Revue-IRS

Revue Internationale de la Recherche Scientifique (Revue-IRS)
ISSN: 2958-8413
Vol. 3, No. 6, Octobre 2025

DE LA GESTION D'AUTO-INCREMENTS DANS UNE APPLICATION MULTI-UTILISATEUR

KABANZA JUSTIN

Master in Internet System, Chercheur à l'Institut Supérieur de Commerce de Goma (ISC-GOMA), Province du Nord-Kivu en République Démocratique du Congo

Résumé

Dans ce présent article, nous nous focalisons sur une application connectée à un Système de gestion de base de données MS SQL/Server avec la propriété identity mise sur une colonne de type numérique. Dans une application multi-utilisateur, après enregistrement de données, un utilisateur peut avoir besoin de générer un rapport à l'instar d'un reçu. Sachant que ce dernier dépend d'un numéro automatique et que le chargement tient compte du numéro maximal, cela présente un risque lorsque les utilisateurs font d'opérations simultanément. Il suffit qu'il y ait un intervalle d'exécution de quelques millisecondes et le reçu à charger sera autre que celui entendu.

Cette recherche a pour but de démontrer les défaillances de l'utilisation des numéros automatiques dans des applications multi-utilisateurs et y proposer une solution.

Abstract

In this article, we focus on an application connected to an MS SQL/Server database management system with the identity property set to a numeric column. In a multi-user application, after saving data, a user may need to generate a report similar to a receipt. Given that the latter depends on an automatic number and that loading takes into account the maximum number, this presents a risk when users perform operations simultaneously. Even a few milliseconds of execution time can cause the receipt to be loaded to be different from the one intended.

The purpose of this research is to demonstrate the shortcomings of using automatic numbers in multi-user applications and propose a solution.

Mot-clés : Auto incrémént; Application ; Multi-utilisateur;

Digital Object Identifier (DOI): https://doi.org/10.5281/zenodo.**17495088**

1 Introduction

Le recours à un système de gestion de base de données pour une conservation et traitement de données étant indispensable pour une application, il convient aux développeurs de bien penser pour pallier à certains disfonctionnements pouvant survenir au cours de l'utilisation de leur application.

L'implémentation d'une application fonctionnelle en tenant compte des tests effectués par le développeur (mono-utilisateur) ne suffit pas pour croire avoir apporté une panacée dans la réalisation de certaines fonctionnalités surtout lorsqu'on a intégré les numéros automatiques dans le système.

L'utilisation d'un numéro-auto dans une application multi-utilisateur crée un problème majeur lié au temps d'exécution, à savoir :

- La génération de la valeur maximale d'une colonne de type numérique dans une application multi-utilisateur en cours d'utilisation ;

Au vu de ce problème détecté, nous sommes partis de la question suivante :

- Comment contourner le problème lié à la génération de la valeur maximale d'une colonne de type numérique dans une application multi-utilisateurs en cours d'utilisation dont un rapport à charger dépend de celle-ci ?

2 Définition des concepts

2.1 Auto-incrément

Un auto-incrément est une colonne d'une table, de type numérique, dont la numérotation est automatique. L'utilisateur n'a donc pas besoin d'enregistrer une information dans ladite colonne. En d'autres termes, il est appelé, « auto-incrément ». Pour son bon fonctionnement, le développeur initialise et fixe le pas d'incrémentation.

2.2 Application

Une application est un programme ou ensemble des logiciels destinés à réaliser des tâches spécifiques. Pour son fonctionnement, celle-ci utilise les ressources du système d'exploitation.

Une application est un programme ou ensemble de logiciels destinés à réaliser une tâche ou un ensemble de tâches élémentaires d'un même domaine. Les applications fonctionnent en utilisant des services du système d'exploitation. (quelle-est-la-difference-entre-un-logiciel-et-une-application/, 2025)

2.3 Multi-utilisateur

Le mot multi-utilisateur vient de deux mots, multi qui veut dire multiple (plusieurs) et utilisateur pour signifier une personne qui manipule un système informatique. Ça vient du verbe « utiliser ».

En combinant les trois mots, on attend par application multi-utilisateur, un système d'information automatisée à travers laquelle plusieurs utilisateurs peuvent la manipuler simultanément.

2.4 MS/SQL Server

MS SQL/Serveur est un Système de gestion de base de données relationnelle dont Microsoft en est le propriétaire. Il est utilisé pour créer une base de données.

Un système de gestion de base de données (SGBD) est un logiciel système utilisé pour stocker, manipuler, gérer et partager des données dans une base de données tout en cachant la

complexité des opérations tout en garantissant la qualité, la pérennité et la confidentialité des données.

Pour Oracle, Un système de gestion de base de données (SGBD) est un logiciel système permettant aux utilisateurs et programmeurs de créer et de gérer des bases de données. (Oracle, 2025)

3 Méthodologie

3.1 L'approche systémique

Cette approche se concentre sur les interactions entre les éléments. (CAMBIEN, 2007)

Elle fait référence à une méthode d'analyse, une façon de traiter un système complexe avec un point de vue global sans se focaliser sur les détails. Elle nous a permis de comprendre le fonctionnement du système mono-utilisateur afin de le comparer avec celui qui est multi-utilisateurs.

3.2 Le prototypage

Le prototypage est la démarche qui consiste à réaliser un prototype. Le prototype est un exemplaire incomplet et non définitif de ce que pourra être le produit ou l'objet final.

On emploie indifféremment les termes prototype physique et maquette pour désigner la représentation exploratoire et tridimensionnelle d'un produit, d'un service ou d'un système. Récemment le terme prototype, plus général, s'est imposé. En phase de conception, en amont de la fabrication, on utilise divers prototypes physiques pour simuler l'action en fonction d'un produit sous différentes facettes. (HALLGRIMSSON, 2019)

3.2.1 Le prototype

Pour présenter notre prototype, nous recourrons à MS SQL/Server 2014 afin de créer une base de données contenant deux tables, à savoir :

- Elève
- Paiement

```
Create database ArticleRIRS;
use ArticleRIRS;

Create table Eleve
(
    idEl varchar(6),
    nom varchar(40),
    postnom varchar(40),
    prenom varchar(40),
    sex varchar(10),
    lieu_naiss varchar(30),
    date_naiss smalldatetime,
    numTel varchar(13),
    constraint pk_pers primary key(idEl)
);

Create table Paiement
(
    idp int identity(1,1),
    montant money,
    dateP date,
    idEl varchar(6),
    motif int,
    annScol varchar(9),
    constraint pk_paie primary key(idp),
    constraint fk_stud_pay foreign key(idEl) references Eleve (idEl) on delete cascade on update cascade
);

Alter table Paiement alter Column motif varchar(30)
```

Figure 1Codes SQL

Ce qui retient notre attention ici, c'est la colonne « **idp** » avec la propriété « **identity** » et l'initialisation « **1** » avec le pas d'incrémentation de « **1** ». Cette propriété signifie que l'identifiant du paiement est un numéro automatique. On n'aura donc pas besoin de spécifier ce champ pour enregistrer une information quelconque.
L'exécution de ces codes sql, nous produit le résultat ci-après :

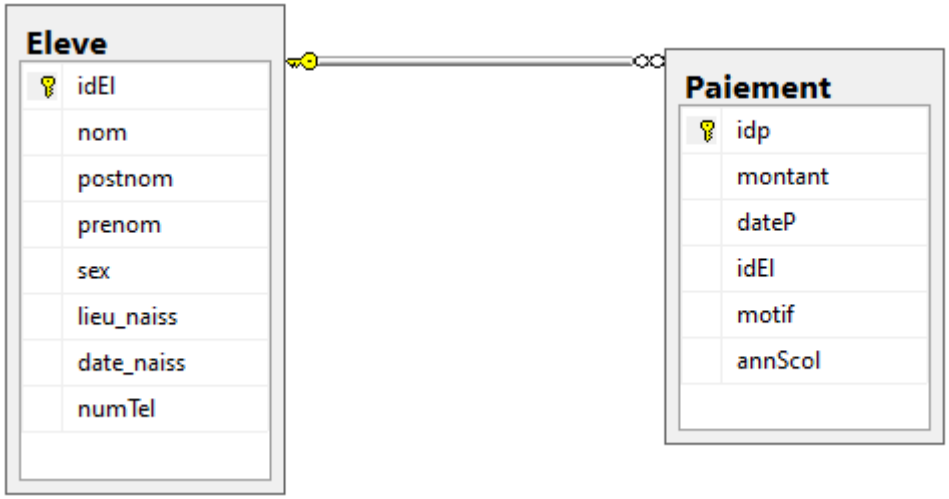

Figure 2Schéma relationnel de base de données

Dans une application, à chaque fois qu'un utilisateur enregistre une opération de paiement, l'identifiant paiement est généré automatiquement. Ce qui peut être traité pour obtenir un reçu en récupérant la valeur maximale de l'identifiant.

- Qu'en est-il d'une application multi-utilisateurs ? Est-ce que la sélection de la valeur maximale sera-t-elle générée comme cela peut se faire pour une application mono-utilisateur ?

Pour répondre à cette préoccupation, allons cas par cas.

- Pour une application mono-utilisateur, enregistrons deux informations illustratives avec les requêtes SQL suivantes :

```
SET DATEFORMAT DMY INSERT INTO Paiement (montant,dateP,idEl,motif,annScol)
VALUES (27,GETDATE(),'431','Frais scolaires','2025-2026');

SET DATEFORMAT DMY INSERT INTO Paiement (montant,dateP,idEl,motif,annScol)
VALUES (15,GETDATE(),'890','Frais techniques','2025-2026')
```

Pour obtenir, la valeur maximale, on recourt à la requête de sélection que voici :

```
Select max(idp) as 'N° Paiement' from Paiement
```

Et le résultat est la suivante :

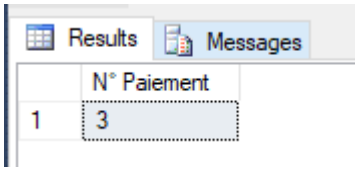

Figure 3 Résultat de la requête sql

- S'agissant d'une application multi-utilisateur, il y a un souci si tous les utilisateurs travaillent sur le processus de paiement au même moment. Il suffit qu'il y ait un décalage de quelques millisecondes et le résultat pourra changer.

Voici la liste d'élèves se trouvant dans notre base de données.

```
Select *from Eleve
```

.00 %

Results | Messages

	idEl	nom	postnom	prenom	sex	lieu_naiss	date_naiss	numTel
1	431	Bryan	KABANZA	MUNYANEZA	M	Goma	2021-07-05 00:00:00	-
2	679	ODINGA	EPUNGU	Michel	M	Goma	2016-04-17 00:00:00	-
3	890	Jeanne	BARAKA	KAHIRE KAHASHI	F	Goma	2018-02-06 00:00:00	-

Figure 4 Liste des élèves

A titre illustratif, l'application multi-utilisateurs dénommée « **Lipa Buke App** » appartient à une école disposant trois cycles dont Maternelle, Primaire et Secondaire et a deux agents percepteurs dont l'un gère les cycles de maternelle et primaire ; et l'autre gère le cycle secondaire.

Au vu du processus de recouvrement des frais scolaires, deux élèves (l'un de la maternelle et l'autre du secondaire) payent simultanément. Sachant que chaque payement est certifié par la remise d'un reçu imprimé dans le système, obtenu à travers la requête sql telle que reprise à la page précédente qui nous a affiché le résultat se trouvant à la figure 3, les deux percepteurs seront butés à la difficulté de voir un même reçu s'afficher partout. Et cela, celui qui a enregistré après l'autre.

Une solution est envisagée pour pallier à ce problème.

- Il s'agit de la création d'un paramètre permettant de distinguer les deux paiements faits simultanément afin de charger le reçu dont le paiement a été effectué chez un utilisateur. Ce paramètre est le nom de l'élève.

  Voici les paiements effectués.

```
Select *from Eleve inner join Paiement on Eleve.idEl=Paiement.idEl
```

00 %

Results | Messages

	idEl	nom	postnom	prenom	sex	lieu_naiss	date_naiss	numTel	idp	montant	dateP	idEl	motif	annScol
1	431	Bryan	KABANZA	MUNYANEZA	M	Goma	2021-07-05 00:00:00	-	2	27,00	2025-10-17	431	Frais scolaires	2025-2026
2	890	Jeanne	BARAKA	KAHIRE KAHASHI	F	Goma	2018-02-06 00:00:00	-	3	15,00	2025-10-17	890	Frais techniques	2025-2026

Figure 5 : Liste des paiements

- Pour sélectionner le reçu dont la valeur maximale dans une application mono-utilisateur, on procède comme suit :

```
Select *from Eleve inner join Paiement on Eleve.idEl=Paiement.idEl where idp=(select max(idp) from Paiement)
```

100 %

Results | Messages

	idEl	nom	postnom	prenom	sex	lieu_naiss	date_naiss	numTel	idp	montant	dateP	idEl	motif	annScol
1	890	Jeanne	BARAKA	KAHIRE KAHASHI	F	Goma	2018-02-06 00:00:00	-	3	15,00	2025-10-17	890	Frais techniques	2025-2026

Figure 6 Reçu application mono-utilisateur

- Comme dit précédemment, pour une application multi-utilisateur, on recourt à un paramètre qui est le nom de l'élève, d'où la requête est la suivante :

```
Select *from Eleve inner join Paiement on Eleve.idEl=Paiement.idEl where idp=
(select max(idp) from Paiement where idEl=(Select idel from Eleve Where CONCAT(nom,' ',postnom,' ',prenom) ='Bryan KABANZA MUNYANEZA'))
```

100 %

Results | Messages

	idEl	nom	postnom	prenom	sex	lieu_naiss	date_naiss	numTel	idp	montant	dateP	idEl	motif	annScol
1	431	Bryan	KABANZA	MUNYANEZA	M	Goma	2021-07-05 00:00:00	-	2	27,00	2025-10-17	431	Frais scolaires	2025-2026

Figure 7 Reçu application multi-utilisateur

Commentaire : la requête dont il est question, est imbriquée c.à.d. que dans une requête, nous mettons une sous-requête pour ajouter le paramètre permettant de charger le reçu dont le numéro est maximal pour un élève. Cela trie les données d'un seul élève dont le paiement est effectué.

```
Select *from Eleve inner join Paiement on Eleve.idEl=Paiement.idEl where idp=
(select max(idp) from Paiement where
idEl=(Select idel from Eleve Where CONCAT(nom,' ',postnom,' ',prenom)='Bryan KABANZA MUNYANEZA'))
```

Figure 8 Requête imbriquée

3.2.2 Discussion

L'autos-incrément est géré différemment, dépendamment du nombre d'utilisateurs pouvant utiliser simultanément l'application pour pouvoir produire un rapport dont le chargement dépend de ceux-ci. La recherche a étayé la logique selon laquelle, une application mono-utilisateur ne fonctionne pas de la même manière que l'application multi-utilisateur lorsqu'on doit gérer les numéros automatiques.

Pour une application mono-utilisateur, il s'observe que :

- L'application n'est pas en conflit avec quoique ce soit ;
- La requête de sélection avec l'usage de la fonction « MAX » à elle-seule suffit pour obtenir un résultat ;

Pour l'application multi-utilisateurs, nous notons que :

- Le conflit existe lorsqu'on veut obtenir le dernier paiement ;
- L'usage de la fonction « MAX » dans une requête SQL dans laquelle on met une sous-requête (requête imbriquée) tenant compte d'un paramètre comme le nom de l'élève, retourne un dernier enregistrement, ce qui permet d'éviter le conflit ;

3.2.3 Figures

4 Conclusion

Gérer les auto-incréments dans une application mono-utilisateur est un travail souvent ignoré, mais lorsqu'il s'agit d'une application multi-utilisateurs telle que « **Lipa Buke App** », le contexte change.

Dans une application multi-utilisateurs, le travail sur l'application peut se faire simultanément et pour plus d'efficacité, il va falloir gérer le chargement des rapports dépendant d'un numéro auto après enregistrement, c'est le cas d'une preuve de paiement à l'instar d'un reçu.

L'objectif principal de cet article est de démontrer la défaillance de l'usage des numéros autos qui survient lors du chargement d'un rapport dont la condition de chargement est la valeur maximale du numéro de paiement ; et y apporter une piste de solution telle que le recours aux sous-requêtes (requêtes imbriquées) avec le nom de l'élève comme paramètre puisque les utilisateurs ne peuvent pas enregistrer simultanément le paiement d'un même élève.

Au vu des discussions effectuées à la page précédente, l'usage des requêtes imbriquées paramétrées est une solution pour éviter le conflit de chargement des rapports.

REFERENCES

[1] CAMBIEN, A. (2007). *Une introduction à l'approche systémique : Appréhender la complexité.* Lyon..

[2] HALLGRIMSSON, B. (2019). *Prototypage et design d'un produit.* Londres: King publishing Ltd.

[3] Oracle. (2025, Octobre 17). *database/systeme-gestion-base-de-donnees-sgbd-definition/*. Retrieved from https://www.oracle.com/: https://www.oracle.com/fr/database/systeme-gestion-base-de-donnees-sgbd-definition/

[4] *quelle-est-la-difference-entre-un-logiciel-et-une-application/*. (2025, Octobre 17). Récupéré sur https://sokeo.fr/: quelle-est-la-difference-entre-un-logiciel-et-une-application/