الكامل

- 11g - في تعلم أوراكل

الكامل

في تعلم أوراكل - 11g -

الطبعة الاولى

2019

تأليف: وسام علي الخزاعي

دار ابن النفيس
للنشر والتوزيع

المملكة الأردنية الهاشمية
رقم الإيداع لدى دائرة المكتبة الوطنية
(2018/8/4244)

005,133
الخزاعي، وسام علي
الكامل في اوراكل(g11)/ وسام علي الخزاعي–عمان: دار ابن النفيس للنشر والتوزيع،2018
( ) ص.
ر.إ : 2018/8/4244
الواصفات: /لغات البرمجة//الحواسيب/

ISBN -978-9923-718-65-0

+962797135504
+962780080648
Dar ibn alnafees
dar_ibnalnafees@yahoo.com
alnafees02@gmail.com

إلى كل من يبحث عن العلم والمعرفة

علمتني الحياة ان العقل يتوقف عن النمو يوم يتوقف عن القراءة والتأمل، وانه لا جديد ولا مفيد في حياة من لم يقرأ، وان من هجر العلم والمعرفة فقد تعجل لنفسه الفناء. فأدركت لماذا كانت اول كلمة تنزل من الوحي ("أقرأ").

شكر وتقدير

إلى الأستاذ المساعد...

الدكتور عمار سلمان المسعودي...

لمراجعته الكتاب لغويا ... هذا هو ديدن العلماء صديقي.

المحتويات:

المقدمة:

أن هذا الكتاب الذي بين يديك يتعامل مع ما يعرف بقواعد البيانات التي تمثل البنية الأساسية لخزن ومعالجة المعلومات التي تخص مؤسسة أو جهة ما . وان هنالك العديد من البرامج التي تتعامل مع قواعد البيانات تلك ، ومن اشهر تلك البرامج هو برنامج أوراكل المصمم من قبل شركة أوراكل OracleCorporation والتي تعد من أهم وأضخم شركات تقنية المعلومات بشكل عام وقواعد البيانات بشكل خاص والتي تأسست في عام 1977 على يد Larry Alison.

يتناول هذا الكتاب في جزئه الأول ثلاثة فصول أساسية . حيث يتناول الفصل الأول البيانات وكيفية تجميعها وتشكيل قواعد البيانات منها بالإضافة إلى البيئات المختلفة التي من الممكن أن تعمل معها قواعد البيانات وابرز الشركات المصممة والمنتجة لبرامج قواعد البيانات . أما الفصل الثاني فيتحدث عن لغة الاستعلام البنيوية SQL Server ويشمل ذلك كل من عملها وهيكليتها والايعازات التابعة لها ، وعمل كل أيعاز من تلك الايعازات بالإضافة إلى السيطرة على المستفيدين الذين يستخدمون قاعدة البيانات من خلال معرفة كيفية أعطاء اسم المستخدم والرقم السري لكل مستخدم أو حذفه بالإضافة إلى كيفية منحه الامتيازات أو سلبها منه .

أما الفصل الثالث فيتناول اللغات الإجرائية المعروفة بالـ PL / SQL Server والتي تتضمن هيكليتها ، أجزائها ، كيفية بناء البرنامج، أنواع المتغيرات المستخدمة وكيفية تعريفها ، الدوال الممكن استخدامها..... الخ. هذا هو الجزء الأول من هذا الكتاب ، أما الجزء الثاني فسيتضمن أن شاء الله كل من النماذج

والتقـــارير Form & Report وكيفية السيطرة على قاعدة البيانات والتي تتضمن كل من Database Administer1 و Database Administer2 وفي الختام اشكر كل من ساعدني في تعلم هذه التقنية وكل من شجعني وساعدني في تأليف هذا الكتاب ومن الله التوفيق، مع التمنيات بالدعاء والتوفيق عسى أن تعم الفائدة للجميع.

الفصل الأول

قواعد البيانات

البيانات:

هي سلسلة غير مترابطة من الحقائق الموضوعية التي يمكن الحصول عليها عن طريق الملاحظة أو عن طريق البحث والتسجيل. وبشكل عام فالبيانات هي مجموعة من الحروف أوالكلمات أو الأرقام أوالرموز أوالصور (الخام) المتعلقة بموضوع معين، ومثال ذلك بيانات الموظفين التي من الممكن ان تشمل (الأسماء – الأرقام الوظيفية – المهن – الصور) بدون ترتيب، وينتج عن هذه البيانات بعد المعالجة ما يطلق عليه مصطلح معلومات.

تتكون البيانات من مجموعة من المواد الأولية (الخام) التي في صورتها الحالية، لا يمكن الاستفادة منها ولكن عن طريق المعالجة نستطيع الحصول على المعلومة. و يمكن القول أن المصدر الأساسي للبيانات هو الإنسان الذي يقوم بتجميع البيانات من خلال المشاهدة والملاحظة والتجربة على الواقع المحيط به سواء الاجتماعي أو الطبيعي أو الاقتصادي إلا أن في المجال الإداري يمكن القول أن مصدر البيانات يمكن أن يكون مصدر داخلي أو خارجي فمصدر البيانات الداخلي يقصد به البيانات المتجمعة من الإدارات المختلفة والأقسام والشعب والعاملين في مختلف جوانب النشاط في المنظمة مثل الفواتير وأوامر الشراء والشيكات الواردة والصادرة وأرقام المبيعات وغيرها وهذه البيانات تدون على شكل تقارير أو قد تكون ملاحظات ومناقشات مسجلة.

بينما يقصد بالبيانات التي تأتي من مصادر خارجية هي تلك البيانات التي تأتي من الزبائن والموردين ومن مختلف المنظمات ذات العلاقة مع المنظمة

المدروسة ومن السوق ومن ردود أفعال المستهلكين ومن مندوبي المبيعات ولجان الشراء ومن النشرات والدوريات المتخصصة والاتحادات وغيرها.

ولتحويل البيانات إلى معلومات يتطلب ذلك معالجة البيانات وذلك عن طريق الحصول عليها وتسجيلها ثم مراجعتها والتأكد من مطابقتها مع المصادر ثم تصنيفها إلى مجموعات أو فئات متجانسة 0 وفقا لمعيار معين وهناك العديد من المعايير يمكن استخدامها مثل تصنيف المستهلكين بحسب منطقة جغرافية أو إقليمية معينة ويجري التصنيف عادة على أساس نظام ترميز Codingsystem والذي قد يكون رقميا أو حرفيا أو باستخدام النوعين معا بحسب نوعية البيانات ثم يتم ترتيب البيانات بطريقة معينة تتفق والكيفية التي تستخدم بها تلك البيانات ثم يتم دمج مجموعة من عناصر البيانات وجمعها لكي تتوافق مع احتياجات مستخدميها ثم يتم استخدام العمليات الحسابية والمنطقية لتقديم بيانات جديدة ومفيدة للمستخدم وبعد ذلك يتم حفظها إلى وقت الحاجة إليها وتؤثر الوسيلة المستخدمة في حفظ البيانات على طريقة استرجاعها وكفاءة الاسترجاع ثم يتم بعد ذلك تقديم البيانات بشكل يمكن أن يفهمها ويستخدمها من يطلبها وقد يتم تقديمها على شكل تقرير مكتوب أو على شكل رسومات بيانية أو هندسية أو يتم عرضها على شاشة الحاسوب مباشرة.

المعلومات:

تعرف المعلومات على أنها البيانات التي تمت معالجتها بحيث أصبحت ذات معنى وباتت مرتبطة بسياق معين. والمعلومات مصطلح واسع يستخدم لعدة معاني حسب سياق الحديث، وهو بشكل عام مرتبط بمصطلحات مثل (المعنى، المعرفة، التعليمات، التواصل).

خصائص المعلومات:

1. خاصية التميع والسيولة، فالمعلومات ذات قدرة هائلة على التشكيل (إعادة الصياغة)، فعلى سبيل المثال يمكن تمثيل المعلومات نفسها في صورة قوائم أو أشكال بيانية أو رسوم متحركة أو أصوات ناطقة.
2. قابلية نقلها عبر مسارات محددة (الانتقال الموجه) أو بثها على المشاع لمن يرغب في استقبالها.
3. قابلية الاندماج العالية للعناصر المعلوماتية، فيمكن بسهولة تامة ضم عدة قوائم في قائمة أو تكوين نص جديد من فقرات يتم استخلاصها من نصوص سابقة.
4. بينما تتسم العناصر المادية بالندرة وهو أساس اقتصادياتها، تتميز المعلومات بالوفرة، لذا يسعى منتجوها إلى وضع القيود على انسيابها لخلق نوع من (الندرة المصطنعة) حتى تصبح المعلومة سلعة تخضع لقوانين العرض والطلب، وهكذا ظهر للمعلومات أغنياؤها وفقراؤها وأباطرتها وخدامها وسماسرتها ولصوصها.
5. خلافا للموارد المادية التي تنفذ مع الاستهلاك لا تتأثر موارد المعلومات بالاستهلاك بل على العكس فهي عادة ما تنمو مع زيادة استهلاكها لهذا السبب

فهناك ارتباط وثيق بين معدل استهلاك المجتمعات للمعلومات وقدرتها على توليد المعارف الجديدة.

6. سهولة النسخ، حيث يستطيع مستقبل المعلومة نسخ ما يتلقاه من معلومات بوسائل يسيرة للغاية ويشكل ذلك عقبة كبيرة أمام تشريعات الملكية الخاصة للمعلومات.

7. إمكان استنتاج معلومات صحيحة من معلومات غير صحيحة أو مشوشة، وذلك من خلال تتبع مسارات عدم الاتساق والتعويض عن نقص المعلومات غير المكتملة وتخليصها من الضوضاء.

8. يشوب معظم المعلومات درجة من عدم اليقين، إذ لا يمكن الحكم إلا على قدر ضئيل منها بأنه قاطع بصفة نهائية.

9. تتغير المعلومات بمرور الزمن وفقاً لأهميتها.

تحرير المعلومات:

تحرير المعلومات أو معالجة المعلومات كلمة تعني الكثير، فهي تشير إلى إن عملية تغير محتويات صورة أو ملف أو نص أو معالجة الصور وغير ذلك بهدف تحسين أو توضيح أفضل للمعلومات الأصلية. وتكون معالجة الصورإما معالجة ذات بُعد واحد مثل (إشارة الرادار) (أو صورة ذات بُعدين)أو ثلاثة أبعاد أو أكثر. وتشمل معالجة البيانات أو تحريرها ثلاثة أنواع رئيسية هي:

1- خزن البيانات (سواء مرتبة أو مصنفة أو مشفرة........ الخ).

2- استعادة البيانات المخزونة سواء بشكلها العادية أو بعد معالجتها.

3- إرسال البيانات واستقبالها وقد ترسل مشفرة أو مضغوطة ويتم فكها بعد استقبالها.

أنواع المعلومات:

1- المعلومات التطويرية أو الإنمائية: مثل قراءة كتاب أو مقال والحصول على مفاهيم وحقائق جديدة الغرض منها تحسين المستوى العلمي والثقافي للإنسان وتوسيع مداركه.

2- المعلومات الإنجازية : وهي المعلومات التي يحصل الإنسان من خلالها على مفاهيم وحقائق تساعده على إنجاز عمل أو مشروع أو اتخاذ قرار.

3-المعلومات التعليمية : وهذه تتمثل في قراءة الطلبة في مراحل حياتهم العملية للمقررات الدراسية والمواد التعليمية.

4-المعلومات الفكرية : وهي الأفكار والنظريات والفرضيات حول العلاقات التي من الممكن أن توجد بين تنوعات عناصر المشكلة.

5-المعلومات البحثية : وهي تشمل التجارب وإجرائها ونتائج الأبحاث وبياناتها التي يمكن الحصول عليها من تجارب المرء نفسه أو تجارب الآخرين ويمكن أن يكون ذلك حصيلة تجارب معملية أو حصيلة أبحاث أدبية.

6-المعلومات الأسلوبية النظامية : وتشمل الأساليب العلمية التي تمكن الباحث من القيام ببحثه بشكل أكثر دقة ويشمل هذا النوع من المعلومات الوسائل التي تستعمل للحصول على المعلومات والبيانات الصحيحة من الأبحاث والتي تختبر بموجبها صحة هذه البيانات ودقتها.

7-المعلومات الحافزة والمثيرة.

8-المعلومات السياسية:وهذا النوع من المعلومات مركز قضية وعملية اتخاذ القرار.

قاعدة البيانات Database:

هي مجموعة من عناصرِ البيانات المنطقية المرتبطة مع بعضها البعض بعلاقة رياضية، وتتكون قاعدة البيانات من جدول واحد أو أكثر. ويتكون الجدول من سجل(Record) أو أكثر ويتكون السجل من حقـل (Field) أو أكثر. ومثال عليه السجل الخاص بموظف معين يتكون من عدة حقول مثل رقم الموظف – اسم الموظف – درجة الموظف – تاريخ التعيين – الراتب – والقسم التابع له، وغير ذلك من بيانات الموظف تخزن في جهاز الحاسوب عَلى نحو منظّم، حيث يقوم محرك قاعدة البيانات (Database Engine) بتسهيل التعامل معها والبحث ضمن هذه البيانات، وتمكين المستخدم من الإضافة والتعديل عليها.

كما يتم استرجاع البيانات باستخدام أوامر من لغة الاستعلام حيث تعتبر معلومات تساعد في عملية اتخاذ القرار.

الفرق بين قواعد البيانات و نظام الملفات:

لو تسآلت لماذا نستخدم قواعد البيانات ؟ لماذا لا نستخدم ملف ونضع فيه معلوماتنا، الاجابه بسيطه. طرق التخزين ، المساحه ، سهولة الاستعلام.

- ✓ **طرق التخزين :**

في الملفات نقوم بالتخزين بطريقه صعبه ، فلو قلنا كل سطر يحتوي على معلومات مستخدم لدينا بالموقع ، وفي هذا السطر فواصل تفصل كل معلومه على حده مثل (الاسم ، العمر ، كلمة المرور) تخيل معي كيف ستقوم بقرائة هذا الملف وماهي الطريقة التي ستستخرج بها المعلومات ، طريقه طويله لتقوم بعمل هذا كله ولكن مع قواعد البيانات فهي سهله جدا ، فقط تقوم بتعيين نوع

الحقل الذي تريده وتضع به القيم وهي تقوم بعمل الباقي من ترتيب ومن استخراج ومن وضع كل قيمه لوحدها وغيرها من الشروط مثل عدم التكرار.

- ✓ **المساحه :**

تخيل موقع الياهو الذي يملك اكثر من مليون مستخدم ، كم ملف سوف يقوم بإستيعاب جميع هؤلاء المستخدمين ؟ وكم سوف تكون احجام الملفات ؟ هل هو شي منطقي ؟ غير ان استخراج معلومه ما سوف يكون بطيئ جدا فهو غير منطقي اساسا، لكن مع قواعد البيانات فهو مسهل بطريقه ممتازه، مرتب بطريقه جيده، ولا يأخذ حيزا كبيرا مثل الملفات ، وعند القراء يكون اسرع.

- ✓ **سهولة الاستعلام:**

لو نرجع الى مثالنا ، الملف الذي يحوي على اسماء المسجلين بالموقع ، تخيل انهم 1000 مستخدم ، كل مستخدم في سطر بالملف، لو اردت ان تقارن الاسم في الملف بالاسم الذي ادخله المستخدم فسوف تحتاج الى تقوم بقراء الملف كاملا اي 1000 سطر، لكي تقوم بإيجاد المعلومه المطلوبه ، سوف تقول ماذا لو كان المستخدم في اول الملف ، اذن ماذا لو كان في اخر الملف وكان رقم 1000 ؟ اعتقد ان الصوره وضحت.

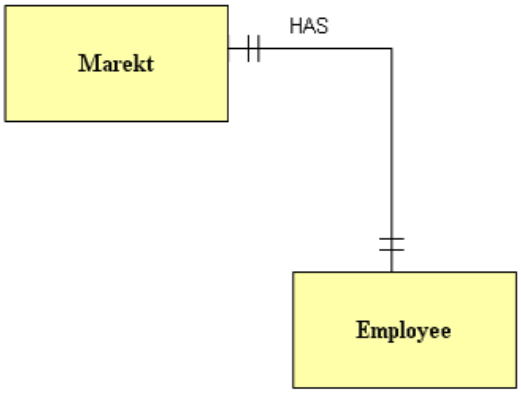
لكن مع قواعد البيانات فالاستعلام سهل جدا عن طريق اللغه المسماه SQL، فهي لغة منطقيه جدا وطريقتها سهله جدا فمجرد ان تقول اختر من الجدول (مستخدمين) القيمه التي تساوي (اسم المستخدم المدخل) فقط ، وسوف يقوم الاستعلام بأخبارك هل يوجد اسم المستخدم ام لا .

العلاقات في قواعد البيانات :

من اهم الاشياء التي يجب ان نراعيها في قواعد البيانات هي عدم التكرار ، اي نجعل كل شي فريد من نوعه، لن استطيع شرح هذا كله الان ، ولكن سوف اتكلم عن العلاقات في قواعد البيانات بشكل مختصر ، حيث ان لدينا ثلاث انواع من العلاقات هي :

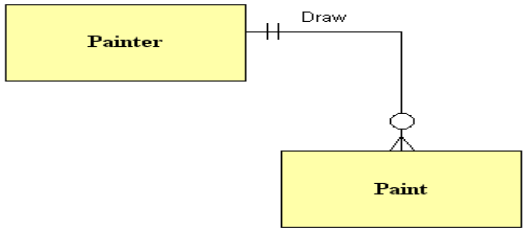
1- علاقة واحد لواحد One To One.

لنأخذ مثال على هذه العلاقه ، تخيل ان لدينا متجر ، وفي هذا المتجر يوجد عامل واحد فقط، فالعلاقه تكون ان الموظف ينتمي الى متجر واحد و المتجر يحتوي على موظف واحد فقط لاحظ الصوره التاليه.



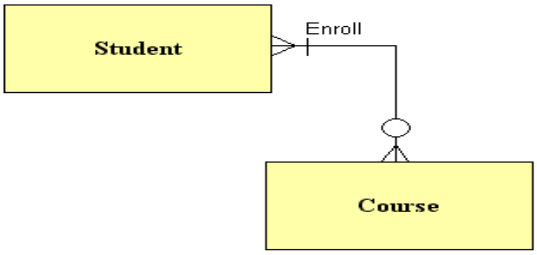
2- علاقة واحد لكثير One To Many.

تخيل ان لدينا رسام ، هذا الرسام يقوم برسم عدة لوحات ، ولكن كل لوحة من عمل رسام واحد ففي تلك الحالة سوف تكون لدينا العلاقة هي OneTo Many فقط لاحظ الصوره التاليه :

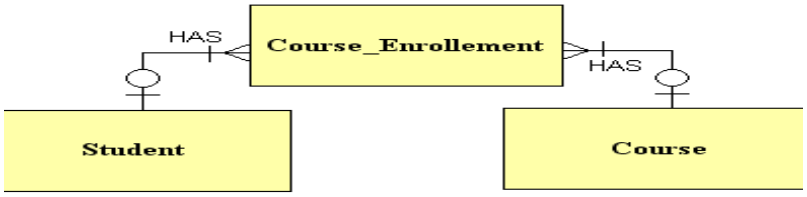


3- علاقة كثير لكثير Many To Many.

لو لدينا طلاب ولدينا مواد ، فكل طالب يمكنه ان يأخذ ماده او اكثر ، وكل ماده يمكن ان يأخذها اكثر من طالب اليس ذلك صحيحا ؟ ولكن هذه العلاقه ليست جيده لانها لن تحل مشكلة التكرار ، ولكن نقوم بحلها عن طريق شي يسمى (Bridge Entity) او (الجسر) لاحظ الصورتين التاليه:



ولكي نقوم بحل هذه العلاقه او ايجاد طريقه اخرى لها نستخدم الجسر مثل الصوره التاليه:



طرق ربط العلاقات في قواعد البيانات:

ان كل جدول نقوم بإنشاءه يجب ان نجعل له ما يسمى المفاتح الرئيسي (PRIMARY KEY) وهذا المفتاح لا يتم تكراره في الجدول ، حيث ان كل مستخدم لديه رقم رئيسي نعرفه به ، ونحدد به اسمه ومعلوماته الاخرى ولكي نقوم بربطه في علاقة ، في الجدول المربوط به نجعل هذه القيمه على شكل مفتاح اجنبي (FOREIGN KEY) لاحظ معي الصوره التاليه :

CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

ORDERS

OrderID	CustomerID	Amount	Date
1	3	27.50	02-Apr-2000
2	1	12.99	15-Apr-2000
3	2	74.00	19-Apr-2000
4	4	6.99	01-May-2000

لاحظ ان العمود CustomerID في جدول CUSTOMERS هو مفتاح رئيسي ولا يمكن تكراره ، ولكنه في جدول ORDERS على شكل (FOREIGN KEY) ويمكن تكراره ، لان عدة طلبات يمكن عملها عن طريق مستخدم واحد ، ولكن كل طلب مربوط بمستخدم واحد فقط.

تصميم قاعدة بيانات: في تصميم قاعدة البيانات يجب علينا ان نقوم بأشياء كثيره ، اولها ان نقوم بتحليل ومعرفة ماذا نريد من هذه القاعده ، كيفية عملها ، كيفية ترتيب المعلومات المفيده لنا بالقاعده ، ويجب ان نقوم برسم القاعده على الورق اولا لكي نستنتج هل منطقنا صحيح في تصميم هذه القاعدة ام لا. ويجب

ان نحل مشكلة التكرار ، فتخيل لو كان لدينا جدول يحتوي على معلومات العميل ومعلومات الموظف الذي قام بخدمته ، في كل مرة ندرج مستخدم يجب ان ندرج العميل مرة اخرى ولو قام العميل بتغيير اسمه او رقمه ، فسنحتاج الى تغيير كل الصفوف التي يوجد بها هذا العميل ، ولكن لو جعلنا العميل في جدول والموظف في جدول ، فسيكون لكل عميل صف واحد في جدول العملاء ولكل موظف صف واحد في جدول الموظفين ثم نربطهما ببعض. فأذا قام العميل بتغيير رقم هاتفه سوف نقوم بالتعديل مرة واحدة فقط.

أجزاء قاعدة البيانات:-

الجداول. من ناحية المظهر يشبه جدول قاعدة البيانات جدول البيانات حيث يتم تخزين البيانات في صفوف وأعمدة. لذلك في معظم الأحوال يكون من السهل استيراد جدول بيانات إلى جدول قاعدة البيانات. أما نقطة الاختلاف الرئيسية بين تخزين البيانات في جدول بيانات أو تخزينها في جدول قاعدة بيانات فتكمن في كيفية تنظيم البيانات. للحصول على أعلى معدل من المرونة في قاعدة بيانات، يجب تنظيم البيانات في جداول بحيث لا يحدث تكرار. على سبيل المثال، إذا كنت تخزن معلومات عن الموظفين، يجب أن يتم إدخال كل موظف مرة واحدة فقط في جدول تم تعيينه ليتضمن بيانات الموظفين. يتم تخزين بيانات المنتجات في الجدول الخاص بها، ويتم تخزين البيانات الخاصة بالمكاتب الفرعية في جدول آخر. تسمى هذه العملية بالتسوية. حيث تتم الإشارة إلى كل صف في الجدول كسجل.

السجلات والحقول. السجلات هي أماكن تخزين قطع المعلومات الفردية. يحتوي كل سجل على حقل واحد أو أكثر. تقابل الحقول الأعمدة الموجودة في الجدول. على سبيل المثال، قد تمتلك جدولاً يسمى "الموظفون" حيث يحتوي كل سجل

(صف) على معلومات عن موظف مختلف، ويحتوي كل حقل (عمود) على نوع آخر من المعلومات، كالاسم الأول واسم العائلة والعنوان وهكذا. يجب تعيين نوع بيانات محدد لكل حقل، سواء أكان نص أو تاريخ أو وقت أو عدد أو نوع آخر. يمكن وصف السجلات والحقول بطريقة أخرى عن طريق تصور كتالوك بطاقات ذو نمط قديم خاص بالمكتبة. تتقابل كل بطاقة في الخزانة مع سجل في قاعدة البيانات. وتتقابل كل قطعة من المعلومات في بطاقة مفردة (الكاتب والعنوان وما إلى ذلك) مع حقل في قاعدة البيانات.

نظام إدارة قواعد البيانات. هو البرنامج الذي يتم من خلاله استرجاع البيانات، أو الإضافة أو التعديل عليها، أو حذفها، حيث يقوم البرنامج بالربط بين المستخدم وبين محرك قاعدة البيانات، لأداء تلك المهمة.وفى حال وجود علاقة بين جداول قاعدة البيانات يسمى هذا بنظام قواعد البيانات العلائقيةRelational (Database Management System RDBMS) ، وان الهدف الأساسي لقواعد البيانات هو التركيز على طريقة تنظيم البيانات وليس على التطبيقات الخاصة .أي أن الهدف الرئيسي لمصمم قاعدة البيانات هو تصميم البيانات بحيث تكون خالية من التكرار ويمكن استرجاعها وتعديلها والإضافة عليها دون المشاكل التي يمكن أن تحدث مع وجود التكرار فيها. يتم ذلك عن طريق أيجاد ثلاث مستويات من التجريد أو النماذج لقواعد البيانات تسمى نماذج التطبيع (Normalizing Forms)، ويقصد بها جعل تركيبة البيانات أقرب للطبيعة التصنيفية.

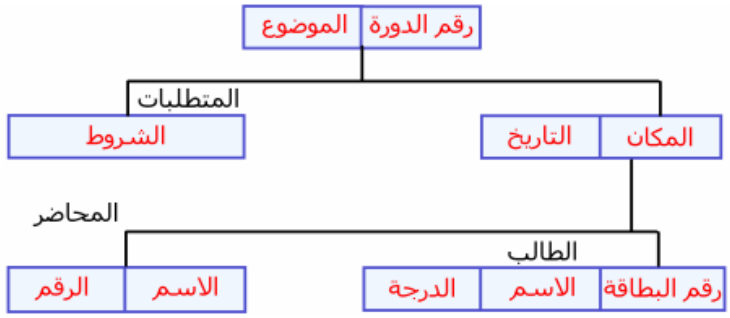
أنواع نظم إدارة قواعد البيانات:أن نماذج البيانات هي تمثيل بيانات العالم الحقيقي بصورة يسهل استخدامها بواسطة الحاسب وهناك أنواع من نماذج

البيانات تتوقف على نظام إدارة قواعد البيانات المستخدم وكذلك على طبيعية البيانات وتبعا لأنواع نماذج البيانات.

فهناك ثلاثة أنواع شائعة من نظم إدارة قواعد البيانات وهي.

- ✓ نظم إدارة قواعد البيانات الهرمية Hierarchical DBMS .
- ✓ نظم إدارة قواعد البيانات الشبكية Network DBMS .
- ✓ نظم إدارة قواعد البيانات العلائقية Relational DBMS .

1- نظم إدارة قواعد البيانات الهرمية: Hierarchical DBMS : ان قواعد البيانات الهرمية أو النظم الهرمية Hierarchical DBMS تقوم بتنظيم البيانات على شكل هرمي أو علي شكل شجرة مقلوبة أي جذرها في القمة وتخرج منها الفروع . شأن هذه التركيبة شأن شجرة الأسرة فلها جد واحد والجد له عدة أبناء و الأبناء هم أباء الأحفاد ويستحيل وجود حفيد له اكثر من أب .وهذا شكل توضيحي ليوضح لك النظم الهرمية وتفرعاتها.

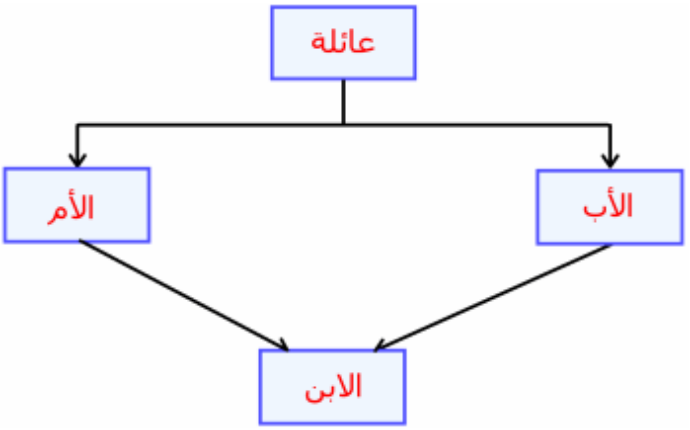


والملفات الهرمية هي ملفات لها نفس البناء الشجري ولها نفس العلاقات بين السجلات مثالا لبعض أنواع السجلات التي يمكن أن تتواجد في تكوين هرمي فهناك سجلات مبيعات متعددة لكل بائع حيث يوجد سجل إحصائيات واحد لكل عملية جارية كما يوجد أيضا سجلات عديدة للعملاء لكل بائع حيث أن كل

بـائع لـه عمـلاء محـددين ويمكـن أن يكـون لكـل عميـل عـدة سـجلات حسـابات مدينين سجل واحد لكل عملية شراء لم يتم تسديد ثمنها. ومن المهم أن نفهم انه ليس من الضروري أن تتصل كل الملفات الموجودة في قاعدة البيانات مع بعضـها. وكل ما هو مطلوب أن تتصـل الملفات التـي تسـتخدم كمجموعـة مـع بعضـها فـي التطبيقـات. وسـجلات المبيعـات السـابقة لهـا مثـل هـذه العلاقـة المنطقية تسمى فئة . والفئة Set عبارة عن مجموعة من السجلات متصلة مع بعضـها منطقيا، وعلـى هذا تصبح قاعدة البيانـات الهرميـة عبـارة عن تجميـع لملفـات وفئـات ملفـات متصـلة مـع بعضـها منطقيـا. ويسـتخدم نظـام إدارة المعلومات IMS الذي أعدته شركة IBM التكوين الهرمي وهو من اكبر نظم إدارة قواعد البيانات DBMS الموجودة حاليا واعقدها. ولهذا السبب فأنه يتطلب مستوى رفيـع مـن الخبـرة لإمكانيـة بنائـه وعلـى أي حـال فهـو قـوي واثبـت كفـاءة كبيرة في معاملة قواعد بيانات كبيرة جدا كما انه يقدم إجراءات استرجاع و أمن جيدة هذا بالإضافة إلى إمكانية استخدامه في نظام الاتصال النشط من خلال شبكة الاتصالات.

2 – نظم إدارة قواعد البيانات الشبكية Network DBMS .

رغم أن كلمة الشبكة استخدمت كثيرا في شبكات الحاسب ومعالجة البيانات فقد وجد من الأفضل استخدام مسمى قواعد البيانات الضفيرة Plex رغم أن مسمى قواعد البيانات الشبكية لازال شائع الاستخدام. ويتغلب هيكل بيانات التركيب الشبكي على معوقات التكوين الهرمي الذي لا يسمح للابن أن يكون له اكثر من أب واحد ويظهر ذلك في الشكل التوضيحي للتكوين الشبكي .



ومثل هذا النوع من قواعد البيانات حل كثيرا من مشاكل العلاقات فإذا فرضنا أن هناك اكثر من مورد يورد قطع غيار فإن كل مورد قادر على توريد اكثر من نوعية قطعة غيار وبالتالي فإن كل قطعة غيار يوردها اكثر من مورد مما يحتم لفهم المثال عرض العلاقة بين قطعة الغيار و الموردون.

ان اوجه التشابه بين نظم قواعد البيانات الشبكية و نظم قواعد البيانات الهرمية إنها تتطلب إلى ذاكرات ذات أحجام كبيرة وعادة تحتاج إلى لغات راقية لبرمجتها وهي صعبة التعلم ولها مزايا كثيرة فهي بالطبع اكثر كفاءة من قواعد البيانات العلائقية وتتعامل مع كم كبير جدا من البيانات و المعلومات بالإضافة إلى إنها توفر بناء على طريقة تنظيم البيانات التي تتبعها مساحات كبيرة من وسائط تخزين البيانات.

3- نظم إدارة قواعد البيانات العلائقيةRelational DBMS .

أثبتت الأيام صحة القول الشائع أن الأبسط هو الأجمل والأكفأ . فكلما كان سكنك بسيط وكلما عشت في بساطة و بعدت عنك المشاكل وكلما كانت الآلة بسيطة سهلت إدارتها وصيانتها. وهذا ما أكده التعامل مع قواعد البيانات الهرمية و الشبكية التي تعقدت ملفاتها وأساليب إدارتها لدرجة كادت تؤدي بها كلما أضيفت تطبيقات جديدة أو متطلبات جديدة تحتاج مؤشرات جديدة مما ضخم

منها وعقدها. وهذه المشاكل كانت المنطلق للبحث عن حلول تحقق جملة أهداف منها:

أ- يمكن فهم قاعدة البيانات لمن لم يدرسوا علوم الحاسب.

ب- يمكن تعديل وإضافة وحذف بيانات دون تغيير المخطط المنطقي للقاعدة.

ت- تتيح للمستخدم اعلى درجة من المرونة في التعامل مع البيانات.

في عام 1970 أستحدث E.E.Codd أسلوبا لتنظيم وفرز بيانات قواعد البيانات. وهي قواعد البيانات العلائقية. وقد وجد العالم الأمريكي E.E.Codd أن هذا لا يتحقق ألا برص البيانات على هيئة جداول لان الإنسان تعود على الجداول منذ طفولته بداية من جدول الحصص إلى جدول الضرب إلى كشف الأسماء و الدرجات.

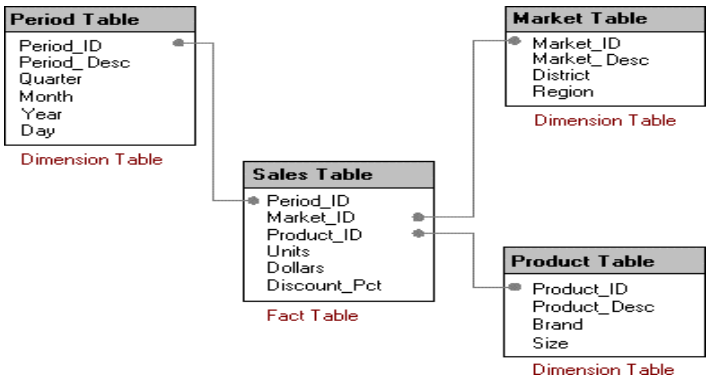
وهذه النظم تتعامل مع اكثر من ملف في نفس الوقت وتعامل البيانات داخل الملف كما لو كانت جدولا مكونا من صفوف و أعمدة ويسمى علاقة Relation وتمثل أعمدة الجدول حقول قاعدة البيانات Fields وتسمى أيضا Attributes بينما تمثل صفوفها سجلات قاعدة البيانات وتسمى Tuples و النظام العلائقي Relationيقوم بربط البيانات بين العلاقات بناء على حقل مشترك بينهما.

والنظم العلاقية قامت أساسا على النظريات العلائقية في الرياضيات وقد بدأ تطبيقها على الحاسبات الكبيرة أولا مثل SQL. ORACLE ثم ظهرت عدة نظم علائقية على الحاسبات الشخصية PCs مثل برامج .DBaseII. DBaseIII .DBaseIII+. DBaseIV. FoxBase. FoxPro.

ويمكن القول عن هذا النوع من قواعد البيانات مايلي:

تنظيم البيانات في قواعد البيانات العلائقية في جداول ذات بعدين ويمكن اعتبار كل جدول ملف ويستخدم مصطلح ملف مسطح Flat File لان محتويات الملف مرتبة على محورين (س ، ص) فقط.

نشأت مجموعة جديدة من المصطلحات تستخدم في وصف قواعد البيانات العلائقية هذه المصطلحات التي تستخدم في وصف قواعد البيانات الهرمية أو الشبكية ففي النموذج العلائقي يستخدم مصطلح نموذج بيانات علاقية جزئي أو رؤية لبيانات علائقية Relational Data Submodel or View بدلا من المخطط الجزئي Subschema ومصطلح رؤية View مناسب فهو لجزء المستفيد من قاعدة البيانات كما استخدمت بالإضافة إلى ذلك أسماء لوصف مكونات الملفات المسطحة ويوضح الجدول التالي عينة لملف ويشار إلى أعمدة الملف بأنها مسطح رأسي (نطاق) والى الصفوف بأنها مسطح أفقي و الجدول عبارة عن تجميع من المسطحات الأفقية خاصة بموضوع معين و الجدول خاص بالبائعين ويمكن استخدامه في توفير أسمائهم ومبيعاتهم منذ بداية العام.



وقد وضحنا المشاكل السابق ذكرها نظرا لأنه أمكن تجنبها في قواعد البيانات العلائقية فالتكوين العلائقي تكوين منطقي يستخدم علاقات ضمنية Implicit Relationships بدلا من استخدامه لعلاقات صريحةExplicitRelationships وهي التي تستخدم في كل من قواعد البيانات الهرمية والشبكية.

وحتى نوضح مفهوم العلاقات الضمنية بين ملفات قاعدة البيانات العلائقية وكيفية استخدامها في تجميع البيانات مع بعضها من ملفات منفصلة عن بعضها نفرض أن لـــديــنا جـــدولين فـــي قاعـــدة البيانـــات جـــدول [أ] وجـــدول [ب] . جدول [أ] يعرف منطقة المبيعات لكل بائع باستخدام رقم البائع كحقل مفتاحي و الجدول [ب] يحدد اسم كل بائع و الجدولان منفصلان عن بعضهما أي لا يوجد أي اتصال طبيعي بينهما وتحدد العلاقة ضمنيا وذلك بإدخال حقل رقم البائع في كل من الجدولين .

وبذلك نكون قد استعرضنا التسلسل التاريخي وانتهينا من تعريف نظم إدارة قواعد البيانات DBMS وأنواعها الأكثر انتشارا وفائدتها لإدارة البيانات.

الفرق بين الهرمية والشبكية والعلائقية:

يستخدم النموذجان الهرمي والشبكي روابط (links) أو مؤشرات (pointers) لوصل السجلات بعضها ببعض في النظام ، وتدعى هذه الأنظمة بالأنظمة الستاتيكية (ststic) أوالمتراصة (monolithic) لأن السجلات فيها مربوطة ببعضها بشكل فيزيائي من خلال تعاريفها ، وتتميز هذه الأنظمة بأنها معقدة العمل وصعبة التعديل ، إلا أن سرعة الوصول فيها تغطي عيوبها.

أما في الأنظمة العلائقية فالربط بين السجلات لا يجري فيزيائياً عن طريق المؤشرات ، وإنما عن طريق الأسماء الحقيقية للحقول ، كحقل رقم الموظف ID أو حقل الإسم أو حقل رقم البطاقة ..الخ ، فالسجلات في هذا النظام قابلة للعنونة بمحتوياتها (connect–addressable) بحيث يجري الوصول إليها بمطابقة قيم البيانات المخزنة مع بعضها.

مفاتيح الجداول:

هنالك اربعة انواع من تلك المفاتيح ، وهي مبينة في ادناه :

أولاً: المفتاح الرئيسي Primary Key .

وهو المفتاح الذي يحدد بشكل وحيد ومتفرد بحيث يتميز عن غيره، فلا تتكرر قيمته في أكثر من حقل واحد، ولا يقبل أي قيمة Null أي لا يمكننا أن نترك الحقل فارغاً بدون قيمة.

ثانياً : المفتاح المركب أو المجمع Composite Key .

وهو المفتاح الذي يستخدم لتعريف السجل بشكل وحيد ومتفرد، ولكنه يختلف عن المفتاح الرئيسي بأنه يشمل على أكثر من صفة حقل .(مثال على ذلك :لو كان لدينا جدول فيه أسماء الطلاب وأسماء المواد التي يدرسونها إضافة إلى علاماتهم فأنه لا يمكن اعتبار اسم الطالب واسم المادة أو العلامة كمفتاح رئيسي يحدد السجل بشكل وحيد ومتفرد، فيتم اللجوء في هذه الحالة إلى اعتبار اسم الطالب مع اسم المادة مفتاح مركب، على اعتبار أن اسم الطالب قد يتكرر واسم المادة قد يتكرر، ولكن اسم الطالب مع اسم المادة كمفتاح مركب لن يتكرر .

ثالثاً: المفتاح المرشح Candidate Key .

عند البدء بتصميم الجدول يتم ترشيح عدد من الحقول الصفات كي تصبح مفاتيح رئيسية، وعند إدخال البيانات، قد يتبين أن هذه المفاتيح يمكن أن تأخذ قيمة Null، فالمفتاح الذي يأخذ قيمة Null يُستثنى، والمفاتيح التي لا تأخذ قيمة Null ولا تكرر تبقى وتصبح مفاتيح رئيسية، بمعنى آخر : فإن المفتاح المرشح هو الصفة أو مجموعة الصفات التي يتم اختيارها وفحصها حتى يتقرر فيما بعد أنها ستبقى مفاتيح مرشحة أو يتم اعتمادها كمفتاح رئيسي.

رابعاً: المفتاح الأجنبي Foreign Key .

وهو عبارة عن حقل صفة أو أكثر يستخدم للربط بين جدولين، وسُمي المفتاح الأجنبي بهذا الاسم لأنه ليس من الحقول الموجودة أصلاً في الجدول، أي أنه عبارة عن حقل أو أكثر تُضاف إلى جدول لربطه مع جدول آخر . سيتم شرح تلك المفاتيح بصورة اوسع في الفصول اللاحقة ان شاء الله.

مكونات نظام إدارة قواعد البيانات.

يتكون نظام إدارة قواعد البيانات من اربعة اجزاء رئيسية هي :

- ✓ لغة نمذجة Modeling language: وتستخدم لتعريف رسم قاعدة البيانات.
- ✓ بنية البيانات أو هياكل بيانات Data structures : وتتكون من (جدول، سجل، حقل وبطاقية) وتكون مصممة بطريقة فعالة من أجل التعامل مع كمية ضخمة من البيانات
- ✓ لغة استعلام Query language : وتستخدم لتمكين المستخدمين، حسب صلاحياتهم، من مسائلة قاعدة المعطيات بطريقة مباشرة وتحليل البيانات وتعديلها وتغذيتها بالجديد.

✓ آلية تعامل Transaction mechanism: وتتضمن خصائصالـ ACID .

نماذج قواعد البيانات:-

نموذج قاعدة البيانات المترابطة هو نموذج تصميمي لقاعدة البيانات يعتمد على المنطق الضمني. ظهر هذا النموذج ضمن ورقة علمية نشرها العالم إدجار كودعام 1970.

ففي القرن التاسع عشر، قام عالم الرياضيات الألماني جورج كانتوربتقديم نظريته التي عرفت باسم نظرية المجموعات. عند اختراع الحاسوب، تم توظيف هذه النظرية في ابتكار أشكال للبيانات يمكن تمثيلها والتعامل معها من خلال الحواسيب، وقد تحقق هذا الهدف في الصيغة التي طرحها عالم الرياضيات الأمريكي د. ل. شيلدز en:D. L. Childs عام 1969 في ورقة علمية بعنوان "وصف تنظيري لبنى البيانات الفئوية "Description of Set-Theoretic Data Structure"، وقد أعتمد عالم الرياضيات والحاسوب البريطاني إدجار كود على هذه الورقة في أعداد ورقته العلمية التي اسماها "نموذج مترابط لبيانات بنوك البيانات الكبيرة المشتركة A Relational Model of Data for Large Shared Data Banks" التي قدم فيها لأول مرة نموذج قاعدة البيانات المترابطة.

يعتبر إدجار كود المبتكر الأصلي لنموذج قاعدة البيانات المترابطة، لكن هذا النموذج خضع للتطوير على يد عدد من الباحثين، ويعتبر جزء كبير من تطوير نموذج قاعدة البيانات المترابطة عائد إلى أبحاث كريس داتChrisDate وهيو داروينHugh Darwin.

في كتابهما المنشور عام 1995 والمسمى" البيان الثالث The Third Manifesto"بين كريس داتوهيو داروينكيف يمكن تكييف نموذج قاعدة البيانات

المترابطة ليكون متوائماً نموذج للكائنات ObjectOriented دون تغيير في المبادئ التي بني عليها.

بيئة قواعد البيانات:-

هنالك ثمانية أنواع من بيئات قواعد البيانات، وهي مبينة في أدناه.

1- نظام أوراكل لإدارة قواعد البيانات المترابطة (العلائقية).

أن شركة أوراكل Oracle Corporation هي واحدة من أضخم وأهم شركات تقنية المعلومات بشكل عام وقواعد البيانات بشكل خاص. تأسست شركة أوراكل في العام 1977على يد" لاري اليسون "ولدى الشركة مراكز خدمة للعملاء في أكثر من 145 دولة. يعمل لاري اليسون كمدير تنفيذي لشركة أوراكل لعدة سنوات الآن وابتداءً من العام 2003 عمل لاري كرئيس لمجلس إدارة الشركة ذاتها. أعجب لاري بالورقة التي كتبها" إدجار كود "والتي تناول فيها قواعد البيانات وبالخصوص قواعد البيانات ذات الحجم الكبير كقاعدة بيانات نظام التأمين الاجتماعي والتي عادة ما تضمّ العدد الهائل من المعلومات، فقام لاري بإنشاء شركة أوراكل ليتسنّى له تطبيق قاعدة البيانات التي وصفها ايدجار في ورقته البحثية. لم تكن الشركة التي أسسها لاري آنذاك تعرف باسمها الحالي أوراكل، ولكن كان اسمها في العام 1977" مختبرات تطوير البرامج، أراد لاري أن يجعل قاعدة البيانات "أوراكل" متطابقة مع قاعدة بيانات شركة" آي بي أم " والتي تعرف آنذاك بنظام R لقاعدة البيانات إلا ان شركة "آي بي أم" حالت دون ذلك بجعل الرسائل المتعلقة بالأخطاء الصادرة من قاعدة بيانات النظام R سريّة.

أن قاعدة البيانات أوراكل هي المنتج الرئيسي لشركة أوراكل وبدمج البرمجية جافا مع قاعدة البيانات أوراكل مكّن قاعدة البيانات من استخدامها لخوادم الويب

وتمكين المبرمجين من إضافة برامجهم الخاصة على قاعدة بيانات أوراكل ليتصرفوا بشكل أفضل ويتحكموا بمخرجات البرامج التي يستعملونها على الويب . تنتج شركة أوراكل برامج مساندة لقاعدة البيانات كـ (مُصمم أوراكل) و(مُطوّر أوراكل) وتقوم هذه الأدوات البرمجية بالمساعدة على كتابة برامج تتعلق بقاعدة البيانات أوراكل بشكل أفضل وأسرع. يقع المركز الرئيسي لشركة أوراكل في مدينة "ريد وود" في سان فرانسيسكو الواقعة على الساحل الغربي من الولايات المتحدة الأمريكية في ولاية كاليفورنيا .وتجدر الإشارة إلى أن كلمة "أوراكل" تقابل كلمة العرّاف أو العرافة بالعربية، والمقصود بأوراكل هو الذي/التي لها إطّلاع على الأمور الغيبية نتيجة المعرفة، الحكمة أو الاتصال بالآلهة.

2- My SQL.

هو نظام إدارة قواعد البيانات العلائقي يعتمد التعامل معه على لغة إس كيو إل . وسمي بهذا الاسم تبعا لابنة مبرمجهالأصلي Michael Widenius ، والتي اسمها .My ماي إس كيو إل . يعتبر الـ My SQL هو من المنتجات مفتوحة المصدرينشر كوده المصدر تحت رخصة العمومية بالإضافة إلى بعض الاتفاقيات الاحتكارية. كانت تمتلكه وترعاه الشركة الربحية السويدية MySQL AB لكن تملكه الآن شركة صن ميكروسيستمز Sun Microsystems والتي هي حاليا فرع من أوراكل.

3- فيربيرد Ferbird .

هي محرك قاعدة بيانات علائقية مفتوحة المصدر. تم تطويرها من النسخة مفتوحة المصدر من قاعدة البيانات إنتربيز Interbers التي أنتجتها شركة بورلاند Porland.وهي تمتاز بخفتها وسهولة تثبيتها وأنها موجودة في أكثر من منصة نظام تشغيل، مثل وندوز، لينكس وماكنتوش.

وان الموقع الرسمي لها هو:

/http://www.firebirdsql.org

ويمكـن التعامـل مـع قواعـد بياناتهـا بواسـطة بـرامج إدارة قواعـد البيانـات المصممة لها مثل:

1- Flamerobin.

2- TurboBird.

4- مايكروسوفت أكسس Microsoft Access .

هو برنامج لإدارة قواعد البيانات من تطوير شركة مايكروسوفت .يأتي البرنامج مرافقا لحزم مايكروسوفت أوفيس Microsoft Office كجزء منها وله واجهة رسومية. كانت هناك عدة إصدارات للبرنامج، فأولها كان مع أوفيس 97 ثم أوفيس 2000 وأوفيس 2003 وآخر إصدار هو مع أوفيس 2016. يتميز البرنامج بقدرته على استدعاء البيانات من نظم مختلفة لقواعد البيانات، كقواعد بيانات أوراكل و SQL وأي قاعدة بيانات مفتوحة الاتصال .(ODBC) يستعمله مطورو البرامج وعلماء البيانات لصنع قواعد بياناتية معقدة، ووصلها مع مختلف أنواع البرامج المستدعية، وينطوي تحت ذلك البرامج الكائنية وبرمجيات الإنترنت. بينما يستطيع المبتدئون أن يستعملوه لصنع قواعد بيانات صحيحة أو إنشاء تقارير عنها.

كذلك تم اعتماد شهادات أكثر المبرمجين والمصممين على برنامج أكسس دولياً، وذلك لزيادة طلب أكثر الشركات لهؤلاء المصممين، كما أن هناك العديد من المنتديات على الشبكة العنكبوتية لتدريس وتعليم الراغبين في تعلم هذا البرنامج الرائع والذي يمتاز بدعمه للغة العربية.

5- Postgre SQL.

هو نظام إدارة قواعد البيانات العلائقي يعتمد التعامل معه على لغة إس كيو إل وقد تم إصدارها بموجب ترخيص معهد ماساتشوستس للتكنولوجيا وبالتالي فهو يعتبر من البرمجيات مفتوحة المصدر. كما هو الحال مع العديد من البرامج المفتوحة المصدر لا تخضع للسيطرة من قبل أي شركة.

6- قاعدة بيانات بيركيلي (Berkeley DB).

هي قاعدة بيانات مجانية ومفتوحة، تحتوي على نظام عالي وسريع لإدارة البيانات. ميزة بيركيلي عن قواعد البيانات المعهودة الأخرى مثل اس كيو ال (SQL)هو أنه لا يوجد عمليات بحث أو فهرسة عن طريق جهاز ثالث أو شبكة معلومات، وإنما كل العمليات تتم محليا على جهاز المستخدم.

الفرق الرئيسي الآخر هو أن حفظ البيانات يتم في عدة قواعد بيانات منفصلة ذو حجم صغير، يمكن الربط فيما بينها، على عكس الطرق المعهودة في هذا المجال والتي تعمل جداول كبيرة ذو تركيبة معقدة في قاعدة بيانات واحدة ذو حجم كبير. ويمكن شرح قواعد بيانات بيركيلي بأنها جدول به عمودين من البيانات : عمود يحتوي على مفتاح البيانات (عادة رقم أو اسم محدد) وعمود به البيانات. بيركيلي يوفر واجهة (API) مكتوبة في عدة لغات برمجة، منها C++ Java C، C و XML. وتعمل على أنظمة لينكس و ويندوز.

7- Microsoft SQL Server .

هو برنامج لقواعد البيانات العلائقية من إنتاج مايكروسوفت، لغة الاستعلام الرئيسية فيه هي SQL و T-SQL ، وان النسخ التي أنتجت من Microsoft SQL Server هي.

- ✓ SQL Server 6.5.
- ✓ SQL Server 7.
- ✓ SQL Server 2000.
- ✓ SQL Server 2005.
- ✓ SQL Server 2008.
- ✓ SQL Server 2012.
- ✓ SQL Server 2013.
- ✓ SQL Server 2014.

8- Sybase.

أصبحت سايبيس ثاني نظام لقواعد البيانات وراء أوراكل، وبعد إجراء صفقة مع مايكروسوفت لتبادل الـ source code، لكي تستطيع مايكروسوفت التسويق على نظام التشغيل(OS 2) مزود الخدمة في ذلك الوقت، سايبيس سمى خادم قاعدة البيانات (Sybase SQL Server)، حتى الإصدار 4.9، سايبيس ومزود خادم مايكروسوفت كانت متطابقة تقريبا، وبسبب خلافات بين الشركتين حول تقاسم العائدات، قررت سايبيس ومايكروسوفت تقسيم الأكواد وذهب كل منهما بطريقه الخاصة، على الرغم من أن التراث المشترك واضح جدا للعمليات في T) (SQL –إجرائية اللغة فضلا عن البنية الأساسية العملية. الفرق الكبير هو أن سايبيس لديه تراث يونكس، في حين أن مايكروسوفت SQLServer تم تكييفها مع Microsoft Windows NT operating system فقط ،كما استأنفت سايبيس تقديم إصدارات لـ Window، وأصناف عدة لـ Unix ولـ Linux

سـايبيس عـانى مـن تراجـع كبيـر فـي ثروتـه فـي أواخـر 1990 عنـدما بـدأت Informix البيع أكثر بهامش كبير، مع ذلك تم الحصول على Informix من قبل IBM في عام 2001، ولم تعد تنافس كشركة مستقلة، وفى نوفمبر عام 2005 يؤرخ الكتاب الذي كتبه موظفو Informix عن تاريخ المعركة بين سايبيس و Informix منذ وقت طويل.

اعتبارا من عام 2006 أوراكل تعد الشركة الرائدة في سوق قواعد البيانات بحصة العائدات، تليها IBM ، ثم مايكروسوفتSQL SERVER ، ثم سايبيس من وراء منافسيها الرئيسيين بـ 3% من حصة السوق، الاستثمارات المصرفية هي واحدة من أكبر القواعد العملاء لسايبيس، ولا تزال بصمة سايبيس في منشآت البورصة وضبط الإجراءات هي أكبر بصمة ، حيث ان سايبيس لا تزال الأقوى في نظم قواعد البيانات.

قواعد البيانات العلائقية: هي عبارة عن قطع من الأنظمة لإدارة قواعد البيانات العلائقية والتي تدعى بالـ Relational Database Management System (RDBMS).

والتي تتضمن ثلاثة عناصر رئيسية هي:

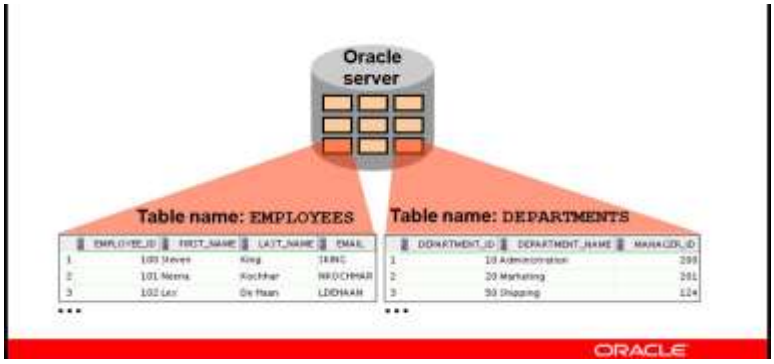
1- مجموعة من العناصر والعلاقات.

2- مجموعة من العمليات التي يتم تنفيذها على العلاقات.

3- كتلة من البيانات المضبوطة والمتكاملة.

كما من الممكن تعريفها على انها عبارة عن مجموعة من العلاقات والروابط التي تتم بين جدولين أوأكثر ، مثال ذلك لو كان لدينا جدولين الأول جدول خاص بالموظفين باسمEmployees والثاني خاص بالأقسام باسم Departments

نستطيع عندئذ بناء علاقة بين الجدولين باستخدام خادم اوراكل Oracle Server وكما هو موضح في الشكل التالي.



هنالك خمسة مراحل لدورة حياة تطوير أي نظام هي:

1- الإستراتيجية والتحليل.

2- التصميم.

3- البناء والتوثيق.

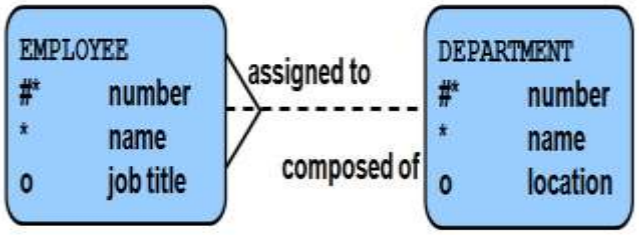
4- التحويل.

5- الحماية.

كيفية خلق علاقة بين جدولين.

أن عملية خلق اي علاقة بين جدولين تعتمد على طبيعة الحقول الموجودة في الجدولين والية العمل للنظام. على سبيل المثال لو كان لدينا جدولين الأول خاص بالموظفين ويدعى Employees يحتوي على ثلاثة حقول هي (Number, Name, Job-title) والجدول الثاني خاص بالأقسام ويدعىDepartments ويحتوي على ثلاثة حقول أيضا هم (NumberName, Location) في تلك الحالة نستطيع بناء علاقة بين

الجدولين من خلال ربط حقل أوأكثر من الجدول الأول مع حقل أوأكثر من الجدول الثاني شرط ان يكون وحيد Unique وكما هو موضح في الشكل التالي.

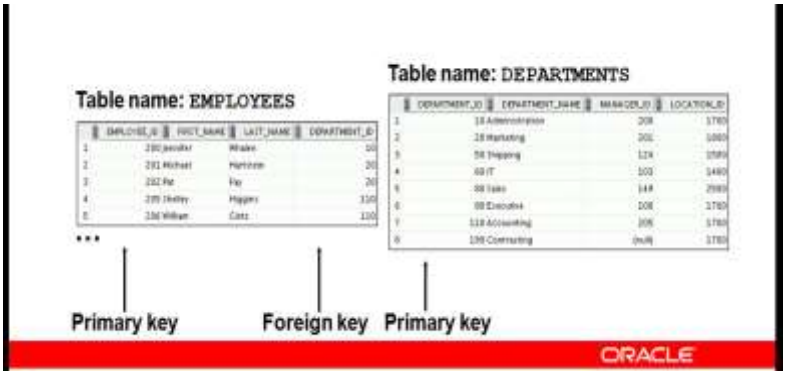


- المفتاح الأولي Primary Key .

هو عبارة عن حقل يحتوي على قيم فريدة لا تتكرر في بقية الحقول والقيود مثل رقم الموظف حيث ان لكل موظف رقم قيد خاص به لا يمكن ان يتكرر او يشترك مع موظف اخر.

- المفتاح الخارجي Foreign Key.

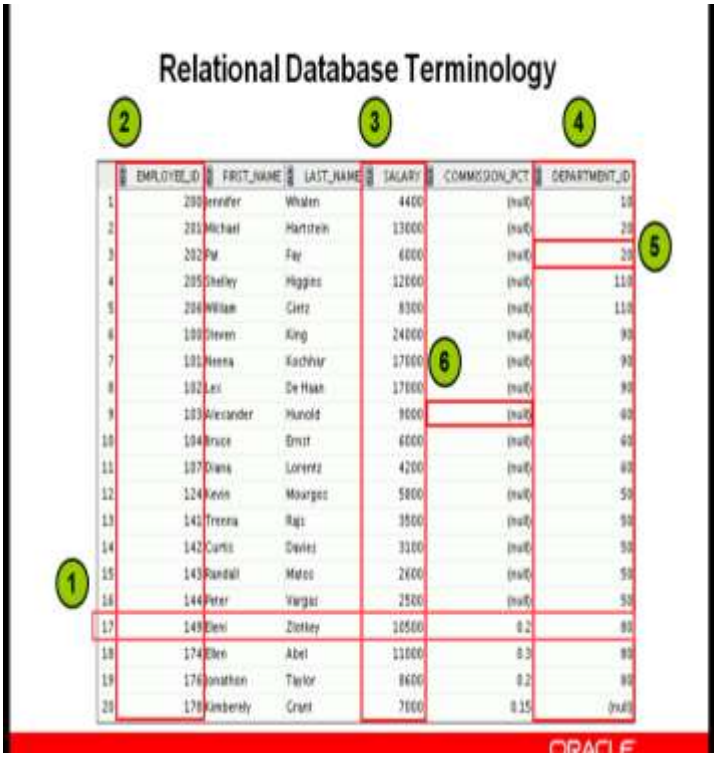
هو عبارة عن حقل استخدم في جدولين، ففي الجدول الاول عبارة عن Primary Key وفي الجدول الثاني عبارة عن Key اعتيادي وكما موضح في الشكل التالي.



أن كل سطر ببيانات في الجدول أعلاه يحتوي على Primary Key هو رقم الموظف Employee ID في جدول الموظفين ورقم القسم Department ID

في جدول الأقسام ، كما يحتوي على Foreign Key وهو الـ Department ID حيث استخدم كـ Primary Key في جدول الأقسام وكحقل اعتيادي في جدول الموظفين ، لذلك اعتبر كـ Foreign Key .

حيث من الممكن هنا ربط الجدولين باستخدام Department ID الموجود في كلا الجدولين، أي إن عملية الربط هنا تمت باستخدام الـ Foreign Key وبناء جدول واحد ، وكما هو موضح في الشكل أدناه.



Relational Database Terminology

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
1	200	Jennifer	Whalen	4400	(null)	10
2	201	Michael	Hartstein	13000	(null)	20
3	202	Pat	Fay	6000	(null)	20
4	205	Shelley	Higgins	12000	(null)	110
5	206	William	Gietz	8300	(null)	110
6	100	Steven	King	24000	(null)	90
7	101	Neena	Kochhar	17000	(null)	90
8	102	Lex	De Haan	17000	(null)	90
9	103	Alexander	Hunold	9000	(null)	60
10	104	Bruce	Ernst	6000	(null)	60
11	107	Diana	Lorentz	4200	(null)	60
12	124	Kevin	Mourgos	5800	(null)	50
13	141	Trenna	Rajs	3500	(null)	50
14	142	Curtis	Davies	3100	(null)	50
15	143	Randall	Matos	2600	(null)	50
16	144	Peter	Vargas	2500	(null)	50
17	149	Eleni	Zlotkey	10500	0.2	80
18	174	Ellen	Abel	11000	0.3	80
19	176	Jonathon	Taylor	8600	0.2	80
20	178	Kimberely	Grant	7000	0.15	(null)

خصائص قاعدة البيانات:

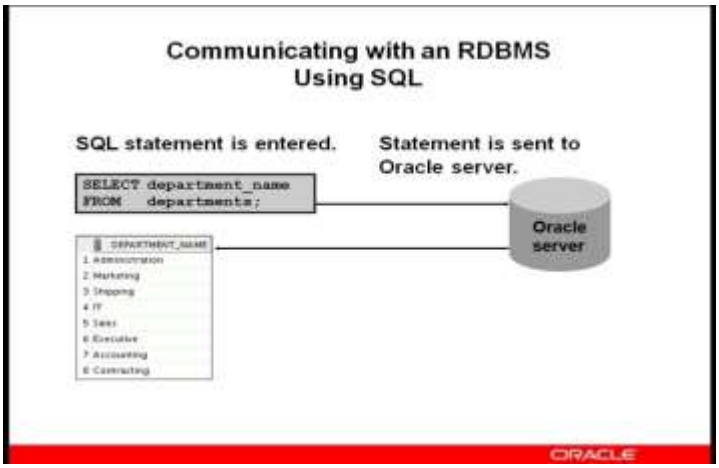
هنالك ثلاثة خصائص رئيسية لقاعدة البيانات هي:

1- من الممكن الوصول الى قاعدة البيانات وتحديثها بواسطة استخدام ايعازات لغة الاستعلام البنيوية (SQL) StructureQuery Language.

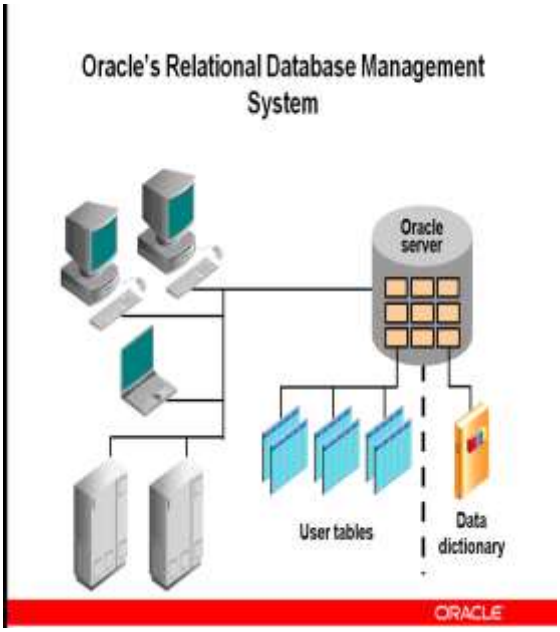
2- ان كل قاعدة بيانات تتكون من مجموعة من الجداول المرتبطة مع بعضها البعض.

3- تستخدم مجموعة من العمليات.

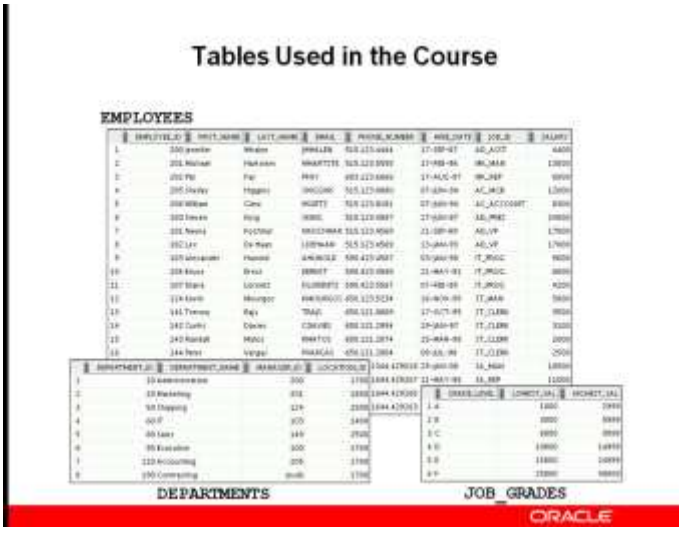
من الممكن استخدام جمل الـ (SQL) للاتصال بنظام قاعدة البيانات العلائقية RDBMS من خلال كتابة الجملة المطلوبة الخاصة بالعملية المراد تنفيذها ، مثال ذلك عندما نريد استرجاع مجموعة من القيود الخاصة بالاقسام نقوم بكتابة العبارة التالية والتي سوف يتم شرحها لاحقا. SELECTdepartment_name FROMdepartments;، حيث سوف تقوم هذه الجملة بالاتصال بخادم اوراكل Oracle Server الذي سيقوم بدوره بالاتصال بالجدول المطلوب في قاعدة البيانات واسترجاع البيانات المطلوبة وفق الشروط المحددة والشكل التالي يوضح ذلك.



أما في حالة وجود عدة مستخدمين يستخدمون عدة حاسبات وفي أماكن مختلفة فان جميع أولائك المستخدمين يتصلون بخادم أوراكل واحد والذي بدوره يتصل بقاعدة بيانات واحدة بحيث ان عمليات الإضافة والحذف والتعديل تتم على تلك القاعدة والشكل التالي يوضح ذلك.



أما بالنسبة إلى الجداول الموجودة في قاعدة البيانات فأنها ترتبط مع بعضها البعض عن طريق الـ Primary Key والـ Foreign Key وحسب ماتم شرحه سابقا والشكل التالي يوضح ذلك، حيث أن الجداول الثلاثة مرتبطة عن طريق الحقل الأول في تلك الجداول .



مستخدمو قواعد البيانات:

وينقسمون إلى ثلاث فئات هي:

أ- مخططو البرامج الذين يكتبون برامجهم و يستخدمون إمكانيات قاعدة البيانات.

ب- مختصو قواعد البيانات ، و هم المسئولون عن صيانة و تشغيل قاعدة البيانات ويدعى بـ (مدير قاعدة البيانات Database administrator).

ت- المستخدمون لقواعد البيانات الذين يتعاملون مع قاعدة البيانات عبر النهايات الطرفية ويدعى بالـ (مستخدم قاعدة البيانات DatabaseUser)

.

واجبات مدير قواعد البيانات (Database Administrator DBA). ان وظيفة مدير قواعد البيانات هو إنشاء و توسيع و صيانة قاعدة البيانات و يقوم بدور الوسيط بين البيانات و مستخدميها ويشرف على إدارتها، ويوزع مناطق التخزين على البيانات وينشأ لها الفهارس والمؤشرات اللازمة لاسترجاعها، كما يمكن تغيير الشكل البنائي لها ، كما يتيح عمليات إضافة أو حذف أو تعديل السجلات، كما يقوم بدور الوسيط بين البرامج وبين البيانات.إلى جانب ذلك كله يمكنه القيام بالأعمال التالية :

- إنشاء قاعدة بيانات جديدة وإدارة قواعد البيانات الحالية.
- معالجة السجل المطلوب بمفرده ضمن أي ملف.
- استرجاع السجلات سجلاً سجلاً.
- يقي البيانات من الدخول عليها من شخص غير ذي صلاحية.
- حماية البيانات ضد التخريب.

- وضع نقاط إرشادية تستخدم في حالة عطل الآلات أو عطل البرامج مما يساعد على سهولة استئناف العمل دون العودة إلى بداية الملف.
- رصد الحركة على البيانات إحصائياً.
- تسجيل كل تعامل على البيانات.
- وضع البيانات الهامة التي يشتد عليها الطلب في مواقع ذات أسبقيات.
- يحتفظ بقاموس بيانات شامل أي بيانات عن البيانات، والمستخدمين، والصلاحيات المتاحة لكل مستخدم.

واجبات مستخدم قواعد البيانات Database User:

ان وظيفة مستخدم قواعد البيانات هي الإشراف على قاعدة البيانات ، حيث ان هذه الوظيفة مهمة في مراكز المعلومات ويتولاها أفراد ذوي كفاءة و مقدرة إذ عليهم يتوقف ما يلي:

- ضبط البيانات المخزنة في قواعد البيانات بحيث تلبي حاجات مستخدمي المعلومات.
- يتولى الإشراف على إصدار التقارير المطلوبة في النظام.
- تحقيق أمن وسلامة البيانات و قواعد البيانات.
- الإشراف على إضافة بيانات جديدة وتحديث البيانات القديمة.
- التحقق من عدم تكرارية البيانات.
- التحقق من تكاملية البيانات.

مزايا قواعد البيانات في اتخاذ القرار.

تغطي قواعد البيانات عدة مزايا لمتخذ القرار منها:

- تقدم للإدارة تقارير مبنية على معلومات محدثة شاملة مما يساعد على اتخاذ قرارات سليمة عكس نظام الملفات المرتبط ببيانات محددة.
- تقدم للإدارة الوسطى تقارير مفصلة جيدة يصعب الحصول عليها من نظام الملفات.
- التوفير في التكلفة نتيجة عدم تكرارية البيانات.
- توفر الجهد المبذول في إدخال البيانات نتيجة توحيد المدخلات لكل نظام فرعي.
- البساطة الشديدة في استخدام لغة الاستفسار لأن مدير قاعدة البيانات يتولى مهام التعامل مع البيانات.
- الاستجابة السريعة لاحتياجات المستخدمين.
- الإقلال من عدد الأفراد العاملين في مراكز المعلومات.
- إدارة جيدة للبيانات حيث تحفظ البيانات في مكان مركزي موحد لكل المؤسسة أو المنظمة.
- الاسترجاع المتعدد MULTIPLE ACCESS باستخدام أساليب بسيطة نسبياً ومن خلال استخدام مفاتيح
- (حقول خاصة).

أمن المعلومات:

مع تطور التكنولوجيا ووسائل تخزين المعلومات وتبادلها بطرق مختلفة أو ما يسمى نقل البيانات عبر الشبكة من موقع لاخر أصبح النظر إلى أمن تلك البيانات والمعلومات بشكل مهم للغاية. يمكن تعريف أمن المعلومات بأنه العلم الذي يعمل على توفير الحماية للمعلومات من المخاطر التي تهددها أو الاعتداء عليها وذلك من خلال توفير الأدوات والوسائل اللازم توفيرها لحماية المعلومات من المخاطر الداخلية أو الخارجية. المعايير والإجراءات المتخذة لمنع وصول

المعلومات إلى أيدي أشخاص غير مخولين عبر الاتصالات ولضمان أصالة وصحة هذه الاتصالات.. إن حماية المعلومات هو أمر قديم ولكن بدأ استخدامه بشكل فعلي منذ بدايات التطور التكنولوجيا ويرتكز أمن المعلومات إلى:

- أنظمة حماية نظم التشغيل.
- أنظمة حماية البرامج والتطبيقات.
- أنظمة حماية قواعد البيانات.
- أنظمة حماية الولوج أو الدخول إلى الأنظمة.

المبادئ الأساسية لأمن المعلومات:

وهي من أهم المفاهيم المستخدمة قبل أكثر من عشرين عاما، وأمن المعلومات قد عقدت السرية والنزاهة في تبادل تلك المعلومات، وان امن المعلومات بني على مبدأين اساسيين هما:

1- السرية.

السرية هو المصطلح المستخدم لمنع الكشف عن معلومات لأشخاص غير مصرح لهم بالأطلاع عليها أو الكشف عنها. على سبيل المثال، بطاقة الائتمان والمعاملات التجارية على شبكة يتطلب رقم بطاقة الائتمان على أن تنتقل من المشتري إلى التاجر ومن التاجر لإنجاز وتجهيز المعاملات على الشبكة. يحاول النظام فرض السرية عن طريق تشفير رقم البطاقة أثناء الإرسال، وذلك بالحد من الأماكن ظهور تسلسل رقم البطاقة (في قواعد البيانات، وسجل الملفات، النسخ الاحتياطي، والإيصالات المطبوعة)، وذلك بتقييد الوصول إلى الأماكن التي يتم تخزين الرقم والبيانات بها. اماإذا كان الطرف غير المصرح به قد حصل على

رقم البطاقة بأي شكل من الأشكال ،وبذلك فقد تم انتهاك مبدأ السرية في حفظ وتخزين البيانات.

خرق السرية يتخذ أشكالا عديدة. تجسس شخص ما على شاشة الكمبيوتر لسرقة كلمات سر الدخول ،أو رؤية بيانات سرية لديك بدون علم منك يمكن أن يكون خرقا للسرية. إذا الكمبيوتر المحمول يحتوي على معلومات حساسة عن موظفي الشركة هو سرق أو بيع، يمكن أن يسفر عن انتهاك لمبدأ السرية. إعطاء معلومات سرية عبر الهاتف هو انتهاك لمبدأ السرية إذا كان الطالب غير مخول أن يحصل على المعلومات.

السرية أمر ضروري (لكنها غير كافية) للحفاظ على الخصوصية من الناس الذين يخترقون الأنظمة لسرقة المعلومات الشخصية في المخزنة في قاعدة البيانات.

2- السلامة.

في مجال أمن المعلومات، السلامة تعني الحفاظ على البيانات من التغيير والتعديل من الأشخاص الغير مخول لهم بذلك. عندما يقوم شخص بقصد أو بغير قصد بانتهاك سلامة أو الإضرار أو حذف ملفات البيانات الهامة وهو غير مخول بذلك فهذا انتهاك لسلامة البيانات، وعندما يصيب فيروس كمبيوتر ويقوم بتعديل بيانات أو اتلافها فهذا انتهاك سلامة بيانات، وعندما يكون الموظف قادرا على تعديل راتبه في قاعدة البيانات والمرتبات، وعندما يقوم مستخدم غير مصرح له بتخريب موقع على شبكة الإنترنت، وهلم جرا. و تعني سلامة البيانات كذلك، أن تكون التغيرات في البيانات مطردة، فعندما يقوم عميل البنك بسحب أو إيداع، ينبغي أن ينعكس ذلك على رصيده في البنك.

إن الإخلال بسلامة البيانات ليس بالضرورة نتيجة عمل تخريبي، فمثلاً الانقطاع في النظام قد ينشئ عنه تغيرات غير مقصودة أو لاتحفظ تغيرات قد تمت فعلاً.

3- إدارة المخاطر.

هي عملية التعرف على نقاط الضعف والتهديدات الموجهة إلى موارد المعلومات التي تستخدمها المنظمة أو الشبكة المعلوماتية في تحقيق الأهداف التجارية أو الاخرى، والحد والتقليل من نقاط الضعف إن وجدت، لتأخذ في الحد من المخاطر إلى مستوى مقبول، على أساس قيمة موارد المعلومات إلى المنظمة .

هناك أمران في هذا التعريف قد يحتاجان إلى بعض التوضيح. أولا، عملية إدارة المخاطر هي تكرار العمليات الجارية ويجب أن يتكرر إلى ما لا نهاية لان بيئة العمل المتغيرة باستمرار، والتهديدات الجديدة والضعف تظهر كل يوم .والثانية اختيار التدابير المضادة (الرقابة) المستخدمة لإدارة المخاطر يجب أن توازن بين الإنتاجية، والتكلفة، وفعالية التدابير المضادة، وقيمة الموجودات وحماية البيانات. الخطر هو احتمال أن شيئا ما سيئا سيحدث يسبب الأذى لأحد الأصول المعلوماتية (أو الخسارة في الأصول). الضعف هو الضعف الذي يمكن أن يستخدم لتعريضها للخطر أو التسبب في ضرر لأحد الأصول المعلوماتية. التهديد أي شيء فعل (من صنع الإنسان او فعل من أفعال الطبيعة) لديه القدرة على التسبب في ضرر. احتمال أن يشكل تهديدا سوف تستخدم من التعرض للضرر يتسبب في خطر. عندما لا يشكل تهديدا استخدام الضعف لإلحاق الاذى، لما له من أثر. في سياق أمن المعلومات، وأثر هو خسارة لتوافر والنزاهة والسرية، وربما غيرها من الخسائر) الدخل المفقود، والخسائر في الأرواح وخسائر في الممتلكات العقارية (وتجدر الإشارة إلى أنه ليس من الممكن تحديد

جميع المخاطر، ولا هو ممكن القضاء على جميع المخاطر. المخاطر المتبقية تسمى المخاطر المتبقيه.

تقييم المخاطر:

وتشمل ما يلي:

1. السياسة الأمنية.
2. تنظيم أمن المعلومات، وإدارة الوصول الى البيانات.
3. امن الموارد البشرية.
4. الأمن البيئي.
5. الاتصالات وإدارة العمليات، والتحكم في الوصول.
6. اقتناء نظم المعلومات وتطويرها وصيانتها أو ما يسمى بالتحديث.
7. إدارة استمرارية الأعمال.
8. التوافق التنظيمي.

تألف عملية إدارة المخاطر من ما يلي:

- ✓ تحديد الموجودات وتقدير قيمتها. وتشمل الأفراد والمباني والأجهزة والبرامج والبيانات (الإلكترونية والمطبوعة وغيرها)، واللوازم.
- ✓ إجراء تقييم التهديد. وتشمل: أفعال الطبيعة، أعمال الحرب والحوادث والأفعال الضارة القادمة من داخل أو خارج المنظمة.
- ✓ إجراء تقييم الضعف، ولكل الضعف، وحساب احتمال أن يكون للاستغلال. تقييم السياسات والإجراءات والمعايير، والتدريب، الأمن المادي، مراقبة الجودة والأمن التقني.

- ✓ في حسبانها تأثير كل ذلك من شأنه أن يكون خطرا على كل الموجودات. استخدام التحليل النوعي أو التحليل الكمي.
- ✓ تحديد واختيار وتطبيق الضوابط المناسبة. تقدم ردا متناسبا. النظر في الإنتاجية، وفعالية التكاليف، وقيمة الموجودات.
- ✓ تقييم فعالية تدابير المكافحة. ضمان توفير الضوابط اللازمة لحماية فعالة من حيث التكلفة دون فقدان ملحوظ في الإنتاجية. عند أي خطر معين، يمكن أن تختار الإدارة التنفيذية قبول المخاطرة استنادا إلى انخفاض القيمة النسبية للموجودات، وتواتر حدوث منخفضة نسبيا، وأثر انخفاض نسبي على الأعمال التجارية. أو، قد تختار القيادة التخفيف من المخاطر من خلال تحديد وتنفيذ تدابير الرقابة المناسبة للحد من المخاطر. في بعض الحالات، يمكن أن يكون خطر نقل إلى آخر أعمال التأمين عن طريق شراء أو التسنيد إلى آخر الأعمال. قد واقع بعض المخاطر يمكن الجدال فيها. في مثل هذه الحالات قد تختار القيادة إنكار المخاطر. هذا هو في حد ذاته يشكل خطرا محتملا.

اعتماد وتدقيق أمن المعلومات:

أصبحت النظم المعلوماتية وقواعد البيانات وشبكات الاتصال عصب العالم المعرفي والصناعي والمالي والصحي وغيرها من القطاعات. حيث أصبح من المهم الحفاظ على أمن المعلومات بعناصره الرئيسية الثلاث: السرية والصوابية والاستمرارية. وعلى المستوى العالمي يبرز نظام الأيزوللاعتماد والتقييم والتقييس 27001 لضمان أمن المعلومات. كما يوجد نظام HIPAA في الولايات المتحدة

الأمريكية لضمان أمن المعلومات الصحية ونظام COBIT من ISACA لأمن المعلومات.

ضوابط أمن المعلومات:

عندما تختار الإدارة للتخفيف من المخاطر، فإنها تفعل ذلك من خلال تنفيذ واحد أو أكثر من ثلاثة أنواع مختلفة من الضوابط هي.

أ- الضوابط الإدارية.

ان الرقابة الإدارية (وتسمى أيضا الضوابط الإجرائية) تتألف من الموافقة الخطية والسياسات والإجراءات والمعايير والمبادئ التوجيهية. بحيث ان الرقابة الإدارية تشكل إطارا لإدارة الأعمال التجارية وإدارة الأفراد.

إنها إطلاع الناس على كيفية عمل ما وهو كيفية تشغيل العمليات اليومية وكيف يجب أن تجرى. كما ان القوانين واللوائح التي أنشأتها الهيئات الحكومية هي أيضا نوع من الرقابة الإدارية. وان معيار أمن البيانات المطلوبة مثل تأشيرة الدخول وماستر كارد هو مثال على ذلك. ومن الأمثلة الأخرى على الضوابط الإدارية وتشمل الشركات السياسة الأمنية، سياسة كلمة السر، سياسات التوظيف، والسياسات التأديبية. والرقابة الإدارية تشكل أساسا للاختيار وتنفيذ الضوابط المنطقية والفيزيائية.

ب-الضوابط المنطقية.

الضوابط المنطقية (وتسمى أيضا الضوابط التقنية) وتشمل استخدام البرمجيات والبيانات لرصد ومراقبة الوصول إلى نظم المعلومات والحوسبة.

على سبيل المثال: كلمات السر، والجدران النارية، وكشف التسلل، قوائم التحكم بالولوج، وتشفير البيانات والضوابط المنطقية.

ت-الضوابط المادية

الضوابط المادية تشمل رصد ومراقبة البيئة في مكان العمل ومرافق الحوسبة. كما رصد ومراقبة الدخول والخروج من هذه المرافق. على سبيل المثال الأبواب والأقفال، والتدفئة وتكييف الهواء والدخان وأجهزة إنذار الحريق ونظم إخماد الحريق، والكاميرات، ووضع المتاريس، والمبارزة، وحراس الأمن، وتأمين الكابلات، وما إلى ذلك فصل الشبكة، ومكان العمل في مجالات وظيفية هي أيضا من الضوابط المادية.

مهددات أمن المعلومات:

هنالك ستة انواع اساسية من التهديدات الخاصة بأمن المعلومات، وتشمل:

1- الفيروسات:

الفيروس هو برنامج صغير مكتوب بأحد لغات الحاسوب ويقوم بإحداث أضرار في الحاسوب والمعلومات الموجودة عليه ، بمعني انه يتركز على ثلاث خواص وهي التخفي، التضاعف، وإلحاق الأذى. و يكمن مصادر الفيروس من خلال الرسائل الإلكترونية المجهولة، صفحات الإنترنت المشبوهة، نسخ البرامج المقلدة، استخدام برامج غير موثقة، كذلك تبادل وسائل التخزين دون عمل فحص مسبق مثل الأقراص والذاكرة المتنقلة وارسال الملفات داخل الشبكة المحلية .للفيروس ثلاث خواص مؤثرة وهي:

التضاعف: تتم عملية تضاعف الفيروس عند التحاق الفيروس بأحد الملفات وهنا تتم عملية زيادة عدد العمليات التي تتم إلى ملايين العمليات مما يسبب البطء في العمل أو توقف الحاسب عن العمل.

التخفي :لابد للفيروس من التخفي حتى لا ينكشف ويصبح غير فعال، ولكي يتخفي فأنة يقوم بعدة أساليب منها علي سبيل المثال، صغر حجم الفيروس لكي سيناله الاختباء بنجاح في الذاكرة أو ملف آخر.

إلحاق الأذى: قد يتراوح الأذى الذي يسببه الفيروس بالاكتفاء بإصدار صوت موسيقي أو مسح جميع المعلومات المخزنة لديك، ومن الأمثلة الاخري في إلحاق الأذى: إلغاء بعض ملفات النظام،إغلاق الحاسب من تلقاء نفسه عند الدخول على الإنترنت مثلا أو إلغاء البرنامج المكتوب على BIOS .

1- هجوم تعطيل الخدمه:

وهذا النوع من الخدمة يقوم فيه القرصان أو المعتدي بإجراء أعمال خاصة تؤدي إلى تعطيل الأجهزة التي تقدم الخدمة Server في الشبكات.

2- مهاجمه المعلومات المرسله:

وهو اعتراض المعلومات عند ارسالها من جهة إلى أخرى، ويحدث هذا التعامل غالباً أثناء تبادل الرسائل خلال الشبكات، الإنترنت ، الشبكات التي تستخدم شبكة الهاتف العامة.

3- هجوم السيطره الكامله:

في هذا النوع يقوم القرصان بالسيطرة الكاملة على جهاز الضحية والتحكم في جميع ملفاته كما لو كانت في جهازه هو ويمكن للقرصان مراقبة الضحية بصورة كاملة. يتم الهجوم بعد أن يضع القرصان ملف صغير على جهاز الضحية (عن طريق البريد الإلكتروني أو أي وسيلة أخرى) أو عن طريق استغلال نقاط الضعف في أنظمة التشغيل.

4- هجوم التضليل:

وفيه يقوم القرصان بانتحال شخصية موقع عام. كما يمكن للقرصان أن ينتحل شخصية مستخدم موثوق به للحصول على معلومات غير مصرحة له.

5- الوصول المباشر لكوابل التوصيل:

يقوم المهاجم بالوصول المباشر لأسلاك التوصيل والتجسس على المعلومات المارة. ولكنه هجوم صعب ويتطلب عمليات خاصة.

طرق وأدوات لحمايه امن المعلومات.

أ- التأمين المادي للأجهزة والمعدات.

ب- تركيب مضاد فيروسات قوي وتحديثه بشكل دوري.

ج- تركيب أنظمة كشف الاختراق وتحديثها.

د- تركيب أنظمة مراقبة الشبكة للتنبيه عن نقاط الضعف التأمينية.

ذ- عمل سياسة للنسخ الاحتياطي.

هـ - استخدام أنظمة قوية لتشفير المعلومات المرسلة.

و- دعم أجهزة عدم انقطاع التيار الكهربائي.

ي- نشر التعليم والوعي الأمني.

الفصل الثاني

لغة الاستعلام البنيوية

SQL Server

ملاحظات هامة عن جمل لغة الاستعلامات البنيوية SQL:

قبل الدخول في تفاصيل جمل لغة الاستعلامات البنوية، يجب عليك ملاحظة ما يلي.

1- ليس هنالك فرق بين الحروف الصغيرة والحروف الكبيرة عند كتابة جمل الـ SQL .

2- من الممكن أن تكتب جمل الـ SQL على سطر واحد أو أكثر.

3- أن الكلمات المحجوزة في لغة الـ SQL لا يمكن ان تكتب على أكثر من سطر واحد.

4- هنالك نوعان من لغة الـ SQL الأول يدعى بالـ SQL *Plus والثاني يدعى بالـ SQL Developer كل سطر في لغة الـ SQL Developer ينتهي بفارزة منقوطة (;) وان عملية وضع تلك الفارزة تكون بصورة اختيارية ، فإذا كنا نقوم بتنفيذ جملة واحدة فقط يجب في تلك الحالة وضع الفارزة المنقوطة ، أما أذا كنا نقوم بتنفيذ أكثر من جملة واحدة فهنالك ضرورة لوضع الفارزة المنقوطة لتعليم كل سطر أو بالأحرى بعد نهاية كل أمر. أما في الـ SQL *Plus فيجب ان ينتهي كل أمر بفارزة منقوطة (;) .

ايعازات لغة الاستعلام البنيوية SQL Statements .

هنالك خمسة أنواع من المجاميع الخاصة بلغة الـ SQL هي.

1- لغة تعريف البيانات Data Definition Language (DDL) : وتشمل ستة ايعازات أساسية هي:

أ- CREATE.

ب- ALTER.

ت- DROP.

ث- RENAME.

ج- TRUNCATE.

ح- COMMENT.

2- لغة معالجة البيانات (DML) Data Manipulation Language:

وتشمل أربعة ايعازات أساسية هي.

أ- INSERT.

ب- UPDATE.

ت- DELETE.

ث- MERGE.

3- لغة استعادة البيانات: Data Retrieval Language وتشمل أيعاز واحد فقط هو:

أ- Select.

4- ايعازات السيطرة على العمليات (TC) Transaction Control: وتشمل ثلاثة ايعازات أساسية هي.

أ- COMMIT.

ب- ROLLBACK.

ت- SAVEPOINT.

5- لغة السيطرة على البيانات (DCL) Data Control Language:

وتشمل ايعازين أساسيين هما.

أ- GRANT.

ب-REVOKE.

هنالك خمسة أنواع من عناصر قاعدة البيانات هي.

1- **Table:** يعتبر الوحدة الأساسية لخزن البيانات ويتكون من مجموعة من الأسطر والأعمدة .

2- **View:** يعتبر التمثيل المنطقي للمجاميع الجزئية للبيانات ويتكون من واحد أوأكثر من الجداول.

3- **Sequence:** ويتكون من مجموعة من القيم العددية.

4- **Index:** يستخدم لتحسين الأداء لبعض الـ queries.

5- **Synonym:** يقوم بإعادة تسمية العناصر.

قواعد التسمية الخاصة بالعناصر (الجداول والأعمدة) في لغة أوراكل:

1- يجب أن يبدأ الاسم بحرف.

2- يجب أن يكون طوله من (1 – 30) كحد أقصى.

3- يجب أن يحتوي فقط على الأحرف الكبيرة (A-Z) ، الأحرف الصغيرة (a-z) ، الأرقام (9 – 0) ، الرموز الخاصة فقط (_ , $, #) .

4- يجب أن لايكون هنالك تكرار لأسماء العناصر الموجودة.

5- يجب أن لايكون الاسم من الأسماء الخاصة والمحجوزة للغة أوراكل Reserved Word.

1-لغة تعريف البيانات Data Definition Language (DDL) .

وتشمل ستة ايعازات رئيسية ، وهذه الايعازات هي:

أ- **أيعاز CREATE** :

يستخدم هذا الإيعاز لخلق عنصر من العناصر الخمسة التابعة لقاعدة البيانات، ومن أشهر استخداماته هي عملية خلق وتكوين الجداول CREATE Tables. فعند خلق جدول معين يجب توفر شرطين أساسيين هما:

- يجب أن يملك المستخدم الصلاحية لخلق جدول جديد، حيث أن كل مستخدم يمتلك صلاحيات محددة.
- يجب أن يملك المستخدم مساحة كافية على القرص الصلب.

أن الصيغة العامة لإيعاز CREATE TABLE هي:

SQL> **CREATE TABLE table_name**

Column1_name column 1_type (size)

Column2_name column 2_type (size)

Column n_name columnn n_type (size)

عند خلق جدول جديد يجب تحديد ثلاثة أشياء رئيسية هي:

أ- اسم الجدول Table Name.

ب-اسم العمود Column Name.

ت- نوع البيانات الخاصة بالعمود Column Type.

ث-حجم العمود Column Size.

Example:

SQL> CREATE TABLE student

Id number(4),

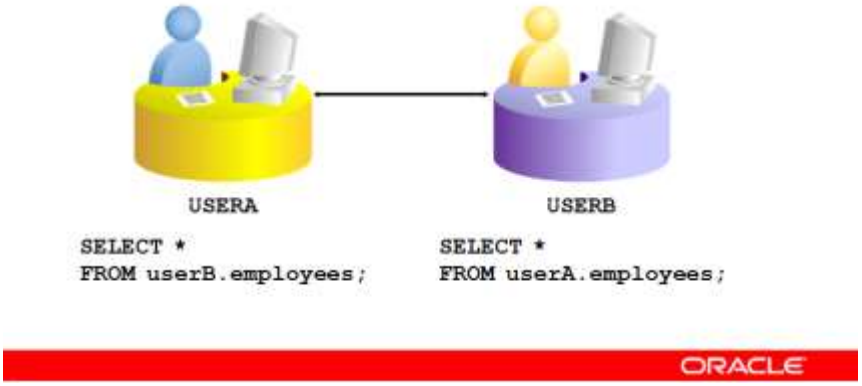
Name varchar1(30),

Age number(3)

When you press at Enter key the result is

Table Created Succefully.

أذا كان هنالك جدول ينتمي إلى مستخدم معين يدعى user B ويريد مستخدم أخر يسمى User A استخدامه فان ذلك الجدول لا يعتبر تابع إلى الوحدات الخاصة بـ user A ، ولغرض الوصول إليه يجب في تلك الحالة ذكر اسم المستخدم المالك لهذا الجدول قبل اسم الجدول لغرض الوصول إليه والشكل التالي يوضح ذلك.



من الممكن جعل القيم الخاصة باسم العمود وحجمه ونوعه تلقائية وتتحدد من خلال الإدخال وذلك من خلال استخدام الإيعاز التالي.

SQL> CREATE TABLE student (

Id number(4),

Student DATA DEFAULT SYSDATE

حيث أن الإيعازأعلاه يحدد ما يلي:

أ- القيم الحرفية والتعابير ودوال الـ SQL هي قيم قانونية.

ب- أن الأسماءالأخرى للأعمدة والأعمدة الوهمية هي قيم غير قانونية.

ت- أن نوعية البيانات التلقائية يجب أن تتطابق مع نوعية بيانات العمود.

Example2:

SQL> CREATE TABLE dept Depno number(3),

Depname varchar2(30),

Loc varchar2(13),

Create-data DATA DEFAULT SYSDATE

Press Enter Key.

Create Table Successes.

لغرض التأكد من أن الجدول تم خلقه أم لا نستطيع استخدام الإيعاز التالي:

SQL> DESCRIBE dept;

```
Name                           Null     Type
------------------------------ -------- -------------
DEPTNO                                  NUMBER(2)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
CREATE_DATE                             DATE

4 rows selected
```

أنواع البيانات في لغة أوراكل:

أ- VARCHAR2 (SIZE): يستخدم لتعريف البيانات الحرفية ذات الأطوال المختلفة.

ب-VARCHAR (SIZE): يستخدم لتعريف البيانات الحرفية ذات الأطوال الثابتة.

ت-NUMBER (P, S) : يستخدم لتعريف القيم العددية الصحيحة والكسرية بحيث أن الـ P تمثل عدد المراتب الصحيحة والـ S تمثل عدد المراتب الكسرية.

ث-DATE: تستخدم لتعريف قيم التاريخ والوقت.

ج- LONG: تستخدم لتعريف البيانات الحرفية المتغيرة الطول والتي يزيد حجمها عن (2 GB).

ح- CLOB: تستخدم لتعريف البيانات الحرفية التي يزيد حجمها عن(4 GB)

خ- RAW and LONG RAW: تستخدم لتعريف بيانات الأسطر الثنائية.

د- BLOB: يستخدم لتعريف البيانات الثنائية التي يزيد حجمها عن (4GB)

ذ- BFILE: يستخدم لتعريف البيانات الثنائية المخزنة في ملفات خارجية يزيد حجمها عن (4 GB).

ر- ROW ID: يستخدم لتمثيل الأنظمة العددية ذات الأساس (64) والتي تمتلك عنوان وحيد للسطر في جداولها.

ز- TIMESTAMP: يستخدم هذا النوع لتمثيل البيانات الخاصة بالوقت والتاريخ ، حيث يستخدم هذا النوع لخزن السنة، الشهر، اليوم للنوع DATEبالإضافة إلى الساعة، الدقيقة، الثانية، أجزاء الثانية. كما يمكننا من خلاله تحديد الوقت المحلي TIME ZONEللمنطقة الحالية، مثال ذلك.

Example1:

SQL> TIMESTAMP [(Fractional_Seconds_Precision)];

Example2:

SQL> TIMESTAMP [(Fractional_Seconds_Precision)] with TIME ZONE;

Example3:

SQL> TIMESTAMP [(Fractional_Seconds_Precision)] with LOCAL TIME ZONE;

س- INTERVAL YEAR TO MONTH:

يستخدم لخزن البيانات التي تحتوي على فترة زمنية (سنة، شهر) مثال ذلك.

Example:

SQL> **INTERVAL YEAR [(year_precision)] TO MONTH**;

ش- INTTERVAL DAY TO SECOND : يستخدم لتمثيل الفقرات الخاصة بالأيام، الساعات، الدقائق، الثواني مثال ذلك.

SQL> **INTERVAL DAY [(day_precision)]**

SQL> **TO SECOND [(fractional_seconds_precision)]**;

تقييد الإدخال Constraint:

هو عبارة عن مجموعة من القواعد والشروط التي تحدد عملية إدخال البيانات في الجداول، حيث انه يكون في تماس مع الأعمدة (يحدد دائما عند تعريف

الأعمدة). ومن الممكن وضع شروط معينة يوفرها برنامج الاوراكل عند إدخال البيانات أو عند توليد الاسم بواسطة الـ Oracle Server بصورة أوتوماتيكية ، حيث أن هنالك طريقتين لإنشاء التقييد الخاص بإدخال البيانات هما:

أ- عند تعريف الجدول At Table level.

ب- عند تعريف العمود At column level.

ت- بواسطة استخدام أمر التعديل Alter.

حيث من الممكن تعريف الـ Constraint عند خلق الجدول او العمود ، كما من الممكن رؤية نوعية التقييد الذي تم بنائه من خلال الـ data dictionary.

هنالك خمسة أنواع من الـ :Constraint:

1-Not Null :

يستخدم هذا النوع لمنع حالة الـ Null للحقل، أي انه يجب دائما أن يحتوي على بيانات ولا يكون فارغ، وان هذا النوع من الـ Constraint نستطيع تعريفه فقط عند تعريف العمود ولا يمكن تعريفه في الحالتين الاخريتين مثال ذلك.

- At Column Level:

```
SQL> CREATE TABLE student
No number(3) NOT NULL,
Name varchar2(30) NOT NULL
```

2-CHECK:

يستخدم هذا النوع لتحديد إدخال القيم إلى الحقول تحت شروط معينة، ونستطيع تعريف هذا النوع بالطرق الثلاثة المذكورة سابقا والأمثلة التالية توضح ذلك.

- At Column Level:

```
SQL> CREATE TABLE student
No number(3),
Name varchar2(30),
Mark number(3) check (mark >=0 (and mark<=100)
```

أن هذا الشرط وضع لتقليل أخطاءالإدخال وإجبار المستخدم أن يدخل درجة الامتحان بقيمة سالبة أو اكبر من (100).

- At Table Level:

```
SQL> CREATE TABLE student
  No number(3),
  Name varchar2(30),
  Mark number(3),
  Constraint check(no>=0)
```

هنا بعد تعريف الحقل (no) قمنا بذكر العبارة الأخيرة Constraintلإجبار المستخدم على إدخال قيم التسلسل الخاصة بالطالب اكبر من صفر(قيم غير سالبة).

- **By using Alter command:**

بعد أنشاء الجدول وتذكر المبرمج انه يجب عليه تحديد نوعية الإدخال لحقل معين في تلك الحالة نستطيع استخدام الأمر Alter لتعديل الجدول وتحديد الـ Constraint الجديد وكما موضح في المثال أدناه.

```
SQL> ALTER TABLE student add check (name>='a' and
name<='z');
```

3-UNIQUE:

يستخدم هذا الـ Constraint لمنع تكرار قيمة معينة في العمود ، حيث انه جميع قيم الحقول تكون وحيدة وغير مكررة في الحقول الأخرى ولكنها تسمح في نفس الوقت بوجود الـ NULL .كما من الممكن تعريف هذا النوع في جميع المستويات وكما موضح في الأمثلة التالية.

- **At Column Level.**

```
SQL> CREATE TABLE student
No number(3)unique,
Name varchar2(30),
Mark number(3)
```

- **At Table Level:**

```
SQL> CREATE TABLE student
No number(3),
Name varchar2(30),
Mark number(3),
Unique (no)
```

By using Alter Command:

SQL> ALTER TABLE student add unique(no);

4-Primary Key.

يستخدم هذا الـ Constraint لمنع حالة التكرار في قيم الحقول لعمود معين ومنع حالة الـ NULL في نفس الوقت، أي انه يعمل عمل الـ UNIQUE والـ NOT NULL معا، كما نستطيع إضافة ذلك الـ Constraint في المستويات الثلاثة والأمثلة التالية توضح ذلك.

- **AT Column Level:**

SQL> CREATE TABLE student

No number(3) primary key,

Name varchar2(30) primary key,

Mark number(3) check(mark >=0 and mark<=100)

At Table Level:

SQL> CREATE TABLE student

No number(3),

Name varchar2(30),

Mark number(3),

Primary key (no)

- **By using Alter command:**

SQL>alter table student add primary key(no);

5-Foreign key.

يستخدم هذا النوع للإشارة إلى الجدول الأب Parent table حيث يتم ذكره دائما في الـ child table . حيث أن هذا الـ key يعمل كـ primary key في الـ parent table وكـ key اعتيادي في الـ child table ومن الممكن إضافة هذا الـ constraint في table level والـ Alter command وكما موضح في الأمثلة التالية:

- **At Table Level:**

```
SQL> CREATE TABLE employee
Empno no(3),
Empname varchar2(30),
Deptno number(2),
Primary key (empno),
Foreign key (deptno)references dept(deptno)
```

- **At Alter Command:**

```
SQL> ALTER TABLE employee add foreign key (deptno)
references dept (deptno);
```

ملاحظة:

عند تكوين علاقة بين الـ primary key والـ foreign key فانك لن تستطيع حذف أي قيد من الـ parent table إذا كان هنالك قيد في الـ child table المرتبط به موجود. لكن هنالك أمر يدعى بالـ ON DELETE CASCADE

عند استخدامه نستطيع حذف الـ parent record حتى لو كان الـ child record الخاص به موجود ، وذلك لأنه عند استخدام هذا الأمر وقمت بحذف الـ parent record فان الاوراكل يقوم أوتوماتيكيا بحذف كل القيود المتعلقة به في الـ child table والأمثلة التالية توضح ذلك.

- **At Table Level.**

SQL> CREATE TABLE employee

Empno number(3),

Empname varchar2(30),

Deptno number(3),

Primary key (empno),

Foreign key (deptno) references dept (deptno) ON DELETE CASCADE

- **At Alter Command:**

SQL> ALTER TABLE employee add foreign key (deptno) references dept (deptno) ON DELETE CASCADE;

6-Composite Key:

نستطيع من خلال هذا النوع من تعريف مزيج من الأعمدة ، حيث نستطيع تعريف الـ composite key في الكيان الواحد والكيانات المرجعية المقيدة كما نستطيع تعريف هذا النوع في كل من الـ Table Level والـ Alter Command.

هنالك أربعة ايعازات مهمة تستخدم مع الـ Constraints هي:

أ- Enable: تستخدم هذه العملية لتفعيل الـ Constraint مثال ذلك.

SQL> ALTER TABLE employee ENABLE CONSTRAINT;

ب-Disable: تستخدم هذه العملية لإلغاء تفعيل الـ Constraint مثال ذلك.

SQL> ALTER TABLE employee DISABLE CONSTRAINT;

ث-Enforce: تستخدم هذه العملية لتأكيد تفعيل الـ Constraintأكثر من عملية الـ Enable لإجراء العمليات المستقبلية مثل عمليات إدخال البيانات وتحديثها.

ج- Drop: تستخدم هذه العملية لحذف الـ Constraint الذي تم عمله سابقا مثال ذلك.

SQL> ALTER TABLE employee DROP CONSTRAINT;

ب-أيعاز ALTER TABLE.

يستخدم هذا الإيعازلإضافة عمود جديد إلى الجدول أو حذف عمود موجود أصلا أوإجراء عملية التعديل لنوعية البيانات الخاصة بعمود معين وإعادة تسمية عمود والأمثلة التالية توضح ذلك.

- **Add new column.**

SQL> ALTER TABLE employee add (empage number(3));

- **Delete exist column.**

SQL> ALTER TABLE employee drop column(empage);

- **Modify exist column.**

SQL> ALTER TABLE employee modify (empage number(4));

- **Rename Exist column.**

SQL> ALTER TABLE employee rename column empage to age;

ملاحظة : هنالك أيعاز اسمه SET UNUSED يستخدم لتعليم واحد أوأكثر من الأعمدة كأعمدة غير مستخدمة والصيغة العامة لهذا الإيعاز هي.

SQL>ALTERTABLEtable_nameSETUNUSED (column_name);

Example:

SQL> ALTER TABLE emp SET UNUSED (job_id);

ت-أيعاز TRUNCATE.

يستخدم هذا الإيعاز لحذف البيانات الموجودة في جدول معين مع الاحتفاظ بالهيكلية الخاصة بهذا الجدول من أعمدة وسطور في قاعدة البيانات الخاصة به، أي انه لا يقوم بحذف الجدول وإنما يقوم فقط بمسح محتوياته ، وبعد استخدام هذا الإيعاز نستطيع إدخال بيانات جديدة إلى الجدول. وان الصيغة العامة لهذا الإيعاز هي:

SQL> TRUNNCATE TABLE table_name;

Example: SQL> TRUNCATE TABLE employee;

ث- أيعاز DROP TABLE.

أن وظيفة هذا الإيعاز هو حذف جدول معين من قاعدة البيانات، بحيث أن البيانات التي يحتويها ذلك الجدول إضافة إلى الهيكلية الخاصة به سوف يتم حذفها من قاعدة البيانات وان الصيغة العامة لهذا الإيعاز هي:

SQL> DROP TABLE table_name;

Example: SQL> DROP TABLE employee;

ج- أيعاز RENAME.

يستخدم هذا الإيعاز لتغيير تسمية جدول معين أو تغيير تسمية قاعدة البيانات بصورة عامة، وان الصيغة العامة لهذا الإيعاز هي:

SQL> RENAME table _ old name TO table _ new name;

Example: SQL> RENAME employee TO emp;

ح- أيعاز COMMENT.

يستخدم هذا الإيعازلإضافة ملاحظات معينة إلى قاعدة البيانات .

2- لغة معالجة البيانات (Data Manipulation Language(DML.

تحتوي هذه المجموعة على أربعة ايعازات مهمة تستخدم لإدارة البيانات من داخل العناصر المعرفة (الجداول وغيرها) ، آما الايعازات التي تندرج ضمن هذه المجموعة فهي:

أ- أيعاز INSERT.

يستخدم هذا الإيعازلإدخال البيانات إلى الجداول، والصيغة العامة لهذا الإيعاز هي:

SQL>INSERTINTOtable_name

(column1,column2,.....,columnn)

VALUES (value1,value2,……,valuen);

باستخدام هذه الصيغة سوف يتم إدخال سطر واحد فقط في المرة الواحدة،كما أن تسلسل القيم يجب أن يكون مطابق لتسلسل الحقول في الجدول ونوعيتها إضافة إلى أن البيانات المدخلة من نوع (نصوص) Characters وتاريخ Date يجب أن توضع بين Single quotation ، وهنالك أربعة أنواع من عمليات الإدخال هي:

1- الإدخال بواسطة القيم مثال ذلك.

```
SQL> CREATE TABLE employee
No number(3),
Name varchar2(30),
Age number(3)
SQL> INSERT INTO employee VALUES (1,'ali',30);
SQL> INSERT INTO employee VALUES (2,'ahmed',25);
SQL> INSERT INTO employee VALUES (3,'hussain',35);
```

2- الإدخال باستخدام العناوين مثال ذلك.

```
SQL> INSERT INTO employee VALUES (&no,&name,&age);
```

3- إدخال البيانات لأعمدة محددة باستخدام القيم مثال ذلك.

```
SQL> INSERT INTO employee (no, name) VALUES (4,'sara'); SQL> INSERT INTO employee (no, age) VALUES (5,35);
```

4- إدخال البيانات لأعمدة محددة باستخدام العناوين مثال ذلك.

SQL> INSERT INTO employee (no, name) VALUES (&no, &name);

ملاحظة:

من الممكن استخدام أيعاز Insert لإدخال بيانات خاصة لأعمدة مثل استخدام الأمر SYSDATE الذي يقوم بتحضير التاريخ والوقت الحالي بصورة أوتوماتيكية ، مثال ذلك لو قمنا بتعديل الجدول employee من خلال إضافة عمود جديد اسمه current _ date من خلال كتابة الإيعاز التالي:

SQL> ALTER TABLE employee ADD (current _ date date);

بعد ذلك نستطيع استخدام أيعاز Insert في إدخال بيانات التاريخ بصورة أوتوماتيكية من خلال الإيعاز التالي.

SQL> INSERT INTO employee (no , name , age , current _ date) values (7 , 'rasha' , 32 , SYSDATE);

حيث سوف يتم هنا إدخال رقم الموظف واسمه وعمره التي تم إعطائها من قبل المبرمج إضافة إلإدخال التاريخ والوقت الحالي بصورة أوتوماتيكية من خلال الإيعاز SYSDATE.

ملاحظة.

من الممكن استخدام أيعاز Insert في إدخال الوقت من خلال وضع الوقت داخل Single quotation والمثال التالي يوضح ذلك.

SQL> INSERT INTO employee (no , name , age , current _ date) values (8 , ' maha' , 31 , '03-02-1999');

من الممكن استخدام إيعاز Insertلإدخال البيانات الموجودة في جدول ثاني إلى الجدول الذي تم خلقه حديثا وكما موضح في المثال التالي:

A. Create first table named (st1).

```
SQL> CREATE TABLE st1
No1 number(3),
Name1 varchar1(30),
Age1 number(3),
Address1 varchar2(30)
```

Create second table named (st2).

```
SQL> CREATE TABLE st2
No2 number(3),
Name2 varchar2(30)
```

Now we can select some fields from the first table and insert it to the second table throw the following statement:

```
SQL> INSERT INTO st2 (no2, name2)
SELECT no1, name1 from st1
WHERE (age<=30);
```

ملاحظة : عند استخدام الطريقة أعلاه يجب الانتباه إلى عدم استخدام كلمة Value مع أيعاز Insert بالإضافة إلى يجب تطابق أنواع الـ Columns وأحجامها وأعدادها المنقولة من الجدول الأول إلى الجدول الثاني.

ب-أيعاز Update.

يستخدم هذا الإيعاز لتحديث البيانات الموجودة في الجدول، حيث يقوم هذا الإيعاز بتحديث أكثر من سطر واحد في نفس الوقت وحسب الشرط الموجود، والصيغة العامة لهذا الإيعاز هي:

SQL> UPDATE *table _ name* SET (*column1 = value1*, *column2= value2… column=valuen*]

WHERE *(specific condition*);

Example: SQL> UPDATE st1 SET age=35 WHERE (age>=30);

ملاحظة.

من الممكن استخدام إيعاز Update بدون استخدام الـ Where ففي تلك الحالة سوف يتم تحديث القيود جميعها وبدون أي شرط، وكما موضح في المثال أدناه:

SQL> UPDATE st1 SET age=20;

ملاحظة.

من الممكن تحديث أكثر من عمود واحد في نفس الوقت وباستخدام شرط واحد وإيعاز Update واحد فقط والمثال التالي يوضح ذلك.

SQL> UPDATE st1 SET age=30, gov='kut' WHERE name='ali';

ملاحظة.

من الممكن تحديث أكثر من عمود واحد في نفس الوقت وبنفس الإيعاز من خلال استخدام استعلام معين وكما هو موضح في المثال التالي:

```
SQL> UPDATE   employees SET no = (SELECT no FROM
employees WHERE    no = 205), salary = (SELECT salary
FROM employees WHERE no = 205)
WHERE    no = 114;
  1  rows updated
```

ملاحظة.

من الممكن تحديث البيانات الحالية الموجودة في الجدول باستخدام شروط خاصة ببيانات موجودة في جدول أخر وكما موضح في المثال أدناه:

```
SQL> UPDATE copy _ emp SET dep_id = (SELECT dep_id
FROM emp WHERE no = 100)
  WHERE   job_id =  (SELECT job_id FROM emp   WHERE
  no = 200);
```

ت-أيعاز DELETE.

أن هذا الإيعاز يستخدم لحذف البيانات المحددة من جدول معين ولكن هيكلية الجدول تبقى كما هي في قاعدة البيانات أي أننا نستطيع إضافة بيانات إلى ذلك الجدول في وقت لاحق وان لهذا الإيعاز مجموعة صيغ هي:

- من الممكن عدم تحديد شرط معين عند حذف البيانات، أي حذف بيانات الجدول بصورة كاملة وكما موضح في المثال أدناه.

SQL> DELETE st1; أن الإيعازأعلاه سوف يقوم بحذف بيانات الجدول بصورة كاملة.

- من الممكن حذف قيود محددة من خلال تحديد شرط معين والمثال التالي يوضح ذلك.

SQL> DELETE st1 WHERE age=30;

- من الممكن حذف بيانات موجودة في جدول معين بالاعتماد على شرط وضع على بيانات موجودة في جدول أخر مثال ذلك.

SQL> DELETE FROM emp WHERE no = (SELECT dep_id FROM dep WHERE dep_name = 'Public');

3- لغة استعادة البيانات Data Retrieval Language:

وتشمل أيعاز واحد فقط هو:

أ- **أيعاز Select .**

يستخدم هذا الإيعاز لاستعادة البيانات من قاعدة البيانات سواء أكان من الجداول أو Sequences أو الـ View أو غيرها ولهذا الإيعازثلاثة صيغ أساسية هما:

- من الممكن استخدام هذا الإيعاز لاستعادة كل البيانات الموجودة في الجدول، والصيغة العامة لتلك العملية هي.

SQL> SELECT * FROM table _ name;

Example: SQL> SELECT * FROM st1;

- من الممكن استخدام هذا الإيعاز لاستعادة أعمدة محددة من الجدول، والصيغة العامة لتلك العملية هي:

SQL> SELECT (column1, column2… column n) FROM table _ name;

Example: SQL> SELECT (no, name, age) FROM st1;

- من الممكن استخدام هذا الإيعاز لاستعادة البيانات من الجدول تحت شروط معينه وذلك باستخدام جملة الـ WHERE والصيغة العامة لهذا النوع هي:

SQL> SELECT * FROM table _ name WHERE (Specific Condition);

Or SQL> SELECT (column1… column n) FROM table _ name WHERE (Specific Condition);

Example: SQL> SELECT * FROM st1 WHERE (age>=30);

SQL> SELECT (no, name) FROM st1 WHERE (age > = 30);

هنالك ثلاثة أنواع من العمليات المستخدمة مع جملة الـ WHERE وهذه العمليات هي:

أ- **العمليات الرياضية Arithmetic Operations:** وتشمل العمليات التالية.

Operator Symbol	Description
+	Add Operation
–	Subtract Operation
*	Multiply Operation
/	Division Operation

ب- **عمليات المقارنة Comparison operation :** وتشمل خمسة أنواع أساسية هي.

- (=, !=, >, <, >=, <=) .
- Between, not between.
- In, not in.
- Null, not null.
- Like.

ت-**العمليات المنطقية Logical operations** : وتشمل ثلاثة أنواع أساسية هي.

- And.
- Or.
- Not.

Examples for the comparison operators:

A) – USING (= , > , < , >= , <= , ! = , <>).

SQL> select * from student where no = 2;

SQL> select * from student where no < 2;

SQL> select * from student where no > 2;

SQL> select * from student where no <= 2;

SQL> select * from student where no >= 2;

SQL> select * from student where no != 2;

SQL> select * from student where no <> 2;

B) – BETWEEN AND:

عند استخدام هذا الإيعاز يجب تحديد القيمة الأعلى والقيمة الاوطيء ،حيث أن القيم التي سوف يتم استرجاعها سوف تكون محصورة بين القيمة الأعلى والقيمة الاوطأ. وكما موضح في المثال الأتي.

SQL> select * from student where age between 25 and 35;

C) – NOT BETWEEN AND:

أن هذا الإيعاز يعمل عكس سابقه حيث انه يرجع القيم التي تقع خارج المدى المحدد لأعلى واقل قيمة، والمثال التالي يوضح ذلك.

SQL> select * from student where marks not between 200 and 400;

D) – IN:

أن هذا الإيعاز يقوم بإرجاع القيم التي يتم تحديدها له مسبقا، مثال ذلك.

SQL> select * from student where no in (1, 2, 3);

E) – NOT IN:

أن هذا الإيعاز يعمل عكس سابقه حيث يقوم بإرجاع القيم التي تكون غير موجودة ضمن التحديد، مثال ذلك. SQL> SELECT * FROM student WHERE no NOT IN (1, 2, 3);

F) – NULL:

أن هذا الإيعاز سوف يقوم بإرجاع القيود التي تحتوي على حقول فارغة في العمود المحدد، مثال ذلك.

SQL> select * from student where age is null;

G) – NOT NULL:

أن هذا الإيعاز يعمل عكس سابقه حيث سيقوم بإرجاع جميع القيم التي لا تحتوي على **NULL** في العمود المحدد، مثال ذلك.

```
SQL> select * from student where age is not null;
```

H) – LIKE:

أن هذا الإيعاز يقوم بإرجاع القيود التي تكون مشابه للشرط المعطى في العمود المحدد، ولهذا الإيعاز عدة صور هي.

- المثال أدناه سوف يقوم بإرجاع جميع القيود التي يكون فيها العمر مساوي إلى(**30**).

```
SQL> select * from student where marks like 100;
```

- المثال أدناه سوف يقوم بإرجاع جميع القيود التي يكون فيها الاسم يبدأ بحرف (**S**) .

```
SQL> select * from student where name like 'S%';
```

- المثال أدناه سوف يرجع جميع القيود التي تنتهي بالحرف (h).

```
SQL> select * from student where name like '%h';
```

- المثال أدناه سوف يرجع جميع القيود التي يكون فيها الحرف الثاني (a).

```
SQL> select * from student where name like '_a%';
```

- المثال أدناه سوف يرجع جميع القيود التي يكون فيها الحرف الثالث (a).

```
SQL> select * from student where name like '__d%';
```

- أمثلة أخرى.

```
SQL> select * from student where name like '%t_';
SQL> select * from student where name like '%e__';
SQL> select * from student where name like '%a% a %';
```

I) – AND:

أن هذا الإيعاز يقوم بإرجاع القيود التي تتحقق فيها جميع الشروط المحددة، مثال ذلك.

SQL> select * from student where no = 2 and marks >= 200;

J) – OR:

أن هذا الإيعاز يقوم بإرجاع القيود التي يتحقق فيها شرط واحد على الأقل، مثال ذلك.

SQL> select * from student where no = 2 or marks >= 200;

K) – ORDER BY:

أن هذا الإيعاز يقوم بترتيب بيانات العمود بصورة تصاعدية أو تنازلية، حيث أن القيمة التلقائية لعملية الترتيب في برنامج أوراكل هي الترتيب تصاعديا، أما أذا أردنا أجراء عملية الترتيب بصورة تنازلية في تلك الحالة يجب كتابة كلمة **(desc)** بعد اسم العمود والأمثلة التالية توضح ذلك.

SQL> select * from student order by no;

SQL> select * from student order by no desc;

L) – CASE:

تستخدم لإرجاع مجموعة من القيود التي تنطبق عليها شروط معينه يتم تحديدها من قبل المبرمج، وان الصيغة العامة لهذا الإيعاز موضحة في المثال أدناه.

```
SQL> Select sal,
  Case sal
  When 500 then ‘low’
```

When 5000 then ‘high’

Else ‘medium’

End case

From emp;

ملاحظة:

عندما يراد البحث واسترجاع قيم معينة عن الـ (String or Characters or Date) فيجب في تلك الحالة حصرها بين Single quotation ، حيث يجب الانتباه بان عملية البحث هنا تفرق بين الحروف الصغيرة والحروف الكبيرة إضافة إلى أن الصيغة التلقائية لكتابة التاريخ هي (DD-MM-YY) مثال ذلك.

SQL > SELECT (name, no) FROM student WHERE name = 'Ahmed';

ملاحظة :

من الممكن البحث عن قيمة معينه بالجدول من خلال قراءتها من الكيبورد وذلك باستخدام الصيغة التالية الموضحة في المثال أدناه:

SQL> SELECT * FROM student WHERE age = &age;

4- ايعازات السيطرة على العمليات Transaction Control (TC):

وتشمل ثلاثة ايعازات أساسية هي.

أ- أيعاز COMMIT .

يستخدم هذا الإيعاز لحفظ العمل وهنالك نوعان من هذا الإيعاز هما:

- **الإيعاز الضمني Implicit :**

أن هذا النوع ينفذ من قبل لغة أوراكل بصورة ضمنية في حالتين رئيسيتين ، الأولى عند تنفيذ أيعاز من ايعازات الـ DDL ، أما الحالة الثانية فهي عند الخروج من الـ SQL * Plus .

- **الإيعاز الصريح Explicit :**

ويتم تنفيذ هذا الإيعاز من قبل المبرمج من خلال كتابته ضمن ايعازات الـ SQL وكما موضح في المثال أدناه.

```
SQL> COMMIT;
```

ب-أيعاز ROLLBACK.

يستخدم هذا الإيعاز للتراجع عن تنفيذ العملية الحالية، حيث يتم تنفيذه في حالتين الأولى عند العمل على برنامج أوراكل وفجأة انقطع التيار الكهربائي عن الكمبيوتر وتم غلق برنامج أوراكل فجأة في تلك الحالة سوف يتم العودة إلى أخر نقطة تم عمل COMMIT عندها، والحالة الثانية عند استخدام هذا الإيعاز بصورة مباشرة من قبل المستخدم ، حيث ان الصيغة العامة لهذا الإيعاز موضحة في المثال التالي:

```
SQL> ROLLBACK;
```

ت- أيعاز SAVEPOINT.

يستخدم هذا الإيعازلغرض حفظ العمل عند نقطة معينة والعودة إليه بواسطة استخدام أيعاز الـ ROLLBACK في وقت لاحق وكما هو موضح في الأمثلة التالية.

```
SQL> INSERT INTO st1 VALUES (8,'MAHA',26,'ANBAR');
```

SQL> SAVEPOINT s1;

SQL> INSERT INTO st1 VALUES (9,'BAHAA',31,'DIALA');

SQL> SAVEPOINT s2;

SQL>INSERTINTOst1VALUES (10,'JWAD',33,'KARBALLA');

SQL> ROLLBACK to s1;

أن الإيعازأعلاه سوف يعود إلى النقطة S1 أي انه سوف يلغي أخر عمليتي إدخال.

5-لغة السيطرة على البيانات (Data Control Language (DCL:

وتشمل ايعازين أساسيين هما.

أ- أيعاز Grand.

يستخدم هذا الإيعاز لغرض منح صلاحيات معينة لمستخدم أخر، والصيغة العامة لهذا الإيعاز هي.

SQL> GRAND (privileges) (*object t_ name*) to < *user _ name* > [with grant option];

SQL> grant (select) on student to user1; -- you can give individual privilege.

SQL> grant (select, insert) on student to user2; -- you can give set of privileges.

SQL> grant all on student to user3; -- you can give all privileges.

- أن كل الـ users أعلاه لا يستطيعون منح الصلاحيات إلى الـ Users الأخرى ولعمل ذلك نقوم بكتابة الإيعاز التالي.

SQL> grant all on student to user4 with grant option;

- فمن خلال الإيعازأعلاه سوف يستطيع User4 منح الصلاحيات إلى الـ Usersالأخرى.

ب-أيعاز REVOCK.

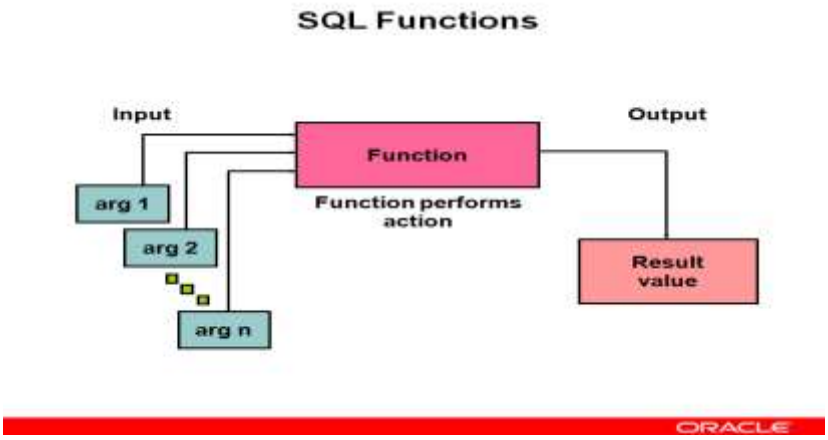
أن وظيفة هذا الإيعاز هي سحب الصلاحيات التي منحت إلى USER معين في وقت سابق باستخدام أيعاز GRANT، والصيغة العامة لهذا الإيعاز هي.

SQL> Revoke < *privileges*> on <*object _ name*> from < *user _ name*>;SQL> revoke (select) on student from user1; --- you can revoke individual privilege.

SQL> revoke (select, insert) on student from user2; -- you can revoke set of privileges.

SQL> revoke all on student from user3; -- you can revoke all privileges.

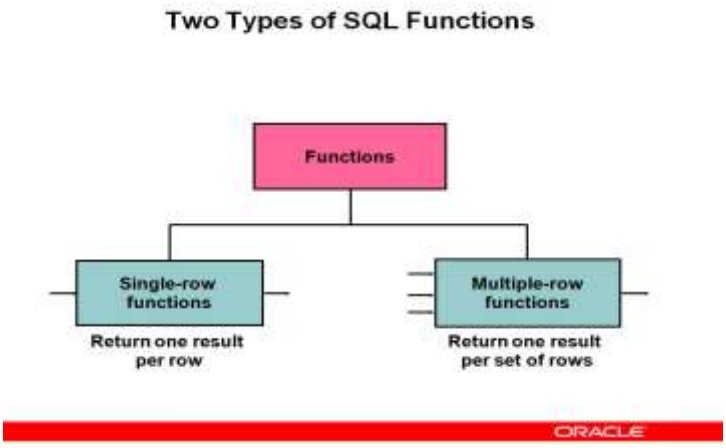
دوال الـ SQL:



تقسم دوال الـ SQL إلى قسمين رئيسيين هما:

1- Single row function.

2- Group function.



وتشمل ستة دوال رئيسية هي:

1- Single row function :

a) NUMBER FUNCTIONS

b) CHARACTER FUNCTIONS

c) DATE FUNCTIONS

d) ANALYTICAL FUNCTIONS

e) MISCELLANEOUS FUNCTIONS

f) CONVERSION FUNCTIONS

2- Group function: أو ما تدعى بالـ Multi Row Function وتحتوي على ثلاثة مجاميع رئيسية هي.

a) AGGREGATE FUNCTION.

- SUM (c1 of table1).

- COUNT (c1 of table1).
- MAX (c1 of table1).
- MIN (c1 of table1).
- AVG (c1 of table1).

b) NUMBER FUNCTIONS.

- ROUND.
- TRUNCATE.
- CEIL.
- FLOOR.
- MODULUS.
- POWER.
- SQRT.
- ABS.
- SIGN.

c) CHARACTER FUNCTIONS

- LOWER
- UPPER
- INITCAP
- CONCAT

1- دوال الـ Single Row Function .

تتميز هذه الدوال بالخصائص التالية:

أ- تستقبل مجموعة معاملات وترجع قيمة واحدة فقط لكل سطر.

ب- تستطيع تحديث نوع البيانات.

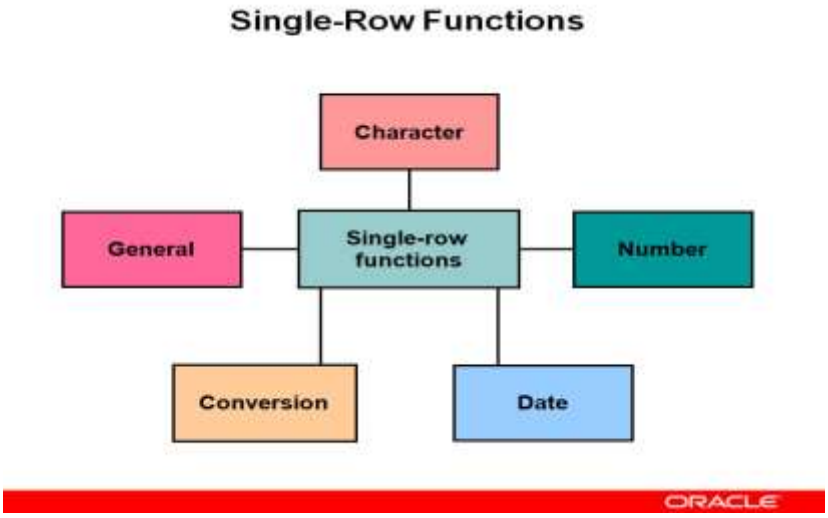
ت- من الممكن أن تكون متداخلة.

ث- من الممكن أن تقبل المعاملات التي تكون عبارة عن أعمدة أو تعابير رياضية.

الصيغة العامة لهذه الدوال هي.

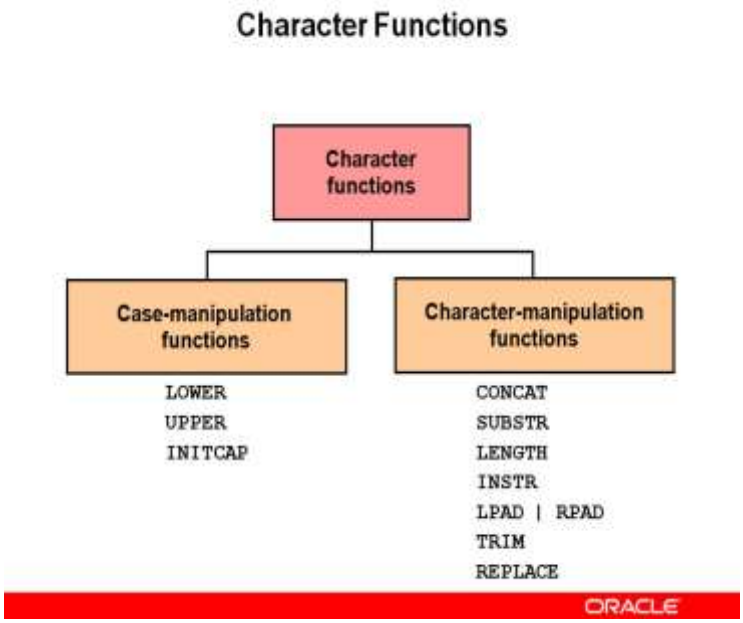
SQL>***function _ name*** **[(*arg1, arg2,…,argn*)];**

هنالك خمسة أنواع من البيانات تتقبلها دوال الـ Single Row Function هي:



أنواع الدوال الخاصة بالـ Single Row Function .

1- **الدوال الحرفية Character Function :** وتقسم إلى قسمين أساسين هما الـ Case Manipulation Function. والـ CharacterManipulation Function وهي موضحة في الشكل التالي.



أ- **دوال الـ Case Manipulation Function :**

تحتوي على ثلاثة دوال أساسية هي.

1) دالة الـ Lower وظيفتها تحويل الأحرف الموجود إلى أحرف صغيرة.
2) دالة الـ Upper وظيفتها تحويل الأحرف الموجودة إلى أحرف كبيرة.
3) دالة الـ INITCAP وظيفتها تحويل بداية أول حرف في الكلمة إلى حرف كبير وباقي أحرف الكلمة إلى حروف صغيرة، وكما موضح في المثال أدناه.

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

ب-**دوال الـ Character Manipulation Function** : وتحتوي على سبعة دوال رئيسية هي:

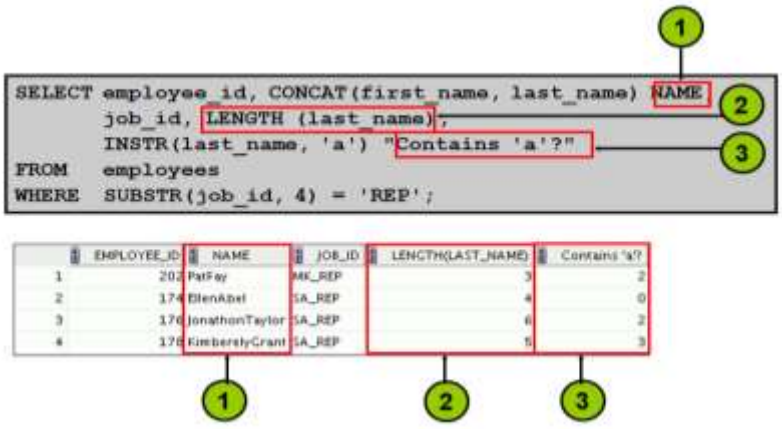
1) دالة الـ CONCAT وظيفتها دمج two string .
2) دالة الـ SUBSTR وظيفتها قطع جزء من نص معين حيث أنها تحتوي على ثلاثة معاملات الأول يمثل النص المطلوب قصه والثاني يمثل بداية القص والثالث يمثل عدد الحروف المقصوصة.
3) دالة الـ LENGTH وظيفتها إرجاع طول string معينة .
4) دالة الـ INSTR وظيفتها البحث عن حرف محدد داخل نص معين وإرجاع رقم موقع ذلك الحرف.
5) دالة الـ LPAD &RPAD وظيفتها تكملة مراتب الرقم المعطى بمجموعة من الرموز، حيث أنها تحتوي على ثلاثة معاملات رئيسية المعامل الأول

يمثل الرقم المطلوب تكملة مراتبه والمعامل الثاني يمثل عدد المراتب والمعامل الثالث يمثل الرمز المطلوب تكملة المراتب فيه.

6) دالة الـ REPLACE وظيفة هذا الدالة هي استبدال حرف معين او مجموعة من الأحرف بأحرف أخرى.

7) دالة الـ TRIM وظيفتها قص حرف محدد او مجموعة من الأحرف من نص معين، والأمثلة التالية توضح عمل الدوال أعلاه.

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld



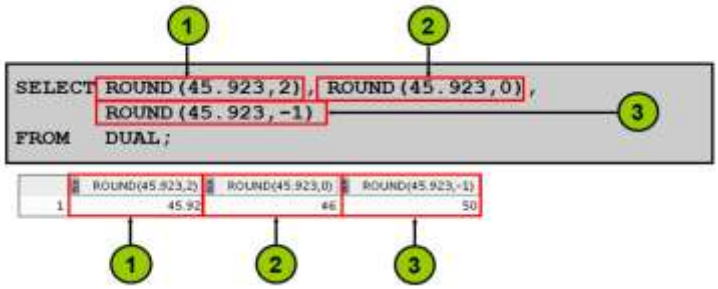
2- **الـدوال الرقميـة Number Functions** : ويحتوي هذا النوع على ثلاثة دوال رئيسية هي.

أ) دالة الـ ROUND وظيفتها تقريب العدد إلى اقرب عدد عشري حيث أن هذه الدالة تحتوي على معاملين الأول يمثل الرقم المطلوب تقريبه والثاني يمثل عدد المراتب العشرية المطلوبة.

ب) دالـة الـ TRUNC أن وظيفة هذا الدالة هي قطع العدد العشري إلى اقرب عدد يتم تحديد مراتبه من قبل المستخدم حيث أن هذه الدالة تحتوي على معاملين الأول يمثل العدد العشري والثاني يمثل عدد المراتب العشرية المطلوب قطعها.

ت) دالة الـ MOD أن وظيفة هذه الدالة هي إرجاع باقي عملية القسمة، والأمثلة التالية توضح عمل الدوال أعلاه.

Function	Result
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100



3- دوال التاريخ Date Function.

أن قاعدة بيانات برنامج أوراكل تقوم بخزن التاريخ داخليا بشكل صيغة عددية تضم (القرن ، السنة ، الشهر ، اليوم ، الساعة ، الدقيقة ، الثانية) حيث أن

الصيغة التلقائية لعرض التاريخ هي DD-MON-RR وكما موضح في المثال التالي.

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-88';
```

	LAST_NAME	HIRE_DATE
1	Whalen	17-SEP-87
2	King	17-JUN-87

- هنالك دالة تدعى بالـ **SYSDATE**وظيفتها أعادة كل من التاريخ والوقت.
- هنالك ثلاثة أنواع من العمليات نستطيع استخدامها مع الـ **DATE** هي.

1- نستطيع إضافة رقم إلى التاريخ أو طرح رقم معين من التاريخ.

2- طرح تاريخين لإيجاد عدد الأيام المحصورة بين تلك التواريخ.

3- إضافة ساعات إلى التاريخ بواسطة تقسيم التاريخ على (24) ساعة في اليوم الواحد.

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

	LAST_NAME	WEEKS
1	King	1116.148574735449735449735449735449735449735...
2	Kochhar	998.005717592582592592592592582592592593
3	De Haan	825.148574735449735449735449735449735449735...

هنالك ستة أنواع من دوال الـ DATE هي.

أ) دالـة الـ **MONTHS_BETWEEN** وظيفتهـا إرجـاع عـدد الأشـهر بـين تاريخين.

ب) دالة الـ **ADD_MONTHS** وظيفتها إضافة عدد من الأشهرإلى التاريخ.

ت) دالة الـ **NEXT_DAY** وظيفتها أيجاد اليوم التالي للتاريخ المحدد.

ث) دالة الـ **LAST_DAY** وظيفتها أيجادأخر يوم في الشهر المحدد.

ج) دالة الـ **ROUND** وظيفتها تقريب التاريخ .

ح) دالة الـ **TRUNC** وظيفتها قطع التاريخ.

والأمثلة التالية توضح عمل الدوال أعلاه.

Function	Result
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

Assume SYSDATE = '25-JUL-03':

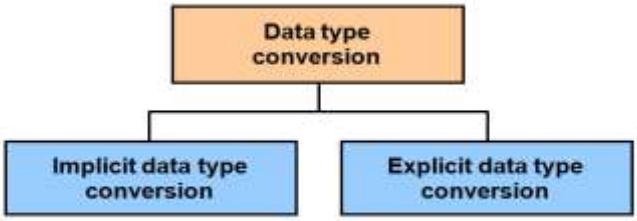
Function	Result
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE ,'YEAR')	01-JAN-04
TRUNC(SYSDATE ,'MONTH')	01-JUL-03
TRUNC(SYSDATE ,'YEAR')	01-JAN-03

4- دوال التحويل **Conversion Function** .

يستخدم هذا النوع من الدوال للتحويل بين أنواع البيانات المختلفة ويوجد من هذه الدوال نوعين هما:

أ) دوال التحويل الضمنية **Implicit data type conversion function** .

ب) دوال التحويل الصريحة **Explicit data type conversion function.**



أ) دوال التحويل الضمنية :

عند استخدام جملة المساواة في برنامج الاوراكل فان الـ Oracle Server يستطيع التحويل بصورة أوتوماتيكية عند الحاجة بين الأنواع التالية.

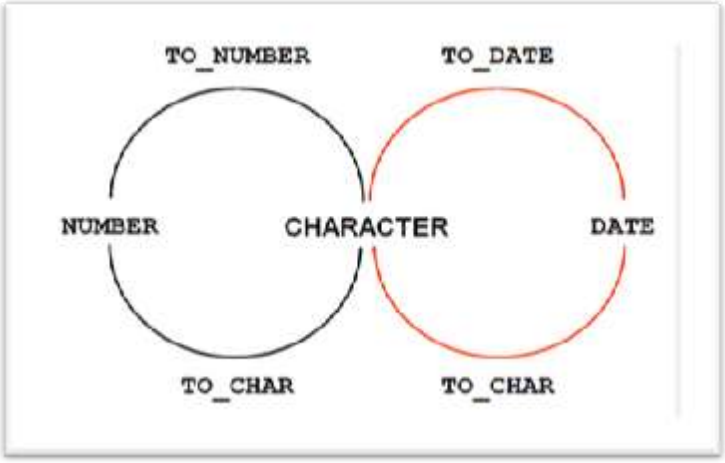
From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

وعند استخدام تعابير التقييم فان الـ Oracle Server يستطيع التحويل بين القيم التالي.

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

ب) استخدام دوال التحويل الصريحة:

هنالك أربعة دوال مهمة في لغة أوراكل تستخدم للتحويل بين الأنواع المختلفة للبيانات عند الحاجة ، وهذه الدوال مبينة في المخطط التالي:



1- استخدام دالة الـ TO_CHAR مع التاريخ ، حيث ان الصيغة العامة لاستخدام دالة الـ TO_CHARمع التاريخ هي.

SQL> TO_CHAR (*date,* '*format _ model*);

وان الصيغة العامة لهذه الدالة يجب أن تملك الخواص التالية:

- ✓ يجب أن يوضع الـ *format _ model* ' بين علامتي اقتباس.
- ✓ عند استخدام النصوص فأنها تميز بين الحروف الصغيرة والحروف الكبيرة.
- ✓ من الممكن أن تحتوي على أي صيغة تاريخ صحيحة.
- ✓ توجد هنالك أداة اسمها (fm) من الممكن استخدامها لحذف الفراغات والأصفار الغير ضرورية وغير مرغوب فيها.
- ✓ يتم الفصل بين قيمة التاريخ بواسطة (,).

توجد هنالك ثمانية صيغ مختلفة تستخدم للتاريخ، هذه الصيغ هي:

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

- Time elements format the time portion of the date:

HH24:MI:SS AM	15:45:32 PM

- Add character strings by enclosing them in double quotation marks:

DD "of" MONTH	12 of OCTOBER

- Number suffixes spell out numbers:

ddspth	fourteenth

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
...		
19	Taylor	24 March 1998
20	Grant	24 May 1999

2- استخدام دالة الـ TO_CHAR مع الأعداد Number ، وان الصيغة العامة لاستخدام هذه الدالة مع الأعداد هي:

SQL> TO_CHAR (*number,* '*format _ model*');

وهذه بعض الصيغ التي تستخدم مع هذه الدالة لغرض عرض القيم الرقمية كأحرف، وهي مبينة كالتالي:

Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as thousands indicator

والمثال التالي يوضح كيفية استخدام الدالة أعلاه مع القيم الرقمية.

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	SALARY
1	$6,000.00

3- استخدام دالة الـ TO _ NUMBER لتحويل الـ String إلى القيم العددية ، والصيغة العامة لهذه الدالة هي:

SQL> TO_NUMBER (char ['format _ model']);

4- استخدام دالـة الـ DATE _ TO لتحويل الصيغ النصية الى تاريخ ، والصيغة العامة لهذه الدالة هي:

SQL> TO_DATE (char ['format _ model']);

عند استخدام التاريخ هنالك صيغة تدعى باالـ DATE _ RR وهي موضحة في الجدول المبين في أدناه:

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">If the specified two-digit year is:</td></tr>
<tr><td>0–49</td><td>50–99</td></tr>
<tr><td rowspan="2">If two digits of the current year are:</td><td>0–49</td><td>The return date is in the current century</td><td>The return date is in the century before the current one</td></tr>
<tr><td>50–99</td><td>The return date is in the century after the current one</td><td>The return date is in the current century</td></tr>
</table>

مثال ذلك لو كنا نريد معرفة الموظفين الذين تم تعيينهم قبل عام 1990في تلك الحالة سوف نستخدم صيغة الـ FORMAT DATE RR والموضحة في المثال التالي.

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

5- دوال عامة General Functions .

هنالك أربعة أنواع من الدوال العامة التي توفرها لغة اوراكل للتعامل مع أي نوع من البيانات والمتعلقة أيضا بالـ NULL ، وهذه الدوال هي:

- ✓ NVL (expr1, expr2)
- ✓ NVL2 (expr1, expr2, expr3)
- ✓ NULLIF (expr1, expr2)
- ✓ COALESCE (expr1, expr2, …, exprn)

أ) دالة الـ NVL .

أن وظيفة هذه الدالة هي تحويل الـ NULL VALUE الى قيمة حقيقية ACTUAL VALUE ، حيث أن البيانات المستخدمة مع هذه الدالة هي (NUMBER,CHARACTER,DATE)

كما أن نوعية البيانات المستخدمة يجب أن تكون متطابقة في الدالة الواحدة، وكما موضح في الأمثلة التالية.

SQL> NVL (commission_pct, 0);

SQL> NVL (hire_date,'01-JAN-97');

SQL> NVL (job_id,'No Job Yet');

```sql
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000

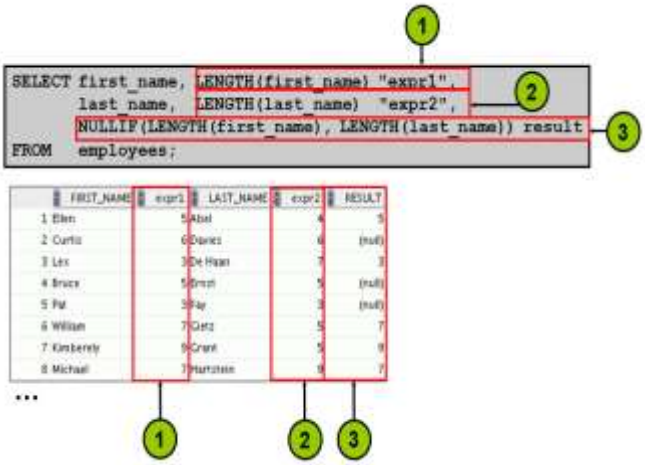
...

ب) المثال التالي يوضح كيفية استخدام دالة NVL2 .

```sql
SELECT last_name,  salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

ج) والمثال أدناه يوضح كيفية استخدام دالة الـ NULLIF .



د) دالة الـ COALESCE.

أن من مزايا استخدام هذه الدالة هي إمكانيتها على اخذ القيم المتعددة والمكررة، فإذا كان أول تعبير هو NOT NULL فان هذه الدالة سوف تقوم بإرجاع التعبير، عدا ذلك فان هذه الدالة سوف تبقي التعبير نفسه والمثال التالي يوضح عمل هذه الدالة.

```
SELECT last_name,
       COALESCE(manager_id,commission_pct, -1) comm
FROM   employees
ORDER BY commission_pct;
```

	LAST_NAME	COMM
1	Grant	149
2	Taylor	149
3	Ziotkey	100
4	Abel	149
5	King	-1
6	Kochhar	100
7	De Haan	100
8	Hunold	102

...

التعبير الشرطي Conditional Expression.

يوفر برنامج الاوراكل دالتين أساسيتين لغرض تنفيذ التعبير الشرطي Conditional Expression وهذه الدالتين هما :

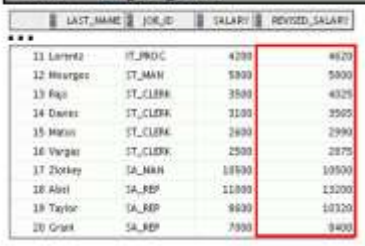
1- دالة الـ Case Statement .

تقوم هذه الدالة بتسهيل تنفيذ عدة شروط في نفس الوقت وبسهولة تامة أفضل من الـ IF THEN ELSE والصيغة العامة لهذه الدالة هي:

```
SQL> CASE expr
WHEN (comparison_expr1) THEN return_expr1
WHEN (comparison_expr2) THEN return_expr2
WHEN (comparison_exprn) THEN return_exprn
ELSE else_expr]
 END
```

- المثال التالي يوضح كيفية استخدام هذه الدالة.

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE       salary END     "REVISED_SALARY"
FROM   employees;
```



2- دالة الـ DECODE STATEMENT .

أن هذه الدالة تقوم بنفس عمل الدالة السابقة مع سهولة أكثر في الهيكلية، والصيغة العامة لهذه الدالة هي:

```
SQL> DECODE
(Col _expression, search1, result1
```

[, *search2, result2…]*

[, *default*])

المثال التالي يوضح كيفية استخدام هذه الدالة.

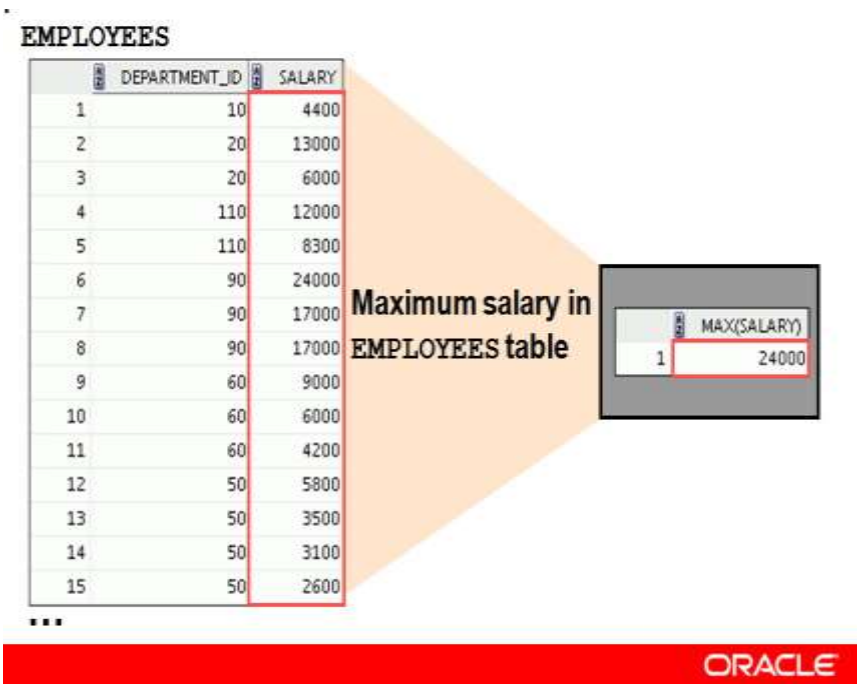
```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

والمثال التالي يوضح كيفية استخدام هذه الدالة في تطبيق نسبة الضريبة على كل موظف في القسم ذات الرقم (80).

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                       0, 0.00,
                       1, 0.09,
                       2, 0.20,
                       3, 0.30,
                       4, 0.40,
                       5, 0.42,
                       6, 0.44,
                          0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

دوال المجموعة Group Functions.

أن وظيفة هذه الدوال هي العمل على مجموعة من القيود لإعطاء قيمة واحدة كنتيجة لكل مجموعة، كما موضح في المثال التالي.



أنواع دوال المجموعة.

هنالك سبعة أنواع من هذه الدوال موضحة أدناه:

1- دالة الـ AVG .

أن وظيفة هذه الدالة هي أيجاد معدل مجموعة من القيم المحددة من قبل المبرمج في عمود معين مع تجاهل قيم الـ Null ، وتستخدم فقط مع البيانات الرقمية Numeric، والصيغة العامة لهذه الدالة هي.

```
SQL>AVG ( [DISTINCT |ALL] n);
SQL>SELECT AVG (mark)
FROM student3
WHERE name LIKE 'ali';
```

2- دالة الـ COUNT .

أن وظيفة هذه الدالة هي أيجاد عدد القيود التابعة لعمود معين يتم تحديده من قبل المستخدم من ضمنها القيم المكررة وقيم الـ Null ، والصيغة العامة لهذه الدالة هي.

```
SQL> COUNT ((* | [DISTINCT |ALL] expr));
SQL>SELECT COUNT (mark)
FROM student3
WHERE name LIKE 'ali';
```

3- دالة الـ Max.

أن وظيفة هذه الدالة هي أيجاد اكبر قيمة من مجموعة من القيم المحددة من قبل المستخدم لعمود معين مع تجاهل قيمة الـ Null، وتستخدم فقط مع البيانات الرقمية Numeric والبيانات النصية Varchar والتاريخ Date ولا يمكن استخدامها مع البيانات الكبيرة الحجم LOB & LONG، والصيغة العامة لهذه الدالة هي.

```
SQL> MAX ( [DISTINCT | ALL] expr );
SQL>SELECT MAX (mark)
FROM student3
WHERE name LIKE 'ali';
```

4- دالة الـ Min .

أن وظيفة هذه الدالة هي أيجاد اصغر قيمة من مجموعة من القيم المحددة من قبل المستخدم والتابعة لعمود معين مع تجاهل قيمة الـ Null، وتستخدم فقط مع البيانات الرقمية Numeric والبيانات النصية Varchar والتاريخ

Date ولا يمكن استخدامها مع البيانات الكبيرة الحجم LOB & LONG، والصيغة العامة لهذه الدالة هي.

```
SQL> MIN ( [DISTINCT | ALL] expr );
SQL> SELECT MIN (mark)
        FROM student3
       WHERE name LIKE 'ali';
```

5- دالة الـ STDDEV.

أن وظيفة هذا الدالة هي أيجاد ناتج القسمة لمجموعة من القيم المحددة من قبل المستخدم مع تجاهل قيمة الـ Null ، وتستخدم فقط مع البيانات الرقمية Numeric ، والصيغة العامة لهذه الدالة هي .

```
SQL> STDDEV ([DISTINCT | ALL] x);
SQL> SELECT STDDEV (mark)
FROM student3
WHERE name LIKE 'ali';
```

6- دالة الـ SUM.

أن وظيفة هذه الدالة هو أيجاد المجموع الخاص بمجموعة من القيم المحددة من قبل المستخدم لعمود معين مع تجاهل قيمة الـ Null، وتستخدم فقط مع البيانات الرقمية Numeric، والصيغة العامة لها هي.

```
SQL> SUM ([DISTINCT | ALL] n);
SQL> SELECT SUM (mark)
FROM student3
WHERE name LIKE 'ali';
```

7- دالة الـ VARIANCE .

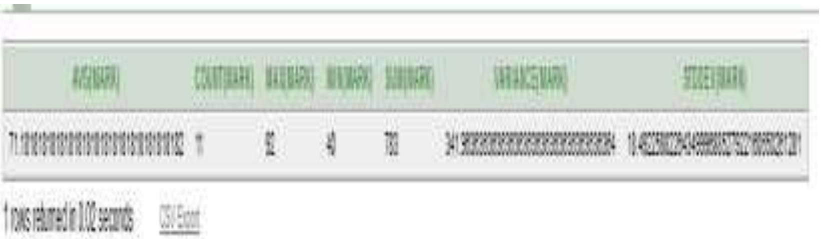
أن وظيفة هذه الدالة هو أيجاد درجة التفاوت والاختلاف لمجموعة من القيم مع تجاهل قيمة الـ Null ، وتستخدم فقط مع البيانات الرقمية Numeric ، والصيغة العامة لهذه الدالة هي.

```
SQL> VARIANCE ([DISTINCT | ALL] x);
SQL> SELECT VARIANCE (mark)
FROM student3
WHERE name LIKE 'ali';
SQL> SELECT AVG (mark) , MAX (mark) , MIN (mark) ,
SUM (mark)
FROM student3
WHERE name LIKE 'ali';

SQL> SELECT AVG (salary) , MAX (salary) , MIN (salary) ,
SUM (salary)
FROM employees
WHERE dep LIKE 'registration';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

```
SQL> select avg (mark),
Count (mark),
```

```
Max (mark),
Min (mark),
Sum (mark),
Variance (mark),
Stddev (mark)
From student3;
```



AVG(MARK)	COUNT(MARK)	MAX(MARK)	MIN(MARK)	SUM(MARK)	VARIANCE(MARK)	STDDEV(MARK)
[illegible]	11	92	40	783	[illegible]	[illegible]

1 rows returned in 0.02 seconds   CSV Export

ملاحظة:

أن كل دوال الـ Group Function تستثني قيم الـ Null من العمليات التي تقوم بها ، ولغرض اخذ قيمة الـ Null بنظر الاعتبار عند الحاجة إليها فيمكن استخدام دالة الـ NVL وكما موضح في المثال التالي.

```
1  SELECT AVG(commission_pct)
   FROM   employees;
```

	AVG(COMMISSION_PCT)
1	0.2125

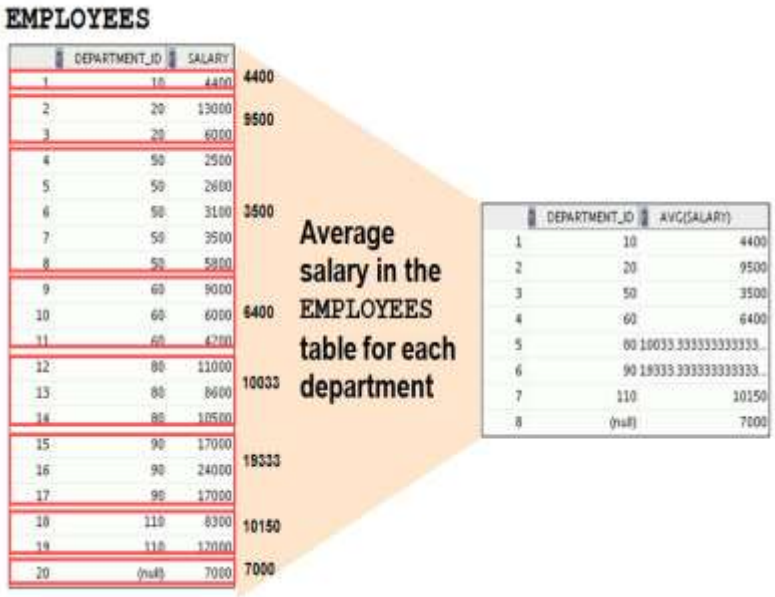
The NVL function forces group functions to include null values:

```
2  SELECT AVG(NVL(commission_pct, 0))
   FROM   employees;
```

	AVG(NVL(COMMISSION_PCT,0))
1	0.0425

استخدام دالة الـ Group BY.

تستخدم هذه الدالة لغرض تجزئة القيود الموجودة في جدول معين إلى مجموعات صغيرة، ومن ثم استخدام الدوال السابقة Group Functions لاسترجاع ملخص بالمعلومات لكل Group. وكم موضح في الشكل أدناه.



والصيغة العامة لهذه الدالة هي :

```
SELECT    column, group_function(column)
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

وكما موضح في المثال التالي.

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id ;
```

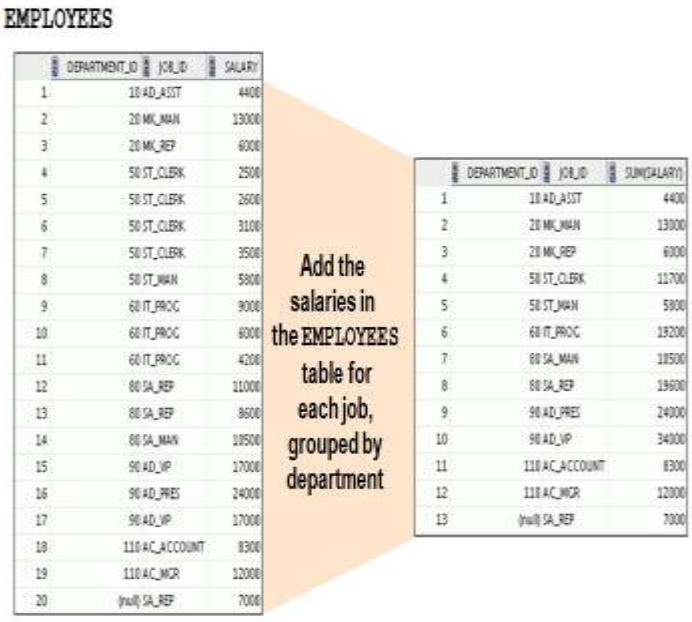
	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

```
SQL> SELECT    department_id , AVG(salary)
 FROM    employees
 GROUP BY department_id
 ORDER BY AVG(salary);
```

	DEPARTMENT_ID	AVG(SALARY)
1	50	3500
2	10	4400
3	60	6400
4	(null)	7000
5	20	9500
6	80	10033.3333333333...
7	110	10150
8	90	19333.3333333333...

ملاحظة :

من الممكن استخدام دالة الـ Group By مع أكثر من عمود واحد ، أي عمل Group داخل Group ، وكما موضح في الشكل أدناه.



Example:

```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department_id, job_id;
```

	DEPT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	90	AD_VP	34000
3	50	ST_CLERK	11700
4	80	SA_REP	19600
5	110	AC_MGR	12000
6	50	ST_MAN	5800
7	80	SA_MAN	10500
8	20	MK_MAN	13000
9	90	AD_PRES	24000
10	60	IT_PROG	19200
11	(null)	SA_REP	7000
12	10	AD_ASST	4400
13	20	MK_REP	6000

ملاحظة .

من الممكن تقييد مخرجات دالة الـ Group By بشرط معين بواسطة استخدام دالة أخرى تدعى بدالة الـ HAVING ، فعند استخدام هذه الدالة يتم في البداية تجميع الأسطر الموجودة في الجدول وحسب الشرط الموجود مع دالة الـ Group By بعد ذلك يتم تطبيق الشرط الموجود مع دالة الـ Having لإظهار النتائج النهائية ، وان الصيغة العامة لهذه الدالة هي:

```
SELECT    column, group_function
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[HAVING   group_condition]
[ORDER BY column];
```

Example:

```
SELECT   department_id, MAX(salary)
FROM     employees
GROUP BY department_id
HAVING   MAX(salary)>10000 ;
```

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
11	60	4200
12	50	5800
13	50	3500
14	50	3100
15	50	2600

...

The maximum salary per department when it is greater than $10,000

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	80	11000
3	90	24000
4	110	12000

التجزئة Partition :

هي عملية تقسيم جدول معين إلى عدة جداول بصورة فيزيائية ومستقلة عن بعضها البعض ومرتبة اعتمادا على قيم الـ Keys الموجودة ، حيث أن كل قسم من هذا الجدول يدعى بالـ Partition ، مع ملاحظة ان اي جدول من الممكن إلغاء عملية الـ Partition له وإرجاعه إلى وضعه الطبيعي ولكن في تلك الحالة من غير الممكن عمل Partition له مرة أخرى.

مزايا استخدام الـ Partition:

1- اختصـار الوقـت المطلـوب لأجـراء عمليـة الصـيانة علـى أجـزاء الجـدول ، وجعل الأجزاء الأخرى متاحة للمستخدم لغرض العمل عليها، فعلى سبيل المثال أن عملية الصيانة على جدول معين سوف تتم على Partition واحد تلو الأخر أما بقية الـ Partitions فأنها تكون متاحة للمستخدمين لغرض العمل عليها .

2- عند فشل عملية معالجة البيانات او نقلها لـ Partition معين فأنها لن تؤثر على بقية الـ Partitions الأخرى التابعة للجدول.

3- أن عمليـة اسـتقلال الـ Partition تسـمح للمسـتخدمين بالاسـتخدام المشـترك للأنواع المختلفة للـ Partitions لأغراض مختلفة.

4- اختصار إمكانية تلف البيانات.

5- أجراء عمليـة النسـخ الاحتيـاطي والتصـحيح للبيانـات لكل Partition بشكل مستقل.

مساوئ استخدام طريقة الـ Partition .

أن من مسـاوئ عمليـة تقسيم الجـدول إلـى مجموعـة من الـ Partition هـي عدم احتواء الأعمدة والصفوف على بيانات كبيرة الحجم من نوع LOB.

أنواع الـ Partition .

هنالك نوعان أساسيان من الـ Partition هما:

1- التجزئة بالمدى Range Partition .

من الممكن عمل هذا النوع من التجزئة كما موضح في المثال التالي.

```
SQL> CREATE TABLE student3
No number(3),
Name varchar2(30)
```

Partition P1 values less than(10),

Partition P2 values less than(20),

Partition P3 values less than(30),

Partition P4 values less than(max value)

ملاحظة:

عند استخدام دالة الـ max value في أخر Partition في تلك الحالة لا يمكن إضافة Partition أخر بعدها.

عند أجراء عملية إدخال البيانات إلى الجدول فأنها سوف تكون بالصيغة التالية.

SQL> INSERT INTO student3 VALUES (1,'ali');

- ✓ سوف تذهب بصورة مباشرة إلى Partition1 .

SQL>INSERT INTO student3 VALUES (13,'ahmed');

- ✓ سوف تذهب بصورة مباشرة إلى Partition2.

SQL>INSERT INTO student3 VALUES (24,'hasan');

- ✓ سوف تذهب بصورة مباشرة إلى Partition3.

SQL>INSERT INTO student3 VALUES (36,'mohammed');

- ✓ سوف تذهب بصورة مباشرة إلى Partition4.

كيفية استعادة البيانات من الـ range partition .

يتم استعادة البيانات من الـ range partition كما موضح في الأمثلة التالية.

SQL> SELECT * FROM student3;

SQL> SELECT * FROM student3 partition (P3);

العمليات الممكن إجرائها على الـ range partition .

هنالك أربعة عمليات مهمة من الممكن إجرائها على الـ range partition وهذه العمليات هي:

1- إضافةPartition. ويتم ذلك كما موضح في المثال التالي:

```
SQL> alter table student3 add partition P5 values less than
(40);
```

2- حذف Partition. ويتم ذلك كما موضح في المثال التالي:

```
SQL> alter table student3 drop partition P4;
```

3- أعادة تسمية Partitionمعين. ويتم ذلك كما موضح في المثال التالي:

```
SQL> alter table student3 rename partition P3 to P7;
```

4- قطع Partition معين.ويتم ذلك كما موضح في المثال التالي:

```
SQL> alter table student3 truncate partition P6;
```

2- التجزئة باستخدام القوائم List Partition .

تتم هذه العملية بواسطة استخدام مجموعة من التحديدات والقوائم، ويتم بنائها كما موضح في المثال التالي.

```
SQL>CREATE TABLE student4
No number (3),
Name varchar2 (30),
Branch varchar2 (20)
Partition by list (branch)
Partition P1 values ('MSC','MBA'),
  Partition P2 values ('MTIC','BTIC')'
  Partition P3 values ('DBA','NMA'));
```

من الممكن إضافة قيم إلى الجدول كما موضح في المثال التالي.

SQL> insert into student4 values (1,'ali','MSC');

- ✓ سوف يذهب بصورة مباشرة إلى P1.

SQL> insert into student4 values (2,'ahmed','MTIC');

- ✓ سوف يذهب بصورة مباشرة إلى P2.

SQL> insert into student4 values (3,'sarha','DBA').

- ✓ سوف يذهب بصورة مباشرة إلى P3.

من الممكن استعادة البيانات من الجدول كما موضح في الأمثلة التالية.

SQL> SELECT * FROM student4;

SQL> SELECT * FROM student4 partition P3;

العمليات التي من الممكن إجرائها على الـList Partition.

1- إضافة Partition.

2- حذف Partition.

3- قطع Partition.

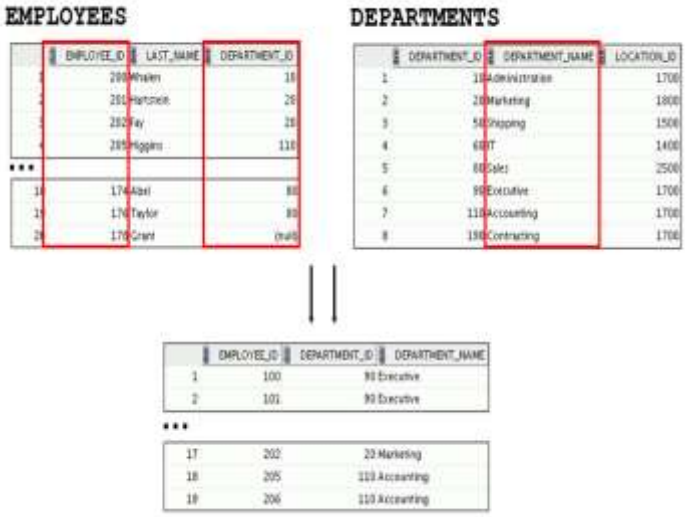
4- إعادة تسمية Partition.

الروابط Joins .

أن الوظيفة الأساسية لهذه العملية هي ربط البيانات الموجودة في أكثر من جدول مع بعضها البعض وحسب شروط معينة يتم تحديدها من قبل المبرمج ، بحيث انه أذا تم ربط جدولين مع بعضهما يتم في تلك الحالة ربط الجدولين حسب الشروط الموضوعة.

أما في حالة ربط أكثر من جدولين في تلك الحالة يتم مقارنة أول جدولين مع بعضهما حسب الشرط الموضوع.

ومن ثم مقارنة ناتج الجدولين مع الجدول الثالث لإيجاد النتيجة النهائية، وكما موضح في الشكل أدناه.



أنواع الروابط :

هنالك خمسة أنواع مهمة من الروابط، كل نوع يستخدم حسب الحاجة إليه وهذه الأنواع هي:

1- الربط الطبيعي Natural Join.

2- الربط الداخلي Inner Join.

أ- الربط الداخلي باستخدام عبارة الـ WHERE.

ب- الربط الداخلي باستخدام عبارة الـ USING.

ت- الربط الداخلي باستخدام عبارة الـ ON.

3- الربط الخارجي Outer Join .

أ – الربط الخارجي الأيمنRIGHT OUTER JOIN.

ب– الربط الخارجي الأيسر LEFT OUTER JOIN.

ت– الربط الخارجي المتكامل FULL OUTER JOIN.

4– ربط الجدول بنفسه Itself Join.

5– الربط المتقاطع Cross Join.

1– الربط الطبيعي Natural Join.

يستخدم هذا النوع من الربط عندما نريد ربط كل الأعمدة الموجودة في جدولين يحتويان نفس الاسم ونفس نوعية البيانات ، ويتم ربط هذا النوع بواسطة استخدام كلمة NaturalJoin ، مع التأكيد على أن الأعمدة المربوطة في كلا الجدولين يجب ان تحتوي على نفس الاسم ونفس نوعية البيانات ، فإذا كانت تحتوي على نفس الاسم ولكنها لا تحتوي على نفس نوعية البيانات في تلك الحالة سوف ترجع Syntax error، والمثال التالي يوضح ذلك.

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

في المثال أعلاه سوف يتم ربط الجدول المسمى location مع الجدول المسمى department بواسطة العمود المسمى location_id والذي يحتوي على نفس الاسم ونفس نوعية البيانات في كلا الجدولين ، أما أذا كان هنالك عمود أخر يحتوي على نفس الاسم ونفس نوعية البيانات فيجب استخدامه أيضا في عملية الربط .

ملاحظة :

من الممكن تقييد عملية الربط بشرط معين باستخدام جملة الـ Where وكما موضح في المثال التالي.

```
SQL> SELECT department_id, department_name,
location_id, city
FROM departments
 NATURAL JOIN locations
 WHERE  department_id IN (20, 50);
```

2- الربط الداخلي Inner Join .

وهنالك عدة أشكال من هذا الربط تستخدم عند الحاجة وهذه الأنواع هي:

أ- الربط الداخلي Inner Join باستخدام دالة الـ WHERE – CLAUSE

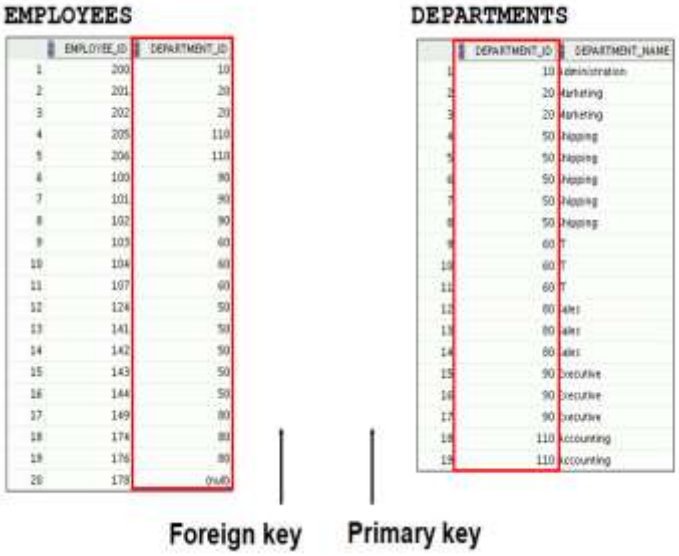
أذا كان هنالك عدة أعمدة تملك نفس الاسم ولكن نوعية البيانات غير متطابقة ، في تلك الحالة من الممكن استخدام الـ Where _ Clause لتحديد الأعمدة المطلوب ربطها في الجداول الموجودة ، مع الأخذ بنظر الاعتبار أن أسماء الجداول والأسماء المستعارة يجب عدم ذكرها في عبارة الـ WHERE – CLAUSE.

وكما موضح في الأمثلة التالية.

```
SQL> SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE location_id = 1400;
```

حيث أن الإيعاز أعلاه هو أيعاز صحيح وذلك لأننا ذكرنا اسم الحقل فقط في جملة Where ، أما الإيعاز التالي فهو أيعاز خاطئ وذلك لأننا ذكرنا اسم الجدول مع عبارة الـ Where.

```
SQL> SELECT   l.city, d.department _ name
FROM locations l JOIN departments d USING (location _ id)
WHERE d.location _ id = 1400;
```



حيث أننا في أعلاه لغرض أيجاد اسم القسم الخاص بالموظف ، سوف نقوم بمقارنة قيمة عمود الـ id – department الموجود في جدول الموظفين Employee ، مع قيمة عمود الـ department _ id الموجودة في جدول الـ department ، وذلك لان العلاقة التي تربط الجدولين أعلاه هي من خلال

عمود الـ department _ id . كما من الملاحظ ان هذا النوع كثيرا ما يتضمن primary key و Foreign key ، كما ان هذا النوع يدعى في بعض الأحيان بالربط الداخلي Inner Join او الربط البسيط Simple Join.

ب- الربط الداخلي للجداول باستخدام عبارة الـ Using .

من الممكن استخدام هذه العبارة كما موضح في المثال التالي:

```
SELECT employees.employee_id, employees.last_name,
       departments.location_id, department_id
FROM   employees JOIN departments
USING (department_id) ;
```

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	205	Higgins	1700	110
5	206	Gietz	1700	110
6	100	King	1700	90
7	101	Kochhar	1700	90
8	102	De Haan	1700	90
9	103	Hunold	1400	60
10	104	Ernst	1400	60

...

ت- الربط الداخلي للجداول باستخدام عبارة الـ ON .

تستخدم هذه الأداة لتحديد الشروط التحكمية أو تحديد الأعمدة المطلوب ربطها مع بعضها البعض في الجداول المختلفة، كما أن الشروط المخصصة لربط الجداول يجب ان تفصل عن شروط البحث . وان ميزة هذه الأداة هي أنها من الممكن ان تستخدم لربط الأعمدة التي تحتوي على أسماء مختلفة ، وان عبارة الـ ON تستخدم لتسهيل قراءة الـ Code وجعله قابل للفهم، والمثال التالي يوضح ذلك.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	205	Higgins	110	110	1700
5	206	Gietz	110	110	1700
6	100	King	90	90	1700
7	101	Kochhar	90	90	1700
8	102	De Haan	90	90	1700
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400

...

في المثال أعلاه سوف يقوم بربط الـ Employee _ id والـ last _ name والـ department _ id في جدول الـ employee مع الـ department _ id والـ location _ id في جدول الـ department عندما يكون قيمة الـ department _ id متساوية في كلا الجدولين.

ث- ربط الجدول بنفسه Itself Join .

من الممكن استخدام عبارة الـ ON لربط الجدول الواحد نفسه بنفسه ، ففي بعض الأحيان نحتاج لربط الجدول الواحد بنفسه.على سبيل المثال لإيجاد اسم مدير كل موظف نحتاج هنا لربط جدول الموظفين بنفسه . ولإيجاد اسم المدير الخاص بالموظفة Lorentz نحتاج للقيام بما يلي:

- ✓ أيجاد اسم الموظفة Lorentz عن طريق البحث عنها في حقل الـ Last _ name الموجود في جدول الموظفين.
- ✓ أيجاد رقم المدير الخاص بالموظفة Lorentz عن طريق البحث في حقل الـ Manager_id
- ✓ (حيث ان رقم المدير الخاص بالموظفة Lorentz هو 103).

✓ أيجاد اسم المدير المرقم (130) في الـ Employee _ id عن طريق البحث في حقل الـ Last _ name حيث سوف يكون ناتج البحث هو Hunold.

في العملية أعلاه قمنا بالبحث في الجدول بصورة ثنائية . ففي المرة الأولى قمنا بالبحث عن اسم الموظفة Lorentz في حقل الـ Last _ name ورقم المدير الخاص بها في حقل الـ Manager _ Id ، أما في المرة الثانية فقمنا بالبحث عن رقم المدير في حقل Employee _ Id واسمه في حقل الـ Last _ name وكما موضح في المثال التالي.

EMPLOYEES (WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

...

EMPLOYEES (MANAGER)

	EMPLOYEE_ID	LAST_NAME
1	100	King
2	101	Kochhar
3	102	De Haan
4	103	Hunold
5	104	Ernst
6	107	Lorentz
7	124	Mourgos
8	141	Rajs
9	142	Davies
10	143	Matos

...

Example:

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

	EMP	MGR
1	Abel	Zlotkey
2	Davies	Mourgos
3	De Haan	King
4	Ernst	Hunold
5	Fay	Hartstein
6	Gietz	Higgins
7	Grant	Zlotkey
8	Hartstein	King

...

ملاحظة مهمة:

من الممكن إضافة شرط معين إلى عملية ربط الجدول بنفسه باستخدام عبارة الـ ON وكما موضح في المثال أدناه.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149;
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	176	Taylor	80	80	2500

ج- الربط الخارجي Outer Join.

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst

...

عند أجراء عملية الربط الداخلي Inner Join بين جدولين أو أكثر ، أذا كان السطر المقروء حاليا لا يتوافق مع شرط الربط فان ذلك السطر سوف لن يظهر في ناتج الاستعلام. في المثال أعلاه أن الشرط الخاص بربط جدولي الموظفين Employees وجدول الأقسام Departments سوف يؤدي الى عدم ظهور القيد الذي يكون الـ department _ id له يساوي (190) ، وذلك لأنه ليس هنالك موظف يملك هذا الـ department _ id مسجل في جدول الموظفين وبالتالي بدلا من ان نرى (20 قيد) للموظفين سوف نرى فقط (19 قيد).

ولإرجاع قيود الأقسام التي لا تحتوي على أي موظفين فأننا نستطيع ذلك باستخدام ما يعرف بالربط الخارجي Outer Join.

ملاحظات مهمة:

- ✓ أن الربط الموجود بين جدولين والذي يقوم بإرجاع القيود المتطابقة فقط يدعى بالربط الداخلي.

- ✓ أن الربط الموجود بين جدولين والذي يقوم بإرجاع القيود المطابقة للشرط الموجود إضافة الى القيود الغير متطابقة في الجدول من جهة اليسار يدعى بالـ Left Outer Join.
- ✓ أن الربط الموجود بين جدولين والذي يقوم بإرجاع القيود المطابقة للشرط الموجود إضافة إلى القيود الغير مطابقة والموجودة في الجدول من جهة اليمين ، فان ذلك يدعى بالـ Right Outer Join.
- ✓ ان الربط الموجود بين جدولين والذي يقوم بإرجاع القيود المطابقة للشرط الموجود إضافة إلى القيود الغير مطابقة للشرط والموجودة في الجدول من كلا الجهتين (اليمين واليسار) ، فان ذلك يدعى بالـ Full Outer Join.

نستنتج من أعلاه بان هنالك ثلاثة أنواع من الـ Outer Join هي:

1- الربط الخارجي الايسر Left Outer Join.

2- الربط الخارجي الايمن Right Outer Join.

3- الربط الخارجي الكلي Full Outer Join.

أ. الربط الخارجي الأيسر Left Outer Join.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting
...			
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)

أن المثـال أعـلاه سـوف يعيـد كـل القيـود الموجـودة فـي جـدول المـوظفين Employee والمطابقة للشرط الموجود في عبارة الـ (ON) ، إضافة إلى القيود الموجودة في جدول الموظفين والتي تكون غير مطابقة للشرط الموجود في عبارة الـ (ON).

ب. الربط الخارجي الأيمنRight Outer Join.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	(null)	Contracting

أن هـذا النـوع سـوف يقـوم بإرجـاع جميـع القيـود الموجـودة فـي جـدول الأقسـام Departments، والتي لا تتطابق شروطها مع جدول الموظفين.

ت. الربط الخارجي الكلي Full Outer Join.

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing

...

18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

أن النوع أعلاه يقوم بإرجاع كل القيود المطابقة للشرط الموجود في جملة الـ (ON) ، إضافة إلى القيود الموجودة في كل من جدولي الموظفين Employee والأقسام Departments والتي تكون غير مطابقة للشرط الموجود.

3- الربط المتقاطع الـ Cross Join .

ويدعى أحيانا بالناتج الديناميكي Cartesian Product بين جدولين ، وكما موضح في المثال التالي:

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

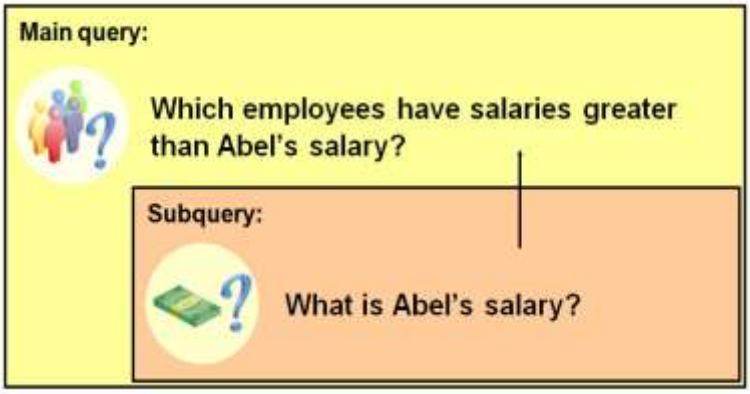
	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration

...

159	Whalen	Contracting
160	Zlotkey	Contracting

حيث أن المثال أعلاه يقوم بعملية الربط التكتيكي المتقاطع لكل من جدول الموظفين Employee Table وجدول الأقسام Departments Table. بمعنى أخر بأننا قمنا باختيار حقل معين من جدول الموظفين وهو حقل Last name _ وتم ربطه مع حقل معين من جدول الأقسام وهو حقل الـ Department _ name وربطهما بشكل متقاطع Cross Join . كما يجب أن نأخذ بنظر الاعتبار أننا من الممكن أن نختار أكثر من حقل واحد من كل جدول لربطها بشكل Cross Table.

الاستعلام الجزئي Sub Query.



ويدعى بالاستعلام الداخلي Inner Query ويكتب دائما داخل الاستعلام الخارجي ، أي إننا في تلك الحالة سوف نملك استعلام متداخل NestedQuery علما أن الاستعلام الداخلي سوف يتم تنفيذه دائما قبل الاستعلام الخارجي ، كما أن النتيجة التي سوف نحصل عليها من الاستعلام الداخلي سوف تستخدم في الاستعلام الخارجي.

على سبيل المثال في الشكل أعلاه لو أردنا أيجاد الموظفين الذين يتقاضون راتبا أعلى من راتب الموظف الذي يسمى Abels في تلك الحالة يجب معرفة مقدار الراتب الذي يتقاضاه Abels والذي سيمثل الاستعلام الداخلي ومن ثم مقارنته بالرواتب الباقية الموجودة حيث ان عملية المقارنة ستمثل الاستعلام الخارجي.

والصيغة العامة لتعريف الـ Sub Query هي.

```
SELECT   select_list
FROM     table
WHERE    expr operator
                  (SELECT     select_list
                  FROM        table);
```

وان المثال الموضح أعلاه سوف يكتب بالصيغة التالية:

```
SELECT last_name, salary
FROM   employees
WHERE  salary >
              (SELECT salary
               FROM   employees
               WHERE  last_name = 'Abel');
```

11000

	LAST_NAME	SALARY
1	Hartstein	13000
2	Higgins	12000
3	King	24000
4	Kochhar	17000
5	De Haan	17000

من الممكن استخدام العبارات التالية مع جملة الـ Sub Query:

1- عبارة الـ Where clauses.

2- عبارة الـ Having clauses.

3- عبارة الـ From clauses.

من الممكن استخدام عمليات الـ Single Row Operators مع الـ Sub Query ، وهذه العمليات هي:

(<> , =< , >= , = , < , >).

من الممكن استخدام عمليات الـ Multi Row Operator مع الـ Sub Query وهذه العمليات هي :

(IN, ANY, ALL).

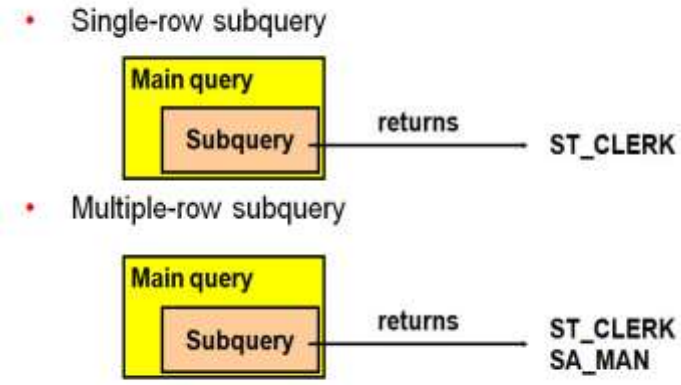
هنالك مجموعة من الشروط الواجب أتباعها عند كتابة الـ Sub Query وهذه الشروط هي:

1- يجب وضع الـ Sub Query داخل أقواس للتأكيد.

2- يجب وضع الـ Sub Query دائما في الجهة اليمنى من التعبير الشرطي.

3- هنالك نوعين من العمليات التي تستخدم مع الـ Sub Query هما الـ Single Row Operator والتي تستخدم مع الـ Single Row Sub query والـ Multiple Row Operator والتي تستخدم مع الـ multiple Row Sub Query .

أنواع الـ Sub queries.



هنالك نوعان من الـ Sub queries هما:

أ- Single Row Sub queries.

أن هذا النوع يقوم بإرجاع قيمة واحد فقط من جملة الـ Select الداخلية ، ومع هذا النوع من الممكن استخدام عمليات Single Row Operator مثل عمليات المقارنة التالية.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

مثال ذلك عندما نريد عرض أسماء الموظفين الذين يملكون job _ id مشابه للموظف الذي Employee _ id له هو (141) نقوم بكتابة الايعازات التالية.

```
SQL> SELECT last _ name, job _ id
FROM    employees
WHERE job _ id =
(SELECT job _ id
 FROM    employees
WHERE employee _ id = 141);
```

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

من الممكن جمع two sub queries داخل الـ main query عن طريق استخدام عبارة الـ AND وكما موضح في المثال التالي.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                    ST_CLERK
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >                    2600
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

	LAST_NAME	JOB_ID	SALARY
1	Rajs	ST_CLERK	3500
2	Davies	ST_CLERK	3100

ملاحظة :

أن الاستعلام الداخلي والخارجي من الممكن أن يحصل على البيانات من جداول مختلفة.

ملاحظة :

من الممكن استخدام دوال المجموعات التي تم شرحها سابقا مع الـ Sub query وكما موضح في المثال التالي.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

ملاحظة:

من الممكن استخدام عبارة الـ HAVING CLAUSES مع الـ SUBQUERY ، حيث أن الـ oracle server سوف يقوم بتنفيذ الـ sub query ويرجع النتيجة من خلال عبارة الـ Having الموجودة في الـ query main ليتم استخدام النتيجة في الـ main query وكما موضح في المثال التالي الذي يقوم بعرض جميع الأقسام التي تملك اقل salary اكبر من القسم (50).

```
SELECT   department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) >
                       (SELECT MIN(salary)
                        FROM   employees
                        WHERE  department_id = 50);
```

2500

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	20	6000
3	90	17000
4	110	8300
5	80	8600
6	10	4400
7	60	4200

ب- Multiple Row Sub Query.

أن هذا النوع يقوم بإرجاع أكثر من سطر واحد ويستخدم دائما الـ Multiple Row Operator والموضحة في الجدول التالي:

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

1- استخدام عملية الـ IN مع الـ Multiple Row Operator.

المثال التالي يقوم بإيجاد الموظف الذي يكسب Salary الذي يمثل اقل Salary لكل قسم باستخدام عبارة الـ IN.

```
SQL> SELECT last_name, salary, department_id
FROM   employees
WHERE salary IN (SELECT   MIN (salary)
FROM employees
GROUP BY department_id);
```

2- استخدام عملية الـ ANY مع الـ Multiple Row Operator.

تقوم هذه العبارة بمقارنة القيمة المعطاة مع كل قيمة يتم إرجاعها من قبل الـ Sub Query ، والمثال التالي يقوم بعرض جميع الموظفين الذين لا ينتمون إلى قسم الـ IT Programmers ويتقاضون راتبا اقل من الراتب الذي يتقاضاه الـ IT Programmer .

بحيث أن أعلى راتب يتقاضاه الـ ITProgrammer هو (9000 $) ، علما أن الـ ANY تعني اقل من الـ Maximum واعلي من الـ Minimum .

```
SELECT employee_id, last_name, job_id, salary
FROM   employees        9000, 6000, 4200
WHERE  salary < ANY
                      (SELECT salary
                       FROM   employees
                       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

3- استخدام عملية الـ ALL مع الـ Multiple Row Operator.

تقوم هذه العبارة بمقارنة القيمة المعطاة مع كل قيمة يتم إرجاعها من قبل الـ Sub Query ، والتي تعني اكبر من الـ Maximum واقل من الـ Minimum وكما موضح في المثال التالي الذي يقوم بعرض الموظفين الذين يتقاضون راتبا اقل من الراتب الذي يتقاضاه الـ IT Programmer وهم ليسوا IT Programmer .

```
SELECT employee_id, last_name, job_id, salary
FROM   employees        9000, 6000, 4200
WHERE  salary < ALL
                      (SELECT salary
                       FROM   employees
                       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

ملاحظة مهمة:

من الممكن استخدام عبارة الـ NOT مع كل من العمليات السابقة (IN , ANY , ALL).

عمليات المجموعة SET OPERATORS.

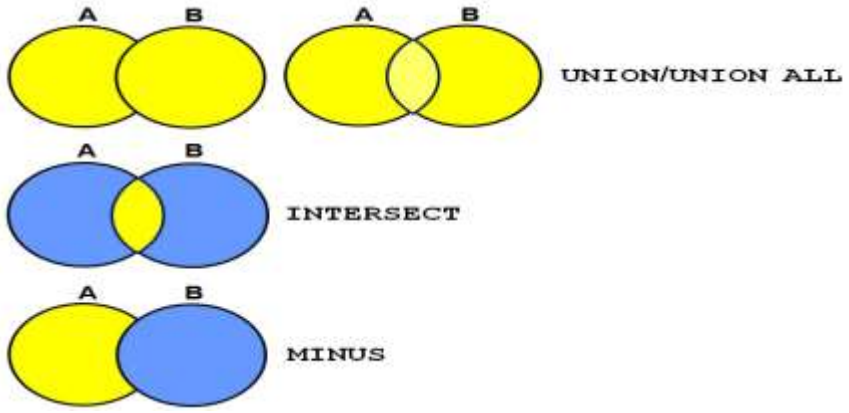
هنالك أربعة عمليات أساسية من الممكن أجرائها على المجموعات، وهذه العمليات هي.

1- الاتحاد UNION.

2- اتحاد الكل UNION ALL.

3- التقاطع INTERSECT.

4- الطرح MINUS.



1- عملية الاتحاد UNION OPERATOR.

أن هذه العملية تقوم بإرجاع جميع العناصر الموجودة في الـ Queries بعد حذف العناصر المكررة منها، والمثال التالي يقوم بعرض تفاصيل الوظيفة الحالية

والسابقة لجميع الموظفين مع العلم انه يقوم بعرض تفاصيل كل مجموعة (موظف) مرة واحدة.

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	100	AD_PRES
2	101	AC_ACCOUNT
...		
22	200	AC_ACCOUNT
23	200	AD_ASST
...		
28	206	AC_ACCOUNT

2- عملية الاتحاد الكلي UNNION ALL OPERATOR.

أن هذه العملية تقوم بإرجاع جميع العناصر الموجودة في الـ Two Queries بدون حذف العناصر المكررة منهما، والمثال التالي يقوم بعرض الأقسام السابقة والحالية لجميع الموظفين.

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM   job_history
ORDER BY  employee_id;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
2	101	AD_VP	90
...			
23	200	AD_ASST	10
24	200	AC_ACCOUNT	90
25	200	AD_ASST	90
...			
30	206	AC_ACCOUNT	110

3- عملية التقاطع INTERSECT OPERATOR .

أن عملية التقاطع تقوم بإرجاع جميع العناصر المشتركة في كلا الـ Sub Queries ، والمثال التالي يقوم بعرض الموظفين الذين يملكون عنوان وظيفي مشابه للعنوان الوظيفي السابق.

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

4- عملية الطرح MINUS OPERATOR .

أن وظيفة هذه العملية هو إرجاع القيود الموجودة في الاستعلام الأول والغير موجودة في الاستعلام الثاني، والمثال التالي يقوم بعرض الموظفين الذين لم يغيروا عناوينهم الوظيفية ولا مرة.

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
...	
14	205
15	206

مثلما تعلمنا سابقا أن عناصر قاعدة البيانات هي خمسة عناصر أساسية مبينة في أدناه.

أ- Tables.

ب- View.

ت- Sequence.

ث- Index.

ج- Synonym.

لقد كان عملنا في الدروس السابقة يتركز على الجداول ، فكل الدوال والعبارات والطرق والأمثلة التي تم دراستها كان عملها يتركز فقط على الجداول ، أماالآن فسوف نتعلم كيفية خلق العناصر الباقية والتعامل معها.

.VIEW

هو عبارة عن جدول منطقي Logical Table يمثل مجموعة جزئية تم خلقها من جدول أخرأو View أخر ، وهذا العنصر لا يحتوي على بيانات خاصة به وإنما هو عبارة عن نافذة يتم رؤية ومعالجة البيانات من خلالها ، وان الجداول التي تبنى على أساسها الـ View تدعى بالـ base table.

EMPLOYEES table

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800
8	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500
9	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	3100
10	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
11	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500
12	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	10500
13	174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	11000
14	176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	8600
15	178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000
16	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400
17	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000
18	202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
20	206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

EMPLOYEE_ID	LAST_NAME	SALARY
104	Ernst	6000
107	Lorentz	4200
124	Mourgos	5800

مزايا استخدام الـ View.

هنالك أربعة مزايا رئيسية لاستخدام الـ View وهذه المزايا هي.

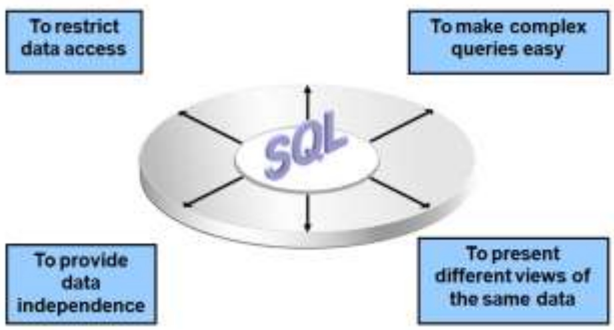
i. تقييد الوصول إلى البيانات بصورة مباشرة، وذلك لأنه يقوم بعرض الحقول المختارة فقط من الجداول.

ii. منع استقلالية البيانات عن المستخدم والبرامج التطبيقية، حيث أن الـ View الواحد يستخدم لاستعادة البيانات من عدة جداول.

iii. لإدارة الاستعلامات المعقدة بصورة سهلة، حيث أن الـ View يستخدم لعمل استعلام بسيط يستخدم لاستعادة البيانات من الاستعلامات المعقدة، على سبيل المثال أن الـ View يستخدم لبناء استعلام من عدة جداول بدون

iv. أن يعرف المستخدم كيفية كتابة جمل الربط Join Statement.

ت- تقديم Views مختلفة لنفس البيانات.



أنواع الـ View.

هنالك نوعان أساسيان للـ View هما:

1- Simple View.

2- Complex View.

أن الفرق بين النوعين أعلاه يكمن في استخدام عبارات الـ (DML) مثل ايعازات الـ (Insert , Update , Delete)، وان كل منهما يحتوي على خصائص معينة موضحة في أدناه.

هنالك ثلاثة خصائص أساسية للـ Simple View هي:

1- يشتق البيانات من جدول واحد فقط.

2- لا يحتوي على دوال أو مجاميع للبيانات.

3- يستطيع تنفيذ عمليات الـ DML من خلال الـ View.

خصائص الـ Complex View:

1- يشتق البيانات من عدة جداول.

2- يحتوي على مجموعة من الدوال ومجاميع للبيانات.

3- ليس في كل الأحوال يسمح لعمليات الـ DML بالتنفيذ من خلال الـ View.

كيفية خلق الـ VIEW.

أن الصيغة العامة لخلق أي VIEW هي.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

علما أن الـ Sub query من الممكن أن يحتوي على عبارة Select بسيطه او مركبة. حيث أن الصيغة أعلاه تبين لنا أننا نستطيع خلق VIEW من خلال

تضمين استعلام جزئي Sub query في عبارة الـ Select ، مع أمكانية استخدام العبارات التالية:

أ- OR REPLACE: نستطيع من خلالها أعادة خلق VIEW من واحد موجود سابقا.

ب-FORCE: نستطيع من خلالها خلق VIEW مهمل سواء أكان الجدول الأساسي موجود أم لا.

ت-NOFORCE: نستطيع من خلالها خلق VIEW فقط أذا كان الجدول الأساسي موجود.

ث-VIEW: تستخدم لخلق VIEW بالطريقة الاعتيادية.

ج- Alias: نستطيع من خلالها تحديد أسماء للتعابير المختارة من قبل استعلام الـ VIEW.

ح- Sub Query: نستطيع من خلالها بناء جملة Select متكاملة.

خ- With check Option: نستطيع من خلالها تحديد فقط القيود القابلة للوصول في الـ view من أضافتهاأو تعديلها.

د- CONSTRAINT: نستطيع من خلالها بناء شروط معينه للوصول إلى القيود في الـ view.

ذ- With Read Only: نستطيع من خلالها منع ايعازات الـ DML من العمل على الـ VIEW.

1- خلق VIEW بالطريقة الاعتيادية باستخدام أيعاز CREATE VIEW.

المثال التالي يقوم بخلق VIEW اسمه EMPVE80 خاص بجدول الموظفين يحتوي على تفاصيل الموظفين في القسم رقم (80).

```
SQL> CREATE VIEW empve80
        AS SELECT employee_id, last_name, salary
        FROM employees
        WHERE department_id = 80;
        VIEW CREATED SUCESSFULLY.
```

وللتأكد من أن هذا الـ VIEW قد تم خلقه أم لا نستخدم أيعاز DESCRIBE.

```
SQL> DESCRIBE empve80;
```

2- خلق VIEW باستخدام أسماء مستعارة للأعمدة في الـ Sub query.

```
CREATE VIEW  salvu50
 AS SELECT  employee_id ID_NUMBER, last_name NAME,
            salary*12 ANN_SALARY
    FROM    employees
    WHERE   department_id = 50;
CREATE VIEW succeeded.
```

ملاحظة :

من الممكن استعادة البيانات من الـ VIEW، كما في استعادة البيانات من اي جدول وذلك بواسطة استخدام عبارة الـ SELECT .

وكما موضح في المثال التالي.

```
SQL> SELECT * FROM salvu50;
```

ملاحظة:

عندما نريد استعادة البيانات من هذا الـ VIEW يجب أعطاء الاسم المستعار للعمود، وكما موضح في المثال التالي

```
SQL> SELECT ID_NUMBER, NAME, ANN_SALARY FROM
salvu50;
```

ملاحظة:

من الممكن عرض هيكلية الـ view الذي تم خلقه بواسطة استخدام أيعاز describe وكما موضح في المثال التالي.

```
SQL> DESCRIBE salvu50;
```

```
Name                          Null     Type
----------------------------- -------- -------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
LAST_NAME                     NOT NULL VARCHAR2(25)
SALARY                                 NUMBER(8,2)

3 rows selected
```

ملاحظة:

وبعد اسم الـ CREATE من الممكن وضع الأسماء المستعارة ضمن جملة الـ VIEW مباشرة ، والمثال التالي يوضح ذلك.

```
SQL> CREATE OR REPLACE VIEW salvu50 (ID_NUMBER,
        NAME, ANN_SALARY)
        AS SELECT employee_id, last_name, salary*12
        FROM  employees.
 WHERE   department_id = 50;
```

تعديل الـ VIEW.

من الممكن تعديل الـ VIEW موجود مسبقا بواسطة استخدام عبارة CREATE OR REPLACE بدون حذفه وإعادة خلقه من جديد علما انه عندما يتم ذكر الأسماء المستعارة للأعمدة يجب في تلك الحالة كتابة ترتيب تلك الأعمدة حسب التسلسل الموجود في الـ view الأصلي، وكما موضح في المثال التالي.

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
           || last_name, salary, department_id
   FROM    employees
   WHERE   department_id = 80;
CREATE VIEW succeeded.
```

خلق view معقد.

من الممكن خلق view معقد يحتوي على مجموعة من الدوال لعرض القيم المطلوبة والتي يتم اختيارها من جدولين أوأكثر ، والمثال التالي يقوم بخلق view من جدولين هما جدول الموظفين وجدول الأقسام ، حيث سوف يتم أظهار اسم القسم واقل راتب في القسم وأعلى راتب في القسم بالإضافة إلى معدل الرواتب في القسم وكما موضح أدناه.

```
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT   d.department_name, MIN(e.salary),
            MAX(e.salary),AVG(e.salary)
   FROM     employees e JOIN departments d
   ON       (e.department_id = d.department_id)
   GROUP BY d.department_name;
CREATE VIEW succeeded.
```

ولعرض نتائج هذا الاستعلام نستخدم الإيعاز التالي:

SQL> select * from dept _ sum _ vu;

	NAME	MINSAL	MAXSAL	AVGSAL
1	Administration	4400	4400	4400
2	Accounting	8300	12000	10150
3	Purchasing	4200	9000	6400
4	Executive	17000	24000	19333.33333333333333333333333333333333
5	Shipping	2500	5800	3500
6	Sales	8600	11000	10033.33333333333333333333333333333333
7	Marketing	6000	13000	9500

استخدام عبارة WITH READ ONLY.

من الممكن تقييد الـ VIEW وجعله للقراءة فقط وعدم السماح بمسح أي قيد منه وذلك من خلال استخدام عبارة WITH READ ONLY وكما هو موضح في المثال التالي.

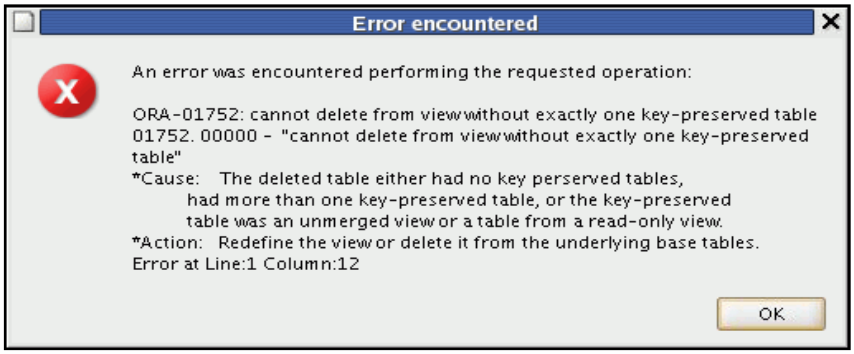
```
CREATE OR REPLACE VIEW empvu10
    (employee_number, employee_name, job_title)
AS SELECT    employee_id, last_name, job_id
   FROM     employees
   WHERE    department_id = 10
   WITH READ ONLY ;
CREATE VIEW succeeded.
```

وعند استخدام الإيعاز التالي :

SQL> DELETE FROM empvu10

WHERE employee _ number = 200;

سوف تظهر لنا العبارة التالية.



أي إن أي محاولة لإضافة أو تعديل أو حذف قيد معين سوف لن يسمح لنا بها وذلك لأننا حددنا ذلك الـ VIEW هو للقراءة فقط.

حذف الـ VIEW.

من الممكن حذف view تم خلقه سابقا وذلك بواسطة استخدام أيعاز DROP ، علما أن عملية الحذف سوف لن تؤثر على الجدول الذي بني الـ VIEW على أساسه ، وكما موضح في المثال التالي.

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
DROP VIEW empvu80 succeeded.
```

الـ Sequence.

هو احد العناصر الأساسية لقاعدة البيانات الذي يتم خلقه من قبل المستخدم ، والذي من الممكن مشاركة استخدامه فيما بعد من قبل بقية المستخدمين لتوليد القيم الصحيحة IntegerValue ، حيث أن الاستخدام النموذجي للـ Sequences هو في خلق الـ Primary Key والتي يجب أن تكون

أحادية لكل سطر، وبعد عملية بناء الـ Sequence فان قيمه سوف تزداد او تنقص بصورة أوتوماتيكية بواسطة روتين تابع إلى برنامج الاوراكل حيث يمتاز هذا العنصر بالمميزات التالية:

1- يستطيع الـ Sequence توليد الأعدادالأحاديةUnique numbers بصورة أوتوماتيكية.

2- عنصر قابل للتشارك.

3- من الممكن استخدامه لخلق قيم الـ Primary Key.

4- يستطيع أعادة code التطبيق.

5- يستطيع تسريع كفاءة الوصول للعناصر المتسلسلة الموجودة في الـ Cash Memory.

أن قيم الـ Sequence تولد وتخزن اعتمادا على الجداول، لذلك فان نفس الـ Sequence من الممكن استخدامه لعدة جداول.

والصيغة العامة لخلق أي Sequence هي كما يلي:

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE}]
       [{CACHE n | NOCACHE}];
```

بحيث أن:

- Sequence: يمثل اسم الـ Sequence الذي يتم خلقه.
- Increment By n: تمثل مقدار الزيادة في قيمة الـ Sequenceبصورة أوتوماتيكية ي كل خطوه ، وإذا تم إهمال هذه القيمة فان مقدار الزيادة سوف يكون بمقدار واحد بصورة أوتوماتيكية.
- STARTWITHn: يمثل مقدار القيمة التي سوف يبدأ بها الـ Sequence وعند إهمال هذه القيمة وعدم كتابتها فان الـ Sequence سوف يبدأ بصورة أوتوماتيكية من (1).
- MAXVALUEn: تمثل أعلى قيمة من الممكن أن يصل إليها الـ Sequenceويجب في كل حال من الأحوال ذكرها عند خلق أي Sequence وذلك لأنه بدون ذكرها سوف يدخل الـ Sequence في Infinite Loop.
- NOMAXVALUE: من الممكن من خلالها تحديد أعلى قيمة يستطيع الـ Sequenceمن الوصول إليها وهي في القيم التصاعدية 27 ^ 10 وللقيم التنازلية (1 -) .
- MINVALUEn: تمثل اقل قيمة من الممكن أن يصلها الـ .Sequence
- NOMINVALUE: نستطيع من خلالها تحديد اقل قيمة من الممكن أن يبدأ بها الـ Sequenceوهي (1) للترتيب التصاعدي ، و (10^27) - للترتيب التنازلي، حيث ان هذه القيم تمثل القيم الافتراضية.

مثال :

المثال التالي يقوم بخلق Sequence اسمه seq _ deptid _ dept وظيفته خلق مجموعة من الـ Primary Keys خاصة بجدول الموظفين Employee table والذي سوف يستخدم فيما بعد في جدول الأقسام department table.

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
CREATE SEQUENCE succeeded.
```

بحيث أن :

- NOCYCLE / CYCLE: تحدد هذه الخاصية ما أذا كان الـ Sequence يستمر في توليد القيم في حالة وصوله إلى الـ Minimum value أو وصوله إلى الـ Maximum value وان الـ NOCYCLE هي القيمة التلقائية لتلك الحالة.
- NOCACHE / CACHE: تحدد هذه الخاصية عدد القيم التي سوف يوزعها الـ Serverالخاص ببرنامج الاوراكل بصورة أولية ويحتفظ بها في الذاكرة، وبصورة تلقائية فان برنامج الـ Oracle Serverسوف يخبأ (20) قيمة.

في المثال أعلاه سوف نولد Sequence اسمه seq _ deptid _ dept والذي سوف نستخدمه فيما بعد في حقل الـ id _ department الموجود في جدول الـ Department ، حيث أن ها الـ Sequence سوف يبدأ بالـ (120) ويزداد

بمقدار (10) في كل مرة إلىأن يصل الى (9999) ، مع إيقاف خاصية الـ Cache من خلال الـ NOCACHEوإيقاف خاصية الـ CYCLE من خلال الـ NOCYCLE .

ملاحظـة : لا يمكن استخدام خاصية الـ CYCLE في حالة استخدام الـ Sequence لتوليد قيم الـ Primary Key ، بحيث انه على الأقل يجب أن يملك تقنية موثوقه .

ملاحظـة: عند توليد الـ Sequence لا يجب ربطه مع الجدول ، بحيث انه يجب ان نقوم بتسميته بعد أكمال الاستخدام الخاص به ، وفي كل حال من الأحوال من الممكن استخدام الـ Sequence في أي مكان بغض النظر عن اسمه.

- ❖ أن الـ NEXTVAL تقوم بإرجاع القيمة المتاحة التالية الموجودة في الـ Sequence ، بحيث أنها تقوم بإرجاع قيمة أحادية في كل وقت .
- ❖ الـ CURRVAL تقوم بإرجاع القيمة الحالية في الـ Sequence .

كيفية استخدام الـ Sequence :

في المثال التالي سوف نقوم بإضافة قسم جديد اسمه "Support" الـ Location ID له (2500).

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
1 row created.
```

ولعرض القيمة الحالية للـ Sequence dept _ deptid _ seq نقوم بكتابة الايعاز التالي.

```
SELECT   dept_deptid_seq.CURRVAL
FROM     dual;
```

كيفية تحديث الـ Sequence :

من الممكن تحديث أي Sequence تم خلقه سابقا من خلال تعديل القيمة الابتدائية او القيمة النهائية او مقدار التغيير في قيمة العداد ..... الخ، وذلك من خلال استخدام أيعاز Alter. وكما موضح في المثال التالي.

```
ALTER SEQUENCE dept_deptid_seq
               INCREMENT BY 20
               MAXVALUE 999999
               NOCACHE
               NOCYCLE;
ALTER SEQUENCE dept_deptid_seq succeeded.
```

الفهرس الـ Index:

هو أحدى العناصر الأساسية لقاعدة البيانات والذي يستخدم لتحسين الأداء لبعض الـ Queries ، حيث ان الوظيفة الأساسية له هي القيام بعملية الوصول السريع والمباشر للقيود في قاعدة البيانات. كما من الممكن خلق الـ Index بصورة أوتوماتيكية من خلال الـ Server عند خلق الـ Primary Key أو الـ Unique Key .

أما بالنسبة إلى فوائد الـ Index فهي كما يلي.

1- يستخدم لتسريع استعادة القيود بواسطة استخدام الـ Pointer.

2- يستخدم لاختصار عمليات الإدخال والإخراج للقرص من خلال تسريع طرق الوصول للمسارات.

3- يتم استخدامه وصيانته أوتوماتيكيا بواسطة الـ Oracle Server.

أن الـ Index تعمل بصورة مستقلة عن الجدول ، وهذا يعني أننا يمكننا خلقها أو حذفها في أي وقت وبدون التأثير على الجدول الأساس أو أي Index أخر. لكن بالمقابل عند حذف أي جدول فان ذلك سوف يؤدي إلى حذف الـ Index التابع له.

كيفية تعريف الـ Index.

هنالك طريقتان أساسيتان لتعريف الـ Index هما:

1- **أوتوماتيكيا** : وذلك من خلال الـ Oracle Server ، حيث يتم خلقه عند تعريف الـ Primary Key أو الـ Unique Key عند تعريف الجدول وتحديد الـ Constraint وان اسم الـ Index هنا هو نفس الاسم الذي سوف يعطى للـ Constraint.

2- **يدويا** : وذلك من خلال المبرمج ، حيث انه يستطيع خلق Index معين وذلك لتسريع الوصول إلى القيود في الجدول.

كيفية خلق الـ Index.

إن الصيغة العامة لخلق أي Index مبينه كما يلي :

```
CREATE INDEX index
ON table (column[, column]...);
```

ولتحسين وتسريع عملية الوصول الى العمود المسمى Last _ name في جدول الموظفين سوف نقوم بتعريف الـ Index التالي.

```
CREATE INDEX emp_last_name_idx
ON           employees(last_name);
CREATE INDEX succeeded.
```

نستطيع خلق Index في الحالات التالية:

1- في حالة كون العمود يحتوي على مديات عريضة من القيم.

2- في حالة كون العمود يحتوي على أعداد كبيرة من الـ Null Value.

3- إذا كان العمود يستعمل كثيرا مع الـ Where Clauses أو مع الـ Join Conditions.

4- إذا كان الجدول كبير ومعظم الـ Queries الموجودة يتوقع أن ترجع اقل من (2% - 4%) من القيود.

لا نستطيع خلق الـ Index في الحالات التالية:

1- في حالة كون العمود لا يستعمل بكثرة مع الـ Condition Query .

2- إذا كان الجدول صغير أو معظم الـ Queries يتوقع لها أن ترجع أكثر من (2% - 4%) من القيود.

3- إذا كان الجدول يتم تحديثه باستمرار.

4- إذا كان عمود الـ Index يشير إلى جزء من تعبير.

حذف الـ Index .

من الممكن حذف الـ Index من خلال استخدام أيعاز Drop ، حيث أن الـ User يجب أن يمتلك السماحية للحذف لغرض القيام بتلك العملية ، وان الصيغة العامة لحذف أي Index هي.

```
DROP INDEX index;
```

على سبيل المثال عندما نريد حذف الـ Index المسمى UPPER _ LAST _ NAME _ IDX . فان ذلك يتم حسب المثال التالي.

```
DROP INDEX emp_last_name_idx;
DROP INDEX emp_last_name_idx succeeded.
```

ملاحظة :

عند حذف أي جدول فان الـ Index الخاص به سوف يحذف بصورة أوتوماتيكية.

الـ Synonyms.

هي عبارة عن احد العناصر الأساسية لقاعدة البيانات التي نستطيع من خلالها استدعاء الجدول باسم أخر، أي أننا نستطيع من خلالها أعطاء اسم أخر للجدول. حيث أن الغرض من ذلك هو لتبسيط الوصول إلى تلك الجداول ، بحيث اننا نستطيع من خلالها :

1- أعطاء عناوين أسهل للجداول المملوكة من قبل مستخدم أخر.

2- أعطاء أسماء اقصر للعناصر الموجودة في قاعدة البيانات.

وان الصيغة العامة لخلق اي Synonym موضحة كما يلي:

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

المثال التالي يقوم بخلق Synonym اسمه d _ sum للعمود المسمى dept _ sum _ vu .

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
CREATE SYNONYM succeeded.
```

ملاحظة :

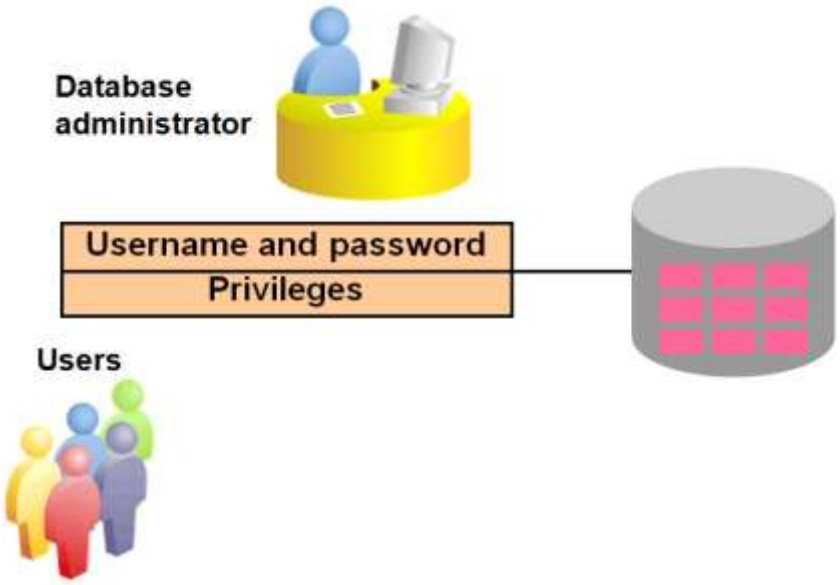
أن الفرق بين الـ Synonym من نوع Public Synonym والنوع العادي هو أن الـ Public Synonym يستطيع الوصول إلى جميع المستخدمين Users لذلك فان هذا النوع لا يمكن خلقه أو حذفه إلا من خلال الـ database administer أما الـ Synonym العادي فانه يستطيع الوصول إلى الـ User الخاص به فقط ، لذلك فمن الممكن خلق هذا النوع من خلال أي مستخدم.

حذف الـ Synonym :

من الممكن حذف Synonym معين من خلال استخدام أيعاز Drop وكما موضح في المثال التالي ، علما أن الـ Public Synonym لا يمكن حذفه إلا من خلال الـ database administer .

```
DROP SYNONYM d_sum;
DROP SYNONYM succeeded.
```

السيطرة على وصول المستخدم Controlling User Access.



الامتيازات Privileges .

هي عبارة عن مجموعة من الخصائص والصفات والأعمال التي يتم منحها للمستخدم User لغرض القيام بعمله تجاه قاعدة البيانات ، وتختلف هذه الامتيازات من User إلى أخر وحسب طبيعة العمل الذي يقوم به وتسلسله ضمن فريق العمل. هنالك أربعة أنواع رئيسية من تلك الامتيازات هي:

1- أمنية قاعدة البيانات Database Security وتشمل:

- ✓ أمنية النظام System Security.

- ✓ أمنية البيانات Data Security.

2- امتيازات النظام System Privileges : عند اكتسابها يتمكن المستخدم من الوصول إلى قاعدة البيانات.

3- امتيازات العناصر Object Privileges : عند اكتسابها يتمكن المستخدم من معالجة محتويات عناصر قاعدة البيانات.

4- الوحدات Schemas : هي عبارة عن مجموعة من العناصر التي تكون قاعدة البيانات مثل الـ Tables ، Views ، Sequences.

امتيازات النظام System Privileges .

هنالك أكثر من (100) خاصية من الممكن أعطائها للمستخدم لغرض التعامل مع النظام ، وهنالك نوعان من المتعاملين مع النظام الأول يدعى بالمستخدم العادي Simple User والثاني يدعى بمدير قاعدة البيانات Database Administrator، الذي يملك كل الخصائص والمميزات التي تمكنه من التعامل مع النظام مثل:

- خلق مستخدم جديد Create New User.
- حذف مستخدم موجود Removing User.
- حذف الجداول Removing Tables.
- عمل النسخ الاحتياطية للجداول Backing up Tables.

خلق مستخدم جديد Creating New User.

وهذا الامتياز يتم منحه فقط للـ database Administrator ، حيث أن له الحق في خلق مستخدم جديد يستطيع التعامل مع قاعدة البيانات من خلال

أعطائه User name و Password خاص به لغرض الدخول الى قاعدة البيانات والتعامل معها . وان الصيغة العامة لخلق مستخدم جديد هي.

```
CREATE USER user
IDENTIFIED BY   password;
```

```
CREATE USER  USER1
IDENTIFIED BY   USER1;
CREATE USER succeeded.
```

أعطاء الامتيازات للمستخدم User System Privilege:

بعد خلق مستخدم جديد يجب إعطائه مجموعة من الامتيازات التي تشمل الأعمال الممكن القيام بها على قاعدة البيانات ، والتي بدونها لا يستطيع العمل على تلك القاعدة . ومن الممكن أن تشمل تلك الأعمال:

1- CREATE SESSION.

2- CREATE TABLE.

3- CREATE SEQUENCE.

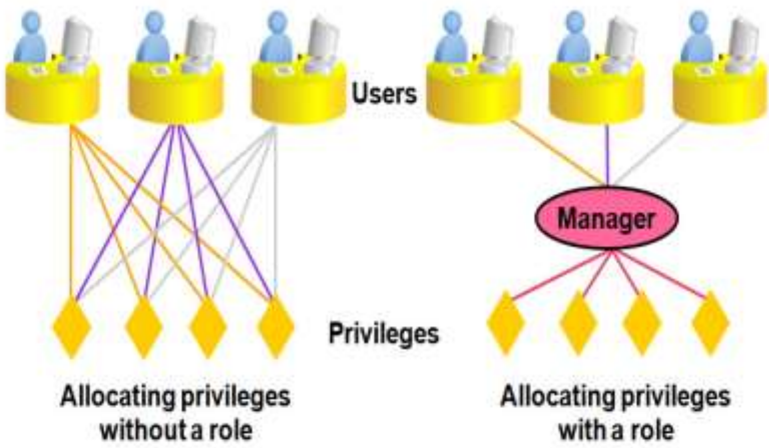
4- CREATE VIEW.

5- CREATE PROCEDURE.

أن الشخص الوحيد الذي يستطيع أعطاء تلك الامتيازات للـ User هو الـ DBA وان الصيغة العامة لها موضحة في المثال التالي.

```
GRANT  create session, create table,
       create sequence, create view
TO     scott;
GRANT CREATE succeeded.
```

القواعد Roles.



تمثل الـ Role عملية خلق قالب جديد يتم فيه تعيين مجموعة من الامتيازات ومن ثم أعطاء ذلك القالب لمجموعة من المستخدمين ، وتعتبر هذه الطريقة من التقنيات الواسعة الاستخدام وذلك لأنها تؤدي إلى توفير الوقت والجهد في توزيع الامتيازات على المستخدمين الموجودين ، وان الصيغة العامة لهذه الطريقة مبينة في الأمثلة التالية:

- Create a role:

```
CREATE ROLE manager;
CREATE ROLE succeeded.
```

- Grant privileges to a role:

```
GRANT create table, create view
TO manager;
GRANT succeeded.
```

- Grant a role to users:

```
GRANT manager TO BELL, KOCHHAR;
GRANT succeeded.
```

تغيير كلمة السر Password Changing .

بعد خلق مستخدم جديد من قبل الـ DBA وإعطائه كلمة السر الخاصة به ، فمن الممكن تغيير كلمة السر بواسطة استخدام أيعاز ALTER USER وكما موضح في المثال التالي.

```
ALTER USER HR
IDENTIFIED BY employ;
ALTER USER HR succeeded.
```

الجدول التالي يمثل العناصر الأساسية لقاعدة البيانات والأعمال التي من الممكن القيام بها على تلك العناصر لغرض منحها إلى الـ User .

Object Privilege	Table	View	Sequence	Procedure
ALTER	√		√	
DELETE	√	√		
EXECUTE				√
INDEX	√			
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

امتيازات العناصر Object Privileges.

أن الامتيازات الخاصة بالعناصر تتباين من عنصر إلى أخر ، وان الـ DBA يملك جميع الامتيازات الخاصة بالعناصر ويستطيع أعطائها إلى أي مستخدم أخر والى أي role أخر ، وان الصيغة العامة لإعطاء العناصر Objects لأي مستخدم موضحة في أدناه.

```
GRANT       object_priv [(columns)]
ON          object
TO          {user|role|PUBLIC}
[WITH GRANT OPTION];
```

أن المثال التالي يقوم بإعطاء امتياز الاستعلام select الخاص بجدول الموظفين لمستخدمين هما (sue , rich).

```
GRANT  select
ON     employees
TO     sue, rich;
GRANT succeeded.
```

والمثال التالي يعطي امتياز الـ Update الخاص بجدول الأقسام departments إلى two roles وليس مستخدمين هما (Scott , manager) مع تحديد الأعمدة التي تجري عليها عملية الـ Update وهي (Department _ name , location _ id).

```
GRANT  update (department_name, location_id)
ON     departments
TO     scott, manager;
GRANT succeeded.
```

كيفية منح السلطة إلى مستخدم أخر.

من الممكن منح السلطة إلى مستخدم أخر وجعله يعمل كـ DBA ، وان ذلك لا يمكن أن يتم إلا من خلال DBA معرف مسبقا . وان المثال التالي يوضح كيفية منح السلطة الى DBA .

```
GRANT  select, insert
ON     departments
TO     scott
WITH   GRANT OPTION;
GRANT succeeded.
```

أما المثال التالي فيبين كيفية أعطاء امتياز الـ Select الخاص بجدول الأقسام department إلى جميع المستخدمين الذين يعملون على النظام.

```
GRANT  select
ON     alice.departments
TO     PUBLIC;
GRANT succeeded.
```

الجدول التالي يوضح كيفية تأكيد الخصائص الممنوحة إلى مستخدم معين.

Data Dictionary View	Description
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
USER_ROLE_PRIVS	Roles accessible by the user
USER_TAB_PRIVS_MADE	Object privileges granted on the user's objects
USER_TAB_PRIVS_RECD	Object privileges granted to the user
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user's objects
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns
USER_SYS_PRIVS	System privileges granted to the user

إلغاء الامتيازات Privilege Revoke.

من الممكن إلغاء الامتيازات التي تم منحها الى مستخدم معين في وقت سابق وذلك من خلال استخدام أيعاز REVOKE ، وان الصيغة العامة التالية توضح كيفية استخدام ذلك الإيعاز.

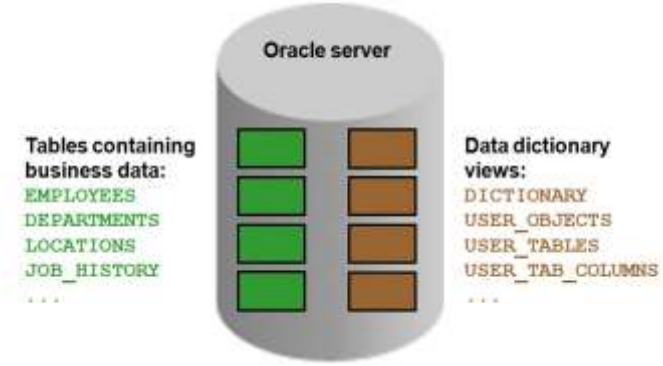
```
REVOKE {privilege [, privilege...]|ALL}
ON     object
FROM   {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

```
REVOKE  select, insert
ON      departments
FROM    scott;
REVOKE succeeded.
```

من الممكن تلخيص كيفية خلق مستخدم جديد وإعطاءه الامتيازات أو تعديلها او إلغائها من خلال الجدول التالي.

Statement	Action
CREATE USER	Creates a user (usually performed by a DBA)
GRANT	Gives other users privileges to access the objects
CREATE ROLE	Creates a collection of privileges (usually performed by a DBA)
ALTER USER	Changes a user's password
REVOKE	Removes privileges on an object from users

قاموس البيانات Data Dictionary .



هي عبارة عن مجموعة من الجداول Tables والـ Views الموجودة في قاعدة بيانات اوراكل والتي يتم خلقها وصيانتها من قبل Oracle Server ، وتحتوي على معلومات مهمة عن قاعدة البيانات مثل أسماء الجداول الموجودة وإحجام الأعمدة وأنواع البيانات التي تحتويها ، ويتم خزن البيانات الخاصة بالـ Data Dictionary على شكل Tables & Views .

علما ان تلك البيانات الخاصة بالـ Data Dictionary هي بيانات للقراءة فقط.

وتشمل تلك البيانات التي تحتويها على :

1- تعريف كل وحدات العناصر التابعة الى قاعدة البيانات مثل الـ (Tables – Views – Sequences – ….etc) .

2- القيم التلقائية للأعمدة.

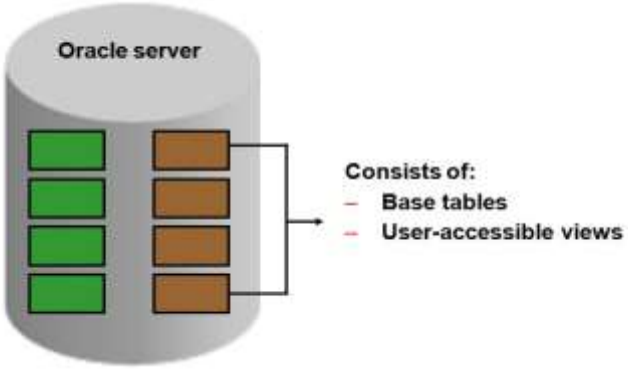
3- معلومات التقييد المتكاملة.

4- أسماء المستخدمين الذين يحق لهم العمل على قاعدة البيانات.

5- الامتيازات والقواعد الممنوحة لكل مستخدم.

6- معلومات عامة أخرى عن قاعدة البيانات.

هيكلية الـ Data Dictionary.



تتكون الـ Data Dictionary من مجموعة من الجداول والـ Views التي تلخص وتعرض المعلومات الخاصة بالجداول التي تحتويها قاعدة البيانات.

ويوجد هنالك أربعة أنواع من تلك الـ Views هي:

1- User View: وتحتوي على المعلومات الخاصة بكل مستخدم وما هي الوحدات التي يمتلكها.

2- ALL View: وتحتوي على المعلومات الخاصة بالوحدات التي يستطيع المستخدم الوصول إليها.

3- DBA View: وتحتوي على المعلومات الخاصة بكل العناصر الموجودة في قاعدة البيانات والمملوكة من قبل المستخدمين.

4- V$ View: وتحتوي على المعلومات الخاصة بالأداء وتكون هذه المعلومات متغيرة من وقت إلى أخر، كما أن هذا النوع لا يستطيع معظم المستخدمين من الوصول إليه ، وإنما فقط الـ DBA يستطيع الوصول إليه ، بالإضافة إلى انه يستطيع منح الامتيازات الخاصة بالوصول إلى هذا النوع إلى بقية المستخدمين.

كيفية الوصول إلى الـ Dictionary View.

نستطيع الوصول إلى الـ Dictionary View عن طريق استخدام أيعاز DESCRIBE ، وكما موضح في المثال أدناه.

```
DESCRIBE DICTIONARY
```

```
Name                          Null?    Type
----------------------------- -------- ----------------
TABLE_NAME                             VARCHAR2(30)
COMMENTS                               VARCHAR2(4000)
```

الفصل الثالث

اللغات الإجرائية

PL / SQL

اللغات الإجرائية PL / SQL.

أن جمل الـ SQL تعمل بصورة مستقلة عن بعضـها البعض ، حيث أن التأثير الخاص بها على الجمل الأخرى يكون قليل جدا وتستخدم بقلة في كتابة البرامج ، ولكتابـة برنـامج متكـون مـن مجموعـة مـن الـ Codes لتنفيـذ عمليـة معينـة فأننـا نسـتعمل لغـة تابعـة لبرنــامج الاوراكـل تـدعى باللغــة الإجرائيــة Procedural Language وتختصر بـ PL ، ومن مميزات استخدام هذه اللغة هو :

- ✓ عبارة عن كتلة مهيكلة.
- ✓ أدائها أفضل.
- ✓ إنتاجية عالية.
- ✓ قابلة للنقل.
- ✓ أمنية محكمة.
- ✓ معالجة الأخطاء.
- ✓ الوصول إلى الحزم المعرفة بصورة أولية.
- ✓ تدعم البرمجة من نوع OOP.
- ✓ تدعم تطوير صفحات الـ Web Application .

هيكلية لغة PL / SQL .

أن الوحدة الأساسية لأي برنامج مكتوب بلغة الـ PL / SQL هو الـ Block .
حيث أن كل برنامج يتكون من مجموعة من الـ blocks المتسلسلة او المتداخلة
وان الصيغة العامة لكتابة اي block بلغة PL/SQL هي.

- DECLARE (optional)
  - Variables, cursors, user-defined exceptions
- BEGIN (mandatory)
  - SQL statements
  - PL/SQL statements
- EXCEPTION (optional)
  - Actions to perform when errors occur
- END; (mandatory)

علما أن كل من جزء الـ declarative section والـ exception section هي اختيارية ، أي أن كتابتها أو عدم كتابتها لا يؤثر على سير البرنامج.

أن لغة الـ SQL / PL تتكون من ما يلي:

- ✓ المتغيـرات variables ، الثوابـت Constant ، أنـواع البيانـات الأخـرى another data type.
- ✓ جمل السيطرة مثل الجمل الشرطية Control Statements والـ Loop.
- ✓ أعادة استخدام وحدات برمجية سابقة تكتب في البرنامج مرة واحدة وتستدعى لتنفذ عدة مرات.

أنواع الـ Blocks . يوجد هنالك ثلاثة أنواع من الـ blocks هي:

1- Anonymous blocks: أن الهيكليـة العامـة لهـذا النـوع تكـون مشـابهه للهيكلية الخاصة بالـ blockالعادي.

2- Procedure block: أن الهيكلية العامة لهذا النوع تبدأ بكلمة procedure ومن ثم اسم الـ procedure يتبع ذلك الهيكلية العادية للـ block ، علما ان هذا النوع لا يرجع أي نتائج إلى الجزء الرئيسي للـ block.

3- Function block: أن هذا النوع يبدأ بكلمة Functionومن ثم اسم الـ Function بعد ذلك القيم التي يرجعها ، حيث أن هذا النوع يقوم بإرجاع قيم الى الـ Main block ومن ثم الهيكلية العامة للـ block العادي.

وان كل من الأنواع الثلاثة أعلاه موضحة في الشكل التالي:

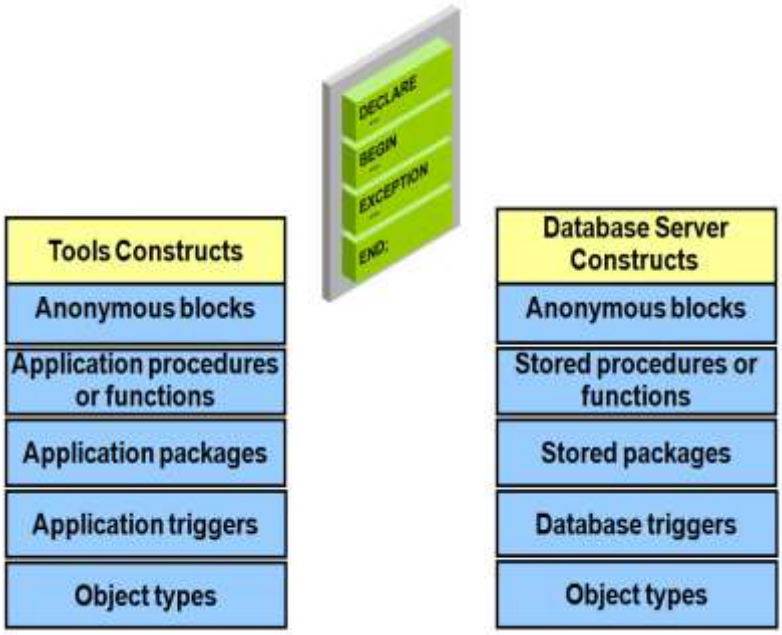
Anonymous
```
[DECLARE]

BEGIN
  --statements

[EXCEPTION]

END;
```

Procedure
```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

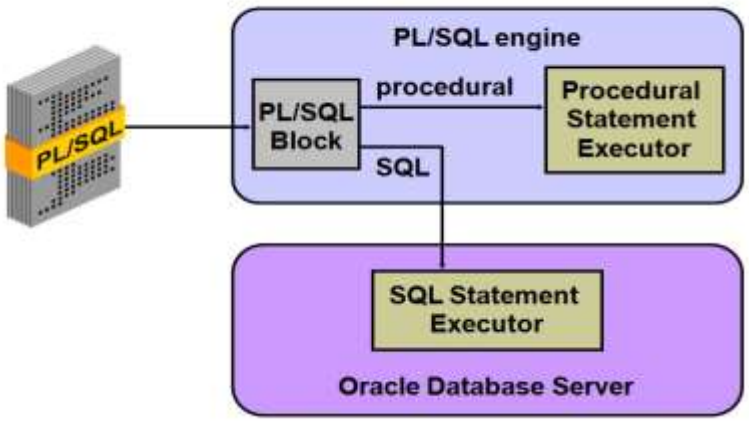
Function
```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```

بناء البرنامج في الـ SQL / PL.

أن بناء أي برنامج بلغة الـ SQL / PL يتم وفق المتطلبات الموضحة في الشكل أدناه.



بيئة الـ SQL / PL .



مثال1.

البرنامج التالي يقوم بقراءة حقل الاسم في جدول الطلبة وخزنه في متغير اسمه Vname يتم تعريفه مسبقا عندما يكون الـ 1 = no ، وذلك باستخدام الـ Anonymous PL / SQL ؟

```
SQL >DECLARE
        Vname VARCHAR2 (30);
        BEGIN
                SELECT name
                INTO Vname
                FROM stud3
                WHERE no=1;
        END;
```

وعند تنفيذ ذلك الـ Block سوف نحصل على process statement ولن نحصل على النتيجة ، ولغرض الحصول على نتيجة معينة في لغة الـ PL / SQL نستخدم عبارة الـ DBMS_OUTPUT والصيغة العامة لها هي :

```
DBMS_OUTPUT.PUT_LINE (var. name);
```

```
SQL > DECLARE
        Vname VARCHAR2 (30);
        BEGIN
                SELECT name
                INTO Vname
```

```
        FROM stud3
        WHERE no=1;
    DBMS_OUTPT.PUT_LINE ('The first name in this
table is:' || Vname);
    END;
```

استخدام المتغيرات في لغة الـ PL / SQL:

تستخدم المتغيرات هنا لخزن المتغيرات الوقتية لحين معالجتها وتجهيزها للمستخدم ، ولكل متغير اسم خاص به نستطيع استدعاء المتغير به منذ بداية البرنامج إلى نهايته، وتخضع تسمية المتغيرات الى مجموعة من القواعد التي يجب الالتزام بها منها:

1- يجب أن يبدأ بحرف.

2- من الممكن أن يحتوي على مجموعة من الأحرف والأرقام.

3- من الممكن أن تحتوي على رموز خاصة مثل ($, # , @, ...,etc).

4- يجب أن يكون اكبر طول لاسم المتغير هو (30 Characters).

5- يجب أن لا يكون كلمة معرفة للغة مثل (BEGIN , END , ... , etc).

معالجة المتغيرات في لغة الـ PL / SQL:

يتم معالجة المتغيرات في لغة الـ PL / SQL بالصيغة التالية:

1- يتم تعريفها وإعطائها قيم ابتدائية في الجزء التعريفي للبرنامج.

2- يتم استخدامها وإعطائها قيم جديدة في الجزء التنفيذي للبرنامج.

3- تمرر كمعاملات PARAMETORS من subprogram إلى أخر في البرنامج.

4- نستطيع بواسطتها تجهيز النتائج للمستخدم بواسطة طباعتها على الشاشة باستخدام عبارة الـ DBMS_OUTPUT.

تعريف المتغيرات في لغة الـ PL / SQL.

أن الصيغة العامة لتعريف المتغيرات في لغة الـ PL / SQL هي:

```
identifier [CONSTANT] datatype [NOT NULL]
    [:= | DEFAULT expr];
```

مثال ذلك:

```
DECLARE
  emp_hiredate    DATE;
  emp_deptno      NUMBER(2) NOT NULL := 10;
  location        VARCHAR2(13) := 'Atlanta';
  c_comm          CONSTANT NUMBER := 1400;
```

مثال أخر:

1

```
SET SERVEROUTPUT ON
DECLARE
  Myname VARCHAR2(20);
BEGIN
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
  Myname := 'John';
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
END;
/
```

2

```
SET SERVEROUTPUT ON
DECLARE
  Myname VARCHAR2(20):= 'John';
BEGIN
  Myname := 'Steven';
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
END;
/
```

أنواع المتغيرات في لغة الـ SQL / PL:

1- الثوابت Scalar.

2- المركبة Composite.

3- العناوين References.

4- العناصر الكبيرة Large object.

قواعد تعريف المتغيرات في لغة الـ SQL / PL.

1- تتبع التسمية دائما نوع المتغير ومواصفاته.

2- يجب إعطاء أسماء ذو معنى للمتغيرات.

3- من الممكن استعمال عبارة الـ Not Null والـ Constant مع اسم المتغير.

4- من الممكن اعطاءء قيمة ابتدائية للمتغير أثناء تعريفه من خلال استخدام. (=:) أو استخدام جملة.

5- الـ **Default** مثال ذلك: SQL > myname VARCHAR2 (30):='ahmed'; Or SQL > myname VARRCHAR2 (30) .DEFAULT 'ahmed'

6- تعريف متغير واحد فقط في كل سطر وذلك من اجل سهولة القراءة والصيانة فيما بعد.

7- يمنع استخدام اسم العمود الأصلي كمتغير، مثال ذلك.

```
DECLARE
  employee_id NUMBER(6);
BEGIN
  SELECT    employee_id
  INTO      employee_id
  FROM      employees
  WHERE     last_name = 'Kochhar';
END;
/
```

أنواع المتغيرات Scalar:

هنالك ستة عشر أنواع رئيسية تابعة لمتغيرات الـ Scalar في لغة الـ PL / SQLوهذه المتغيرات موضحة في أدناه.

- CHAR [(maximum_length)]
- VARCHAR2 (maximum_length)
- LONG
- LONG RAW
- NUMBER [(precision, scale)]
- BINARY_INTEGER
- PLS_INTEGER
- BOOLEAN
- BINARY_FLOAT
- BINARY_DOUBLE
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

وكما موضح في الأمثلة التالية:

```
DECLARE
  emp_job         VARCHAR2(9);
  count_loop      BINARY_INTEGER := 0;
  dept_total_sal  NUMBER(9,2) := 0;
  orderdate       DATE := SYSDATE + 7;
  c_tax_rate      CONSTANT NUMBER(3,2) := 8.25;
  valid           BOOLEAN NOT NULL := TRUE;
  ...
```

تعريف المتغيرات المنطقية.

أن المتغيرات المنطقية من الممكن أن تقوم بخزن ثلاثة قيم أساسية وهي الـ True والـ False والـ Null ، وتنتج تلك القيم من التعابير الشرطية التي تستخدم العمليات المنطقية مثل الـ (AND , OR) ، والعمليات الأحادية مثل الـ NOT التي تستخدم لفحص المتغيرات .

تعريف متغيرات الإحلال Substitution variables.

وهي المتغيرات التي تستخدم للحصول على قيم مدخلة من المستخدم اثناء وقت التنفيذ، والتي يشار إليها من داخل block سابق من خلال علامة الـ (&). وكما موضح في المثال التالي.

```
VARIABLE emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
  empno NUMBER(6):=&empno;
BEGIN
  SELECT salary  INTO :emp_salary
  FROM  employees WHERE employee_id = empno;
END;
/
```

1

```
VARIABLE emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
empno NUMBER(6):=100;
BEGIN
SELECT salary  INTO :emp_salary
FROM  employees WHERE employee_id = empno;
END;
anonymous block completed
emp_salary
-----
24000
```

2

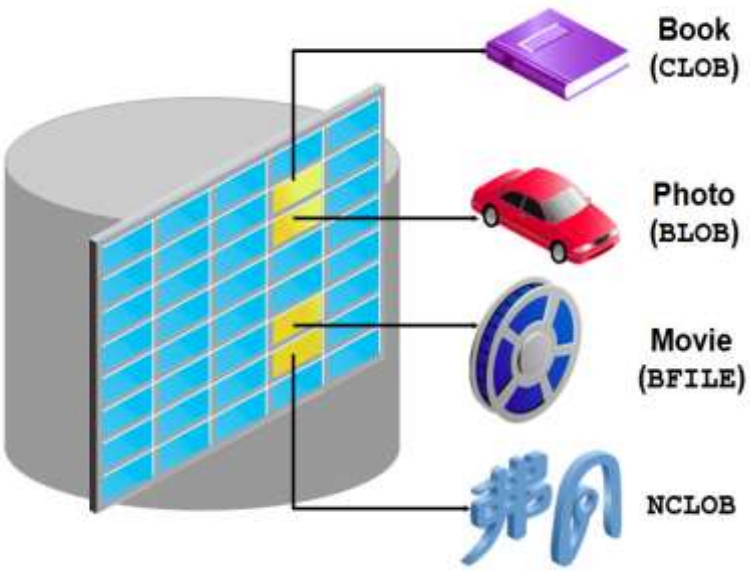
ملاحظة :

من الممكن وضع جمل توضيحية مع صندوق إدخال البيانات وكما موضح في المثال التالي:

```
SET VERIFY OFF
VARIABLE emp_salary NUMBER
ACCEPT empno PROMPT 'Please enter a valid employee
number: '
SET AUTOPRINT ON
DECLARE
  empno NUMBER(6) := &empno;
BEGIN
  SELECT salary  INTO :emp_salary  FROM  employees
  WHERE employee_id = empno;
END;
/
```

متغيرات العناصر الكبيرة Large Object :

ويرمز لها بالرمز LOB وتشمل المتغيرات الأنواع التالية :



ملاحظات هامة:

أ- أن المتغيرات الحرفية والتاريخ عندما يتم استخدامها في الـ PL / SQL يجب أن يتم حصرها داخل علامتي اقتباس مفردة، مثال ذلك:

```
SQL> name = 'Ali';
SQL> birthdate = ' 12-10-2013';
```

ب- من الممكن ان تستمر كتابة الجملة الواحدة على أكثر من سطر واحد.

ت- من الممكن كتابة الملاحظات في لغة الـ PL / SQL وذلك لتوضيح عمل الـ code المكتوب لتسهيل قراءته من قبل المبرمجين الآخرين ولسهولة الصيانة فيما بعد ، وهنالك نوعان من الملاحظات. الأول هي الملاحظات التي تحتاج إلى سطر واحد فقط لكتابتها وهي تحتاج إلى أن تسبق.

ث- بـ (--) قبلها حتى يفهم

ج- الـ Compiler الخاص باللغة أنها مجرد ملاحظات ولا تتبع الـ code التابع للبرنامج . النوع الثاني هي الملاحظات التي تحتاج إلى أكثر من سطر واحد وهذا النوع يحتاج إلى أن يبدأ بـ (*/) وينتهي بـ (/*) وكما موضح في المثال التالي.

Example

```
DECLARE
...
annual_sal NUMBER (9,2);
BEGIN    -- Begin the executable section

/* Compute the annual salary based on the
   monthly salary input from the user */
annual_sal := monthly_sal * 12;
END;     -- This is the end of the block
/
```

دوال لغة الـ PL / SQL .

هنالك مجموعة من الدوال التي من الممكن استخدامها في لغة الـ PL / SQL والتي تم استخدامها سابقا في الـ SQL والتي وظيفتها القيام بمجموعة من العمليات الحسابية والمنطقية وحسب الغرض المكتوب لأجلها، وهذه الدوال هي :

- Single-row number
- Single-row character
- Data type conversion
- Date
- Timestamp
- GREATEST and LEAST
- Miscellaneous functions

أما بالنسبة إلى الدوال التي تم استخدامها سابقا في الـ SQL ولا يحق لنا استخدامها حاليا في الـ PL / SQL فهي:

- DECODE
- Group functions

مثال1.

من الممكن استخدام دالة الـ LENGTH لإرجاع طول نص معين وكما موضح في المثال التالي:

```
desc_size INTEGER(5);
prod_description VARCHAR2(70):='You can use this
product with your radios for higher frequency';

-- get the length of the string in prod_description
desc_size:= LENGTH(prod_description);
```

Example:

```
SQL> DECLARE
        len NUMBER(10);
        str VARCHAR2(50);
        BEGIN
        str := ' MY NAME IS WISSAM ALI , I LIFE IN
        KARBAL' ;
        len := LENGTH (str);
        DBMS_OUTPUT.PUT_LINE ( ' the string is: '|| str);
        DBMS_OUTPUT.PUT_LINE ( ' the length of this string
        is: '|| len);
        END;
```

مثال 2.

المثال التالي يستخدم دالة الـ LOWER لتحويل النص ذو الحروف الكبيرة إلى نص ذو حروف صغيرة.

```
SQL> DECLARE
        Str1 VARCHAR2 (50);
        Str2 VARCHAR2 (50);
        BEGIN
        Str1:= ' MY NAME IS WISSAM ALI , I LIFE IN
        KARBAL' ;
        Str2:= LOWER (str1);
        DBMS_OUTPUT.PUT_LINE ( ' the original string is:
        '|| str1);
        DBMS_OUTPUT.PUT_LINE ( ' the new string is: '||
        str2);
        END;
```

التحويل بين أنواع البيانات:

هنالك نوعان من عمليات التحويل بين الأنواع المختلفة للبيانات هما:

- ✓ التحويل الضمني Implicit conversion.
- ✓ التحويل الصريح Explicit conversion.

هنالك أربعة أنواع من دوال التحويل بين أنواع المتغيرات التي تم استخدامها سابقا في الـ SQL ونستطيع استخدامها حاليا في الـ PL / SQL

وهي:

- ✓ TO_CHAR.
- ✓ TO_DATE.
- ✓ TO_NUMBER.
- ✓ TO_TIMESTAMP.

الكتل المتداخلة Nested Block.

من الممكن أن تحتوي لغة الـ PL / SQL على block متداخل ، حيث من الممكن أن يكون هنالك جزء تنفيذي متداخل (nested begin) بالإضافة إلى انه من الممكن ان يكون هنالك جزء Exception متداخل أيضا ، وكما موضح في الأمثلة التالية.

Example.

```
DECLARE
 outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
  DECLARE
   inner_variable VARCHAR2(20):='LOCAL VARIABLE';
  BEGIN
   DBMS_OUTPUT.PUT_LINE(inner_variable);
   DBMS_OUTPUT.PUT_LINE(outer_variable);
  END;
 DBMS_OUTPUT.PUT_LINE(outer_variable);
END;
/
```

Example2.

```
DECLARE
 father_name VARCHAR2(20):='Patrick';
 date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   child_name VARCHAR2(20):='Mike';
   date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||child_name);
  END;
 DBMS_OUTPUT.PUT_LINE('Date of Birth: '||date_of_birth);
END;
/
```

(1) — the three DBMS_OUTPUT.PUT_LINE statements of the inner block; (2) — the DBMS_OUTPUT.PUT_LINE statement of the outer block.

العمليات المستخدمة في الـ PL/SQL:

هنالك خمسة أنواع من العمليات المستخدمة في الـ SQL / PL وهذه العمليات هي:

- Logical
- Arithmetic
- Concatenation
- Parentheses to control order of operations

} Same as in SQL

- Exponential operator (**)

Examples:

- Increment the counter for a loop.

```
loop_count := loop_count + 1;
```

- Set the value of a Boolean flag.

```
good_sal := sal BETWEEN 50000 AND 150000;
```

- Validate whether an employee number contains a value.

```
valid  := (empno IS NOT NULL);
```

التفاعل مع لغة الـ PL/SQL:

من الممكن التفاعل مع لغة الـ PL/SQL عن طريق استخدام الأوامر والجمل المتوفرة في لغة الـ SQL في الـ PL/SQL حيث من الممكن :

- ✓ استعادة القيود من قاعدة البيانات باستخدام الأمرSELECT.
- ✓ تغيير قيم القيود ومعالجة بياناتها باستخدام أوامر الـ DML.
- ✓ استخدام أوامر السيطرة ControlTransaction مثل أوامر (COMMIT , ROLLBACK , SAVEPOINT).

استعادة القيود في PL/SQL:

من الممكن استعادة القيود في لغة الـ PL/SQL عن طريق استخدام الامر SELECT ، حيث ان الصيغة العامة لاستخدام هذا الامر موضحة كما يلي:

```
SELECT  select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
[WHERE  condition];
```

يجب هنا استخدام عبارة INTO حيث أن استخدامها واجب ، إضافة إلى أن كل استعلام يجب أن يرجع قيد واحد فقط لا أكثروالأمثلة التالية توضح ذلك.

Example1.

```
SET SERVEROUTPUT ON
DECLARE
 fname VARCHAR2(25);
BEGIN
 SELECT first_name INTO fname
 FROM employees WHERE employee_id=200;
 DBMS_OUTPUT.PUT_LINE(' First Name is : '||fname);
END;
/
```

Example2:

```
SQL> DECLARE
        name1 VARCHAR2 (30);
        BEGIN
        SELECT stuname
        INTO name1
        FROM student
        WHERE stuid =1;
        DBMS_OUTPUT.PUT_LINE ('THE FIRST RECORD
        IS:' || name1);
       END;
```

ملاحظة مهمة:

من الممكن استرجاع حقلين في نفس الاستعلام وكما موضح في المثال أدناه.

Example 3:

```
SQL> DECLARE
        name1 VARCHAR2 (30);
        gov1 VARCHAR2 (10);
        BEGIN
        SELECT stuname, governorate
        INTO name1, gov1
        FROM student
        WHERE stuid =1;
```

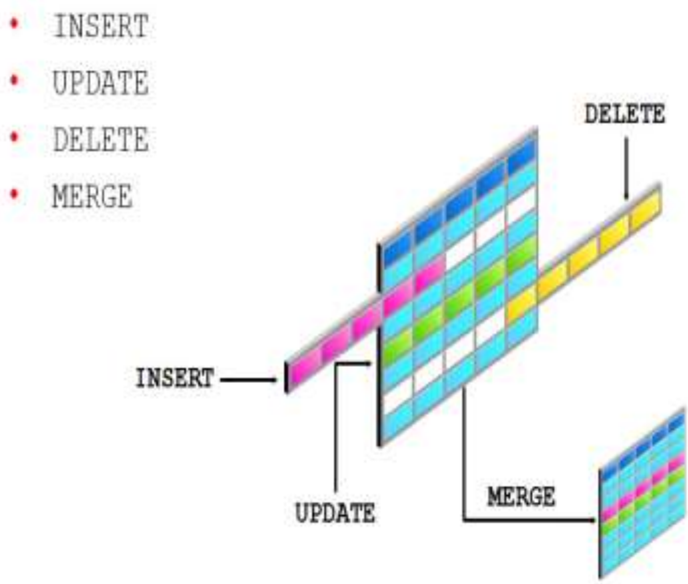
```
DBMS_OUTPUT.PUT_LINE ('THE FIRST RECORD IS:'
|| name1);
DBMS_OUTPUT.PUT_LINE ('THE GOVERNORATE IS:'
|| gov1);
END;
```

من الممكن استخدام الدوال الرياضية لإتمام عملية معينة عند استعادة قيد محدد، على سبيل المثال البرنامج التالي يقوم بإرجاع مجموع المبيعات لكل الموظفين في القسم رقم (60).

```
SET SERVEROUTPUT ON
DECLARE
   sum_sal  NUMBER(10,2);
   deptno   NUMBER NOT NULL := 60;
BEGIN
   SELECT  SUM(salary)  -- group function
   INTO sum_sal FROM employees
   WHERE  department_id = deptno;
   DBMS_OUTPUT.PUT_LINE ('The sum of salary is '
   || sum_sal);
END;
/
```

معالجة البيانات باستخدام لغة PL / SQL :

من الممكن معالجة البيانات المخزونة في الجداول باستخدام أوامر DML التي استخدمناها سابقا في الـ SQL ومن هذه الأوامر:



1- استخدام أيعاز INSERT.

من الممكن استخدام هذا الإيعاز لإدخال قيد جديد إلى الجدول، والمثال أدناه يقوم بإدخال معلومات موظف جديد إلى جدول الموظفين.

Example:

```
BEGIN
 INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
   VALUES(employees_seq.NEXTVAL, 'Ruth', 'Cores',
   'RCORES',sysdate, 'AD_ASST', 4000);
END;
/
```

2- استخدام أيعاز UPDATE .

ويستخدم هذا الإيعاز لتحديث المعلومات الموجودة في الجدول والمخزنة سابقا، والمثال التالي يقوم بزيادة رواتب كل الموظفين الذين يقومون بتخزين الكتب.

```
DECLARE
  sal_increase   employees.salary%TYPE := &sal_incr;
BEGIN
  UPDATE     employees
  SET        salary = salary + sal_increase
  WHERE      job_id = 'ST_CLERK';
END;
/
```

3- استخدام أيعاز DELETE.

ويستخدم هذا الإيعاز لحذف قيد تابع لجدول معين، على سبيل المثال البرنامج التالي يقوم بحذف القيد الذي رقم القسم له هو (10) والتابع إلى جدول الموظفين.

Example:

```
DECLARE
  deptno   employees.department_id%TYPE := 10;
BEGIN
  DELETE FROM   employees
  WHERE  department_id = deptno;
END;
/
```

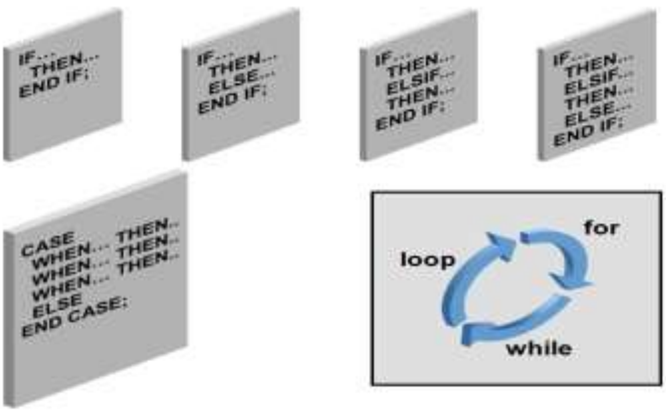
4- أيعاز MAREGE .

ويستخدم هذا الإيعاز لدمج القيود في جدول معين، وكما موضح في المثال أدناه.

```
DECLARE
     empno employees.employee_id%TYPE := 100;
BEGIN
MERGE INTO copy_emp c
     USING employees e
     ON (e.employee_id = c.empno)
   WHEN MATCHED THEN
     UPDATE SET
       c.first_name     = e.first_name,
       c.last_name      = e.last_name,
       c.email          = e.email,
       . . .
   WHEN NOT MATCHED THEN
     INSERT VALUES(e.employee_id, e.first_name, e.last_name,
          . . .,e.department_id);
END;
/
```

استخدام عبارات السيطرة:

هنالك أربعة أنواع من عبارات السيطرة المستخدمة في لغة الـ PL/SQL وهذه العبارات هي.



1- جملة الـ IF Statement :

تستخدم هذه العبارة الشرطية لتنفيذ جملة معينة أو مجموعة من الجمل عند تحقق شرط معين او مجموعة من الشروط ، والصيغة العامة لهذه العبارة هي:

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

مثال1.

المثال التالي يقوم بقراءة العمر عن طريق متغير معين اذا كان العمر اقل من (11) سنة يطبع (انا طفل).

```
DECLARE
  myage number:=31;
BEGIN
  IF myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  END IF;
END;
/
```

```
anonymous block completed
```

مثال 2 :

المثال التالي يقوم بقراءة العمر عن طريق متغير معين اذا كان العمر اقل من (11) سنة يطبع (أنا طفل) أما إذا كان اكبر من ذلك فيطبع (أنا لست طفل)

```
SET SERVEROUTPUT ON
DECLARE
myage number:=31;
BEGIN
IF myage < 11
 THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
 ELSE
    DBMS_OUTPUT.PUT_LINE(' I am not a child ');
END IF;
END;
/
```

```
anonymous block completed
 I am not a child
```

مثال3.

البرنامج التالي يقوم بفحص الجنسية الخاصة بالطالب الذي يملك الرقم (1) أذا كانت الجنسية عراقية يطبع (الطالب محلي الجنسية) أما أذا كان من جنسية أخرى فيطبع (الطالب أجنبي الجنسية).

```
SQL> DECLARE
        not1 VARCHAR2 (10);
        BEGIN
        SELECT nationality
        INTO not1
        FROM student
        WHERE stuid = 1;
        DBMS_OUTPUT.PUT_LINE (' Nationality is: ' || not1);
        IF not1 = 'Iraq' THEN
        DBMS_OUTPUT.PUT_LINE ('He is National Student');
```

```
ELSE
DBMS_OUTPUT.PUT_LINE(' He is Forign Student');
END IF;
END;
```

مثال 4.

المثال التالي يوضح عمل IF المتداخلة.

```
DECLARE
myage number:=31;
BEGIN
IF myage < 11
 THEN
       DBMS_OUTPUT.PUT_LINE(' I am a child ');
   ELSIF myage < 20
     THEN
       DBMS_OUTPUT.PUT_LINE(' I am young ');
   ELSIF myage < 30
     THEN
       DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
   ELSIF myage < 40
     THEN
       DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
 ELSE
    DBMS_OUTPUT.PUT_LINE(' I am always young ');
END IF;
END;
/
```

```
anonymous block completed
 I am in my thirties
```

2- عبارة الـ Case Statement:

تستخدم هذه العبارة عندما يكون لدينا مجموعة من الشروط التي لا نستطيع تنفيذها بواسطة الـ IF Statement أو أن عملية التنفيذ تتم بصورة معقدة ، حيث ان هذه العبارة تقوم بتنفيذ جملة أو مجموعة من الجمل عند تحقق شرط معين .

والصيغة العامة لهذه الجملة هي:

```
CASE selector
   WHEN expression1 THEN result1
   WHEN expression2 THEN result2
   ...
   WHEN expressionN THEN resultN
  [ELSE resultN+1]
END;
/
```

Example1.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DECLARE
   grade CHAR(1) := UPPER('&grade');
   appraisal VARCHAR2(20);
BEGIN
   appraisal :=
      CASE grade
         WHEN 'A' THEN 'Excellent'
         WHEN 'B' THEN 'Very Good'
         WHEN 'C' THEN 'Good'
         ELSE 'No such grade'
      END;
DBMS_OUTPUT.PUT_LINE ('Grade: '|| grade || '
                       Appraisal ' || appraisal);
END;
/
```

Example 2.

```
DECLARE
   grade CHAR(1) := UPPER('&grade');
   appraisal VARCHAR2(20);
BEGIN
    appraisal :=
     CASE
         WHEN grade = 'A' THEN 'Excellent'
         WHEN grade IN ('B','C') THEN 'Good'
         ELSE 'No such grade'
     END;
   DBMS_OUTPUT.PUT_LINE ('Grade: '|| grade || '
                   Appraisal ' || appraisal);
END;
/
```

Example 3.

```
DECLARE
   deptid NUMBER;
   deptname VARCHAR2(20);
   emps NUMBER;
   mngid NUMBER:= 108;
BEGIN
  CASE  mngid
   WHEN  108 THEN
    SELECT department_id, department_name
     INTO deptid, deptname FROM departments
     WHERE manager_id=108;
    SELECT count(*) INTO emps FROM employees
     WHERE department_id=deptid;
   WHEN  200 THEN
     ...
 END CASE;
DBMS_OUTPUT.PUT_LINE ('You are working in the '|| deptname||
' department. There are '||emps ||' employees in this
department');
END;
/
```

3- الجمل التكرارية Iterative Statements :

تستخدم الجمل التكرارية لتكرار تنفيذ جملة معينة أو مجموعة من الجمل عدة مرات لحين انتهاء الشرط المحدد من قبل المبرمج ، وهنالك ثلاثة أنواع من الجمل التكرارية :

- ✓ Basic loop.
- ✓ For loop.
- ✓ While loop.

أ- التكرار البسيط basic loop:

وهو ابسط أنواع التكرار ويبدأ دائما بكلمة loop وينتهي بـ End loop وكما موضح في الصيغة التالية:

```
LOOP
  statement1;
  . . .
  EXIT [WHEN condition];
END LOOP;
```

Example

```
DECLARE
  countryid    locations.country_id%TYPE := 'CA';
  loc_id       locations.location_id%TYPE;
  counter       NUMBER(2) := 1;
  new_city     locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = countryid;
  LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((loc_id + counter), new_city, countryid);
    counter := counter + 1;
    EXIT WHEN counter > 3;
  END LOOP;
END;
/
```

جملة الـ While :

وتستخدم لتنفيذ جملة أو مجموعة من الجمل إلى حين الانتهاء من تنفيذ شرط معين يتم تحديده من قبل المبرمج، والصيغة العامة لهذه الجملة موضحة أدناه:

```
WHILE condition LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

Example

```
DECLARE
  countryid   locations.country_id%TYPE := 'CA';
  loc_id      locations.location_id%TYPE;
  new_city    locations.city%TYPE := 'Montreal';
  counter     NUMBER := 1;
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = countryid;
  WHILE counter <= 3 LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((loc_id + counter), new_city, countryid);
    counter := counter + 1;
  END LOOP;
END;
/
```

جملة الـ For Statement:

تستخدم هذه العبارة لتكرار جملة أو مجموعة من الجمل عدة مرات ، وتختلف عن سابقاتها في أنها تحتوي على قيمة بداية الـ Loop وقيمة نهايته ، والصيغة العامة لهذه الجملة هو :

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

Example:

```
DECLARE
  countryid   locations.country_id%TYPE := 'CA';
  loc_id      locations.location_id%TYPE;
  new_city    locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO loc_id
    FROM locations
    WHERE country_id = countryid;
  FOR i IN 1..3 LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((loc_id + i), new_city, countryid );
  END LOOP;
END;
/
```

ب-التكرار المتداخل Nested Loop:

من الممكن كتابة جملة تكرارية داخل جملة أخرى عند الحاجة لذلك والمثال التالي يوضح ذلك.

```
...
BEGIN
  <<Outer_loop>>
  LOOP
    counter := counter+1;
  EXIT WHEN counter>10;
    <<Inner_loop>>
    LOOP
      ...
      EXIT Outer_loop WHEN total_done = 'YES';
      -- Leave both loops
      EXIT WHEN inner_done = 'YES';
      -- Leave inner loop only
      ...
    END LOOP Inner_loop;
    ...
  END LOOP Outer_loop;
END;
/
```

البيانات المركبة Composite Data Type.

هي البيانات التي تستطيع خزن ومعالجة عدة قيم في نفس الوقت ، على العكس من البيانات الثابتة Scalar Data type. وهنالك نوعان من الـ Composite Data Type هي :

1- PL/SQL Records.

أن هذا النوع يستخدم للتعامل مع البيانات المتصلة كوحدات منطقيه ويستخدم لخزن البيانات التي تكون مختلفة الأنواع ولكنها مرتبطة مع بعضها البعض، حيث أن كل قيد من الممكن أن يحتوي على بيانات مختلفة في النوع ، مثال ذلك من الممكن استخدامه في خزن بيانات الموظفين التي تتضمن خزن رقم الموظف والاسم الأول له والاسم الأخير وتاريخ الميلاد .... الخ، حيث أننا بإمكاننا معالجة تلك البيانات بسهولة عند خزنها على شكل قيود.

2- PL/SQL Collections.

أن هذا النوع يستخدم للتعامل مع البيانات كوحدات مفردة ، حيث انه يستخدم لخزن البيانات التي تكون من نفس النوع، وهنالك ثلاثة أنواع من تلك المجاميع هي.

- ✓ INDEX BY tables or associative arrays.
- ✓ Nested tables.
- ✓ VARRAY.

أن عملية استخدام الـ Composite Data في عملية خزن البيانات المتصلة تسهل من عملية الوصول إلى تلك البيانات وتحديثها ومعالجتها بالإضافة إلى سهولة نقلها من مكان إلى أخر ، وكشيء مشابه لذلك في حياتنا هو أننا نملك حقيبة نستطيع جمع كل مكونات اللابتوب داخلها وفي نفس الوقت فأننا نملك حقيبة خاصة لكل مكون من مكونات اللابتوب.

مثلما تم ذكره سابقا فان الـ Record يعرف على انه مجموعة من البيانات المتصلة والمختلفة في النوع والتي تخزن على شكل حقول كل حقل يملك اسم خاص به ، ويمتلك القيد الخصائص التالية:

1- أن كل قيد معرف من الممكن أن يحتوي على عدة حقول.
2- أن كل قيد من الممكن أن يعطى قيمة ابتدائية في بداية تعريفه، ومن الممكن أن يعرف على انه Not Null.
3- أن كل قيد لا يتم أعطاء قيمة ابتدائية له أثناء تعريفه ، فان ذلك يعني أننا نسمح بوجود حالة الـ NULL.
4- من الممكن استخدام الكلمة المحجوزة default أثناء تعريف الحقول.
5- من الممكن تعريف أي قيد من قبل الـ User من خلال الجزء التعريفي لأي block أو subprogram أو package.
6- من الممكن تعريف قيود متداخلة والإشارة إليها . حيث أن احد هذه القيود يكون القيد الأساسي لبقية القيود المعرفة.

أن الصيغة العامة لتعريف أي قيد يتم حسب الصيغة التالية:

Syntax:

1
```
TYPE type_name IS RECORD
    (field_declaration[, field_declaration]…);
```

2
```
identifier   type_name;
```

field_declaration:

```
field_name {field_type | variable%TYPE
          | table.column%TYPE | table%ROWTYPE}
          [[NOT NULL] {:= | DEFAULT} expr]
```

أن الصيغة أعلاه يتم تعريفها في الجزء التعريفي لأي block حيث يتم أعطاء اسم لكل record بالإضافة إلى تعريف مكوناته الداخلية ، إضافة إلى أننا نستطيع أعطاء قيم ابتدائية لكل حقل من الحقول الـ record أو يتم تركها فارغة ليتم فيما بعد ملئها بالقيم. ويمكن تعريف أجزاء الصيغة أعلاه كما يلي:

- ✓ **Type _ name**: يستخدم لتعريف اسم القيد.
- ✓ **Field _name**: يمثل أسماء الحقول الموجودة داخل القيد.
- ✓ **Field_type**: يمثل نوع الحقل ، حيث من الممكن استخدام جميع الأنواع التي تم تعريفها سابقا عدا الـ REF CURSOR ، إضافة إلى أننا نستطيع استخدام الـ %TYPE & %ROWTYPE في تعريف تلك الحقول.
- ✓ **Expr**: يمثل القيمة الابتدائية التي يتم أعطائها للحقل.

مثال :

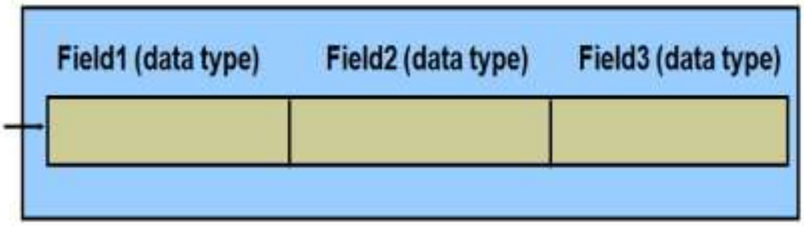
الكود التالي يقوم بتعريف قيد اسمه emp_record يقوم بخزن الاسم و الوظيفة ومقدار الراتب لأي موظف جديد، وكما موضح أدناه.

Example:

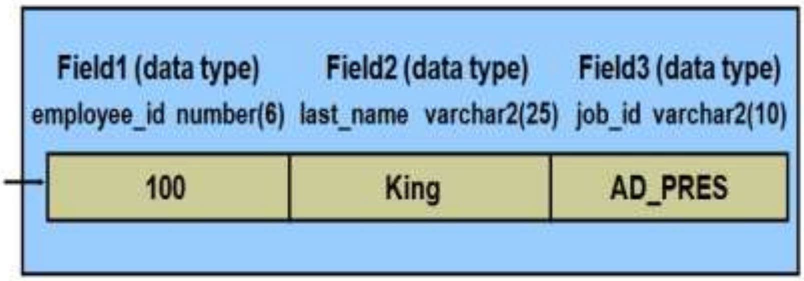
```
...
 TYPE emp_record_type IS RECORD
   (last_name   VARCHAR2(25),
    job_id      VARCHAR2(10),
    salary      NUMBER(8,2));
 emp_record     emp_record_type;
...
```

هيكلية القيد .

من الممكن تخيل الهيكلية التي يتم فيها تخزين القيود داخل قاعدة البيانات وحسب المثال أعلاه بالشكل التالي.



Example:



من الممكن الوصول إلى كل حقل داخل القيد لغرض إعطاءه قيمه معينه أو تحديث القيم التي يحتويها عن طريق ذكر اسم القيد أولا ومن ثم ذكر اسم الحقل، وكما موضح في أدناه.

```
SQL> record_name.field_name;
```

EXAMPLE.

```
SQL> emp_record.job_id;
```

من الممكن أعطاء قيمة لأي قيد وكما موضح في المثال التالي.

EXAMPLE.

```
SQL> emp_record.job_id:= 'Registration';
```

3- PL/SQL Collections.

أن هذا النوع يستخدم للتعامل مع البيانات كوحدات مفردة ، حيث انه يستخدم لخزن البيانات التي تكون من نفس النوع، وهنالك ثلاثة أنواع من تلك المجاميع هي.

- ✓ **Index Bytables or associative arrays.**

هي عبارة عن collection من البيانات المركبة composite data type التي يتم تعريفها من قبل المستخدم user ، ويتميز هذا النوع بكونه يقوم بخزن البيانات باستخدام primary key يعمل كـ Index . حيث تكون قيمة هذا الـ Key متسلسلة .

وإذا كانت غير متسلسلة فانه يقوم بخزن تلك البيانات على شكل أزواج ثنائية Pairs (حيث من الممكن تخيلها بان تلك البيانات يتم تخزينها في عمودين).

وان الصيغة العامة لتعريف القيد من النوع INDEX BY موضحة أدناه :

Syntax:

```
TYPE type_name IS TABLE OF
     {column_type | variable%TYPE
     | table.column%TYPE} [NOT NULL]
     | table%ROWTYPE
     [INDEX BY PLS_INTEGER | BINARY_INTEGER
      | VARCHAR2(<size>)];
identifier    type_name;
```

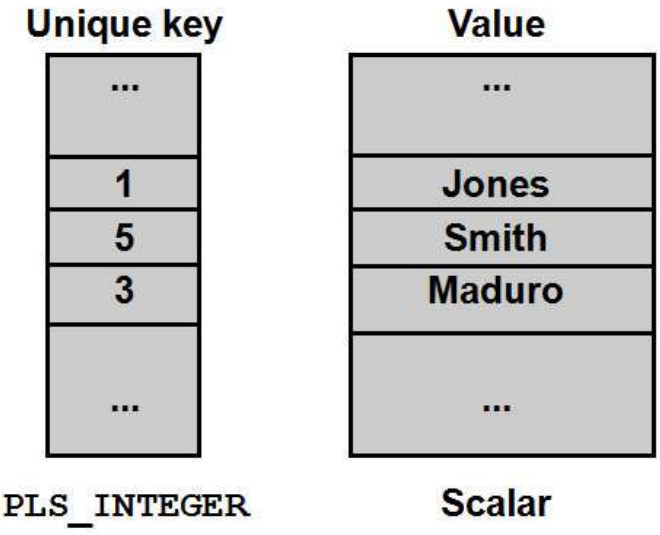
Declare an INDEX BY table to store the last names of employees:

```
...
TYPE ename_table_type IS TABLE OF
 employees.last_name%TYPE
 INDEX BY PLS_INTEGER;
 ...
ename_table ename_table_type;
```

EXAMPLE.

```
DECLARE
  TYPE ename_table_type IS TABLE OF
    employees.last_name%TYPE
    INDEX BY PLS_INTEGER;
  TYPE hiredate_table_type IS TABLE OF DATE
    INDEX BY PLS_INTEGER;
  ename_table        ename_table_type;
  hiredate_table     hiredate_table_type;
BEGIN
  ename_table(1)     := 'CAMERON';
  hiredate_table(8) := SYSDATE + 7;
    IF ename_table.EXISTS(1) THEN
    INSERT INTO ...
    ...
END;
/
```

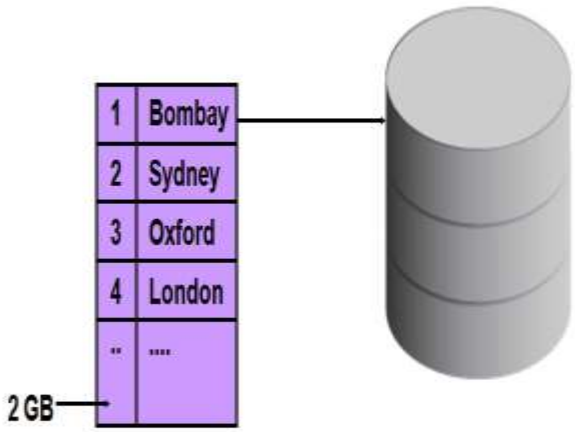
حيث من الممكن تخيل الهيكلية الخاصة بهذا النوع كما يلي:



✓ **الجداول المتداخلة NESTE TABLE:**

أن هذا النوع يكون مشابه إلى النوع السابق في الإجراءات ، ولكن هنالك اختلاف في التنفيذ . بحيث أن نوعية البيانات التابعة للوحدات الخاصة بالجدول تكون من النوع القانوني Valid data type بينما في النوع السابق تكون من النوع Invalid data type .

كما أن الـ Index Key في كلا النوعين لايمكن أن يكون عدد سالب Negative number ، وان الهيكلية العامة للـ Nested Table موضحة في أدناه.



أن الصيغة العامة لتعريف الـ Record من النوع Nested Table هي :

SQL **>** TYPE *type_name* IS TABLE OF

{ column_type | variable%TYPE | table.column%TYPE

{ [NOT NULL]

| Table%ROWTYPE;

EXAMPLE.

SQL > SET SERVEROUTPUT ON

DECLARE

TYPE location_type IS TABLE OF locations.city%TYPE;

offices location_type;

table_count NUMBER;

BEGIN

offices := location_type('Bombay', 'Tokyo','Singapore', 'Oxford');

table_count := offices.count();

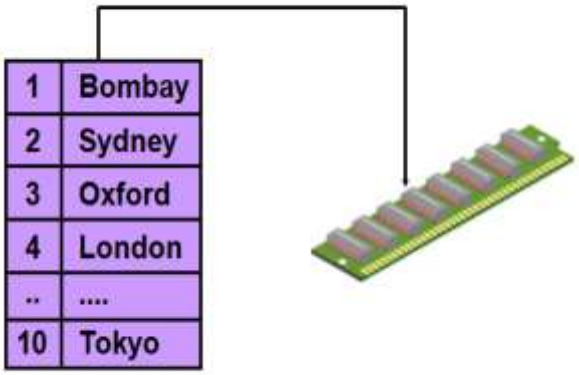
```
FOR i in 1..table_count LOOP
DBMS_OUTPUT.PUT_LINE (offices (i));
END LOOP;
END;
```

```
anonymous block completed
Bombay
Tokyo
Singapore
Oxford
```

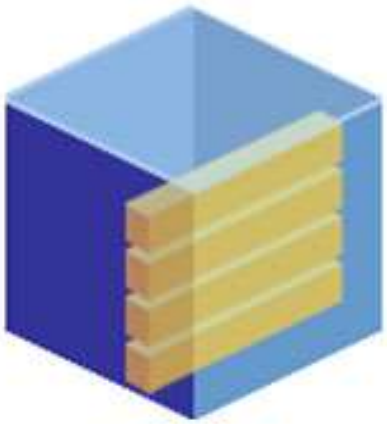
هيكلية الـ VARRAY .

أن الهيكلية العامة لهذا النوع تكون مشابهة إلى الجداول Tables . عدا أن الـ VARRAY عند بنائها يتم تحديد الحجم الخاص بها مسبقا ، أي أنها تملك حد أعلى ثابت من العناصر ممكن أن تصل إليه (أي أنها تكون مشابهة للمصفوفات في لغة الـ c++) ، وان اكبر حجم نستطيع خزنه باستخدام هذا النوع هو (2GB) . وان الصيغة العامة لتعريف هذا النوع موضحة في المثال أدناه.

```
SQL>TYPE location_type IS VARRAY (3) OF
locations.city%TYPE;
offices location_type;
```



المؤشر Cursor.



أن أي جملة SQL تنفذ بواسطة الـ Oracle Server تمتلك مؤشر مستقل مرفق ، بحيث انه يوجد هنالك نوعان من ذلك المؤشر هما.

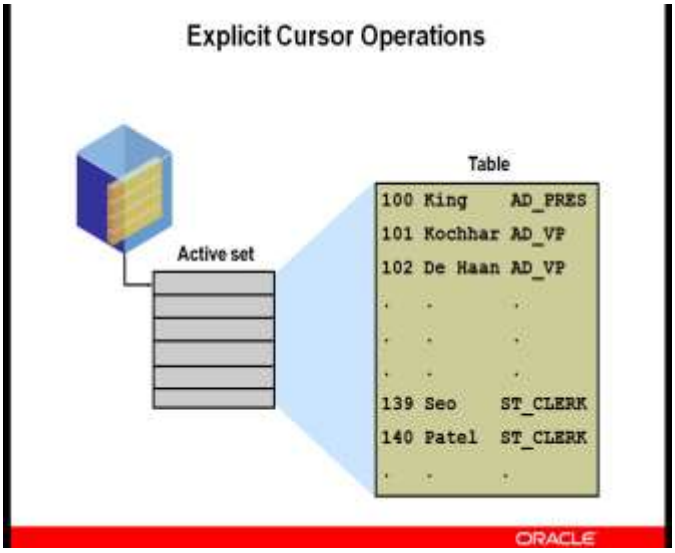
1- المؤشر الضمني Implicit Cursor .

أن هذا النوع يعرف ويدار بصورة أوتوماتيكية من قبل الـ PL/SQL ولكل عبارات الـ DML

والـ Select Statement .

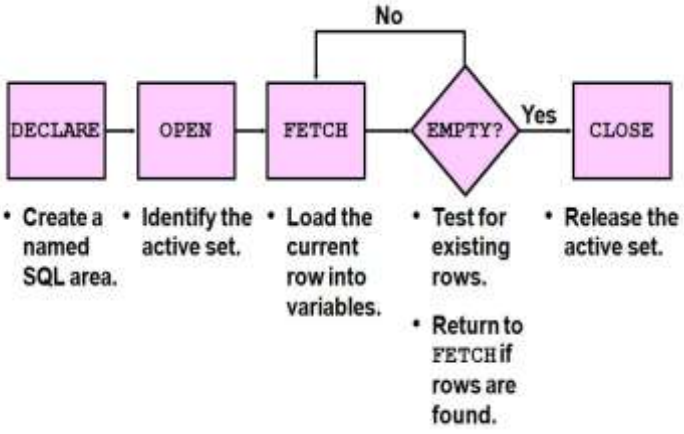
2- المؤشر الصريح Explicit Cursor.

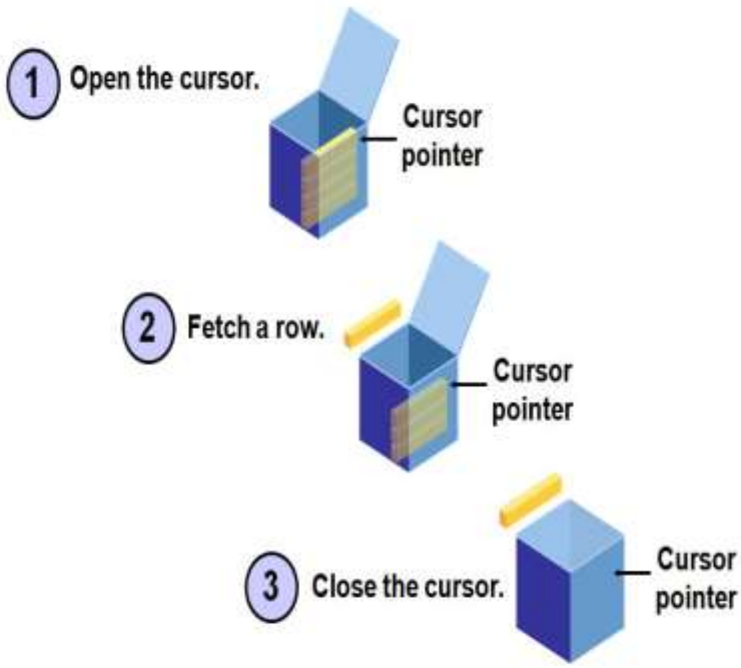
أن هذا النوع يعرف ويدار بصورة يدوية من قبل المبرمج Programmer .



السيطرة على المؤشرات الصريحة.

أن المخططات الموضحة والمبينة في أدناه توضح كيفية السيطرة على المؤشرات الصريحة الني يتم خلقها ومعالجتها من قبل المبرمج .





تعريف المؤشر Declare Cursor.

أن الصيغة العامة لتعريف المؤشر الصريح هي كما يلي.

Syntax:

```
CURSOR cursor_name IS
     select_statement;
```

Examples:

```
DECLARE
  CURSOR emp_cursor IS
  SELECT employee_id, last_name FROM employees
  WHERE department_id =30;
```

```
DECLARE
  locid NUMBER:= 1700;
  CURSOR dept_cursor IS
  SELECT * FROM departments
  WHERE location_id = locid;
...
```

فتح المؤشر Opening the Cursor.

أن المثال التالي المبين في أدناه يوضح كيفية فتح المؤشر بعد تعريفه لغرض استخدامه.

```
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
...
BEGIN
  OPEN emp_cursor;
```

جلب البيانات من المؤشر.

أن المثال التالي المبين في الصيغة أدناه يوضح كيفية تعريف المؤشر ومن ثم فتح ذلك المؤشر وبعد ذلك كيفية جلب البيانات منه .

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
  empno employees.employee_id%TYPE;
  lname employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO empno, lname;
  DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  ...
END;
/
```

مثال.

المثال التالي يوضح كيفية تعريف المؤشر ومن ثم كيفية فتحه بعد ذلك يبين كيفية جلب المتغيرات التي يحتويها بواسطة استخدام الـ Loop.

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE  department_id =30;
  empno employees.employee_id%TYPE;
  lname employees.last_name%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO empno, lname;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  END LOOP;
  ...
END;
/
```

غلق المؤشر Close the Cursor.

بعد الانتهاء من عمل المؤشر يجب غلقه وذلك لان عدم غلقه سوف يؤدي إلى هدم التسلسل الخاص بالبرنامج، وان المثال المبين في أدناه يوضح كيفية غلق المؤشر بعد الانتهاء منه.

```
...
  LOOP
    FETCH emp_cursor INTO empno, lname;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
  END LOOP;
 CLOSE emp_cursor;
END;
/
```

استخدام المؤشرات مع القيود.

من الممكن استخدام المؤشر Cursorمع القيود Recordsلأجل جلب البيانات المخزنة في القيود، وان المثال أدناه يوضح كيفية استخدام المؤشر مع القيود.

```
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE  department_id =30;
   emp_record  emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
  ...
```

استخدام المؤشر مع الـ Loop.

من الممكن أيضا استخدام المؤشرات مع الـ Loop ، وان الصيغة العامة لاستخدام المؤشرات Cursors مع الـ Loops مبينه في أدناه.

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

Example:

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR emp_cursor IS
   SELECT employee_id, last_name FROM employees
   WHERE department_id =30;
BEGIN
   FOR emp_record IN emp_cursor
    LOOP
     DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
     ||' ' ||emp_record.last_name);
    END LOOP;
END;
/
```

صفات المؤشر Cursor Attributes .

هنالك عدة صفات للمؤشر وهذه الصفات موضحة في الجدول أدناه.

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returns a row; complement of %NOTFOUND
%ROWCOUNT	Boolean	Evaluates to the total number of rows returned so far

صفة %ISOPEN.

أن هذا الصفة تقوم بفحص المؤشر هل هو مفتوح أم لا، فإذا كان مفتوح يقوم بجلب البيانات الموجودة داخله ، أما أذا كان غير مفتوح فلا يقوم بعمل أي شيء والمثال التالي يوضح ذلك.

Example:

```
IF NOT emp_cursor%ISOPEN THEN
   OPEN emp_cursor;
END IF;
LOOP
  FETCH emp_cursor...
```

صفة %ROWCOUNT and %NOTFOUND :

أن الـ %ROW COUNT تقوم بإرجاع رقم السطر الحالي الذي يؤشر عليه المؤشر ، أما بالنسبة إلى %NOTFOUND فانه يقوم بفحص القيد هل هو موجود أم لا ، وكما موضح في المثال أدناه.

```
SET SERVEROUTPUT ON
DECLARE
  empno  employees.employee_id%TYPE;
  ename  employees.last_name%TYPE;
  CURSOR emp_cursor IS SELECT employee_id,
  last_name FROM employees;
BEGIN
  OPEN emp_cursor;
  LOOP
   FETCH emp_cursor INTO empno, ename;
   EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                    emp_cursor%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(TO_CHAR(empno)
                        ||' '|| ename);
  END LOOP;
  CLOSE emp_cursor;
END ;
/
```

المؤشر المستخدم مع عبارة الـ For أن هذا النوع من المؤشرات لا يحتاج إلى تعريف ، وان الصيغة العامة لاستخدام مثل هذا النوع موضحة في المثال التالي.

```
SET SERVEROUTPUT ON
BEGIN
  FOR emp_record IN (SELECT employee_id, last_name
   FROM employees WHERE department_id =30)
  LOOP
   DBMS_OUTPUT.PUT_LINE( emp_record.employee_id ||'
   '||emp_record.last_name);
  END LOOP;
END;
/
```

استخدام المؤشر مع المعاملات Cursor With Parameters .

من الممكن استخدام مجموعة من المعاملات مع المؤشر والصيغة العامة لذلك موضحة في المثال ادناه.

Syntax:

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

To open the cursor we use: -

```
OPEN  cursor_name(parameter_value,.....) ;
```

Example:

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR   emp_cursor (deptno NUMBER) IS
   SELECT  employee_id, last_name
   FROM    employees
   WHERE   department_id = deptno;
   dept_id NUMBER;
   lname   VARCHAR2(15);
BEGIN
  OPEN emp_cursor (10);
  ...
  CLOSE emp_cursor;
  OPEN emp_cursor (20);
  ...
```

استعمال المؤشر Cursor مع الـ Sub queries.

من الممكن استعمال المؤشرات مع الاستعلامات الجزئية ، وان المثال التالي يوضح كيفية استخدام تلك الـ Cursor مع الـ Sub queries .

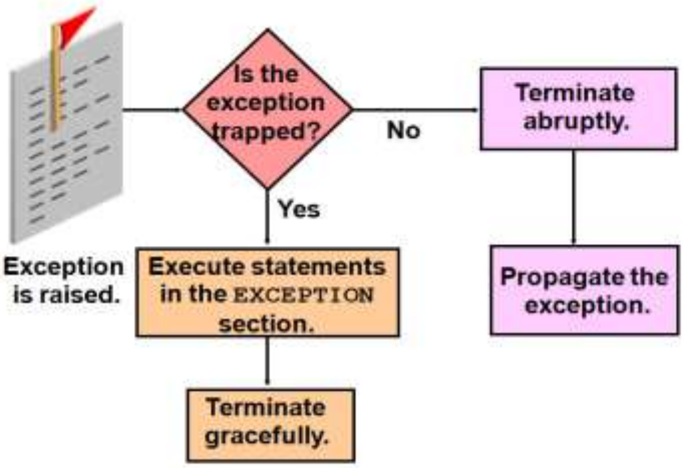
Example:

```
DECLARE
  CURSOR my_cursor IS
    SELECT t1.department_id, t1.department_name,
           t2.staff
    FROM   departments t1, (SELECT department_id,
                                   COUNT(*) AS staff
                            FROM employees
                            GROUP BY department_id) t2
    WHERE t1.department_id = t2.department_id
    AND   t2.staff >= 3;
...
```

معالجة الاعتراضات Handling Exception.

أن الـ Exception هو عبارة عن خطأ Error يظهر أثناء تنفيذ البرنامج نتيجة حدوث خلل في التنفيذ، وان تلك الأخطاء من الممكن أن تظهر أما ضمنيا بواسطة الـ Oracle Server أو بصورة صريحة من قبل البرنامج . أما بالنسبة لمعالجة تلك الأخطاء فان هنالك طريقتين للمعالجة الأولى أما من خلال حصرها مع المعالج Handler أو من خلال نشرها في البيئة التي تعمل فيها.

وان المخطط التالي يوضح كيفية معالجة تلك الأخطاء في حال ظهورها .



أنواع الاعتراضات Exception Types.

هنالك نوعان رئيسيان من الاعتراضات التي من الممكن أن تظهر عند تنفيذ الكود الخاص ببرنامج الاوراكل هما:

1- الاعتراض الضمني Implicit Exception ويوجد نوعان منه :

- ✓ الاعتراض المعرف من قبل الاوراكل Predefined Oracle Server.
- ✓ الاعتراضات الغير معرفة من قبل الاوراكل Non _ Predefined oracle server.

2- الاعتراضات الصريحة Explicit Exception.

معالجة الاعتراضات Trapping Exception.

أن الصيغة العامة لمعالجة أي Exception في برنامج الاوراكل مبينة كما يلي:

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

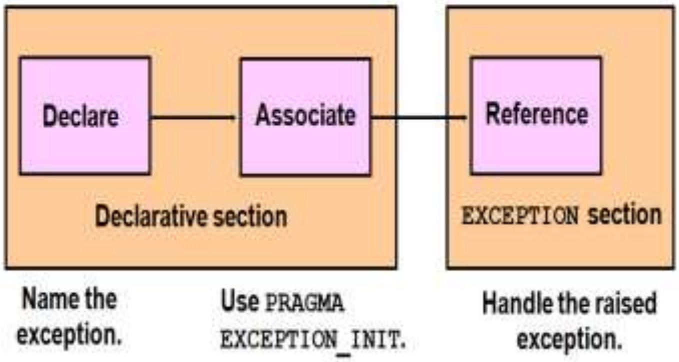
ملاحظات مهمة :

- ✓ أن كل Exception يجب أن يبدأ بالكلمة المحجوزة EXCEPTION والتي تدل على بداية معالجة الاستثناء.
- ✓ من الممكن معالجة أكثر من خطأ في كل Exception.
- ✓ على الأقل يجب معالجة خطأ واحد قبل الخروج من الـ block.
- ✓ أن الجزء WHEN OTHER يكتب دائما في نهاية الـ block.

هنالك بعض الأخطاء التابعة للنوع الأول والمعرفة من قبل الـ Oracle Server ومنها:

- ✓ NO_DATA_FOUND.
- ✓ TOO_MANY_ROWS.
- ✓ INVALIDE_CURSOR.
- ✓ ZERO_DIVIDE.
- ✓ DUP_VAL_ON_INDEX.

أما بالنسبة إلى None Predefined oracle server errors ، المخطط التالي يوضح كيفية تعريف ومعالجة تلك الأخطاء والبرنامج الذي يتبعه يوضح كيفية كتابة block كامل لمعالجة تلك الـ Exception.



البرنـامج التـالي يوضـح كيفيـة كتابـة block كامـل الـذي يتضـمن تعريـف الـ Exception ومعالجته وكيفية إنهاء الـ block.

```
SET SERVEROUTPUT ON
DECLARE
 insert_excep EXCEPTION;                    (1)
 PRAGMA EXCEPTION_INIT
 (insert_excep, -01400);                    (2)
BEGIN
 INSERT INTO departments
 (department_id, department_name) VALUES (280, NULL);
EXCEPTION                                   (3)
 WHEN insert_excep THEN
 DBMS_OUTPUT.PUT_LINE('INSERT OPERATION FAILED');
 DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

دوال الـ Exception.

هنالك نوعان من الدوال المستخدمة مع الـ Exception هي :

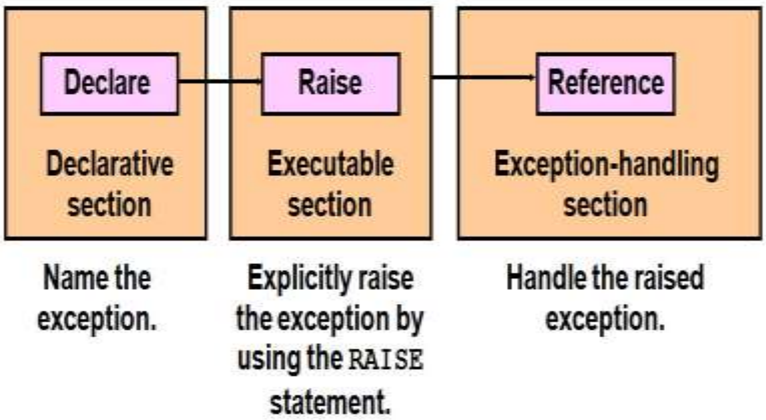
- ✓ دالة الـ SQL CODE : تستخدم هذه الدالة القيمة الرقمية للخطأ الذي يظهر ، حيث أن لكل Error يوجد هنالك رقم خاص به يعرف من قبل الـ Oracle Server ويدعى بالـ Error Code.
- ✓ دالة SQL ERRM وظيفة هذه الدالة هي إرجاع رسالة تتضمن نوع الخطأ الحـادث حاليا ورقمـه.والبرنامج التـالي يوضـح كيفيـة اسـتخدام تلـك الـدوال أعلاه.

Example:

```
DECLARE
  error_code      NUMBER;
  error_message   VARCHAR2(255);
BEGIN
...
EXCEPTION
...
  WHEN OTHERS THEN
    ROLLBACK;
    error_code := SQLCODE ;
    error_message := SQLERRM ;
   INSERT INTO errors (e_user, e_date, error_code,
   error_message) VALUES(USER,SYSDATE,error_code,
   error_message);
END;
/
```

الأخطاء المعرفة من قبل المستخدم User defined Exceptions.

هي الأخطاء التي يتم تعريفها من قبل المستخدم والتي يتوقع حدوثها في البرنامج ، ففي تلك الحالة يجب على المستخدم تعريف تلك الأخطاء من خلال block خاص بها وكيفية معالجتها في حالة حدوثها إضافة إلى كيفية إنهاء ذلك الـ block والمخطط التالي يوضح كيفية كتابة تعريف تلك الـ Exceptions ومعالجتها.



البرنامج التالي يوضح كيفية كتابة block كامل من قبل المستخدم لتعريف الأخطاء المتوقع حدوثها وكيفية معالجتها ، وكيفية إنهاء ذلك الـ block.

```
ACCEPT deptno PROMPT 'Please enter the department number:'
ACCEPT name   PROMPT 'Please enter the department name:'
DECLARE
  invalid_department EXCEPTION;            --(1)
  name VARCHAR2(20):='&name';
  deptno NUMBER :=&deptno;
BEGIN
  UPDATE  departments
  SET     department_name = name
  WHERE   department_id = deptno;
  IF SQL%NOTFOUND THEN
    RAISE invalid_department;              --(2)
  END IF;
  COMMIT;
EXCEPTION
  WHEN invalid_department THEN             --(3)
    DBMS_OUTPUT.PUT_LINE('No such department id.');
END;
/
```

العمل مع الـ Procedure & Function .

هي عبارة عن blocks تكتب في الـ PL/SQL من قبل المستخدم لتنفيذ عمليات معينة ، تسمى بالـ PL/SQL blocks وتستدعى كـ PL/SQL Subprograms . وان الـ block الخاص بهما يتكون من الأجزاء الثلاثة التالية:

1- الجزء التعريفي Declaration Section ويبدأ بكلمة Declare ويستخدم لتعريف جميع المتغيرات المعرفة داخل الـ block وهو جزء اختياري من الممكن كتابته أو لا عند عدم الحاجة إليه، حيث انه لا يؤثر على سير عمل البرنامج.

2- الجزء التنفيذ Section Execution وهو الجزء الأساسي ويبدأ بكلمة BEGIN وينتهي بكلمة END ويتم فيه كتابة العمليات المطلوب تنفيذها ، وعملية كتابته تكون اجباريه حيث لا يمكن تنفيذ الـ PROCEDURE or FUNCTION بدونه.

3- الجزء الخاص بمعالجة الاعتراضات Exception Handling Section ويتم فيه كتابة الكود الخاص بكيفية معالجة الأخطاء التي من الممكن أن تظهر بالبرنامج ، علما انه جزء اختياري حيث من الممكن كتابته أو لا.

أن الصيغة العامة لكتابة اي Procedure موضحة في أدناه.

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(argument1 [mode1] datatype1,
  argument2 [mode2] datatype2,
  . . .)]
IS|AS
procedure_body;
```

Example.

```
CREATE TABLE dept AS SELECT * FROM departments;
CREATE PROCEDURE add_dept IS
 dept_id dept.department_id%TYPE;
 dept_name dept.department_name%TYPE;
BEGIN
 dept_id:=280;
 dept_name:='ST-Curriculum';
 INSERT INTO dept(department_id,department_name)
 VALUES(dept_id,dept_name);
 DBMS_OUTPUT.PUT_LINE(' Inserted '||
  SQL%ROWCOUNT ||' row ');
END;
/
```

تنفيذ الـ Procedure.

أن المثال المبين في أدناه يوضح كيفية استدعاء الـ Procedure وتنفيذه .

```
BEGIN
 add_dept;
END;
/
SELECT department_id, department_name FROM dept
WHERE department_id=280;
```

```
anonymous block completed
 Inserted 1 row
```

	DEPARTMENT_ID	DEPARTMENT_NAME
1	280	ST-Curriculum

أن الصيغة العامة لكتابة أي Function موضحة في أدناه.

```
CREATE [OR REPLACE] FUNCTION function_name
 [(argument1 [mode1] datatype1,
  argument2 [mode2] datatype2,
  . . .)]
RETURN datatype
IS|AS
function_body;
```

Example.

```
CREATE FUNCTION check_sal RETURN Boolean IS
 dept_id employees.department_id%TYPE;
 empno   employees.employee_id%TYPE;
 sal     employees.salary%TYPE;
 avg_sal employees.salary%TYPE;
BEGIN
 empno:=205;
 SELECT salary,department_id INTO sal,dept_id
 FROM employees WHERE employee_id= empno;
 SELECT avg(salary) INTO avg_sal FROM employees
 WHERE department_id=dept_id;
 IF sal > avg_sal THEN
  RETURN TRUE;
 ELSE
  RETURN FALSE;
 END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   RETURN NULL;
END;
/
```

ولتنفيذ أي Function فان ذلك يتم بالطريقة التالية:

```
SET SERVEROUTPUT ON
BEGIN
 IF (check_sal IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal) THEN
 DBMS_OUTPUT.PUT_LINE('Salary > average');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Salary < average');
 END IF;
END;
/
```

```
anonymous block completed
Salary > average
```

من المعلوم أن فرق الـ Procedure عن الـ Function هو أن الـ Function نستطيع إرسال معاملات إليها وتقوم بإرجاع نتائج ، أما الـ Procedure فلا حيث انه مصمم لتنفيذ عمليات محددة . ولمعرفة كيفية تمرير المعاملات إلى الـ Function فان ذلك موضح في أدناه.

```
DROP FUNCTION check_sal;
CREATE FUNCTION check_sal(empno employees.employee_id%TYPE)
RETURN Boolean IS
 dept_id employees.department_id%TYPE;
 sal      employees.salary%TYPE;
 avg_sal employees.salary%TYPE;
BEGIN
 SELECT salary,department_id INTO sal,dept_id
 FROM employees WHERE employee_id=empno;
 SELECT avg(salary) INTO avg_sal FROM employees
 WHERE department_id=dept_id;
 IF sal > avg_sal THEN
  RETURN TRUE;
 ELSE
  RETURN FALSE;
 END IF;
EXCEPTION ...
...
```

المثال التالي يوضح كيفية تنفيذ تلك الـ Function مع المعاملات.

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Checking for employee with id 205');
 IF (check_sal(205) IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal(205)) THEN
 DBMS_OUTPUT.PUT_LINE('Salary > average');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Salary < average');
 END IF;
DBMS_OUTPUT.PUT_LINE('Checking for employee with id 70');
 IF (check_sal(70) IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal(70)) THEN
 ...
 END IF;
END;
/
```

<u>**السيرة الذاتية:**</u>

الاسم الثلاثي : وسام علي حسين سلمان الخزعلي .

التولد : بغداد / 1977 .

الحالة الاجتماعية : متزوج .

التحصيل الدراسي : بكالوريوس علوم الحاسبات / جامعة بغداد .

سنة التخرج : 2002-2003.

ماجستير علوم الحاسبات وتقنية المعلومات SHIATS University / India

سنة التخرج 2013 .

عنوان اطروحة الماجستير :

Design&Implementationof Students Information Management Systems for Kerbela University.

اللقب العلمي : مدرس مساعد.

رقم الموبايل : 009647724918820

عنوان السكن : كربلاء / حي الحر .

اللغات التي أتقنها : اللغة العربية + اللغة الانكليزية.

البريد الالكتروني: wesali77@yahoo.com .

wissamali77@gmial.com.

الشهادات التي حصلت عليها:

1- بكالوريوس علوم حاسبات / جامعة بغداد.

2- ماجستير علوم الحاسبات وتقنية المعلومات / SHIATS University India

3- شهادة IC3 العالمية بتقدير (جيد جدا).

4- شهادة التوفل في اللغة الانكليزية.

5- شهادة Web Design من أكاديمية Microsoft / فرع جامعة القادسية شهادة تصميم قواعد البيانات Data Base من أكاديمية Microsoft / فرع جامعة القادسية.

6- حصلت على شهادة في نظم المعلومات الجغرافية GIS من قبل منظمة دعم الحكومات المحلية الـ RTI.

7- حصلت على دورة مكثفة في برنامج Microsoft Office من قبل برنامج دعم الحكومات المحلية RTI.

8- حصلت على دورة مكثفة في الـ Java programming من قبل Crest institute في الهند – حيدر أباد.

9- حصلت على دورة مكثفة في الـ Oracle & SQL Server program من قبل IVAIS institute في الهند – حيدر أباد.

10- حصلت على دورة مكثفة في الـ Visual Basic.NET program من قبل Crest institute في الهند – حيدر أباد.

11- حصلت على دورة مكثفة في اللغة الانكليزية من قبل IELETS guru في الهند – حيدر أباد.

12- حصلت على شهادة 1 Database Administrator من جامعة أوراكل الأميركية Oracle University .

13- حصلت على شهادة 2 Database Administrator من جامعة أوراكل الأميركية Oracle University.

14- حصلت على شهادة تصميم المواقع الالكترونية باستخدام تقنية الـ HTML + Java Script من كلية البنجاب للمعلوماتية Sushant IT Collage .

15- حصلت على شهادة تصميم المواقع الالكترونية باستخدام تقنية الـ ASP.Net 2013 من كلية البنجاب للمعلوماتية Sushant IT Collage

16- حصلت على شهادة في صيانة اللابتوب من قبل SCS India institute في حيدر أباد / الهند.

17- حصلت على شهادة اجتياز دورة تقنيات طرائق التدريس من وزارة التعليم العالي والبحث العلمي العراقية / جامعة كربلاء.

18- حاصل على شهادة مشاركة في دورة (Web Application Hacking & Security) المقامة في معهد الكفيل لتقنية المعلومات وتطوير المهارات / العتبة العباسية المقدسة.

19- حاصل على مجموعة من شهادات الخبرة من المؤسسات والشركات التي عملت فيها سابقا .

<u>عضوية المنظمات.</u>

1- عضو نقابة المعلمين العراقيين.

2- عضو نقابة المبرمجين العراقيين.

3- عضو رابطة التدريسيين التربويين في العراق.

4- عضو في الملتقى العلمي لـ SHIATS University / India.

5- عضو في اكاديمية NIIT Computer Education & Training Center / Hyderabad / India.