SELECT (2)

نستخدم الكلمة المفتاحية WHERE مع تعليمة SELECT لاستعادة مجموعة من السجلات التي تحقق شرط أو مجموعة من الشروط التي نعبر عنها بعبارة شرطية.

- تُعيد العبارة الشرطية قيمة منطقية (صح أو خطأ)
- يمكن للعبارة الشرطية أن تتضمن عمليات مقارنة مثل (= ، < > ، > ، < ، => ، =<) ويتم ضم السجل الذي يحققها إلى جدول النتائج.

الكلمات المفتاحية LIKE و BETWEEN

- تُستخدم الكلمة المفتاحية LIKE ضمن العبارة الشرطية، كشرط لوجود مثيل. غالباً ما تُستخدَم هذه الكلمة مع إشارة (%)، التي تضاف إلى القيمة التي نبحث عن مثيلاتها، كبديل عن أي رقم من الأرقام أوالأحرف.
- تُستخدم الكلمة المفتاحية BETWEEN ضمن العبارة الشرطية، كشرط لوجود قيمة محصورة بين قيمتين محددتين
- تقبل الكلمة المفتاحيةWHERE أكثر من شرط يفصل بينها عمليات منطقية مثل AND أو OR ويمكن أن يسبق الشرط العملية NOT لنفيه.

تعليمة DELETE

تقوم تعليمة DELETE بحذف سجل أو مجموعة من السجلات من جدول ما.

تعليمة INSERT

تُستخدم تعليمة INSERT لإدراج سجل في جدول محدد.

يمكن لتعليمة INSERT إدراج أكثر من سجل بأمر واحد ولكن ستحتاج إلى استخدام ما ندعوه الاستعلامات الفرعية (Sub Queries) التي سنأتي على ذكرها لاحقاً.

تعليمة UPDATE

تُستخدم تعليمة UPDATE لتعديل البيانات في سجل أو في مجموعة من السجلات.

ويمكن استخدام الكلمة المفتاحية WHERE مع تعليمة UPDATE لتحديد شروط التعديل.

بعض الملاحظات العملية

من المهم استخدام تعليقات SQL وخصوصاً عند كتابة نصوص SQL تحتوي على عدد كبير من الأسطر والتعليمات.

كذلك، من المهم تلافي استخدام أسماء أعمدة (حقول) حاوية على فراغات. أما في الحالات الاضطرارية، فيمكن استخدام إشارات تنصيص أو أقواس مربعة لإحاطة اسم الحقل (أقواس في Oracle و إشارة تنصيص في Access و SQL Server).

أساسيات لغة معالجة المعطيات
Data Manipulation Language (DML) Basics

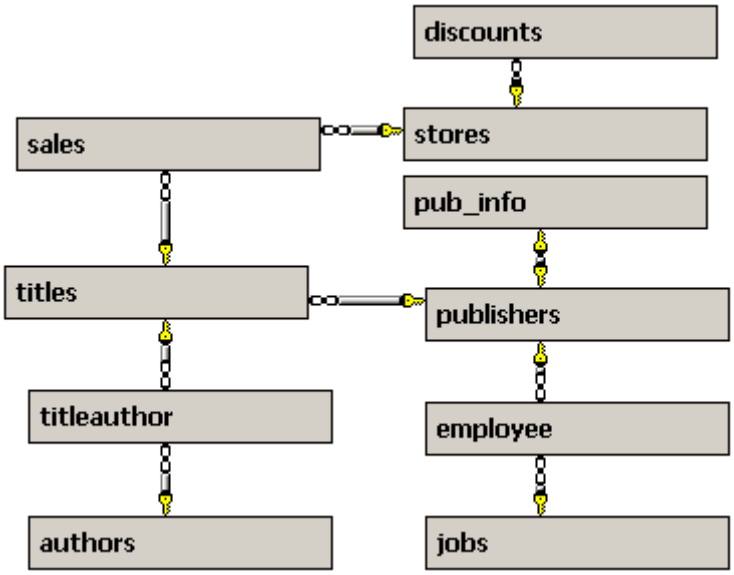
الهدف من الجلسة

١. شرح قاعدة المعطيات Pubs
٢. شرح قاعدة المعطيات Northwind.
٣. استخدام تعليمة Select في استعادة البيانات من قاعدة البيانات.
٤. استخدام تعليمة Insert لإدراج سجلات ضمن جدول في قاعدة البيانات.
٥. استخدام تعليمة Delete لحذف سجل أو مجموعة من السجلات من جدول في قاعدة البيانات.
٦. استخدام تعليمة Update لتعديل سجل أو مجموعة من السجلات في جدول في قاعدة البيانات.
٧. التعرف على الإطار التطبيقي لاستخدام هذه التعليمات مع الكلمات المفتاحية المرافقة لها.

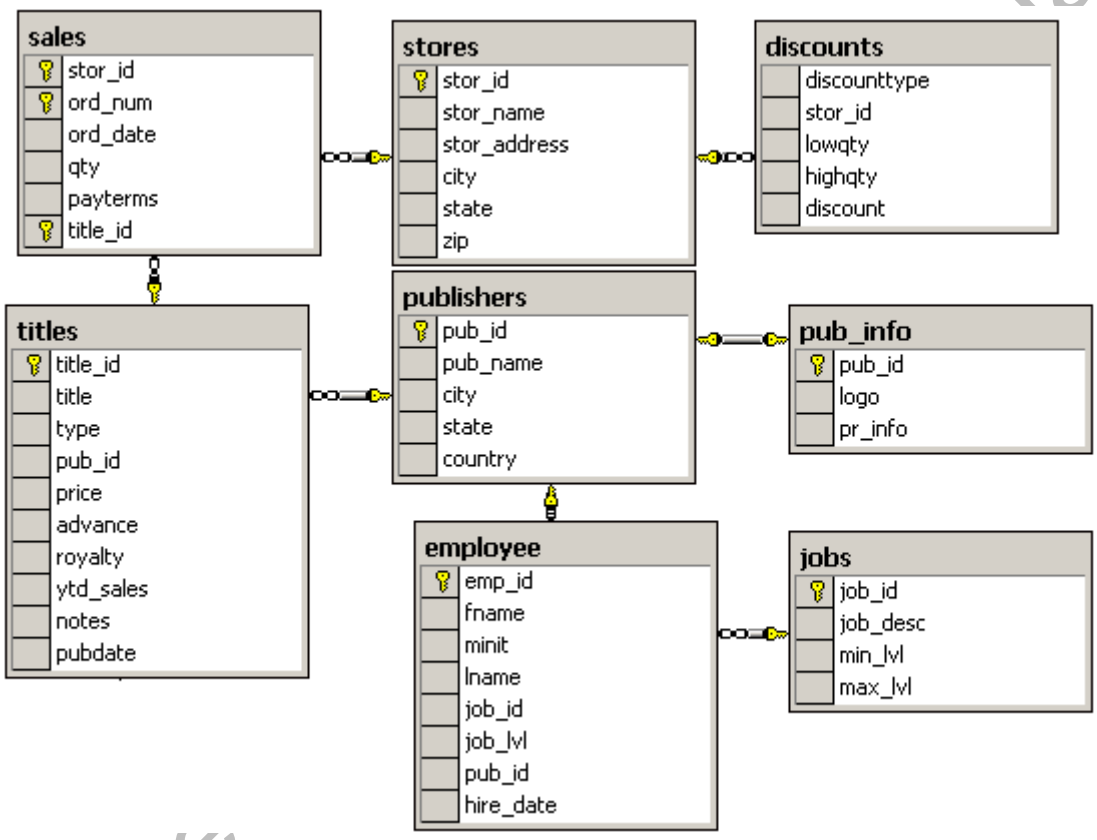
شرح قاعدة المعطيات Pubs

وهي قاعدة معطيات لشركة تقوم ببيع الكتب. يتبع لهذه الشركة مجموعة من المتاجر Stores التي يجري كلا منها حسومات Discounts على المبيعات. يقوم كل متجر بعمليات بيع Sales للكتب Titles المتوفرة لديه. لكل كتاب مجموعة من المؤلفين TitleAuthor. كل مؤلف Author يشارك في تأليف مجموعة من الكتب. لكل كتاب دار نشر Publishers معين .في كل دار نشر مجموعة من الوظائف Jobs التي يعمل في كل منها عدة موظفين Employees. لكل ناشر شعار وتوصيف لعنوان الناشر التي تخزن في الجدول Pub_info.

المخطط العام



Part I – Titles Authors



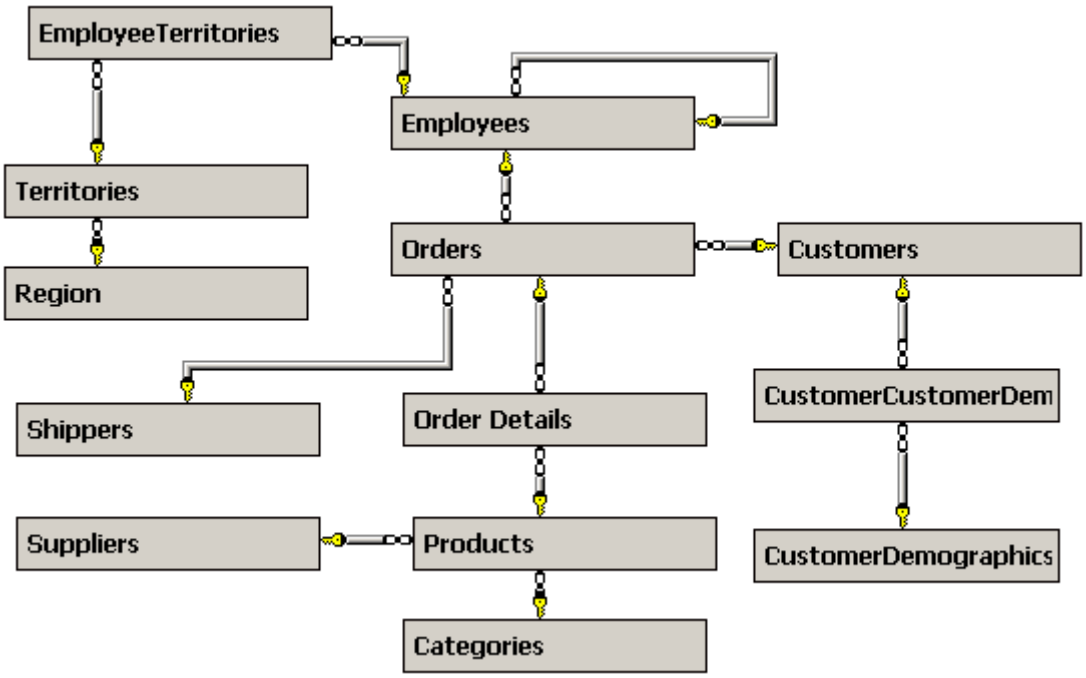
Part II

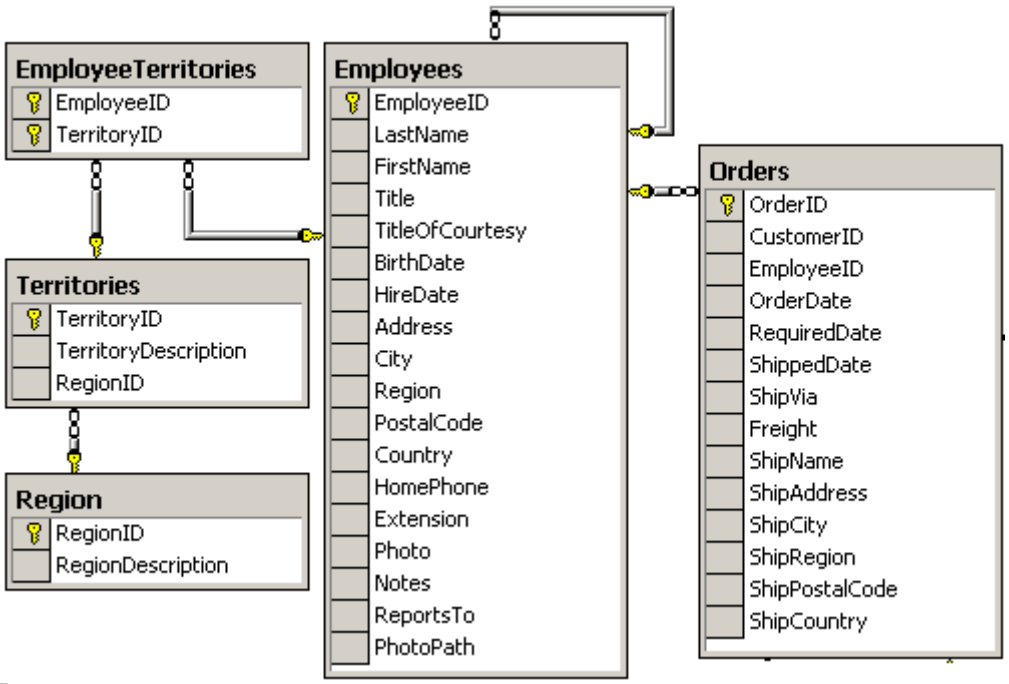
Titles Stores, Titles Publishers, Publishers Jobes

قاعدة المعطيات Northwind

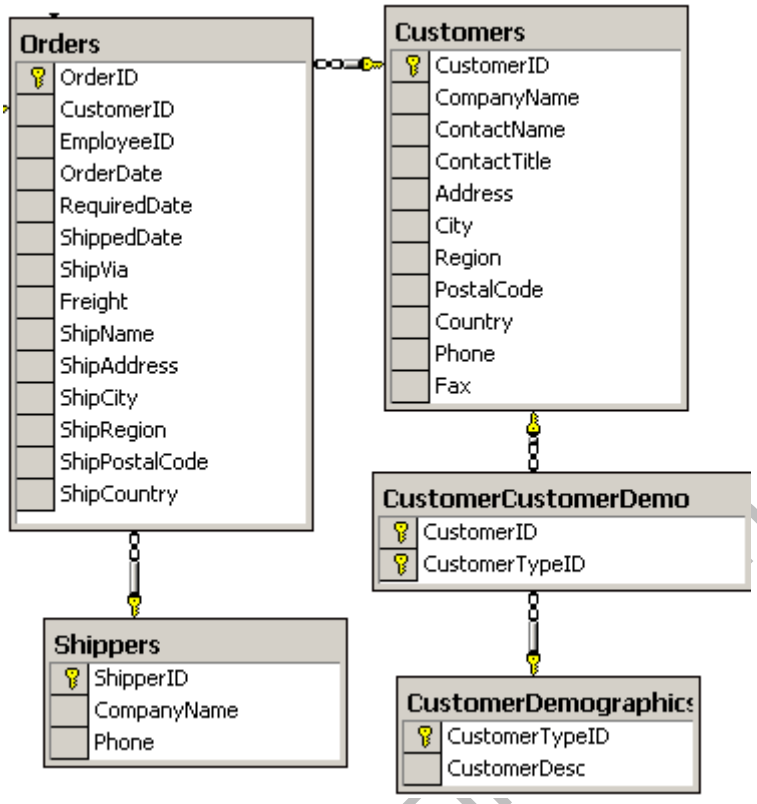
وهي قاعدة بيانات لشركة افتراضية تدعى Northwind Traders Company. تقوم هذه الشركة بتزويد زبائنها Customers بالطلبيات Orders التي قام موظفو الشركة Employees بتوقيعها مع الزبائن. لكل طلبية مجموعة من البنود Order Items التي يوافق كل منها منتجا Product. تؤمن الشركة كل منتج عن طريق موردين Suppliers. تقوم الشركة بتوصيل الطلبية إلى الزبائن عن طريق موزعين Shippers.

المخطط العام

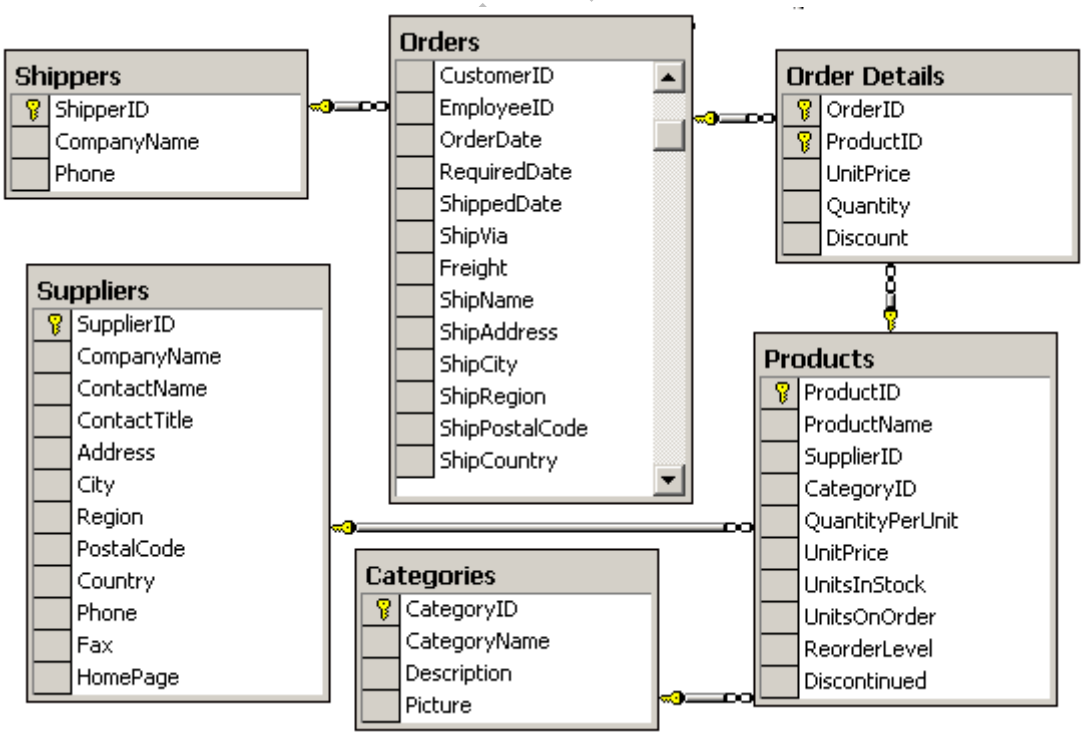




(Part I) Employees orders



(Part II) Customers orders



(Part III) Products orders

استخدام تعليمة Select في استعادة البيانات من قاعدة البيانات.

تُعتبر تعليمة SELECT من أشهر تعليمات اللغة وأكثرها استخداماً. تُستخدَم هذه التعليمة لاستعادة و انتقاء مجموعة من البيانات من قاعدة البيانات و ذلك بإعادة جدول يحتوي مجموعة البيانات المطلوبة.

a. تُستخدم إشارة * كبديل لأسماء الحقول (عادة لا ننصح باستخدامها في الحالات التطبيقية لأنها تُحمِّل برنامج إدارة قاعدة البيانات عبء تحديد الحقول وتحديد عددها و أسماءها).
b. يُستخدم تعبير DISTINCT لاستعادة جميع السجلات مع إلغاء التكرار في السجلات المعادة
c. يُستخدم التعبير ORDER BY لترتيب السجلات المُعادة ترتيباً تصاعدياً أو تنازلياً حسب التعبير المرافق المستخدم: ASC للترتيب التصاعدي أو DESC للترتيب التنازلي
d. في حال الرغبة باستخدام أسماء بديلة لحقول جدول القيم المعادة نستخدم التعبير AS

أمثلة:

من قاعدة المعطيات Pubs، اكتب تعليمة اختيار جميع السجلات من جدول المؤلفين.	Select * From Authors Select Authors.* From Authors
من قاعدة المعطيات Pubs، اكتب تعليمة اختيار جميع السجلات من جدول المؤلفين. السجلات يجب أن تكون مرتبة حسب كنية المؤلف تصاعديا، وحسب اسم المؤلف تصاعديا	Select * From Authors Order by Au_lname asc, au_fname asc
من قاعدة المعطيات Pubs، اكتب تعليمة اختيار اسم المؤلف، كنية المؤلف، رقم الهاتف، المدينة، الولاية من جدول المؤلفين	Select au_fname, au_lname, phone as telephone, City, State from authors Order by Au_fname asc, au_lname asc
من قاعدة المعطيات Pubs، اكتب تعليمة اختيار جميع أسماء المؤلفين وأسماءهم الذين يسكنون في ولاية CA ، مرتبين حسب الاسم والكنية	Select Au_fname, au_lname From authors Where state = 'CA' Order by au_fname asc, au_lname asc
من قاعدة المعطيات Northwind، ما هي البلدان التي يوجد فيها زبائن؟ نلاحظ التكرار في أسماء البلدان	Select country From customer Order by Country asc
من قاعدة المعطيات Northwind، ما هي البلدان التي يوجد فيها زبائن؟ نلاحظ عدم تكرار في أسماء البلدان	Select distinct country From customers Order by Country asc

e. نستخدم الكلمة المفتاحية WHERE مع تعليمة SELECT لاستعادة مجموعة من السجلات التي تحقق شرط أو مجموعة من الشروط التي نعبر عنها بعبارة شرطية

i. تُعيد العبارة الشرطية قيمة منطقية (صح أو خطأ)

ii. يمكن للعبارة الشرطية أن تتضمن عمليات مقارنة مثل (= ، < > ، < ، > ، => ، =<) ويتم ضم السجل الذي يحققها إلى جدول النتائج

iii. تُستخدم الكلمة المفتاحية LIKE ضمن العبارة الشرطية، كشرط لوجود مثيل. غالباً ما تُستخدَم هذه الكلمة مع إشارة (%)، التي تضاف إلى القيمة التي نبحث عن مثيلاتها، كبديل عن أي رقم من الأرقام أوالأحرف

iv. تُستخدم الكلمة المفتاحية BETWEEN ضمن العبارة الشرطية، كشرط لوجود قيمة محصورة بين قيمتين محددتين

v. تقبل الكلمة المفتاحيةWHERE أكثر من شرط يفصل بينها عمليات منطقية مثل AND أو OR ويمكن أن يسبق الشرط العملية NOT لنفيه

أمثلة

من قاعة المعطيات Northwind، من هم الزبائن من خارج 'Argentina'؟	Select Contactname, contacttitle, country From customers Where county <> 'argentina' Order by country asc
من قاعدة المعطيات Northwind، من هم الزبائن الذين يسكنون في بلدان يبدأ اسمها بحرف U.	Select Contactname, contacttitle, country From customers Where country like 'u%' Order by country asc
من قاعدة المعطيات Northwind، من هم الزبائن الذين يسكنون في بلدان لا يبدأ اسمها بحرف U.	Select Contactname, contacttitle, country From customers Where not country like 'u%' Order by country asc
من قاعدة المعطيات Northwind، ما هي المنتجات التي يتوفر منها في المخزن من ١٠٠ إلى ٢٠٠ وحدة؟	Select Productname, unitsinstock From products Where unitsinstock between 100 and 200 Order by unitsinstock

٢. استخدام تعليمة Insert لإدراج سجلات ضمن جدول في جدول في قاعدة البيانات

في قاعدة المعطيات Northwind،

أضف شركات الشحن التالية إلى الجدول shippers

Shipperid	Companyname	phone
101	private	(011) 111-1234
102	dabool	(043) 222-4444
103	cookers	(041) 333-3333
104	leverage	(031) 888-8888

Insert into shippers values

(101,'private','(011) 111-1234')

التعليمة السابقة لا تعمل لأن الواصفة Shipperid يتم تزويدها من قبل DBMS.

إذا يجب إضافة فقط العمودين Companyname و phone؛ لذلك يصبح شكل التعليمات :

```
Insert into shippers (companyname, phone) values ('private','(011) 111-1234')
Insert into shippers (companyname, phone) values('dabool','(043) 222-4444')
Insert into shippers (companyname, phone) values('cookers','(043) 333-3333')
Insert into shippers (companyname, phone) values('leverage','(031) 888-8888')
```

٣. استخدام تعليمة Update لتعديل سجل أو مجموعة من السجلات في جدول في قاعدة البيانات

من قاعدة المعطيات Northwind،

عدل البيانات التي قمت بإضافتها إلى الجدول shippers

Companyname	phone
private	(011) 111-1234
dabool	(043) 222-4444
cookers	(041) 333-3333
detergent	(031) 888-8888

لتصبح:

Companyname	phone
special	(011) 111-1234
dabool	(043) 222-4444
cookers	(041) 666-6666
detergents	(031) 999-9999

```
Update shippers set   Companyname='special'  Where Companyname='private'
Update shippers set  Phone ='(041) 666-6666'  Where companyname = 'cookers'
Update shippers set  Companyname = 'detergents', phone = '(031) 999-9999'
Where companyname = 'detergent'
```

٤. استخدام تعليمة Delete لحذف سجل أو مجموعة من السجلات من جدول في قاعدة البيانات

في قاعدة المعطيات Northwind، اكتب التعليمات اللازمة لحذف شركات الشحن التي لها الأسماء التالية: Companyname Special Dabool Cookers Detergents	Delete shippers Where companyname = ‘special’ Delete shippers Where companyname = ‘dabool’ Delete shippers Where companyname = ‘detergents’
في قاعدة المعطيات Northwind، اكتب التعليمات اللازمة لحذف كل شركات الشحن	Delete shippers يقوم بحذف كل السجلات ولا يحذف الجدول

المحاضرة السادسة

التوابع التجميعية ـ التوابع الدرجية

Scalar Functions - Aggregate Functions

تعريف التابع التجميعي

هو تابع يولد قيم مختصرة Summary. يقوم التابع التجميعي بمعالجة جميع القيم المختارة في عمود ما لتوليد ناتج وحيد.

تطبق التوابع التجميعية على الأسطر المختارة في عملية الانتقاء.

المخطط العام للتوابع التجميعية

AGG-FUNC ([ALL | DISTINCT] *expression*)

ALL: تطبيق التابع التجميعي على جميع القيم .
DISTINCT: تطبيق التابع التجميعي على القيم المختلفة فقط (تجاهل التكرار)
expression: هو قيمة ثابتة، أو اسم عمود، أو تعبير حسابي أو محرفي ما.

استخدام التوابع التجميعية في تعليمة الانتقاء SELECT

تابع المجموع SUM

يقوم بحساب مجموع القيم في تعبير حسابي يتضمن عمودا واحدا أو أكثر.

مثال EX_03_01

احسب مجموع المبيعات إلى الآن من جميع الكتب

```
SELECT SUM(YTD_SALES) 'YTD SALES' FROM TITLES
```

تابع الوسطي AVG

يقوم بحساب القيمة الوسطية لتعبير حسابي يتضمن عمودا واحدا أو أكثر

مثال EX_03_02

احسب وسطي أسعار الكتب فيما لو تم زيادة سعر كل كتاب بمقدار 10$

```
SELECT AVG(PRICE + 10) AS 'AVG PRICE' FROM TITLES
```

تابع العدد COUNT

يقوم بحساب عدد الأسطر الموجودة في مجموعة ما. يمكن الاستعاضة عن التعبير بالرمز * للدلالة على أننا نريد عدد جميع الأسطر المحققة لشرط ما بغض النظر عن التكرار.

مثال EX_03_03

احسب عدد المدن التي يوجد فيها كتاب.

```
SELECT COUNT (DISTINCT CITY) AS 'City Count' FROM AUTHORS
```

مثال EX_03_04

أوجد عدد الكتب الموجودة

```
SELECT COUNT(*) AS 'TITLES COUNT' FROM TITLES
```

كما يمكن استخدام أكثر من تابع تجميعي في نفس التعليمة الواحدة.

مثال EX_03_05

أوجد العدد والسعر الوسطي للكتب التي يزيد سعر كل منها عن 10$

```
SELECT COUNT(*) 'BOOK COUNT',AVG(PRICE) 'AVG PRICE' FROM
TITLES WHERE PRICE>10
```

تابع القيمة العظمى MAX

يعيد القيمة العظمى في تعبير وذلك بعد إهمال القيم NULL.

مثال EX_03_06

أوجد أعلى قيمة مبيعات لكتاب في هذا العام

```
SELECT MAX(YTD_SALES)'BEST SALES' FROM TITLES
```

تابع القيمة الدنيا MIN

يعيد أدنى قيمة في تعبير وذلك بعد إهمال القيم NULL.

مثال EX_03_07

```
SELECT MIN(YTD_SALES) 'LOWEST SALES'  FROM TITLES
```

تجميع الأسطر وتعليمة GROUP BY

تحدد المجموعات التي سيتم فرز أسطر الخرج ضمنها. إذا احتوت التعليمة على توابع تجميعية فإن التعليمة تحسب قيم موجزة للمجموعات.

كل عمود تم تحديده في تعليمة الاختيار SELECT، يجب وضعه في فقرة الـ GROUP BY.
الأسطر العائدة من التعليمة لا تكون في ترتيب محدد لذلك يفضل دوما استخدام الترتيب ORDER BY لتحديد الترتيب المرغوب.

[GROUP BY [ALL] *group_by_expression* [,*...n*] [WITH { CUBE | ROLLUP }]]

لا يمكن أن يحوي تعبير التجميع على توابع تجميعية.
تفيد كلمة ALL في إعادة جميع المجموعات الناتجة بما في ذلك المجموعات الفارغة.

CUBE, ROLLUP سوف تشرح بشكل منفصل.

غالبا ما تستخدم فقرة HAVING مع فقرة GROUP BY لتحديد شرط معين على المجموعات المختارة وهي تشبه فقرة WHERE بالنسبة لتعليمة SELECT.

مثال EX_03_08

أوجد قائمة بالسنوات وعدد الموظفين الذين تم توظيفهم في كل منها.

```
SELECT      DATEPART(yy, HIREDATE) AS 'YEAR',
            COUNT(*) AS 'HIRED COUNT'
FROM EMPLOYEES
GROUP BY DATEPART(YY, HIREDATE)
```

مثال EX_03_09

احسب عدد الموظفين في كل مدينة مرتبين حسب عدد الموظفين (من القاعدة NORTHWIND)

```
SELECT CITY, 'EMPLOYEES' = COUNT(*)
FROM EMPLOYEES
GROUP BY CITY
```

```
ORDER BY 'EMPLOYEES'
```

مثال EX_03_10

اوجد قائمة بأنواع الكتب مع السعر الوسطي لكل نمط ومجموع المبيعات الجارية له وذلك بالنسبة لكل ناشر.

```
SELECT      TYPE,
            PUB_ID,
            'AVG' = AVG(PRICE),
            'YTD' = SUM(YTD_SALES)
FROM TITLES
GROUP BY TYPE, PUB_ID
ORDER BY TYPE, PUB_ID
```

عند كتابة تعليمة اختيار، يمكنك تحديد الفقرة WHERE. في هذه الحالة جميع الأسطر التي لا تحقق الشرط تهمل قبل أن تجرى عملية التجميع.

مثال EX_03_11

من قاعدة البيانات Northwind، أوجد السعر الوسطي لمبيعات كل نوع من أنواع الكتب وذلك فقط للكتب التي يزيد سعرها عن $10.

```
SELECT      TYPE,
            'AVG'= AVG(PRICE)
FROM TITLES
WHERE PRICE >10
GROUP BY TYPE
ORDER BY TYPE
```

بعكس الفقرة WHERE التي تضع شروطا على البيانات قبل تجميعها، فإن الفقرة HAVING تضع شروطا على البيانات قبل أو بعد تجميعها.

مثال EX_03_12

أوجد قائمة بالناشرين الذين تجاوزت مبيعاتهم الجارية مبلغ $10,000. (شرط بعد التجميع)

```
SELECT      PUB_ID,
            TOTAL = SUM(YTD_SALES)
FROM TITLES
GROUP BY PUB_ID
HAVING SUM(YTD_SALES)>10000
ORDER BY PUB_ID
```

مثال EX_03_13

أوجد قائمة بالناشرين مع قيمة المبيعات الجارية لكل ناشر الذين ينشرون أكثر من خمسة كتب

```
SELECT      PUB_ID,
            TOTAL = SUM(YTD_SALES)
FROM TITLES
GROUP BY PUB_ID
HAVING COUNT(*)>5
ORDER BY PUB_ID
```

كما في الفقرة WHERE، يمكن وضع أكثر من شرط في الفقرة HAVING. في حال وجود أكثر من شرط، نستخدم العمليات المنطقية AND, OR, NOT.

مثال EX_03_14

من أجل كل ناشر، أوجد مجموع الدفعات والسعر الوسطي لكتب هذا الناشر، شريطة أن يكون رقم لناشر أكبر من '0800' و مجموع الدفعات المقدمة أكبر من 10.000$ ووسطي سعر الكتاب أقل من 20$

```
SELECT      PUB_ID,
            TOTAL = SUM(ADVANCE),
            'AVG'=AVG(PRICE)
FROM        TITLES
WHERE       PUB_ID >'0800'
GROUP BY    PUB_ID
HAVING
            SUM(ADVANCE) > 10000
            AND
            AVG(PRICE) <$20
ORDER BY PUB_ID
```

يكون استخدام كلمة ALL مفيدا إذا احتوى الاستقصاء على شرط اختيار WHERE. وفي هذه الحالة فإن نتيجة الاستقصاء ستتضمن جميع المجموعات بما في ذلك المجموعات التي لا تحوي على بيانات.

مثال EX_03_15

نريد قائمة بأنواع الكتب والسعر الوسطي لكتب كل نوع وذلك فقط للكتب التي ضريبتها 10$

```
SELECT      TYPE,
            'AVG'= AVG(PRICE)
FROM        TITLES
WHERE       ROYALTY=$10
GROUP BY    TYPE
ORDER BY    TYPE
```

تظهر التعليمة السابقة أنواع الكتب التي تتضمن كتبا ضريبتها 10$ ولا تظهر الأنواع التي لا تتضمن أي كتاب ضريبته 10$

مثال EX_03_16

نريد قائمة بأنواع الكتب والسعر الوسطي لكتب كل نوع وذلك للكتب التي ضريبتها 10$ مع إظهار جميع الأنواع بما فيها تلك التي لا تحوي أي كتاب ضريبته 10$

```
SELECT      TYPE,
            'AVG'= AVG(PRICE)
FROM        TITLES
WHERE       ROYALTY=$10
GROUP BY    ALL TYPE
ORDER BY    TYPE
```

PATINDEX

REPLACE

QUOTENAME

REPLICATE

REVERSE

RIGHT

RTRIM

SOUNDEX

SPACE

STR

STUFF

SUBSTRING

UNICODE

UPPER

Date and Time Functions توابع التاريخ والزمن

DATEADD

DATEDIFF

DATENAME

DATEPART

DAY

GETDATE

GETUTCDATE

MONTH

YEAR

توابع الحماية Security Functions

HAS_DBACCESS

IS_MEMBER

SUSER_SID

SUSER_SNAME

USER_ID

USER

توابع النظام System Functions

FORMATMESSAGE

ISDATE

ISNULL

ISNUMERIC

NULLIF

المحاضرة التاسعة
الاستعلامات الفرعية

أساسيات الاستعلامات الفرعية SUBQUERY FUNDAMENTALS

الاستعلام الفرعي هو تعليمة انتقاء SELECT تعيد قيمة وحيدة بحيث تكون مضمنة ضمن تعليمة SELECT, INSERT, UPDATE, or DELETE أو ضمن استعلام فرعي آخر. يمكن استخدام الاستعلامات الفرعية في أي مكان يمكن وضع تعبير فيه.

مثال ١:

المطلوب تقرير بأرقام الطلبيات وتواريخها مع أعلى سعر لمفردات كل طلبية.

```
SELECT
	o.OrderId,
	o.OrderDate,
	(
		SELECT
			Max(od.UnitPrice)
		FROM
			[order details] od
		WHERE
			od.OrderId = o.OrderId
	) AS MaxItemPrice
FROM
	orders o
```

كما يسمى الاستعلام الفرعي بالاستعلام الداخلي Inner Query، بينما يسمى الاستعلام الأب بالاستعلام الخارجي Outer Query.

يتضمن الاستعلام الفرعي الفقرات التالية:

١. فقرة الانتقاء النظامية Select
٢. فقرة جداول الانتقاء From
٣. فقرة الفلترة Where وهي فقرة اختيارية
٤. فقرة تجميع Group by وهي فقرة اختيارية
٥. فقرة التجميع و الفلترة Having وهي فقرة اختيارية

قواعد الاستعلامات الفرعية SUBQUERY RULES

يجب أن يحاط الاستعلام الفرعي دوما بأقواس ().

يمكن وضع الاستعلام الفرعي في فقرات WHERE, HAVING من تعليمات SELECT, UPDATE, DELETE. كما يمكن تضمين استعلام فرعي ضمن استعلام فرعي آخر حتى ٣٢ مستوى.

يمكن للاستعلام الفرعي أن يظهر في مكان أي تعبير شريطة أن يعيد قيمة واحدة.

لا يمكن استخدام أعمدة الجداول الفرعية في عملية الاختيار الخارجية إذا كانت الجداول المستخدمة في الاستعلام الفرعي غير مستخدمة في الاستعلام الخارجي.

غالبا ما يتم استخدام الاستعلامات الفرعية في فقرة الفلترة WHERE للعمليات SELECT, UPDATE, DELETE.

إذا ضمن الاستعلام الفرعي في فقرة مقارنة فيجب أن يطبق الاستعلام الفرعي على تعبير أو عمود واحد (باستثناء استخدام الاستعلام الفرعي مع EXISTS, and IN)

يمكن استخدام فقرات Group By و Having في الاستعلامات الفرعية المضمنة في فقرة مقارنة متبوعة حكما بإحدى الكلمات ALL أو ANY

لا يمكن استخدام كلمة DISTINCT في الفقرات الفرعية المتضمنة على Group By

يمكن استخدام فقرة Order By مع كلمة TOP فقط في الاستعلامات الفرعية

تحديد أعمدة الاستعلام الفرعي QUALIFYING COLUMN NAMES IN SUBQUERIES

تتبع الأعمدة المذكورة في فقرة الاختيار والفلترة إلى الجداول المذكورة في فقرة FROM ما لم يذكر خلاف ذلك صراحة.

مثال ٢:

نريد قائمة بأسماء الناشرين الذين لا ينشرون كتبا في مجال إدارة الأعمال

```
SELECT
	PUB_NAME
FROM
	Publishers
WHERE
	PUB_ID NOT IN
	(
		SELECT
			PUB_ID
		FROM
			TITLES
		WHERE
			TYPE = 'BUSINESS'
	)
```

أنواع الاستعلامات الفرعية SUBQUERY TYPES

الاستعلامات الفرعية مع أسماء مستعارة Sub queries with aliases

إن استعمال الأسماء المستعارة في الاستعلامات الفرعية مفيد لتوضيح تابعية الأعمدة المختارة في كل من الاستعلامين الخارجي والداخلي، وتظهر أهمية الأسماء المستعارة في حال استخدم الاستعلامين الخارجي والداخلي على بعض الجداول المشتركة مع انتقاء أعمدة بنفس الاسم

مثال ٠٣:

نريد تقريرا باسم ونسبة ومدينة كل مؤلف اسمه الأول Livia ونسبته Karsen

```
SELECT
	a1.au_fname, a1.au_lname, city
FROM
	Authors a1
WHERE
	a1.au_id IN
	(
		SELECT
			a2.au_id
		FROM
			Authors a2
		WHERE
			a2.au_fname = 'Livia'
			AND
			a2.au_lname = 'Karsen'
	)
```

الاستعلامات الفرعية مع الفقرة IN

يمكن للاستعلامات الفرعية الموضوعة في فقرة IN أن تعيد سطرا واحدا أو أكثر. يتم في هذه الحالة تنفيذ الاستعلام الداخلي أولا ثم تنفيذ الاستعلام الخارجي بحيث يشمل الخرج قائمة الأسطر التي تتطابق في جزء منها (جزء المقارنة) مع سطر واحد أو أكثر من الاستلام الداخلي وذلك في حالة IN أما في حالة NOT IN فيتم إعادة أسطر الاستعلام الخارجي المغايرة لجميع أسطر الاستعلام الداخلي

مثال ٠٤:

نريد تقرير بأسماء الناشرين الذين ينشرون كتبا في مجال إدارة الأعمال

```
SELECT
	Pub_name
FROM
	Publishers
WHERE
		pub_id IN
		(
			SELECT
				pub_id
			FROM
				Titles
			WHERE
				type= 'Business'
		)
```

مثال ٠٥:

نريد تقريرا بأسماء المؤلفين الذين يسكنون مدينة كاليفورنيا ويحصلون على أقل من ٣٠% من المبيعات كحق ملكية

```
SELECT
	au_fname, au_lname
FROM
	Authors
WHERE
	state = 'CA'
	AND
	au_id IN
	(
		SELECT
			au_id
		FROM
			titleauthor
		WHERE
			royaltyper<30
	)
```

الاستعلامات الفرعية مع الفقرة NOT IN

مثال ٠٦:

نريد تقريرا بأسماء الكتاب الذين تزيد حقوق ملكيتهم على ٣٠% من نسبة المبيعات

```
USE PUBS
GO
SELECT
	au_fname, au_lname
FROM
	Authors
WHERE
	au_id NOT IN
	(
		SELECT
			au_id
		FROM
			titleauthor
		WHERE
			royaltyper<30
	)
```

الاستعلامات الفرعية في تعليمات الإضافة، الحذف والتعديل

يمكن استخدام الاستعلامات الفرعية مع جميع تعليمات DDL الأربعة

مثال ٠٧:

نريد مضاعفة أسعار جميع الكتب للناشرين في مجال إدارة الأعمال

```
UPDATE
	titles
SET
	price = price * 2
WHERE
	pub_id IN
	(
		SELECT
			pub_id
		FROM
			titles
		WHERE
			type='Business'
	)
```

الاستعلامات الفرعية مع عمليات المقارنة

يمكن استخدام عمليات المقارنة التالية(<> ,= ,!> ,!< ,=< ,< ,=> ,>) في الاستعلامات الفرعية.

عمليات المقارنة الغير متغيرة (لا تحوي على ANY, ALL) المستخدمة في الاستعلامات الفرعية يجب أن تعيد قيمة واحدة.

مثال ٠٨:

نريد قائمة بأسماء المؤلفين الذين يسكنون في نفس المدينة التي تقع فيها شركة Algodata Infosystems. علما أن هناك ناشرا وحيدا في كل مدينة.

```
SELECT
	au_fname, au_lname
FROM
	authors
WHERE
	city =
	(		SELECT
			city
		FROM
			publishers
		WHERE
			pub_name = 'Algodata Infosystems'	)
```

كما يمكن استخدام التوابع التجميعية في الاستعلامات الفرعية المستخدمة مع عمليات المقارنة غير المتغيرة، ذلك أن التوابع التجميعية تعيد قيمة وحيدة.

مثال ٠٩:

نريد قائمة بأسماء الكتب التي سعرها أدنى ما يمكن.

```
SELECT
	title
FROM
	titles
WHERE
	price =
	(
		SELECT
			MIN(price)
		FROM
			titles
	)
```

بما أن الاستعلامات الفرعية المستخدمة في عمليات المقارنة غير المتغيرة يجب أن تعيد قيمة وحيدة فلا يمكن استخدام فقرات Group By و Having فيها ما لم نكن واثقين أن هذه الاستعلامات سوف تعيد قيمة وحيدة.

مثال ١٠:

أوجد قائمة بنمط واسم جميع الكتب التي يساوي سعرها سعر أرخص كتاب من كتب إدارة الأعمال

```
SELECT
	type,title
FROM
	TITLES
WHERE
	price =
(
	SELECT
		MIN(price)
	FROM
		TITLES
	GROUP BY type
	HAVING
		type='Business'
)
```

عمليات المقارنة مع ANY, SOME, ALL

نسمي عملية مقارنة ما بأنها متغيرة إذا سبقت بإحدى الكلمات ANY, SOME, ALL.
يمكن للاستعلامات الفرعية المستخدمة في مقارنة متغيرة أن تعيد أكثر من سطر، كما يمكنها أن تستخدم Group By و Having.

لنأخذ على سبيل المثال المقارنة المتغيرة أكبر من الكل ALL <: لكي يتم اختيار سطر من الاستعلام الخارجي فيجب أن تكون قيمة حقل المقارنة فيه أكبر من جميع القيم العائدة من الاستعلام الداخلي.

أما المقارنة المتغيرة أكبر من أحد ANY < فإنه لكي يتم اختيار سطر من الاستعلام الخارجي فيجب أن يكون حقل المقارنة أكبر من إحدى قيم الاستعلام الداخلي على الأقل (SOME مكافئة لـ ANY التي تكافئ بدورها IN في حالة المساواة)

مثال ١١:

نريد قائمة بالمؤلفين الذين يسكنون في مدن يوجد فيها ناشرين

```
USE PUBS
GO
SELECT
	au_fname, au_lname
FROM
	authors
WHERE
	city = ANY
(
	SELECT
		city
	FROM
		publishers
)
```

نلاحظ أن ANY = مكافئ تماما لـ IN غير أن هذا الكلام غير صحيح من أجل ANY<> و NOT IN. حيث أن ANY<> تعني أن عنصر المقارنة مختلف عن أحد عناصر الاستعلام الداخلي على الأقل، في حين أن NOT IN تعني أن عنصر المقارنة مختلف عن جميع عناصر الاستعلام الداخلي.

الاستعلامات الفرعية مع EXISTS

عندما يسبق الاستعلام الفرعي كلمة EXISTS فإن الاستعلام الفرعي يكافئ اختبار وجود أي أن الاستعلام الفرعي لا يعيد أي بيانات وإنما يعيد إما صح TRUE أو خطأ FALSE. يتم اختيار أحد أسطر الاستعلام الخارجي إذا كانت نتيجة تنفيذ اختبار وجوده في الاستعلام الفرعي تساوي TRUE.

مثال ١٢:

أوجد قائمة بالناشرين الذين ينشرون كتبا في مجال إدارة الأعمال.

```
SELECT
	pub_name
FROM
	publishers p
WHERE
	EXISTS
	(
		SELECT
			*
		FROM
			titles t
		WHERE
			p.pub_id = t.pub_id
			AND
			t.type = 'Business'
	)
```

إن كلمة EXISTS مهمة جدا حيث تكون أحيانا هي الطريقة الوحيدة للحصول على المعلومات المطلوبة. جميع الاستعلامات التي تستخدم IN, SOME, ANY يمكن إعادة صياغتها باستخدام EXISTS.

مثال ١٣:

أوجد قائمة باسم وكنية كل مؤلف يسكن في مدينة يوجد فيها دار نشر.

```
USE pubs
SELECT
	au_lname,
	au_fname
FROM
	authors
WHERE
	city = ANY
 (
		SELECT city FROM publishers
 )
-- Or
USE pubs
SELECT
	au_lname,
	au_fname
FROM
```

```
        authors
WHERE
        exists
  (
                SELECT
                        *
                FROM
                        publishers
                WHERE
                        authors.city = publishers.city
  )
```

الاستعلامات الفرعية مع NOT EXISTS

وهي مماثلة لكلمة EXISTS غير أنها تعطي نتيجة معاكسة لـ EXISTS

إيجاد التقاطعات والفروق باستخدام EXISTS, NOT EXISTS

أحد أهم استخدامات EXISTS و NOT EXISTS هو تنفيذ عمليات الجبر العلاقاتي.

تقاطع مجموعتين من البيانات يعيد الأسطر الموجودة في كليهما، أما الفرق فيعيد الأسطر الموجودة في إحداهما وغير موجودة في الأخرى.

مثال ١٤:

أوجد المدن التي تحوي ناشرين ومؤلفين (أي تقاطع مدن الناشرين مع مدن المؤلفين)

```
SELECT
        DISTINCT city
FROM
        authors
WHERE EXISTS
  (
                SELECT
                        *
                FROM
                        publishers
                WHERE
                        authors.city = publishers.city
        )
```

أما المدن التي يوجد فيها مؤلفين ولا يوجد فيها ناشرين (أي الفرق بين مدن المؤلفين والناشرين) فنحصل عليها باستخدام NOT EXISTS.

مثال ١٥:

```
SELECT
        DISTINCT city
FROM
        authors
WHERE NOT EXISTS
  (
                SELECT
                        *
                FROM
                        publishers
                WHERE
                        authors.city = publishers.city
        )
```

الاستعلامات الفرعية المتعددة المستويات

يمكن نسج الاستعلامات الفرعية إلى أي مستوى (حتى ٣٢).

مثال ١٦:

أوجد قائمة بالناشرين الذين نشروا على الأقل كتابا في مجال Popular-Comp

```
SELECT au_lname, au_fname

FROM authors

WHERE au_id IN

  (SELECT au_id

        FROM titleauthor

        WHERE title_id IN

    (   SELECT title_id

        FROM titles

        WHERE type = 'popular_comp'

                )

        )
```

الاستعلامات الفرعية المترابطة

الاستعلام الفرعي المترابط هو استعلام فرعي تعتمد قيمه على قيم الاستعلام الخارجي. أي أن الاستعلام الفرعي سوف ينفذ من أجل كل سطر في الاستعلام الخارجي.

مثال ١٧:

أوجد قائمة بأسماء المؤلفين الذين يحصلون على ١٠٠% من حقوق الملكية.

```
USE pubs
GO
SELECT
        au_lname, au_fname
FROM authors a
WHERE 100 IN
  (
                SELECT
                        royaltyper
                FROM
                        titleauthor ta
                WHERE
                        ta.au_ID = a.au_id )
```

الاستعلامات الفرعية المترابطة باستخدام أسماء مستعارة

من التمرين السابق نلاحظ أن استعمال الأسماء المستعارة لجداول مفيد لسهولة القراءة ووضوح تبيان التابعية.

الاستعلامات الفرعية المترابطة باستخدام تعابير مقارنة

يمكن استعمال عمليات المقارنة مع الاستعلامات الفرعية المترابطة كغيرها من الاستعلامات الفرعية.

مثال ١٨:

نريد قائمة بنمط واسم الكتب التي يفوق سعرها السعر الوسطي لهذا النمط.

```
SELECT
        t1.type,
        t1.title
FROM
        titles t1
WHERE t1.price >
  (
        SELECT
                AVG(t2.price)
        FROM
                titles t2
        WHERE
                t1.type = t2.type    )
```

الاستعلامات الفرعية المترابطة باستخدام فقرة HAVING

يمكن استخدام الاستعلامات الفرعية المترابطة في فقرة HAVING

مثال ١٩:

أوجد قائمة بأنماط الكتب التي فيها أعلى دفعة مقدمة أكبر من ضعفي وسطي الدفعات لهذا النمط.

```
SELECT
    t1.type
FROM
    titles t1
GROUP BY
    t1.type
HAVING
    MAX(t1.advance) >=ALL
    (
        SELECT
            2 * AVG(t2.advance)
        FROM
            titles t2
        WHERE
        t1.type = t2.type
    )
```

المحاضرة العاشرة
ربط الجداول
(Joining Tables)

يمكن التعبير عن صيغة الربط بالصيغة التالية:

```
SELECT
    Table1.Column1,
    Table2.Column2
FROM
    Table1, Table2;
```

يسمى استعلام الربط البسيط أيضاً بالربط المتصالب Cross join. ويمكن التعبير عن نفس صيغة الربط السابقة، بالصيغة:

```
SELECT
    Table1.Column1,
    Table2.Column2
FROM
    Table1 CROSS JOIN Table2;
```

انتبه:

لا تدعم قواعد بيانات DB2 التعبير CROSS JOIN.

الربط بالتساوي Equal Join

يُعرّف الربط بالتساوي على أنه الربط البسيط بين سجلات جدول أول، وسجلات جدول ثان اعتماداً على مساواة بين قيمة حقل في سجل من الجدول الأول (عادة المفتاح الخارجي) وقيمة حقل في سجل من الجدول الثاني (عادة المفتاح الرئيسي).

يُعبّر عن الربط بالتساوي بالصيغة:

```
SELECT
    Table1.Column1,
    Table1.Column2,
    Table2.Column3
FROM
    Table1,
    Table2
WHERE
    Table1.Column1 = Table2.Column2;
```

عموماً، لا تستخدم عملية الربط بالضرورة نفس الحقول التي يجب أن يعيدها الاستعلام.

مثال:

لنفرض أننا نبحث عن أسماء الموظفين الذين يعملون في قسم المحاسبة.

عندها سيكون الاستعلام على الشكل التالي:

```
SELECT
        DeptName, FirstName, LastName
FROM
        Employee, Department
WHERE
 Employee.DeptID = dbo.Department.DeptID

AND
DeptName = 'Accounting'
```

وستكون النتيجة هي:

```
DeptName            FirstName           LastName
--------------      --------------      --------------
Accounting          Tim                 Wallace
```

Equal Join

Access, SQL Server, Oracle, MySQL

Syntax الصيغة

```
SELECT
      Table1.Column1,
      Table1.Column2,
      Table2.Column3
FROM
      Table1,
      Table2
WHERE
      Table1.Column1 = Table2.Column2;
```

```
SELECT
      Table1.Column1,
      Table1.Column2,
      Table2.Column3
FROM
      Table1 Join
      Table2
ON  Table1.Column1 = Table2.Column2;
```

أمثلة

١. أسماء الموظفين الذين لديهم ولد اسمه Michael .

```
SELECT FirstName, LastName
FROM Employee, Children
WHERE Employee.EmpID = Children.EmpID
      AND ChildName = 'Michael'
```

٢. أسماء الأبناء الذكور للموظفين العاملين في قسم التسويق.

```
SELECT ChildName
FROM Department, Employee, Children
WHERE Department.DeptID = Employee.DeptID
      AND Employee.EmpID = Children.EmpID
      AND Children.Sex = 'Male'
      AND DeptName = 'Marketing'
```

Administration	Tim	Wallace
Administration	Jacob	Anderson
Administration	Laura	Miller
Finance	Tim	Wallace
Finance	Jacob	Anderson
Finance	Laura	Miller
Finance	Anne	Ryan
Marketing	Tim	Wallace
Marketing	Anne	Ryan

الربط الداخلي Inner Join

يعطي الربط الداخلي نفس النتيجة التي يعطيها الربط بالتساوي، الفرق فقط بالصيغة. إذ لا تزودنا جميع أنواع أنظمة إدارة قواعد المعطيات بالربط الداخلي. فنسخ Oracle ما قبل ٩ لا تدعم الربط الداخلي.

بالنسبة للصيغة، يوجد اختلافان:

- يفصل بين أسماء الجداول الكلمات Inner Join بدلاً من الفواصل ’,‘.
- يتغير موضع تحديد العلاقة بين الجداول من Where إلى On، تاركين بذلك التعبير Where للشروط التقليدية.

الأمثلة الواردة في الجدول التالي هي نفسها الواردة في فقرة الربط بالتساوي، وبالتالي يمكنك المقارنة.

Inner Join	
Access, SQL Server, Oracle 9i, MySQL	
SELECT Field | Field, Field, Field | * **FROM** Table1 **INNER JOIN** Table2 **ON** Table1.Field = Table2.Field	Syntax الصيغة
١. أسماء الموظفين الذين لديهم ولد اسمه Michael . **SELECT** FirstName, LastName **FROM** Employee **Inner Join** Children **ON** Employee.EmpID = Children.EmpID **WHERE** ChildName = 'Michael'	مثال
٢.أ. أسماء الأبناء الذكور للموظفين العاملين في قسم التسويق.	Oracle 9i,

SQL Server, MySQL	``` SELECT ChildName FROM Department Inner Join Employee ON Department.DeptID = Employee.DeptID Inner Join Children ON Employee.EmpID = Children.EmpID WHERE Children.Sex = 'Male' AND DeptName = 'Marketing' ```
Access, Oracle 9i, SQL Server, MySQL	٢.ب. أسماء الأبناء الذكور للموظفين العاملين في قسم التسويق. ``` SELECT ChildName FROM (Department Inner Join Employee ON Department.DeptID = Employee.DeptID) Inner Join Children ON Employee.EmpID = Children.EmpID WHERE Children.Sex = 'Male' AND DeptName = 'Marketing' ```
مثال	٣. أسماء وأعداد أولاد الموظفين في جميع الأقسام. ``` SELECT FirstName, LastName, Count(ChildID) As ChidrenCount FROM Employee Inner Join Children ON Employee.EmpID = Children.EmpID GROUP BY FirstName, LastName ```

انتبه:

يواجه MS Access صعوبة في ترجمة عمليات الربط بصورة مباشرة، لذلك نلجأ إلى توليد بنى ربط متداخلة مجمعة بأقواس. كما هو مبين في المثال ٢.ب في الجدول السابق.

ملاحظة:

رأينا فيما سبق كيفية تغيير اسم حقل أو عمود ضمن التعبير Select في أي استعلام، وذلك عبر ما يسمى Alias. يمكننا أيضاً تغيير أسماء الجداول ضمن SQL، بهدف تقصير طول الاستعلام أو تسهيل قراءته.

لاستعمال alias ضمن التعبير From، أتبِع الاسم الحقيقي للجدول بفراغ ثم الاسم الجديد الذي ترغب بالتعامل معه. ويمكنك إضافة As بين الاسم الحقيقي للجدول والاسم الجديد له.

وعند تغيير تسمية جدول، لا يمكن استخدام الاسم الحقيقي في بقية الاستعلام. يبيّن الجدول التالي عدة أمثلة على استخدام إعادة تسمية الجداول.

Table Aliases	
Access, SQL Server, Oracle, MySQL	
SELECT Field | Field, Field, Field | * **FROM** Table1 Alias1 **Inner Join** Table2 Alias2 **On** Alias1.Field = Alias2.Field **SELECT** Field | Field, Field, Field | * **FROM** Table1 Alias1, Table2 Alias2 **Where** Alias1.Field = Alias2.Field	Syntax الصيغة
١. أسماء الموظفين الذين لديهم ولد اسمه Michael . **SELECT** FirstName, LastName **FROM** Employee E, Children C **WHERE** E.EmpID = C.EmpID **AND** ChildName = 'Michael'	أمثلة
٢. أسماء الأبناء الذكور للموظفين العاملين في قسم التسويق. **SELECT** ChildName **FROM** Department D **Inner Join** Employee E **ON** D.DeptID = E.DeptID **Inner Join** Children C **ON** E.EmpID = C.EmpID **WHERE** Children.Sex = 'Male' **AND** DeptName = 'Marketing'	

المحاضرة الحادية عشر

الإجرائيات المخزنة
Stored Procedures

الهدف من الجلسة

في نهاية هذه الجلسة سوف نكون قد عرضنا مفهوم الإجرائيات المخزنة، التعامل مع الإجرائيات المخزنة.

الكلمات المفتاحية

STORED PROCEDURE,
CREATING,
DELETING,
MODIFYING,
EXECUTING.

سوف نتعرف في هذه الجلسة على:

١. مقدمة.
٢. إنشاء إجرائية مخزنة Creating Stored Procedure.
٣. تحديد وسطاء إجرائية Specifying Parameters
   a. تحديد الأسماء Specifying a Name
   b. تحديد الأنماط Specifying a Data Type
   c. تحديد اتجاه الوسطاء Specifying the Direction of a Parameter
   d. تحديد القيم الإفتراضية Specifying a default Value
٤. برمجة الإجرائيات المخزنة
   a. تضمين الإجرائيات المخزنة Nesting
   b. تأجيل تحديد الأسماء والمعالجة Deferred Name Resolution and Compilation
٥. استعادة بيانات من الإجرائيات المخزنة Returning Data From Stored Procedures.
   a. استعادة بيانات باستخدام OUPUT.
   b. استعادة بيانات باستخدام RETURN.
٦. تنفيذ الإجرائيات المخزنة Executing.

٧. تعديل وإعادة تسمية الإجرائيات المخزنة Modifying and Renaming.

٨. إعادة ترجمة الإجرائيات المخزنة Recompiling.

٩. استعراض إجرائية مخزنة Viewing.

مقدمة :

TSQL: Transact- SQL هي لغة برمجة تشكل الواجهة الرئيسية بين أي نظام يتعامل مع قواعد البيانات وقاعدة البيانات.

عند استخدام TSQL هناك طريقتين لتخزين وتنفيذ البرامج:

a. يمكن تخزين البرامج محليا Locally. ومن ثم إنشاء برامج تقوم بإرسال البرامج المخزنة محليا إلى مخدم قواعد البيانات للتنفيذ من ثم معالجة النتائج.

b. يمكن تخزين البرامج على مخدم قواعد البيانات Stored Procedures، ومن ثم إنشاء برامج تنفذ هذه البرامج المخزنة ومن ثم معالجة النتائج.

الإجرائيات المخزنة في مخدم قواعد المعطيات **تشبه** الإجرائيات في لغات البرمجة من حيث:

١. لها **متحولات دخل** parameters، وتعيد مجموعة من القيم على شكل **متحولات خرج** Output Parameters إلى البرنامج الذي قام بتنفيذ الإجرائية.

٢. تحوي على **مجموعة من التعليمات** Statements التي تقوم بمجموعة من **العمليات** Operations على قاعدة البيانات، كما يمكن **استدعاء إجرائية مخزنة من قبل إجرائية مخزنة أخرى**.

٣. تعيد الإجرائيات المخزنة عند التنفيذ **متحول حالة** Status Parameter **لتعلم** البرنامج المُستدَعِي **عن نجاح أو فشل العملية**.

ما هي **فوائد تخزين** الإجرائيات على **مخدم قواعد البيانات** بدلا من تخزين الإجرائيات بشكل **محلي**:

١. تسمح الإجرائيات المخزنة **بالبرمجة الكتلية** Modular Programming: يتم إنشاء الإجرائية مرة واحدة، تخزن على مخدم قواعد المعطيات، لتيم استدعاؤها عدد من المرات.

٢. **تنفيذ أسرع للإجرائيات** Faster Execution: يتم **تهجئة** Parsing الإجرائية، **تحسن الأداء** Optimization، وتنفيذ نسخة من الإجرائية موجودة في الذاكرة **In Memory** بعد تنفيذها لأول مرة.

٣. **تخفيف الضغط على الشبكة** Reduce Network Traffics: إذا كان عملية تتطلب تنفيذ مئات من التعليمات على مخدم قواعد البيانات، فإن الإجرائيات المخزنة تساعد بتنفيذها بالإسم.

٤. **يمكن فرض قيود على التنفيذ** Execution Security Mechanism: يمكن منح **صلاحيات تنفيذ** للإجرائيات المخزنة.

إنشاء إجرائية مخزنة Creating Stored Procedure:

يمكن إنشاء إجرائية مخزنة باستخدام التعليمة CREATE PROCEDURE. قبل إنشاء إجرائية مخزنة يجب أخذ بعين الاعتبار النقاط التالية:

١. لا يمكن أن تحوي مجموعة التعليمات BATCH أوامر أخرى غير التعليمة CREATE PROCEDURE. مثلا مجموعة التعليمات التالية **لا تنفذ**:

```
USE NORTHWIND
CREATE PROCEDURE TEMP
```

٢. يجب ان يملك **صلاحية مالك قاعدة بيانات** DBOWNER، كل من يريد أن ينشأ إجرائية مخزنة.

٣. يجب تحديد **اسم الإجرائية المخزنة**. الإجرائية المخزنة هي غرض من أغراض قاعدة البيانات التي يتبع اسمها **قواعد تسمية المتحولات**.

٤. يمكن إنشاء إجرائية مخزنة في **قاعدة البيانات الحالية**.

عند إنشاء إجرائية يجب تحديد النقاط التالية:

١. **اسم الإجرائية** Procedure Name

٢. **معاملات الدخل** Input Parameters

٣. **معاملات الخرج** Output Parameters

٤. **القيمة المعادة** إلى الإجرائية المستدعية للإجرائية الحالية والتي تدل على **فشل أو نجاح** الإجرائية. كما يجب تحديد **رسالة الخطأ المرافقة للقيمة المعادة**.

تحديد وسطاء إجرائية Specifying Parameters

تتخاطب الإجرائية مع البرنامج الذي يستدعيها عبر مجموعة **وسطاء** Parameters. حيث تمكن الوسطاء البرنامج من تمرير قيم **إلى** الإجرائية Input Parameters كما تمكنه من الحصول على قيم **من** الإجرائية Output Parameters.

تحديد الأسماء Specifying a Name

يجب أن تكون وسطاء الإجرائية **وحيدة**. (أي أنه لا يمكن أن ننشئ إجرائية تحوي على **وسيطين بنفس الإسم**). جميع وسطاء الإجرائيات يجب أن تبدأ بـ @. كما أن وسطاء الإجرائيات تتبع في التسمية قواعد تسمية المتحولات (تبدأ بحرف، تحتوي على أحرف أو أرقام أو...).

عند استدعاء إجرائية يمكن أن تمرر القيم إلى الإجرائية بطريقتين.

مثال لنفرض أننا أنشأنا إجرائية لها الوسطاء التالية : @first, @second, @third. كما أنه اسم الإجرائية هو temp. يمكن تمرير قيم إلى الإجرائية بإحدى الطريقتين التاليتين:

a. **Specifying Parameters' Names**:

EXECUTE temp @second = 2, @first = 1, @third = 3

b. **Without Specifying Parameter Names**:

EXECUTE temp 2, 1, 3

في الحالة الثانية تكون قيم وسطاء الدخل : @first = 2, @second = 1, @third = 3

تحديد الأنماط Specifying a Data Type

يتم تحديد **نمط لكل وسيط لإجرائية**. تماما كما نعرف نمط اسم حقل في جدول. عند استدعاء إجرائية يجب أن تكون **القيم** التي تمرر إلى الوسطاء **تحترم نمط وحجم الوسيط**. **مثال** إذا كان نمط وسيط هو tinyint فإن الوسيط يجب أن يأخذ قيم طبيعية تخزن على 1 Byte.

تحديد اتجاه الوسطاء Specifying the Direction of a Parameter

يمكن للإجرائيات المخزنة أن تتلقى قيم عبر الوسطاء من قبل البرنامج المستدعي لها.

مثال:

في قاعدة البيانات Pubs، اكتب إجرائية تقوم بإيجاد مبيعات كتاب title.

إن تحليل هذا الطلب يتطلب كتابة إجرائية وليكن اسمها get_sales_for_title، لهذه الإجرائية وسيط **هو اسم الكتاب**@Title، وهو وسيط دخل يتم تزويد قيمته من قبل البرنامج المستدعي للإجرائية. وعليه تكون تعلمية إنشاء الإجرائية على الشكل التالي:

```
-- Ex. 01
use pubs
GO
CREATE PROCEDURE get_sales_for_title
@title varchar(80) -- This is the input parameter.
AS

BEGIN

-- Get the sales for the specified title.

SELECT "YTD_SALES" = ytd_sales

FROM titles

WHERE title = @title

RETURN

END
```

إن الوسيط @title في المثال السابق هو وسيط دخل، لجعل وسيط لإجرائية **وسيط خرج يجب** إضافة الكلمة OUTPUT إلى تعريف الوسيط، كما **سنرى لاحقا**.

تحديد القيم الإفتراضية Specifying a default Value

يمكن إنشاء إجرائيات مخزنة بوسطاء **اختيارية** Optional. يتم ذلك عبر تحديد **قيمة افتراضية للوسطاء**. يتم ذلك كما في المثال التالي:

مثال: في قاعدة البيانات Pubs، اكتب إجرائية تقوم بإيجاد مبيعات كتاب title. إذا **لم يتم تحديد إي كتاب فإن الإجرائية تعيد مبيعات جميع الكتب**.

```
-- Ex. 02
use pubs
GO
CREATE PROCEDURE get_sales_for_title_or_titles
@title varchar(80)= NULL
-- NULL is the default value.
AS
BEGIN
IF @title IS NULL
BEGIN
SELECT

Title,

"YTD_SALES" =
ytd_sales
FROM

titles
      RETURN
END
-- Get the sales for the specified title.
SELECT "YTD_SALES" = ytd_sales
FROM titles
     WHERE title = @title
RETURN
END
```

برمجة الإجرائيات المخزنة

قواعد لبرمجة الإجرائيات المخزنة Rules for Programming Stored Procedures:

- **لا يمكن تضمين** تعليمات الإنشاء CREATE التالية في إجرائية مخزنة :

CREATE PROCEDURE
CREATE TRIGGER
CREATE VIEW
CREATE RULE
CREATE DEFAULT

غير الذي ذكر في الجدول السابق، **يمكن تضمين** تعليمة CREATE لأي غرض من أغرض قاعدة بيانات.
يمكن استخدام أي غرض تم إنشاؤه ضمن إجرائية مخزنة ما دام الإنشاء يتم قبم الإستخدام.

- يمكن استخدام الجداول المؤقتة ضمن الإجرائيات. الجدول المؤقت هو جدول يبدأ اسمه بـ #.
- إذا تم إنشاء جدول مؤقت ضمن إجرائية فإن هذا الجدول يزول بإنتهاء الإجرائية من التنفيذ.
- العدد الأعظمي للوسطاء في إجرائية مخزنة 2100 وسيط.
- الحد الأعظمي للمتحولات المحلية ضمن إجرائية محدود فقط بسعة الذاكرة المتوفرة.
- الحجم الأعظمي لإجرائية مخزنة 128MB.

إن **استخدام أغراض في إجرائية تحوي تعليمات** INSERT, UPDATE, DELETE, SELECT **تسبق أسماء هذه الأغرض بشكل افتراضي** بـ DBO حيث DBO هو مالك قاعدة البيانات (DATABASE OWNER) **مثال**: الجدول TITLES ضمن إجرائية يسبق بـ DBO ليصبح DBO.TITLES.

لنفرض أن المستخدم Mary قام بإنشاء الجدول Marks. فإن الجدول في قاعدة البيانات سيصبح اسمه Mary.Marks. وبالتالي استخدام الجدول Marks محصور بـ Mary.

فإذا حاول المستخدم John تنفيذ إجرائية تحوي تعليمة select من الجدول Marks فإن مخدم قواعد البيانات سوف يبحث عن جدول اسمه John.Marks وبالتالي سوف يرسل مخدم قواعد البيانات رسالة مفادها أن الجدول غير موجود.

إذا أردت أن **تشفر الإجرائية** التي قمت بإنشائها، بحيث لا يستطيع أحد رؤية محتوى الإجرائية فإن ذلك ممكنا عبر استخدام التعليمة WITH ENCRYPTION.

مثال:

```
-- EX 03
USE NORTHWIND
GO
CREATE PROCEDURE MyProc WITH ENCRYPTION
AS
BEGIN
SELECT * FROM EMPLOYEES
END
```

تضمين الإجرائيات المخزنة Nesting

التضمين هنا بمعنى **الإستدعاء**. (كنا قد ذكرنا أنه لا يمكن أن نستخدم تعليمة CREATE PROCEDURE ضمن إجرائية) . يمكن أن يصل **عمق الإستدعاء لـ 32 مستوى**.

هناك متحول عام @@NESTLEVEL يحوي في كل لحظة مستوى التضمين الحالي. إذا **تجاوز** مستوى التضمين ٣٢ ، فإن مخدم قواعد البيانات يقوم بإيقاف سلسلة الاستدعاءات ويعطي رسالة **خطأ**.

استعادة بيانات من الإجرائيات المخزنة Returning Data From Stored Procedures.

تعيد الإجرائيات المخزنة البيانات إلى البرنامج المستدعي **بأربع** طرق:

- **وسطاء الخرج** OUTPUT PARAMETERS.
- **القيمة المعادة** RETURN CODE والذي هي دوما قيمة INTEGRAL.
- **من أجل كل** تعليمة **اختيار** SELECT مضمنة في إجرائية هناك **مجموعة نتائج** RESULT SET مقابلة تعبر عن **خرج للإجرائية**.
- **مؤشر عام** GLOBAL CURSOR يمكن أن يستخدم كخرج لإجرائية.

استعادة بيانات باستخدام OUPUT.

مثال: يظهر المثال التالي إجرائية بوسيط دخل ووسيط خرج. المتحول الأول هو @title وهو وسيط دخل، الوسيط الثاني هو وسيط خرج @ytd_sales الذي يمثل مجموعة المبيعات حتى الوقت الحاضر بالنسبة للكتاب @title:

```
--  ex 04

use pubs
GO
DROP PROCEDURE get_sales_for_title
GO
CREATE PROCEDURE get_sales_for_title
@title varchar(80), -- input
@ytd_sales int output -- output parameters
AS
BEGIN
-- Get the sales for the specified title.
SELECT

@ytd_sales = ytd_sales
FROM

titles
WHERE

title = @title
RETURN (0)
END
```


استعادة بيانات باستخدام RETURN.

يمكن لإجرائية مخزنة أن تعيد إلى البرنامج المستدعى قيمة INTEGRAL باستخدام تعليمة RETURN. إن القيمة المعادة باستخدام هذه التعلمية تخزن في متحول يسند إليه تنفيذ الإجرائية:

مثال: في المثال السابق للحصول على القيمة الراجعة عن التعليمة **RETURN** نعرف متحول عام من نمط **int** نسند إليه تنفيذ الإجرائية :

```
DECLARE @RES INT
EXEC @RES = GET_SALES_FOR_TITLE …….
```

تنفيذ الإجرائيات المخزنة Executing.

كما هو واضح من الأمثلة السابقة، فإنه لتنفيذ إجرائية مخزنة نستخدم التعليمة **EXECUTE** أو EXEC. في الحالة التي يكون فيها تنفيذ الإجرائية هو أو ل تعليمة فإنه لا حاجة لاستخدام كلمة EXECUTE.

مثال:

لتنفيذ الإجرائية المعرفة في المثال EX 04: (بالطبع هناك طرق أخرى لتنفيذ إجرائية)

- نعرف متحول لكل متحول دخل للإجرائية
- نعرف متحول لكل متحول خرج للإجرائية
- نعرف متحول نسند إليه القيمة الراجعة من تنفيذ الإجرائية .
- نسند قيم لمتحولات الدخل.
- ننفذ الإجرائية

```
-- EX 05
DECLARE @ytd_sales INT
DECLARE @title VARCHAR(30)
DECLARE @res INT
SET @title = 'Sushi, Anyone?'
EXEC @res = get_sales_for_title @title, @ytd_sales OUTPUT

IF @ytd_sales IS NULL
PRINT 'null'
ELSE
PRINT ' sales for ' + @title +
CONVERT(VARCHAR(6),@ytd_sales)

PRINT ' the result of return is ' + convert(varchar(6), @res)
```

تعديل وإعادة تسمية الإجرائيات المخزنة Modifying and Renaming.

لحذف إجرائية مخزنة نستخدم التعليمة DROP.

مثال:

```
-- EX 06
DROP PROCEDURE get_sales_for_title
```

لإعادة تسمية إجرائية تحذف الإجرائية ثم تنشأ من جديد. هذا الأمر يؤدي إلى حذف جميع الصلاحيات التي أعطيت للإجرائية.

يمكن استخدام التعليمة ALTER PROCEDURE لتعديل إجرائية دون الحاجة إلى استخدام الأمر DROP PROCEDURE.

إعادة ترجمة الإجرائيات المخزنة Recompiling.

يمكن إعادة ترجمة الإجرائيات المخزنة بالتعليمة التالية: sp_recompile.

كما يمكن تحديد with recompile عند إنشاء الإجرائية. الأمر الذي يعني أن الإجرائية سوف يعاد ترجمتها في كل مرة تنفذ.

```
-- ex 07
sp_recompile  get_sales_for_title

CREATE PROCEDURE test WITH RECOMPILE
AS
BEGIN
RETURN 0
END
```

استعراض إجرائية مخزنة Viewing.

لعرض إجرائية مخزنة نستخدم التعليمة التالية: sp_helptext.

مثال:

```
-- ex 08
sp_helptext get_sales_for_title
```

الفهارس Indexes والمناظير Views

الفهارس Indexes:

ما هو الفهرس؟

لو أردنا مثلاً أن نبحث عن اسم شخص في جدول الأشخاص Persons عبر استخدام الجملة التالية:

```
SELECT *

FROM Persons

WHERE First_Name = "Ibrahim";
```

فإن نظام إدارة قاعدة البيانات سيمرّ على كل السجلات الموجودة في الجدول لترشيح السجلات وإرجاع تلك التي توافق الشرط في جملة where.

ستظهر لنا مشكلة الوقت اللازم لتنفيذ جملة الاستعلام – وتزداد -كلما زاد عدد السجلات في الجدول، فلو كان لدينا مثلا مليون سجل في الجدولPersons ، ولنفترض جدلاً أن النظام باستطاعته المرور على 10 آلاف سجل في الثانية، فإننا بحاجة إلى 100 ثانية لتنفيذ جملة الاستعلام السابقة.

لحل المشكلة السابقة، فإن نُظم إدارة قواعد البيانات تقدم خاصية الفَهْرَسة.

فهرس قاعدة البيانات : هو عبارة عن هيكلية بيانات هدفه تحسين سرعة عملية استرجاع البيانات من جدول في قاعدة البيانات على حساب البطيء الناتج عن تخزين البيانات و زيادة الحجم المستهلك للتخزين.

الفَهْرَسة بشكل أبسط هي عبارة عن مؤشر يحتوي على نسخة من جزء من البيانات في الجدول، بحيث تقوم هذه النسخة من البيانات بمهمة "الدليل" أو "المُؤَشّر" الذي يسرع الوصول إلى البيانات الأصلية الكاملة الموجودة في الجدول، بحيث لا تحتاج المرور الكامل على كل الجدول (No Full Table Scan) عند البحث عن البيانات.

يمكن أن تخلق الفهارس باستخدام عامود أو أكثر من أعمدة جدول في قاعدة بيانات، و بذلك يتم تحسين عمليتي الحصول على بيانات بشكل سريع و عشوائي و الوصول للسجلات المرتبة. المساحة التي يحتاجها الفهرس للتخزين على القرص عادة ما تكون أقل من المساحة التي يحتاجها الجدول، ذلك لأن الفهارس تحتوي فقط على الأعمدة الأساسية التي يجب أن يُرتّب الجدول بناء عليها، و لا تحتوي على بقية أعمدة الجدول.

يعدّ الفهرس عملياً طريقةً من طُرُق تراكيب البيانات، وهو عنصر مرتبط بوجود جدول في قاعدة البيانات، ولكن نستطيع تعريفه وحذفه منفصلا عن تعريف الجدول، ولا يكون له أي تأثير على نفس البيانات، فعند حذف الفهرس، فإن البيانات الموجودة في الجدول لا تتأثر.

ملاحظة: المستخدم لا يرى الفهارس وإنما هي أسلوب هيكلة لتسهيل البحث وتريع تنفيذ الاستعلام.

كيف تُعرَّف الفهارس؟

يُعرَّف الفهرس بطريقتيْن:

- تعريفه ضمنيًّا: تُبنَى الفهارس ضمنيا على الأعمدة التي يُطَبَّق عليها القيد الفريد وقيد المفتاح الرئيسي، فعند تعريف أحد القيود السابقة، يُبنى فهرس تلقائيًّا على العمود أو الأعمدة المُقيَّدة.
- تعريفه صراحةً: يُبنَى الفهرس بطريقة مباشرة على العمود أو الأعمدة الذي نرغب وذلك باستخدام جملة Create Index.

ملاحظة : على الرغم من أنه لا يوجد تعريف للفهرس في معايير SQL، إلا أن أغلب أنظمة إدارة قواعد البيانات تقدم الإمكانية لتعريف الفهرس ويتفق أغلبها على الصيغة العامة لذلك.

الصيغة العامة لتعريف الفهرس :

```
CREATE INDEX index_name

ON table_name (column1, column2, ...);
```

ملاحظة : عند تعريف الفهرس، لابد أن يكون اسمه متوافقا مع القيود الخاصة بنظام إدارة قاعدة البيانات المستخدم، كما أنه يجب ألا يكون مُكررا، فأسماء الفهارس في قواعد البيانات يجب أن تكون فريدة ولا تتكرر.

مثال : لو أردنا إنشاء فهرس لتسريع عمليات البحث اعتماداً على عمود الاسم الأخير من جدول الموظفين emp :

```
CREATE INDEX idx_lastname

ON Persons (LastName);
```

لإنشاء فهرس من أجل أكثر من عمود فهرس يعتمد على عامودي الاسم الأول والاسم الأخير :

```
CREATE INDEX idx_pname

ON Persons (LastName, FirstName);
```

حذف الفهارس :

الصيغة العامة لحذف الفهرس كالتالي:

```
DROP INDEX droptable.Index_Name ;
```

متى نستخدم الفهارس؟

يفضل أن يتم بناء الفهارس على الأعمدة التي:

- يُبحث عنها في جملة Where.
- تُكتَب في جملة الترتيب Order By.
- تُكتَب في جملة التجميع Group By.

- تُستخدَم في جمل الربط Joins.
- تُستخدَم في الدوال الإحصائية مثل min وmax average.

متى نتجنب استخدام الفهارس؟

لا تعدّ الفهارس مناسبة على الأعمدة التي:

- تحتوي على قيم فريدة قليلة مثل عمود الجنس (قيمتان فقط)، أو الحالة الاجتماعية.

- نادرة الاستخدام في جمل الاستعلام SELECT.

- التي تكون جزءًا من جدول ذي سجلات قليلة.

المناظير Views :

لكي لا تصبح عملية كتابة الصيغ في SQL معقدة في كل مره خاصة اذا كانت هذه الصيغ تُستخدم بكثرة يتم تضمين تعابير خاصة في SQL تمكننا من إنشاء جداول افتراضية او ما يسمى بالمنظار او View والتي تساعدنا في استعادة البيانات التي يرجعها استعلام معين, فهي توفر جداول افتراضية تحتوي على البيانات ضمن تشكيل مطلوب .

ويمكن معاملته كأي جدول من جداول قواعد البيانات اذ يكمن الفرق الوحيد بينه وبين الجداول في كون البيانات التي تحتويه مخزنة في جداول اخرى .

ايضا تساعد الجداول الافتراضية في منع المستخدمين من الوصول الى الجداول الاصلية في قواعد البيانات اذ يحتوي المنظار او الجدول الافتراضي على جزء من البيانات المتوفرة في الجدول الاصلي.

كما يساعدنا على اخفاء بعض البيانات ان اردنا مثلا عرض جدول موظف ولا نريد عرض راتبه نقوم بإنشاء جدول افتراضي يحتوي على بيانات الموظف كاملة عدا حقل الراتب ونتيح لأي موظف اخر التعامل مع هذا الجدول على انه جدول الموظف الحقيقي.

عادةً لا يمكن تعديل البيانات من خلال جدول افتراضي فهو يشكل نسخة بيانات قابلة للقراءة فقط يمكن تخصيصها للمستخدمين ذوي الصلاحيات المنخفضة.

لإنشاء المنظار :

```
CREATE VIEW view_name AS query;
```

حيث view_name هو اسم المنظار المنشئ أي اسم الجدول الافتراضي أما query فهو الاستعلام الذي سيجلب حقول محددة من الجدول وقد تحقق شرط معين أيضاً.

مثال :

لنفرض انه لدينا استعلام معقد نسبياً يعيد قيم من جدولين مرتبطين بأسلوب Inner Join واردنا انشاء جدول افتراضي ليحتوي على قيم الحقول المعادة من هذا الاستعلام ستكون الصيغة كما يلي :

```
CREATE VIEW MySimpleView Projects.projectName,
count (Tasks.taskID) AS TasksNumber
From Tasks Inner Join Projects
ON Tasks.projectID = Projects.projectID
Group by projectName;
```

بذلك نكون أنشئنا جدول افتراضي باسم MySimpleView فيه حقلين حقل اسم المشروع وحقل عدد المهام الخاصة به , بعد انشائه يمكننا الاستعلام منه ببساطة :

```
Select projectName from MySimpleView;
```

وهذه تعتبر من فوائد الجداول الافتراضية وهي تبسيط الاستعلامات المعقدة كما اسلفنا سابقا وذلك بتجميع البيانات من عدة جداول وكتابة الاستعلام في جدول افتراضي دون استخدام الربط بين جدولين كلما احتجنا ذلك والمثال السابق يبين ذلك.

تعديل البيانات عبر تعديل الجدول الافتراضي:

عادة ليس من الممكن تعديل البيانات عبر تطبيق التعابير Insert و Updateو Delete على الجدول الافتراضي ولكن يمكن تحقيق هذا الغرض في حالة توفرت الشروط التالية :

- يجب ألا يحوي الاستعلام الخاص بالمنظار أي تابع تجميعي كما يجب ألا يستخدم تعبير Group By .
- يجب الا يحتوي الاستعلام الخاص بالجدول الافتراضي التعبير Top او Distinict .
- يجب الا يحتوي الاستعلام الخاص بالجدول الافتراضي حقول تم اجراء عمليات حسابية عليها او تم حسابها .

لتعديل بنية جدول افتراضي نستخدم الصيغة :

```
ALTER VIEW viewName AS newQuery;
```