القسم الخامس

الفصل الثانى عشر

مقدمة عن قواعد البيانات وأوامر لغة الاستعلامات SQL

فى هذا الفصل Programming Concepts 12

فى هذا الدرس نشرح استخدامات لغة الاستعلامات SQL وذلك من **خلال النقاط التالية :**

- ما المقصود بقواعد البيانات
- مزايا استعمال الكمبيوتر فى إنشاء قاعدة بيانات
- تركيب قاعدة البيانات
- كيفية استعمال MS ACCESS كمثال بسيط
- عناصر قاعدة البيانات
- التكامل فى قاعدة البيانات
- تصحيح تصميم الجداول NORMALIZATION
- ما هى لغة الاستعلامات SQL
- أجزاء لغة SQL
  - لغة التعريفات DDL
  - لغة التحكم DCL
  - لغة الوصول للبيانات DML

قواعد البيانات database

فى الفترات القديمة والأولى لتطوير الكمبيوتر لم يكن لقواعد البيانات مكان بل كانت البيانات تحفظ فى ملفات تسمى ملفات مستوية Flat Files ، ولكن مع التطور اصبح مفهوم قواعد البيانات ، من أهم المفاهيم ، وهو تسجيل البيانات فى شكل منظم عبارة عن حقول وسجلات فى جداول ، ومجموعة الجداول تشكل قاعدة بيانات وتوفر عمليات متعددة مثل البحث ، والترتيب ، والعمليات الحسابية على الحقول وغيرها .

وبدأت قواعد البيانات قديما برامج بسيطة مثل dbase III+ ، وطورت بعد ذلك لتشتمل الآن على الكثير من البرامج منها :

1- برنامج MS Access الذى يأتى ضمن مجموعة MS Office

2-قاعدة بيانات Oracle والتى تمثل أقوى قاعدة بيانات حتى الآن

3- MS SQL server والذى أصدرته مايكروسوفت ليضاهى قاعدة بيانات Oracle

4- قاعدة بيانات DB2 من شركة IBM ، والكثير من لغات قواعد البيانات

ما المقصود بقواعد البيانات WHAT IS DATABASE

بصرف النظر عن استعمال الكمبيوتر لو نظرت ورقيا تجد أن أى مؤسسة تحتفظ بالبيانات فى شكل قاعدة بيانات فمثلاً.

بيانات العاملين

تجد إدارة شئون العاملين تحتفظ بملف لكل موظف يحتوى هذا الملف على بياناته الشخصية ومؤهلاته والشهادات الحاصل عليها حتى الآن. وتواريخ الترقي وكذلك العلاوات والجزاءات والدورات الحاصل عليها وهذه تسمى قاعدة بيانات فى أبسط صورة.

صديقى

اذا كنت على دراية بأساسيات قواعد البيانات واستخدام قواعد البيانات MS ACCESS أو SQL SERVER أو ORACLE (كما هو متوقع) يمكنك الانتقال إلى الفقرة المناسبة لمستواك حتى لا تشعر بالملل حيث أننا فى هذه الفقرة نراجع أساسيات فى قواعد البيانات يجب الاتفاق عليها

بيانات العملاء

فى إدارة المبيعات تلاحظ الإدارة تحتفظ بيانات العملاء الشخصية وحركات البيع وحجم المبيعات والمواقف المالية وغيرها ، وهذه تسمى قاعدة بيانات.

وهناك الكثير من الأمثلة الأخرى ، ولذلك تلاحظ أنه لا غنى عن وجود قواعد بيانات ولو ورقية ، ولكن السؤال لماذا نحول قاعدة البيانات الورقية إلى كمبيوتر؟

الإجابة:

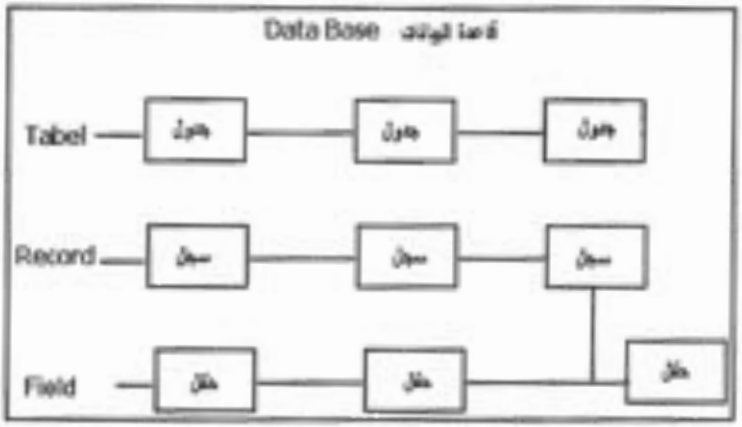
- صعوبة البحث فى قاعدة البيانات الورقية والوقت المستغرق فى حين يتوفر ذلك فى قاعدة البيانات الكمبيوتر بسهولة.
- صعوبة الحصول على معلومات تحليلية من البيانات فى حين يتوفر ذلك فى قاعدة بيانات الكمبيوتر.
- صعوبة التأكد من دقة المعلومات لاحتمال الخطأ فى حين توفر الدقة فى معلومات قاعدة بيانات الكمبيوتر وكثير من المزايا التى سوف نشرحها.

مزايا استعمال الكمبيوتر فى إنشاء قاعدة بيانات

- الاحتفاظ بكم هائل من البيانات بعيداً عن الأوراق وسهولة عمل نسخ كثيره من هذه البيانات.
- سهولة ربط البيانات ببعضها مثل بيانات العاملين وبيانات أصناف المخازن.
- سرعة البحث الهائلة عن البيانات التي لا تقارن مع قاعدة البيانات الورقية.
- إمكانية استخلاص البيانات (المعلومات) التحليلية من كثير من البيانات مثل بيانات موظفين على درجة معينة أو موظفين حاصلين على دورة معينه ..إلخ.
- دقة المعلومات المستخلصة حيث يخرج الكمبيوتر ما تم تسجيله.
- إمكانية إجراء العمليات المختلفة على جدول أو أكثر من البيانات فى وقت قصير مثل زيادة أجر الموظفين بنسبة معينه أو صرف مكافأة لهم.
- إمكانية استخراج التقارير بأشكال مختلفة مثل الرسوم البيانية وغيرها.

تركيب قاعدة البيانات

تأخذ قاعدة البيانات التركيب التالي (1-12):



شكل(1-12)

من هذا الشكل نلاحظ أن:

- قاعدة البيانات مثلا العاملين تتكون من جداول مثل جدول البيانات الأساسية - جدول الإجازات - جدول الدورات. [TABLES]
- والجدول (TABEL) يتكون من سجلات (RECORDS) حيث يمثل كل موظف داخل الجدول بسجل ويسمى أحيانا ROW أي صف.
- والسجل (RECORD) يتكون من حقول (FIELDS) حيث يتكون سجل الموظف من حقول مثل كود الموظف، اسم الموظف العنوان، التلفون، تاريخ الميلاد ..إلخ ، أي تأخذ قاعدة البيانات التركيب التالي :

DATA BASE –TABLES – RECORDS- FIELDS

كيفية استعمال MS ACCESS كمثال بسيط

بعد شرح مقدمات عن قاعدة البيانات وأجزائها نعال معنا نشاهد ذلك عملياً باستعمال البرنامج MS ACCESS وهو أحد البرامج الموجودة في حزمة البرامج (MICROSOFT OFFICE) والتي أنتجتها شركة مايكروسوفت ولا يخلو جهاز كمبيوتر الآن من هذه المجموعة ، ووظيفة برنامج MS ACCESS إنشاء قواعد البيانات والتعامل معها

فإذا كنت تجيد استعماله انتقل إلى الخطوة التالية .

خطوات الاستعمال

قم بتشغيل برنامج MS ACCESS تحصل على أول شاشة كما بالشكل التالي (2-12):

شكل(2-12)

في هذا الشكل يعرض لك البرنامج ثلاثة اختيارات هي:

- قاعدة بيانات فارغة BLANK DATA BASE.
- وفيها لا يوفر البرنامج أي مساعدات وعليك أنت إنشاء كل شيء حسب رغبتك كما سبق شرح ذلك.
- معالج قاعدة البيانات DATA BASE WIZARD.
- وفيها يوفر البرنامج معالج عبارة عن مجموعة من الخطوات تساعدك في إعداد قاعدة بيانات حسب مجموعة كبيرة من قواعد البيانات المقترحة.
- فتح قاعدة بيانات موجودة OPEN EXISTING DATA BASE.
- وفيها يتم فتح ملف قاعدة بيانات موجودة بالفعل وتم إنشائها من قبل.
- وللسهولة ابدأ بالاختيار الثاني وهو DATA BASE WIZARD (معالج قواعد البيانات) ثم انقر الزر OK.

- تحصل على مجموعة من قواعد البيانات الجاهزة.
- فى هذا الشكل تجد مثلاً قاعدة بيانات بالاسم ADDRESS BOOK وهى قاعدة بيانات لعناوين وتليفونات الأصدقاء، وتجد أيضا قاعدة بيانات بالاسم INVENTORY CONTROL وهى خاصة بنظام مخزون وإضافة، وقاعدة بيانات بالاسم STUDENTS & CLASSES وهى خاصة بمتابعة فصل تدريب به طلبه والكثير من قواعد البيانات الأخرى.

دعنا نجرب إحداها ولتكن ADDRESS BOOK كما يلى:

1- اختر ADDRESS BOOK ثم انقر الزر OK.

2- تحصل على مربع لكتابة اسم لها وافق على الاسم الموجود ثم انقر CREATE إنشاء.

3- تحصل على الشاشة الأولى من المعالج.

تابع خطوات المعالج بتحديد البيانات التى سوف توضع فى قاعدة البيانات لتضيف أو تحذف منها ما تريد.

- ثم اضغط الزرالتالى (NEXT) حتى تحصل على شاشة لكتابة عنوان انقر الزر NEXT.

فى النهاية انقر الزر إنهاء (FINISH) فيتم إنشاء قاعدة البيانات وتشغيلها كما فى الشكل التالى:

- تظهر الشاشة الرئيسية لقاعدة بيانات العناوين والتليفونات.
- انقر الزر إدخال / عرض العناوين تحصل على الشاشة الأساسية لتسجيل البيانات.
- جرب استعمال هذه الشاشة فى تسجيل بيانات تهمك ثم إغلق الشاشة تلاحظ العودة إلى الشاشة الرئيسية.
- انقر الزر معاينة التقارير تحصل على شاشة فرعية.
- اختر أي تقرير تلاحظ عرض البيانات التى أدخلتها من قبل.
- جرب استعمال باقى اختيارات البرنامج.
- اغلق البرنامج وعد إلى برنامج النوافذ.

النوع
نص TEXT
رقم NUMBER
تاريخ / وقت
عملة
ترقيم تلقائي
نعم/ لا
كائن OLE
ارتباط تشعبي
معالج البحث
مذكرة

الحقل المميز PRIMARY KEY

هو الحقل الذى لن تتكرر قيمته فى الجدول مثل كود الموظف أو الرقم القومى.

خصائص الحقول FIELD PROPERTIES

بالإضافة لأنواع بيانات الحقول يوجد أيضا مواصفات يتم تحديدها لكل حقل لزيادة التحكم فى بيانات هذا الحقل تظهر فى النصف السفلى من الشاشة عند النقر على الحقل فى مربع التصميم كما بالشكل (5-12).

شكل(5-12)

والجدول التالي يعرض الخصائص الأساسية للحقول ومعنى كل واحدة.

الخاصية	المعنى
حجم الحقل	عدد الحروف
عنوان	عنوان يظهر بدلا من اسم الحقل
القيمة الافتراضية	قيمة إذا لم يسجل المستخدم قيمة
قاعدة التحقق من الصحة	شرط للقيمة المدخلة
مطلوب	لابد للمستخدم أن يدخل قيمته
مفهرس	إنشاء جدول فهرس مرتب على أساسه

تدريب على الجداول

من فضلك قم بإنشاء جدول الموظفين السابق EMP2 بتركيب جديد وأنواع حقول جديدة كما يلي:

اسم الحقل	نوع البيانات	عدد الحروف
EMP_NO	رقم NUMBER	-
EMP_NAME	نص TEXT	30
EMP_PHONE	نص TEXT	12
EMP_BDATE	تاريخ/ وقت DATE/TIME	
EMP_PHOTO	كائن OLE	-
EMP_STATUS	نعم/ لا BOOL	-

ما هو ملف الفهرس INDEX؟

هو جدول مصاحب للجدول الأساسي يتم فيه حفظ رقم السجل والحقل المفهرس على أساسه الجدول فمثلاً عند إنشاء فهرس حسب حقل NAME يتم إنشاء جدول فهرس به حقل NAME مرتب ترتيباً أبجديا ومعه رقم السجل.

يستعمل عند البحث بالحقل المفهرس INDEX على أساسه حيث يتم إنشاء هذا الجدول المفهرس من خصائص الحقل وذلك بتحديد خاصية فهرس (INDEXED).

ويوجد من جدول الفهرس نوعين.

- الأول بدون تكرار وبالتالى لا يسمح بتكرار قيمة هذا الحقل
- والثانى بتكرار وفيه يتم ترتيب البيانات حسب الحقل مع السماح بتكرار قيمته

التكامل فى قواعد البيانات INTEGRITY

من الخصائص التى تتمتع بها البيانات فى قواعد البيانات DATABASES التكامل فى قواعد البيانات والذى يتم على أساسه قياس مدى الدقة ACCURACY والصحة CORRECTNESS ونتناول ذلك من خلال النقاط التالية:

أ- المفاتيح KEYS:

ماهى المفاتيح KEYS وما هى أنواعها وما هى أهميتها.

ب- القيود التكاملية INTEGRITY CONSTRAINTS:

نتناول فى هذه الفقرة التعريف الخاص بالقيود CONSTRAINTS وأهميتها وأنواعها وتطبيق ذلك على بيانات فعلية.

ج- خرق العمليات فى قاعدة البيانات OPERATION VIOLATION

ونعرض فيها كيف يتم خرق القيود CONSTRAINTS من خلال العمليات المختلفة وأساليب التغلب عليها و التعامل معها.

أ- المفاتيح KEYS:

المفاتيح KEYS من الموضوعات الهامة فى قواعد البيانات DATABASES وذلك لكونها العنصر المتحكم فى العلاقات RELATIONSHIPS بين الجداول وبعضها البعض.

وتأخذ الانواع التالية:

1- المفتاح الرئيسى(المميز) PRIMARY KEY.

2- المفتاح الخارجى FOREIGN KEY.

3- المفاتيح المرشحة لتكون مفاتيح رئيسية CANDIDATE KEYS.

4- المفتاح SUPER KEY.

1. المفتاح الرئيسى PRIMARY KEY :

المفتاح الرئيسى هو أحد حقول الجدول TABLE يحدد على أساس الحقل الذى لا تتكرر أى قيمة من قيمه ، أى تكون له قيم وحيدة UNIQUE VALUES ، وأيضا لا يسمح بترك قيمة هذا الحقل خالية NULL VALUE.

مثلا، حقل الرقم القومى فى جدول المواطنين حيث لا يمكن أن يأخذ أكثر من مواطن نفس الرقم القومى.

2. المفتاح الخارجى FOREIGN KEY :

المفتاح الخارجى FOREIGN KEY الخاص بجدول TABLE ما هو عبارة عن حقل FIELD يشير إلى حقل آخر(هو المفتاح الرئيسى PRIMARY KEY) فى جدول آخر مع مراعاة أن تكون قيمه موجودة فى الرئيسى.

ويظهر ذلك فى جدول الموظفين حيث يوجد حقل كزود الإدارة DEPTNO الذى يعتبر FOREIGN KEY حيث يوجد هذا الحقل كمفتاح رئيسى PRIMARY KEY فى جدول الإدارات.

3. المفاتيح المرشحة لتكون مفاتيح رئيسية CANDIDATE KEYS :

بفرض إننا لدينا الجدول R. فمن تعريف الجدول، إذا كان هناك أكثر من حقل يمكن أن يكون مفتاح رئيسى PRIMARY KEY بمعنى أن كل واحد منهم من الممكن أن يتم تحديد الصفوف الخاصة بالجدول من خلاله . وهذه الحقائق تدفعنا الى تكوين التعريف الخاص بالمفتاح المرشح الذى هو كالآتى:

بفرض أن K هى مجموعة خصائص بيانات خاصة بالعلاقة R . إذن K يكون مفتاح مرشح للعلاقة R وذلك إذا وفقط إذا كانت تتمتع بالصفات الآتية:

- التفرد: حيث ان كل قيم R لا تحتوى لا صفين لهم ذات القيم.
- عدم القدرة على تجزئتها: لا يمكن عمل مجموعة فرعية من K يمكنها ان تتمتع بخاصية التفرد.

4. المفتاح SUPER KEY

إن المجموعة الكلية من المفتاح المرشح هو الـSUPER KEY .

مثلا، فان مجموعة خصائص البيانات (S#, CITY) هى SUPER KEY للعلاقة S . ان SUPER KEY يمتاز بخاصية التفرد لكن ليس من الضرورى ان يمتاز بخاصية عدم القدرة على تجزئته (وخاصة ان المفتاح المرشح هو حالة خاصة من الـSUPER KEY).

ب- القيود التكاملية INTEGRITY CONSTRAINTS:

والقيود التكاملية هى القواعد التى تتخذ لضمان صحة البيانات والعمليات التى تتم عليها مثل الحذف DELETE والادخال INSERTION و التعديل UPDATE و ذلك فى قواعد البيانات DATABASES.

ومن هذه القيود ما يلى:

1- قيد التكامل فى وحدة البياناتENTITY INTEGRITY CONSTRAINT

2- قيد النوع TYPE CONSTRAINT

3- قيد القيمة الخاليةCONSTRAINTS ON NULL

4- قيود الخصائص ATTRIBUTE CONSTRAINTS

5- القيود المتعلقة بقاعدة البيانات DATABASE CONSTRAINT

6- القيود الراجعة للتكامل REFERENTIAL INTEGRITY CONSTRAINT

7-القيد الخاص بوحدة الجدولUNIQUE CONSTRAINT

8-قيد التحقق من شرط CHECK CONSTRAINT

1- قيد التكامل فى وحدة البياناتENTITY INTEGRITY CONSTRAINT:

ويحدد قيد التكامل فى وحدة البيانات ENTITY INTEGRITY CONSTRAINT أنه لا يوجد مفتاح رئيسى PRIMARY KEY يمكن أن تكون أحد قيمه بـ NULL .

وذلك لأن قيمة المفتاح الرئيسى PRIMARY KEY هى التى تحدد كل صفTUPLE على حدة فى الجدول TABLE حيث إنه عندما يأخذ المفتاح الرئيسى PRIMARY KEY قيمة بـNULL فلا يمكن هنا تحديد بعض الصفوف.فعلى سبيل المثال إذا كان هناك صفين أو أكثر يحتويان على القيمة NULL فى المفتاح الرئيسى PRIMARY KEY فلا يمكننا التمييز بينهما.

2. قيد النوع TYPE CONSTRAINT:

ويحدد قيد النوع القيم المتاحة LEGAL VALUES والتى يمكن تسجيلها فى الحقل ، فليس من المنطقى أن نحدد نوع البيانات DATA TYPES والخاصة بحقل معين على إنها أرقام NUMBERS ثم نقوم بإدخال نص عليها.

3. قيد القيمة الخالية CONSTRAINTS ON NULL:

ويحدد هذا القيد ما إذا كان حقل من حقول الجدول يسمح بوجود NULL VALUE فى أحد قيم صفوفه (أو أكثر) أم لا.

فعلى سبيل المثال فى جدول STUDENT إذا كان حقل الـ NAME لا يسمح بوجود قيمة خالية NULL VALUE إذن فهذا هو القيد على القيمة الخالية NULL VALUE .

4. قيود الخصائص ATTRIBUTE CONSTRAINTS:

تتعلق قيود الخصائص ATTRIBUTE CONSTRAINTS بأنه لابد من أن يتم تحديد نوع البيانات المرتبطة بالحقل FIELD بمعنى آخر أنه لابد من أن يكون الـ FIELD من نوع محدد.

5. القيود المتعلقة بقاعدة البيانات DATABASE CONSTRAINT:

قيد قاعدة البيانات هو القيد الخاص بالتداخل بين جدولين أو أكثر

6. القيود التكاملية المرجعية REFERENTIAL INTEGRITY CONSTRAINT

تحدد القيود التكاملية المرجعية REFERENTIAL INTEGRITY CONSTRAINT بين جدولين TWO TABLES وذلك لضمان اتساق صفوف هذين الجدولين ، بمعنى أن الصف فى أحد الجدولين يشير إلى صف آخر موجود فى الجدول الأخر.

7. القيد الخاص بوحدة الجدول UNIQUE CONSTRAINT

ويعنى صفة التفرد أو عدم التكرار UNIQUE ، فكل منها يستطيع أن يكون محدد لباقى الصفوف و الأعمدة ولكن لابد وأن يكون هناك مفتاح رئيسى واحد وباقى الحقول FIELDS تحدد قيد CONSTRAINT خاص بها يسمى الـ UNIQUE CONSTAINT ، وذلك

جدول الموظفين EMP

اسم الحقل	نوع الحقل
EM-NO	TEXT
EM-NAME	TEXT
EM-PHONE	TEXT
EM-ADDRESS	TEXT
EM-MOBILE	TEXT

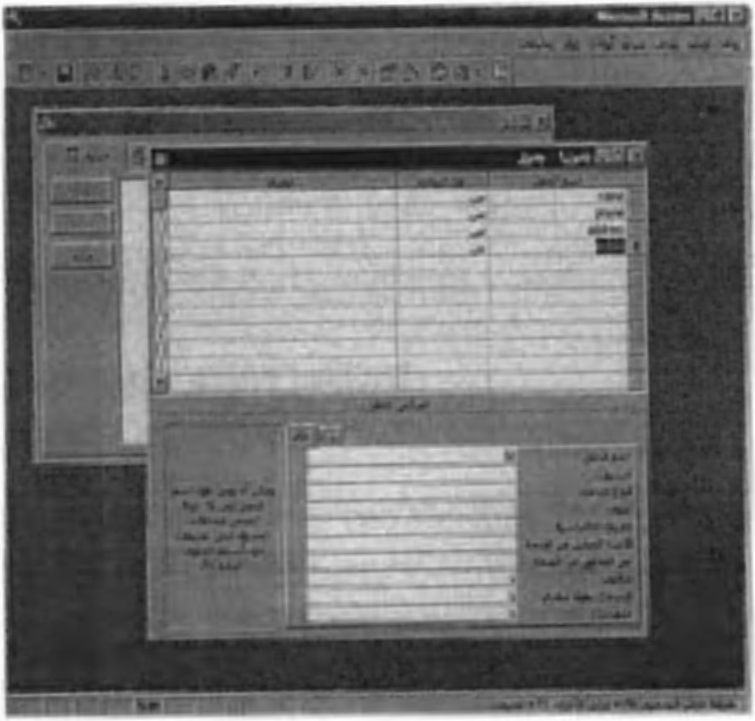
حاول إنشاء هذه الجداول للتدريب على خطوات إنشاء الجدول.

أنواع بيانات الحقول DATA TYPES

بعد إنشاء جدول نتعرض لأنواع بيانات الحقول ، فبعد كتابة اسم الحقل لابد من تحديد نوع الحقل حتى يمكن إدخال بيانات تناسب الحقل ، فمثلا عند تحديد نوع الحقل تاريخ (DATE) ، لابد على المستخدم من إدخال قيمة تاريخية سليمة في هذه الخانة ، ولمشاهدة أنواع البيانات أرجع إلى صفحة التصميم بنقر اسم الجدول ثم نقر الزر تصميم DESIGN ثم انقر نوع الحقل كما بالشكل التالي (4-12).

شكل(4-12)

والجدول التالي يعرض أنواع بيانات الحقول.



شكل (3-12)

إغلق شاشة تصميم الجدول تحصل على سؤال هل تريد الحفظ اختر نعم واكتب اسماً للجدول وليكن FRIENDS.

بهذا تم إنشاء جدول بالاسم FRIENDS بالتركيب السابق ، انقر اسمه ثم اختر فتح OPEN فيتم فتح الجدول لتسجيل بيانات.

تدريب على الجداول

من فضلك حاول تكرار الخطوات السابقة لإنشاء جدول مماثل وجرب ذلك بإنشاء الجداول التالية:

جدول الإضافة ITEMS

اسم الحقل	نوع الحقل
ITEM-CODE	TEXT
ITEM DESK	TEXT
ITEM – UNIT	TEXT
ITEM MODAL	TEXT
ITEM PRICE	TEXT

جرب استعمال المعالج فى إنشاء قواعد البيانات الجاهزة المختلفة والتى عرضناها مثل:

نظام المخزون	INVENTORY
نادى فيديو	VIDEO COLLECTION
مكتبة الصور	PICTURE LIBRARY
مركز تدريب	STUDENTS & CLASSES

وبهذا تكون قد قدمنا لك مقدمة عن قواعد البيانات والبرنامج MS ACCESS .

وفى الفقرات التالية نراجع معا بعض المفاهيم أساسية .

عناصر قاعدة البيانات

1ـ الجدول TABLE

الجدول هو أول عنصر يتم إنشاؤه فى قاعدة البيانات ، فهو العنصر الوحيد الذى يحتوى على البيانات ، وعليه تبنى باقى عناصر قاعدة البيانات ، وقاعدة البيانات عبارة عن مجموعة جداول ، والجدول عبارة عن مجموعة سجلات و السجل عبارة عن مجموعة حقول.

النماذج FORMS

النموذج (FORM) هو العنصر الثانى ، وهو عبارة عن شاشة إدخال للبيانات تصب فى الجدول ، ويتم إنشاء النموذج حتى لا يقوم المستخدم بتسجيل البيانات مباشرا فى الجدول ، والنموذج هو الشكل الذى يراه المستخدم ويتعامل معه فلابد أن يوفر الشكل الجيد والخطوات السهلة لتسجيل البيانات.

الاستعلام QUERY

الاستعلام (QUERY) عنصر من عناصر قاعدة البيانات الهامة وهو عبارة عن عنصر يعرض بيانات معينه تظهر من جدول أو أكثر بشروط معينة ، فمثلا يمكنك تكوين

استعلام بيانات العاملين الخاصين بقسم معين ، أو تكوين استعلام بالعاملين فى منطقه معينة.

التقرير REPORT

التقرير هو الطريقة الأساسية لعرض البيانات فى شكل تقرير مع إمكانية طباعته، و يمكن بناؤه على جدول أو استعلام.

الماكرو MACROS

الماكرو هو طريقه مختصره لتنفيذ عمليه معينة بشكل سريع مثل إنشاء ماكرو لفتح نموذج (FORM) ، أو ماكرو لفتح استعلام فهو يختصر كثيرا من الخطوات.

الوحدة النمطية MODULE

هى وسيلة لكتابة الأوامر بلغة VBASIC والتى تسمى VBA لتحقيق عمليات معينة لا تتحقق باستعمال الماكرو .

بعد عرض عناصر برنامج MS ACCESS بشكل مختصر تعالى معنا نتابع شرح هذه العناصر.

إنشاء الجداول TABLES

كما ذكرنا أن الجدول هو أهم عنصر فى قاعدة البيانات فهو الذي يحتفظ بالبيانات، وللقيام بتصميم جدول نفترض أن بياناته تخص الأصدقاء وهى عبارة عن الاسم ، والعنوان ، والتليفون ، والمحمول طبقا للخطوات التالية :

- قم بتشغيل برنامج MS ACCESS.
- انقر صفحة TABLE جدول.
- انقر زر جديد NEW.
- اختر عرض تصميمى DESIGN VIEW وقم بانشاء جدول.
- فى خانه اسم الحقل FIELD NAME اكتب NAME وفى السطر الثانى اكتب PHONE ثم ADDRESS ثم MOBILE كما فى الشكل (3-12).

القيد الـ CONSTRAINT موجود في معظم قواعد البيانات اختيار OPTION يتم اختياره عند إنشاء الجدول.

8. قيد التحقق CHECK CONSTRAINT:

هذا القيد هو عبارة عن شرط CONDITION لابد من حدوثه ، ويتم تحديده في مرحلة إنشاء قاعدة البيانات DATABASE ونجد أنه يعرف عن طريق اختيار أيضا في قاعدة البيانات أو من الممكن تحديده عن طريق اللغة الأساسية في التعامل مع قواعد البيانات SQL .

جـ. خرق العمليات في قاعدة البيانات OPERATION VIOLATION

العمليات الأساسية التي نقوم بها في قاعدة البيانات DATABASES هي الإدخال INSERTION والحذف DELETION و التعديل UPDATE، وقد يحدث عند إجراء بعض العمليات خرق VIOLATION للقيود الموضوعة CONSTRAINTS لذا وجب دراسة أنواع خرق القيود و طرق التعامل معها و تجنبها .

وسوف نتناول في هذا الجزء الآتي:

1- عملية الادخال INSERT OPERATION

2- عملية الحذف DELETE OPERATION

3- عملية التعديل UPDATE OPERATION

عند عملية الإدخال قد يحدث خرق للقيود الآتية والموضوعة على قاعدة البيانات DATABASE:

- فبالنسبة لقيد النطاق DOMAIN CONSTRAINT، يتم خرقه إذا كانت القيمة الموضوعة في الحقل FIELD لا توجد في النطاق المحدد DOMAIN.
- وبالنسبة للقيود الموجودة على المفاتيح KEY CONSTRAINTS نجد أنه يتم خرقها إذا كانت قيمة المفتاح في صف معين في جدول ما موجودة في نفس الجدول.
- وبالنسبة للقيد الموجود على وحدة البيانات ENTITY INTEGRITY CONSTRAINT إذا كان المفتاح الرئيسي PRIMARY KEY يحتوي على NULL.

- وبالنسبة لقيد التكامل المرجعي REFRENTIAL INTEGRITY CONSTRAINT نجد أنه يتم خرقه إذا كانت القيمة المراد ادخالها في المفتاح الخارجي FOREIGN KEY ليست موجودة في المفتاح الرئيسي PRIMARY KEY الذي يشير إليه المفتاح الخارجي FOREIGN KEY.

2. عملية الحذف DELETE OPERATION:

القيد الوحيد الذي يمكن خرقه في عملية الحذف DELETE هو قيد التكامل المرجعي REFRENTIAL INTEGRITY CONSTRAINT، فإذا تم حذف صف من جدول TABLE كان يشير إليه مفتاح خارجي من جدول آخر إذن فهذا خرق لقيد التكامل المرجعي REFERENTIAL INTEGRITY CONSTRAINT، وسيتضح ذلك من خلال الأمثلة الآتية:

1- عندما نحذف الصف الموجود في الجدول WORKS_ON والذي فيه:

ESSN=999887777 وفيه DNO=10

نجد هنا أن عملية الحذف ستقبل.

2- عندما نحذف الصف الموجود في الجدول EMPLOYEE والذي فيه SSN = 999887777

نجد هنا أن عملية الحذف لن تقبل و ذلك لأن هناك صفوف في الجدول WORKS_ON تشير إلي هذا الصف.

وهناك طريقة حل بالنسبة لحالات الخرق VIOLATION الناتجة عن عملية الحذف و هي CASCADING OPTION وهو موجود في اللغة الأساسية SQL و موجود بصورة مباشرة أو غير مباشرة في كل قواعد البيانات فعلى سبيل المثال في قواعد بيانات ACCESS نجد أن هناك اختيارا OPTION عند عمل العلاقة بين جدول وآخر وهو CASCADE DELETE RELATED RECORDS والذي يقوم بحذف كل المفاتيح الخارجية FOREIGN KEYS التابعة لمفتاح رئيسي PRIMARY KEY قد تم حذفه.

3. عملية التعديل UPDATE OPERATION :

تستخدم عملية التعديل في تغيير القيم لخاصية أو أكثر في صف أو صفوف، لذلك

فمن الضرورى أن يكون هناك شرط معين لصحة عملية التعديل حتى لا يحدث خرق للقيود CONSTRAINTS.

2- عدل الخاصية DNO بحيث تساوى7 فى الموظف الذى فيه خاصية ال SSN تساوى 999887777.

نجد أن عملية التعديل هنا لن تقبل وذلك لأنها خرقت قيد التكامل المرجعى REFERENTIAL INTEGRITY CONSTRAINT.

3- عدل الخاصية SSN فى الموظف الذى فيه الخاصية SSN تساوى 999887777 إلى 987654321.

وسنجد أيضا أنها لن تقبل لأنها خرقت المفتاح الرئيسى PRIMARY KEY و قيد التكامل المرجعى REFERENTIAL INTEGRITY CONSTRAINT.

وسنجد أيضا أنه كما فى عملية الحذف DELETE كان هناك طريقة لتجنب الخرق الذى يمكن أن تحدثه عملية الحذف نجد أن هناك طريقة مشابهة لتجنب الخرق الذى يمكن أن تحدثه عملية التعديل UPDATE

تصحيح تصميم الجداول NORMALIZATION

نتناول كيفية تصحيح تصميم الجداول أو بمعنى آخر سوف نتناول المراحل التى يمكننا من خلالها الوصول إلى نظام قاعدة بيانات صحيح قائم على تصميم جيد للجداول وذلك من خلال النقاط التالية:

1- المراجعة الاولى لتصميم قاعدة البيانات FIRST NORMAL FORM (FNF).

2- المراجعة الثانية لتصميم قاعدة البيانات SECOND NORMAL FORM (SNF).

3- المراجعة الثالثة لتصميم قاعدة البيانات THIRD NORMAL FORM (TNF).

المراجعة الأولى لتصميم قاعدة البيانات: FIRST NORMAL FORM (FNF)

وفيه يتم مراجعة تركيب الجدول بحيث لا يحتوى الجدول على أى حقول مركبة ولتوضيح ذلك تابع تصميم الجدول الموجود فى الشكل (6-12).

EmpNo	EName	Project	
		ProjNo	ProjHours

EmpNo	EName	ProjNo	ProjHours
1	Mohamed	10	33
1	Mohamed	20	77
1	Mohamed	30	88
2	omr	55	74
3	Nada	61	65
4	Esraa	34	69

شكل (6-12)

ولحل هذه المشكلة نقوم بتطبيق قاعدة النموذج المحسن الأول 1NF بتقسيم الجدول إلى جدولين يصبح كما فى الشكل (7-12).

EmpNo	EName

EmpNo	ProjNo	ProjHours

شكل(7-12)

وبالتالى لا تتكرر البيانات بل تصبح كما فى الشكل (8-12)

EmpNo	EName
1	Mohamed
2	omr
3	Nada
4	Esraa

EmpNo	ProjNo	ProjHours
1	10	33
1	20	77
1	30	99
2	55	74
3	61	65
4	34	69

شكل(8-12)

وهذا ما يسمى1NF اى النموذج المحسن الأول أما النموذج المحسن الثانى (SNF) يتم فيه مراجعة نتيجة 1NF فإذا كانت هناك حقول لا تعتمد على الحقل PRIMARY KEY اعتماد كامل أى لا يصلح لأن يميزها تفصل هذه الحقول وينشأ لها وحدة جديدة ENTITY و النموذج المحسن الثالث TNF يتم فيه مراجعة SNF فاذا كانت هناك حقول تعتمد على حقول أخرى يتم فصلها فى وحدات ENTITIES أخرى وسوف يتم تناول ذلك فى بالتفصيل فى فصل منفصل.

ما هى لغة الاستعلامات SQL

الحروف SQL اختصاراً للعبارة Structure Query Language أى لغة الاستعلامات المركبة وهى عبارة عن لغة (مجموعة من الأوامر والدوال) للتعامل مع قواعد البيانات حيث توفر كيفية انشاء وتعديل الجدول وتوفر الاستعلام بشتى أنواعه ، ولا يكاد يوجد برنامج قواعد بيانات ألا ويستعمل اللغة SQL مثل برنامج Access وبرنامج Informix و Oracle وكذلك لغات البرمجة مثل Delphi, VC++ visual basic ويفضل لأي مصمم برنامج قواعد بيانات أن يلم بقواعد لغة SQL لما توفره من إمكانيات كثيرة بأوامر قليلة .

أجزاء لغة الاستعلامات SQL

تنقسم لغة SQL إلى الأجزاء التالية :

1. لغة التعريفات DDL (Data Defination Language)

هو الجزء المسئول عن إنشاء عناصر ملف قواعد البيانات مثل إنشاء وتعديل وحذف الجداول والفهارس وغيرها .

2. لغة التحكم DCL (Data Control Language)

هى مجموعة من الجمل المفيدة عند تصميم برنامج متعدد للمستخدمين فهى، فتوفر التحكم فى درجة السماحية لكل مستخدم للبرنامج وكذلك الاستعلام عن البيانات.

3. لغة الوصول للبيانات DML (Data Manipulation Language)

وهى أهم الأجزاء حيث نوفر معظم العمليات المطلوبة من قواعد البيانات مثل الاستعلامات وإضافة وحذف البيانات للجداول وفيما يلى تفصيل كل جزء من هذه الأجزاء.

لغة التعريفات DDL

هى مجموعة الجمل المسئولة عن إنشاء وحذف عناصر قواعد البيانات مثل الجداول والفهارس وغيرها .

ويتكون هذا الجزء من الكلمات الثلاثة التالية :

Create : لإنشاء عنصر جديد مثل جدول بيانات او ملف فهرس Index مع تحديد مواصفات حقول الجدول

Alter : للتعديل فى عنصر موجود (جدول ، فهرس او حقل فى جدول) و يقصد هنا بالتعديل في بناء الجدول و ليس البيانات الموجودة داخله.

Drop : حذف عنصر موجود (جدول ، فهرس ، .....)

أنواع البيانات

قبل شرح كيفية استعمال الأوامر السابقة مثل إنشاء الجدول ، نقوم أولاً بعرض أنواع البيانات التى نحتاجها عند تصميم الجدول والمتاحة فى لغة SQL والجدول التالى يعرض هذه الأنواع ومعنى كل نوع.

اسم النوع	المعنى
Smallint	حقل رقم صحيح طوله 2 بايت
Integer	حقل رقم صحيح طوله 4 بايت
Real	حقل رقم حقيقي طوله 4 بايت
Bit	حقل من نوع بت واحد
Tinyint	حقل رقم صحيح طوله 1 بايت
Bigint	حقل رقم صحيح طوله 8 بايت
Binary (n)	حقل ثنائي (Binary) ثابت الطول بعدد n من البايت
Varbinary(n)	حقل ثنائي متغير الطول بحد أقصى n من البايت
Date	حقل من نوع تاريخ
Time	حقل من نوع وقت
Timestemp	حقل يحتوى على تاريخ ووقت
Varchar (n)	حقل حرفي بالعدد n من الحروف .
Number(p,d)	حقل رقمي حقيقي عدد خاناته p وبعد العلامة d من الأرقام

جدول بانواع البيانات المستعملة فى SQL

إنشاء الجداول Create Table

غالباً ما يتم إنشاء وتعديل وحذف الجداول باستخدام أحد برامج قواعد البيانات مثل برنامج Access ، ولكن تحتاج أحياناً بعض العمليات استخدام الأوامر لإنشاء الجداول ، ويتم إنشاء الجدول باستعمال الأمر Create حيث يأخذ الصورة العامه التالية :

```
Create Table Table_Name ( Column - list - .....data types )
```

وفى هذه الصيغة يتم استعمال الأمر Create ثم كلمة Table لإنشاء جدول بالاسم Table_name وتركيب هذا الجدول هو Column List أي حقول الجدول ، وأنواع الحقول هى Data types ويوضح ذلك المثال التالى :-

```
Create Table Employee ( Emid   Integer,Emname Varchar ( 60),
Salary          Numeric ( 8,2 ),
Dept            Varchar (12 )
```

فى السطور السابقة تم إنشاء جدول بالاسم Employee ويحتوى على أربعة حقول أسمائها DEP,NO , NAME , TEL

وأمام كل حقل نوع هذا الحقل ، فالحقل الأول مثلاً من النوع Integer اى رقم صحيح وهكذا والثاني حرفي ويمكن تحديد شروط لمواصفات كل حقل مثل عدم احتواء الحقل على NULL. بكتابة ذلك أمام الحقل ولتوضيح ذلك نعدل المثال السابق ليصبح كما يلى :-

مثال :

```
Create Table Employee ( NO     Integer, not null,
NAME            Varchar ( 60), not null,
TEL             Nmmeric ( 8,2 ),
Dept            VARCHAR  (12 0)      Defautt      "Eng")
```

فى السطور السابقة يتم وضع العبارة NOT NULL أمام الحقل الأول والثاني وذلك لعدم السماح بتركهم فارغين (NULL) .

وفى السطر الأخير تم وضع الكلمة Default لوضع قيمة افتراضية للحقل Dept وهى الكلمة .Eng وهناك كثير من الكلمات التي يمكن استخدامها لتحديد مواصفات الحقل فمثلاً يمكن استعمال الكلمة Unique لعدم تكرار قيمة حقل معين.

حذف الجداول باستعمال الأمر Drop

ويأخذ الصورة العامة التالية:

```
Drop Table Table-Name
```

وفيها المتغير Table-Name هو اسم الجدول المطلوب حذفه ويقوم هذا الأمر بحذف الجدول وما فيه من بيانات ، أما حذف البيانات فقط فيكون باستعمال الأمر Delete كما سيرد فيما بعد.

تعديل مواصفات جدول Alter Table

أحيانا نحتاج لتعديل مواصفات جدول - مثل إضافة حقل إلى الجدول أو حذف

حقل غير مرغوب فيه ، ويتم ذلك باستعمال الأمر Alter حيث يأخذ الصورة العامة التالية :

```
Alter Table   Tabl-name   Add Column
Column - name     data type
```

وفي هذه الصورة تم استعمال كلمة Add Column لإضافة حقل جديد وهذا هو التعديل المطلوب.

ولتوضيح ذلك نقوم بإضافة حقل جديد للجدول السابق كما يلي :-

```
Alter table Employee Add Column (ADDRESS,VARCHAR)
```

في هذا المثال يتم إضافة حقل جديد بالاسم Address وهذا هو التعديل.

```
Alter table Employee Drop Column Bdate
```

وفي هذا السطر يتم حذف الحقل باستخدام الكلمة Drop.

ملفات الفهارس Indexes

هي ملفات مصاحبة للجداول تزيد من سرعة الوصول (البحث) والاستعلام عن السجلات :

- وهي عبارة عن ترتيب(فرز) الجدول حسب حقل أو أكثر ، وغالباً ما يتم إنشائها عند إنشاء الجدول.
- و يمكن إنشائها باستعمال الأمر CREATE .

كما يلي:

```
Create ( Unique ) Index  Index - name
On Table -name ( Column - nam (s) { Asc, desc }
```

والمثال التالي يوضح طريقة أستعمال هذه الصيغة :

```
Create Unique Index  Empindx

ON  EMployed ( Emid )
```

وفي هذا المثال يتم إنشاء فهرس مرتب حسب الحقل Emid وذلك بالاسم Empindex

ويتضح ذلك في المثال التالي :

```
Create Index Empindx2 ON  Eployee  ( Emname , Dept )
```

وفى هذا المثال تم بناء ملف الفهرس على أساس الحقلين (Emname , Dept)
ويتم حذف ملف الفهرس بالأمر DROP.

```
Drop Index  Index - name
```

لغة التحكم DCL

كما أشرنا أن هذا الجزء يقوم بتحديد من المسموح له باستعمال هذه البيانات ودرجة السماحيه ، هل هى للاستعلام أم للإضافة أم للحذف أم للتعديل ، ولتوضيح ذلك نبدأ بالمثال التالى :

```
Grant Select ON Employee To Public
```

هذا المثال يسمح لجميع المستخدمين باستعمال الجدول ، حيث يبدأ بالكلمة Grant وهى سماحية ثم Select لتحديد السماحية للاستعلام وختم بـ To Public أى عام. أما أذا أردنا تحديد الإضافة والاستعلام نستعمل الكلمتين Insert , Select ثم تحديد المستخدم المسموح له بذلك بالعبارة TO Hamdy,Samy.
ويمكن تحديد حقول معينة لأشخاص معينة كما فى المثال التالى:

```
Grant UpDate ( Salary , Dept ) ON Employee
TO Nabil , Azab
```

فى هذا المثال يتم تحديد سماحية تحديث حقلين بالعبارة Up date (Salary , Dept) وذلك لشخصين هما Azab , Nabil ومن هذا يتضح أهمية هذا الجزء عند تصميم برنامج متعدد المستخدمين:

لغة الوصول للبيانات DML

يعتبر هذا الجزء من لغة SQL من أهم الأجزاء ويتكون من الأوامر التالية:

```
Select
Insert
Delete
Update
```

الأمر Select :

يعتبر هذا الأمر هو قلب لغة SQL فهو الذى يوفر اختيار بيانات مختلفة من جدول أو أكثر ، والصيغة العامة له بسيطة ولكن يمكن إضافة معاملات لها تزيد من قوتها وتأخذ

هذه الصيغة الصور التالية :

```
Select    Select-List    From    Table-name
```

وفيها المتغير Select-List عبارة عن الحقول المطلوبة ، ويمكن أن تأخذ صور بسيطة كاختيار جميع حقول جدول (*) أو حقول مختارة من أكثر من جدول، والمتغير Table-name هو اسم الجدول . وفيها يلى عدة صور لاستخدامها.

الصورة الأولى :

```
Select * From My table
```

فى هذه الصورة يتم اختيار جميع الحقول وذلك بالعلامة (*) من الجدول My table.

الصورة الثانية :

```
Select Emid , Emname From My table
```

فى هذه الصورة تم اختيار حقلين Emid , Emname من نفس الجدول My Table

الصورة الثالثة :

```
Select Emid , "....." Emname From My table
```

فى هذه الصورة تحديد صورة تحديد قيمة الحقل التالى للحقل Emid بالقيمة " ....." وبالتالى يتم اختيار الحقول التى قيمتها " " وهى مفيدة فى كثير من التطبيقات لعرض السجلات التى تحتوى على قيمة معينة.

العامل Where

هذا الأمر مفيد جداً فى تحديد حقول معينة (شرط معين) باستعمال مؤثرات المقارنة (=، < >، = <، = >، < ، >) لتحديد الشرط المطلوب وفيها يلى عدة صور لاستعمالها.

الصورة الأولى :

```
Select * from Employee where Emid=123
```

فى هذه الصورة يتم اختيار جميع حقول الجدول Employee التى يساوى حقل Emid فيها القيمة 123 ويتم عرض سجل واحد وهو الموظف رقم 123.

الصورة الثانية :

```
Select * From Employee Where Salary > 500
and Salary <900
```

وفى هذه الصورة كان الشرط أن تقع قيمة الحقل Salary.

بين القيمتين 500 و 900 وبالتالى يتم عرض الموظفين التى تقع مرتباتهم بين 500، 900

الصورة الثالثة :

```
Select * From Employee Where Dept = ( Eng) OR Dept = ( Mid )
```

الصورة الرابعة : يتم استعمال المؤثر المنطقى OR.

```
Select * From Employee Where Not Dept = ( tec )
```

فى هذه الصورة كان الشرط لاختيار السجلات هو أن لا يساوى الحقل Dept القيمة TEC وهو استعمال المؤثر المنطقى NOT.

ويمكن كتابة الشرط بالشكل `Where Dept < > " Tec "`

```
Select * From Employee where Dept != Tec
```

التعبير الرياضي (الحسابي)

يمكن استعمال التعبير الحسابي مع الأمر Select أو Where ويظهر ذلك كما يلى :

```
Select  Partno , Pcost + oth cost From Parts
```

فى هذا المثال يتم اختيار الحقل Partno ، ثم اختيار مجموعة الحقلين P Cost – Oth Cost من الجدول Parts ويمكن استعمال التعبير مع Where كما يلى :

```
Select Partno from Parts Where ( Oth Cost / Pcost ) > 0.20
```

وذلك معناه إمكانية استعمال عملية حسابية مكان حقل سواء فى الاختيار أو الشرط.

مجموعة دوال الحرفيات String Functions

هذه المجموعة من الدوال توفر عمليات الحرفيات مثل معرفة طول عبارة حرفية ( ) Length حذف المسافات من العبارة الحرفية وغيرها من الوظائف .

الدوال الرقمية

هى الدوال التى توفر العمليات الحسابية مثل الدوال Asin() ، Acos() ، Atan() ، Cos() ، Log() ، PI() ، Power() وكثيراً من الدوال .

دوال التاريخ والوقت Date&Time Functions

مجموعة من الدوال التى تتعامل مع التاريخ والوقت مثل الدالة Current Date التى تعيد التاريخ الحالى ، الدالة Current-Time() التى تعيد الوقت الحالى .

- الدالة Day of week تعيد رقم اليوم الحالى فى الاسبوع وهو الرقم من 1إلى 7.
- الدالة Day of Month() تعيد رقم اليوم الحالى فى الشهر .
- الدالة Hour() تعيد الساعة وكذلك الدالة Now() وغيرها من الدوال .

عرض بيانات من أكثر من جدول

Displaying Data from multiple tables

ويتم تناول ذلك من خلال النقاط التالية:

- استعمال جملة select لعرض بيانات من اكثر من جدول
- إنشاء علاقة خارجية outer join
- ربط جدول بنفسه joining table to it self

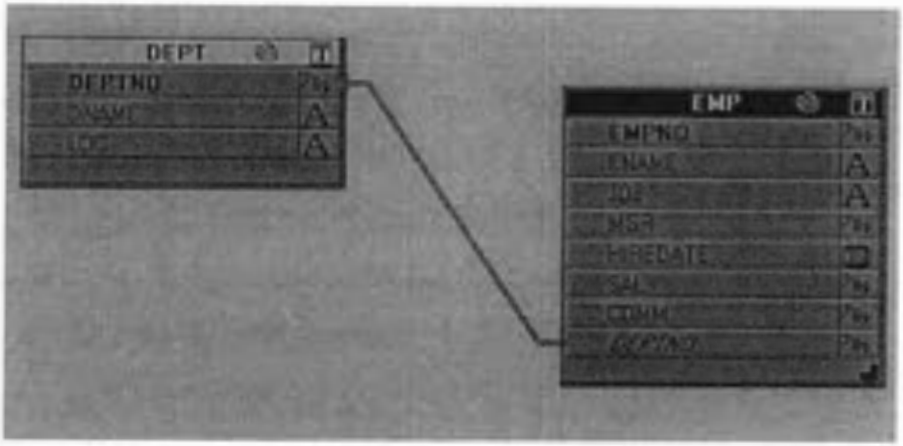
تعريفات

الحقل primary key

هو الحقل الذي يميز الجدول والذي لا يمكن تكرار قيمته مثل كود الموظف في جدول الموظفين ، كود الصنف في جدول الأصناف كود الإدارة فى جدول الإدارات.

الحقل Foreign key

هو حقل أصلاً primary key في جدول أخر يضاف إلي جدول أخر ليمثل بيانات الجدول الأصلي مثل حقل كود الإدارة في جدول الموظفين ومثل كود المنطقة فى جدول الموظفين ولتوضيح الحقلين انظر إلي الشكل(9-12).



الشكل(9-12)

اختيار بيانات من أكثر من جدول

ويتم ذلك عندما يحتوي جدول علي الحقل foreign key الذى هو primary key في جدول أخر كما أشرنا ونريد عرض البيانات الموجودة في الجدول الأصلي و التي تنتمي إلي هذا الحقل .

فمثلاً يحتوي جدول الموظفين علي بيانات الموظفين بما فيها كود الإدارة deptno بينما توجد بيانات الإدارات مثل اسم الإدارة وغيره في جدول الإدارات فنريد عرضها بشرط أن يساوي حقل الإدارة في جدول الموظفين حقل الإدارة في جدول الإدارات ويكتب الأمر كما بالشكل التالي:

```
Select EMP. deptno, Dept. Dname From EMP, Dept
Where EMP. DeptNO =Dept. DeptNO;
```

في هذا المثال تم اختيار الحقل deptno من الجدول EMP لذلك تم وضع . بين اسم الجدول و اسم الحقل.

وتم اختيار الحقل DName من الجدول Dept لذلك تم وضع . بينه وبين جدوله Dept، ثم كتابة from EMP, Dept لأنه بالفعل يختار بيانات من الجدولين وشرط الاختيار where هو أن كود الإدارة في جدول الموظفين يساوي كود الإدارة في جدول الإدارات

وبالتالي يتم عرض الاسم الأول والاسم الثاني من جدول EMP ويتم عرض اسم

الإدارة من جدول الإدارات الذي كود إدارته موجود في جدول الموظفين .

اكتب الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (10-12)

```
DEPTNO DNAME
------ --------------
    20 RESEARCH
    30 SALES
    30 SALES
    20 RESEARCH
    30 SALES
    30 SALES
    10 ACCOUNTING
    20 RESEARCH
    10 ACCOUNTING
    30 SALES
    20 RESEARCH

DEPTNO DNAME
------ --------------
    30 SALES
    20 RESEARCH
    10 ACCOUNTING
```

الشكل (10-12)

يمكن كتابة الأمر السابق بشكل أخر وهو كما يلي :

```
Select E.deptno, D. Dname
From EMP. E, Dept .D
Where E Dept NO=D. Dept NO;
```

في هذا المثال تم وضع الحرف E أمام اسم الجدول EMP.

وضع الحرف D أمام اسم الجدول Dept.

وبالتالي جميع التعاملات مع الجدولين تتم بالتعامل مع الحرفين D,E وهذا أسهل من كتابة أسماء الجداول كل مرة اكتب ونفذ الأمر وشاهد النتيجة .

ملحوظة هامة: عند كتابة جملة SELECT للاختيار من جدولين بدون الأمر Where تحصل على نتائج غريبة حيث يتم تكرار جميع سجلات الجدول الثاني مع كل سجل من الجدول الأول .

العلاقة الخارجية OUTER Join

تقوم هذه العلاقة بعرض البيانات التي ليس لها مقابل في الجدول الثاني عكس ما تم

فى الفقرة السابقة حيث كنا نعرض بيانات الإدارات من جدول الإدارات التى يوجد لها مقابل فى جدول الموظفين ولكن العلاقة الخارجية تمثل العكس فمثلاً لمعرفة : الإدارات التى لم يسكن عليها موظفين أى التى لا يوجد لها مقابل فى جدول الموظفين نستخدم هذه العلاقة كما فى السطور التالية:

```
Select EMP. deptno, Dept. Dname
From  Dept RIGHT OUTER JOIN  EMP;
```

أكتب ونفذ الأمر لتحصل على النتيجة كما فى الشكل (11-12).

```
DEPTNO DNAME
------- --------------
     10 ACCOUNTING
     10 ACCOUNTING
     10 ACCOUNTING
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     30 SALES
     30 SALES
     30 SALES

DEPTNO DNAME
------- --------------
     30 SALES
     30 SALES
     30 SALES
        OPERATIONS
```

الشكل(11-12)

نجد العبارة Right Outer لتعبر عن الطرف الذى تكون بعض بياناته لا تقابلها قيم محددة بل تقابلها القيمة NULL وهذا الطرف هو الجانب الأيمن.

أما إذا أردنا الجانب الأيسر فنجد العبارة Left Outer لتعبر عنه كما يتضح من المثال التالى:

```
Select EMP. deptno, Dept. Dname
From  emp LEFT OUTER JOIN  dept;
```

وتكون النتيجة كما بالشكل (12-12).

```
DEPTNO DNAME
------- --------------
     10 ACCOUNTING
     10 ACCOUNTING
     10 ACCOUNTING
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     20 RESEARCH
     30 SALES
     30 SALES
     30 SALES

DEPTNO DNAME
------- --------------
     30 SALES
     30 SALES
     30 SALES
        OPERATIONS
```

الشكل(12-12)

ربط الجدول مع نفسه Jonning a table to itself

أحياناً نحتاج لعرض بيانات من جدول واحد ولكن بشرط يربط بين حقلين من حقوله لتحقيق شرط معين ، واشهر مثال هو جدول الموظفين الذي يحتوي على كود الموظف وكود المدير وفي نفس الوقت هذا الموظف له مدير هو موظف له كود كموظف وفي نفس الوقت يسجل رقمه في حقل المدير ، ويظهر ذلك من الأمر التالي:

```
SELECT a.empid, a.location, a.date, b.location, b.date
FROM test_results a, test_results b
WHERE a.empid = b.empid
AND a.location <> b.location
AND a.date > sysdate-365
AND b.date > sysdate-365;
A.EMPID A.LOCATION A.DATE B.LOCATION B.DATE
------- ---------- --------- ---------- ---------
94839 St. John 04-NOV-97 Wendt 03-JAN-98
04030 Stridberg 27-JUN-97 Wendt 03-AUG-97
```

دوال المجموعات Group functions

دوال المجموعات هي دوال تنفذ على جميع سجلات الجدول أو مجموعة السجلات حسب شرط معين مثل دالة الجمع ()sum أو دالة المتوسط ()AVG وغيره.

وفي هذه الفقرة نتناول ما يلي:

- دوال المجموعات Group functions
- استعمال دوال المجموعات Using Group function
- الأمر Group BY
- تحديد البيانات باستعمال Having

الدوال:

- اسم الدالة: **AVG(x)**

الاستخدام: تستخدم لحساب متوسط قيمة حقل معين مثل متوسط مرتبات الموظفين.

- اسم الدالة: **COUNT(x)**

الاستخدام: تستخدم لحساب عدد سجلات جدول أو عدد سجلات حقل معين.

- اسم الدالة: **MAX(x)**

الاستخدام: تستخدم لمعرفة أقصى (اكبر) قيمة من مجموعة قيم الحقل مثل معرفة أعلى قيمة مرتب.

- اسم الدالة: **MIN(x)**

الاستخدام: تستخدم لمعرفة أقل قيمة من مجموعة قيم لحقل مثل معرفة أقل قيمة مرتب.

- اسم الدالة: **SUM(x)**

الاستخدام: تستخدم لمعرفة مجموع قيم حقل مثل مجموع مرتبات الموظفين او مجموع مرتبات موظفي إدارة.

- اسم الدالة: **VARIANCE(x)**

استخدام دوال المجموعات

• الدوال: SUM() ,AVG() ,MIN() ,MAX()

```
SQL> SELECT     AVG(sal), MAX(sal),
  2       MIN(sal), SUM(sal)
  3 FROM     emp
  4 WHERE   job LIKE 'SALES%';
```

في هذا المثال يتم استعمال مجموعة دوال على حقل Sal من جدول EMP.

اكتب هذا الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (13-12).

```
AVG(SAL)  MAX(SAL)  MIN(SAL)  SUM(SAL)
---------- ---------- ---------- ----------
      1400       1600       1250       5600
```

الشكل (13-12)

• الدالة: count()

```
Select Count (*),
from EMP;
```

يمكن استعمال الدالة Count() بثلاثة أشكال الأول بالعلامة *.

ويتم عد جميع السجلات وهذا أبطأ الاختيارات ،الصورة الثانية Count(1) .

وهو عدد الحقل الأول في الجدول ، والثالث هو Count() مع اسم الحقل empno.

اكتب الاوامر ونفذها تحصل على الشكل (14-12).

```
SQL>  Select Count (*)
  2  From emp;

  COUNT(*)
----------
        14

SQL>
```

الشكل(14-12)

• الأمر Group BY

يستعمل هذا الأمر لتقسيم سجلات جدول إلى مجموعات فمثلاً لعرض إجمالي مرتبات كل إدارة كلاً على حدة ويتضح ذلك من المثال التالي:

```
SELECT   deptno, AVG(sal)
  2 FROM     emp
  3 GROUP BY deptno;
```

في هذا المثال يتم استدعاء مجموعة من دوال المجموعات على حقل المرتب Salary وكذلك عرض حقل كود الإدارة DeptNO من جدول الموظفين بعد تقسيم السجلات إلى مجموعات حسب كود الإدارة، وبالتالي يتم عرض كود الإدارة وأمامه مجموع ومتوسط وعدد سجلات هذه الإدارة . اكتب الأمر ونفذه تحصل على الشكل (15-12) .

```
DEPTNO  AVG(SAL)
--------- ---------
10 2916.6667
20        2175
30 1566.6667
```

الشكل(15-12)

ملحوظة هامة:

في حالة استعمال GROUP BY لا يمكن اختيار حقل فردي بدون دالة غير الحقل الذي يتم عمل GROUP BY له.

فمثلاً في المثال السابق تم اختيار الحقل EMPNO وهو الحقل الذي تم عمل GROUP BY له، ولكن لا يمكن عرض الحقل EName أو غيره ونحن نختار تجميع سجلات الإدارات حسب كود الإدارة . لتوضيح هذه الملحوظة انظر المثال التالي:

```
SQL> SELECT   deptno, job, sum(sal)
  2 FROM     emp
  3 GROUP BY deptno;
```

اكتب هذه الأوامر ونفذه تحصل على رسالة خطأ كما في الشكل (16-12)

```
SQL> SELECT    deptno, job, sum(sal)
  2   FROM      emp
  3   GROUP BY deptno;
SELECT    deptno, job, sum(sal)
                  *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

الشكل(16-12)

تم تنفيذ هذا المثال من خلال الـSQL Plus الموجودة في Oracle

ولتصحيح ذلك اكتب الصيغة التالية.

```
SQL> SELECT   deptno, job, sum(sal)
  2 FROM     emp
  3 GROUP BY deptno;
```

وعند تنفيذها نحصل على الشكل (17-12).

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900

الشكل(17-12)

والخطأ في هذا المثال هو محاولة عرض الوظيفة job في حين يتم التجميع باستعمال الحقل EMPNO.

استعمال ORDER BY مع GROUP BY

لنفرض أننا نريد عرض متوسطات مرتبات الإدارات كما سبق ولكن نريد عرض هذه المتوسطات مرتبة تنازلياً يتم ذلك باستعمال الأمر ORDER BY مع الأمر GROUP BY ويظهر ذلك من المثال التالي:

```
SQL> SELECT   job, SUM(sal) PAYROLL
  2 FROM     emp
  3 WHERE    job NOT LIKE 'SALES%'
  4 GROUP BY job
  6 ORDER BY SUM(sal);
```

في هذا المثال تم طلب عرض الوظيفة ومتوسط المرتب من جدول الموظفين وتجميع ذلك بشرط الوظيفة ثم طلب ترتيب ذلك حسب متوسط المرتب تنازلياً وذلك باستعمال order by اكتب الأمر ونفذه تحصل على الشكل (18-12).

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

الشكل (18-12)

ـ استعمال الأمر Having

يستعمل هذا الأمر كبديل للأمر where الذي يستعمل في عدم وجود دوال مجموعات (Group function) ويؤدي نفس الغرض وباستعماله تستطيع وضع شروط لعرض البيانات مع Group BY مثل عرض بيانات الإدارات ذات المرتب أكبر من قيمة معينة ولتوضيح ذلك تابع المثال التالي:

```
SQL> SELECT   job, SUM(sal) PAYROLL
  2 FROM     emp
  3 WHERE    job NOT LIKE 'SALES%'
  4 GROUP BY job
  5 HAVING   SUM(sal)>5000
  6 ORDER BY  SUM(sal);
```

في هذا المثال تم إضافة الأمر HAVING بمعني WHERE.

إن متوسط المرتب (Sal) AVG اكبر من 5000 بحيث يتم عرض الإدارات التي متوسط مرتبتها أكبر من 5000.

اكتب الأمر ونفذه تحصل علي الشكل (19-12).

```
JOB       PAYROLL
--------- ---------
ANALYST      6000
MANAGER      8275
```

الشكل (19-12)

استعمال استعلامات فرعية SUB QUERIES

يمكن استعمال استعلام داخل استعلام أخر حسب الحاجة فيمكن أن يحتوي الشرط علي استعلام ، لهذا عدة أشكال منها:

الاستعلامات المتداخلة Nested subqueries

نفرض أنك تريد عرض بيانات موظف ولكن لا تعلم عنه معلومات غير أنه في نفس إدارة الموظف الذي كود الموظف له 7900.

تحقق ذلك بالشكل:

```
Select EMPNO, DeptNO, SAL FROM EMp
  Where DeptNO= (select deptNO from EMP WHERE EMPNO =7900);
```

في هذا المثال يتم عرض بيانات الموظف (كود، اسم، مرتب الموظف) الذي كود الإدارة له يساوي نتيجة الاستعلام الصغير .

الاستعلام الصغير يعيد كود الإدارة للموظف المعلوم كوده وهو 7900 وبالتالي الكود العائد من الاستعلام الصغير يوضع كطرف للشرط في الاستعلام الكبير .

اكتب هذا الأمر ونفذه تحصل على نتيجة التنفيذ كما بالشكل (20-12)

EMPNO	DEPTNO	SAL
7499	30	1600
7521	30	1250
7654	30	1250
7698	30	2850
7844	30	1500
7900	30	950

الشكل(20-12)

استعلامات متداخلة :

يمكن وضع استعلام داخل استعلام ، نفرض أنك لا تعلم كود الموظف في المثال السابق كل ما تعلمه أن الموظف الذي تريد عرض بياناته يوجد في نفس إدارة الموظف الذي له فاتورة INVOICE-NUMBER هي 5644.

اكتب الأمر التالي :

```
Select E.EMPNO, E.Sal from EMP
Where E.DEPTNO = (select DeptNo from EMP
Where EMPNO = (Select EMPNO from invoice
Where invoice_number = 5644));
```

في هذا المثال يوجد ثلاث استعلامات

الأول: يستعلم عن EMPNO، Sal من جدول الموظفين EMP ولكن الشرط أن يساوي كود الإدارة Dept NO نتيجة الاستعلام .

الثاني: يستعلم عن كود الإدارة من جدول الموظفين للموظف الذي كوده هو نتيجة الاستعلام الثالث .

الثالث: يستعلم عن كود الموظف من جدول الفواتير الذي رقم الفاتورة فيه هو 5644

وبالتالي الاستعلام الثالث يأخذ رقم الفاتورة ويعطي كود الموظف للاستعلام الثاني الذي يأخذ هذا الكود ويعطي كود الإدارة والاستعلام الأول يأخذ الكود هذا الكود ليعطي بيانات الموظف المطلوب الاستعلام عنه.

استعمال IN مع الاستعلامات الفرعية.

يمكن استعمال IN بدلاً من = إذا كان الحقل المطلوب مقارنته بمجموعة قيم وليس قيمة واحدة 0 ويظهر ذلك كما بالشكل :

```
>Select * from EMP Where EMPNO IN
(Select EMPNO from EMP where DeptNO = 20)
```

أكتب الأمر ونفه تحصل على النتيجة كما بالشكل

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7902	FORD	ANALYST	7566	03-DEC-81	3000		20

في هذا المثال يتم عرض بيانات الموظفين الذين تقع أكوادهم في القيمة العائدة من الاستعلام الصغير الذي يعيد أكواد الموظفين الذين كود الإدارة لهم 20.

استعمال ORDER BY مع الاستعلامات الفرعية

تستعمل ORDER BY لترتيب البيانات وتكتب في نهاية الاستعلام الكبير ويظهر ذلك كما بالشكل :

```
>select empno,ename from emp order by ename;
```

لاحظ أن الأمر ORDER BY عائد على الاستعلام الكبير الخارجي نتيجة ترتيب البيانات حسب كود الإدارة .

نفذ الامر لحصل على الشكل.

EMPNO	ENAME
7876	ADAMS
7499	ALLEN
7698	BLAKE
7782	CLARK
7902	FORD
7900	JAMES
7566	JONES
7839	KING
7654	MARTIN
7934	MILLER
7788	SCOTT

EMPNO	ENAME
7369	SMITH
7844	TURNER
7521	WARD

الأمر Insert

يستخدم هذا الأمر لإضافة بيانات إلى سجلات الجدول ويأخذ الصورة العامة التالية :

```
Insert Into Table-name { ( Columns - List ) }
        Values ( Value - List )
```

وفى هذه الصورة يتم إضافة القيم Value - List فى الحقول Columns - List فى الجدول Table-name ويتم تحقيق ذلك بأكثر من من صورة نوضحها فيما يلى :

الصورة الأولى :

```
Insert Into Employee Values ( 123, Ahmed , 500 eng )
```

فى هذه الصورة يتم إضافة هذه القيم إلى حقول الجدول Employee بترتيب حقول الجدول وفى هذه الصورة لم يتم تحديد الحقول وبالتالي يتم إضافة قيم للحقول كلها وبترتيب الحقول .

الصورة الثانية :

```
Insert Into Employee ( Emid , EmName , Dept )
        Values (30 , sAMY , tec )
```

وفى هذه الصورة تم إضافة بيانات إلى ثلاثة حقول وتم تحديد الحقول ثم القيم المقابلة لها.

الصورة الثالثة :

```
Insert Into Fixed - hist Select { Fixed - Data } . *
    From { Fixed - Data } *
```

و فيها يتم اختيار كل حقول الجدول Fixed - Data وإضافتها إلى الجدول Fixed - hist وبالتالى يتم إضافة جميع سجلات الجدول Fixed - Data إلى نهاية الجدول Fixed – hist.

الأمر Delete

يقوم هذا الأمر بحذف البيانات من الجدول أما كلها أو حسب الشرط الصورة العامة لها كما يلى

```
Delete From Table - name
```

وفيها يتم مسح جميع بيانات الجدول Table - name ولتوضيح ذلك تابع السطر التالى :

```
Delete From Employee
```

يؤدى هذا السطر إلى حذف جميع بيانات الجدول Employee

ويمكن مسح السجلات التى توافق شرط معين كما فى السطر التالى :

```
Delete From Employee Where Emid = 20
```

وفيها يتم مسح السجل الذى يأخذ الحقل Emid فيه القيمة 20 .

الأمر UpDate

يسمح هذا الأمر بتعديل قيمة حقل معين ويتضح ذلك من السطر التالى

```
UpDate Employee Set Salary = Salary + 200
```

وفيه يتم إضافة القيمة 200 إلى القيمة الموجودة فى الحقل Salary

ويمكن تعديل الحقول التى توافق شرط معين كما فى السطر التالى :

```
Up Date Employee Set Salary = 900 Where Salary < 900
```

وفى هذا السطر يتم وضع القيمة 900 فى الحقل Salary لجميع السجلات التى بها الحقل Salary أقل من 900 .