مخبر
قواعد البيانات
المحاضرة الأولى

ما هي قواعد البيانات؟؟..

- ما هي البيانات؟

  هي كل ما يصف شيء ما كاسم شخص ما وعنوانه ورقمه الوطني كلها بيانات ترتبط بهذا الشخص و.....

- قاعدة البيانات

هي مجموعة البيانات المرتبة، بشكل منطقي وتسلسلي واضح، تربطها علاقات فيما بينها، تترتب على شكل جداول من جدول فأكثر، في كل جدول مجموعة من الصفوف والأعمدة، والتي تميز البيانات التي تكون فيه.

- ويتم استرجاع هذه البيانات عند الحاجة لاستخدامها في إدارة المنشآت أو ما يعرف بـ ERP (Enterprise Resource Planning) أو في أنظمة اتخاذ القرار (Decision Making) أو شركات الاتصالات والبنوك وغيرها الكثير من المجالات.

أنظمة إدارة قواعد البيانات
Data Base Management System

- يعرف نظام إدارة قواعد البيانات على أنه مجموعة من البرامج التي تتيح للمستخدم الوصول إلى قاعدة البيانات, معالجتها, والمساعدة في استرجاعها, كما تتيح إمكانية التحكم لسماحيات كل مستخدم لضمان سلامة البيانات.
- تنقسم قواعد البيانات إلى عدة أنواع ومن أهمها:

1- قواعد البيانات المسطحة Flat ومن الأمثلة عليها ملفات Word – Excell.

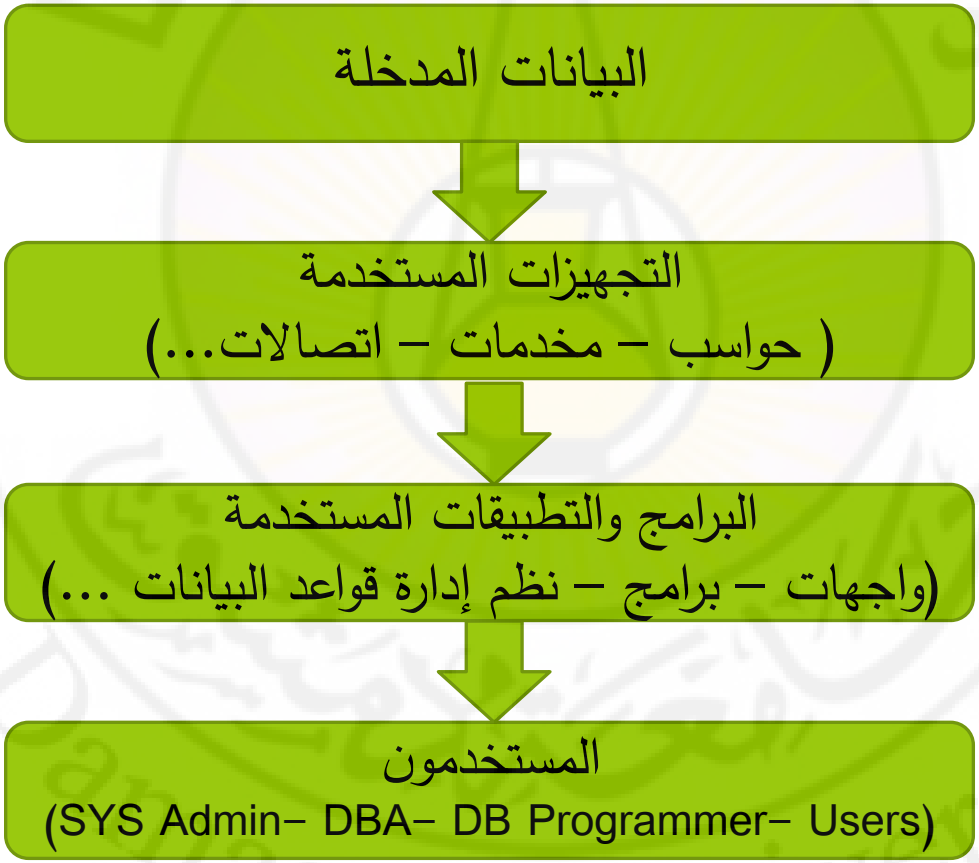
2- قواعد البيانات الشجرية Hierarchical: ومن الأمثلة عليها سجلات النظام registry

3- قواعد البيانات العلائقية Relational وهي الأكثر استخداماً, ومن الأمثلة عليها قواعد بيانات Access.

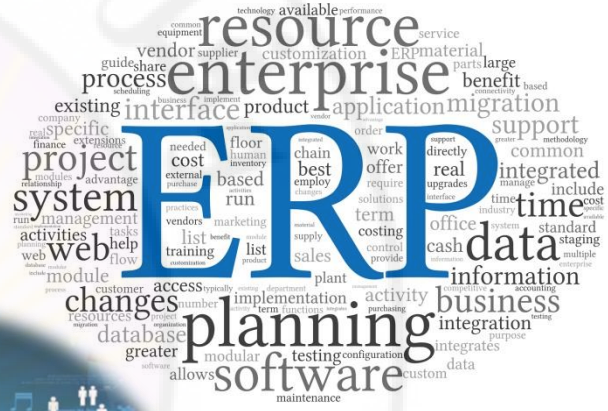
أنظمة إدارة قواعد البيانات العلائقية
Relational Data Base Management System

- يتم ضمن قواعد البيانات العلائقية تمثيل البيانات ضمن مجموعة من جداول, تربطها عدد من العلاقات.
- من أشهر نظم إدارة قواعد البيانات العلائقية:
  Oracle , MySQL, SQL Server ….
- يتم التعامل مع قواعد البيانات العلائقية بواسطة لغة الإستعلامات المعيارية Standard Query language والتي تختصر بـ (SQL).

مكونات قواعد البيانات



أمثلة على استخدام قواعد البيانات



فوائد قواعد البيانات

1. **تخزين البيانات:** يضمن وجود بيانات مخزنة يستند عليها برنامج المعلومات الذي يعالج بيانات المؤسسة أو المنظمة مثل بيئة ERP.

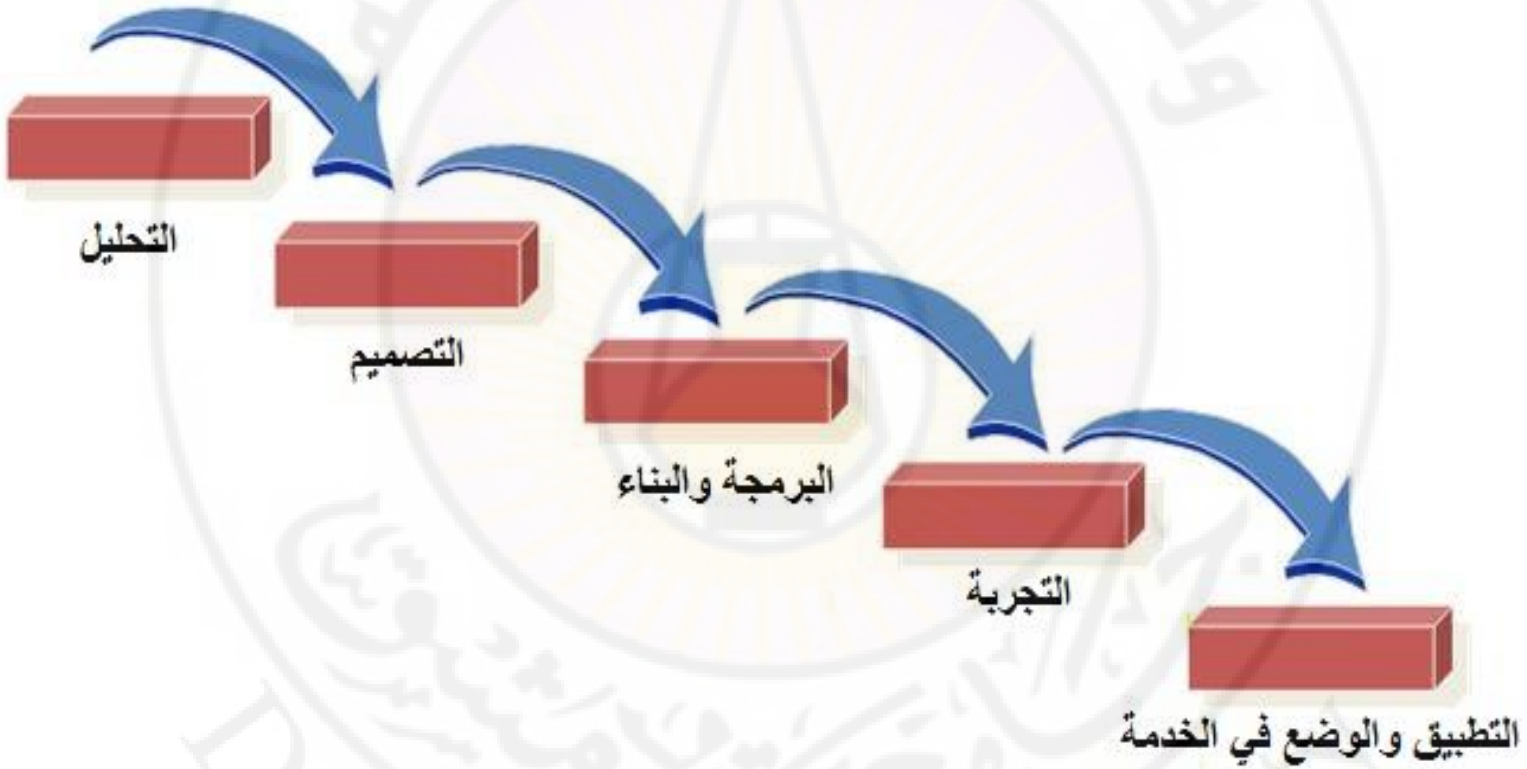
2. **توفير المساحات المستخدمة:** يمنع تكرار وازدواجية البيانات الموجودة بالملفات المختلفة مما يقلل من المساحة التخزينية وبالتالي التكلفة وذلك بربط المواقع المختلفة بالموقع الأصلي للمعلومة.

3. **سرعة الوصول واسترجاع البيانات:** يوفر مزيد من السرعة والدقة التي تمكن المستخدم من إضافة البيانات ,تعديلها ,والإستعلام والإستفسار عن جزء أو كامل البيانات.

4. **الأمان والسرية:** يوفر مزيد من سرية البيانات (الدخول بكلمات سر وصلاحيات).

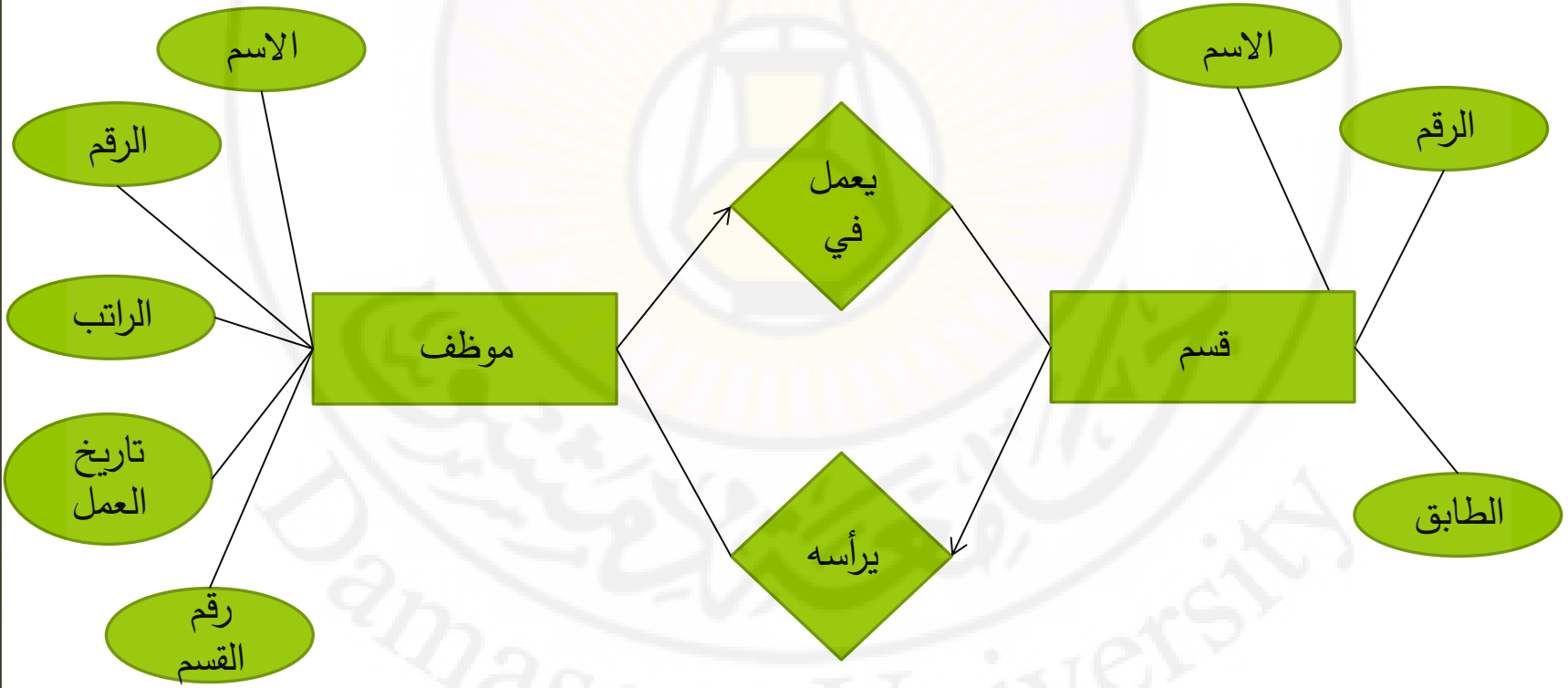
5. **تكامل البيانات:** يمكن من تكامل قاعدة البيانات في النظام المعين مع النظم الفرعية .

مراحل تطوير قواعد البيانات



تحليل المعطيات Data modeling

- هنا نقوم بمعرفة الكينونات التي لدينا بالإضافة إلى العلاقات فيما بينها, أي أننا نحصل على ERD (Entity Relation Diagram) .



تصميم قاعدة البيانات Database Design

- نقوم بتحويل ما توصلنا إليه في ERD إلى Database objects (جداول والقيود وفهارس و...) وتتعلق هذه objects بالبيئة المستخدمة للتصميم مثل Oracle أو SQL server أو MS Access أو....

- جدول الموظفين

الاسم	الرقم	الراتب	تاريخ العمل	رقم القسم	تاريخ الميلاد

- جدول الأقسام

اسم القسم	رقم القسم	الطابق

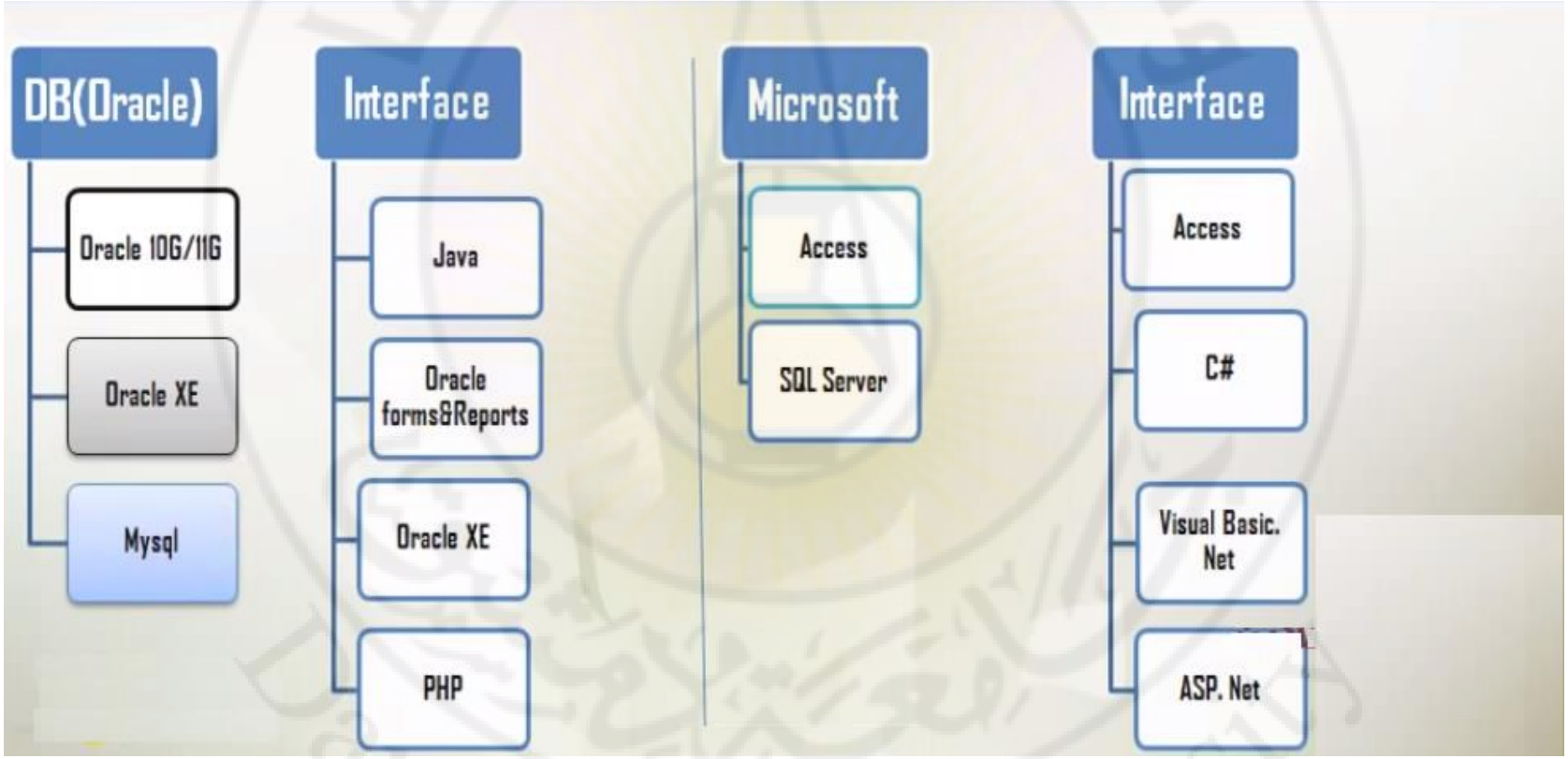
بناء قاعدة المعطيات Database Build

- هنا نقوم باستخدام واجهة التخاطب التي يؤمنها لنا محرك قاعدة البيانات مثلاً Access لبناء قاعدة البيانات وتشمل المراحل التالية:

- بناء الجداول مع وضع القيود المناسبة
- تحديد العلاقات بين الجداول
- كتابة البرامج اللازمة حسب لغة البرمجة المستخدمة
- تصميم واجهات البرامج
- تأمين الاتصالات (إعدادات شبكية وإنشاء حسابات وصلاحيات)بين المخدم الرئيسي والحواسب الفرعية

واجهات قواعد البيانات Database Interface

الواجهات ونظم قواعد البيانات



SQL , MySQL

Next Week.....

مخبر
قواعد البيانات
المحاضرة الثانية

SQL(Structured Query Language)

- تُعرَّف لغة الاستعلام المهيكلة SQL بأنها لغة التعامل مع قواعد البيانات وتعتمد عليها كافة التطبيقات التي تتعامل مع قواعد البيانات العلائقية.

إذاً هي لغة تستخدم لإصدار جميع الأوامر التي تتعلق بقاعدة البيانات، حيث تعمل SQLبمبدأ توجيه طلب إلى محرك قاعدة البيانات والحصول على جواب من محرك قاعدة البيانات الذي يُرجع مجموعة نتائج.

قواعد البيانات العلائقية:

- الكينونة ويتم تمثيلها بالجدول.
- العلاقة ويتم تمثيلها بآلية الربط بين الجداول (مفاتيح).
- الخصائص ويتم تمثيلها بالأعمدة أو حقول الجدول.
- الصفوف ويتم تمثيلها بالسجلات في الجداول.
- البيانات ويتم تمثيلها بالمعلومات المحتواة ضمن حقول الجدول.
- النطاق ويتم تمثيلها بالمجال الذي يمكن لقيم الخصائص أن تتحرك ضمنه.
- توابع تساعد على الوصول إلى البيانات وتساعد على تعديلها واسترجاعها.

أنظمة إدارة قواعد البيانات العلائقية

- من الأخطاء الشائعة، إطلاق اسم قواعد البيانات على أنظمة إدارة قواعد البيانات العلائقية.
فأنظمة MS SQL Server، وOracle، وMYSQL, MS Access هي أنظمة إدارة قواعد بيانات وليست قواعد بيانات.

من أهم أنظمة إدارة قواعد البيانات العلائقية :

- DB2
- Firebird
- Informix
- Microsoft SQL Server
- MySQL
- Oracle
- Ingres
- HSQLDB

SQL languages:

- لغة معالجة البيانات DML (Data Manipulation Language): والتي تتضمن الأوامر الخاصة باستعادة البيانات و تعديلها مثل:
  SELECT: وهي مخصصة لقراءة البيانات و استخلاصها من قاعدة البيانات.
  INSERT: وهي مخصصة لإضافة سجلات جديدة إلى قاعدة البيانات.
  DELETE: وهي مخصصة لحذف سجل أو مجموعة سجلات من قاعدة البيانات.
  UPDATE: وهي مخصصة لتعديل سجل أو مجموعة من السجلات في قاعدة البيانات.
- لغة تعريف البيانات DDL (Data Definition Language): المخصصة لتعريف بنية البيانات، وتتضمن تعليمات مثل:
  CREATE TABLE: وهي مسؤولة عن توليد جدول .
  DROP TABLE: وهي مسؤولة عن حذف جدول .
  ALTER TABLE: وهي مسؤولة عن تعديل جدول .
  CREATE INDEX: وهي مسؤولة عن توليد مؤشرات.
- لغة التحكم بالبيانات DCL (Data Control language) التي تُستخدم للتحكم و ضبط السماحيات على قاعدة البيانات مثل:
  GRANT: تستخدم لإعطاء الصلاحيات للمستخدمين للدخول على البيانات .
  REVOKE : تستخدم لسحب الصلاحيات من المستخدمين .

- MySQL is the world's most popular open source database
- Relational
- SQL-based
- Database Management System

العمل مع MYSQL:

- **إنشاء قاعدة بيانات:**
- CREATE DATABASE database-name;
- **استخدام قاعدة البيانات:**
- Use database-name;
- **لمعرفة قواعد البيانات الموجودة:**
- Show databases;
- **لمعرفة الجداول ضمن القاعدة المستخدمة:**
- Show tables;
- **للمساعدة:**
- help

إنشاء جدول وإدخال البيانات:

- إنشاء جدول:
- CREATE TABLE table-name (column1 data type, column2 data type,…);
- وعندما نريد عرض البناء الداخلي للجدول الذي تم إنشاؤه نقوم بكتابة الأمر التالي:
- DESCRIBE table-name;
- إدخال بيانات إلى الجدول المنشئ:

تُستخدم تعليمة INSERT لإدراج سجل في جدول محدد.

حيث تأخذ تعليمة INSERT الصيغة:

INSERT INTO table_name VALUES (value_1, value_2, …value_n)
value_3…value_n);

- حيث تُمثل value_1,..,value_n القيم التي سوف يتم تخزينها في السجل تبعاً لترتيب حقوله. كما يمكننا استخدام صيغة بديلة تُمكننا من تحديد حقول السجل التي نود إدراج البيانات فيها فقط:

- INSERT INTO table_name (field_1, field_2, …, field_n)
  VALUES (value_1,value_2,..,value_n);

قواعد البيانات العلائقية – مثال:

Emp-ID	F-name	L-name	Job	Dept-ID	E-mail	phone
1	Alaa	….	Engineer	10	….	5648
2	Ahmad	….	Technical	10	….	5689
3	Ola	….	Secretary	20	….	5622
4	Samar	….	IT	10	….	5612
5	maher	….	manager	30	….	5627

Emp-ID	Month	Salary	Bouns	Net Salary
1	Jan 2014	25000	2000	27000
2		20000	1000	21000
…….		……		
1	Feb 2014	25000	2500	27500

الاسقاط على لغة الـ SQL

- Create database company;
- Use company;
- CREATE TABLE Employee(

emp_id int primary key ,
f_name varchar(20) NOT NULL ,
l_name varchar(30) NOT NULL,
job varchar(30) NOT NULL,
Dep_id int NOT NULL
E-mail varchar(30),
phone int Default (0000));

تعليمة SELECT:

- 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.
- تأخذ تعليمة SELECT الصيغة العامة التالية:

```
SELECT    *   or   columns[alias]
  FROM       [table_name]
  WHERE        condition or conditions
 ORDER BY    columns or alias [Asc or Desc]
```

ملاحظات عامة:

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

ملاحظات عامة:

- يمكن كتابة مكونات جملة SQL بالأحرف الكبيرة أو الصغيرة فهذا لا يؤثر على سلامة الجملة وذلك لأن جملة SQL غير حساسة للحروف.
- يُفصل بين أسماء الحقول باستخدام الفاصلة (,).
- يمكن كتابة جملة SQL في عدة سطور فهذا لا يؤثر في صحة الجملة.
- لا يمكن استخدام الكلمات المحجوزة للغة أو اختصارها، والكلمات المحجوزة تسمى Keywords وهي مثل(select ,from ,where order by....).
- يفضل كتابة الجملة على أسطر ليسهل قراءتها وفهمها.
- تتطلب بعض برامج قواعد البيانات إنهاء كل تعليمة SQL بـ (;) ومنها MYSQL.

مخبر
قواعد البيانات
المحاضرة الثالثة

جملة الشرط (WHERE):

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

الصيغة **العامة لجملة الشرط WHERE**:

WHERE Column Comparison_Operator Value

```
Select * from employee
Where emp_id >=1000;
```

ملاحظات:

- عند استخدام قيم نصية أو قيم تعبر عن تاريخ لابد من وضعها داخل علامة تنصيص فردية ' '.
- في حالة استخدام القيم النصية لابد من مراعاة حالة الأحرف كبيرة أم صغيرة.
- في حالة استخدام قيم تعبر عن تاريخ لابد من مراعاة صيغة التاريخ المستخدمة علماً بأن الصيغة الأساسية للتاريخ داخل لغة SQL هي (YY-MON-DD) حيث (DD: تعبر عن اليوم،MON: تعبر عن الشهر،YY: تعبر عن السنة).

معاملات المقارنة المستخدمة في جملة الشرط WHERE:

- تستخدم معاملات المقارنة التالية للمقارنة بين طرفي الشرط في جملة WHERE:

معامل المقارنة	المعنى
=	يساوي
>	أكبر من
>=	أكبر أو يساوي
<	أصغر من
<=	أصغر أو يساوي
<> أو =!	لايساوي

- هناك معاملات مقارنة أخرى تستخدم في جملة الشرط WHERE ، وهذه المعاملات تسهل عملية حصر البيانات بشكل أكبر وهي كالتالي:

معامل المقارنة	المعنى
Between value And value	حصر البيانات بين رقمين
IN(مجموعة من القيم)	حصر البيانات ضمن مجموعة من القيم
LIKE{%,_}	حصر البيانات حسب مطابقة النص أو الحروف
IS NULL	حصر البيانات الخالية NULL

ملاحظات:

- معامل LIKE{%,_}: يستخدم للبحث عن نص معين أو حقل نصي، حيث يتم مطابقة حروف النص المذكورة في جملة الشرط.
- (%):هذا الرمز يعني أي حرف أو حروف, مثلاً ('A%') هذا التعبير يعني مطابقة النصوص التي تبدأ بحرف A مهما كانت باقي الحروف التالية له , فهو يستخدم للبحث عن نص يبدأ بالحرف A. والتعبير ('%A') يعني مطابقة النصوص التي تنتهي بحرف A مهما كانت الحروف التي تسبقه ويستخدم للبحث عن النصوص التي تنتهي بالحرف A.أما التعبير ('%A%')فهو يستخدم للبحث عن النصوص التي تحتوي على الحرف A.
- (_):هذا الرمز يعني مطابقة حرف واحد فقط ، فمثلاً ('A%_') يعني أنه بغض النظر عن الحرف الأول ويستخدم للبحث عن نص يكون الحرف الثاني له A،أما التعبير('A _ _') فهو يستخدم للبحث عن نص يكون الحرف الثالث فيه هو A.

تمارين عملية:

- تمرين 1:

ما هي عبارة SQL التي تقوم بعرض حقول أسماء ورواتب الموظفين الذين تنحصر رواتبهم بين 10000و 15000ل.س من جدول الموظفين EMP.

- SELECT name, salary from EMP where salary between 10000 and 15000;

- تمرين 2:

ما هي عبارة SQL التي تقوم بعرض حقول أرقام و أسماء ورواتب ورقم المديرية التي يتبع لها كل موظف من الموظفين بحيث تكون أرقام مديرياتهم (7788,7566,7902) من جدول الموظفين EMP.

- SELECT empno, name, salary, mgrno from EMP where mgrno In (7902, 7566, 7788);

- تمرين 3:

ما هي عبارة SQL التي تقوم بعرض أسماء الموظفين الذين يكون الحرف الثاني في أسمائهم هو F من جدول الموظفين EMP.

- SELECT name from EMP where name LIKE '_F%';

المعاملات المنطقية في جملة الشرط WHERE

تستخدم المعاملات المنطقية لتكوين أكثر من شرط في جملة WHERE ، وهي معاملات تربط بين جملتين شرطيتين أو أكثر وتكون النتيجة إما TRUE أي تحقق الشرط أو FALSE عدم تحقق الشرط.

المعامل	المعنى
AND	ترجع النتيجة TRUE إذا كانت جملتا الشرط TRUE
OR	ترجع النتيجة TRUE إذا كانت إحدى جملتي الشرط TRUE
NOT	تنفي النتيجة،أي ترجع النتيجة TRUE إذا كانت جملة الشرط FALSE

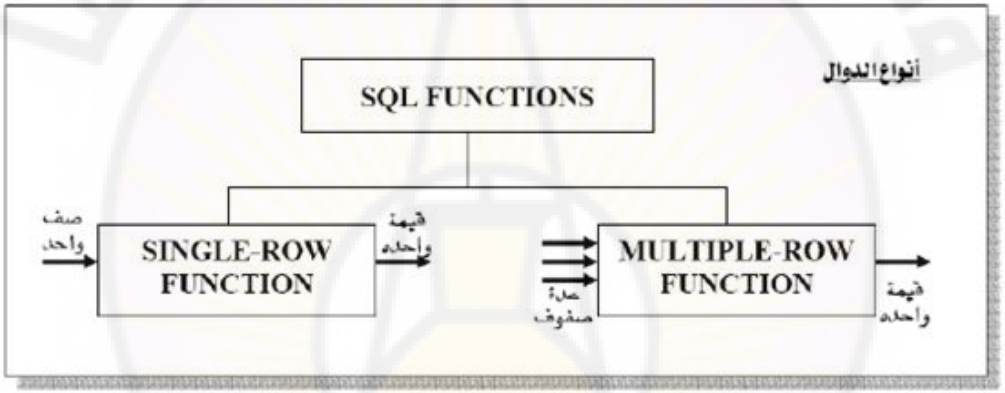
تمرين:

ما هي عبارة SQL التي تقوم بعرض رقم و أسماء ووظيفة ورواتب الموظفين الذين يزيد راتبهم عن 20000ل.س أوتكون وظيفتهم Manager من جدول الموظفين EMP.

- SELECT emp_id, name, job, salary from EMPLOYEE

  where salary >20000 OR job='Manager';

التوابع في SQL



التوابع التجميعية: وهي التوابع التي تأخذ كمُعاملات مجموعة من القيم وتعيد قيمة وحيدة(تقوم بالتعامل مع بيانات أكثر من صف لإخراج قيمة واحدة) مثل .....SUM–AVG–MAX–MIN

توابع الصف الواحد: وهي التوابع التي تأخذ مُعاملاً وحيداً وتُعيد قيمة وحيدة (تقوم بالتعامل مع بيانات صف واحد فقط لإخراج قيمة واحدة) مثل ....–ABS–FLOOR

توابع SQL التجميعية (SQL Aggregate Functions)

التابع AVG

يحسب تابع AVG المتوسط الحسابي لقيم حقل معين وذلك بتقسيم مجموع قيم هذا الحقل على عددها.

يُستخدم الخيار ALL للحصول على المتوسط الحسابي لجميع القيم بما فيها القيم المكررة. يُعتبر هذا الخيار هو الخيار التلقائي في حال عدم تحديد أي من الخيارين DISTINCT أو ALL.

يُستخدم الخيار DISTINCT للحصول على المتوسط الحسابي لجميع القيم، مع استبعاد أي تكرار لقيمة ما.

صيغة التابع :

- SELECT AVG ([ALL| DISTINCT] column_name)
  FROM table_name;

التابع COUNT

يحسب التابع COUNT عدد البيانات الموجودة في الجدول من أجل حقل معين.

يُستخدم الخيار ALL عندما نريد الحصول على عدد البيانات الموجودة في الجدول، بالنسبة لحقل معين، مع استبعاد القيم التي تساوي NULL. يُعتبر هذا الخيار هو الخيار التلقائي في حال عدم تحديد أي من الخيارين DISTINCT أو ALL.

يُستخدم الخيار DISTINCT عندما نريد الحصول على عدد البيانات الموجودة في الجدول، بالنسبة لحقل معين، مع استبعاد القيم التي تساوي NULL واستبعاد القيم المكررة.

يُستخدم الخيار * عندما نريد الحصول على عدد الصفوف الموجودة في الجدول، بما فيها البيانات ذات القيمة NULL.

صيغة التابع:

- SELECT COUNT ([* |ALL|DISTINCT] column_name)
  FROM table_name;

توابع SQL التجميعية (SQL Aggregate Functions)

التابع MIN والتابع MAX

يُعيد التابع MIN القيمة الصغرى من قيم حقل معين في حين يُعيد التابع MAX القيمة العظمى من قيم حقل معين.
صيغة التابعين:

- SELECT MIN (column_name) FROM table_name;
- SELECT MAX (column_name) FROM table_name;

التابع SUM

يحسب التابع SUM مجموع قيم حقل معين:
يُستخدم الخيار ALL عندما نريد الحصول على مجموع قيم حقل معين بما فيها القيم المكررة.
يُعتبر هذا الخيار هو الخيار التلقائي في حال عدم تحديد أي من الخيارين DISTINCT أو ALL.
يُستخدم الخيار DISTINCT عندما نريد الحصول على مجموع القيم حقل معين مع استبعاد أي تكرار في القيم.

صيغة التابع:

- SELECT SUM ([ALL|DISTINCT] column_name)
  FROM table_name;

تجميع البيانات (Data Aggregation) باستخدام GROUP BY:

عندما نتكلم عن التوابع التجميعية فلابد لنا أن نتساءل: هل نستطيع أن نطبق التوابع التجميعية على مجموعات جزئية من السجلات بدلاً من تطبيقها على كامل السجلات؟

فإذا كان لدينا جدول منتجات، وأردنا حساب مجموع كلف المنتجات التي نحصل عليها من المورّد الأول، وحساب مجموع كلف المنتجات التي نحصل عليها من المورّد الثاني، وحساب مجموع كلف المنتجات التي نحصل عليها من المورّد الثالث،

فإننا سنحتاج لكتابة ثلاث تعليمات منفصلة تعتمد على التابع التجميعي SUM،

```
SELECT SUM(unitCost) FROM products WHERE supplierID = 1;
SELECT SUM(unitCost) FROM products WHERE supplierID = 2;
SELECT SUM(unitCost) FROM products WHERE supplierID = 3;
```

لكن هل يمكننا أن نصل إلى نفس النتيجة بتعليمة واحدة فقط؟

تغطي SQL هذا المتطلب بآلية تساعد على تقسيم السجلات إلى مجموعات جزئية وتساعد على تطبيق التوابع التجميعية على كل مجموعة جزئية على حدى.

- لتجميع البيانات في SQL نستخدم تعليمة GROUP BY.

-
```
SELECT columnA, aggFunc (aggFuncSpec)
FROM table
WHERE WHERESpec
GROUP BY columnA;
```

تمرين عملي:

يصبح المثال السابق على الشكل التالي:

```
SELECT supplierID, SUM(unitCost)
FROM products
WHERE supplierID Between 1 AND 3
GROUP BY supplierID;
```

ملاحظات على استخدام الدوال التجميعية:

- عند كتابة أي عمود داخل قائمة select لابد من كتابته مع الجزء GROUP BY وذلك لأن الدوال التجميعية تتعامل مع عدة صفوف.
- يمكن استخدام الجزء GROUP BY لترتيب الصفوف مع الدوال التجميعية.
- لايمكن استخدام الدوال التجميعية في الجزء where ولكن نستخدم الجزء having بدلاً منها.

الكلمة المفتاحية HAVING

عندما نفكر في وضع شرط ما على استعلام معين، يرد إلى ذهننا استخدام الكلمة المفتاحية WHERE مع شرط مناسب.

ولكن هذا الحل البديهي يكون قاصراً في بعض الحالات مثل الحالة التي يكون فيها أحد عناصر الشرط تابعاً تجميعياً!!!

عندها يأتي دور الكلمة المفتاحية HAVING في حال أردنا أن نضع شرطاً على استعلام ما، بحيث يكون أحد عناصر الشرط تابعاً تجميعياً..

نستعمل HAVING ضمن الصيغة التالية:

```
SELECT columnA, aggFunc (aggFuncSpec)
FROM table
WHERE WHERESpec
GROUP BY columnA
HAVING filterCondition;
```

تمرين :

تريد شركة هاتف تجهيز قائمة بأسماء المستخدمين الذين تجاوزت اتصالاتهم فترة الساعة، لإجراء دراسة على النتائج، علماً أن المعلومات المتوفرة هي جدول باسم phoneCalls يحوي المعلومات المبينة أدناه:
إسم المستخدم (UserName)، عدد دقائق الاتصال (Minutes)، رقم الهاتف (PhoneNumber):

PhoneNumber	Minutes	UserName
091223358	3	Ahmad Ammar
099545465	8	Adel Massoud
...	...	...

» يمكن إظهار الجدول المطلوب باستخدام تعبير SQL التالي:

```
SELECT userName, Sum(minutes)
FROM phoneCalls
GROUP BY username
HAVING Sum(minutes) > 60;
```

تدريب:

بعد إنشاء جدول الموظفين الذي يحوي على الحقول (الرقم ،الاسم ،نوع العمل ،الراتب).
استعلم عن مجموع رواتب الموظفين حسب نوع العمل بشرط استبعاد الوظيفة sales واستبعاد مجموع الرواتب الأصغر من 9000 وترتيب المخرجات حسب مجموع الرواتب تنازلياً.

الحل:

```
Select job, sum(sal) from emp
Where job !="sales"
Group by job
Having sum(sal)>9000
Order by sum(job) desc;
```

التوابع التجميعية GROUP FUNCTION

➢ الأهداف :

١) فهم ومعرفة التوابع التجميعية لأكثر من صنف وانواعها.

٢) إنشاء مجموعات من البيانات باستخدام Group by .

٣) فهم ومعرفة جملة الشرط المستخدمة مع التوابع التجميعية Having .

تخدم التوابع التجميعية للتعامل مع بيانات مجموعة من الصفوف للحصول على قيمة واحدة فمثلاً لإيجاد مجموعة اتب الموظفين نستخدم التابع SUM كما موضح في الجدول :

ENAME	SAL
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLARK	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

➢ أنواع التوابع التجميعية : الجدول التالي يبين ذلك :

	وظيفتها	الدالة Function
١	دالة تستخدم لإيجاد المجموع لعدد من القيم	SUM
٢	دالة تستخدم لإيجاد اكبر قيمة من بين مجموعة من القيم	MAX
٣	دالة تستخدم لإيجاد أقل قيمة من بين مجموعة من القيم	MIN
٤	دالة تستخدم لإيجاد المتوسط الحسابي لمجموعة من القيم	AVG
٥	دالة تستخدم لإيجاد عدد القيم او عد الصفوف وهذه الدالة تتجاهل القيم الفارغة NULL عند عملية العد	COUNT
٦	دالة تستخدم لإيجاد الانحراف المعياري لمجموعة من القيم	STDDEV DEVIATION
٧	دالة تستخدم لإيجاد مقدار التباين (التشتت) لمجموعة من القيم	VARIANCE

وجميع هذه التوابع تتجاهل القيمة NULL داخل الأعمدة.

مثال (1) :

```
SQL > SELECT SUM(SAL) , MAX(SAL) , MIN(SAL), AVG(SAL)
  2     FROM emp
```

النتائج			
SUM(SAL)	MAX(SAL)	MIN(SAL)	AVG(SAL)
29025	5000	800	2073.21429

ملاحظة : التابعان (MAX , MIN) يتعاملان مع جميع البيانات فمثلاً عند التعامل مع المحارف تكون النتيجة حسب الترتيب الأبجدي لنلاحظ المثال التالي :

مثال (2) :

```
SQL > SELECT MAX(ename) , min(ename)
  2     FROM emp
```

النتائج	
MAX(ENAME)	MIN(ENAME)
WARD	ADAMS

مثال (3) :

```
SQL > SELECT AVG(NVL(comm , 0))
  2      FROM emp ;
```

النتائج

AVG(NVL(comm , 0))
157.14286

في المثال السابق تم حساب المتوسط الحسابي لمكافآت الموظفين ونلاحظ استخدام التابع NVL لتجاوز مشكلة القيمة NULL إذ ان التوابع التجميعية تتجاهل القيم NULL .

```
SQL > SELECT AVG(comm)
  2      FROM  emp ;
```

AVG(comm)
550

التعامل مع التابع COUNT :

- يوجد حالتين :

- التابع Count (*) لعد جميع الصفوف داخل الجدول بما فيها الصفوف المكررة والصفوف التي تحوي القيم NULL وإذا كانت الجملة تحوي عبارة WHERE فعندها تقوم بعد الصفوف حسب الشرط.
- التابع Count(column) لعد بيانات عمود معين مع تجاهل القيم NULL.

والأمثلة توضح :

مثال (4) :

```
SQL > SELECT COUNT(*),COUNT(comm), COUNT(DEPTNO)
  2      FROM    emp ;
```

النتائج

COUNT(*)	COUNT(comm)	COUNT(DEPTNO)
14	4	4

مثال (5) :

```
SQL > SELECT COUNT(comm) , COUNT(*)
  2       FORM emp
  3       WHERE deptno=30;
```

النتائج

COUNT(*)	COUNT(comm)
6	4

إنشاء مجموعات من البيانات باستخدام الجزء GROUP BY :

لنفرض أننا نريد إيجاد أكبر راتب في كل قسم لذلك نقوم بتقسم الجدول إلى أقسام ونوجد أكبر راتب في كل قسم ونحقق ذلك باستخدام **GROUP BY** والتي تقسم الجدول إلى مجموعات حسب عمود معين او أكثر:

مثال (6) :

```
SQL > SELECT deptno , AVG(sal)
  2       FROM emp
  3       GROUP BY deptno
  4       ORDER BY AVG(sal) ;
```

النتائج

Deptno	AVG(sal)
30	1566.66667
20	2175
10	2914.66667

ملاحظات لاستخدام التوابع التجميعية :

- عند كتابة اسم عمود داخل الجملة SELECT لا بد من كتابته ضمن group by وذلك لأن التوابع التجميعية تتعامل مع عدة صفوف.
- يمكن استخدام العبارة ORDER BY مع التوابع التجميعية كما مبين في المثال السابق.
- لا يمكن استخدام التوابع التجميعية ضمن العبارة WHERE ولكن نستخدم بدلاً عنها HAVING.

مثال (7) :

```
SQL > SELECT deptno , AVG(sal)
      2       FROM    emp
      3       ORDER BY AVG(SAL) ;
ERROR at Line 1 :

ORA-00937: Not A Single-Group Group Function <_____ رسالة الخطأ
```

مثال (8) :

```
SQL > SELECT deptno , AVG(SAL)
  2       FROM    emp
  3       WHERE avg(sal)>2000
  4        GROUP BY deptno ;
ERROR at Line 3 :

ORA-00934: group function is not allowed here <_____ رسالة الخطأ
```

ظهرت رسالة الخطأ هنا لاستخدامنا عبارة الشرط where مع التابع التجميعي ونصحح ذلك باستخدامنا عبارة HAVING كما في المثال التالي

مثال (9) :

```
SQL > SELECT deptno , AVG(SAL)
  2         FROM    emp
  3         WHERE  AVG (sal)>2000
  4          HAVING  AVG(SAL) > 2000 ;
```

النتائج

deptno	AVG(sal)
10	2916.66667
20	2175

مثال (10) :

```
SQL > SELECT JOB , SUM(SAL)
  2         FROM    emp
  3         WHERE JOB NOT Like 'SALES%'
  4        GROUP BY JOB
  5        HAVING SUM(SAL) > 5000
  6      ORDER BY SUM(SAL) ;
```

النتائج

JOB	SUM(SAL)
ANALYST	6000
MANAGER	8275

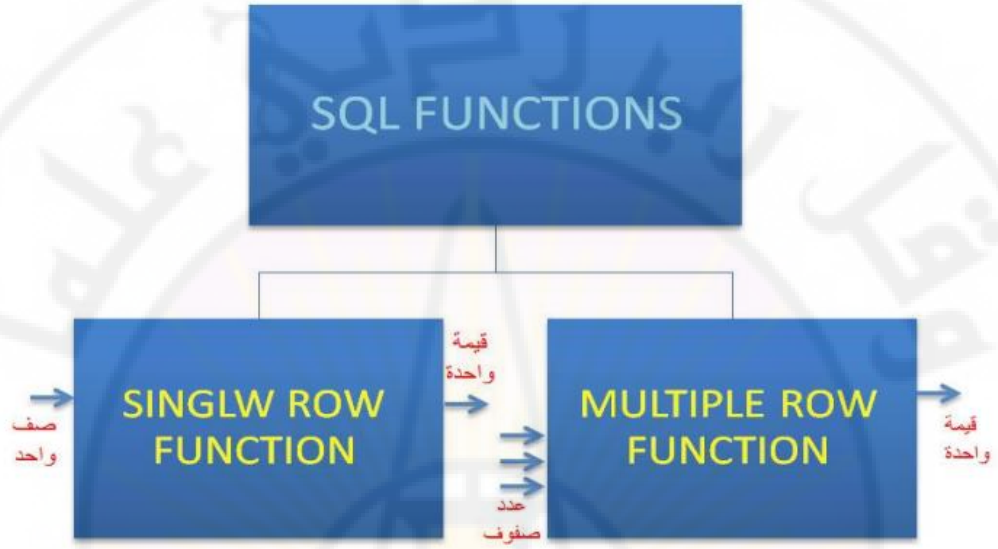
وقل رب زدني علما
جامعة دمشق
Damascus University

المحاضرة : ٥

Data Base1

تستخدم التوابع في لغة SQL وهي أداة قوية مفيدة عند استخدام SELECT وتنقسم هذه التوابع إلى نوعين كما في الشكل التالي :

أنواع الدوال



- **توابع الصف الواحد Single Row Function .**

حيث تتعامل مع البيانات صف وحيد وتكون قيمتها وحيدة وتنقسم لعدة أنواع:

- توابع محرفية.
- توابع رقمية.
- توابع التاريخ.

➢ التوابع المحرفية Character Function:

وتتعامل مع البيانات المحرفية وتكون نتيجتها حروفاً أو أرقاماً والجدول التالي يبين هذه التوابع :

	وظيفتها	الدالة FUNCTION
١	دالة تستخدم لتحويل جميع الحروف (عمود او سلسلة) إلى حروف صفيرة Small	**LOWER** (Column \ Expression)
٢	دالة تستخدم لتحويل حروف (عمود أو سلسلة) إلى حروف كبيرة Capital	**UPPER** (Column\ Expression)
٣	دالة لتحويل الحرف الاول فقط من (عمود أو سلسلة)إلى حرف كبيرCapital وباقي الحروف تحول الى حروف صغيرة	**INITCAP** (Column\ Expression)
٤	دالة ربط عمودين أو سلسلتين معاً وهي تماماً مثل أداة الربط (||)	**CONCAT**(Column1\ Expression1, Column2\ Expression2)
٥	دالة تستخدم لقطع جزء من عمود أو سلسلة بدايةً من الحرف رقك M وعدد الحروف المقطوعة هي n	**SUBSTR** (Column\ Expression,M,N)
٦	دالة تستخدم لإيجاد عدد حروف السلسلة أو العمود (الناتج عدد)	**LENGTH** (Column\ Expression)
٧	دالة تستخدم لتحديد مكان حرف معين داخل سلسلة عمود (الناتج عدد)والحرف M يعبر عن الحرف المراد تحديد مكانه	**INSTR** (Column\ Expression,M)
٨	دالة تستخدم لضبط بيانات سلسلة أو عمود ناحية اليمين وذلك بملء حرف معين من اليسار والحرف n لتحديد الطول بعد الضبط	**LPAD** (Column\ Expression,N,'string')
٩	دالة تستخدم لضبط بيانات عمود أو سلسلة ناحية اليسار وذلك بملء حرف معين من اليمين والحرف n لتحديد الطول بعد الضبط	**RPAD** (Column\ Expression,N,'string')
١٠	دالة تستخدم لقطع حرف معين من بداية او نهاية الكلمة فقط	**TRIM** ('character' FROM Column\ Expression,M)

➢ والجدول التالي يوضح بعض الأمثلة عن التوابع المحرفية:

#	النتيجة	المثال
١	good by	Select LOWER ('GOOD BY') from dual;
٢	GOOD BY	Select UPPER ('GOOD BY') from dual;
٣	Good	Select INITCAP ('GOOD) from dual;
٤	GOODBY	Select CONCAT ('GOOD' , 'BY') from dual;
٥	OOD	Select STBSTR ('GOOD BY',2,3) from dual;
٦	4	Select LENGTH ('GOOD') from dual;
٧	4	Select INSTR ('GOOD' , 'D') from dual;
٨	*****AHMED	Select LPAD ('AHMED',10,'*') from dual;
٩	AHMED*****	Select RPAD ('AHMED',10,'*') from dual;
١٠	AMI	Select TRIM ('S' FROM 'SAMI') from dual;

ملاحظة : الجدول DUAL هو جدول وهمي ضمن أوراكل يستخدم لإجراء العمليات التي لا يدخل فيها أي جدول.

➢ بعض الامثلة عن التوابع المحرفية :

المثال (1) :

```
SQL > SELECT LOWER (ename), UPPER (job), INITCAP(job),
CONCAT (ename,job)
2       FROM emp
3       WHERE SAL= 3000;
```

النتائج

LOWER (ENAME)	UPPER (JOB)	INITCAP(JOB)	CONCAT (ENAME, JOB)
scott	ANALYST	Analyst	SCOTTANALYST
ford	ANALYST	Analyst	FORDANALYST

مثال (2) :

```
SQL > SELECT ename, SUBSTR (ename,2,3), LENGTH(ename),
INSTR (ename,'K')
2       FROM emp
3       WHERE LOWER(job)= 'manager';
```

النتائج

ENAME	SUBSTR (ename,2,3)	LENGTH(ename)	INSTR (ename,'K')
JONES	ONE	5	0
BLAKE	LAK	5	4
CLARK	LAR	5	5

مثال (3):

```
SQL > SELECT ename, TRIM ('S' FROM  ename),
LPAD(ename,10,'*'), RPAD (ename,10,'#')
2       FROM  emp
3       WHERE  SAL > 2500;
```

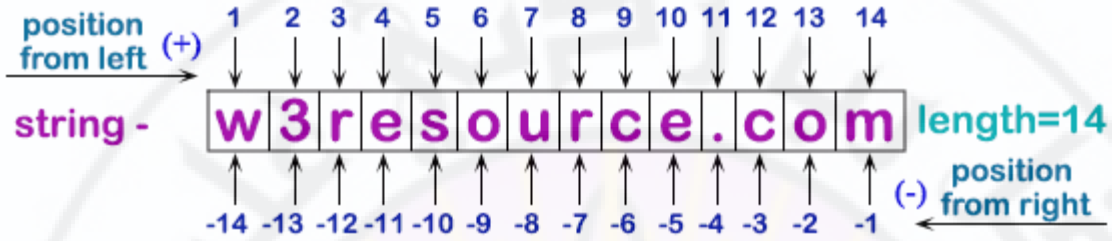
النتائج

ENAME	TRIM ('S' FROM ename)	LPAD(ename,10,'*')	RPAD (ename,10,'#')
JONES	JONES	*****JONES	JONES#####
BLAKE	BLAKE	***** BLAKE	BLAKE#####
SCOTT	SCOTT	***** SCOTT	SCOTT#####
KING	KING	***** KING	KING#####
FORD	FORD	***** FORD	FORD#####

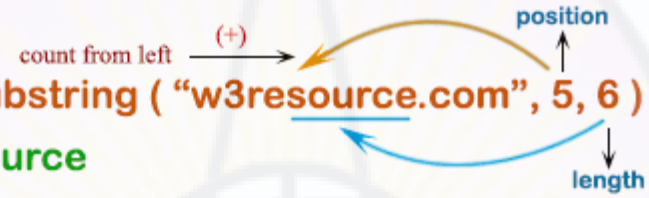
MySQL substirng() function

Extracts specific number of characters from specific position

Syntax : substring (string, position, length)

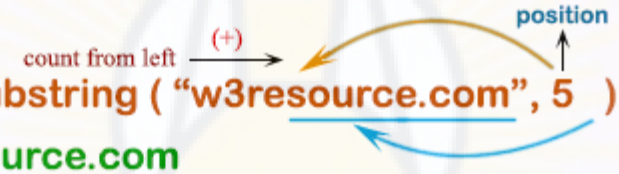


Example :



Result : source

Example :

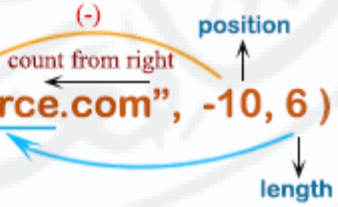


Result : source.com

Example :

Result : e.com

Example :



Result : source

Example :

Result : source

```sql
SELECT
    country.country_name,
    COUNT(city.lat) AS lat_count,
    SUM(city.lat) AS lat_sum,
    AVG(city.lat) AS lat_avg,
    MIN(city.lat) AS lat_min,
    MAX(city.lat) AS lat_max
FROM city
INNER JOIN country ON city.country_id = country.id
GROUP BY country.id, country.country_name;
```

100 %

Results | Messages

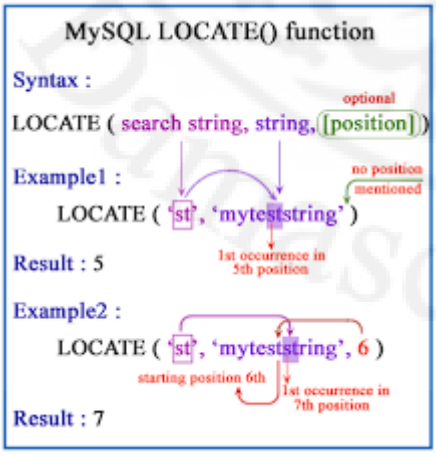
	country_name	lat_count	lat_sum	lat_avg	lat_min	lat_max
1	Deutschland	1	52.520008	52.520008	52.520008	52.520008
2	Hrvatska	1	45.815399	45.815399	45.815399	45.815399
3	Polska	1	52.237049	52.237049	52.237049	52.237049
4	Srbija	1	44.787197	44.787197	44.787197	44.787197
5	United States of America	2	74.782845	37.391422	34.052235	40.730610

- `CONCAT(str1,str2,...)`

  Returns the string that results from concatenating the arguments. May have one or more arguments. If all arguments are nonbinary strings, the result is a nonbinary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent nonbinary string form.

  `CONCAT()` returns `NULL` if any argument is `NULL`.

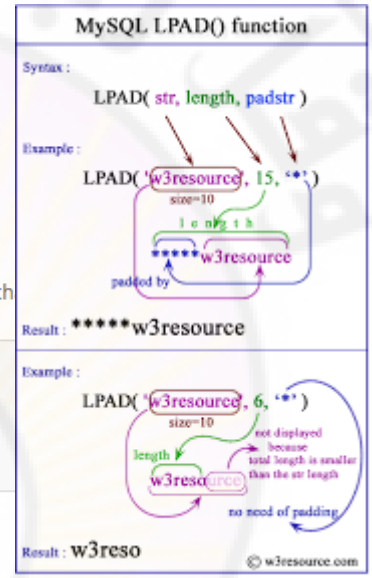
```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
        -> '14.3'
```



- `LOCATE(substr,str)`, `LOCATE(substr,str,pos)`

  The first syntax returns the position of the first occurrence of substring *substr* in string *str*. The second syntax returns the position of the first occurrence of substring *substr* in string *str*, starting at position *pos*. Returns `0` if *substr* is not in *str*. Returns `NULL` if any argument is `NULL`.

```
mysql> SELECT LOCATE('bar', 'foobarbar');
        -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
        -> 0
mysql> SELECT LOCATE('bar', 'foobarbar', 5);
        -> 7
```



- `LPAD(str,len,padstr)`

  Returns the string *str*, left-padded with the string *padstr* to a length of *len* ch

```
mysql> SELECT LPAD('hi',4,'??');
        -> '??hi'
mysql> SELECT LPAD('hi',1,'??');
        -> 'h'
```

- `LTRIM(str)`

```
SQL

SELECT MAX(TaxRate)
FROM Sales.SalesTaxRate;
GO
```

Here is the result set.

```
-------------------
19.60
Warning, null value eliminated from aggregate.

(1 row(s) affected)
```

```
SQL

SELECT MIN(TaxRate)
FROM Sales.SalesTaxRate;
GO
```

Here is the result set.

```
-------------------

5.00

(1 row(s) affected)
```

قواعد البيانات

المحاضرة السادسة

توابع الصف الواحد(الدرجية) أنواعها واستخداماتها :

سبق وعرفنا التوابع الدرجية في SQL على أنها التوابع التي تأخذ مُعاملاً وحيداً وتُعيد قيمة وحيدة.

تصنِّف التوابع الدرجية إلى أربعة أصناف أساسية هي التالية:

- التوابع الرقمية: وهي التوابع الخاصة بالعمليات على الأرقام، مثل تابع التقريب إلى أقرب فاصلة عشرية.
- توابع سلاسل المحارف: وهي التوابع الخاصة بالعمليات على سلاسل المحارف، مثل تابع تحديد طول سلسلة محرفية.
- توابع التاريخ والوقت: وهي التوابع الخاصة بالعمليات على التاريخ والوقت.
- توابع التحويل: وهي التوابع الخاصة بعملية تحويل مُعامل الدخل، من نمط بيانات إلى آخر.

1- التوابع الرقمية :

التوابع الرقمية وهي التوابع الدرجية الخاصة بالعمليات على القيم الرقمية ومن أهمها التوابع التالية:

- التابع Floor: يُقرِّب مُعامل الدخل إلى أقرب عدد صحيح أصغر من مُعامل الدخل. $\lfloor x \rfloor$
- التابع Ceiling: يُقرِّب مُعامل الدخل إلى أقرب عدد صحيح أكبر من مُعامل الدخل. $\lceil x \rceil$
- التابع Round: يُقرِّب مُعامل الدخل ذو الفاصلة العشرية إلى أقرب عدد صحيح أو عدد حقيقي بدقة محددة.
- التابع Abs: يعيد القيمة المطلقة لمُعامل الدخل. $| x |$
- التابع Sqrt: يُعيد قيمة الجذر التربيعي لمُعامل الدخل.
- التوابع Sin, Cos, Tan, Atan: توابع تحسب قيم ظل، تظل، جب، تجب الزاوية التي نأخذها كمُعامل دخل.
- التابع mod: تابع يستخدم لإيجاد باقي القسمة.

(أمثلة):

- **التابع FLOOR:**

نستخدم في هذا المثال تابع التدوير FLOOR لتدوير علامات الطلاب إلى الرقم الصحيح بإزالة الجزء ذو الفاصلة العشرية.

```
SELECT FLOOR(studentMark) FROM marks;
```

فإذا كان لدينا طالب علامته: 66.5، أو 66.2، أو 66.7، ستظهر علامته ضمن نتيجة هذا الاستعلام: 66

- **التابع CEILING:**

إذا كان المطلوب تدوير علامات الطلاب إلى الرقم الصحيح الأعلى، نستخدم التابع CEILING كما في الصيغة التالية:

```
SELECT CEILING(studentMark) FROM marks;
```

فإذا كان لدينا طالب علامته: 66.5، أو 66.2، أو 66.7، ستظهر علامته ضمن نتيجة هذا الاستعلام: 67

(أمثلة):

- **التابع ROUND:**

دالة تستخدم لقص عدد معين من الجزء العشري مع تقريب الاعداد إلى أقرب عدد عشري أو إلى عدد صحيح والحرف N يبين عدد الارقام بعد العلامة العشرية وتوجد حالات للحرف N :

- إذا كان (N=0) فإن التقريب يكون إلى أقرب عدد صحيح.
- إذا كان (N>0) عدد موجب،فإن التقريب يكون في الجزء بعد العلامة العشرية.
- إذا كان (N<0) عدد سالب،فإن التقريب يكون في الجزء قبل العلامة العشرية.

إذا كان المطلوب تدوير عدد ما إلى رقم بعدد خانات عشرية محدد نستطيع استخدام التابع ROUND كما في الصيغة التالية:

```
SELECT ROUND(studentMark, 1) FROM marks;
```

تظهر قيم الخرج كقيم ذات خانة عشرية واحدة. فإذا كانت علامة طالب 66 ستظهر القيمة ضمن نتيجة الاستعلام 66 وإذا كانت 66.55 ستصبح 66.6.

مثال :

(أمثلة):

```
SQL>SELECT ROUND(45.923,0),ROUND(45.923,2),ROUND(45.923,-1),ROUND(45.923,-2)
  2 FROM dual ;
```

الأعداد

ROUND(45.923,0)	ROUND(45.923,2)	ROUND(45.923,-1)	ROUND(45.923,-2)
46	45.92	50	0

تمرين :

- إذا كان لدينا جدول Goods يحتوي لائحة بالبضائع التي نريد نقلها(Good Description)، ووزن كل منها(GoodWeight)، وإذا علمنا أن عربة النقل تستطيع شحن 1.5 طن. حدد عدد رحلات النقل اللازمة لنقل كل مادة من المواد.

GoodWeight	GoodDescription
2.5	Apple boxes
...	...

- الحل :

```
SELECT goodDescription, CEILING(goodWeight / 1.5) AS Trips FROM
    Goods;
```

(أمثلة):

- **التابع ABS:**

إذا أردنا حساب القيمة المطلقة لحقل height يشير إلى ارتفاع نقاط جغرافية عن سطح البحر من الجدول geoTable، وذلك لتحديد الارتفاع الأكبر بينها، يمكننا استخدام الاستعلام:

```
SELECT MAX(ABS(height)) FROM geoTable;
```

يحسب التابع ABS القيمة المطلقة لكل ارتفاع، ويستخرج التابع التجميعيmax الإرتفاع الأعلى.

- **التوابع SIN, COS, TAN, ATAN :**

إذا أردنا حساب قيمة، جيب أو تجب أو ظل مجموعة من الزوايا الواردة في الحقل Angle من الجدول Angles يكفي كتابة الاستعلام:

```
SELECT SIN(angle), COS(angle), TAN(angle) FROM Angles;
```

(أمثلة):

- **التابع SQRT:**

يحسب التابع SQRT الجذر التربيعي لقيمة معينة، إذ تعطي نتيجة الاستعلام التالي الرقم 3:

```
SELECT SQRT(9);
```

- **التابع mod(m,n):**

تابع يستخدم لإيجاد باقي قسمة m على nكمايلي:

```
SELECT MOD(1600,300);
```

2- توابع سلاسل المحارف :

توابع سلاسل المحارف هي التوابع الدرجية الخاصة بالعمليات على السلاسل المحرفية، ومن أهمها التوابع التالية:

التابع	استخدامه
LEFT()	يُعيد جزء من السلسلة يبتدئ من بدايتها حتى عدد محدد من المحارف
RIGHT()	يُعيد جزء من السلسلة يبتدئ من نهايتها حتى عدد محدد من المحارف
SUBSTR()	يُعيد جزء من السلسلة يبتدئ من موقع محدد فيها وبطول عدد محدد من المحارف
LENGTH()	يُعيد طول السلسلة المحرفية
CONCAT()	يُستخدم لدمج أكثر من سلسلة محرفية
LOWER() / UPPER()	يحوِّل جميع محارف السلسلة إلى أحرف كبيرة أو صغيرة
TRIM()	يلغي الفراغات من بداية ونهاية السلسلة المحرفية
INSTR()	يُستخدم لتحديد موقع سلسلة جزئية ضمن سلسلة رئيسية

(أمثلة):

- **التابعان LEFT و RIGHT:**

يُعيد التابعان LEFT و RIGHT جزء من سلسلة المحارف مؤلف من عدد محدد من المحارف ابتداءً من يسار السلسلة في حالة التابع LEFT، ومن يمين السلسلة في حالة التابع RIGHT. فإذا أردنا كتابة الاستعلام الذي يعيد أول 5 محارف من قيم الحقل Title في الجدول News يمكننا استخدام الصيغة التالية:

```
SELECT LEFT(title, 5) FROM News;
```

- **التابع SUBSTR:**

يُعيد التابع SUBSTR جزء من سلسلة محارف، ابتداءً من موقع محدد في تلك السلسلة، وبطول عدد محدد من المحارف. فإذا أردنا كتابة الاستعلام الذي يعيد سلسلة محارف بطول 5 محارف اعتباراً من قيم الحقل Title مبتدئاً من المحرف رقم 10، نكتب الصيغة التالية:

```
SELECT SUBSTR (title, 10, 5) FROM News;
```

(أمثلة):

- **التابع LENGTH:**

يحسب التابع LENGTH طول سلسلة محارف معينة. فإذا أردنا كتابة الاستعلام الذي يعيد أطوال سلاسل المحارف الخاصة بالحقل Title نستخدم الصيغة:

```
SELECT LENGTH(title) FROM News;
```

- **التابع CONCAT:**

يدمج التابع CONCAT أكثر من سلسلة محرفية. فلإعادة قائمة بسلاسل محرفية تمثل حاصل دمج السلاسل المحرفية الخاصة بالحقل Title، مع السلاسل المحرفية الخاصة بالحقل Details في الجدول News، نكتب الصيغة:

```
SELECT CONCAT(title, details) FROM News;
```

(أمثلة):

- **التابعان Upper و Lower:**

يحوِّل التابعان Upper و Lower جميع محارف السلسلة إلى أحرف كبيرة أو صغيرة (فقط للمحارف A-Z). فإذا أردنا تحويل السلاسل المحرفية في الحقل Title إلى سلاسل مؤلفة من أحرف كبيرة، نستخدم الاستعلام:

```
SELECT UPPER(title) FROM News;
```

- **التابع Trim:**

يزيل التابع Trim الفراغات من بداية ونهاية السلسلة المحرفية. فإذا أردنا الحصول على السلاسل المحرفية في الحقل Titleبعد التخلص من الفراغات، في بداية ونهاية سلاسل المحارف تلك، نستخدم الصيغة:

```
SELECT TRIM(title) FROM News;
```

- **التابع Instr:**

يحدد التابع Instr موقع أول ظهور لسلسلة جزئية في سلسلة رئيسية. يعيد التابع القيمة(0) في حال عدم إيجاد السلسلة. فإذا كان المطلوب مثلاً كتابة استعلام يعيد موقع أول ظهور للسلسلة الجزئية 'Test' ضمن سلاسل الحقل Title من جدول News، نستخدم الصيغة:

```
SELECT INSTR(title, 'Test') FROM News;
```

تمرين :

ليكن لدينا الجدول userInfo الذي يحتوي أسماء وعناوين البريد الالكتروني (userEmail)، لمجموعة من الأشخاص، المطلوب إعادة قائمة بأسماء نطاقات عناوين البريد الالكتروني.

الحل:

```
Select substr (userEmail,instr(useEmail,'@')+1,10) from userInfo
```

3- توابع التاريخ والوقت :

توابع التاريخ والتوقيت هي التوابع الخاصة بالعمليات على التاريخ والوقت ومن أهمها التوابع التالية:

التابع	استخدامه
CURRENT_DATE()	يعيد التاريخ الحالي الخاص بنظام إدارة قاعدة البيانات
CURRENT_TIME()	يُعيد التوقيت الخاص بنظام إدارة قاعدة البيانات
CURRENT_TIMESTAMP()	يُعيد التاريخ والتوقيت الخاصين بنظام إدارة قاعدة البيانات

(أمثلة):

- **التابع CURRENT_DATE :**

يُعيد هذا التابع قيمة التاريخ الخاص بنظام إدارة قاعدة البيانات فإذا أردنا إظهار التاريخ الحالي نستخدم الصيغة:

```
SELECT CURRENT_DATE AS myDate;
```

- **التابع CURRENT_TIME :**

يُعيد هذا التابع قيمة التوقيت الحالي الخاص بنظام إدارة قاعدة البيانات فإذا أردنا إظهار التوقيت الحالي نستخدم الصيغة:

```
SELECT CURRENT_TIME AS myTime;
```

- **التابع CURRENT_TIMESTAMP:**

يُعيد هذا التابع قيمة التاريخ والتوقيت الخاص بنظام إدارة قاعدة البيانات.

```
SELECT CURRENT_TIMESTAMP;
```

محاضرة ٧

Data Base1

➢ التوابع الرقمية NUMBER Function :

وهي توابع تتعامل مع البيانات رقمية والنتيجة أرقاماً والجدول التالي يبين هذه التوابع وعملها :

	وظيفتها الدالة	Function
١	دالة تستخدم لقص عدد معين من الجزء العشري مع تقريب الاعداد إلى أقرب عدد العشري أو إلى عدد صحيح والحرف n يبين عدد الأرقام بعد العلامة العشرية، وتوجد حالات للعرف n. • إذا كان (n=0) فإن التقريب يكون إلى أقرب عدد صحيح. • إذا كان (n>0) أي عدد موجب فإن التقريب يكون في الجزء بعد العلامة العشرية (الجزء العشري). • إذا كان (n<0) أي عدد سالب فإن التقريب يكون في الجزء قبل العلامة العشرية (الجزء الصحيح)	**ROUNF** (Column\ Expression,N)
٢	دالة تستخدم لقص عدد معين من الجزء العشري بدون تقريب، وأيضاً توجد حالات للحرف N • إذا كان (n=0) فإنه يتم قص الجزء العشري كله ويكون الناتج عدد صحيح . • إذا كان (n>0) أي عدد موجب فغن القص يكون في الجزء بعد العلامة العشرية (الجزء العشري). • إذا كان (n<0) أي عدد سالب فإن القص يكون في الجزء قبل العلامة العشرية (الجزء الصحيح)	**TRUNC** (Column\ Expression,N)
٣	دالة تستخدم لإيجاد باقي قسمة العدد M على العدد N	**MOD**(m,n)

مثال (4) :

```
SQL > SELECT ROUND(45.923,0), ROUND(45.923,2),
ROUND(45.923,-1), ROUND(45.923,-2)
2       FROM dual ;
```

الناتج			
ROUND(45.923,0)	ROUND(45.923,2)	ROUND(45.923,-1)	ROUND(45.923,-2)
46	45.92	50	0

مثال (5) :

```
SQL > SELECT TRUNC(45.923,0), TRUNC (45.923,2), TRUNC (45.923,–1),
TRUNC (45.923,–2)
2       FROM dual ;
```

النتائج			
TRUNC (45.923,0)	TRUNC (45.923,2)	TRUNC (45.923,-1)	TRUNC (45.923,-2)
45	45.92	40	0

مثال (6) :

```
SQL > SELECT ename , sal , comm , MOD(SAL,COMM)
2       FROM  emp
3       WHERE sal= 1600;
```

النتائج			
ENAME	SAL	COMM	MOD(SAL,COMM)
ALLEN	1600	300	100

➢ توابع التاريخ Date Function :

وهي توابع تتعامل مع التاريخ واهمية التاريخ في حياتنا قامت أوراكل بتوفير توابع بسيطة للتعامل مع التاريخ والجدول التالي يبين هذه التوابع وطريقة عملها:

	وظيفتها	الدالة Function
١	دالة تستخدم لعرض تاريخ النظام الموجود بجهاز الحاسب الآلي (تاريخ اليوم الحالي)	SYSDATE
٢	دالة لإيجاد عدد الأشهر بين تاريخين	MONTHS_BETWEEN(date1,date2)
٣	دالة لإضافة عدد معين من الأشهر على تاريخ معطى	ADD_MONTHS(date,n)
٤	دالة لإيجاد تاريخ يوم معين بعد تاريخ معطى	NEXT_DAY(date,'day')
٥	دالة لإيجاد آخر يوم في الشهر لتاريخ معطى	LAST_DAY(date)
٦	دالة تستخدم لتقريب التاريخ لأقرب شهر أو سنة	ROUND(date)
٧	دالة تستخدم لقص التاريخ لأقرب شهر او سنة	TRUNC(date)

- والجدول التالي يوضح امثلة عن هذه التوابع :

	المعنى	النتيجة	المثال
١	تم إيجاد الأشهر بين التاريخين المعطيين	19.6774194	MONTHS_BETWEEN('01-SEP-95', '11-JAN-94')
٢	تم إضافة ٦ أشهر على التاريخ المعطى	'11-JUL-94'	ADD_MONTHS('11-JAN-94',6)
٣	إيجاد تاريخ يوم الجمعة بعد التاريخ '01-SEP-95'	'08-SEP-95'	NEXT_DAY('01-SEP-95' , 'FRIDAY')
٤	إيجاد آخر يوم في شهر ستمبر September	'30-SEP-95'	LAST_DAY('01-SEP-95')
٥	تم تقريب التاريخ المعطى إلى أقرب شهر وبما ان اليوم خو (25) أي أكبر من (15) فتم تقريب إلى أول الشهر الذي يليه	01-AUG-95	ROUND('25-JUL-95' , 'MONTH')
٦	تم تقريب التاريخ المعطى إلى أقرب سنة وبما ان شهر	'01-JUL-95'	ROUND('25-JUL-95' , YEAR)

		(JULY) هو شهر (7) فإنه تم التقريب إلى أول السنة التالية	
٧	تم قص التاريخ المعطى بدون تقريب إلى أول يوم في الشهر	'01-JUL-95'	TEUNC('25-JUL-95' ,' MONTH)
٨	تم قص التاريخ المعطى بدون تقريب إلى أول السنة الحالية	'01-JUL-95'	TEUNC('25-JUL-95' ,' YEAR')

➢ بعض الأمثلة عن توابع التاريخ :

مثال (7) :

```
SQL > SELECT SYSDATE FROM dual ;
```

الناتج

SYSDATE
25-01-2017

مثال (8) :

```
SQL > SELECT empno, hiredate,
MONTHS_BETWEEN(sysdate,hiredate)
2       FROM emp
3       WHERE hiredate like '%1987' ;
```

الناتج

EMPNO	HIREDATE	MONTHS_BETWEEN(sysdate,hiredate)
7788	19-04-1987	201.200404
7876	23-05-1987	200.071371

مثال (9) :

```
SQL > SELECT empno , hiredate
,ADD_MONTHS(HIREDATE,6),LAST_DAY(HIREDATE)
2       FROM emp
3       WHERE hiredate like '%1987' ;
```

النتائج

EMPNO	HIREDATE	ADD_MONTHS(HIREDATE,6)	LAST_DAY(HIREDATE)
7788	19-04-1987	19-10-1987	30-04-1987
7876	23-05-1987	23-11-1987	31-05-1987

مثال (10) :

```
SQL > SELECT empno,hiredate,NEXT_DAY(hiredate,'FRIDAY')
2       FROM    emp
3       WHERE hiredate like '%1987';
```

النتائج

EMPNO	HIREDATE	NEXT_DAY(hiredate,'FRIDAY')
7788	19-04-1987	24-04-1987
7876	23-05-1987	09-05-1987

➢ توابع التحويل Conversion Functions :

يوجد أنواع كثيرة من البيانات يمكن تخزينها في الجدول ومنها الرقمية number والمحرفية character والتاريخ date وتوابع التحويل تقوم بتحويل البيانات من نوع لآخر وهي عبارة عن ثلاثة أنواع، الجدول التالي يوضحها :

	وظيفتها	الدالة Functions
١	تستخدم هذه الدالة لتحويل البيانات الرقمية او بيانات التاريخ إلى بيانات حرفية بشكل معين (FORMAT) حسب الطلب fmt .	TO_CHAR(DATE / NUMBER , 'fmt')
٢	تستخدم لتحويل البيانات الحرفية الى بيانات من نوع التاريخ بشكل معين (FORMAT) حسب الطلب fmt	TO_DATE (CHAR , 'fmt')
٣	تستخدم لتحويل البيانات الحرفية الى بيانات رقيمة بشكل معين (FORMAT) حسب الطلب fmt	TO_NUMBER (CHAR , 'fmt')

- **دالة التحويل TO_CHAR :**

أولاً – استخدامها لتحويل بيانات التاريخ إلى بيانات محرفية:

الشكل العام

TO_CHAR(DATE , 'fmt')

مثال (11) :

```
SQL > SELECT sysdate , TO_CHAR(sysdate, 'DD/MM/YYYY')
  2      FROM   dual ;
```

النتائج

SYSDATE	TO_CHAR(SYSDATE,'DD/MM/YYYY')
26-01-2004	26/01-2004

- الجدول التالي يوضح بعض التنسيقات التي يمكن استخدامها عند تحويل التاريخ الى بيانات محرفية :

١	إظهار السنة كاملة بالأرقام (القرن + السنة) مثل 2004	YYYY
٢	إظهار رقمين فقط من السنة 04	YY
٣	إظهار السنة كاملة كتابة (TWO THOUSAND FOUR) وحالة احرف الكتابة تتوقف على حالة أحرف كلمة YEAR	YEAR
٤	إظهار الشهر في شكل رقمين 01	MM
٥	إظهار الشهر كتابة (JANURY)	MONTH
٦	إظهار الثلاث حروف الاولى من اليوم (JAN)	DY
٧	إظهار اليوم كاملاً كتابتاً (FRIDAY)	DAY
٨	إظهار الوقت بنظام 12 ساعة وهل هو صباحاً أم مساءً (04:30:50 PM)	HH12:MI:SS AM

مثال (12) :

```
SQL > SELECT empno, TO_CHAR(hiredate, 'DAY "OF" MONTH
YYYY HH12:MI:SS AM')
2       FROM   emp
3       WHERE ename=upper('king') ;
```

النتائج

EMPNO	TO_CHAR(hiredate, 'DAY "OF" MONTH YYYY HH12:MI:SS AM')
7839	TUESDAY OF NOVEMBER 1981 12:00:00 AM

ثانياً : استخدام التابع TO_CHAR في تحويل البيانات الرقمية إلى بيانات محرفية :

الشكل العام

TO_CHAR(NUMBER, 'fmt')

مثال (13) :

```
SQL > SELECT empno, TO_CHAR(sal, '$99.999') salary
  2      FROM  emp
  3      WHERE sal > 2500;
```

الناتج

EMPNO	SALARY
7566	$2,975
7698	$2,850
7788	$3,000
7839	$5,000
7902	$3,000

- الجدول التالي يبين بعض الرموز التي تستخدم في تحويل البيانات الرقمية إلى محرفية لتظهر بشكل معين:

١	عدد تكرار هذا الرقم يمثل عدد الخانات التي تظهر، مثلاً عندما نكتب (99) معناها ظهور رقمين وهكذا	9
٢	يعني ظهور الرقم وقبله صفر	09
٣	يعني ظهور صفراً إذا كانت القيمة معدومة	990
٤	إظهار علامة $ قبل الرقم	$99
٥	إظهار العلامة العشرية	.
٦	إظهار فواصل بين كل ثلاثة أرقام (فاصلة الالوف)	,
٧	إظهار علامة السالب (-) يمين الرقم إذا كان سالباً	MI

- تابع التحويل TO_DATE :

الشكل العام

TO_DATE (CHAR, 'fmt')

تقوم بتحويل البيانات المحرفية التي تعبر عن التاريخ إلى بيانات من نوع DATE بشكل معين :

مثال (14) :

```
SQL > SELECT TO_DATE('FEBRUARY 22, 1981', 'MONTH
DD, YYYY')
2       FROM   emp
```

الناتج
TO_DATE('FEBRUARY 22, 1981', 'MONTH DD, YYYY')
22-FEB-1981

- تابع التحويل TO_NUMBER:

الشكل العام

TO_NUMBER(CHAR , 'fmt')

يستخدم هذا التحويل البيانات المحرفية التي تعبر عن أرقام إلى بيانات رقمية بتنسيق معين وذلك لإجراء العلميات الحسابية.

وقل رب زدني علما
جامعة دمشق
Damascus University

مخبر
قواعد البيانات
المحاضرة السادسة

إنشاء جدول Table

- أسهل الطرق لبناء جدول هو إنشاء الجدول باستخدام تعبير (CREATE TABLE) مع جميع علاقاته من مفاتيح أولية Primary keys، ومفاتيح ثانوية Foreign keys، مع القيود على البيانات دفعة واحدة. أما الطريقة الثانية فهي إنشاء جدول والتعامل معه. ثم وفي مراحل لاحقة يمكن التعديل على بنية الجدول باستخدام التعليمة (ALTER TABLE).

الشروط الواجب توافرها عند اختيار اسم الجداول أو الأعمدة:

- يجب أن يبدأ الاسم بحرف.
- يجب أن لا يزيد طول الاسم عن 30 حرف.
- من الممكن أن يتكون من حروف كبيرة وصغيرة وأرقام ورموز خاصة مثل(...$,#).
- يجب أن لا يتكرر اسم الجدول أكثر من مرة داخل قاعدة البيانات الواحدة.
- يجب أن لايتكرر اسم عمود أكثر من مرة داخل الجدول الواحد.
- يجب أن لايكون من الاسماء المحجوزة لأوراكل مثل (from, select…).
- يفضل أن يكون اسم الجدول له معنى بحيث يعبر عن نوع بيانات الجدول.

محددات بناء الجداول :

عند إنشاء جدول يجب أن تضع مخططاً للنقاط التالية:

- أنماط البيانات التي سوف يحتويها الجدول.
- الأعمدة في الجداول، أنماط معطيات الأعمدة، الطول الأعظمي لأنماط الأعمدة.
- ما هي الأعمدة التي تقبل القيمة null.
- القيم الإفتراضية، القيود.
- المفاتيح الأولية.
- المفاتيح الثانوية.

إنشاء القيود على الحقول Constraints:

يمكنك فرض قيود معينة على حقول الجداول التي تنشئها. سنتعرف فيما يلي على مجموعة من القيود على الحقول. من أهم القيود و أكثرها استخداماً:

- NOT NULL
- DEFAULT
- PRIMARY KEY
- UNIQUE
- AUTO_INCREMENT

مفاتيح الجداول:

المفتاح الرئيسي PRIMARY KEY

معنى هذا القيد هو إنشاء مفتاح أساسي Primary key داخل الجدول وذلك لتمييز عمود معين بحيث أن هذا العمود يكون له خاصيتان هما:

- عدم قبول تكرار القيم داخله.

- عدم السماح بترك قيمته فارغة (NULL).

المفتاح الثانوي (الغريب) FOREIGN KEY

يستخدم لعمل مفتاح ربط بين جدولين ويطبق على مستوى العمود أو الجدول.

```
CREATE TABLE Employee(
emp_id int primary key ,
f_name varchar(20) NOT NULL ,
l_name varchar(30) NOT NULL,
job varchar(30) NOT NULL ,
Dep_id int NOT NULL ,
E-mail varchar(30),
phone int Default 0);
```

تعليمة INSERT:

تُستخدم تعليمة INSERT لإدراج سجل في جدول محدد، حيث تأخذ تعليمة INSERT الصيغة:

```
INSERT INTO table_name  VALUES (value_1, value_2,
value_3...value_n);
```

حيث تُمثل value_1,..,value_n القيم التي سوف يتم تخزينها في السجل تبعاً لترتيب حقوله. كما يمكننا استخدام صيغة بديلة تُمكننا من تحديد حقول السجل التي نود إدراج البيانات فيها فقط:

```
INSERT INTO table_name (field_1, field_2, ..., field_n)
VALUES (value_1,value_2,..,value_n);
```

➢ إضافة قيم خاصة داخل الاعمدة :

إذا أردنا تاريخ اليوم SYSDATE إلى العمود HIREDATE في جدول الموظفين يمكن ذلك كالآتي :

```
SQL > INSERT INTO emp
2   (empno , ename , job , mgr , hiredate , sal , comm , deptno)
3   VALUES
4   (7196 , 'AHMAD' ,' SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10) ;
```

مثال (3) إضافة بيانات موظف جديد إلى جدول الموظفين :

في المثال السابق من الممكن عدم ذكر أسماء الأعمدة في الجملة INSERT وفي هذه الحالة لا بد من إدخال كل القيم حسب ترتيب الأعمدة في الجدول كالآتي :

```
SQL > INSERT INTO  emp
      VALUES
      (7196 , 'AHMED' , 'SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10)
```

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.

تأخذ تعليمة UPDATE الصيغة:

```
UPDATE table_name
SET Field1  =new_field_value1, Field2 = new_field_value2;
```

يمكن استخدام الكلمة المفتاحية WHERE مع تعليمة UPDATE لتصبح الصيغة:

```
UPDATE table_name
SET Field1= new_field_value1, Field2 =new_field_value2
WHERE condition;
```

➢ التعديل في سجل أو عدد من السجلات في جدول معين (UPDATE) :

الصيغة العامة :

```
SQL > UPDATE جدول
      SET عمود ١ = قيمة ١ ، عمود ٢ = قيمة ٢.....
      WHRER N شرط ;
```

➢ القواعد الواجب التقيد بها عند التعديل :

- يجب أن تكون البيانات الجديدة من نفس بيانات الأعمدة المراد التعديل بها.
- عند تعديل قيم النصوص والتاريخ يجب وضع القيم الجديدة بين علامتي التنصيص الفردية.
- يجب الانتباه عند كتابة الجزء WHERE في جملة التعديل لتحديد أي الصفوف التي سيتم التعيد بها .

مثال (7) تعديل القسم رقم (30) ليصبح EDUCATION بدلاً من SALES :

```
SQL > UPDATE  dept
  2       SET  dname = 'EDUCATION'
  3       WHERE deptno = 30 ;
```

القيمة الجديدة لاسم الإدارة رقم(30)

```
1 Row  updated
```

رسالة تدل على أن التعديل تم في الإدارة رقم (30) فقط

يجب الانتباه لمضمون الشرط where في جملة update فمثلاً في المثال التالي سيتم تعديل كافة الأقسام لتصبح EDUCATION :

مثال (8) :

```
SQL > UPDATE   dept
  2       SET  dname = 'EDUCATION' ;
```

```
2  Row updated
```

رسالة تدل على أن التعديل تم في جميع الصفوف

التعديل في أكثر من عمود :

إذا أردنا تعديل القسم والوظيفة للموظف BLAKE ليصبح مثل الموظف WARD لنرى المثال التالي :

مثال (9) :

```
SQL > UPDATE   emp
  2     SET (job , deptno) = (select job , deptno  from emp where ename =
'WARD')
```

نتيجة الاستعلام الفرعي (30 MANAGER)

```
1   Row updated
```

تعليمة DELETE

تقوم تعليمة DELETE بحذف سجل أو مجموعة من السجلات من جدول ما، تأخذ تعليمة DELETE الصيغة :

```
DELETE FROM [table_name]   WHERE condition;
```

➢ حذف سجل أو عدد من السجلات داخل جدول DELETE FROM :

الصيغة العامة :

```
SQL > DELETE FROM    جدول
      WHERE          شرط
```

➢ القواعد التي يجب التقيد بها عند الحذف :

يجب الانتباه لمضمون الشرط WHERE لتحديد الصفوف المراد حذفها وعند عدم كتابة الشرط سيتم حذف كافة الصفوف في الجدول.

مثال (10) : حذف القسم رقم (40) من جدول الأقسام :

```
SQL > DELETE  FROM dept
WHERE  deptno = 40 ;
```

1 Row deleted.

مثال (11) حذف جميع الموظفين من جدول الموظفين

```
SQL > DELETE FROM  emp
```

14 Row deleted

مثال (12) حذف موظفي القسم SALES :

```
SQL > DELETE FROM  emp
2     WHERE deptno =(select deptno  from  dept  where dname=
'SALES')
```

6 Row deleted

التعديل في الجداول باستخدام ALTER TABLE:

توفر لنا لغة الاستعلام أوراكل (SQL) إمكانية مهمة جداً وهي إمكانية التعديل في هيكل (البناء الداخلي) لجدول قد تم إنشاؤه مسبقاً باستخدام الأمر ALTER TABLE ، وعملية التعديل في الجداول تشتمل على :

أوجه التعديل في الجدول باستخدام الأمر ALTER TABLE	
ADD	تستخدم لإضافة أعمدة جديدة إلى الجدول
MODIFY	تستخدم للتعديل في نوع البيانات للجدول
DROP	تستخدم لإلغاء عمود معين من الجدول

1-إضافة أعمدة جديدة على الجدول.

2- التعديل في نوع بيانات عمود معين.

3- إلغاء عمود معين .

4-تغيير اسم الجدول rename:

Alter table table_name rename new table_name;

التعديل في الجداول باستخدام ALTER :

توفر لغة SQL إمكانية لتعديل البنية الداخلية للجداول باستخدام الامر ALTER TABLE وتشمل ثلاثة تعديلات إما إضافة اعمدة أو التعديل في نوع البيانات أو حذف اعمدة كما موضح في الجدول التالي :

أوجه التعديل في الجدول باستخدام الأمر ALTER TABLE	
ADD	تستخدم لإضافة اعمدة جديدة إلى الجدول
MODIFY	تستخدم للتعديل في نوع البيانات للجدول
DROP	تستخدم لإلغاء عمود معين من الجدول

مثال (4) إضافة عمود اسمه REGION إلى جدول القسم DEPT20 :

```
SQL > ALTER   TABLE   DEPT2
  2         ADD     (region   VARCHAR2(20))
```

Table altered .

مثال (5) التعديل في طول بيانات العمود DNAME ليصبح بطول (20) بدلاً من (14) :

```
SQL > ALTER   TABLE   DEPT2
  2       MODIFY     (dname   VARCHAR2(20))
```

Table altered .

بعض الاعتبارات عند التعديل في أعمدة الجداول :

- يمكن زيادة حجم البيانات للأعمدة.
- يمكن تغيير البيانات من نوع لآخر دون التأثير على البيانات الموجودة.
- لا يمكن إنقاص حجم الأعمدة إذا كانت تحتوي على بيانات.

مثال (6) إلغاء العمود REGION من جدول القسم DEPT2 :

```
SQL > ALTER   TABLE    dept2
  2      drop  COLUMN   REGION
```

Table altered.

ملاحظات

- يمكن زيادة حجم (طول) البيانات للأعمدة.
- يمكن تغيير نوع البيانات من نوع لآخر بحيث لا يؤثر ذلك في بيانات الأعمدة إذا كانت موجودة.
- لا يمكن تقليل حجم (طول) الأعمدة إذا كانت تحتوي على بيانات.
- لا يمكن إلغاء أكثر من عمود واحد فقط في الأمر الواحد.
- يجب أن يتبقى عمود واحد على الأقل بعد عملية الإلغاء داخل الجدول.
- لا يمكن استعادة العمود بعد إلغاؤه.

إلغاء جدول باستخدام الأمر DROP:

هي عبارة عن إلغاء الجدول تماماً من قاعدة البيانات وحذف كل بياناته وكل القيود المتعلقة به.

مثال:

قم بإلغاء جدول الموظفين المسمى EMP:

```
DROP TABLE EMP;
```

المحاضرة : ٩

Data Base 1

أنواع القيود Constraints الجدول التالي يبينها :

	القيد (Constraint)	معنى القيد (Description)
١	NOT NULL	• يمنع هذا القيد ترك عمود معين فارغ (لابد أن يدخل قيمة للعمود) • يطبق على مستوى العمود فقط
٢	UNIQUE	• يمنع هذا القيد تكرار القيم داخل العمود (القيم داخل العمود وحيدة) • يطبق على مستوى العمود أو الجدول .
٣	PRIMARY KEY	• يستخدم لعمل مفتاح أساسي داخل الجدول، والمفتاح الأساسي يتميز بعدم تكرار القيم، وعدم ترك القيم فارغة أي إنه عبارة عن القيدين السابقين. • يطبق على المستوى العمود أو الجدول
٤	FOREIGN KEY	• يستخدم لعمل مفتاح ربط بين جدولين. • يطبق على المستوى العمود أو الجدول
٥	CHECK	• يستخدم لاختبار قيمة عمود بحيث لا يقبل هذا العمود إلا قيم حسب شرط معين. • يطبق على المستوى العمود أو الجدول.

إنشاء القيود Create Constraint :

بطريقتين

- عمل القيود أثناء إنشاء الجداول.
- عمل القيود بعد إنشاء الجداول .

- القيد PRIMARY KEY :

ويتم إنشاؤه على مستوى الجدول أو العمود ومعناه المفتاح الأساسي داخل الجدول وذلك لتمييز عمود معين بحيث يكون له خاصيتان :

١- عدم قبول التكرار للقيم داخله.

٢- عدم السماح لقيمة NULL داخله.

مثال (1) إنشاء قيود على جدول الأقسام وتطبيقها على مستوى الأعمدة :

```
SQL > CREATE TABLE dept (
  2                deptno  NUMBER(2),   PRIMARY KEY ,
  3                dname   VARCHAR2(14), NOT NULL ,
  4                loc     VARCHAR2(13)
  5                );
```

القيود على مستوى الأعمدة

مثال (2) إنشاء القيود أثناء إنشاء جدول الأقسام وتطبيقها على مستوى الجدول :

```
SQL > CREATE TABLE dept (
  2                deptno  NUMBER(2) ,
  3                dname   VARCHAR2(14)  NOT NULL ,
  4                loc     VARCHAR2(13) ,
  5      CONSTRAINT dept_deptno_pk  PRIMARY KEY (deptno) ;
```

القيد على مستوى الجدول

والفرق بين هذه الطريقة والمثال السابق هو أننا نستطيع حذف القيد بسهولة إذا كان على مستوى الجداول كما نلاحظ في هذا المثال اسم القيد dept_deptno_pk.

- القيد UNIQUE KEY :

يتم إنشاؤه على مستوى العمود أو الجدول ومعناه عدم السماح بكرار القيم داخله.

مثال (3) إنشاؤه القيد unique أثناء إنشاء جدول الأقسام وتطبيقه على مستوى الجدول :

```
SQL > CREATE  TABLE dept (
2                    deptno   NUMBER(2) ,
3                    dname   VARCHAR2(14) ,
4                    loc       VARCHAR2(13) ,
5     CONSTRAINT  dept_deptno_uk  UNIQUE(dname) ;
```

القيد على مستوى الجدول

Table cerated

ويمكن تطبيق هذا القيد على المستوى العمود.

- القيد FOREIGN KEY :

يتم إنشاؤه على المستوى الجدول أو العمود ونستخدم هذا القيد عندما نريد ربط جدولين مع بعض فمثلاً لربط جدول الموظفين بجدول الأقسام لمعرفة موظفي قسم معين لا بد من وجود عمود PRIMARY KEY في الجدول الأقسام ونفس هذا العمود موجود في جدول الموظفين كعمود Foreign Key كما في الشكل التالي :

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

Foreign Key Primary Key

ولتنفيذ هذا الربط لا بد من تطبيق قيد Foreign key على العمود deptno في الجدول الموظفين كما في المثاليين التاليين :

مثال (4) إنشاء القيد foreign key على العمود deptno في الجدول الموظفين وتطبيقه على المستوى العمود :

```
SQL > CREATE TABLE emp (
2     empno      NUMBER(4) ,
3     ename      VARCHAR2(10)  NOT NULL ,
4     JOB        VARCHAR2(9) ,
5     mgr        NUMBER(4)   ,
6     hiredate   DATE  ,
7     SAL        NUMBER(7,2) ,
8     Comm       NUMBER(7,2) ,
9     deptno     NUMBER(2)  REFERENCES  dept (deptno) ) ;
```

القيد على مستوى العمود

TABLE Creatrd

في هذا المثال تم إنشاء جدول الموظفين وإنشاء قيد FOREIGN KEY على العمود deptno فالكلمة REFERENCES تشير إلى تطبيق القيد FOREIGN KEY على العمود DEPTNO الذي يشير إلى العمود DEPTNO في الجدول DEPT وهكذا تم تطبيق القيد على المستوى العمود.

- أما تطبيق القيد على المستوى الجدول سيتضح في المثال التالي :

مثال (5) إنشاء القيد FOREIGN KEY على العمود DEPTNO في الجدول الموظفين وتطبيقه على المستوى الجدول :

```
SQL > CREATE TABLE emp (
2     empno      NUMBER(4) ,
3     ename      VARCHAR2(10)  NOT NULL ,
4      JOB       VARCHAR2(9) ,
5      mgr       NUMBER(4)   ,
6     hiredate   DATE  ,
7      SAL       NUMBER(7,2) ,
8     Comm       NUMBER(7,2) ,
9     deptno     NUMBER (2) ,
10  CONSTRAINT  EMP deptno  fk FOREIGN KEY (deptno)
11          REFERENCES dept (deptno)  )  ;
```

القيد على مستوى الجدول

TABLE Creatrd

- القيد CHECK :

يتم إنشاؤه على مستوى العمود او الجدول ويُستخدم عندما نريد تحديد مجال قيم معينة لعمود في الجدول فمثلاً إذا اردنا تحديد القيم المدخلة في العمود DEPTNO في جدول الأقسام لتكون محصورة بين 10 و 99 ونوضح ذلك في المثال التالي :

مثال (6) إنشاء القيد CHECK على العمود DEPTNO في الجدول الأقسام وتطبيقه على المستوى الجدول

```
SQL > CREATE TABLE dept (
2                    deptno    NUMBER(2)  ,
3                    dname     VARCHAR2(14)  ,
4                    loc       VARCHAR2(13)  ,
5      CONSTRAINT dept_deptno_ck CHECK(deptno BETWEEN 10
AND 99 ) ;
```

القيد على مستوى الجدول

Table Created

إضافة قيود إلى الجداول Adding Constraints :

يتم إضافة قيود للجدول بعد إنشائه باستخدام الأمر Alter Table كما في الصيغة التالية :

```
SQL > ALTER TABLE أسم الجدول
2      ADD CONSTRAINT أسم القيد نوع القيد
```

مثال (7) إضافة القيد FOREIGN KEY على العمود MGR في جدول الموظفين :

```
SQL > ALTER TABLE emp
2      ADD CONSTRAINT emp mgr fk
3      foreign key (mgr) REFERENCES emp (empno) ;
```

Table altered.

مثال (8) إضافة القيد Primary Key إلى العمود deptno في جدول الأقسام :

```
SQL > ALTER TABLE dept
  2       ADD CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno) ;
```

Table altered.

إزالة القيود من الجدول DROP :

وذلك باستخدام ALTER TABLE

مثال (9) إزالة القيد المسمى (emp_mgr_fk) من جدول الموظفين :

```
SQL > ALTER TABLE emp
  2       DROP CONSTRAINT emp_mgr_fk
```

Table altered.

كما نعرف أن جدول الأقسام له علاقة بجدول الموظفين وذلك بوجود مفتاح أساسي Primary key على العمود deptno في جدول الأقسام ومفتاح foreign key على العمود dept في جدول الموظفين وعندما نريد إزالة المفتاح الأساسي في جدول الأقسام والعلاقة مع جدول الموظفين أي إزالة foreign key من جدول الموظفين الوطفين نقوم بما يلي :

مثال (10) إزالة العلاقة بين جدولي الموظفين والإدارات عن طريق إزالة المفتاح الأساسي وتوابعه

```
SQL > ALTER TABLE DEPT
  2       DEOP Primary key CASCADE
```

Table Altered.

- **استعراض القيود المطبقة على جدول معين :**

كما نعلم أن أوراكل ينشئ جداول لتسجيل التغيرات التي تتم في قواعد البيانات وتسمى هذه الجداول DATA DICTIONARY ومن خلالها يمكن عرض القيود المطبقة على جدول معين

مثال (11) عرض القيود المطبقة على جدول الموظفين

```
SQL > SELECT Constraint name , Constraint _typy
  2     FROM user_constraints
  3     WHERE TABLE_NAME ='EMP'
```

CONSTRAINT_NAME	C
SYS_C00674	C
SYS_C00675	C
EMP_EMPNO_PK	P
FK_DEPTNO	R

فيما يلي معنى المحارف في الجدول

- الحرف **C** يعني ان نوع القيد هو **CHECK** .
- الحرف **P** يعني أن نوع القيد هو **Primary key** .
- الحرف **R** يعني أن وع القيد هو **Foreign key**.
- الحرف **U** يعني أن نوع اقيد هو **UNIQUE** .

أما نوع القيد NOT NULL فيظهر مثل القيد CHECK .

والمثال التالي يبين كيفية عرض القيود المطبقة على الأعمدة

مثال(12) عرض أسماء الأعمدة وأسماء القيود المطبقة عليها :

```
SQL > SELECT CONSTRAINT_NAME , COLUMN_NAME
  2         FROM   user_cons_column
  3         WHERE    Table_name  = 'EMP'   ;
```