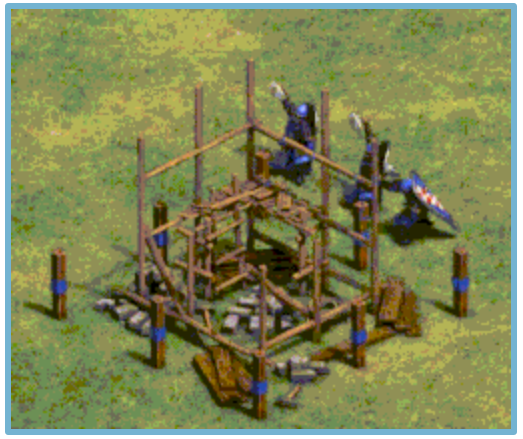
Rencontre 6

CTE et quelques autres notions nouvelles

Bases de données et programmation Web



On va finalement *construire* la base de données !

Requête de données

- Requête de données
  - IIF
  - CASE
  - CTE

Requête de données

❖ IIF

```
SELECT Colonne1, IIF( boolean_expression, true_value, false_value ) , ... FROM Table1;
```

Équivalent à un if ternaire de la programmation.

```
SELECT C.Nom,P.DatePartie,CP.Nom, IIF(CP.Nom IN ('Mario','Luigi','Princesse'),'Intéressante','Pas Intéressante') as [Mon Intérêt]
 FROM Courses.Course C
 INNER JOIN Courses.Participation P
 ON C.CourseID = P.CourseID
 INNER JOIN Joueurs.Choix PC
 ON P.ChoixID = PC.ChoixID
 INNER JOIN Courses.Personnage CP
 ON PC.PersonnageID = CP.PersonnageID
 ORDER BY C.Nom
```

Résultats | Messages

	Nom	DatePartie	Nom	Mon Intérêt
1	Aéroport Azur	2020-04-18 17:21:06.000	Mario	Intéressante
2	Aéroport Azur	2021-06-14 20:59:42.000	Mario	Intéressante
3	Aéroport Azur	2020-08-22 01:50:33.000	Luigi	Intéressante
4	Aéroport Azur	2020-04-02 05:28:42.000	Mario	Intéressante
5	Aéroport Azur	2022-05-13 15:59:42.000	Luigi	Intéressante
6	Aéroport Azur	2021-09-20 19:19:22.000	Luigi	Intéressante
7	Aéroport Azur	2022-09-01 15:42:29.000	Mario	Intéressante
8	Aéroport Azur	2021-02-18 20:38:25.000	Luigi	Intéressante
9	Aéroport Azur	2021-12-09 17:43:10.000	Mario	Intéressante
10	Aéroport Azur	2021-10-17 02:15:21.000	Mario	Intéressante
11	Aéroport Azur	2021-12-28 13:35:17.000	Mario	Intéressante
12	Aéroport Azur	2020-03-23 13:02:50.000	Bébé Mario	Pas Intéressante
13	Aéroport Azur	2022-11-05 17:54:59.000	Roi Boo	Pas Intéressante
14	Aéroport Azur	2021-07-01 18:56:09.000	Daisy	Pas Intéressante
15	Aéroport Azur	2021-11-10 17:16:19.000	Donkey Kong	Pas Intéressante
16	Aéroport Azur	2022-02-21 03:01:54.000	Daisy	Pas Intéressante
17	Aéroport Azur	2021-05-18 23:23:46.000	Waluigi	Pas Intéressante

Requête de données

❖ CASE cas où on examine les différentes valeurs d'une variable

```
SELECT colonne1, colonne2,[nouvelle colonne] = CASE colonne3
                                                  WHEN Value1 THEN Result1
                                                  WHEN Value2 THEN Result2
                                                  …
                                                  ELSE  Result3
                                               END

FROM …
```

Similaire à un Switch en C#

```
SELECT C.Nom,P.DatePartie,CP.Nom, [Mon Intérêt] = CASE CP.Nom
                                                    WHEN 'Mario' THEN 'Super Intéressante'
                                                    WHEN 'Luigi' THEN 'Très Intéressante'
                                                    WHEN 'Princesse' THEN 'Intéressante'
                                                    ELSE 'Pas Intéressante'
                                                  END
FROM Courses.Course C
INNER JOIN Courses.Participation P
ON C.CourseID = P.CourseID
INNER JOIN Joueurs.Choix PC
ON P.ChoixID = PC.ChoixID
INNER JOIN Courses.Personnage CP
ON PC.PersonnageID = CP.PersonnageID
WHERE C.CourseID = 2
ORDER BY C.Nom
```

Résultats | Messages

	Nom	DatePartie	Nom	Mon Intérêt
1	Aéroport Azur	2020-03-14 14:06:17.000	Yoshi	Pas Intéressante
2	Aéroport Azur	2022-01-23 14:23:50.000	Daisy	Pas Intéressante
3	Aéroport Azur	2020-04-18 17:21:06.000	Mario	Super Intéressante
4	Aéroport Azur	2022-04-16 09:01:50.000	Roi Boo	Pas Intéressante
5	Aéroport Azur	2022-01-06 20:06:08.000	Bébé Peach	Pas Intéressante
6	Aéroport Azur	2021-06-16 05:56:23.000	Toadette	Pas Intéressante
7	Aéroport Azur	2021-06-14 20:59:42.000	Mario	Super Intéressante
8	Aéroport Azur	2021-09-22 00:02:45.000	Daisy	Pas Intéressante
9	Aéroport Azur	2021-08-23 15:18:55.000	Yoshi	Pas Intéressante
10	Aéroport Azur	2020-08-22 01:50:33.000	Luigi	Très Intéressante
11	Aéroport Azur	2020-12-28 13:22:09.000	Donkey Kong	Pas Intéressante
12	Aéroport Azur	2022-06-17 06:59:47.000	Bébé Luigi	Pas Intéressante
13	Aéroport Azur	2021-09-25 03:33:52.000	Koopa	Pas Intéressante
14	Aéroport Azur	2020-04-02 05:28:42.000	Mario	Super Intéressante

Requête de données

❖ CASE

```
SELECT colonne1,[nouvelle colonne] = CASE
                                        WHEN ExpressionBooléenne1 THEN Value1
                                        WHEN ExpressionBooléenne2 THEN Value2
                                        …
                                        ELSE Value3
                                     END
FROM …
```

```
-- CASE sans variable
SELECT Nom, [Categorie] = CASE
                              WHEN Vitesse >= 4 THEN 'Très rapide'
                              WHEN Vitesse >=3 AND Vitesse < 4 THEN 'Rapide'
                              WHEN Vitesse <3 THEN 'Lent'
                          END
FROM Courses.Kart
ORDER BY 2 DESC
```

Résultats | Messages

Nom	Categorie
Blue Falcon	Très rapide
Yoshimoto	Très rapide
Autorhino	Rapide
Propulsar	Rapide
Proto 8	Rapide
Chabriolet	Rapide
Beat-bolide	Rapide
Paracoccinelly	Lent
Caravéloce	Lent
Rétro	Lent
Scootinette	Lent
Quad Wiggler	Lent

❖ CTE (Common Table Expression)

```
GO
WITH
Q1 AS (... requête select ...),
Q2 AS (... requête select ...),
Q3 AS (... requête select ...)
SELECT Q1.Colonne1, Q2.Colonne2, ...
FROM Q1 INNER JOIN Q2 ON ...
WHERE Q1.Colonne1 > ( SELECT max(Q3.Colonne5) FROM Q3 )
GO
```

Le mot-clé WITH permet d'obtenir des ensembles de résultats créés à l'aide de requêtes intermédiaires pour ensuite les utiliser dans une requête.

CTE signifie Common Table Expression.

C'est en fait considéré comme une table temporaire dont l'existence (scope) est seulement le temps de la requête
ICI les tables temporaires seraient Q1, Q2 et Q3

On peut les utiliser pour faire des jointures dans la requête SELECT **qui doit absolument suivre la création de ces CTE.**

❖ Sous-requêtes CTE

Le mot-clé WITH, et donc des CTE, permet de briser une requête complexe en plusieurs petites requêtes.

Ainsi la requête suivante est complexe:
Afficher le nom des courses et leur chrono pour les courses dont le chrono est supérieur à la moyenne des chronos pour les karts dont la vitesse est supérieure à la moyenne de vitesse des karts?

Je peux briser cela en CTE (Et on commence vers la fin):
Q1) Quels sont les Karts dont la vitesse est supérieure à la moyenne de vitesse des karts
Q2) Quel est le chrono moyen des Karts obtenus en Q1)
Q3) Quelles courses ont un chrono supérieur au chrono moyen obtenu en Q2)
Pour obtenir enfin les courses et leur chrono dont le chrono est supérieur au chrono moyen obtenu en Q3

```
GO
WITH
Q1 AS ( SELECT KartID FROM Courses.Kart WHERE Vitesse >= (   SELECT AVG(Vitesse) FROM Courses.Kart ) ),
Q2 AS ( SELECT AVG(Chrono) AS [MoyenneChrono] FROM Courses.Participation WHERE KartID IN (SELECT * FROM Q1) ),
Q3 AS ( SELECT CourseID, chrono  FROM Courses.Participation WHERE Chrono > ( SELECT * FROM Q2) )
SELECT Nom, chrono
FROM Courses.Course C
INNER JOIN Q3
ON C.CourseID = Q3.CourseID
GO
```

Requête de données

❖ Sous-requêtes CTE

Pour vérifier si tout est OK et que le résultat est bon, vous pouvez exécuter chaque CTE séparément

```
Q1 AS ( SELECT KartID FROM Courses.Kart WHERE Vitesse >= (   SELECT AVG(Vitesse) FROM Courses.Kart ) ),
```

KartID
2
3
4
7
8
9
11

```
Q2 AS ( SELECT AVG(Chrono) AS [MoyenneChrono] FROM Courses.Participation WHERE KartID IN (SELECT * FROM Q1) ),
--Pour tester
-- Q1 a donné les kartID 2,3,4,7,8,9,11
 SELECT AVG(Chrono) AS [MoyenneChrono] FROM Courses.Participation WHERE KartID IN (2,3,4,7,8,9,11)
```

MoyenneChrono
14282

```
Q3 AS ( SELECT CourseID, chrono  FROM Courses.Participation WHERE Chrono > ( SELECT * FROM Q2) )
 --Pour tester
 -- Q2 a donné 14282
SELECT CourseID, chrono  FROM Courses.Participation WHERE Chrono > 14282
-- Ce qui donne 492 courses dont le chrono est supérieur à 14282
```

CourseID	chrono
10	16491
2	17761
9	14728
3	15542
4	14866
6	14680
2	15046

Et enfin, l'exécution de toute la requête à partir du WITH et avec la requête principale:

```
SELECT Nom, chrono
FROM Courses.Course C
INNER JOIN Q3
ON C.CourseID = Q3.CourseID
```

Nom	chrono
Mine Wario	16491
Aéroport Azur	17761
Piste musicale	14728
Château de Bowser	15542
Désert Toussec	14866

❖ Sous-requêtes CTE

Quand une CTE ne contient qu'un seul nombre, on peut faire une jointure artificielle avec ON 1=1

```
GO
WITH CTE_AVG AS (SELECT AVG(Chrono) AS [AVG_CHRONO] FROM Courses.Participation)
SELECT COUNT(P.ParticipationID) AS [NB Participation Superieures à la moyenne]
FROM Courses.Participation P
INNER JOIN CTE_AVG A
ON 1=1
WHERE P.Chrono > A.AVG_CHRONO
GO
```

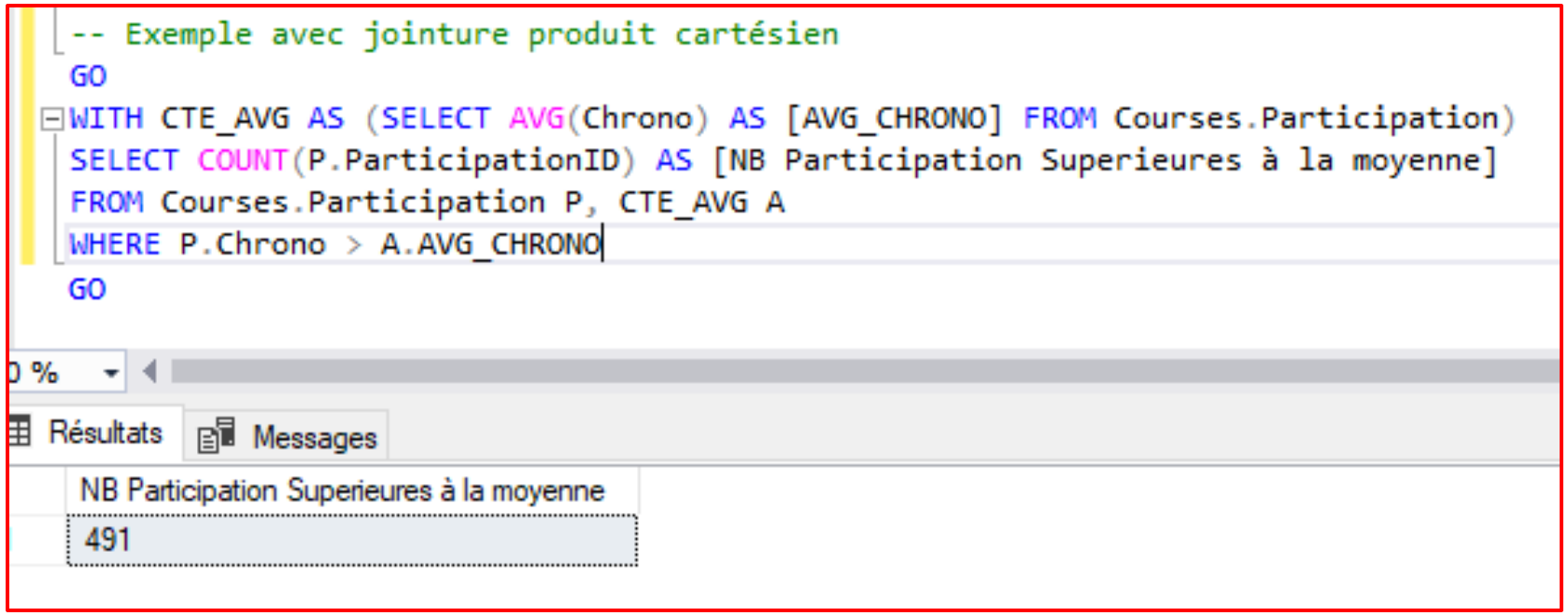
Résultats | Messages

NB Participation Superieures à la moyenne
491

❖ Sous-requêtes CTE

La même requête mais cette fois ci en mettant les deux tables dans le FROM, séparées par une virgule.

Cela donne le produit cartésien des deux tables.



- Jointures
  - Que se passe-t-il physiquement lorsque je fais une jointure entre deux tables ?
  - 1 = 1
  - Table1.PK = Table2.FK

- Que se passe-t-il si je veux faire une jointure complète entre ces deux tables ?

Courses.Kart

KartID	Nom	Vitesse
1	Retro	3,5
2	Intrépide	5
3	Bulldo	2

Courses.Participation

ParticipID	JoueurID	KartID	PersoID	Position
1	10	1	7	1
2	1	2	6	8
3	6	2	12	2
4	4	1	9	3

- SELECT * FROM Courses.Kart K INNER JOIN Courses.Participation P ON 1 = 1
- SELECT * FROM Courses.Kart, Courses.Participation

❖ Voici le résultat

K.KartID	K.Nom	K.Vitesse	P.ParticipID	P.JoueurID	P.KartID	P.PersoID	P.Position
1	Retro	3,5	1	10	1	7	1
1	Retro	3,5	2	1	2	6	8
1	Retro	3,5	3	6	2	12	2
1	Retro	3,5	4	4	1	9	3
2	Intrépide	5	1	10	1	7	1
2	Intrépide	5	2	1	2	6	8
2	Intrépide	5	3	6	2	12	2
2	Intrépide	5	4	4	1	9	3
3	Bulldo	2	1	10	1	7	1
3	Bulldo	2	2	1	2	6	8
3	Bulldo	2	3	6	2	12	2
3	Bulldo	2	4	4	1	9	3

Jointures

- Le produit cartésien de 2 tables vous donne toutes les combinaisons possibles entre les lignes de la table Kart et celles de la table Participation.

- La plupart de ces combinaisons n’ont aucun sens car la valeur de la colonne K.KartID est différente de celle de P.KartID !!

Jointures

K.KartID	K.Nom	K.Vitesse	P.ParticipID	P.JoueurID	P.KartID	P.PersoID	P.Position
1	Retro	3,5	1	10	1	7	1
1	Retro	3,5	2	1	2	6	8
1	Retro	3,5	3	6	2	12	2
1	Retro	3,5	4	4	1	9	3
2	Intrépide	5	1	10	1	7	1
2	Intrépide	5	2	1	2	6	8
2	Intrépide	5	3	6	2	12	2
2	Intrépide	5	4	4	1	9	3
3	Bulldo	2	1	10	1	7	1
3	Bulldo	2	2	1	2	6	8
3	Bulldo	2	3	6	2	12	2
3	Bulldo	2	4	4	1	9	3

Jointures

- La condition ON K.KartID = P.KartID permet de filtrer le résultat de la jointure et de ne garder que les combinaisons qui ont du sens.

K.KartID	K.Nom	K.Vitesse	P.ParticipID	P.JoueurID	P.KartID	P.PersoID	P.Position
1	Retro	3,5	1	10	**1**	7	1
1	Retro	3,5	4	4	**1**	9	3
2	Intrépide	5	2	1	**2**	6	8
2	Intrépide	5	3	6	**2**	12	2

Jointures

- Imaginons la requête suivante :

```
SELECT K.Nom, p.JoueurID, P.Position
FROM Courses.Kart K
INNER JOIN Courses.Participation P ON K.KartID = P.KartID
```

- Toute la procédure de jointure décrite avant se passe en arrière-plan, mais on ne garde que les colonnes K.Nom, P.JoueurID et P.Position

K.Nom	P.JoueurID	P.Position
Retro	10	1
Retro	4	3
Intrépide	1	8
Intrépide	6	2

❖ Retour sur le cas d'une CTE ne retournant qu'une seule colonne et une seule ligne

Quand une CTE ne contient qu'un seul nombre, on peut faire une jointure artificielle avec ON 1=1

```
GO
WITH CTE_AVG AS (SELECT AVG(Chrono) AS [AVG_CHRONO] FROM Courses.Participation)
SELECT COUNT(P.ParticipationID) AS [NB Participation Superieures à la moyenne]
FROM Courses.Participation P
INNER JOIN CTE_AVG A
ON 1=1
WHERE P.Chrono > A.AVG_CHRONO
GO
```

Résultats | Messages

NB Participation Superieures à la moyenne
491

❖ Sous-requêtes CTE

OU l'écrire directement **en mettant les deux tables dans le FROM, séparées par une virgule** en comprenant parfaitement que cela donne le produit cartésien des deux tables.

```
-- Exemple avec jointure produit cartésien
GO
WITH CTE_AVG AS (SELECT AVG(Chrono) AS [AVG_CHRONO] FROM Courses.Participation)
SELECT COUNT(P.ParticipationID) AS [NB Participation Superieures à la moyenne]
FROM Courses.Participation P, CTE_AVG A
WHERE P.Chrono > A.AVG_CHRONO
GO
```

0 %

Résultats | Messages

NB Participation Superieures à la moyenne
491

❖ Produit cartésien des 2 tables?

Courses.Participation

ParticipID	Position	Chrono
1	8	11354
2	4	13599
3	6	12111
4	1	15654
5	2	10245

CTE_AVG

AVG_CHRONO
12458

P.ParticipID	P.Position	P.Chrono	A.AVG_CHRONO
1	8	11354	12458
2	4	13599	12458
3	6	12111	12458
4	1	15654	12458
5	2	10245	12458