A white paper with black text

AI-generated content may be incorrect.

A certificate of completion

AI-generated content may be incorrect.

Part 01

CREATE OR ALTER PROCEDURE GetStudentsCountPerDepartment

AS

BEGIN

SELECT

D.Dept\_Name AS DepartmentName,

COUNT(S.Std\_Id) AS StudentsCount

FROM

Department D

LEFT JOIN

Student S ON D.Dept\_Id = S.Dept\_Id

GROUP BY

D.Dept\_Name

END

EXEC GetStudentsCountPerDepartment

CREATE OR ALTER PROCEDURE CheckEmployeesInProjectP1

AS

BEGIN

DECLARE @EmpCount INT

SELECT @EmpCount = COUNT(\*)

FROM EmployeeProject EP

JOIN Project P ON EP.Project\_Id = P.Project\_Id

WHERE P.Project\_Name = 'P1'

IF @EmpCount >= 3

BEGIN

PRINT 'The number of employees in the project p1 is 3 or more'

END

ELSE

BEGIN

PRINT 'The following employees work for the project p1'

SELECT

E.First\_Name,

E.Last\_Name

FROM EmployeeProject EP

JOIN Project P ON EP.Project\_Id = P.Project\_Id

JOIN Employee E ON EP.Emp\_Id = E.Emp\_Id

WHERE P.Project\_Name = 'P1'

END

END

EXEC CheckEmployeesInProjectP1

CREATE OR ALTER PROCEDURE ReplaceEmployeeInProject

@OldEmpId INT,

@NewEmpId INT,

@ProjectId INT

AS

BEGIN

UPDATE works\_on

SET Emp\_Id = @NewEmpId

WHERE Emp\_Id = @OldEmpId AND Project\_Id = @ProjectId

END

EXEC ReplaceEmployeeInProject @OldEmpId = 101, @NewEmpId = 202, @ProjectId = 5

Part 02

**✅ أولًا: Full Backup – النسخة الاحتياطية الكاملة**

* هي **نسخة كاملة من قاعدة البيانات** بكل بياناتها وجداولها وإعداداتها.
* تشمل **كل ما تحتويه قاعدة البيانات في لحظة النسخ**.
* تُعتبر الأساس لأي نوع آخر من النسخ الاحتياطي.

🔸 **مثال:**  
لو عندك قاعدة بيانات فيها 1000 سجل، فالـ Full Backup بياخد نسخة من الألف سجل كلهم.

🔹 **متى نستخدمه؟**  
عند الحاجة إلى نسخة شاملة يمكن الاعتماد عليها في الاستعادة الكاملة.

**✅ ثانيًا: Differential Backup – النسخة الاحتياطية التفاضلية**

* تأخذ **فقط التغييرات** التي حدثت منذ آخر Full Backup.
* لا تعتمد على النسخ التفاضلية السابقة، فقط على آخر Full.

🔸 **مثال:**  
لو عملت Full Backup يوم الأحد، وجيت يوم الثلاثاء تعمل Differential Backup، فهي بتاخد كل التغييرات من يوم الأحد إلى الثلاثاء، بغض النظر عن وجود نسخ تفاضلية في النص.

🔹 **مميزاته:**

* أسرع من Full Backup.
* حجم الملف أصغر.

🔹 **عيوبه:**

* تحتاج إلى **Full Backup سابق** حتى تستطيع الاستعادة منها.

**✅ ثالثًا: Transaction Log Backup – نسخة سجل المعاملات**

* تأخذ **كل المعاملات (التغييرات)** التي تمت على قاعدة البيانات منذ آخر Log Backup.
* مفيدة جدًا لو عايز ترجع قاعدة البيانات لنقطة زمنية دقيقة (مثل: الساعة 2:13 مساءً بالضبط).

🔸 **مثال:**  
لو عملت INSERT أو UPDATE أو DELETE، فكل هذه التغييرات تُسجل في Transaction Log.  
هذا النوع من النسخ يأخذ هذه التغييرات فقط.

🔹 **متى نستخدمه؟**

* في قواعد البيانات المهمة اللي بيتم فيها تغييرات كثيرة.
* لما نحتاج إلى خاصية **الاستعادة الزمنية الدقيقة (point-in-time recovery)**.

**✅ أولًا: ما هي Permission (الصلاحيات)؟**

الصلاحيات أو الـ **Permissions** هي الأوامر أو الأذونات اللي بتحدد إيه اللي يُسمح للمستخدم يعمله على قاعدة البيانات أو على عنصر معين داخلها.

يعني لما حد يدخل قاعدة بيانات، مش شرط يكون عنده الحق إنه:

* يقرأ جداول
* يعدل بيانات
* يحذف صفوف
* ينفّذ Procedures

كل ده لازم يُحدد له بصلاحيات.  
فـ **Permission = السماح أو المنع من تنفيذ عمليات معينة داخل SQL Server**.

**✅ ثانيًا: ما معنى GRANT؟**

كلمة **GRANT** معناها "منح".

يعني لما تكتب أمر فيه GRANT، إنت كده بتقول لـ SQL Server:

اسمح للمستخدم ده إنه ينفّذ نوع معيّن من الأوامر.

**مثال:**

لو عايز تدي لمستخدم اسمه AliUser حق قراءة جدول الموظفين:

sql

CopyEdit

GRANT SELECT ON Employees TO AliUser;

ده معناه:

* المستخدم يقدر يعمل SELECT فقط.
* مش هيقدر يعدل أو يحذف أو يضيف.

**✅ ثالثًا: ما معنى DENY؟**

كلمة **DENY** معناها "منع".

يعني لما تكتب أمر فيه DENY، بتقول لـ SQL Server:

امنع المستخدم ده من تنفيذ أمر معيّن، حتى لو كان مسموح له به من قبل.

**مثال:**

لو عايز تمنع AliUser من حذف بيانات من جدول Employees:

sql

CopyEdit

DENY DELETE ON Employees TO AliUser;

وده معناه:

* حتى لو المستخدم كان عنده صلاحية عامة للحذف، مش هيقدر ينفذ DELETE على الجدول ده بالذات.
* الأمر هنا أقوى من أي سماح تاني.

**✅ رابعًا: على أي مستوى (Level) نستخدم الصلاحيات؟**

الصلاحيات في SQL Server ممكن نطبّقها على 3 مستويات:

**1. مستوى السيرفر (Server Level):**

ده لما تكون بتتحكم في الدخول إلى السيرفر نفسه أو إنشاء قواعد بيانات جديدة.

مثال: صلاحية CREATE DATABASE أو VIEW SERVER STATE.

**2. مستوى قاعدة البيانات (Database Level):**

ده لما تكون الصلاحية متعلقة بكل قاعدة البيانات، مش جدول معيّن.

مثال: صلاحية BACKUP DATABASE أو EXECUTE على كل الإجراءات في القاعدة.

**3. مستوى الكائن (Object Level):**

ده لما تكون الصلاحية خاصة بعنصر واحد فقط، زي:

* جدول
* View
* Procedure

مثال: السماح بقراءة جدول Students فقط، أو تنفيذ إجراء InsertStudent.

**✅ ما هو SQL Profiler؟**

**SQL Profiler** هو أداة من أدوات **SQL Server** بنستخدمها علشان:

"نتجسس" أو "نراقب" كل ما يحصل داخل SQL Server في الوقت الحقيقي.

يعني هو **أداة مراقبة (Monitoring Tool)** بتعرض لك:

* كل الأوامر اللي بتتكتب أو بتتنفذ.
* كل الاستعلامات (Queries) اللي قاعدة البيانات بتتعامل معاها.
* الزمن اللي بيستغرقه كل استعلام.
* مين المستخدم اللي نفّذ أمر معين.
* إذا حصلت أخطاء أو بطء في النظام.

**✅ إمتى بنستخدم SQL Profiler؟**

فيه حالات كتير نستخدم فيها SQL Profiler، أهمها:

**1. تتبع الأداء (Performance Monitoring):**

لو قاعدة البيانات بطيئة أو التطبيق اللي مربوط بيها فيه تهنيج، نستخدم SQL Profiler عشان نعرف:

* الاستعلامات البطيئة.
* مين المستخدم اللي بيضغط على السيرفر.
* عدد الاتصالات (Connections) المفتوحة.

**2. تحليل استعلامات التطبيق (Application Queries):**

لو عندك برنامج مربوط بقاعدة البيانات ومش عارف إيه الاستعلامات اللي بيبعتهالك من وراه، تفتح SQL Profiler وتشوف كل استعلام بيوصل للسيرفر.

**3. اكتشاف المشاكل والأخطاء (Debugging):**

لو فيه مشاكل في النظام، أو أوامر بتتفذ فجأة، تقدر تراقب اللي بيحصل لحظة بلحظة وتعرف مصدره.

**4. تحليل النشاط في وقت معين (Audit):**

تستخدمه علشان تسجّل كل النشاط اللي حصل على قاعدة البيانات في وقت معين، زي:

* مين عمل تعديل؟
* مين حذف بيانات؟
* مين دخل على النظام؟

**✅ هل SQL Profiler آمن دايمًا؟**

✳ لا. هو أداة قوية، لكن ماينفعش تسيبها شغالة لفترة طويلة في بيئة الإنتاج (Production)، لأنها:

* بتستهلك موارد السيرفر.
* ممكن تبطّأ الأداء لو ضغط كبير.

يفضّل تشغيلها وقت الحاجة فقط، وتسجيل البيانات ثم إيقافها

**✅ ما هو الـ Trigger؟**

الـ **Trigger** (أو المشغّل) هو كود بيتكتب داخل SQL Server، وظيفته إنه **يتنفذ تلقائيًا** لما يحصل حدث معين في قاعدة البيانات.  
يعني مش محتاج تشغله بنفسك، السيرفر هو اللي بيشغله لحاله.

**✅ إمتى بيتنفذ الـ Trigger؟**

بيتنفذ لما يحصل واحد من دول على جدول معيّن:

* **INSERT**: لما يتضاف صف جديد.
* **UPDATE**: لما يتغير أي بيانات في صف.
* **DELETE**: لما يتم حذف صف.

**✅ ليه بنستخدم الـ Trigger؟**

بنستخدمه علشان:

1. **نفرض قواعد عمل (Business Rules)**:  
   زي إنك تمنع حذف موظف لو لسه عنده بيانات مرتبطة (زي مشاريع أو مهام).
2. **ننفذ أوامر تلقائية بعد أي تعديل**:  
   مثلًا لما يتسجل طالب جديد، تكتب اسمه تلقائيًا في جدول سجل (Log Table).
3. **نسجل كل التغييرات اللي بتحصل**:  
   يعني تعمل تتبع (Audit) لأي تعديل أو حذف حصل.
4. **تحافظ على تكامل البيانات**:  
   تقدر تمنع إدخال بيانات غير صحيحة أو ناقصة.

**✅ على أي مستوى بنستخدم الـ Trigger؟**

الـ Trigger بيتكتب على **الجداول (Tables)** أو **العروض (Views)** في قاعدة البيانات.

يعني بتربطه بجدول معيّن، وكل مرة يحصل تعديل عليه، التريجر يشتغل تلقائيًا.

**✅ إيه اللي يميز الـ Trigger عن أي كود تاني في SQL؟**

الميزة الكبيرة للتريجر إنه:

* **يتنفذ تلقائيًا**: مش محتاج تنادي عليه.
* **مربوط بحدث معين**: يعني ما يشتغلش إلا لما يحصل حاجة زي INSERT أو DELETE.
* **يشتغل في الخلفية**: المستخدم غالبًا مش بيكون عارف إنه شغّال.

**✅ مثال بسيط:**

نفترض إن عندك جدول اسمه Student، وعايز كل مرة يتسجل طالب جديد، تطبع رسالة.

sql

CopyEdit

CREATE TRIGGER trg\_StudentInsert

ON Student

AFTER INSERT

AS

BEGIN

PRINT 'طالب جديد تم تسجيله';

END

كل ما حد يعمل INSERT في جدول Student، التريجر ده هيشتغل ويطبع الرسالة.

A screenshot of a phone

AI-generated content may be incorrect.