeFam
            END
        ELSE
        BEGIN
            UPDATE [dbo].[FAMILLE]
            SET nbMedAutorisé = nbMedAutorisé + 1
            WHERE FAM_CODE = @codeFam
        END
    END
END
END

```

Ce trigger permet de mettre à jour le nombre de médicaments validés par famille, lorsque la dernière étape d'un médicament est validée. Si la valeur est « NULL », on vient modifier la valeur à, sinon on vient l'incrémenter de 1.

Evènement : Après la mise à jour d'une étape du workflow (After).

Condition : Si l'étape est bien la dernière étape du workflow du médicament auquel elle est rattachée. Si la décision est bien acceptée. Si la valeur est « NULL » sinon c'est une valeur numérique.

Action : Mettre à jour le nombre de médicaments validés de la famille du médicament auquel la tâche est rattachée : soit à 1 si c'est « NULL », sinon l'incrémenter de 1.