ادارة المستخدمين في SQL Server

تُدار الصلاحيات (Permissions) في SQL Server عبر نظام أمان هرمي
يمنح أو يرفض إجراءات محددة (SELECT, INSERT, UPDATE, DDL) للمستخدمين والأدوار على مستوى الخادم (Server) أو قاعدة البيانات (Database)، باستخدام أوامر
GRANT, DENY, REVOKE. يتم تخصيصها لحسابات تسجيل الدخول، الأدوار، والكائنات لضمان أمن البيانات.
إليك تفصيل شامل حول الصلاحيات في SQL Server:

1. مستويات الصلاحيات (Permission Levels)

تُقسم الصلاحيات في محرك قاعدة البيانات إلى مستويين رئيسيين:

- **مستوى الخادم (Server-Level):** تشمل الأذونات على مستوى الخادم بالكامل، وتُخصص لحسابات تسجيل الدخول (Logins) وأدوار الخادم (Server Roles).
- **مستوى قاعدة البيانات (Database-Level):** تشمل الأذونات داخل قاعدة بيانات معينة، وتُخصص لمستخدمي قاعدة البيانات (Users) وأدوارها.

2. أنواع الصلاحيات الأساسية

- **صلاحيات الكائنات (Object Permissions):** مثل SELECT, INSERT, UPDATE, DELETE, EXECUTE على الجداول، العروض (Views)، والإجراءات المخزنة (Stored Procedures).
- **صلاحيات تعريف البيانات (DDL Permissions):** مثل CREATE TABLE, ALTER, DROP.

3. أدوار الصلاحيات (Roles)

بدلاً من منح الصلاحيات لكل مستخدم على حدة، يُفضل استخدام الأدوار:

- **أدوار قاعدة البيانات الثابتة (Database-Level Roles):**
  - db_datareader: قراءة جميع البيانات.
  - db_datawriter: إضافة/تعديل/حذف البيانات.
  - db_ddladmin: تشغيل أوامر DDL.
- **أدوار الخادم الثابتة (Server-Level Roles):** مثل sysadmin (مسؤول النظام) الذي يمتلك كافة الصلاحيات.

4. أوامر إدارة الصلاحيات (T-SQL Commands)

تُدار الصلاحيات باستخدام ثلاثة أوامر رئيسية:

- **GRANT**: لمنح صلاحية.
- **DENY**: لرفض صلاحية بشكل قاطع.
- **REVOKE**: لإلغاء صلاحية تم منحها أو رفضها سابقاً.

5. كيفية إدارة الصلاحيات (SSMS)

يمكن إدارتها عبر Microsoft SQL Server Management Studio (SSMS) بالنقر بزر الماوس الأيمن على قاعدة البيانات أو الكائن > **Properties (خصائص)** > **Permissions (الأذونات)**.

إدارة المستخدمين في SQL Server

تعتمد على إنشاء **تسجيلات دخول (Logins)** على مستوى الخادم للوصول، ثم ربطها **بمستخدمين (Users)** داخل قواعد بيانات محددة مع تعيين صلاحيات (Permissions) وأدوار (Roles) مناسبة. تتم العملية عبر SSMS أو أوامر T-SQL باستخدام مصادقة Windows أو SQL Server لضمان أمان البيانات.
إليك تفصيل شامل لإدارة المستخدمين في SQL Server:

1. المفاهيم الأساسية

- **تسجيل الدخول (Login):** هوية المستخدم على مستوى الخادم (Server Level)تتيح الاتصال بمثيل SQL Server.
- **المستخدم (User):** هوية المستخدم داخل قاعدة بيانات محددة (Database Level).
- **الأدوار (Roles):** مجموعات صلاحيات لتسهيل الإدارة (مثل `db_owner`, `db_datareader`).

2. طرق إدارة المستخدمين

يمكن الإدارة عبر واجهة SQL Server Management Studio (SSMS) أو باستخدام أوامر T-SQL.

أ. باستخدام (SSMSالواجهة الرسومية):

1. افتح SSMS واتصل بالخادم.
2. انتقل إلى **Security** > **Logins** لإنشاء تسجيل دخول جديد.
3. افتح قاعدة البيانات المطلوبة، انتقل إلى **Security** > **Users** لربط المستخدم بها.
4. انقر بزر الماوس الأيمن لتعديل الصلاحيات والأدوار.

ب. باستخدام (T-SQLالأوامر):

- **إنشاء Login:**

sql

```
CREATE LOGIN [UserName] WITH PASSWORD = 'StrongPassword123';
```

- **إنشاء User في قاعدة البيانات:**

sql

```
USE [DatabaseName];
CREATE USER [UserName] FOR LOGIN [UserName];
```

- **إضافة المستخدم لدور (مثلاً db_datareader):**

sql

```
ALTER ROLE [db_datareader] ADD MEMBER [UserName];
```

3. أنواع المصادقة (Authentication Modes)

- **مصادقة Windows (Windows Authentication):** تعتمد على مستخدمي الشبكة والويندوز.
- **مصادقة SQL Server (SQL Server Authentication):** تعتمد على اسم مستخدم وكلمة مرور داخلية.

4. أفضل الممارسات

- منح أقل صلاحيات ممكنة (Principle of Least Privilege).
- استخدام الأدوار (Rolesبدلاً من تعيين صلاحيات لكل مستخدم على حدة.
- تفضيل مصادقة Windows للأمان.

بشكل متكامل، يتم استخدام جمل SQL Server لإدارة الصلاحيات في
للسحب) على مستوى الخادم أو قاعدة البيانات، مع تحديد المستخدمين والكائنات (**REVOKE**للمنع)، و (**DENY**، (للمنح) **GRANT**
(جداول، إجراءات). مثال عملي يشمل إنشاء مستخدم، منحه صلاحيات قراءة/كتابة، ومنع حذف بيانات، مع إمكانية منح حق التفويض.
إليك مثال عملي متكامل يوضح السيناريو:
سيناريو: إدارة صلاحيات موظف "إدخال بيانات" (DataEntry)
نريد إنشاء مستخدم جديد، والسماح له بالاطلاع وتعديل جدول Employees فقط، ومنعه تماماً من حذف أي بيانات، مع السماح له بإعطاء هذه الصلاحيات لغيره لاحقاً.

sql

```
-- 1. إنشاء تسجيل دخول (Login) على مستوى الخادم
CREATE LOGIN DataEntryUser WITH PASSWORD = 'StrongPassword123!';

-- 2. إنشاء مستخدم (User) داخل قاعدة البيانات المحددة
CREATE USER DataEntryUser FOR LOGIN DataEntryUser;

-- 3. منح صلاحيات أساسية (SELECT, INSERT, UPDATE) على جدول محدد
GRANT SELECT, INSERT, UPDATE ON OBJECT::dbo.Employees TO DataEntryUser;

-- 4. منح صلاحية تنفيذ إجراء مخزن: مثال متقدم (Stored Procedure)
-- GRANT EXECUTE ON OBJECT::dbo.usp_CalculateSalary TO DataEntryUser;

-- 5. تفعيل خاصية منح الصلاحيات للآخرين (WITH GRANT OPTION)
GRANT SELECT ON OBJECT::dbo.Employees TO DataEntryUser WITH GRANT OPTION;

-- 6. منع صلاحية الحذف (DENY) - لها الأولوية القصوى
DENY DELETE ON OBJECT::dbo.Employees TO DataEntryUser;
```

لإنشاء منظار ((View آمن، يتم دمج بيانات من عدة جداول (مثل Employees و Departments) عبر استعلام JOIN لعرض معلومات محددة فقط، ثم باستخدام أوامر DCL مثل REVOKE لحجب الوصول المباشر للجداول الأساسية، و GRANT للسماح بالوصول للمنظار. هذا يضمن خصوصية البيانات وحصر الرؤية.
مثال عملي: منظار بيانات الموظفين والإدارات
نفرض وجود جدولين:

1. **Employees** (يحتوي رواتب، معلومات حساسة).
2. **Departments** (أسماء الإدارات).

الهدف: السماح لموظف الموارد البشرية برؤية أسماء الموظفين وإداراتهم فقط دون الرواتب.
1. إنشاء المنظار ((View
نقوم بإنشاء View يربط الجدولين ويخفي الرواتب:
sql

```
CREATE VIEW EmployeeDeptView AS
SELECT E.EmpName, D.DeptName
FROM Employees E
JOIN Departments D ON E.DeptID = D.DeptID;
```

2. حجب الصلاحيات عن الجداول الأساسية
نمنع المستخدم (HR_User) من الوصول للجداول الأصلية:
sql

```
REVOKE SELECT ON Employees FROM HR_User;
```

```
REVOKE SELECT ON Departments FROM HR_User;
```

3. منح الصلاحية على المنظار

نسمح له فقط بـ SELECT على المنظار:

sql

```
GRANT SELECT ON EmployeeDeptView TO HR_User;
```

فوائد هذه العملية:

- **أمن البيانات:** لا يرى المستخدم الجداول الحقيقية.
- **تبسيط العرض:** يرى بيانات مدمجة جاهزة.
- **التحكم:** الوصول مقيد بما تم تعريفه في الـ View.