تطبيقات رياضية (مقدمة في قواعد المعطيات)

منشورات جامعة دمشق
كلية العلوم

تطبيقات رياضية
(مقدمة في قواعد المعطيات)

تأليف

د. محمد فراس الحلبي
مدرس في قسم الرياضيات

أيمن صالح محمّد
قائم بالأعمال في قسم الرياضيات

جامعة دمشق

1437 – 1438 هـ
2016 – 2017 م

2016-7-24

الفهرس

الموضوع	الصفحة
المقدمة.	13
الفصل الأول: مفاهيم أساسية في قواعد البيانات	15
1. مقدمة في قواعد البيانات	15
1.1. تمهيد	15
1.2. ما هي قاعدة البيانات؟	16
1.3. تعريف قاعدة البيانات	16
2. المعلومات و البيانات	17
3. أهمية قاعدة البيانات	19
3.1. مميزات قواعد البيانات	20
3.2. من يستثمر قاعدة البيانات؟	20
3.3. بيئة قاعدة البيانات	21
4. نظام إدارة قواعد البيانات	26
4.1. تعريف نظام إدارة قواعد البيانات	26
4.2. مستخدمو قاعدة البيانات	28
5. لغات نظام إدارة قواعد البيانات	30
6. أنواع نظم إدارة قواعد البيانات	34
6.1. نظم إدارة قواعد البيانات الهرمية	34
6.2. نظم إدارة قواعد البيانات الشبكية	36
6.3. نظم إدارة قواعد البيانات العلائقية	39
7. اكتشاف المعرفة في قواعد البيانات	41
7.1. مفاهيم التنقيب في البيانات	42

الفصل الثاني: تصميم قاعدة البيانات 55
1. أهمية تصميم قاعدة البيانات 55
2. دورة حياة نظام قاعدة البيانات 55
2.1. الدراسة المبدئية للنظام القائم 55
2.2. تصميم قاعدة البيانات 55
2.3. هيكل (مخطط) قاعدة البيانات 56
3. نماذج قاعدة البيانات: النموذج العلائقي 57
3.1. خواص الجداول في قواعد البيانات العلائقية 58
3.2. أهم مفاهيم قواعد البيانات العلائقية 61
3.2.1. المفتاح الأساسي للجدول 61
3.2.2. المفتاح الغريب (الأجنبي) 61
3.2.3. القيمة Null 62
3.2.4. الفهارس 62
3.3. أنواع العلاقات 63
3.3.1. علاقة واحد لواحد 63
3.3.2. علاقة واحد لمتعدد 64
3.3.3. علاقة متعدد لمتعدد 64
4. المفاهيم الأساسية في نموذج الكينونات العلائقية 65
4.1. الكينونة 65
4.2. نوع الكينونة 66
4.3. الواصفات 67
4.3.1. مفهوم الواصفة 67
4.3.2. القيود و مجالات الواصفات 71
4.4. أنواع العلاقات 72

4.5. درجة العلاقة 76
4.5.1. درجة العلاقة الأحادية 76
4.5.2. درجة العلاقة الثنائية 77
4.5.3. درجة العلاقة الثلاثية 77
4.6. قيد المشاركة 78
4.6.1. قيد المشاركة الكلي 78
4.6.2. قيد المشاركة الجزئي 79
5. نموذج الكينونات المتطور 87
5.1. التخصيص و التعميم 89
5.2. تحديد القيود على علاقة Isa من النوع أصلي/فرعي 91
6. التحويل من نموذج الكينونات والعلاقات الى النموذج العلائقي 94
الفصل الثالث: الجبر العلائقي 105
1. مفهوم العلاقة 105
2. عمليات الجبر العلائقي 105
2.1.الإختيار 105
2.2.الإسقاط 106
2.3.الإتمام 107
2.4.الاجتماع 107
2.5.التقاطع 108
2.6.الفرق 108
2.7.القسمة 109
2.8.الجداء 110
2.9.الربط 110
2.9.1. الربط المشروط 111

2.9.2. الربط الطبيعي 112
2.9.3. الربط الخارجي 113
2.9.4. الربط الجزئي 114
الفصل الرابع: تنظيم قاعدة البيانات 117
1. مفهوم التنظيم 117
1.1. إشكاليات التصميم السيئ لقاعدة البيانات 117
1.2. التبعية الوظيفية 121
1.2.1. التبعية الوظيفية و المفاتيح 123
1.2.2. الاعتمادات الوظيفية التامة و الاعتمادات الوظيفية المباشرة 123
1.2.3. بديهيات الارتباطات الوظيفية 123
1.3. مستويات التنظيم 124
1.3.1. الشكل النظامي الأول 125
1.3.2. الشكل النظامي الثاني 126
1.3.3. الشكل النظامي الثالث 127
1.3.4. الشكل النظامي لـ Boyce Cood 128
1.3.5. الشكلين النظاميين الرابع و الخامس 130
1.3.5.1. الارتباطات متعددة القيم 130
1.3.5.2. قواعد الارتباطات متعددة القيم 132
1.3.5.3. الشكل النظامي الرابع 133
1.3.5.4. الشكل النظامي الخامس 134
1.3.5.4.1. الاعتماد الارتباطي 134
الفصل الخامس: لغة الاستعلامات البنيوية SQL 141
1. مقدمة 141
2. عبارات SQL 142

2.1.لغة تعريف البيانات 142
2.2.لغة معالجة البيانات 142
2.3.لغة التحكم و الصلاحيات 142
3. جملة Select 143
3.1.اختيار كافة الأعمدة 143
3.2.اختيار أعمدة محددة 144
3.3.التعابير الحسابية 146
3.4.تعريف A Column Alias 149
3.5.جملة الشرط 151
3.6.معاملات المقارنة الأساسية 153
3.7.معاملات المقارنة الأخرى 154
3.7.1. المعامل Between 155
3.7.2. المعامل In 155
3.7.3. المعامل Like 157
3.7.4. المعامل Is Null 158
3.8.استخدام المعاملات المنطقية 160
3.8.1. المعامل and 160
3.8.2. المعامل or 161
3.8.3. المعامل not 161
3.9.قواعد أسبقية المعاملات 162
3.10.استخدام عبارة Distinct لمنع تكرار السجلات 164
3.11.ترتيب النتائج باستخدام order by 164
3.12.القيمة Null و تأثيراتها 167
4. أنواع العلاقات 169

4.1.الربط بين جدولين أو أكثر 169
4.2.استرجاع بيانات من جدولين أو أكثر 169
4.3.أنواع الروابط 170
4.3.1. الرابطة المتكافئة 170
4.3.2. الرابطة غير المتكافئة 170
4.3.3. الرابطة الخارجية 170
4.3.4. الروابط الذاتية 171
4.3.5. الرابطة الديكارتية 171
4.4.استخدام أكثر من شرط في جملة Where 177
4.5.استخدام Table Alias في جملة Where 177
4.6.استرجاع البيانات من عدة جداول 178
5. استخدام الدوال 178
5.1.استخدام الدوال لأكثر من سطر 178
5.1.1. استخدام Count 179
5.1.2. استخدام Count(*) 180
5.1.2.1. استخدام Distinct مع Count 180
5.1.3. استخدام الدالة Max 180
5.1.4. استخدام الدالة Min 181
5.1.5. استخدام الدالة Avg 181
5.1.6. استخدام الدالة Sum 182
5.1.7. استخدام الدالة Nvl مع الدوال التجميعية 182
5.2.استخدام الدوال لسطر واحد 183
5.2.1. أنواع الـ Single Row Functions 183
5.2.1.1. الدوال النصية 183

5.2.1.2. الدوال الرقمية 185
5.2.1.3. التاريخ و دوال التاريخ 186
5.3.الدوال التحويلية 187
5.4.الدوال العامة 190
5.4.1. الدالة Nvl 190
6. استخدام Group by 191
7. استخدام Having 194
8. استخدام وعمل استعلام فرعي 196
8.1.كيف تعمل الاستعلامات الفرعية 196
8.2.أنواع الاستعلامات الفرعية 197
8.2.1. الاستعلامات الفرعية التي تعيد سجل واحد 198
8.2.1.1. استخدام الدوال التجميعية في الاستعلام الفرعي 199
8.2.1.2. استخدام Group by و Having في الاستعلام الفرعي 200
8.2.2. الاستعلامات الفرعية التي تعيد أكثر من سجل واحد 201
8.2.3. الاستعلامات الفرعية التي تعيد أكثر من عمود واحد 202
9. عبارة Insert: كيفة إضافة سجلات جديدة 203
9.1.إضافة سجل واحد إلى الجدول 203
9.2.إضافة عدة سجلات إلى الجدول 205
9.3.إضافة سجلات عن طريق المتحولات البديلة 206
10.حذف بيانات موجودة في قاعدة البيانات 208
10.1.حذف سجلات محددة 208
10.2.حذف السجلات كافة 208
10.3.استخدام الاستعلام الفرعي في جملة Delete 209
11.تعديل / تحديث البيانات الموجودة في قاعدة البيانات 210

11.1.تحديث السجلات كافة 211
11.2.تحديث سجلات محددة 211
11.3.استخدام الاستعلام الفرعي في جملة Update 212
12.استخدام تعليمات التثبيت Commit والتراجع Rollback 212
13.إنشاء وتعديل وحذف الجداول 215
13.1.إنشاء الجداول 215
13.1.1. إنشاء الجداول باستخدام الاستعلام الفرعي 218
13.2.تعديل الجدول 219
13.3.حذف جدول 220
14.المحافظة على تكامل قاعدة البيانات 220
14.1.إضافة محددات إلى الجدول من نوع Primary/Foreign key 220
14.2.الشرط Check 221
14.3.خيار القيمة الافتراضية 222
14.4.الشرط Not Null 222
14.5.الشرط Unique 223
14.6.الشرط Primary key 224
14.7.الشرط Foreign key 225
14.8.تعديل تعريف شرط ما 226
الفصل السادس: كيفية تنصيب برنامج أوراكل اسكسبرس 231
المصطلحات العلمية 237
المراجع العلمية 243

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

المقدمة :

يعّد علم الرياضيات ركيزة أساسية لجميع العلوم الأخرى، نظرية كانت أم تطبيقية، فمثلاً يعّد علم الرياضيات روح العمل الهندسي في علم الهندسة لدوره في وضع النماذج والرسومات الهندسية ومحاكاة الواقع. ففي علم الفلك تبرز أهمية الرياضيات في معرفة حركة الشمس وانقلاب الليل والنهار وحركات القمر وحسابها والخسوف والكسوف وغيرها. وتوجد العديد من العلوم التي تعتمد على الرياضيات أيضاً، منها: الطب، والفيزياء، والكيمياء، وعلم الأحياء، والمناخ، والاقتصاد وكثير من العلوم الأخرى، و بذلك يعّد علم الرياضيات أساساً ترتكز عليه العلوم المختلفة. غير أن من أهم تطبيقات الرياضيات هي علم المعلوماتية، وعليه تمّ وضع كتاب تطبيقات رياضية تحت عنوان مقدمة في قواعد البيانات لنجد الصلة بين الرياضيات كعلم مجرد وبين ما يستفاد منه في علم الحاسوب في تخصص هندسة نظم المعلومات التي تبدأ بدراسة قواعد البيانات.

تعّد قواعد البيانات ذات أهمية كبيرة لتقدم أي مجتمع يخطط لبناء مستقبله على أطر علمية وتقنية سليمة، ولاسيما ونحن نعيش في عصر تتحكم فيه متغيرات كثيرة تبنى على بيانات ذات أهمية سواءً كانت اقتصادية أو اجتماعية أو غيرها. إن وضع الخطط التنموية سواءً كانت قصيرة الأجل منها أو طويلة الأمد لا يمكن أن يتمّ من دون وجود ركيزة أساسية تبنى عليها وهنا نقصد بذلك قواعد بيانات سليمة. ولكي يكون لقواعد البيانات دور ملموس في حياتنا اليومية ومعرفة مدى أهميتها، لابد أن يتم بناء أنظمة تطبيقية تقوم بتوفير بيانات ذات أهمية في سير العمل اليومي سواءً كان ذلك في جهة حكومية أو منشأة خاصة، وما أنظمة شؤون الموظفين والرواتب الآلية إلا أبسط مثال على ذلك، حيث يكون لكل موظف سجل خاص به، وهذا السجل يتكون من عدة حقول قد تكون رقم الموظف واسمه ومرتبته ودرجته وتاريخ التعيين وراتبه والإدارة أو القسم الذي يتبع له والكثير من البيانات الأخرى والتي قد تصل إلى سيرته الذاتية. ويتم عن طريق محرك البيانات التعامل مع كل هذا الكم الهائل من البيانات لخدمة عمليات البحث

والتعديل والحذف والإضافة وغيرها بيسر وسهولة. وكي يتم كل ذلك بنجاح لابد أن تتوافر لهذه القواعد من البيانات البنية التحتية Infrastructures السليمة ولاسيما أجهزة خادم الملفات Servers والأهم من ذلك كله الخبرات الفنية من العنصر البشري القادرة على بناء وصيانة وإدارة قواعد البيانات بصورة مستمرة. كما أن لهذه الخبرات دوراً كبيراً جداً في تصميم قواعد البيانات بطريقة تجعلها سهلة التعامل معها من قبل المستفيد وسرعة الوصول إلى المعلومة المطلوبة .

وهنا في هذا الكتاب حاولنا أن نضع في متناول طلاب قسم الرياضيات االتطبيقية مرجعاً مبسطاً يشرح المفاهيم الأساسية لقواعد البيانات ليضعهم على الدرجة الأولى في سلم الاحتراف في تصميم قواعد البيانات .

وفي الختام جَهِدْنا لنتفادى الأخطاء ما وسعنا، فإن وفّقنا فبفضل الله، وما لم ندركه فمن طبيعتنا، والله ولي التوفيق .

المؤلفون

الفصل الأول

مفاهيم أساسية في قواعد البيانات

1. مقدمة في قواعد البيانات

1.1. تمهيد

رغم أن مفردة قاعدة البيانات غير مألوفة للإنسان العادي البعيد عن مجال الحاسب إلا أنَّ كل إنسان يقابل ويتعامل مع قاعدة البيانات يومياً، فعندما يبحث في دليل الهاتف مثلاً فإنه يتعامل مع قاعدة بيانات، وعندما يجري اتصالاً من هاتفه النقال فإنه يبحث في قائمة الأسماء (Contacts) و بالحقيقة يبحث في قاعدة بيانات مخزنة في ذلك الهاتف، وعندما يبحث موظف معين عن بيانات أحد العملاء عن طريق السجلات الورقية الموجودة لديه فإنه يتعامل مع قاعدة بيانات. وهكذا فإن تعامل الإنسان مع قواعد البيانات يأخذ صوراً وأشكالاً متعددة لا يمكن حصرها.

عندما يقوم موظف المالية (المحاسب) بشركة ما بتنظيم البيانات الموجودة لديه فإنه يخصص مصنفاً (أو دفتراً) لكل مجموعة من البيانات التي تخصّ موضوعاً معيناً مثل دفتر الحضور والانصراف، ودفتر البيانات الشخصية، ودفتر الرواتب والأجور، ودفتر المصاريف. كما يقوم الموظف أيضاً بتخصيص صفحة في المصنف لكل موظف بالشركة، وفي صفحة الموظف يقوم بتخصيص أعمدة تمثل بيانات الموظف بالرأس و بيانات تاريخ الحضور والانصراف و ..الخ.

ما يحدث مع الحاسب لا يختلف كثيراً عن ذلك، حيث يتم إنشاء ملفاً خاصاً لكل مجموعة من البيانات التي تخص موضوعاً معيناً وهو يقابل المصنف الذي ينشئه الموظف. ثم يتم إنشاء سجلاً خاصاً بكل موظف وهو يقابل الصفحة التي يتم تخصيصها لكل موظف في المصنف ثم يتم تسجيل بيانات كل موظف في السجل الخاص به في حقل البيانات الذي يمثل كل بيان مطلوب إدخاله.

لكن هناك اختلافاً واضحاً بين إنشاء قاعدة بيانات بوساطة الإنسان وإنشائها بواساطة الحاسب، حيث أن الإنسان مثلاً يمكنه التميز مباشرةً بين البيانات الموجودة

داخل السجل، فمثلاً بالنسبة للإنسان يكون واضحاً أن "Ahmad" تمثل اسماً وليس رقم هاتف، في حين أن الحاسب لا يستطيع أن يميّز ذلك إلا عن طريق وضع قواعد معينة تجعله يستطيع التمييز بين البيانات الحرفية والبيانات الرقمية لذلك يجب تلقيم الحاسب بقيود وشروط واضحة محددة حتى نحصل على قاعدة بيانات صحيحة و مفيدة، وذلك لأن الحاسب رغم سرعته الفائقة بتنفيذ العمليات ومعالجتها إلا أنه لا يتمتع بأي قدر من الذكاء.

لذلك فإن تصميم هيكل قاعدة البيانات (Database Structure) يبدأ بتحديد المعلومات المطلوب تخزينها، ثم يتم تقسيم هذه المعلومات إلى وحدات بيانات صغيرة مثل الإسم، الجنس، تاريخ الميلاد، الهاتف، البريد الإلكتروني، العنوان. هذه الوحدات تكون مشتركة في جميع السجلات لذلك يتم تعريفها للحاسب كحقول، كما يتم تعريفها في الحاسب بنوع البيانات الموجودة ضمن هذه الحقول.

1.2. ما هي قاعدة البيانات ؟

إن قواعد البيانات وتطبيقاتها أصبحت عنصراً أساسياً في تيسير أمور الحياة اليومية في المجتمع المعاصر كما وجدنا ذلك في المقدمة، حيث إن جميع الأنشطة التي يمارسها أفراد المجتمع من تسجيل مواليد ووفيات ونتائج دراسية والسجل المدني ووثائق السفر والعمليات البنكية وغيرها الكثير يجب فيها التعامل مع إحدى قواعد البيانات.

توجد حالياً تطبيقات متقدمة لقواعد البيانات مثل استخدام الذكاء الصنعي والإسهام باتخاذ القرار كأعمال البحث العلمي و الصناعات التقنية والتجارة الإلكترونية.

1.3. تعريف قاعدة البيانات (Database Definition)

تُعرف قاعدة البيانات أنها تجميع لكمية كبيرة من المعلومات أو البيانات وعرضها بطريقة (أو بأكثر من طريقة) تُسهُل الاستفادة منها. ويمكن عدّ قاعدة البيانات مجموعةً من البيانات ذاتَ علاقة ما توصف نشاط مجتمع ما، لكن ليس كل مجموعة من البيانات يمكن تسميتها قاعدة بيانات. فقاعدة البيانات هي مصدر مركزي للبيانات مُجمَّع ومُنظَّم بطريقة يمكن أن تخدم البرامج والتطبيقات المختلفة لإنتاج نظام قواعد

البيانات Database System وهو الشكل المتكامل الذي يحتوي على البرامج والتطبيقات وعلى قواعد البيانات ويستخدم مصطلح قواعد البيانات للدلالة على شيء محدد له الخواص الآتية:

1. قاعدة البيانات تمثل رؤيا محددة للعالم الخارجي.
2. قاعدة البيانات هي مجموعة من البيانات المتماسكة منطقياً ولها معنى وتركيب محدد، أي إنَّه ليس كل مجموعة عشوائية من البيانات تمثل قاعدة بيانات.
3. قاعدة البيانات تصمّم وتُبنى وتُلقَّم بالبيانات لهدف محدد ولها مستخدمون محددون ومجموعة من التطبيقات تبنى حولها لتخدم المستخدمين.

Database: Is a Collection (or Set) of Information or Data

أمثلة:

1. دليل هاتف مدينة دمشق الذي يحوي أسماء وعناوين وأرقام هواتف سكان مدينة دمشق يعدّ قاعدة بيانات.
2. فهرس المكتبة الذي يحوي أسماء الكتب والمؤلفين ودور النشر وتاريخ الإصدار يعدّ قاعدة بيانات.
3. متاجر الكتب على الإنترنت تعدّ قواعد بيانات.
4. محرك البحث العالمي Google يمكن عدّه قاعدة بيانات.
5. موقع الـ Youtube يمكن عدّه قاعدة بيانات.
6. مواقع التواصل الاجتماعي Facebook, Twiter يمكن عدّ كل منها قاعدة بيانات.

2. المعلومات و البيانات (Information and Data)

البيانات عبارة عن تمثيل لحقائق أو مفاهيم، وهي مجموعة من المفردات التي ليس لها معنى صريح بالضرورة، مكتوبة بطريقة مرتبة تصلح لأغراض الاتصال أو أي نوع من أنواع المعالجة بوساطة الإنسان أو الآلة.

Data is defined as:

- A Representation of Facts, Concepts or Instructions In.
- A Known or Fact That Can Be Recorded and That Have Implicit Meaning.
- A Formalized Manner Suitable for Communication Interpretation or Processing by Human or Automatic Means.

كما تُعرَّف البيانات على أنها أصغر وحدة لتمثيل الحقائق والبنية الأساسية للمعلومة (الشكل 1). أما المعلومات فهي المعرفة في شيء ما، خاصة، والمستمدة من حدث ما عامة ، أو هي المعرفة القائمة على البحوث أو الخبرة وهي البيانات التي تمت معالجتها وتنظيمها لاستخدامها بشكل واضح.

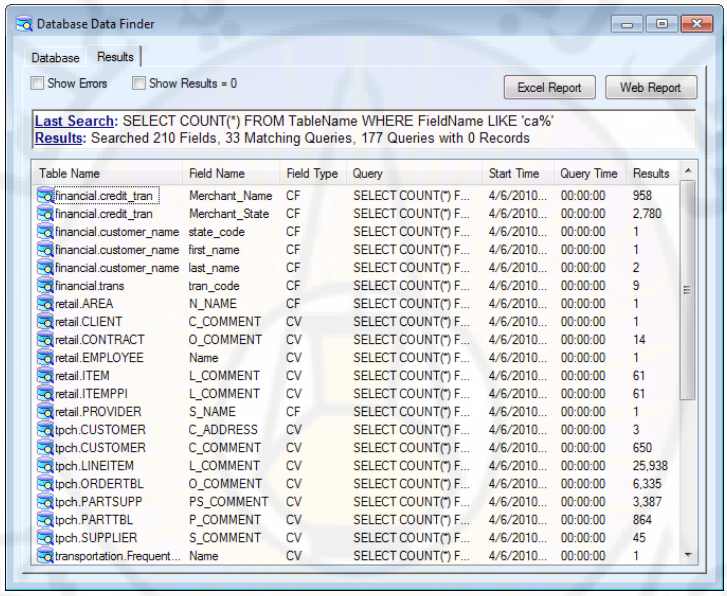


الشكل 1: صورة توضيحية للبيانات الحاسوبية.

Information is defined as:

- The Knowledge of Something; Particularly, an Event, Situation, or Knowledge Derived Based on Research or Experience.
- A Processed ,Organized or Summarized Data

غالباً ما تُستخدَم البيانات والمعلومات بالتبادل فهما على ارتباط وثيق، فالمعلومات هي البيانات التي تمت معالجتها والبيانات هي الوحدة الأساسية للمعلومات، وعندما يتم معالجة المعلومات نحصل على معلومات تخدم صناعة القرار. بالتالي مصطلح المعلومات هو حصيلة معالجة البيانات (Data Processing) حيث تصبح منظمة ولها معنى واضح محدد ويمكن الاستفادة منها بشكل أو بآخر وبالتالي ينتج مايسمى نظام المعلومات.



الشكل 2: صورة توضيحية لاستخدام قاعدة البيانات.

3. أهمية قاعدة البيانات

تكمن أهمية قواعد البيانات في النقاط الآتية:

1. الحاجة إلى آلية لتنظيم البيانات التي أصبحت في عصرنا كبيرة ومتنوعة، والمحافظة عليها من التلف وضمان أمنها وخصوصيتها. وعليه تعدّ قواعد البيانات ذات أهمية قصوى تقدم لأي مجتمع ولاسيما ونحن في عصر التقانة والمعلوماتية.
2. سهولة الإدارة: ويقصد بها التنظيم والفهرسة والقدرة على اتخاذ القرارت.
3. سهولة في التخزين والإضافة والتعديل.

4. السرعة في البحث والاستعلام.
5. السهولة والسرعة في إصدار التقارير والتحليل الإحصائي.

3.1. مميزات قواعد البيانات

تمتاز قواعد البيانات في أنها تمكننا من تحقيق الأمور الآتية:

1. تجنب تكرار البيانات: حيث يتم الوصول إلى صيغة يكون فيها التكرارٌ محدداً ويقتصر على الجوانب الخاصة بربط البيانات بعضها ببعض.
2. تجانس البيانات: ويقصد بها السيطرة على عمليات الإضافة والحذف والتعديل بشكل يحافظ على بنية متجانسة داخل قاعدة البيانات.
3. قابلية التطور عند وجود متطلبات جديدة أو بناء نظام جديد.
4. استقلالية البيانات عن التطبيقات.
5. سهولة استرجاع البيانات بالشكل الذي تفرضه رؤية المستخدم ومتطلبات النظام.
6. إمكانية مشاركة أكثر من مستخدم في الوصول إلى البيانات والتعامل معها.
7. إمكانية ربط أكثر من نظام قاعدة بيانات.
8. السرية والتحكم.

3.2. من يستثمر قواعد البيانات؟

نحن جميعاً نستخدم قواعد البيانات، وغالباً دون أن ندري. فعند استخدام بطاقة الصراف الآلي لسحب مبلغ من المال من أحد البنوك فإنه يتم الوصول إلى قاعدة البيانات بطريقة غير مباشرة، يتم من خلالها تسجيل عملية سحب المبلغ المطلوب ويتم تسجيل الحركة التي تمت بالمبلغ المسحوب وتاريخها ومن أي كوة صرافة تمت، ويتم احتساب الرصيد الجديد أيضاً. يتضح لنا من المثال السابق أنه ليس بالضرورة على مستخدم قاعدة البيانات أن يكون خبيراً أو ماهراً بالتعامل مع الحاسب.

3.3. بيئة قاعدة البيانات (Database Environment)

يمكن لقواعد البيانات أن تعمل في بيئات مختلفة، فقد تعمل على حاسبات كبيرة Mainframe ويتاح الاتّصال بها من خلال وحدات طرفية Terminals، وقد تعمل في بيئة العميل/الخادم Client/Server Environment، وتُحفظ قواعد البيانات على جهاز الخادم ويُتاح الاتصال بمجموعة من أجهزة العميل بها بشكل مباشر باستخدام تطبيقات وسيطة تسمح بذلك، وقد يتم الاتصال بين العميل والخادم عبر جهاز وسيط يستضيف التطبيقات اللازمة للاتصال بقاعدة البيانات والتحكم في عدد المستخدمين لقاعدة البيانات في ظل البيئة متعددة الطبقات Multitier Environment. ويُتاح الاتصال أيضاً بقاعدة البيانات من خلال خادم وب وخادم للتطبيقات عند استخدام بيئة الوب Web Environment.

ومنه نجد أنه ليس من الضروري وجود بيئة حاضنة (موطن) لقاعدة البيانات "إنّ صح التعبير" في المكان الذي يتواجد فيه المستخدمون فهناك دائماً وسيلة اتصال تمكن من الوصول إلى البيانات، فيمكن للمستخدم تنفيذ أنواع مختلفة من المهام حسب البيئة التي يُسمح له بالتخاطب معها، فالمستخدم يمكن أن يقوم بمهمة قد تأتي من داخل بيئة قاعدة البيانات، أوقد تنشأ من خارج البيئة. يمكن للمستخدم أيضاً تنفيذ أنواع مختلفة من المهام وتلبية احتياجات قاعدة البيانات والمستثمر معاً من خلال خلق لبيانات جديدة وتعديل هذه البيانات، أو محاولة لخلق معلومات جديدة، مع ضبط طريقة الوصول إلى البيانات.

بشكل عام يمكن تصنيف بيئات قاعدة البيانات وفق ما يأتي:

1- البيئة المركزية

(Mainframe Environment)

2- بيئة العميل /الخادم

(Client/Server Environment)

3- بيئة الحوسبة الإنترنت

(Internet Computing Environmen-Web Environment)

فيما يأتي سنقوم بتفصيل أنواع بيئات قاعدة البيانات:

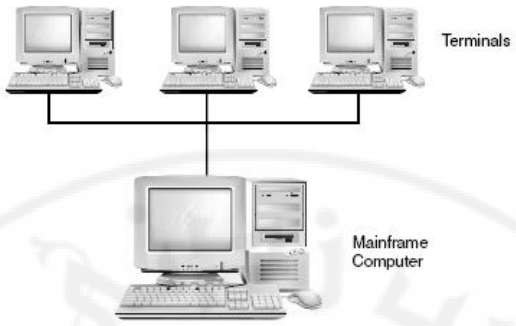
1- البيئة المركزية

(Mainframe Environment)

كانت البيئة التقليدية لنظم قواعد البيانات في وقت سابق هي البيئة المركزية. تتألف البيئة المركزية أساساً من جهاز حاسوب مركزي قوي يتم الوصول إليه من خلال محطات عمل (وحدات طرفية Terminals). ويتم ربط هذه المحطات بشبكة إلى الحاسوب المركزي بطريقة تتيح للمستخدم التواصل مع جهاز الحاسوب المركزي، على أن المحطات هي في الأساس من ملحقات أجهزة الحاسوب المركزي وهي ليست أجهزة مستقلة، أي إن المحطة تعتمد على الحاسوب المركزي لإجراء أي عمل. وجدير بالذكر أن واحدة من المشاكل الرئيسية في البيئة المركزية هي القيود التي يتم وضعها على مستخدم النصوص.

على سبيل المثال، يمكن للمحطة التواصل فقط مع الحاسوب الرئيسي، كما يمكن أن تشمل مهاماً أخرى من العمليات المكتبية كاستخدام معالج.

ومعظم الشركات اليوم هجروا نظمهم للبيئة الجديدة العميل/الخادم. يوضح الشكل 3 البيئة المركزية.



الشكل 3: رسم توضيحي للبيئة المركزية.

2- بيئة العميل /الخادم

(Client/Server Environment)

بشكل عام في هذا النظام، يتم توزيع العمليات بين أجهزة (Clients) وجهاز (Server) بحيث يتم تحزين البيانات على الجهاز الرئيسي (Server) وتتم معالجة البيانات عليه، في حين يتعامل مستخدمو النظام مع أجهزة (Clients) التي تضم برامج المستخدم ويجري بها التفاعل بين البرامج وبين مستخدمي هذه البرامج، وإرسال البيانات واسترجاعها على شاشات البرامج. في هذا النظام يتم توزيع البرامج بشكل عام بين الجهاز الرئيسي (Server) والأجهزة الفرعية (Clients).

وقد تم حلّ العديد من المشاكل التي كانت موجودة في البيئة المركزية مع التكنولوجيا العميل/الخادم.

بيئة العميل/الخادم تقوم بالاعتماد على الحاسوب الرئيسي، يدعى بالخادم، وهو واحد أو أكثر من أجهزة الحاسوب الشخصية التي يتم ربط قاعدة البيانات من خلال الملقم والذي يدعم بوساطة كياناً منفصلاً من أجهزة الحاسوب الشخصية. وكون العميل هو نظام حاسوب منفصل، فواحدة من المشاكل في بيئة العميل /الخادم هو أنه عندما يتم تطوير نسخة جديدة من التطبيق، يجب إعادة تثبيت التطبيق على كل جهاز، والتي يمكن أن تكون مملة للغاية وتستغرق وقتاً طويلاً جداً.

على الرغم من أن تكاليف إضافية يتم تكبدها من خلال إنشاء والحفاظ على التطبيق على جهاز الحاسوب إلا أنه يوجد العديد من الفوائد لهذه البيئة، منها أن العميل هو جهاز حاسوب مستقل بحد ذاته، كونه لديه موارد خاصة به (وحدة المعالجة المركزية والذاكرة والتخزين على القرص)، ويمكن أن تشارك في بعض تجهيز التطبيق، وبالتالي اتخاذ بعض العبء الكلي من الخادم وتوزيع العمل على العملاء كافة، لأن أجهزة الحاسوب يمكنها "التفكير" من تلقاء نفسها وتشغيل التطبيقات الأخرى، فيمكن للمستخدمين أن يكون أكثر إنتاجية. على سبيل المثال، يمكن توصيل المستخدم إلى قاعدة البيانات على الملقم بينما كان يعمل في وقت واحد مع وثيقة والتحقق من البريد الإلكتروني.

3- بيئة الحوسبة الإنترنت

(Internet Computing Environment)

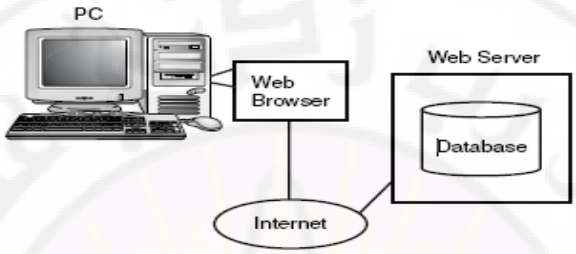
الحوسبة والإنترنت هي مشابهة جداً لـ الحوسبة عميل/الخادم. لكن الحوسبة الإنترنت هي فريدة من نوعها وذلك بسبب اعتمادها على شبكة الإنترنت.

في بيئة العميل/الخادم، التي قد تكون مقيدة للمستخدم في الوصول إلى الأنظمة التي هي داخل الشبكة الداخلية للشركات. في كثير من الحالات، نجد أن الأجهزة العميلة لا يمكن عن طريقها الوصول إلى قواعد البيانات من خارج الشبكة الداخلية للشركات، ولكن برامج إضافية حسب الطلب قد تجعلها متاحة.

إن الشفافية في التطبيق إلى المستخدم النهائي هي واحدة من جوانب الحوسبة الإنترنت التي تجعلها قوية جداً.

في بيئة الحوسبة الإنترنت يتم تثبيت التطبيق على خادم واحد يدعى خادم ويب .يجب أن يكون المستخدم قادراً على الاتصال بالإنترنت و الاتصال بمستعرض ويب معتمد مثبت على جهاز الحاسوب (الشكل 4).

يتم استخدام مستعرض ويب للاتصال إلى الوجهة URL الخاص بخادم الويب .أما ملقم ويب، بدوره يقوم بالوصول إلى قاعدة البيانات بطريقة معتمدة من قبل التطبيق، وإرجاع المعلومات المطلوبة لمستعرض ويب المستخدم، ليتم عرض النتائج على جهاز المستخدم من قبل مستعرض الويب.



الشكل 4: رسم توضيحي لبيئة الحوسبة الإنترنت.

إن إعداد التطبيق للمستخدم النهائي تم لتبسيط الصيانة في بيئة الحوسبة الإنترنت أي أنه لا يوجد شيء لتثبيته أواعداده على جهاز الكمبيوتر الخاص بالمستخدم. وعندما يتم إجراء تغييرات على التطبيق، يتم إجراء تغييرات في مكان واحد، على سبيل المثال، على ملقم الويب الشكل 4 يبين رسم توضيحي لبيئة الحوسبة الإنترنت.

الإنترانت – التطبيق المعتمد على الوب

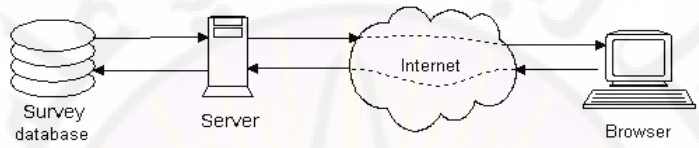
(Web Based Application)

يعتمد هذا النمط على فكرة ثورية للغاية ظهرت منذ فترة ليست ببعيدة، وهي استخدام مخدم ويب (Server Web) وسيطاً بين المستثمر وقاعدة البيانات كما في الشكل 5. أي استخدام تقنيات الإنترنت الشائعة والمتطورة والمجربة والسهلة الاستخدام في تطوير حلول الأتمتة للمؤسسات.

في البداية بدت الفكرة غريبة، ولكن التطبيق العملي أظهر نتائج باهرة وفوائد ممتازة لم تكن في الحسبان. لقد أصبحت هذه الطريقة خيار المؤسسات التي تعمل على مساحات واسعة جغرافياً، لأنها تمثل أفضل حل ممكن حالياً، كما أنها أحد الخيارات الأساسية للمؤسسات الصغيرة لأنها تقدم مجموعة من الميزات لا يمكن الحصول عليها بنمط آخر.

في هذا النوع من الحلول تكتسب البرامج معنى آخر، فالبرنامج هو موقع ويب على المخدم، والواجهة هي صفحة ويب.

يمكن استخدام هذا النمط من الحلول في جميع الحالات، حتى على حاسب واحد، ولكنه يعطي أفضل استثمار على شبكة محلية أو واسعة. وهو الحل الأفضل أمام مؤسسة تنتشر مشاريعها أو فعاليتها على عدة مدن أو دول.



الشكل 5: Web Based Application

4. نظام إدارة قواعد البيانات

(Database Management System, DBMS)

4.1. تعريف نظام إدارة قواعد البيانات

هو النظام الذي يُسهّل إنشاء وصيانة قواعد البيانات، و يأتي عادةً على شكل حزمة برمجية تتكون من تطبيقات تقدم للمستخدمين بمختلف أنواعهم خدمات الوصول إلى البيانات ومعالجتها، ويكون حلقة الوصل بين التطبيقات وبين البيانات، وبين البيانات وبين مسؤول قاعدة البيانات أيضاً. فقاعدة البيانات لا يجوز التعامل معها مباشرة بل تحتاج إلى نظام إداري متكامل لبناء هيكل قاعدة البيانات من بيانات وصفية (Meta-Data) و بيانات فعلية Data.

ويتميز نظام قاعدة البيانات بالنقاط الآتية:

1- يقوم نظام إدارة قواعد البيانات بمعالجة الوصول الآمن للبيانات المخزنة ضمن قاعدة بيانات واحدة وذلك من قبل أكثر من مستخدم واحد في نفس الوقت

2- بوجود لغة أو لغات قياسية (Database Languages) للتعامل مع أنظمة إدارة قاعدة البيانات من إدخال البيانات و تعديل و حذف وإدارة.

3- يستفيد DBMS من المصادر المتاحة على جهاز الحاسوب بشكل ديناميكي وهذا يؤهل عدداً كبيراً من تطبيقات المستثمرين لإنجاز أعمالهم بشكل سريع.

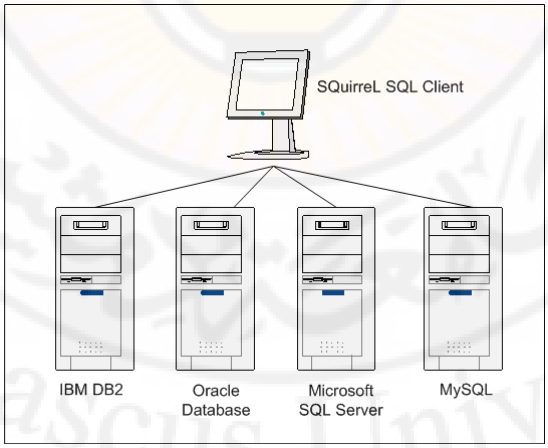
4- يقوم نظام إدارة قواعد البيانات بحماية البيانات بطريقة معينة تسمح ببقاء البيانات سليمة بالإضافة إلى القدرة على إعادة بناء البيانات التالفة في حال حدوث خطأ ضمن التجهيزات المادية للنظام.

5- يمكن أن تكون أنظمة إدارة قواعد البيانات مصممة لتخدم أكثر من مستخدم واحد بالوقت نفسه (Multi-User). ونظام إدارة قواعد البيانات هو أحد مكونات نظام قاعدة البيانات ويتكون نظام قاعدة البيانات من :

I. قاعدة البيانات: والتي تتكون من توصيف (تعريف) البيانات Meta-Data، ومن البيانات الخام المخزنة، كل على حده. ويقصد بتعريف البيانات Meta-Data (بيانات تصف خصائص وسياق بيانات المستخدم)

II. نظام إدارة قاعدة البيانات: مثل MySQL - SQLServer -Oracle.

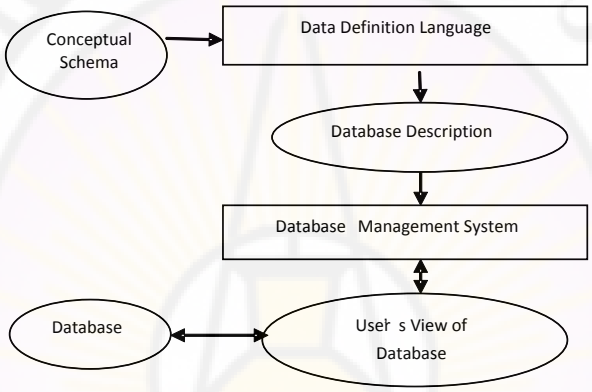
III. التطبيقات البرمجية والاستعلامات Queries and Application Programs : يتم استرجاع البيانات من قاعدة البيانات إما مباشرةً باستخدام لغة الاستعلامات أو



الشكل 6: مخدمات لنظم إدارة قواعد بيانات.

عن طريق واجهة التطبيق البرمجي، وهي واجهات إدخال وواجهات إخراج البيانات والمعلومات المطورة بوساطة إحدى لغات برمجة التطبيقات.

IV. المستخدمين والمبرمجين Users and Programmers: و هو المستخدم النهائي ويدخل في ذلك المبرمج Programmer ومدير قاعدة البيانات (Database Administrator, DBA)، رغم اختلاف المهام و طرق الوصول إضافة إلى المستثمر أو المستخدم النهائي End User.



الشكل 7 نظام إدارة قواعد البيانات.

4.2. مستخدمو قاعدة البيانات

Data Users

هم الأشخاص الذين يستخدمون أو يديرون قاعدة البيانات ومحتوياتها والذين يصممون ويطورون ويحافظون على تطبيقات قواعد البيانات. ويُصنف المستخدمون إلى قسمين (الشكل 8):

1-المستخدمين غير المباشرين ويسمون مجازاً Work Behind The Scene أي عاملون خلف المنصة ومن أمثلتهم مدير قاعدة البيانات Database Administrator ومصمم قاعدة البيانات Database Designer.

a.مدير قاعدة البيانات

Database Administrator

وهو المسؤول عن منح الحقوق وشرعية الوصول إلى قاعدة البيانات وصيانتها، وفي تنسيق ومراقبة الاستخدام والحصول على موارد البرمجيات والعتاد والتحكم في الاستخدام ومراقبة كفاءة العمليات.

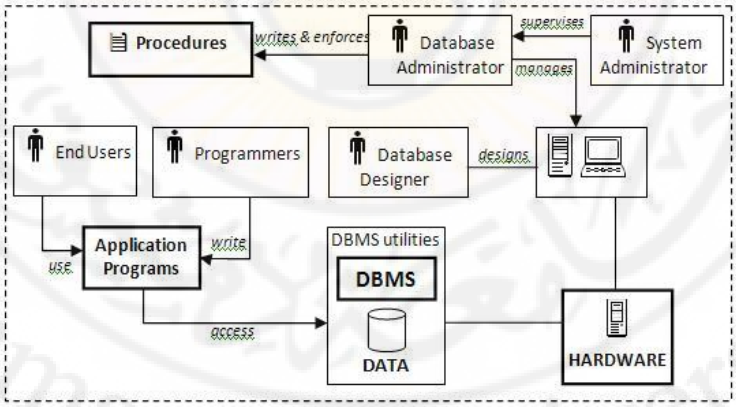
b.مصمم قاعدة البيانات

Database Designer

وهو المسؤول عن تعريف المحتوى وكتالوج قاعدة البيانات وكذلك شروطها ووظائفها وومعاملاتها بحيث يتواصل المصمم مع المستخدم النهائي End-User لتفهم احتياجاته وتضمينها في بناء قاعدة البيانات.

2-المستخدمين المباشرين ويسمون مجازاً Actors on the Scenes ممثلين على المنصة:

وهو المستخدم النهائي وهو القادر على استثمار قاعدة البيانات للإستعلام واستخراج التقارير.



الشكل 8 : مستخدمو نظام إدارة قواعد البيانات.

في النهاية يمكن تلخيص تعريف نظام إدارة قواعد البيانات DBMS بأنه مجموعة من البرامج التي تدير وتتحكم بعملية تخزين واسترجاع البيانات وكذلك توفر إمكانية عدد

كبير من المستخدمين من الوصول إلى قاعدة البيانات والتعامل معها وينظر إليها على أنها حلقة وصل بين المستخدمين وقاعدة البيانات حيث تقوم باستقبال طلبات المستخدمين ومن ثم نقلها إلى قاعدة البيانات وتنفيذ البرامج اللازمة لتلبية هذه المتطلبات ثم تزويد المستخدم بالنتائج المطلوبة.

ومن أشهر نظم إدارة قواعد البيانات هي أوراكل Oracle، مايكروسوفت اكسس Ms Access، بالإضافة لـ SQL-Server. من الناحية الإدارية هناك هدف لنظام إدارة قواعد البيانات وهو المساعدة في التخطيط واتخاذ القرارات.

لابدّ من المقارنة بين قاعدة البيانات التي تتكون من مجموعة من الملفات المرتبطة معاً، ونظام إدارة قواعد البيانات الذي يمثل مجموعة من البرمجيات التي تدير بكفاءة مجموعة من البيانات المترابطة.

والخلاصة العملية العلمية هي أن نظام إدارة قواعد البيانات يتطلب إعادة تنظيم الدور الإستراتيجي للمعلومات والبدء بفاعلية لإدارة تخطيط تلك المعلومات كمورد إستراتيجي، وهذا يعني أنه يجب معرفة المتطلبات من المعلومات حتى تتطور وظيفة إدارة البيانات. فإدارة البيانات عبارة عن وظيفة تنظيمية لإدارة مورد البيانات، و هي المسؤولة عن إيجاد سياسة للمعلومات وإجراءات التأمين للمعلومات بجودة معيارية، والتي تجعل البيانات تدار كمورد تنظيمي، كما تتضمن تطوير سياسات المعلومات، والتخطيط للبيانات، وتصميم قواعد البيانات، وتطوير قاموس البيانات.

5. **لغات نظام إدارة قواعد البيانات**

(DBMS Languages)

تعدّ لغة الاستعلامات البنيوية أو الهيكيلية (Query Structure, SQL Language) اللغة الرئيسية لبناء هيكل قاعدة البيانات ومعالجتها. من خلال هذه اللغة يتم التعامل مع قاعدة البيانات. باعتبار قاعدة البيانات مؤلفة من بيانات وصفية، وبيانات فعلية، فلغة الـ SQL يمكن أن تنقسم إلى عدة لغات. من أهم هذه اللغات يمكن أن

نذكر لغة تعريف أو توصيف البيانات(Data Definition Language, DDL) ولغة معالجة البيانات (Data Manipulation Language, DML) .

لغة الـ DDL هي اللغة المسؤولة عن بناء هيكل قاعدة البيانات من إنشاء وتعديل وحذف (البيانات الوصفية). يستخدم لغة DDL مدير قاعدة البيانات DBA أو المستخدم لإعطاء الوصف الشامل لهيكل قاعدة البيانات Schema و الكينونات Entities و الواصفات Attributes و العلاقات Relationships بالإضافة إلى قيود الخصوصية Constraints والأمان Security.

أما لغة الـ DML فهي اللغة المسؤولة عن معالجة البيانات سواء من قراءة أو كتابة أو حذف أو تعديل على البيانات الفعلية الخاصة بالمستخدم. توفر هذه اللغة عمليات معالجة البيانات الأساسية على البيانات الموجودة في قاعدة البيانات، و إجراءات لغة الـ DML تتيح للمستخدم مخاطبة النظام من خلال ضبط كيفية التعامل مع البيانات كما تسمح للمستخدم أن يذكر ما هو مطلوب من البيانات وصياغتها على أساس معلومات.

بالإضافة للغتين السابقتين هناك لغة أخرى وهي لغة التحكم في البيانات (Data Control Language, DCL). تتولى لغة الـ DCL معالجة صلاحيات المستخدمين وهي تُستخدَم لتأمين قاعدة البيانات عن طريق السماح للمستخدمين في الوصول إلى بعض أجزاء معينة من قاعدة البيانات. يبيّن الجدول 1 بعض تعليمات اللغات السابقة مع شرح موجز لكل تعليمة باللغة الإنكليزية وسنقوم بدراسة هذه التعليمات بالتفصيل في الفصل الخامس من هذا الكتاب.

DCL	DML	DDL
Grant Revoke Grant	Insert Select Update Delete	Create Drop Alter

الجدول 1: بعض تعليمات لغات قواعد البيانات.

What Is The Difference Between DDL, DML and DCL Commands?

- DDL Statements Are Used To Define The Database Structure Or Schema. Some Examples:
  - Create: To Create Objects In The Database
  - Alter: Alters The Structure Of The Database
  - Drop: Delete Objects From The Database
  - Truncate: Remove All Records From A Table, Including All Spaces Allocated For The Records Are Removed
  - Comment: Add Comments To The Data Dictionary
  - Rename: Rename An Object
- DML Statements Are Used For Managing Data Within Schema Objects. Some Examples:
  - Select: Retrieve Data From A Database
  - Insert: Insert Data Into A Table
  - Update: Updates Existing Data Within A Table
  - Delete: Deletes All Records From A Table, The Space For The Records Remain
  - Call: Call A PI/SQL Or Java Subprogram
  - Explain Plan: Explain Access Path To Data
  - Lock Table: Control Concurrency
- DCL Statements. Some Examples:
  - Grant: Gives User's Access Privileges To Database
  - Revoke: Withdraw Access Privileges Given With The
  - Grant Command

يوجد أيضاً لغة تمكننا من إدارة المناقلات Transactions و هي لغة التحكم بالمناقلات (Tcl Transaction Control, Tcl)

- Tcl Statements Are Used To Manage The Changes Made By DML Statements. It Allows Statements To Be Grouped Together Into Logical Transactions.
  - Commit: Save Work Done
  - Savepoint: Identify A Point In A Transaction To Which You Can Later Roll Back
  - Rollback: Restore Database To Original Since The Last Commit
  - Set Transaction: Change Transaction Options Like Isolation Level And What Rollback Segment To Use

مع الملاحظة أنه في نظام إدارة قواعد البيانات هناك ما يعرف بمعالج لغة تعريف أو توصيف البيانات (DDL Compiler) حيث يقوم بمعالجة جمل لغة تعريف البيانات من أجل تحديد وصف لتراكيب اللغة بما فيها مخطط قاعدة البيانات وتخزين هذا الوصف في فهرس نظام إدارة قواعد البيانات.

أنواع البيانات المخزونة في القاعدة

1- البيانات الفعلية Actual Data: هي البيانات التي تم جمعها وتسجيلها في القاعدة من أجل تمثيل ووصف الأحداث التي في القاعدة ومن أمثلتها بيانات العملاء أو بيانات الموردين أو المنتجات أو بيانات الطلاب.

2- البيانات الوصفية Meta Data: وهي بيانات عن البيانات فهي لاتصف الأحداث الداخلة في مجال القاعدة، لكنها تصف هيكلاً أو منظور قاعدة البيانات التي تستخدم في تسجيل البيانات وتتميز بما يأتي:

a. تتضمن منظور قاعدة البيانات، البيانات التي يجب تخزينها في القاعدة وطريقة التخزين واسترجاع البيانات على وسائط التخزين.

b. يتم تسجيلها على شكل تعليمات مكتوبة بلغة تعريف البيانات DDL.

c. يتم تخزين هذه التعليمات داخل مرجع خاص من مراجع النظام يسمى قاموس البيانات Data Dictionary.

3- الفهارس: Indexes وهي جداول تكميلية لفرز السجلات أو البيانات على أساس تسهيل وزيادة سرعة الوصول إلى البيانات المطلوبة التي تحتوي على قيم معينة.

4- البيانات الإحصائية: وهي التي تتضمن مؤشرات عن أحوال البيانات المخزنة في قاعدة البيانات و استخدامتها الحالية. ومن أهم استخداماتها تساعد نظام إدارة قواعد البيانات في اختيار أفضل طريقة لتنفيذ الاستفسار الحالي.

ترجع أهمية قاموس البياناتData Dictionary إلى إمكانية استرجاع ومعالجة أي بيان مخزن في القاعدة. كما يساعد المستخدمون الخبراء في تكوين فكرة كاملة عن الهيكل الحالي لقاعدة البيانات.

6. أنواع نظم إدارة قواعد البيانات

هناك ثلاثة أنواع شائعة من نظم إدارة قواعد البيانات:

1. نظم إدارة قواعد البيانات الهرمية

Hierarchical DBMS

2. نظم إدارة قواعد البيانات الشبكية

Network DBMS

3. نظم إدارة قواعد البيانات العلائقية

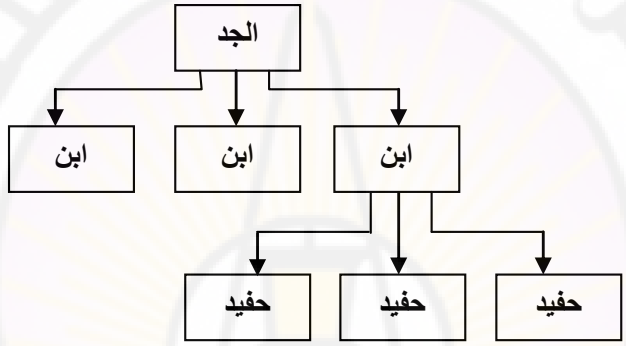
Relational DBMS

فيما يأتي سنقوم بإعطاء فكرة عن النموذج الهرمي و النموذج الشبكي، ثم سنقوم بدراسة النموذج العلائقي بشيء من التفصيل.

6.1. نظم إدارة قواعد البيانات الهرمية

Hierarchical DBMS

قواعد البيانات الهرمية أو النظم الهرمية تقوم بتنظيم البيانات على شكل هرمي أو على شكل شجرة مقلوبة أي جذرها في القمة وتخرج منها الفروع. شأن هذه التركيبة شأن شجرة الأسرة فلها جد واحد والجد له عدة أبناء والأبناء هم أباء الأحفاد ويستحيل وجود حفيد له أكثر من أب. الشكل 9 يوضح النظم الهرمية وتفرعاتها.



الشكل 9 : النموذج الهرمي يوضح النظم الهرمية وتفرعاتها.

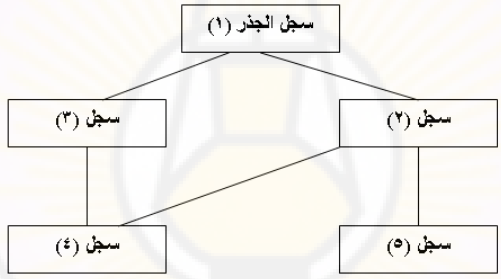
الملفات الهرمية هي ملفات لها البناء الشجري نفسه و لها العلاقات بين السجلات نفسها، ومن الأمثلة لبعض أنواع السجلات التي يمكن أن تتواجد في تكوين هرمي سجلات مبيعات متعددة لكل بائع، حيث يوجد سجل إحصائيات واحد لكل عملية جارية كما يوجد أيضاً سجلات عديدة للعملاء لكل بائع، حيث إن كلّ بائع له عملاء محددون ويمكن أن يكون لكل عميل عدة سجلات حسابات مدينين: سجل واحد لكل عملية شراء لم يتم تسديد ثمنها.

من المهم أن نفهم أنه ليس من الضروري أن تتصل كل الملفات الموجودة في قاعدة البيانات مع بعضها، وكل ما هو مطلوب أن تتصل الملفات التي تُستخدم كمجموعة مع بعضها في التطبيقات. سجلات المبيعات السابقة لها مثل هذه العلاقة

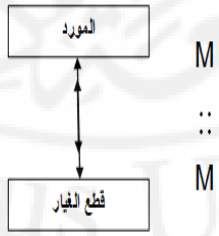
المنطقية تسمى فئة. والفئة Set عبارة عن مجموعة من السجلات متصلة مع بعضها منطقياً. على هذا تصبح قاعدة البيانات الهرمية عبارة عن تجميع لملفات وفئات ملفات متصلة مع بعضها منطقياً. يستخدم نظام إدارة المعلومات (Information Management, System, IMS الذي أعدته شركة IBM التكوين الهرمي وهو من أكبر نظم إدارة قواعد البيانات DBMS الموجودة حالياً وأعقدها. لهذا السبب فهو يتطلب مستوى رفيعاً من الخبرة لإمكانية بنائه وعلى أي حال فهو قوي و أثبت كفاءة كبيرة في المعاملة مع قواعد بيانات كبيرة جداً. كما أنه يقدم إجراءات استرجاع وأمن جيدة. هذا بالإضافة إلى إمكانية استخدامه في نظام الاتصال النشط من خلال شبكة الاتصالات.

6.2. نظم إدارة قواعد البيانات الشبكية

(Network DBMS)



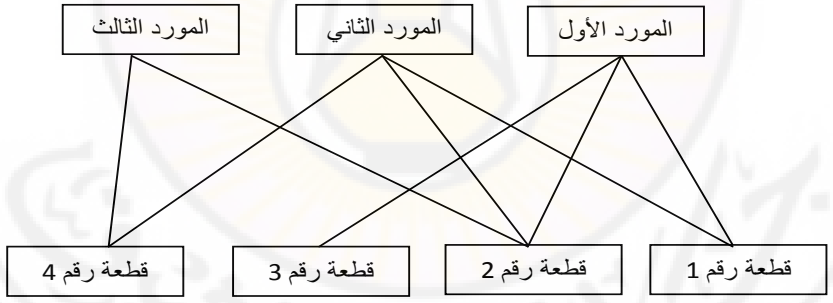
الشكل 10: النموذج الشبكي.



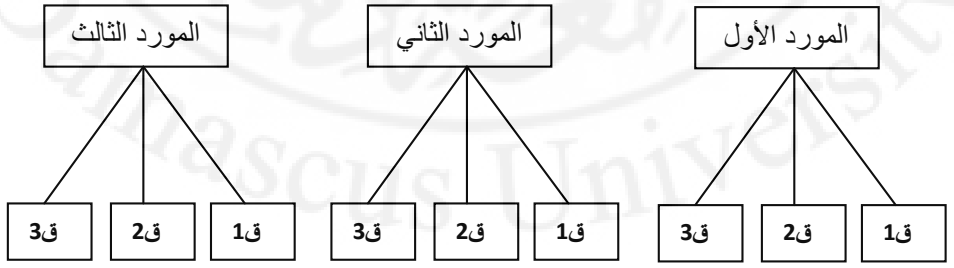
الشكل 11: العلاقة بين المورد و قطع الغيار.

رغم أن كلمة الشبكة استخدمت كثيراً في شبكات الحاسب ومعالجة البيانات فقد وجد من الأفضل استخدام مسمى قواعد البيانات الضفير Plex رغم أن مسمى قواعد البيانات الشبكية لازال شائع الإستخدام. يتغلب هيكل بيانات التركيب الشبكي على معوقات التكوين الهرمي الذي لا يسمح ابن أن يكون له أكثر من أب واحد و يظهر ذلك في الشكل التوضيحي للتكوين الشبكي (الشكل 10) حيث نلاحظ أن للسجل رقم (4) عائلين هما السجل رقم (2) و السجل رقم (3).

مثل هذا النوع من قواعد البيانات حلَّ كثيراً من مشاكل العلاقات. فإذا فرضنا أن هناك أكثر من مورد يورد قطع غيار فإن كل مورد قادر على توريد أكثر من نوعية قطعة غيار. بالتالي كل قطعة غيار يوردها أكثر من مورد و كل مورد قادر على توريد أكثر من نوعية قطعة غيار. أي إن كل قطعة غيار يوردها أكثر من مورد. لفهم هذا المثال بشكل أوضح يتم عرض العلاقة بين قطعة الغيار والموردون على النحو الموضح في الشكل 11.



الشكل 12: توضيح العلاقة بين المورد و قطع الغيار باتجاهين.

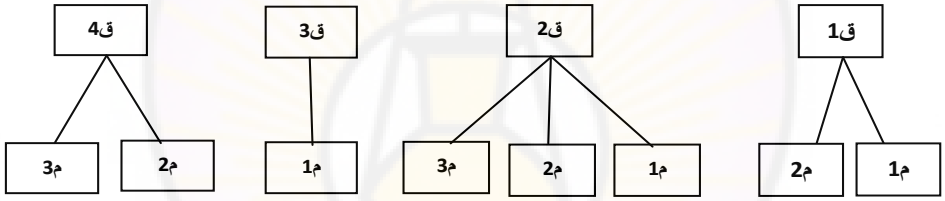


الشكل 13: توضيح العلاقة مورد(م) – قطع غيار (ق).

لتوضيح هذه العلاقة يجب إعادة رسم العلاقات في الاتجاهين كما هو موضح في الشكل 12. الشكل 13 يوضح العلاقة مورد – قطع غيار و الشكل 14 يوضح العلاقة قطع غيار – مورد .

من الأشكال السابقة يتضح لنا بما لا يقبل الشك أن تبسيط العلاقة الشبكية إلى علاقة هرمية أوجد تعقيدات أكثر حيث حولها إلى نوعين من شجرة العلاقات و في هذا جهد إضافي في التنفيذ.

إن ما عرضناه حول العلاقات الشجرية (الهرمية) وقواعد البيانات الشبكية يؤكد أن كليهما يمكن تحقيقه و إن كانت بعض حزم إدارة قواعد البيانات يمكنها التعامل فقط مع الشكل الشجري في حين أن البعض الآخر يمكنه التعامل مع النوع الشبكي. كما أن هناك تنوعاً في برامج إدارة قواعد البيانات، فبعض برامج إدارة قواعد البيانات الهرمية لا



الشكل 14: توضيح العلاقة قطع غيار(ق) –مورد(م).

تتعامل مع العلاقات البسيطة والبعض يمكنه التعامل مع العلاقات المعقدة.

أوجه التشابه بين نظم قواعد البيانات الشبكية وبين نظم قواعد البيانات الهرمية أنها تتطلب ذواكر ذات أحجام كبيرة، وعادةً تحتاج إلى لغات راقية لبرمجتها وهي صعبة التعلم ولها مزايا كثيرة فهي بالطبع أكثر كفاءة من قواعد البيانات العلائقية ويتعاملان مع كم كبير جداً من البيانات و المعلومات بالإضافة إلى أنها توفر بناءً على طريقة تنظيم البيانات التي تتبعها مساحات كبيرة من وسائط تخزين البيانات.

6.3. نظم إدارة قواعد البيانات العلائقية

(Relational DBMS)

أثبتت الأيام صحة القول الشائع أن الأبسط هو الأجمل والأكفأ. فكلما كان سكنك بسيطاً و كلما عشت في بساطة وبعُدت عنك المشاكل و كلما كانت الآلة بسيطة سهُلت إدارتها وصيانتها. وهذا ما أكده التعامل مع قواعد البيانات الهرمية والشبكية التي تعقدت ملفاتها وأساليب إدارتها لدرجة كادت تودي بها، كلما أُضيفت تطبيقات جديدة أو متطلبات جديدة تحتاج مؤشرات جديدة، مما ضخم منها وعقدها. هذه المشاكل كانت المنطلق للبحث عن حلول تحقق جملة أهداف منها:

1. يمكن فهم و استخدام قاعدة البيانات لمن لم يدرسوا علوم الحاسب.
2. يمكن تعديل وإضافة وحذف بيانات دون تغيير المخطط المنطقي للقاعدة.
3. تتيح للمستخدم أعلى درجة من المرونة في التعامل مع البيانات.

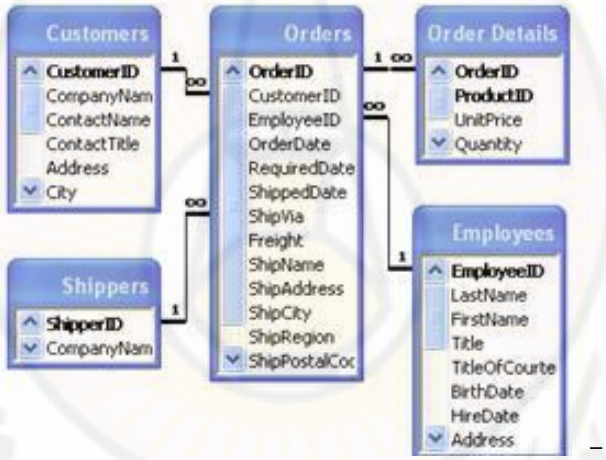
في عام 1970 استحدث العالم الأمريكي E.E.Codd أسلوباً لتنظيم وفرز بيانات قواعد البيانات، وهي قواعد البيانات العلائقية (الشكل 15). وجد E.E.Codd أن هذا لا يتحقق إلا برص البيانات على هيئة جداول كون الإنسان تعود على الجداول منذ طفولته بدايةً من جدول الحصص إلى جدول الضرب إلى كشف الأسماء والدرجات.

هذه النظم تتعامل مع أكثر من ملف في نفس الوقت وتعامل البيانات داخل الملف كما لو كانت جدولاً مكوناً من صفوف وأعمدة و يسمى علاقة Relation. تمثل أعمدة الجدول حقول قاعدة البيانات Fields وتسمى أيضاً Attributes بينما تمثل صفوفها سجلات قاعدة البيانات وتسمىTuples. يقوم النظام العلائقي Relational System بربط البيانات بين العلاقات بناءً على حقل مشترك بينهما.

قامت النظم العلائقية أساساً على النظريات العلائقية في الرياضيات وقد بدأ تطبيقها على الحاسبات الكبيرة أولاً و نذكر منها Ms SQLserver Oracle. بعد ذلك ظهرت عدة نظم علائقية على الحاسبات الشخصية مثل برامج DBAseii و DBAseii و +DBAseiii و DBAseiv و Foxpro Foxbase.

يمكن القول عن هذا النوع من قواعد البيانات أنه تنظيم البيانات في قواعد البيانات العلائقية في جداول ذات بعدين ويمكن اعتبار كل جدول ملف و يستخدم مصطلح ملف مسطح Flat File لأن محتويات الملف مرتبة على محورين فقط.

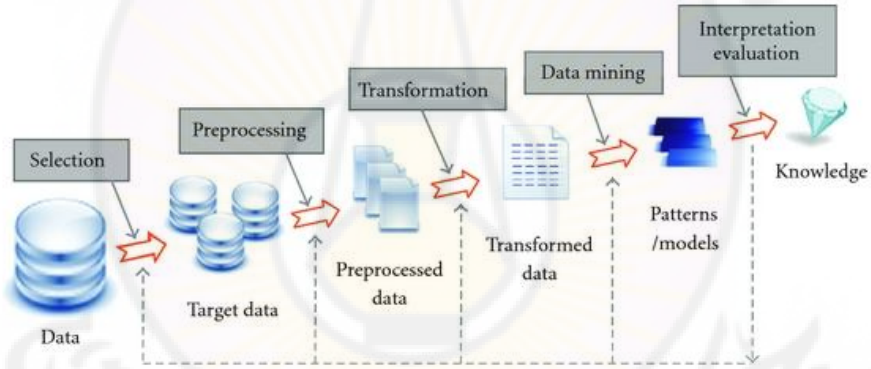
نشأت مجموعة جديدة من المصطلحات تُستخدم في وصف قواعد البيانات العلائقية هذه المصطلحات التي تستخدم في وصف قواعد البيانات الهرمية أو الشبكية ففي النموذج العلائقي يستخدم مصطلح نموذج بيانات علائقية جزئي أو رؤية لبيانات علائقية (Relational Data Submodel or View) بدلاً من المخطط الجزئي Subschema و مصطلح رؤية View مناسب فهو الجزء المستفيد من قاعدة البيانات.



الشكل 15: توضيح قاعدة البيانات العلائقية.

7. اكتشاف المعرفة في قواعد البيانات

مع وجود كميات كبيرة من البيانات المخزنة في قواعد البيانات ومخازن البيانات ازدادت الحاجة إلى تطوير أدوات تمتاز بالقوة لتحليل البيانات واستخراج المعلومات والمعارف منها. من هنا ظهر ما يسمى بالتنقيب في البيانات كتقنية تهدف إلى استخراج المعرفة من كميات هائلة من البيانات (الشكل 16). وهي تقنية حديثة فرضت نفسها بقوة في عصر المعلوماتية، واستخدامها يوفر للشركات والمنظمات في جميع المجالات القدرة على استكشاف والتركيز على أهم المعلومات في قواعد البيانات. كما تركز تقنيات التنقيب على بناء التنبؤات المستقبلية واستكشاف السلوك والاتجاهات وهذا يسمح باتخاذ القرارات الصحيحة في الوقت المناسب.

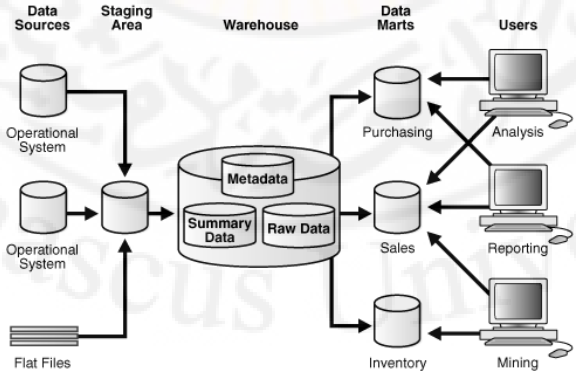


الشكل 16: مخطط يوضح آلية تنفيذ التنقيب عن البيانات.

وتعدّ مرحلة التنقيب بدورها مرحلة من المراحل العملية الأكثر تعقيداً وهي استكشاف المعرفة في قواعد البيانات، والمرتبطة إلى حد بعيد بعملية تطوير أخرى مهمة جداً هي مستودعات البيانات، حيث إن الكثير من الشركات والمنظمات الرائدة اليوم تستخدم عملية استكشاف المعرفة في قواعد البيانات بشكل منهجي ومنظم بوصفها تشكل جوهر العمل الذي يعتمد عليه في تفعيل النشاط و تحقيق الميزة التنافسية.

7.1. مفاهيم التنقيب في البيانات

1. مستودع البيانات (Data Warehousing) هي مجموعة من البيانات، دائمة تاريخية متكاملة للمساعدة في اتخاذ القرارات الإدارية. فهي تساعد على الوصول للبيانات لأغراض التحليلات الزمنية واكتشاف المعرفة واتخاذ القرارات لأنها مصممة خصيصاً لاستخلاص البيانات ومعالجتها وتمثيلها وتقديمها بصورة مناسبة لهذه الأغراض، وتتضمن كمية ضخمة من البيانات قد تكون من مصادر مختلفة، مثلاً عدة قواعد بيانات من عدة نماذج بيانات، وأحياناً من أنظمة ومنصات مختلفة.
2. (On-Line Analytical Processing, OLAP) هي عملية تحليل البيانات المتراكمة في مستودع البيانات.
3. سوق البيانات (The Data Mart) قاعدة بيانات تحتوي على بيانات تصف قطاعاً فقط من عمليات المؤسسة. مثال على ذلك يمكن أن يكون لدى مؤسسة سوق بيانات للتسويق، وسوق بيانات للموارد البشرية.
4. التعدين أو التنقيب عن البيانات (Data Mining) هو مصطلح عادةً ما يُستخدم بالاتصال مع مستودعات البيانات وسوق البيانات ويكون تعدين البيانات عملية إيجاد علاقات في البيانات غير معروفة للمستخدم. ويساعد تعدين البيانات على اتخاذ القرارات المناسبة (عن طريق اكتشاف العلاقات وتقديمها بطريقة مفهومة).



الشكل 17: مخطط يوضح مفاهيم التنقيب عن البيانات.

يتميز عصرنا الراهن (عصر الإنترنت و الاقتصاد الرقمي) بالسيل العظيم و الانتشار الواسع النطاق للبيانات حتى أضحى من المستحيل على المحللين استخلاص معلومات ذات معنى باللجوء فقط إلى المداخل التقليدية للتحليل التمهيدي للبيانات.

مع وجود كميات كبيرة من البيانات المخزنة في قواعد البيانات ومخازن البيانات ازدادت الحاجة إلى تطوير أدوات تمتاز بالقوة لتحليل البيانات واستخراج المعلومات والمعارف منها. من هنا ظهر ما يسمى بالتنقيب في البيانات كتقنية تهدف إلى استخراج المعرفة من كميات هائلة من البيانات.

كما ذكرنا سابقاً تقنية التنقيب في البيانات هي تقنية حديثة فرضت نفسها بقوة في عصر المعلوماتية، واستخدامها يوفر للشركات والمنظمات في جميع المجالات القدرة على استكشاف والتركيز على أهم المعلومات في قواعد البيانات، كما تركز تقنيات التنقيب على بناء التنبؤات المستقبلية واستكشاف السلوك والاتجاهات وهذا يسمح باتخاذ القرارات الصحيحة في الوقت المناسب.



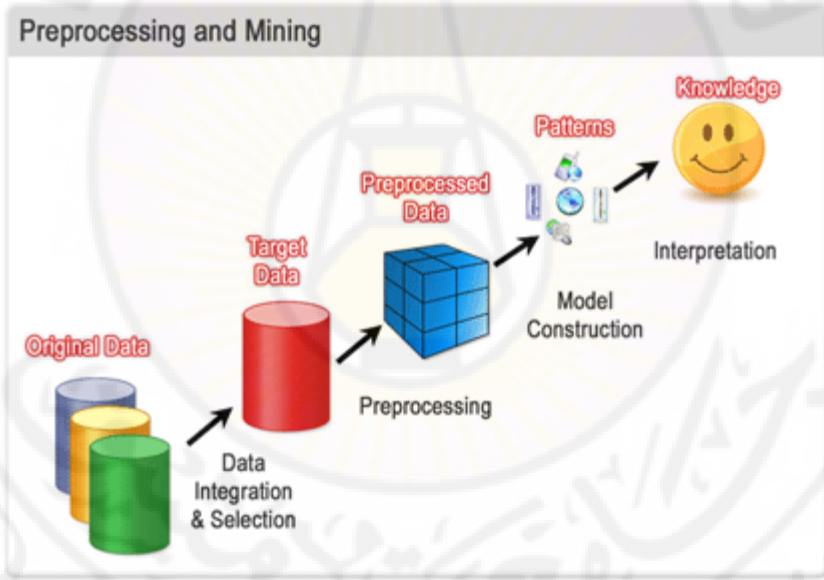
الشكل 18: مخطط يوضح خطوات معالجة التنقيب عن البيانات.

تعدّ عملية التنقيب (الشكل 18) في البيانات بدورها مرحلة من المراحل العملية الأكثر تعقيداً هي استكشاف المعرفة في قواعد البيانات، والمرتبطة إلى حد بعيد بعملية تطوير أخرى مهمة جداً هي مستودعات البيانات، حيث إن الكثير من الشركات والمنظمات الرائدة اليوم تستخدم عملية استكشاف المعرفة في قواعد البيانات بشكل منهجي ومنظم بوصفها تشكل جوهر العمل الذي يعتمد عليه في تفعيل النشاط وتحقيق الميزة التنافسية.

سنتعرف الآن على كيفية التنقيب عن البيانات من خلال الفقرات الآتية:

أولاً: مفهوم التنقيب في البيانات (Data Mining)

ظهر مصطلح التنقيب في البيانات في منتصف التسعينات في الولايات المتحدة الأمريكية، وهو يجمع ما بين الإحصاء و تكنولوجيات الإعلام (قواعد البيانات، الذكاء الاصطناعي، التعلم الآلي). توجد عدة تعاريف لهذا المفهوم منها، "الاستكشاف الآلي أو المؤتمت لأنماط شائقة وغير جلية مخفية في قاعدة بيانات معينة"، أو أنها "عبارة عن تحليلات لكمية كبيرة من البيانات بغرض إيجاد قواعد وأمثلة ونماذج يمكن أن تُستخدم، تقود وتدل أصحاب القرار، وتتنبأ بالسلوك المستقبلي"، كما يمكن تعريفها كذلك بأنها "تحليل لمجموعات كبيرة الحجم من البيانات المشاهدة للبحث عن علاقات محتملة وتلخيص للبيانات في أشكال جديدة لتكون مفهومة ومفيدة لمستخدمها" (الشكل 19).



الشكل 19: مخطط يوضح آلية معالجة التنقيب عن البيانات.

من خلال التعاريف السابقة يمكن القول إن التنقيب في البيانات عبارة عن سيرورة لاستخراج أو اكتشاف معرفة مفيدة وقابلة للاستغلال من خلال مجموعة كبيرة من البيانات، حيث يساعد في استكشاف المعرفة المخفيّة والنماذج غير المتوقّعة، إضافةً إلى استكشاف قواعد جديدة موجودة في قواعد بيانات كبيرة.

ثانياً: العوامل المحركة لثورة التنقيب في البيانات

يمكن تقسيم العوامل المحركة لثورة التنقيب في البيانات إلى قسمين رئيسيين هما:

1. العوامل المتعلقة بجانب العرض

- ✓ تأثيرات تطورات تكنولوجيا المعلومات خاصةً تكنولوجيا تخزين البيانات ومعالجتها وانخفاض تكاليف الاتصالات الإلكترونية وهذا ييسر الوصول إلى قواعد البيانات.
- ✓ ظهور أساليب تحليل جديدة من أبرزها الشبكات العصبونية، نظم الخوارزميات الوراثية، شجرة القرار وقواعد الاستقراء.
- ✓ ثورة إعمار الحاسوب التي تمخضت عن ولادة أجيال جديدة من البرمجيات الصديقة للمستخدم، مثل: مايكروسوفت ويندوز، وبرمجيات العميل / الخادم (Server / Client).
- ✓ التطور الهائل في أساليب التخزين ومن أبرز هذه الأساليب تلك المتعلقة بمستودعات البيانات (Data Warehouse) وأسواقها.

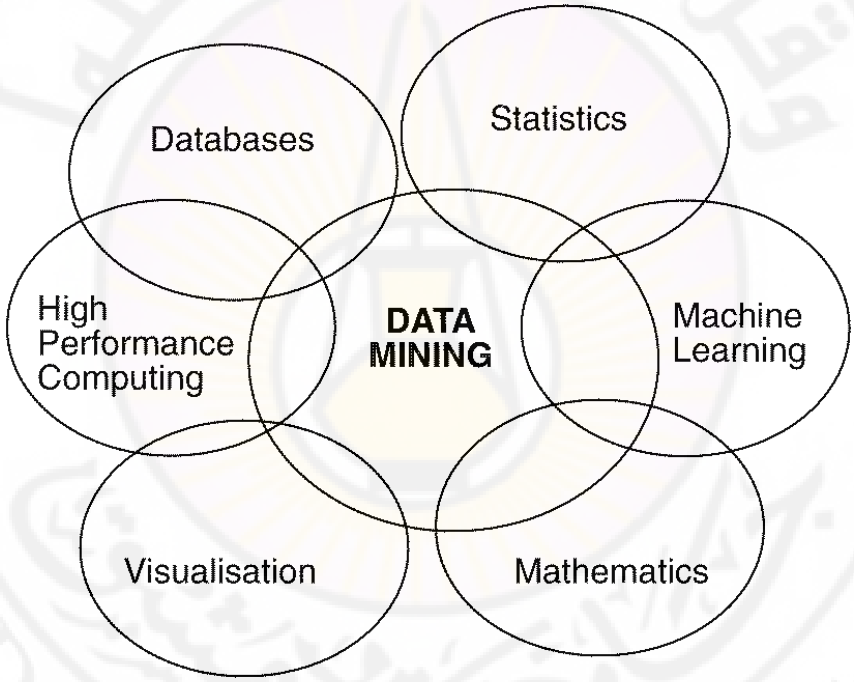
2. العوامل المتعلقة بجانب الطلب

- ✓ تنامي الحاجة إلى نتائج تحليلية سريعة من طرف المؤسسات نتيجة المنافسة الشديدة والتغيرات السريعة في بيئة العمل.
- ✓ انحسار التنظيم الهرمي حيث أصبح لزاماً على المديرين الاعتماد على أنفسهم في الحصول على احتياجاتهم من المعلومات التحليلية، بعدما انحسر دور محللي الدعم الذين كانوا يعتمدون عليهم.
- ✓ ازدهار حركة التأليف والنشر ولاسيما في مجال تكنولوجيا المعلومات والإنترنت وكل ما يرتبط بعصر الاقتصاد الرقمي.

ثالثاً: أدوات التنقيب عن البيانات

نماذج التنقيب في البيانات نوعان: النماذج التنبؤية (Predictives) والنماذج الوصفية (Descriptives).

النماذج التنبؤية تهدف إلى التنبؤ بقيمة بعض الخصائص، مثل التنبؤ باحتمال الشراء للزبون. أما النماذج الوصفية فتنقسم إلى صنفين: نماذج العنقدة التي تسمح بتجميع الأفراد، والأحداث، أو المنتجات في عناقيد، ونماذج الارتباط التي تسمح بتحديد العلاقات بينهم.



الشكل 20: مخطط يشرح ارتباط أدوات التنقيب عن البيانات.

هناك عدة أدوات للتنقيب في البيانات، نذكر أهمها (الشكل 20):

1. التلخيص (Summarization)

يشير التلخيص إلى أساليب تفتيت كتل البيانات الكبيرة إلى مقاييس موجزة، توفر وصفاً عاماً للمتغيرات وعلاقاتها. من الأمثلة على أساليب التلخيص نذكر: المتوسطات،

والمجاميع، والإحصائيات الوصفية التي تتضمن مقاييس النزعة المركزية مثل المتوسط الحسابي والوسيط والمنوال، ومقاييس التشتت مثل الانحراف المعياري. على الرغم من أن مقاييس التلخيص تعطي صورة كبيرة عن بعض التفاصيل ذات العلاقة فإنها غالباً ما تهمل تفاصيل أخرى ذات أهمية كبيرة تتعلق بشكل خاص بسلوك المستهلك.

2. التصنيف (Classification)

يتمثل التصنيف في تفسير أو التنبؤ بخاصية فرد ما من خلال خصائص أخرى. هذه الخاصية هي بصورة عامة كيفية. ويمكن إنجاز التصنيف بالاعتماد على الأساليب الإحصائية القديمة مثل الانحدار والتحليل التمييزي، أو بالاعتماد على أساليب حديثة نسبياً مثل قوى الارتباط والاستنتاج المستند إلى الحالة والشبكات العصبونية. كأمثلة عن طرق التصنيف تلك المستعملة كجزء من تطبيقات استكشاف المعرفة التي تتضمن تصنيف اتجاهات الأسواق المالية، والتحديد الآلي للأشياء المهمة في صورة كبيرة من قواعد البيانات.

3. التنبؤ (Prediction)

يشبه التنبؤ التصنيف أو التقدير، إلا أن البيانات تُصنَف على أساس التنبؤ بسلوكها المستقبلي أو تقدير قيمتها المستقبلية. حيث إن المتغير التابع المُتنبأ به هو متغير كمي. ومن الأدوات التقليدية المُستخدمة في التنبؤ نذكر على سبيل المثال: الانحدارات بأنواعها والتحليل التمييزي. أما الأساليب الجديدة فتشتمل على قواعد الارتباط وشجرة القرار والشبكات العصبونية والخوارزميات الوراثية.

4. العنقدة أو التجزئة (Clustering)

يتمثل التجميع العنقودي أو التجزئة إلى قطاعات في البحث عن مجموعات متجانسة في مجتمع من الأفراد. يشير التجميع العنقودي أو التجزئة إلى قطاعات إلى عملية تشكيل مجموعات أو قطاعات مؤلفة من أفراد أو أصحاب أسر، و ذلك بالاستناد إلى معلومات متضمنة في مجاميع من المتغيرات التي تصفهم. الغرض من التجميع العنقودي المساعدة على تطوير برامج تسويقية مصممة على مقاسات الزبائن أنفسهم،

التي بالإمكان استخدامها لاستهداف أعضاء لكل قطاع من هذه القطاعات على أمل ترغيبهم في تكرار الشراء أو التحول إلى زبائن موالين. تتم أساليب التجميع العنقودي غالباً بمساعدة أساليب التحليل العنقودي الإحصائية والأساليب المستندة إلى شجرة القرار، والشبكات العصبونية والخوارزميات الوراثية.

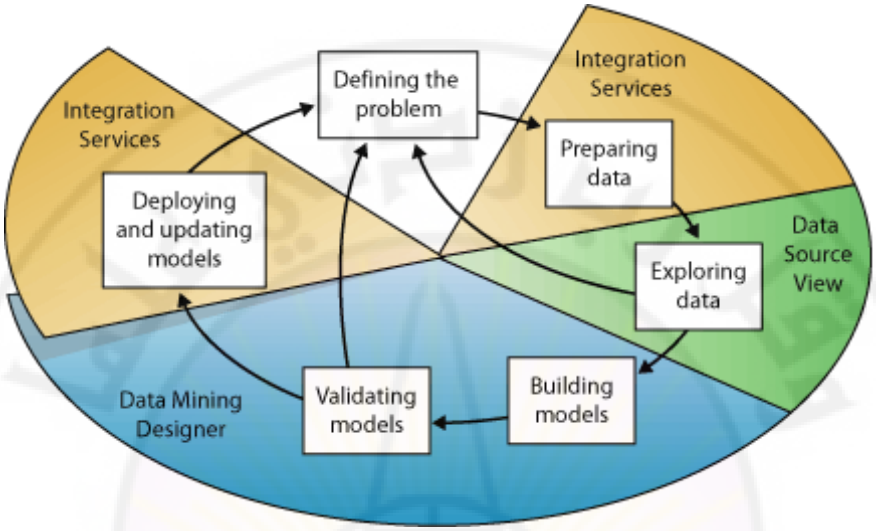
5. تحليل الارتباط (Rule Analysis)

يتمثل الارتباط في البحث عن علاقات أو ارتباطات موجودة بين عدة خصائص. يشير تحليل الارتباط إلى مجموعة من الأساليب التي تُستخدم في ربط أنماط الشراء عبر القطاعات المتقاطعة أو عبر الوقت. فمثلاً يقوم أسلوب تحليل سلة السوق (نوع من أنواع الارتباط) باستخدام المعلومات الكامنة في السلع التي اشتراها المستهلكون فعلياً للتنبؤ بالسلع المحتمل أن يقوموا بشرائها إذا ما تم تقديم عروض خاصة لهم أو إذا ما تم تعريفهم بهذه السلع.

6. الكشف عن التغيرات أو الانحرافات (Change and Deviation Detection)

يرتكز على استكشاف التغيرات المهمة جداً في البيانات من خلال قياسات سابقة أو قيم معيارية.

رابعاً: مراحل عملية التنقيب في البيانات



الشكل 21: مخطط يشرح مراحل التنقيب عن البيانات.

يمكن تلخيص مراحل وخطوات عملية التنقيب في البيانات كما يأتي (الشكل 21):

1. فهم طبيعة الأعمال (Business Understanding): يعدّ فهم المشاكل والمسائل التي تواجهها الأعمال المطلب الأول لاكتشاف المعرفة. بمعنى آخر، كيف يمكن تحقيق المنفعة الأعظم من التنقيب في البيانات، وهذا يتطلب وجود صيغة واضحة ومحددة لأهداف الأعمال.
2. فهم البيانات (Data Understanding): تعدّ مسالة معرفة ماهية وطبيعة البيانات عاملاً مهماً في نجاح عملية التنقيب في البيانات واكتشاف المعرفة، حيث إن معرفة البيانات بصورة جيدة تعني مساعدة المصممين على استخدام الخوارزميات أو الأدوات المستخدمة للمسائل المحددة بدقة عالية. هذا يقود إلى تعظيم فرص النجاح بالإضافة إلى رفع الفعالية والكفاءة لنظام اكتشاف المعرفة. يمكن تلخيص الخطوات الضرورية لعملية فهم البيانات كالآتي:

2.1. تجميع البيانات (Data Collection): و هي الخطوة الموجهة نحو تحديد مصدر البيانات قيد الدراسة بما في ذلك استخدام البيانات العامة الخارجية مثل الضرائب وغيرها.

2.2. توصيف البيانات (Data Description): وهي الخطوة التي تركز على توصيف محتويات الملف الواحد من الملفات أو الجداول.

2.3. جودة البيانات وتحقيقها (Data Quality and Verification): هذه الخطوة تحدد ما إذا كان تقليل أو إهمال بعض البيانات غير الضرورية أو كونها رديئة الجودة وقد لا تنفع في الدراسة. لأن النموذج الجيد يحتاج إلى بيانات جيدة وهذا يتوجب أن تكون البيانات صحيحة وذات مضمون دقيق.

2.4. التحليل الاسترشادي للبيانات (Exploratory Analysis of Data): في هذه المرحلة يتم استخدام أساليب مثل الإظهار المرئي أو التصور أو عملية التحليل المباشر (OLAP) التي تؤدي إلى إجراء التحليل الأولي للبيانات. وتعدّ هذه الخطوة مهمة وضرورية لأنها تركز على تطوير الفرضيات المتعلقة بالمشكلة قيد الدراسة.

2.5. تهيئة البيانات (Data Preparation): و تشمل الخطوات الآتية:

2.5.1. الاختيار (Selection) و تعني اختيار المتغيرات المتوقعة وحجم العينة.

2.5.2. صياغة المتغيرات وتحويلها (Construction and Transformation Variables) حيث يجب دائما أن تُصاغ المتغيرات الجديدة لبناء النماذج الفعالة.

2.5.3. تكامل البيانات (Data Integration): مجاميع البيانات في دراسة التنقيب في البيانات من الممكن خزنها في قواعد بيانات متعددة الأغراض. قد نحتاج إلى توحيد (أو دمج) تلك البيانات في قاعدة بيانات مشتركة.

2.5.4. تنسيق البيانات (Data Formatting) حيث تتعلق هذه الخطوة في إعادة ترتيب حقول البيانات كما يتطلب في نموذج التنقيب في البيانات.

2.6. صياغة نماذج الحل وثبوتها (Model Building and Validation): إن بناء وصياغة نموذج الحل السليم والدقيق يتم من خلال عملية الخطأ والصواب، حيث كثيراً ما تحتاج مثل هذه العملية إلى مساعدة المختصين في التنقيب عن البيانات بهدف اختبار وفحص مختلف البدائل للحصول على أفضل نموذج لحل المشكلة قيد الدراسة.

2.7. التقييم وتعليل نتائج النموذج (Evaluation and Interpretation): عندما يتم صياغة النموذج والتحقق من ثباته وصدقه، تجري مباشرة عملية التحقق من ثبات حزمة البيانات التي يتم تغذيتها بوساطة النموذج. ولما كانت نتاجات هذه البيانات معروفة، فإن النتائج المتوقعة تُقارن مع النتائج الفعلية في ثبات حزمة البيانات قيد التشغيل. تؤدي هذه المقارنة أو المفاضلة إلى التحقق من دقة النموذج.

2.8. نشر وتوزيع النموذج (Model Deployment): تشتمل هذه الخطوة على نشر وتوزيع النموذج داخل المنظمة لمساعدة عملية صنع القرار. النموذج الصالح يجب أيضاً أن يحقق الرضا لدى المستفيدين طالما أن اختيار النموذج لابد أن يتم من خلال الدراسة الاسترشادية أو من خلال نموذج مصغر من الدراسة الشاملة.

لا تحتاج عملية التنقيب في البيانات إلى تجميع البيانات في مستودع البيانات. أما إذا كان مستودع البيانات موجود في المنظمة، فمن الأفضل عدم احتكار المستودع بشكل مباشر لغرض التنقيب في البيانات.



الشكل 22: صورة تعبيرية توضح مفهوم التنقيب عن البيانات.

خامساً: مجالات تطبيق التنقيب في قواعد البيانات في منظمات الأعمال

لقد عرف التنقيب في البيانات نجاحاً كبيراً وازدادت تطبيقاته في منظمات الأعمال و ذاك يرجع إلى النمو الكبير في البيانات ولا سيما الموجودة في قواعد البيانات ومخازن البيانات، والمنافسة الشديدة في السوق، وهذا يدفع هذه المنظمات إلى الاستفادة القصوى من البيانات التي تمتلكها باستخدام التنقيب في البيانات. فبدأ استخدام التنقيب في البيانات في منظمات التوزيع، ثم انتقلت بعدها سريعاً إلى البنوك، ومؤسسات التأمين، وشركات الهاتف (ولاسيما الهاتف النقال)، ومؤسسات المياه والكهرباء، ومؤخراً مؤسسات النقل الجوي والنقل بالسكك الحديدية..الخ. وكانت تطبيقاتها الأولى في ميدان إدارة العلاقة مع الزبون من خلال تحليل سلوك الزبائن من أجل الوصول إلى اقتراح منتجات وفق رغباتهم.

بشكل عام، هناك عدة ميادين لتطبيق التنقيب في البيانات في منظمات الأعمال نذكر منها:

1. التسويق: استخدمت الشبكات العصبونية الاصطناعية في دراسات التسويق المستهدف ولاسيما الحصص السوقية. ساعدت هذه الأساليب في التسويق على استخدام نهج تحصيص الزبائن وفقاً إلى الحقائق الديمغرافية (السكانية) الأساسية مثل الجنس والعمر والمجموعات وكذلك أنماطهم الشرائية.
2. التجزئة: لقد استُخدمت أساليب التنقيب في البيانات بصورة فعالة في التنبؤ بالمبيعات حيث أخذت العديد من المتغيرات في الدراسات مثل متغيرات السوق

المتعددة، قدرات الزبائن المستندة على العادات المتبعة في الشراء. كما ساعدت عدة أساليب، مثل: تحليل السلة الشرائية أو السلة السوقية كثيراً على إيجاد أي من المنتجات التي يمكن أن تُشترى سوية من قبل الزبائن.

3. البنوك: لقد أثبتت تنبؤات الأعمال والمالية على أنها الأساليب الممتازة في تطبيقات أساليب التنقيب في البيانات. وقد استخدمت هذه الأساليب في إيجاد الأسعار المضمونة وتنبؤات الأسعار المستقبلية وأداء الأسهم. قد حققت استخدامات مثل هذه الأساليب النجاح في تطوير أنظمة القياس الرقمية في تحديد مخاطرة القروض والاحتيالات المالية أيضاً.
4. التأمين: لقد استُخدمت أساليب التنقيب في البيانات كذلك و بصورة واسعة في مجال شركات التأمين ولاسيما في تحصيص مجاميع الزبائن بهدف تحديد أسعار البوليصة وذبذبات المطالبات المتوقعة مستقبلاً، وتحديد المطالبات المزيفة.
5. الاتصالات: استُخدمت بصورة فعالة أساليب التنقيب في البيانات مثل الشبكات العصبونية في محاولة تقليل "الخضخضة" التي تظهر عندما تفقد المنظمة هؤلاء الزبائن خلال المنافسة وذهابهم إلى المنافسين الآخرين.
6. إدارة العمليات: استخدمت الشبكات العصبونية في عمليات التخطيط والجدولة وإدارة المشاريع بالإضافة إلى إدارة الجودة.

لقد أدى إدماج مزايا تكنولوجيا المعلومات مع الطرق الإحصائية والخوارزميات إلى توافر الإمكانات اللازمة للتنبؤ بالسلوك المستقبلي ثم وضع الحلول المناسبة للمشكلات قبل وقوعها في حال إمكان حدوثها، أو من باب التنبؤ بهدف التطوير والتحديث بشكل عام في شتى المجالات. يتم تحقيق كل ذلك باستخدام تقنيات التنقيب في البيانات التي تشكل مرحلة من مسار أشمل هو استكشاف المعرفة في قواعد البيانات التي أصبحت من الهموم الكبيرة التي تقع على عاتق الدول بمؤسساتها كافة.

أسئلة البحث:

1. ماهي البيانات؟
2. ماذا نعني بالمعلومات؟
3. ماهو الاختلاف بين المعلومات والبيانات؟
4. من يحتاج إلى قاعدة البيانات؟ لماذا نحتاج إلى نظام قاعدة البيانات؟
5. عرف قاعدة البيانات، نظام قاعدة البيانات.
6. ماهي أنواع البيانات المخزونة في قاعدة البيانات؟
7. ماهي أهمية قاعدة البيانات؟
8. ماهي ميزات قواعد البيانات؟
9. ما المقصود ببيئة قاعدة البيانات (Database Environment)؟
10. كيف يتم تصنيف بيئات قاعدة البيانات؟
11. اشرح البيئة المركزية Environment Mainframe لقاعدة البيانات.
12. اشرح بيئة العميل /الخادم Client/Server Environment لقاعدة البيانات.
13. اشرح بيئة الحوسبة الإنترنت (Internet Computing Environment).
14. اذكر أهم ما يتميز به نظام قاعدة البيانات؟
15. مم يتكون نظام قاعدة البيانات؟
16. عدد أنواع المستخدمين لقاعدة البيانات Data Users.
17. عرف لغات نظام إدارة قواعد البيانات . DBMS Languages
18. تكلم عن نظم إدارة قواعد البيانات الهرمية Hierarchical DBMS
19. تكلم عن نظم إدارة قواعد البيانات الشبكية Network DBMS
20. عرف المصطلحات الآتية (مستودع البياناتData Warehousing – سوق البيانات The Data Mart – التعدين أو التنقيب عن البيانات Data Mining)
21. اشرح الخطوات الضرورية لعملية فهم البيانات.
22. تكلم على ميادين لتطبيق التنقيب في البيانات.

الفصل الثاني

تصميم قاعدة البيانات

1. أهمية تصميم قاعدة البيانات

إن عملية بناء قاعدة بيانات جيدة ليست بالعملية البسيطة، إذ لا بد من بذل مجهود كبير للحصول على قاعدة بيانات جيدة. فالتصميم الجيد لقاعدة البيانات يسهل عملية استخدام وإدارة تلك القاعدة. أما التصميم السيء فسيؤدي إلى تكرار البيانات (وجود البيانات نفسها في أكثر من مكان)، وبالتالي يصعب عملية الحفاظ على توافقية البيانات ويؤدي غالباً تكرار البيانات إلى نتائج غير صحيحة عند طلب المعلومات، وهذا بدوره يؤدي إلى أن القرارات ستكون خاطئة وبالتالي التخطيط المستقبلي سيكون خاطئاً لاعتماده على بيانات مخزنة بطريقة غير صحيحة في قاعدة بيانات صُممت بطريقة سيئة لتعطي معلومات خاطئة.

2. دورة حياة نظام قاعدة البيانات

2.1. الدراسة المبدئية للنظام القائم

تشمل الدراسة المبدئية للنظام القائم المراحل الآتية:

1. تحليل الوضع الحالي ومعرفة طبيعة العمل والإجراءات المستخدمة وقواعد العمل.
2. تحديد القيود والمشاكل التي تواجه النظام القائم بما في ذلك القيود المادية (العامل البشري – الأنظمة والقوانين – التمويل).
3. تحديد المتطلبات الواجب تحقيقها والمزايا المطلوب إبرازها في النظام المطلوب.

2.2. تصميم قاعدة البيانات

تعدّ هذه المرحلة من أهم المراحل في دورة حياة النظام فلا بد من بذل جهد كبير لتصميم النظام. وسندرس في هذا الفصل المبادئ الأساسية لتصميم قاعدة البيانات من خلال مخطط الكينونات العلائقية.

2.3. هيكل (مخطط) قاعدة البيانات

(Database Schema)

هو مخطط يوصف قاعدة البيانات حيث يشمل وصف لبنية البيانات فيها، وشروط تلك البيانات أو القيود التي يجب أن تُطبق عليها. يمكن عدّ مخطط البيانات دليلاً أو قاموساً يصف تلك البيانات، بغض النظر عن البيانات نفسها.

إذا ما تم عمل التصميم المنطقي لقواعد البيانات فإنه من السهل وضع التصميم الفيزيائي لقواعد البيانات تلك، وبالتالي فإن الخطوات الأساسية التي ممكن اتباعها لتصميم قاعدة بيانات هي:

1. جمع البيانات ومتطلبات المستخدمين

(Requirement Gathering)

2. تحليل البيانات

(Data Analysis)

3. إنشاء التصميم المفاهيمي لقواعد البيانات

(Conceptual Model)

4. إنشاء التصميم المنطقي لقواعد البيانات

(Logical Model)

5. إنشاء التصميم الفيزيائي لقواعد البيانات

(Physical Model)

إن عملية جمع البيانات وتحديد المتطلبات للمستخدمين هي خطوة مهمة جداً وذلك لأنها تحدد الاحتياجات المختلفة للمستخدمين من قاعدة البيانات وتعطي وصفاً للنظام القائم داخل البيئة التي تتم عليها الدراسة. هنا لا بد من ذكر أن هذه العملية يجب أن تكون عملية تفصيلية وشاملة لأن أي نقص في المعلومات يؤدي إلى تصميم خاطئ أو خلل في التصميم وهذا يؤدي إلى مشاكل في قواعد البيانات. بناءً على هذه المعلومات

يتم بناء مخطط قواعد البيانات الذي يمثل المعلومات التي تم تجميعها في المرحلة السابقة وهذه المعلومات هي وصف للبيانات المختلفة وعلاقتها مع بعضها ببعض وكذلك تحديد القيود والمحددات المفروضة على هذه البيانات. إن هذه المرحلة تسهل التفاعل بين المستخدمين وبين المصمم حيث يتم عمل مخططات سهلة الفهم وتعكس الواقع قيد الدراسة.

بعد هذه المرحلة يتم التحول إلى تصميم منطقي لقواعد البيانات باستخدام إحدى نظم إدارة قواعد البيانات.

3. نماذج قاعدة البيانات : النموذج العلائقي The Relational Model

إنك تشارك العديد من العلاقات مع أفراد عائلتك .فعلى سبيل المثال، في العائلة الواحدة سيكون لكل منا شكل صلة وعلاقة مع الأم. فأنت ووالدتك ذات صلة (علاقة)، فلديك أم واحدة فقط، لكنها قد تكون لها عدة أطفال. أنت وإخوتك ذات صلة قد يكون لديك العديد من الإخوة والأخوات، وبطبيعة الحال، سيكون لديهم الكثير من الإخوة والأخوات أيضاً. ومن هنا تفهم العلاقة على أنها وصف دقيق للحالة التشاركية مع الكائن الآخر.

يعدّ النموذج العلائقي لقواعد البيانات The Relational Model بأنه الدعامة الأساسية لقواعد البيانات المعاصرة. في هذا الفصل سنعرض الأفكار الأساسية لهذا النموذج، والتي تركز على تنظيم البيانات و إمكانية استرجاعها.

يتكون النموذج العلائقي من :

- مجموعة من هياكل البيانات تُعرف بالجداول .
- مجموعة من الأساليب لبناء الجداول بدءاً من المخطط الأولي للجداول و نشير هنا إلى عمليات الجبر العلائقية.
- مجموعة من القيود المفروضة على البيانات الواردة في الجداول.

تعدّ قواعد البيانات العلائقية من أهم قواعد البيانات وتتصف بأنها قاعدة بيانات يستقبلها المستخدمون على هيئة جداول وليس شيئاً آخر سوى الجداول.

3.1. خواص الجداول في قواعد البيانات العلائقية

1. الأعمدة تُناظر الحقول. كما أن كل القيم المدرجة تحت عمود واحد لها النوع نفسه Data Type.
2. ترتيب الأعمدة في الجدول ليس ذا أهمية .
3. كل عمود له اسم يختلف عن بقية الأعمدة في الجدول نفسه.
4. السطر يعادل السجل. كل سطر في الجدول يختلف عن بقية الأسطر. أي إن كل سطر متفرد في الجدول.
5. ترتيب الصفوف في الجدول ليس ذا أهمية.
6. لكل جدول مسمى وحيد.
7. يوجد لكل جدول حقل يسمى المفتاح الأساسي يمكننا من الوصول لسجل معين في هذا الجدول.
8. كل القيم مُعبر عنها صراحة أي إنها قيم صريحة و ليست متغيرات.
9. الخلية الواحدة تحتوي على قيمة واحده فقط.

على سبيل المثال، جدول البرنامج الأسبوعي لكلية صغيرة قد تبدو كما يأتي:

Time	RoomN°	CN°	DoW
5:00 P.M.	84	Css110	Mon
7:00 P.M.	62	Cs450	Mon
10:00 A.M.	65	Cs110	Wed
12:00 P.M.	63	Cs310	Wed
2:00 P.M.	63	Cs210	Thu
3:00 P.M.	65	Cs450	Thu
5:00 P.M.	84	Cs240	Thu

الجدول 1: الجدول الأسبوعي للمقررات.

عندما نُفكر في الجدول نميز ثلاثة مكونات رئيسية هي : اسم الجدول، في حالتنا (Schedule)، بالإضافة إلى رأس الجدول الذي يحتوي على أعمدة مفردة (أي إدخال واحد لكل عامود). في مثالنا الأعمدة هي DoW، CN^o، $RoomN^o$، Time. ومحتوى الجدول هو البيانات الموجودة في الصفوف السبعة. تماشياً مع الممارسة العملية لقواعد البيانات فإن أسماء الأعمدة ندعوها بـ (Attributes) وتعرف بخاصيات أو سمات.

إذا كان اسم الجدول H فإن رأس الجدول يتألف من مجموعة الأعمدة ذي السمات (Attributes) A1,…,An عندئذ نكتب H كسلسة بدلاً من مجموعة

.H= A1… An

كل سمة A (Attribute) لديها مجموعة خاصة تتعلق بها ندعوها نطاق Domain لـ A. بلغة أبسط نوع البيانات التي يمكن أن تظهر في عمود معين يسمى مجال (Domain) ونكتب Dom(A) يعني المجال A و يضم هذا المجال مجموعة القيم التي يمكن أن تُحفظ في العمود A. على سبيل المثال في الجدول Schedule فإن مجال السمة DoW "يوم من أيام الأسبوع " هو مجموعة مكونة من السلاسل 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'.

نحن بحاجة إلى اسم لكل جدول، حتى نتمكن من الرجوع إليه. من حيث المبدأ، و الأسماء هي ببساطة سلاسل مكونة من الرموز المختارة من أبجدية ثابتة –الرومانية الحديثة– تُضاف مع بعض الرموز الخاصة " _" ، " . " ، " U " ، " [" ، "] " ، .. الخ.

تنطبق هذه القاعدة على أسماء الصفات في معظم قواعد البيانات. هذه الأسماء هي قضية حساسة. وفي تقدم من وقت لآخر حسب الحاجة. مع ذلك، يوجد دائماً فكرة جيدة لاختيار الأسماء التي هي ذات مغزى للقارئ ومدلول للوظيفة المطلوبة من الجدول أو السمة.

مما سبق نجد أنه يمكن وصف النموذج العلائقي في قاعدة البيانات على شكل مجموعة من العلاقات المرتبطة بعضها ببعض. فكل علاقة يمكن عدّها جدولاً من القيم. على هذا

الأساس يحوي كل صف من الجدول على بيانات مترابطة. في النموذج العلائقي كل صف يمثل كائناً أو علاقة من العالم الحقيقي. اسم العلاقة أو الجدول يستخدم لفهم معنى البيانات في الصف.

يمكن تلخيص المصطلحات الأساسية في النموذج العلائقي بالآتي:

الجزء المركزي في النموذج العلائقي هو العلاقة (Relation). المثيل (Instance) هو محتويات الجدول في وقت محدد من الزمن. المخطط (Schema) هو ترويسة الأعمدة في الجدول. المخطط يُحدد باسم العلاقة وأسماء الحقول (أو الخصائص أو الأعمدة) و المجال لكل حقل.

نوع البيانات التي يمكن أن تظهر في عمود معين يسمى مجال (Domain). المجال D هو مجموعة من القيم الأولية (Atomic). نعني بالأولية أي إنها غير قابلة للتجزئة. الطريقة الشائعة في تحديد المجال هي تحديد نوع البيانات التي تكون المجال. من المفيد أن نحدد اسماً للمجال ليساعد في فهم معاني القيم المستخدمة. من الأمثلة على المجالات نذكر:

رقم الهاتف: مجموعة الأرقام المكونة من سبع منازل و تمثل رقم هاتف مستخدم.

رقم الهوية: رقم من 11 خانة.

الأسماء: مجموعة الأسماء المستخدمة للأشخاص.

يمكننا توضيح المعنى من خلال المخطط الآتي الذي يصف قاعدة بيانات لجامعة:

Student (Sid: String, Name: String, Login: String, Age: Integer, Gpa: Real)

هذا المخطط يدلنا على سبيل المثال أن الحقل المسمى Sid مجاله هو سلسلة محارف (String) أي سلسلة من الأحرف و بالتالي القيم المقبولة في هذا الحقل هي سلاسل من المحارف (أحرف و أرقام). جسد العلاقة هو مجموعة من المرتبات (Tuple). المرتب هو عبارة عن الصف في الجدول أو السجل وكل السجلات في جسد العلاقة له عدد الحقول الموصوفة نفسها في المخطط. عنوان العمود يسمى خاصية أو Attribute.

3.2. أهم مفاهيم قواعد البيانات العلائقية

3.2.1. المفتاح الأساسي للجدول: Primary Key

الجدول يتألف من صفوف (أو سجلات Records)، و في قواعد البيانات العلائقية يجب ألا يتكرر الصف كاملاً بل يجب أن توجد قيمة فريدة تميز كل صف من الصف الآخر. هذا ما نسميه بالمفتاح الأساسي للجدول. كمثال على ذلك في جدول المستعيرين يمكن أن يكون لدينا أشخاص لهما الإسم نفسه وتاريخ الميلاد نفسه وكل البيانات عنهما متماثلة لذلك نميز بينها عن طريق رقم الهويه مثلاً ولكن هذا التمييز صعب بعض الشيء وأحياناً غير ممكن. لذلك سهلت لنا قواعد البيانات ميزة الترقيم التلقائي بحيث عندما نضيف سجلاً جديداً فإنه يأخذ رقماً خاصاً به تلقائياً ودون أن ندرج له رقماً في عمود المفتاح الأساسي و بالتالي نميز بين كل صف في الجدول عن طريق رقمه الذي يميزه.

3.2.2. المفتاح الغريب(الأجنبي) Foreign Key

في قواعد البيانات هناك مصطلح اسمه التكامل المرجعي Referential Integrity أي إنه لو فرضنا أنه يُوجد لدينا جدولان مثل Addresses، Employees يتم الربط بينهما عن طريق وضع عمود في أحدهما، هذا العمود له اسم عمود المفتاح الأساسي نفسه في الجدول الآخر، وله نوع البيانات نفسها Data Type والصفات نفسها Attributes. هذا العمود نسميه (مفتاح غريب) بحيث يصبح هذا العمود هو صلة الوصل بين الجدول الأول والجدول الثاني. هذه العملية نسميها بالتكامل المرجعي.

إذاً كتعريف نهائي للتكامل الرجعي: التكامل المرجعي هو علاقة بين جدولين يحوي الجدول الأول على مفتاح أساسي والجدول الثاني يرتبط مع الجدول الأول بمفتاح غريب له اسم المفتاح الأساسي نفسه وبياناته نفسها وصفاته نفسها. أما المفتاح الغريب فهو عمود في جدول يشير إلى عمود في جدول آخر هو المفتاح الأساسي للجدول الأخير و يكون لكل من العمودين الأساسي والغريب الاسم نفسه ونوع البيانات المدخلة فيهما والصفات نفسها. فعند إضافة بند إلى الجدول ذي المفتاح الغريب وليس موجوداً

في الجدول ذي المفتاح الأساسي فسيقوم ملقم قواعد البيانات بتنبيهي إلى أنني أدرج شيئاً غير موجود في الجدول الأساسي.

من الأشياء المفيدة التي يقدمها لنا التكامل المرجعي هي عملية الحذف المتسلسل Deletes Cascading. ففرضاً أن لديك موظفاً مسجلاً في الجدول Employees و لدى هذا الموظف ثلاثة عناوين مسجلة في الجدول Addresses، ثم قدم هذا الموظف طلب استقالة ووافقت أنت عليه عندها ستقوم بحذف سجله من قاعدة البيانات الخاصة بشركتك. تخيل مدى الصعوبة والإرباك الذي ستتعرض له عند كل عملية حذف و ولاسيما مع الزبائن أو مع الأشخاص الذين يسجلون على بريد إلكتروني مجاني ثم لا يعودوا إلى البريد والكثير من العمليات المشابهة. وهنا تبرز أهمية الحذف المتسلسل فعندما أطلب من ملقم قواعد البيانات أن يحذف سجل هذا الموظف من الجدول Employees فإنه سيقوم آلياً بحذف جميع السجلات المقابلة في الجدول الذي يحتوي على مفتاح غريب.

3.2.3. القيمة Null

في قواعد البيانات العلائقية سنتعرض لكثير من القيمة Null وهي تعني اللاقيمة (أو عديم القيمة) أي لا شيء وهي بالطبع تختلف عن الصفر Zero .

3.2.4. الفهارس Indexes

الفهرس في قواعد البيانات كالفهرس المعروف للكتب ويؤدي وظيفته نفسها. ففي الكتاب يساعدنا الفهرس على سرعة الانتقال إلى موضوع معين في الكتاب. كذلك الأمر في قواعد البيانات، فالفهارس تساعدنا على سرعة الانتقال و الحصول على معلومات لاستعلام ما بسرعة أكبر.

3.3. أنواع العلاقات

3.3.1. علاقة واحد لواحد

One-to-One Relationships

في علاقة واحد لواحد كل سجل من الجدول الأول يرتبط بسجل واحد فقط من الجدول الثاني وكل سجل من الجدول الثاني يرتبط بسجل واحد فقط من الجدول الأول أي إن كلا الجدولين يمكن أن يكون سجل واحد فقط على جانبي العلاقة كما في الشكل 1. نلاحظ من هذه القاعدة أن قيمة المفتاح الأساسي في كل سجل من الجدولين هي نفسها، في غياب مثل هذه القاعدة، يمكن الجمع بين كلا الجدولين في جدول واحد دون خرق قواعد التنظيم (سنقوم بدراسة مفهوم التنظيم في الفصل الرابع من هذا الكتاب).



One to One

الشكل 1 علاقة رأس برأس أو علاقة واحد إلى واحد

في علاقة واحد إلى واحد بين الجدول A و الجدول B عدد الصفوف في الجدول A يجب أن يساوي عدد الصفوف في الجدول B.

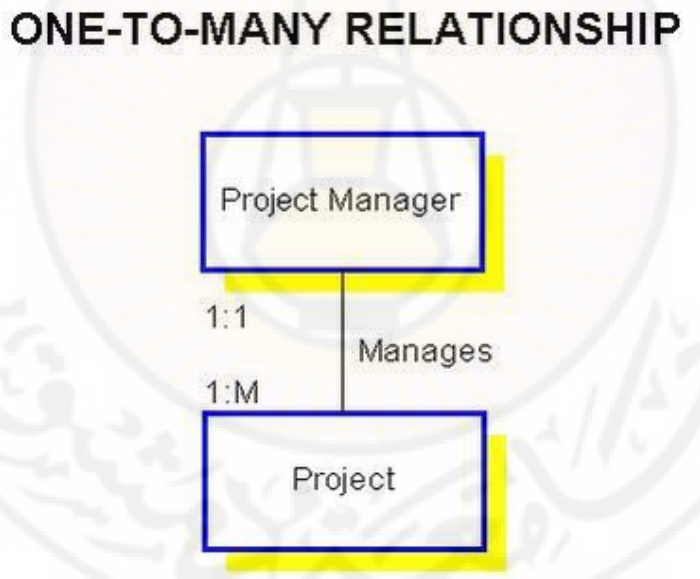
سيكون من الواضح أن العلاقات واحد إلى واحد ليست مفيدة جداً عند مصمم قاعدة البيانات، ببساطة يمكن دمج كلا الجدولين في جدول واحد. مع ذلك، هناك بعض الحالات التي تساعد فيها علاقة واحد إلى واحد في تحسين الأداء. على سبيل المثال: إذا كان جدول قاعدة بيانات يحتوي على عدد قليل من أعمدة البيانات التي يكثر استخدامها والأعمدة المتبقية التي يتم استخدامها بشكل غير منتظم، ومصمم قاعدة البيانات قد يقسم جدولاً واحداً إلى جدولين مرتبطين من خلال علاقة واحد إلى واحد. مثل هذا التصميم يقلل من النفقات العامة اللازمة لاسترداد الأعمدة المستخدمة نادراً عندما يتم تنفيذ الاستعلام على محتويات جدول قاعدة البيانات. مثال على ذلك البطاقة الشخصية تحتوي البيانات التعريفية بالاضافة الى الصورة الشخصية والبصمة، في هذه

الحالة يتم إنشاء جدولين الأول يحوي المعلومات التعريفية والجدول الثاني يحوي الصورة والبصمة ويرتبط بعضهما ببعض بعلاقة واحد إلى واحد .

3.3.2. علاقة واحد لمتعدد

One-to-Many Relationships

كل سجل من الجدول الأول يرتبط بمجموعة سجلات من الجدول الثاني وكل سجل من الجدول الثاني يرتبط بسجل واحد فقط من الجدول الأول كما في الشكل 2. هذه العلاقة هي مماثلة لتلك التي بينك و بين أحد الوالدين. لديك أم واحدة فقط، ولكن والدتك قد يكون لديها عدة أولاد. أو كما في حال جدول الدول مع جدول المدن كل دولة لها مجموعة مدن وكل مدينة لها دولة واحدة فقط .



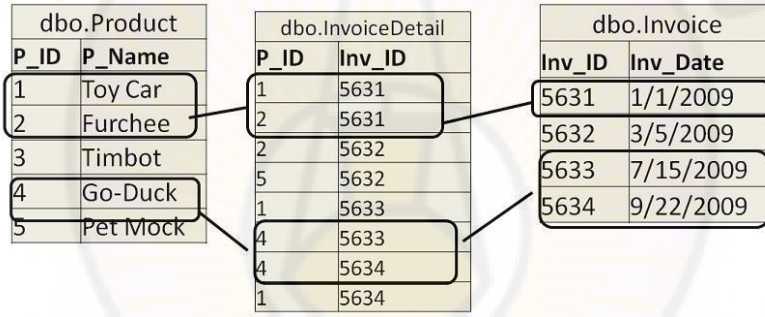
الشكل 2: علاقة رأس بأطراف.

3.3.3. علاقة متعدد لمتعدد

Many-to-Many Relationships

كل سجل من الجدول الأول يرتبط بمجموعة سجلات من الجدول الثاني وكل سجل من الجدول الثاني يرتبط بمجموعة سجلات من الجدول الأول كما في الشكل 3. مثال على ذلك جدول الكتب وجدول المؤلفين فكل كتاب قد يكون له مجموعة مؤلفين و كل مؤلف له مجموعة كتب .

إن تمثيل علاقة متعدد لمتعدد بين جدولين تتم عبر كسر العلاقة بجدول ثالث يحوي المفاتيح الأساسية من الجدولين فقط ويسمى الجدول الوسيط، في حال المثال السابق جدول الكتب وجدول المؤلفين ننشئ جدولاً ثالثاً يربط بينهم وليكن جدول كتب المؤلفين.



Many-to-Many Relationships

الشكل 3: علاقة أطراف بأطراف.

4. المفاهيم الأساسية في نموذج الكينونات العلائقية

(Principle of Entity Relationship Model, ER Model)

إن نموذج الكينونات العلائقية هو وصف للبيانات في البيئة قيد الدراسة والعلاقة بينها. حتى يتم توضيح كيفية تصميم هذا النموذج لا بدَّ من توضيح المفاهيم الأساسية في هذا التصميم وهي مفهوم الكينونة ونوع الكينونة وأنواع العلاقات بين الكينونات وكذلك الواصفات (الصفات) المختلفة لهذه الكينونات.

4.1. الكينونة (Entity)

تُعرَّف الكينونة أنها شيء ما أو شخص أو مكان أو حدث أو مفهوم في البيئة التي يتم دراستها. مثال على ذلك: شخص معين داخل النظام، أو قسم معين أو سلعة معينة أو طالب معين.



الشكل 4: اسم الكينونة.

4.2. نوع الكينونه (Entity Type)

هي عبارة عن مجموعات كينونية لها الخصائص أو الصفات نفسها. فمثلاً:

- جميع الطلاب الذين يدرسون في الجامعة
- جميع المرضى الذين لهم سجلات داخل مستشفى
- جميع الموظفين الذين يعملون في مؤسسة.

يرمز نوع الكينونة داخل نموذج الكينونات العلائقية كما في الشكل 4.

نوع الكينونة قوية ونوع الكينونة ضعيفة

(Strong Entity and Weak Entity Type)

يمكن تصنيف الكينونات داخل مؤسسة معينة على أنها من نوع كينونة قوية أو نوع كينونة ضعيفة. نقصد هنا بقول كينونة قوية هي تلك الكينونات التي لا يعتمد

وجودها على كينونة أخرى. أما الكينونات الضعيفة فهي التي يعتمد وجودها على كينونات أخرى.

مثال: في بيئة نظام المحاسبة داخل الجامعة نرى أن الموظفين داخل النظام لا يعتمد وجودهم على كينونات أخرى. أما إذا أخذنا أبناء الموظفين في نظام المحاسبة فيعتمد وجودهم على وجود الآباء (الموظفين).

فوجود الأبناء يعتمد على وجود الآباء (الموظفين) داخل النظام أي إنه لا يمكن أن يتواجد ابن إلا إذا تواجد الأب (الموظف). تُسمى في هذه الحالة كينونة الأبناء بكينونة ضعيفة ويرمز لها كما في الشكل 5.



الشكل 5: الكينونة الضعيفة.

4.3. الواصفات (الصفات)

4.3.1. مفهوم الواصفة (Attribute)

تُمثل الواصفات الخصائص المختلفة للكينونه.

مثال:

نوع الكينونة	الواصفات
الطالب	رقم الطالب، اسمه. العنوان، التخصص
الموظف	رقم الموظف، الإسم، الراتب، الدرجة

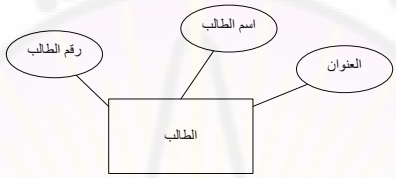
ويرمز لها كما في الشكل 6.



الشكل 6: الواصفة أو الخاصية.

مثال:

الشكل 7 يوضح واصفات الكينونة الطالب.



الشكل 7: واصفات كينونة الطالب.

هناك عدة أنواع من الواصفات:

1. الواصفة البسيطة (Simple Attribute)
2. الواصفة المركبة (Composite Attribute)
3. الواصفة ذات القيمة الواحدة (Single Value Attribute)
4. الواصفة ذات القيم المتعددة (Multi Values Attribute)
5. الواصفة المشتقة (Derived Attribute)

1. الواصفة البسيطة: نقصد هنا بالواصفة البسيطة أي إنّ الواصفة لا يمكن تَجزئتها إلى أجزاء. مثال: كينونة الطالب فإن صفة العمر للطالب لا تُجزأ إلى أجزاء. يمكن تمثيل هذه الواصفة كما هو موضح في الشكل 8.

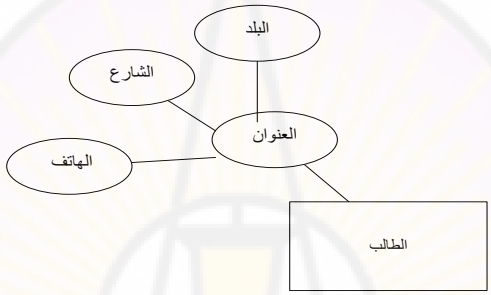


الشكل 8: واصفة بسيطة.

الواصفة المركبة: الواصفة المركبة هي الواصفة التي يمكن تقسيمها إلى أجزاء أخرى ذات دلالة. مثال: الصفة عنوان الطالب يمكن تقسيمها إلى أجزاء أخرى مثل المدينة، الشارع، العنوان البريدي و تمثل كما في الشكل 9.

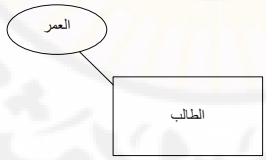
2. الواصفة ذات القيمة الواحدة: نقصد هنا أن الواصفة تأخذ قيمة واحدة لتلك الكينونة ويمكن أن تدعى بالواصفة فردية القيمة.

مثال: في كينونة الطالب صفة العمر تأخذ قيمة واحدة فقط ولا يمكن أن تأخذ أكثر من



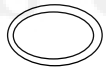
الشكل 9: واصفة مركبة: العنوان.

قيمة. نلاحظ أن العمر يمكن أن يكون ذات قيمة واحدة وكذلك بسيطة كما ذكرنا سابقاً. الشكل 10 يوضح واصفة ذات قيمة واحدة.



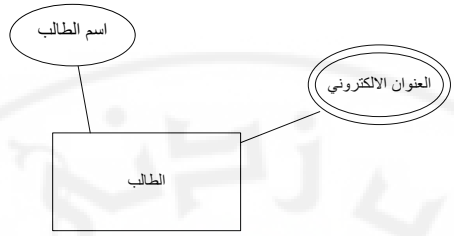
الشكل 10: واصفة ذات قيمة واحدة وهي بسيطة.

3. الواصفات ذات القيم المتعددة: هي الواصفات التي تأخذ أكثر من قيمة لنفس الكينونة ويرمز لها كما في الشكل 11.



الشكل 11: شكل الواصفة ذات قيم متعددة.

مثال: عنوان البريد الالكتروني لطالب حيث أنه يمكن أن يكون للطالب أكثر من بريد الكتروني و يمكن تمثيل ذلك كما في الشكل 12.



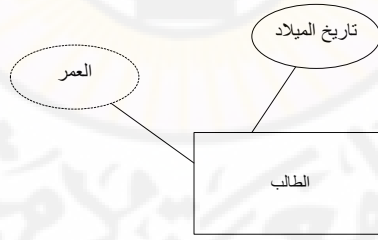
الشكل12: واصفة متعددة القيم: العنوان الإلكتروني.

4. الواصفة المشتقة: وهي الواصفة التي تُحدد من خلال واصفات أخرى. يرمز للواصفة المشتقة كما هم موضح في الشكل 13.



الشكل 13: واصفة مشتقة.

مثال: صفة العمر لطالب معين يمكن أن تُحسب من خلال معرفة تاريخ اليوم وتاريخ ميلاد ذلك الطالب. يمكن تمثيل ذلك كما هو موضح في الشكل 14.



الشكل 14: مثال واصفة مشتقة: العمر.

4.3.2. القيود ومجالات الواصفات

(Domain and Constraints of Attributes)

نقصد بمجال الواصفة مجموعة القيم التي يمكن أن تحدد تلك الواصفة. مثال: لو أخذنا كينونة الطالب وله واصفات رقم الطالب، اسم الطالب، عمر الطالب. يمكن تحديد المجال لكل واصفة كما يأتي:

- واصفة رقم الطالب: يجب ألّا يزيد عن 8 خانات رقمية.
- واصفة اسم الطالب: مكون من 30 حرف، 30 أقصى حد.
- واصفة عمر الطالب: يجب ألّا يقل عن 18 سنة أو لا يزيد على 35 سنة.

أما بالنسبة لخاصية المفتاح لواصفة معينة فإن لكل نوع كينونة هنالك صفة مميزة. نقصد هنا بالواصفة المميزة تلك التي لها قيم لا تتكرر مع الكينونات الأخرى والمكونة لنوع الكينونة وهذه الواصفة تدعى المفتاح.

مثال1: في نوع كينونة الطالب نلاحظ أنه لكل طالب هنالك رقم خاص به لا يتكرر لطالب آخر ولذلك يدعى رقم الطالب بالمفتاح لنوع كينونة الطالب وتمثل كما في الشكل 15.



الشكل 15: مثال واصفة مميزة: رقم الطالب.

مثال2: في نوع كينونة الموظف في شركة معينة نلاحظ أن رقم هوية الموظف لا يتكرر لموظف أخر ولذلك يدعى رقم هوية الموظف بالمفتاح لنوع كينونة الطالب.

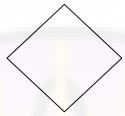
مثال3: في نوع كينونة القسم في جامعة ما نلاحظ أن كل قسم له رقم يميزه من قسم آخر ولذلك يدعى رقم القسم بالمفتاح للقسم.

في بعض الأحيان يتم تكوين المفتاح من مجموعة خصائص لنوع كينونة. مثال: في نوع كينونة مشروع نلاحظ أن هنالك خصائص مختلفة مثل اسم المشروع، رقم المشروع، مكان المشروع، القسم المشرف على المشروع. هنا يمكن القول إن اسم

المشروع و/أو رقم المشروع هي خصائص مفتاح المشروع لأنه لا يمكن أن يكون هنالك أكثر من مشروع بنفس الاسم وكذلك رقم المشروع .

4.4. أنواع العلاقات (Relationship Type)

نقصد بنوع علاقة الرابط هي مجموعة الارتباطات بين كينونات مختلفة ونستخدم الرمز الموضح في الشكل 16 للتعبير عن علاقة.



الشكل 16: رمز العلاقة (الارتباط).

يوجد عدة أنواع للعلاقات:

1. علاقة واحد . إلى . واحد

(One–to –One Relationship)

2. علاقة واحد . إلى . متعدد

(One–to–Many Relationship)

3. علاقة متعدد . إلى . متعدد

(Many–to–Many Relationship)

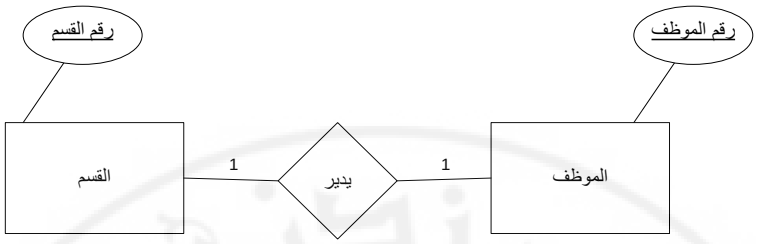
4. العلاقة التعاودية

(Recursive Relationship)

علاقة واحد . إلى . واحد

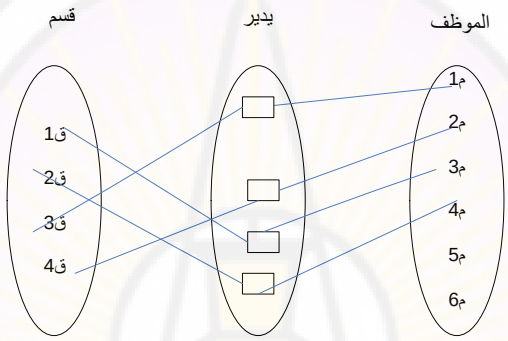
خلال هذه العلاقة يتم ربط كل عنصر في كينونة مع عنصر واحد في كينونة أخرى.

مثال: في شركة ما هنالك مجموعة من الأقسام وكل قسم يُديره موظف واحد. ولا يجوز أن يدير أكثر من موظف أي قسم في الشركة. هنا نحدد الكينونتين؛ الموظف والقسم والعلاقة بينهما (يدير) وفق المخطط الكينوني الموضح في الشكل 17.



الشكل 17: موظف واحد يدير قسم واحد.

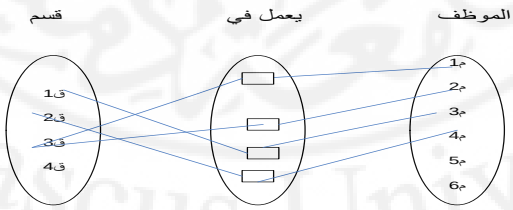
ويمكن توضيح المفهوم بشكل رياضي كما هو موضح في الشكل 18.



الشكل 18: موظف يدير قسم.

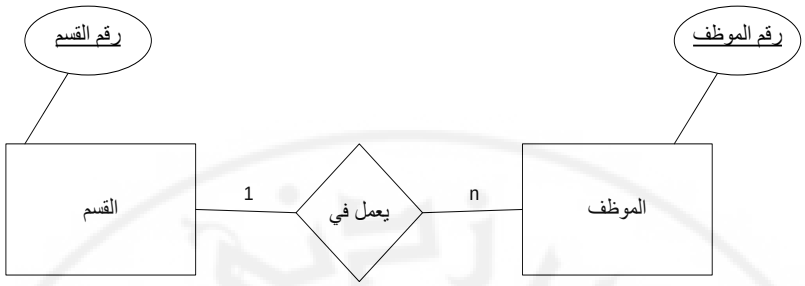
علاقة واحد . إلى . متعدد

في هذا النوع من العلاقات لدينا أكثر من عنصر في كينونة يرتبط مع عنصر واحد من كينونة أخرى. مثال: في شركة ما هناك أقسام. يعمل في كل قسم مجموعة من الموظفين، وكل موظف يمكن أن يعمل في قسم واحد كما في الشكل 19.



الشكل19: موظف يعمل في قسم.

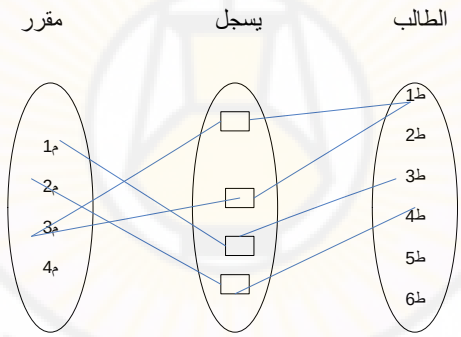
المخطط الذي يمثل هذه العلاقة هو كما في الشكل 20.



الشكل 20: علاقة واحد إلى متعدد.

علاقة متعدد- إلى . متعدد

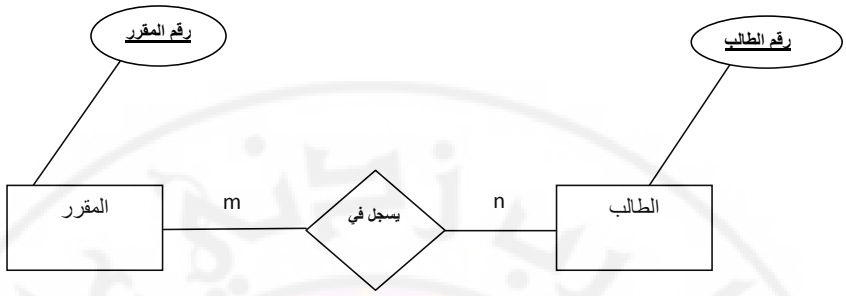
كل عنصر في الكينونة الأولى مرتبطة بأكثر من عنصر في كينونة ثانية. كذلك كل عنصر في الكينونة الثانية مرتبط بأكثر من عنصر في الكينونة الأولى كما في الشكل 21.



الشكل 21: علاقة متعدد الى متعدد.

مثال: في نظام التسجيل في الجامعة كل طالب يسجل أكثر من مقرر، و المقرر الواحد يسجل فيه أكثر من طالب.

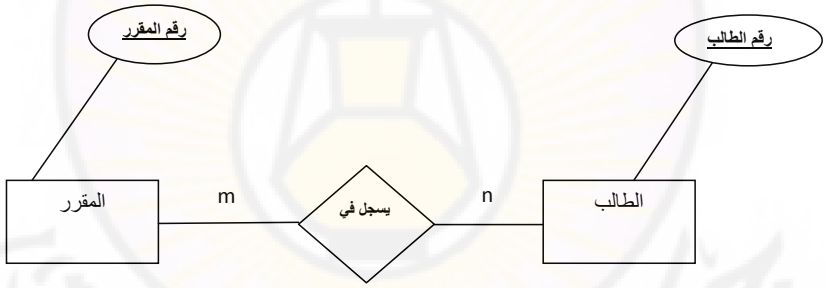
تُمثل العلاقة كما في الشكل 22.



الشكل 22: علاقة متعدد الى متعدد.

ملاحظة:

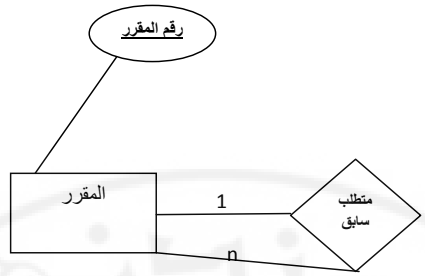
يمكن أن يكون لدينا نوع الكينونة نفسها من أجل أنواع ارتباطات مختلفة كما في الشكل 23.



الشكل 23: نفس نوع الكينونة لأنواع ارتباطات مختلفة.

العلاقة التعاودية

كل عنصر في كينونة يمكن أن يرتبط مع عنصر من الكينونة نفسها. مثال: في نظام التسجيل في الجامعة المقرر يمكن أن يكون له مقرر آخر كمتطلب سابق كما في الشكل 24.



الشكل 24: العلاقة التعاودية.

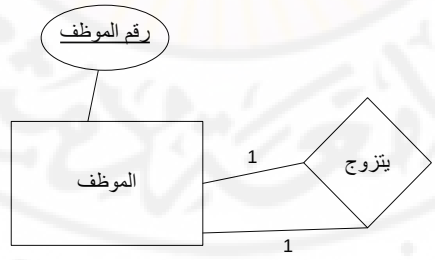
درجة العلاقة (Relation Degree)

تعبر درجة العلاقة عن عدد الكينونات التي ترتبط بعضها ببعض من خلال علاقة واحدة. يوجد هنالك عدة درجات شائعة في نموذج الكينونات العلائقية منها الأحادية (من درجة واحدة) ومنها الثنائية (من درجة 2) و منها الثلاثية (من درجة 3). على الرغم من وجود درجات علائقية أخرى إلا أنها لا تُستخدم بشكل كبير في الواقع العملي.

4.4.1. درجة العلاقة الأحادية

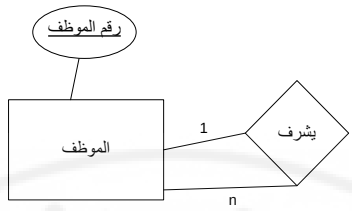
(Unary Relationship)

مثال1: موظف في شركة ما يمكن أن يتزوج من موظفة في تلك الشركة كما في الشكل 25.



الشكل 25: مثال 1 على درجة العلاقة الأحادية.

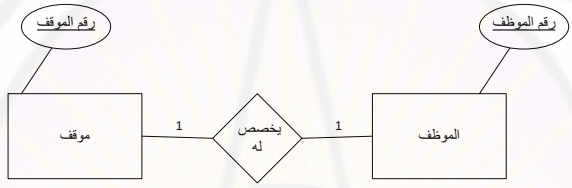
مثال2: كل موظف يشرف على مجموعة من الموظفين.كما في الشكل 26.



الشكل 26: مثال 2 على درجة العلاقة الأحادية.

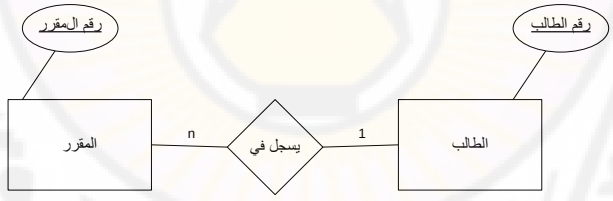
4.4.2. درجة العلاقة الثنائية

مثال1: يُحدَد لكل موظف في شركة ما موقف واحد خاص لسيارته كما في الشكل 27.



الشكل 27: مثال 1 على درجة العلاقة الثنائية.

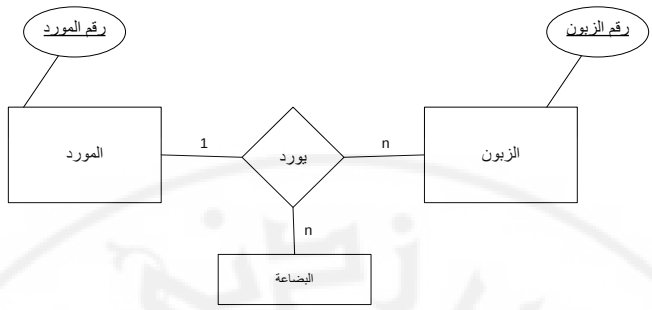
مثال2: كل طالب يسجل في أكثر من مقرر كما هو موضح في الشكل 28.



الشكل 28: مثال 2 على درجة العلاقة الثنائية.

4.4.3. درجة العلاقة الثلاثية

مثال: المورد يورد بضاعة إلى زبائنه كما هو موضح في الشكل 29.



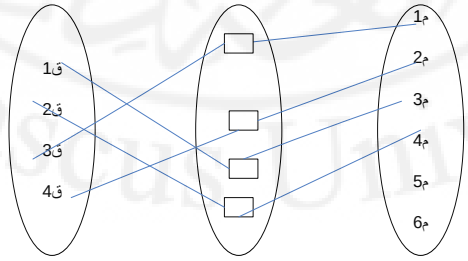
الشكل 29: درجة العلاقة الثلاثية.

4.5. قيد المشاركة (Participation Constraint)

يتم الحديث عن هذا القيد في حال وجود كينونة مرتبطة بكينونة أخرى. يوجد نوعان من قيود المشاركة وهي قيد المشاركة الكلي وقيد المشاركة الجزئي.

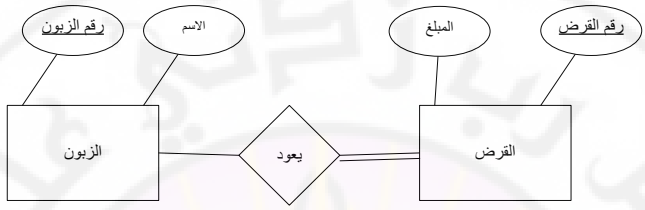
4.5.1. قيد المشاركة الكلي (Total Participation Constraint)

في هذه الحالة كل كينونة يجب أن ترتبط بوحدة (.....) في العلاقة. يتم تمثيل هذا القيد في مخطط الكينونات عن طريق رسم خط مزدوج يربط الكينونات بهذه العلاقة. مثال: في بنك ما إذا فرضنا أن كل قرض يجب أن يعود على الأقل إلى زبون واحد. يمكن توضيح ذلك كما في الشكل 30.



الشكل 30: قيد المشاركة الكلي قرض(ق)، زبون(م).

في هذه الحالة نلاحظ أن كل قرض يجب أن يشارك في علاقة يعود مع الزبون وهذه مشاركة كلية. وفي مخطط الكينونات تُمثل كما في الشكل 31.



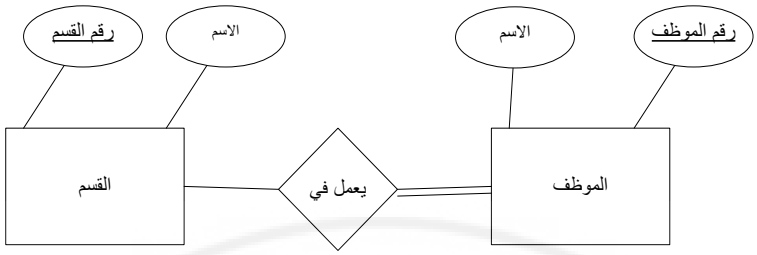
الشكل 31: قيد المشاركة الكلي(يعود لي).

3.1.1.قيد المشاركة الجزئي

(Partial Participation)

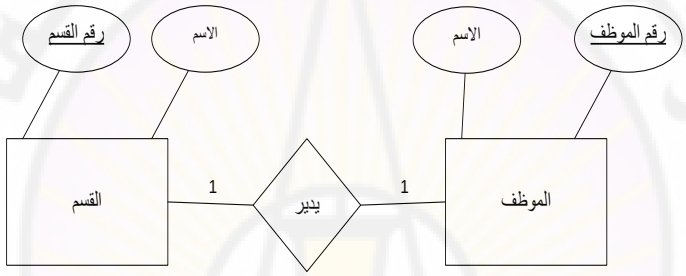
في هذه الحالة فإن بعض الكينونات ترتبط بوحده (Instance) في العلاقة. يتم تمثيلها برسم خط مفرد يربط الكينونات المرتبطة بهذه العلاقة. أي إن بعض الكينونات لا تشارك في العلاقة. في المثال السابق الزبون ليس شرطاً أن يكون مُقترِضاً. وبالتالي فإن بعض الزبائن لا يرتبطون بعلاقة (يعود)، ولهذا مُثلت بخط مفرد.

مثال آخر: في شركة ما تضم مجموعة من الموظفين، إذا فرضنا أن كل موظف يجب أن يعمل في قسم واحد فقط، والعلاقة بين الموظف والقسم هي علاقة يعمل في. إذا فرضنا أيضاً أن كل موظف يعمل في قسم. أي لا يمكن أن يكون هنالك موظف لا يعمل في قسم ما. و بالتالي فإن كينونة الموظف ترتبط في علاقة يعمل بعلاقة مشاركة كلية.



الشكل 32: علاقة المشاركة الكلي.

و عليه فإن التمثيل يتم بخط مزدوج كما في الشكل 32. أما إذا قلنا إن كل قسم يديره موظف ما فإن بعض أو جزء من مجموعة كينونات الموظفين ترتبط بكينونة القسم عن طريق العلاقة يمتلك و ليس الكل كما في الشكل 33.



الشكل 33: علاقة المشاركة الجزئي.

مثال: ارسم مخطط الكينونات (ER-Diagram) والمتعلق بقاعدة البيانات لبنك ما والذي متطلباته كما يأتي:

- يوجد للبنك زبائن ويحتفظ البنك بمعلومات عن هؤلاء الزبائن وتشمل رقم الهوية وهو رقم مفرد واسم الزبون وعنوان الزبون.
- يمكن للزبون أن يملك أكثر من حساب، حيث تشمل معلومات الحساب رقم الحساب وهو مفرد والرصيد ونوع الحساب.
- يمكن للزبون أن يحصل على أكثر من قرض وتشمل معلومات القرض رقم القرض وهو مفرد ونوع القرض وقيمة القرض.
- يمكن لأكثر من زبون أن يشتركوا في قرض واحد.
- لا يُشترط أن يكون للزبون حساب في البنك للحصول على قرض.
- لا يُشترط لفتح حساب بالبنك أن يحصل الزبون على أي قرض.

لنتحدث عن كيفية عمل مخطط الكينونات كما يأتي:

أولاً: نحدد الكينونات.

1. الزبون (رقم الهوية، الاسم، العنوان)
2. الحساب (الرقم، الرصيد، النوع)
3. القرض (الرقم، النوع، القيمة)

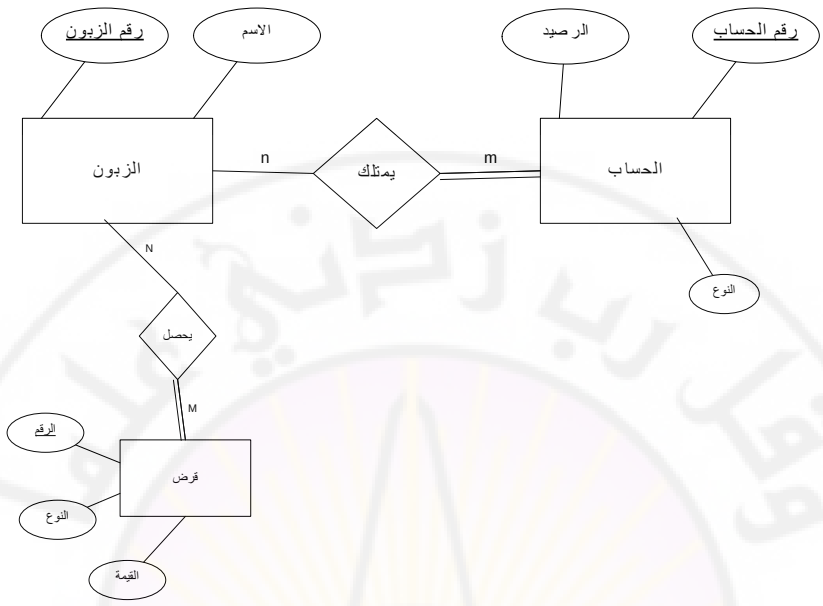
ثانياً: نحدد العلاقة بين الكينونات.

1. يمتلك (الزبون يكون له أكثر من حساب)
2. يحصل (الزبون يحصل على قرض)

ثالثاً: نحدد القيود.

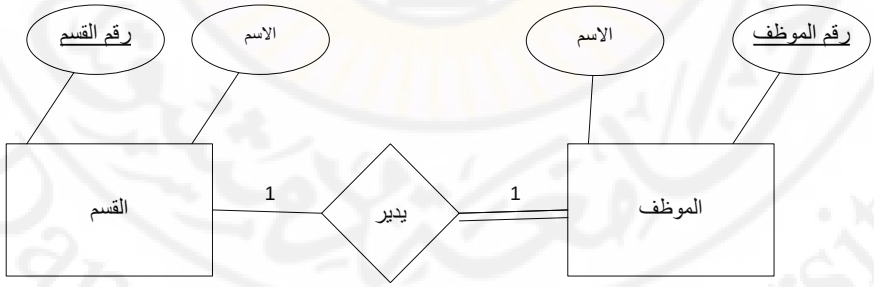
1. لا يُشترط للحصول على قرض أن يكون الزبون له حساب بالبنك.
2. لا يُشترط لفتح حساب في البنك الحصول على أي قرض.

من القيد الأول نلاحظ أن كل حساب بالبنك لا بد أن يكون له زبون، وهذا يعني أن كل حساب يجب أن يمتلكه زبون، وبالتالي فإن مشاركة كينونة الحساب هي مشاركة كلية في العلاقة. كذلك بالنسبة للقيد الثاني فإن كينونة القرض هي مشاركة كلية في العلاقة يحصل لأنه لا يمكن أن يكون هنالك قرض بدون زبون. بناءً على ما سبق نجد أن المخطط سيكون كما في الشكل 34.



الشكل 34: Er–Digram الموافق للنظام البنكي.

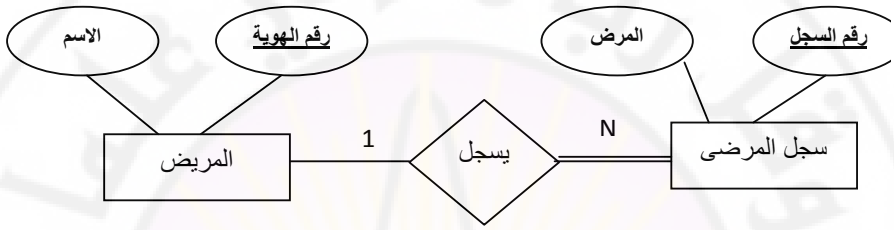
مثال آخر: في علاقة إدارة الموظف لقسم معين، فإن كيان القسم يشارك في العلاقة بمشاركة كلية لأنه لا يمكن أن يكون هنالك قسم بدون موظف يديره. بينما مشاركة الموظف في العلاقة يدير هي مشاركة جزئية لأنه ليس كل موظف يدير قسم كما في الشكل 35.



الشكل 35: Er–Digram لشركة.

أمثلة متنوعة:

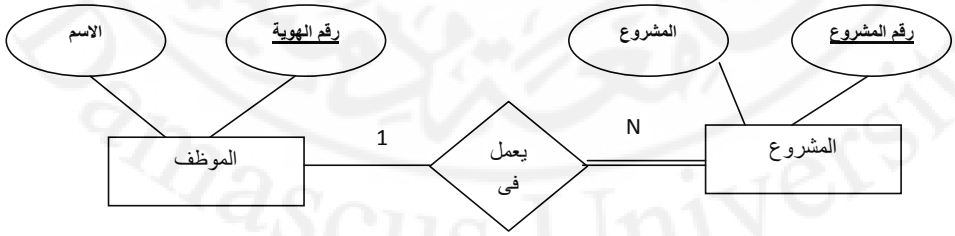
مثال1: المريض يمتلك سجلاً مرضياً واحداً أو أكثر و لا يمكن أن يكون هنالك سجل مرضى ليس مرتبطاً مع مريض كما في الشكل 36.



الشكل 36: تسجيل مريض(أو أكثر) في سجل المرضى.

نلاحظ في هذا المثال أننا مثلنا الرابط بين السجل المرضي والعلاقة يدير بخط مزدوج، وهذا للدلالة على أنها مشاركة كلية ولكن ليس بالضرورة أن يكون لكل مريض سجل مرضي، وبالتالي فإن الرابط بين المريض والعلاقة يمتلك مُثلت بخط واحد وذلك للدلالة على أنها مشاركة جزئية.

مثال2: يعمل في كل مشروع موظف أو أكثر، ولكن يمكن أن يكون هنالك موظفون لا يعملون في مشروع، وكذلك يمكن للموظف أن يعمل في أكثر من مشروع كما في الشكل 37.

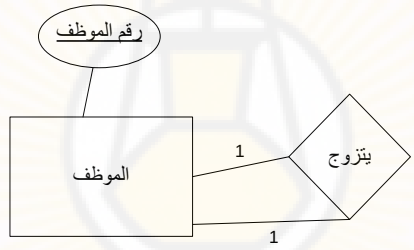


الشكل 37: علاقة موظف (أو أكثر) يعمل في مشروع (أو أكثر).

نلاحظ في هذا المثال أننا مثلنا الرابط بين المشروع والعلاقة يعمل في بخط مزدرج وهذا للدلالة على أنها مشاركة كلية لأنه لا يُعقل أن يكون هنالك مشروع بدون موظفين يعملون به. لكن الرابط من جهة الموظف هي مشاركة جزئية (بخط واحد) لأنه يمكن أن يكون هنالك موظفون لا يعملون في مشروع كما ذكرنا في الفقرة السابقة.

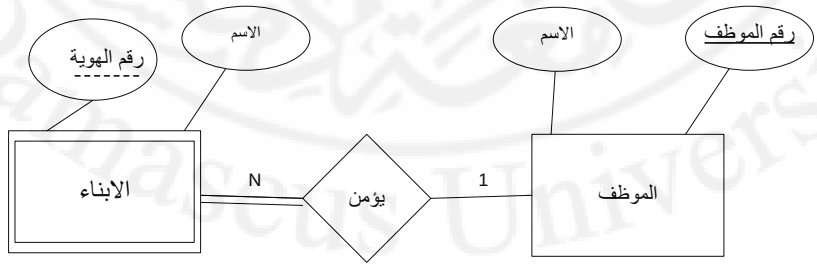
مثال3: في شركة معينة يمكن أن يكون هنالك موظفون متزوجون من أكثر من موظفة، ولكن ليس بالضرورة أن يكون جميعُ الموظفين متزوجين كما في الشكل 38.

هنا نلاحظ أن الرابط بخط واحد، لأن المشاركة جزئية، لأنه من الممكن أن يكون الموظف غير متزوج.



الشكل 38: علاقة يتزوج بين الموظفين في شركة.

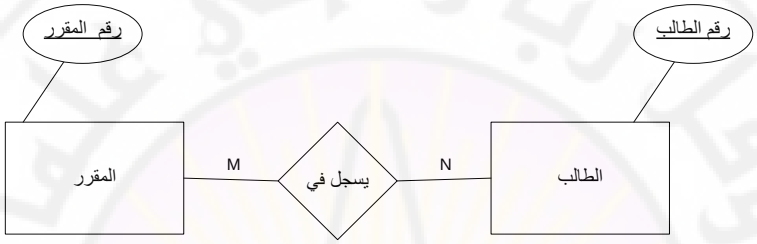
مثال4: تقوم شركة ما بتقديم تأمين صحي لموظفيها، كما أن الشركة توفر التأمين الصحي لأبناء الموظفين كما في الشكل 39.



الشكل 39: علاقة يؤمن بين الموظفين والأبناء.

نلاحظ كذلك أنه إذا تواجد الابن فلا بد أن يشارك في العلاقة مشاركة كاملة وليست جزئية. كما أن كينونة الأبناء هي كينونة ضعيفة لأن وجودها يعتمد على كينونة الموظف. لاحظ أن الكينونة الضعيفة لا يوجد لها مفتاح رئيسي من خلال خصائصه.

مثال5: في نظام التسجيل في جامعة ما كل طالب يمكن أن يُسجل في مقرر واحد أو في أكثر من مقرر. كما أن المادة يُسجل فيها أكثر من طالب كما في الشكل 40.



الشكل 40: علاقة يسجل في بين الطالب والمقرر.

نلاحظ هنا أنه يمكن أن تتواجد مقررات لم يسجل فيها أي طالب وكذلك الطالب يمكن أن لا يسجل في أي مقرر. نلاحظ أيضاً أن كينونة الطالب ترتبط بالعلاقة يسجل في بعلاقة مشاركة جزئية وكذلك كينونة المقرر ترتبط بالعلاقة يسجل في بمشاركة جزئية.

مثال6: تقوم شركة ما بتنفيذ عدد من المشاريع. يعمل في كل مشروع مجموعة من الموظفين. خلال عمليه تحليل النظام وجدنا النقاط الآتية:

1. الشركة مكونة من عدد من الأقسام و كل قسم يملك رقم القسم وهو رقم مفرد، اسم القسم، موقع القسم.
2. الموظفين في الشركة يملكون رقم الموظف وهو رقم مفرد، اسم الموظف، عنوان الموظف، راتب الموظف، عمل الموظف.
3. الموظف يتبع لقسم واحد فقط و كل قسم يعمل فيه أكثر من موظف.
4. المشروع يملك رقم المشروع وهو رقم مفرد، واسم المشروع، موقع المشروع.
5. المشروع يتبع لقسم معين و يمكن للقسم أن يُدير أكثر من مشروع.
6. يعمل في المشروع أكثر من موظف ويمكن أن يعمل الموظف في أكثر من مشروع واحد.

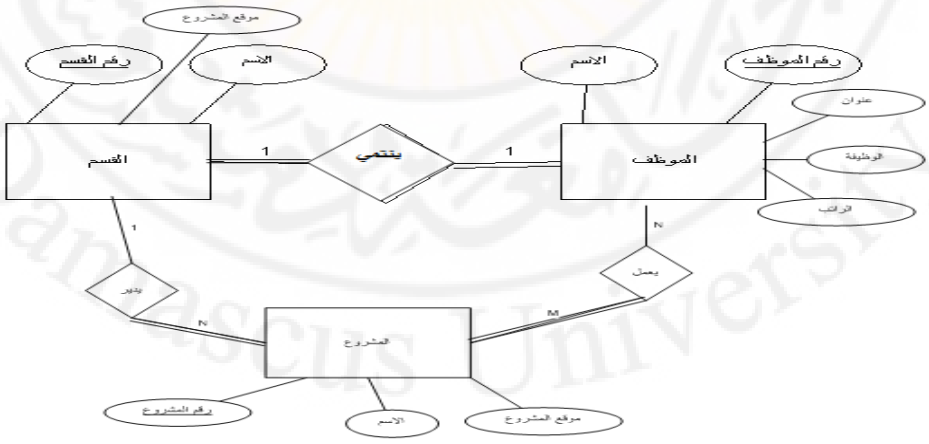
7. كل موظف له عدد ساعات عمل في الأسبوع في كل مشروع.

من خلال ما سبق نلاحظ أن الكينونات الموجودة في النظام هي:

- القسم (رقم القسم، اسم القسم، موقع القسم)
- الموظف (رقم الموظف، اسم الموظف، عنوان الموظف، راتب الموظف، عمل الموظف)
- المشروع (رقم المشروع، اسم المشروع، موقع المشروع)

أما العلاقات فهي:

- ينتمي إلى: و هي علاقة بين الموظف والقسم، حيث إن القسم يعمل به أكثر من موظف. هذه العلاقة من نوع.
- يدير: و هي علاقة القسم والمشروع حيث إن القسم يدير مشروع أو أكثر. لكن المشروع لا يُنفذ إلا في قسم واحد. هذه العلاقة أيضاً من نوع واحد . إلى . متعدد.
- يعمل: و هي علاقة بين الموظف والمشروع، حيث إن الموظف يعمل في أكثر من مشروع، و المشروع يعمل فيه أكثر من موظف. هذه العلاقة من نوع متعدد . إلى . متعدد و هذه العلاقة لها واصفة عدد ساعات العمل كما في الشكل 41.



الشكل 41: المخطط التفصيلي للعلاقات في شركة ينتمي الى

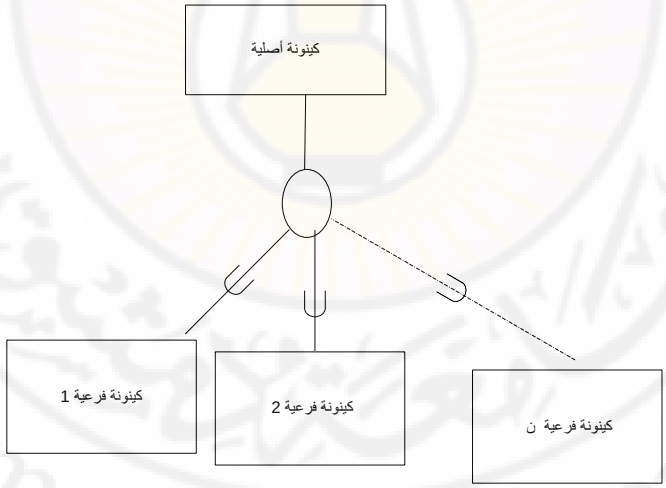
5. نموذج الكينونات المتطور

(Enhanced Entity, Eer Model) Relationship

تحدثنا في كل ما سبق عن مخطط الكينونات الأساسي، والذي تم استخدامه في منتصف عام 1979. لكن نتيجة التطورات التي حدثت في بيئة العمل والتعقيدات في العمليات داخل العمل فإن البيانات في العمل أصبحت أكثر تعقيداً، وبالتالي فإنه لا بد من تطوير مخطط الكينونات الأساسي لإحتواء هذه التطورات والتغيرات .

تتلخص هذه التطورات بإضافة مفاهيم الأنواع الأصلية (Superclass) والأنواع الفرعية (Subclass) و كذلك مفهوم الوراثة (Inheritance).

كمثال على ذلك نوع الكينونة السيارة التي يمكن أن تمثل نوع كينونة أصلية. أما نوع الكينونة سيارة شحن، وسيارة ركاب، وسيارة سباق هي أنواع كينونة فرعية. كذلك فإن الكينونة الفرعية يمكن أن ترث صفة (أو واصفة) أو أكثر من نوع الكينونة الأصلية. يتم تمثيل مفهوم الكينونات الأصلية والفرعية على الشكل 42.



الشكل 42: نموذج الكينونات التطور.

إن الرمز ∪ الموجود على الخط الواصل بين الرمز 0 و كل كينونة فرعية يعني أن الكينونة الفرعية هي مجموعة جزئية من الكينونة الأصلية. هذا الرمز هو اختياري لأن مفهوم المجموعة الجزئية واضحة في هذا الموضوع.

مثال: افترض في شركة معينة وجود ثلاثة أنواع من الموظفين: المهندس، والفني، والسكرتير. افترض أيضاً أن خصائص كل من هؤلاء الموظفين هي كما يأتي:

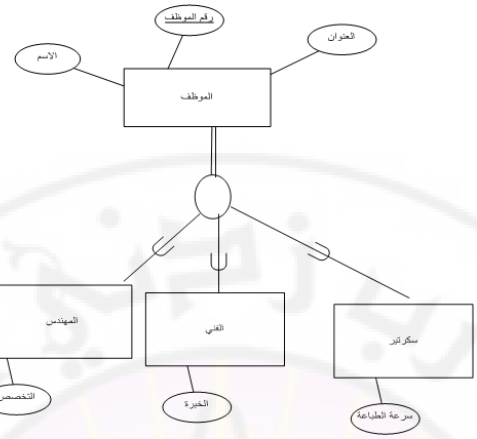
- المهندس: رقم المهندس، اسم المهندس ، عنوان المهندس ، اختصاص المهندس.
- الفني: رقم الفني، اسم الفني ، عنوان الفني ، خبرة الفني.
- السكرتير: رقم السكرتير، اسم السكرتير ، عنوان السكرتير, سرعة طباعة السكرتير.

نلاحظ في هذا المثال أن جميع أنواع الموظفين لهم خصائص متشابهة عدا بعض الخصائص. فمثلاً المهندس له خاصية اختصاص المهندس، التي لا توجد لدى الفني والسكرتير وكذلك الفني له خاصية الخبرة التي لا توجد في كل من المهندس والسكرتير. أما السكرتير فله خاصية سرعة الطباعة التي لا توجد في كل من المهندس والفني. أما باقي الخصائص فهي مشتركة.

في هذه الحالة فإنه بالإمكان عمل مخطط كينونات كما يأتي:

1. عرف نوع الكينونة وسمِها موظف، لها الخصائص المتشابهة في كل أنواع الوظيفة.

2. عرف كينونة لكل نوع من أنواع الوظيفة، مع تحديد الخاصية المميزة لكل نوع كينونة. يمكن توضيح ذلك وفق الشكل 43.

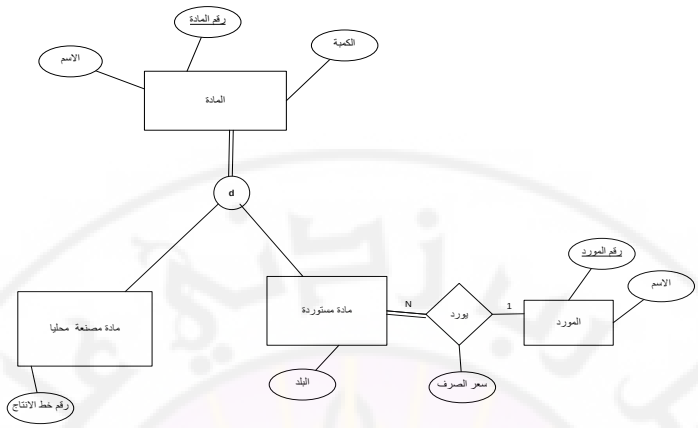


الشكل 43: المخطط الموافق لشركة فيها عدة أنواع من الموظفين

نلاحظ هنا أننا وضعنا خطاً مزدوجاً يربط بين الموظف ونوع الكينونات الفرعية (المتشابهة الكلية)، و هذا للدلالة على أن الموظف يجب أن يكون واحداً من الأنواع الثلاثة من الموظفين و لا يجوز أن يكون موظفاً ليس من أحد هؤلاء الأنواع. أما إذا وضعنا خطاً واحداً فهذا يعني (مشاركة جزئية)، أي إنه يمكن أن يكون هنالك موظف ليس من أحد أنواع الموظفين الثلاثة.

5.1. التخصيص و التعميم (Specialization and Generalization)

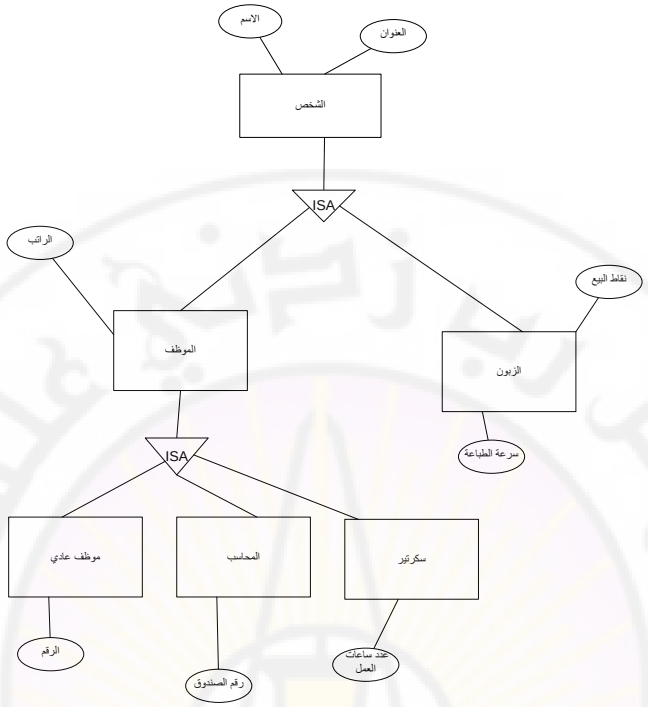
التخصيص هي عملية من الأعلى إلى الأسفل و هي على عكس التعميم التي هي من الأسفل إلى الأعلى ونقصد بالتخصيص هي أنواع فرعية للنوع الأصلي في علاقة أصلي/فرعي.



الشكل 44: مخطط يوضح علاقتي التخصيص و التعميم.

مثال1: في شركة معينة هنالك مواد تُستخدم. هذه المواد تُصنَف على أنها مواد مُصنعة محلياً في الشركة، بالإضافة إلى مواد من خارج الشركة. المورد يورد هذه المواد بسعر معين. خواص المواد المصنعة محلياً هي رقم المادة، رقم خط الإنتاج، اسم المادة، كمية المادة. أما خصائص المواد المشتراة من خارج الشركة فهي رقم المادة، اسم المادة، كمية المادة. الشكل 44 يوضح ذلك.

مثال2: في نظام ما مجموعة الكينونات الأشخاص يمكن تخصيصها إلى مجموعة كيانات موظفين ومجموعة كيانات زبائن. يشار إلى هذه العلاقة بعلاقة Isa وهي علاقة من الأعلى إلى الأسفل كما هو موضح في الشكل 45.



الشكل 45: مخطط يوضح علاقة Isa.

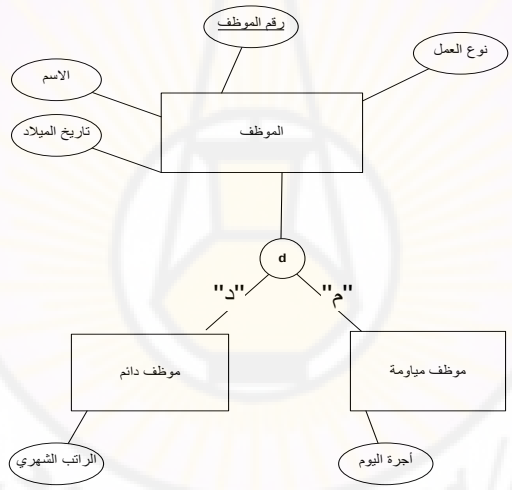
كذلك فإن مجموعة كينونات يمكن تخصيصها إلى مجموعة كيانات سكرتير ومحاسب وموظف عادي.

5.2. تحديد القيود على علاقة Isa من النوع أصلي/فرعي

يوجد عدة قيود يمكن وضعها على علاقة Isa. يوجد هنالك قيد يمكن وضعه على خاصية معينة للنوع الأصلي بهدف تمديد الكينونات التي ستصبح عضواً في كل نوع من الأنواع الفرعية، وهذا القيد يراعي قيد التعريف للأنواع الفرعية. مثال: افترض في شركة ما أنه يوجد مجموعة من الموظفين، حيث إن الموظف له خاصية نوع الوظيفة (Job Type). بناءً على خاصية نوع الوظيفة تلك يمكن تحديد العضوية في النوع الفرعي. فعلى سبيل المثال يمكن تحديد العضوية في النوع الفرعي " سكرتير" (Secretary) و ذلك خلال وضع الشرط. بالتالي فإن أي موظف له نوع الوظيفة "سكرتير" يجب أن ينتمي إلى النوع الفرعي سكرتير فقط.

إذا كانت كل الأنواع الفرعية في التخصص لها شرط العضوية نفسه على الخاصية نفسها في النوع الأصلي فإن هذا التخصيص ..... معرف الخاصية .....
أما إذا لم يكن هنالك مثل هذا الشرط فإن النوع الذي يسمى تعريف المستفيد .....
وتحدد العضوية في مثل هذا النوع بوساطة مستخدمي قاعدة البيانات عند تطبيق عملية إضافة كينونة إلى النوع الأصلي .

مثال : المثال السابق (شكل 45)، نرى أن الموظف إما أن يكون مهندساً أو فنياً أو سكرتيراً و لا يمكن أن يكون اثنين منهم أو كلهم في نفس الوقت. ولذلك أشرنا إلى ذلك علىء طريق الرمز ..... قيد التفكيك (.....) كما في الشكل 46.



الشكل 46: مخطط يوضح القيود على علاقة Isa.

وهنالك قيود أخرى على التخصيص منها:

1. قيد الإتمام (Completence Constraints)

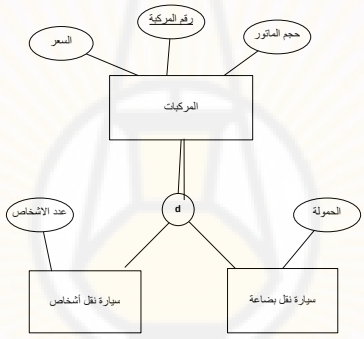
يحدد هذا القيد أن كل كينونة من نوع كينونة أصلية يجب أن تنتمي لنوع كينونة فرعي معين. ويعبر عن ذلك بخط مزدوج .

مثال : في شركة معينة هنالك مجموعة من الموظفين. وهؤلاء الموظفون إما أن يكونوا من الموظفين الدائمين الذين يحصلون على راتب شهري أو لموظفي المياومة ولا يمكن أن يكون موظفاً غير هؤلاء الصنفين .وقيد الإتمام له تصنيفات:

I. التخصيص الكامل: هذا يعني أن كل كينونة من كينونات النوع الأصلي يجب أن تكون واحداً من الكينونات في النوع الفرعي.

II. التخصيص الجزئي: يعني أن تكون كل كينونة من كينونات النوع الأصلي لا تنتمي إلى أي من الكينونات الفرعية.

مثال: في نظام المركبات، هنالك مركبات تدعى مركبة تقلّ أشخاصاً وهناك نقل بضاعى كما هو موضح في الشكل 47.



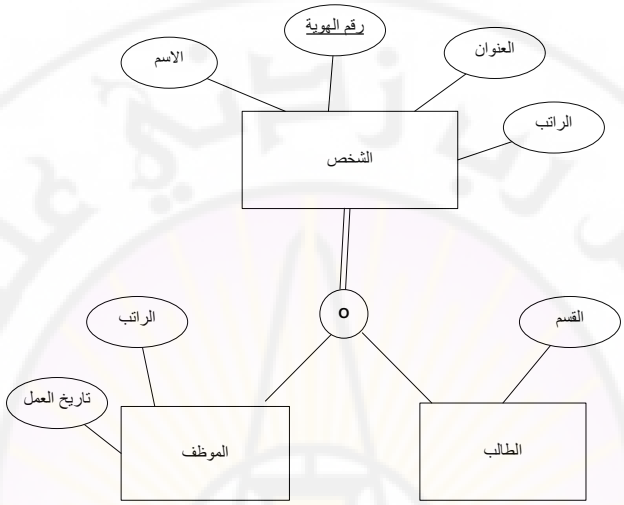
الشكل 47: مخطط يوضح مثال المركبات.

في هذا المثال يمكن أن تكون مركبات نوع دراجة وهي ليست من أي نوع من نوع الكينونات الفرعية. ولذلك سميت هذه بالتخصيص الجزئي ولذلك حدد الخط الواصل بين المركبات والحلقة بخط واحد.

2. قيد التفكيك

يعني قيد التفكيك أن كل كينونة في نوع كينونة أصلي يمكن أن تنتمي إلى نوع كينونة فرعية واحد فقط. أما إذا كانت تنتمي إلى أكثر من نوع كينونة فتدعى قيد التداخل. أي قيد التداخل يسمح بأن تنتمي كينونة إلى أكثر من نوع كينونة فرعية ويرمز لقيد التداخل بالرمز (O) داخل الحلقة بدلاً من D في قيد التفكيك .

مثال: في نظام الجامعة. هناك مجموعة من الأشخاص ويصنف هؤلاء الأشخاص مع تمثيل مجموعات موظفين و طلاب. حيث يمكن للطالب أن يكون موظفاً في الجامعة. يمثل هذا كما في الشكل 48.



الشكل 48: مخطط الكينونات الموافق لنظام الجامعة.

6. التحويل من نموذج الكينونات والارتباطات الى النموذج العلائقي

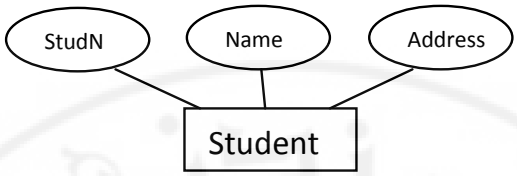
ERD to Realation Model Mapping

يمكننا تحويل نموذج الكينونات والارتباطات إلى النموذج العلائقي، وعملية التحويل هي عبارة عن مجموعة من الخطوات المحددة التي يتبعها مصمم قاعدة البيانات. هذه الخطوات تعتمد على فهم المصمم للمبادئ التي تم عرضها سابقاً والتي تخص الكينونات وأنواعها والواصفات والارتباطات والمفاتيح الرئيسية والغريبة. فيما يأتي سنبين الخطوات الأساسية التي يجب تنفيذها للحصول على النموذج العلائقي انطلاقاً من نموذج الكينونات و الارتباطات.

بشكلٍ عام كل كينونة تُحول إلى علاقة (جدول) لها الواصفات البسيطة الموجودة في الكينونة نفسها، ومفتاح هذه العلاقة هو مفتاح (مميز) الكينونة نفسه.

فمثلاً النموذج المبين في الشكل 49 يعطينا العلاقة الأتية:

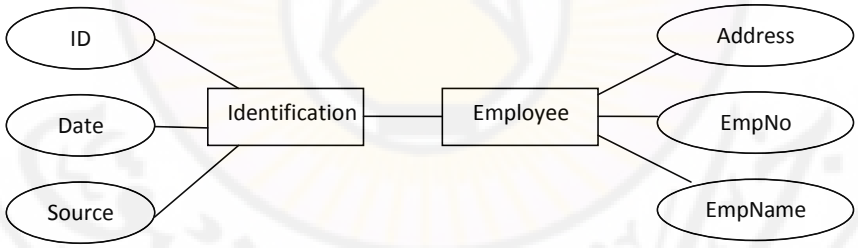
- Student(StudentN°,Name,Address)



الشكل 49: علاقة لها نفس الواصفات البسيطة.

➢ من أجل كل علاقة من النوع one-to-one يتم التحويل كما يأتي:

- نحدد الكينونات المرتبطة بعضها ببعض.
- كل كينونة تُحول إلى علاقة (جدول).
- اختيار أحد الجداول وإضافة المفتاح الأساسي للجدول الآخر كمفتاح غريب في الجدول الذي تم اختياره.
- في هذه الحالة من الممكن دمج الكينونتين معاً في كينونة واحدة.



الشكل 50: علاقة من النوع one-to-one.

كمثال على ذلك انطلاقاً من النموذج المبين في الشكل 50 نحصل على العلاقات التالية:

- Employee(EmpN°,EmpName,Address)
- Identification(ID,Date,Source,EmpN°)

أو

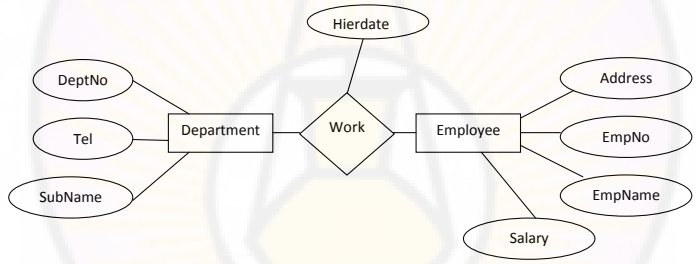
- Employee(EmpN°,EmpName,Address,ID°)
- Identification(ID°,Date,Source)

➢ من أجل كل علاقة ثنائية من الشكل on-to-many نقوم بما يأتي:

•نحدد العلاقة (الجدول) T_n الذي يمثل الكينونة الموجودة عند الجانب N (many) في الارتباط R.

•نضيف المفتاح الأساسي للكينونة الأخرى (الذي يمثل الكينونة الموجودة عند الجانب 1 في الارتباط R) كمفتاح غريب في الجدول T_n.

•نضيف أي واصفات موجودة في العلاقة R للجدول T_n.



الشكل 51: علاقة من النوع one-to-many.

كمثال على ذلك نجد العلاقتين الأتيتين الناتجتين عن النموذج المبين في الشكل 51.

- Employee(EmpN°,EmpName,Address,Salary,DeptN°,Hierdate)
- Department(DeptN°,Dname,Tel)

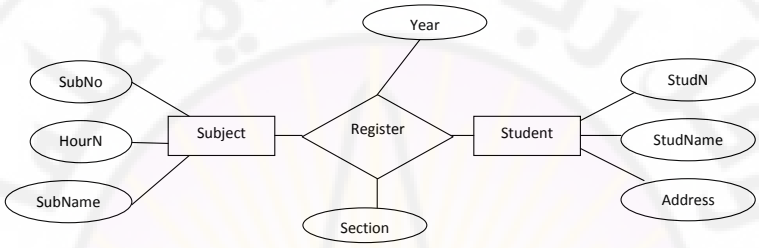
➢ من أجل كل علاقة ثنائية من الشكل many-to-many نقوم بما يأتي:

•ننشئ علاقة جديدة T تمثل الارتباط R.

•نضيف المفاتيح الأساسية للكينونتين المرتبطتين بالارتباط R كمفتاحين غريبين في العلاقة T.

•نضيف الواصفات الموجودة في الارتباط R للجدول T.

•المفاتيح الأساسية للعلاقة T هي مجموعة المفاتيح الغريبة التي تم ضمها للعلاقة T ، وتمثل المفتاتيح الأساسية للكينونتين المرتبطتين بالارتباط R.



الشكل 52: علاقة من النوع many-to-many.

كمثال على ذلك نورد العلاقات الأتية التي تنتج من تحويل النموذج المبين في الشكل 52.

- Student(StudN°,StudName,Address)
- Subject(SubN°,SubName,HourN°)
- Register(StudN°,SubN°,year,Section)

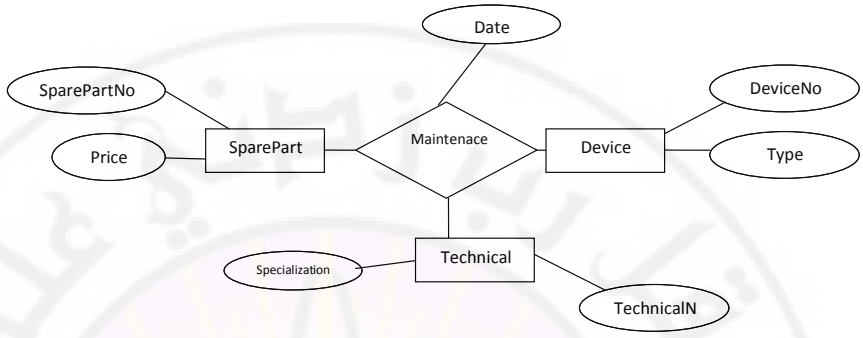
➢ من أجل العلاقات الثلاثية فما فوق:

•ننشئ علاقة جديدة T للارتباط R.

•نضيف المفاتيح الأساسية للكينونات المرتبطة بالارتباط R كمفاتيح غريبة في العلاقة T.

•نضيف الواصفات الموجودة في الارتباط R للعلاقة T.

• المفتاح الأساسي للعلاقة T هو مجموعة المفاتيح الغريبة التي تم ضمها إلى العلاقة T و تمثل المفاتيح الأساسية للكينونات المرتبطة بالارتباط R.



الشكل 53: مثال على علاقة ثلاثية أو أكثر.

فمثلاً النموذج الموضح في الشكل 53 يمكن تحويله للحصول على العلاقات الأتية:

- Device(DeviceN°,Type)
- SparPart(SparPartN°,Price)
- Technical(TechnicalN°,Special,Specialization)
- Maintenance(DeviceN°, SparPartN°, TechnicalN°,Date)

أسئلة البحث:

1. على ماذا تشمل الدراسة المبدئية للنظام القائم ؟
2. عرف هيكل (مخطط) قاعدة البيانات (Database Schema).
3. ماهي الخطوات الأساسية التي يمكن اتباعها لتصميم قاعدة بيانات؟
4. مم يتكون النموذج العلائقي لقواعد البيانات؟
5. اذكر خواص الجداول في قواعد البيانات العلائقية؟
6. عرف كل مما يأتي:

المفتاح الأساسي للجدول: Primary Key – المفتاح الغريب: Foreign Key – التكامل المرجعي – الفهارس Indexes.

7. اشرح أنواع العلاقات في النموذج العلائقي لقواعد البيانات.
8. عرف الكينونة وما المقصود بنوع الكينونة؟. ادعم إجابتك بالأمثلة المناسبة.
9. عدد أنواع الواصفات Attributes مع الشرح وادعم إجابتك بأمثلة.
10. اشرح أنواع العلاقات التي تربط بين الكينونات. ادعم إجابتك بالأمثلة المناسبة.
11. ما المقصود بدرجة العلاقة لعدة كينونات؟ اشرح أنواعها وادعم إجابتك بالأمثلة المناسبة.
12. قارن بين قيد المشاركة الكلي وقيد المشاركة الجزئي للكينونات. ادعم إجابتك بالأمثلة المناسبة.
13. في شركة ما هناك مجموعة من الأقسام وكل قسم يديره موظف واحد، حيث لايجوز أن يديره أكثر من موظف. ارسم مخطط العلاقات الكينونية ER-Digram
14. ارسم مخطط الكيانات العلائقي للمسألة الأتية:

من أجل تحرير كتاب لدينا الكينونات الأتية:

1- الكتاب له رقم الكتاب، اسم الكتاب(عنوانه).
2- الكاتب له رقم الكاتب، اسم الكاتب.
3- المحرر له رقم المحرر، اسم المحرر.

4- المستودع له رقم المستودع، اسم المستودع.

5- لدينا أيضاً القواعد الأتية من أجل كل كتاب:

a. يمكن أن يُكتب من قبل أكثر من كاتب.

b. يُحرر من قبل أكثر من محرر، ولكن لمرة واحدة من أجل كل محرر.

c. يُخزن في أكثر من مستودع وذلك من أجل كل محرر.

d. كمية التخزين تكون بحسب المحرر وبحسب المستودع.

15. ارسم مخطط الكيانات العلائقي للمسألة الآتية:

1- محل لبيع وتأجير الأفلام، يقوم بتأجير أفلام إلى مشتركين في نظام للتأجير خاص بالمحل.

2- يقوم مبدأ التخزين لهذه الأفلام على أساس أن لكل فلم اسماً وكذلك رقم خاص به.

3- كل فلم في المحل يمكن أن يكون مخزناً على شكل VHS أو VCD أو DVD.

4- كل فلم في المحل مصنف على أساس إما كوميدي أو دعائي أو رومانسي أو ترجيدي.

5- كل مشترك له اسم ورقم تلفون خاص به ويحتفظ بها في المحل.

6- كل مشترك يقدم للمحل اسم الفلم المفضل إليه (يستخدم من قبل المحل من أجل التسويق).

7- هنالك نوعان من المشتركين:

- المشترك الذهبي: يُطلب منه بطاقة (Credit Card) ويستطيع أن يستأجر أكثر من فلم واحد في الوقت الواحد.
- المشترك البرونزي: لا يُطلب منه بطاقة (Credit Card) ويستطيع استئجار فلم واحد في الوقت الواحد.

8- يمكن أن يكون لكل مشترك أبناء حيث إن أسماءهم معروفة.

9- يُسمح لكل ابن استئجار فلم واحد في الوقت الواحد.

16. ارسم مخطط الكينونات العلائقية والمتعلق بتصنيف كتب داخل مكتبة عامة، حيث أن المكتبة تتعامل مع عدد كبير من المواضيع التي تتدرج أسفل مجموعة من المواضيع الفرعية. كل موضوع فرعي يندرج أسفله مجموعة من الكتب المؤلفة.

17. وضح الفرق بين الكينونة Entity ونوع الكينونة Entity Type ومجموعة الكينونات Entity Set.

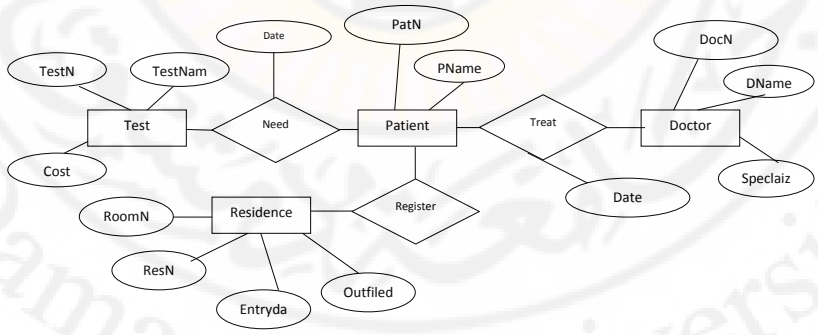
18. ما هو الكيان الضعيف؟ What Is Weak Entity ؟

19. صمم مخطط كيان– ارتباط من أجل ما يأتي:

مشفى يحتاج إلى الاحتفاظ بمعلومات حول مرضاه، وأطبائه، والفحوصات المتعددة التي أُجريت على المرضى. يريد المشفى أن يكون قادراً على إيجاد الطبيب المسؤول عن كل مريض وأن يعرف الفحوصات التي أجريت لكل مريض ومَن مِن الأطباء أمر بإجرائها معاً، مع أخذ الفرضيات الآتية بعين الاعتبار:

- لكل مريض طبيب و طبيب واحد فقط.
- يمكن أن يُشرف الطبيب على أكثر من مريض.
- بإمكان الطبيب أن يأمر بإجراء فحص لأي مريض حتى لو لم يكن يُشرف عليه.

20. أوجد النموذج العلائقي الموافق لنموذج الكينونات المبين في الشكل الآتي:



21. اختر الإجابة الصحيحة:

1– العلاقة بين كينونتين تكون من نوع:

a. علاقة ثنائية
b. علاقة أحادية
c. علاقة ثلاثية
d. لا أعرف

2- أي من الواصفات الآتية يمكن أن تكون واصفة (خاصية) مركبة؟
a. الطالب
b. رقم الطالب
c. العمر
d. العنوان

3- أي من الواصفات الآتية يمكن أن تكون واصفة (خاصية) مشتقة؟
a. الطالب
b. رقم الطالب
c. العمر
d. العنوان

4- أي من الواصفات الآتية يمكن أن تكون مفتاحاً لعلاقة؟
a. الطالب
b. رقم الطالب
c. العمر
d. العنوان

5- تُحدد الكينونات والخصائص خلال عملية التصميم المنطقي
a. صح
b. خطأ

6- كل جدول في قاعدة البيانات يمثل
a. كينونة

b. خاصية
c. علاقة بين كينونتين
d. مخطط بيانات

7- المدير مسؤول عن مجموعة موظفين. يمثل هذا المفهوم من خلال علاقة من نوع:
a. واحد-إلى-واحد
b. واحد-إلى-متعدد
c. متعدد-إلى-متعدد
d. لا شيء مما سبق

8- أي من الآتية يعبر عن أكبر عدد من الكينونات والتي تشارك في علاقة ما؟
a. التقاطع
b. التجميع
c. درجة المشاركة
d. لا شيء

وقل رب زدني علما
جامعة دمشق
Damascus University

الفصل الثالث

الجبر العلائقي

1. مفهوم العلاقة

يمكننا اعتبار الجداول مجموعات عناصرها سجلات، وبالتالي العمليات الممكن إجراؤها على المجموعات يمكن إجراؤها أيضاً على العلاقات (الجداول). في الجبر العلائقي كل عملية تأخذ كمدخلات علاقة أو أكثر وتُعطي كنتيجة علاقة أخرى. تُقسيم العمليات التي يمكن تطبيقها على العلاقات إلى صنفين رئيسيين وهما العمليات الأحادية والعمليات الثنائية. العمليات الأحادية هي العمليات التي تُطبق على علاقة واحدة فقط وتضم الاختيار والإسقاط والإتمام. أما العمليات الثنائية فهي العمليات التي تُطبق على علاقتين وتضم الاجتماع والتقاطع والفرق والجداء والقسمة والربط. فيما يأتي سنقوم بدراسة هذه العمليات.

2. عمليات الجبر العلائقي

2.1. الاختيار (σ, Selection)

الاختيار هي عملية أحادية تقوم بحذف السجلات التي لا تحقق شرطاً معيناً. أي إن هذه العملية تقوم بإرجاع مجموعة جزئية من أسطر الجدول والتي تحقق شرطاً معيناً. يمكن أن يكون هذا الشرط شرطاً مركباً. مثلاً يمكن أن ندرس العلاقة Request(Request N°, Date, Price) الممثلة بالجدول الآتي:

Request N°	Date	Price
23	October	1765
54	October	2765
30	November	254
71	December	3437

إن القيام بعملية الاختيار على التاريخ يمكننا من الإجابة على السؤال الآتي "أعطي الطلبات التي تمت بعد شهر تشرين الثاني". نتائج هذا الاختيار يبينها الجدول الآتي:

Request N°	Date	Price
30	November	254
71	December	3437

2.2. الإسقاط (π, Projection)

الإسقاط هي عملية أحادية تقوم بحذف بعض واصفات علاقة ما. مثلاً في العلاقة Student(Student N°, Student Name, Department, Section) نجد السجلات الموافقة كما في الجدول الآتي:

Student N°	Student Name	Department	Section
346	Mhd	Mathematic	Informatics
234	Jhon	Chemistry	Enver
764	Loulou	Physics	Liser
231	Marc	Statistics	Stat
874	Jhon	Mathematic	Analysis

الإسقاط على الاسم والقسم يعطينا العلاقة Relation 1. أما الإسقاط على القسم والشعبة فيعطينا العلاقة Relation 2 كما تبين الجداول الآتية:

Relation 1

Student Name	Department
Mhd	Mathematic
Jhon	Chemistry
Loulou	Physics
Marc	Statistics
Jhon	Mathematic

Relation 2

Student Name	Section
Mhd	Informatics
Jhon	Enver
Loulou	Liser
Marc	Stat
Jhon	Analysis

2.3. الإتمام (Complement)

الإتمام هي عملية أحادية تقوم ببناء علاقة جديدة تتضمن كافة السجلات الممكنة والتي قيمها غير موجودة في العلاقة الأصلية. أي إن العلاقة الناتجة تمثل نفي العلاقة الأولى.

Teaching Relation

Professor	Student
Alice	Susan
Paul	Jim
Paul	Susan
Alice	Smith
Jennifer	Jim

→

Non Teaching Relation

Professor	Student
Paul	Smith
Alice	Jim
Jennifer	Susan
Jennifer	Smith

2.4. الاجتماع (Union,∪)

الاجتماع هي عملية ثنائية تمكننا من دمج علاقتين في علاقة واحدة. هذه العملية لا يمكن تطبيقها إلا من أجل علاقتين تملكان الواصفات نفسها. بالتالي عملية الاجتماع هي عملية تبديلية تجمع الأسطر كافة من الجدولين دون تكرار. من أجل تطبيق هذه العملية يجب أن يتطابق الجدولان في ترتيب وأنماط الأعمدة.

Worker Relation

Employee N°	Name
15	Jim
17	Alice
23	Lin

Framework Relation

Employee N°	Name
34	Joujou
65	Sophi

→

Employer Relation

N° Employee	Name
15	Jim
17	Alice
23	Lin
34	Joujou
65	Sophi

مثلاً باستخدام الاجتماع يمكننا دمج العلاقة

Worker(Employee N°, Name)

و العلاقة

Framework(Employee N°, Name)

للحصول على العلاقة

Employer(Employee N°, Name).

2.5. التقاطع (∩,Intersection)

التقاطع هي عملية ثنائية تمكننا من الحصول على السجلات المشتركة بين العلاقتين معاً. هذه العملية لا يمكن تطبيقها إلا من أجل علاقتين يتفقان في ترتيب وأنماط الأعمدة (الواصفات) وهي عملية تبديلية. أي إنه ينتج عن هذه العملية جدولاً يضمّ الأسطر المشتركة بين الجدولين الأساسيين، ويجب أن يكون الجدولان المطبق عليهما هذه العملية منسجمان من حيث عدد الأعمدة وترتيبها وأنماطها. مثلاً من الممكن إيجاد التقاطع بين العلاقة Engineer(Employee N°, Name) والعلاقة Chef of Service(Employee N°, Name) للحصول على العلاقة:

Chef of Service Engineer(Employee N°, Name)

Engineer Relation

Employee N°	Name
10	Loulou
12	Jojo
13	Alice
15	Jim

Chef of Service Relation

Employee N°	Name
13	Alice
14	Sophi
15	Jim

→

Chef of Service Engineer Relation

Employee N°	Name
13	Alice
15	Jim

2.6. الفرق (−,Difference)

الفرق هي عملية ثنائية ينتج عنها جدولاً يتضمن الأسطر التي تظهر في الجدول الأول ولا تظهر في الجدول الثاني. من أجل إتمام هذه العملية يجب أن يمتلك الجدولان الأعمدة نفسها من حيث الترتيب والنمط. هذه العملية غير تبديلية.

مثلاً حساب الفرق بين العلاقة

Registration(Name, Subject)

والعلاقة

Receive(Name, Subject)

يعطينا العلاقة

Refuse(Name, Subject)

Registration Relation

Name	Subject
Lin	Math
Loulou	Physic
Alice	Physic
Jim	Math
Loulou	Chemic

Receive Relation

Name	Subject
Lin	Math
Loulou	Physic
Jim	Math

→

Refuse Relation

Name	Subject
Alice	Physic
Loulou	Chemic

2.7. القسمة (/ ,Division)

القسمة هي عملية ثنائية ينتج عنها جدولاً بقيم أعمدة مرافقاتها من أعمدة الجدول الأول موجود كأسطر في الجدول الثاني. هذه العملية غير تبديلية.

مثلاً عملية القسمة تمكننا من الإجابة على السؤال الآتي "ما هي أسماء المدرسين الذين يقومون بتدريس الطالبين Jim و Loulou معاً". الجداول الآتية تبين أن القسمة بين العلاقة Teaching والعلاقة Student يعطينا العلاقة Conjoin Teaching.

Teaching Relation

Student Name	Professor Name
Jim	Ann
Alice	Paul
Loulou	Jhon
Jim	Jhon
Loulou	Ann

Student Relation

Student Name
Jim
Loulou

→

Conjoin Teaching Relation

Professor Name
Ann
Jhon

2.8. الجداء (Product, ×)

الجُداء هو عملية ثنائية ينتج عنها أزواج السجلات الممكنة كافة من كلا الجدولين من خلال إجراء ما يشبه عملية الجداء الديكارتي.

مثلاً جداء العلاقة

Book(Title, Author)

والعلاقة

Presentation(Color, Edition)

يعطينا العلاقة

Bookpresentation(Title, Author, Color, Edition)

Book Relation

Title	Author
X	Alice
Y	Paul

Presentation Relation

Color	Edition
Red	Ieee
Blue	Lncs

→

Bookpresentation Relation

Title	Author	Color	Edition
X	Alice	Red	Ieee
X	Alice	Blue	Lncs
Y	Paul	Red	Ieee
Y	Paul	Blue	Lncs

2.9. الربط (Join)

الربط هو أكثر العمليات فائدة وأكثرها استخداماً في الجبر العلائقي لأنه الطريقة الأساسية لاسترجاع البيانات من عدة جداول. تسمح هذه العملية بربط جداول مستقلة عن بعضها من خلال واصفات متشابهة. يمكن التعبير عن الربط بتركيب عملية جُداء

متبوعة بعملية اختيار، حيث نحسب الجداء ثم نختار فقط السجلات التي تحقق شرطاً منطقياً معيناً. يوجد عدة أنواع من الربط ندرجها في الفقرات الآتية.

2.9.1. الربط المشروط

(Conditional Join,θ)

يُعدّ الربط المشروط الشكل الأكثر استخداماً للربط وهو ربط بين علاقتين بالإضافة إلى شرط ما ($\neq$,=, <, >). شرط الربط أن يكون مطابقاً للشرط المستخدم في جملة الاختيار. يسمى الربط المشروط أيضاً بـ Theta-Product.

انطلاقاً من العلاقة

Employee(Employee Name, Employee Salary)

والعلاقة

Chef(Chef Name, Chef Salary)

يمكننا الإجابة على السؤال الآتي "من هم الموظفون الذين يتقاضون رواتبً أعلى من راتب مدير".

إذا اعتبرنا العلاقتين:

Employee Relation

Employee Name	Employee Salary
E1	20000
E2	15000
E3	10000

Chef Relation

Chef Name	Chef Salary
Jhon	17000
Rimi	13000

في المرحلة الأولى نقوم بتطبيق عملية الجداء فنحصل على الجدول الآتي:

Employee Name	Employee Salary	Chef Name	Chef Salary
E1	20000	Jhon	17000
E2	15000	Jhon	17000
E3	10000	Jhon	17000
E1	20000	Rimi	13000
E2	15000	Rimi	13000
E3	10000	Rimi	13000

في المرحلة الثانية نقوم باختيار السجلات الموافقين للموظفين الذين يتقاضون راتباً أعلى من راتب مدير. أي تلك التي تحقق الشرط الآتي:

Salary Employee Salary > Chef

Earn More Relation

Employee Name	Employee Salary	Chef Name	Chef Salary
E1	20000	Jhon	17000
E1	20000	Rimi	13000
E2	15000	Rimi	13000

يوجد نوع خاص من الربط المشروط يُدعى الربط المتساوي (Equi Join). يقتصر هذا النوع على علاقة المساواة فقط.

2.9.2. الربط الطبيعي(⋈ ,Natural Join)

الربط الطبيعي هو حالة خاصة من حالات الربط المشروط حينما تكون جميع الحقول التي تحمل الأسماء نفسها في الجداول المشاركة في العملية بينها إشارة مساواة. في هذه الحالة نسمي عملية الربط بالربط الطبيعي. أي بدون شرط للربط لأن الشرط مفهوم ضمنياً وهو أن تكون جميع الحقول التي تحمل نفس الاسم من كلا الجدولين بينها إشارة مساواة. تظهر هذه الحقول مرة واحدة في النتيجة. أي إن الربط الطبيعي هو عبارة عن تركيب لثلاث عمليات: جُداء وإسقاط واختيار في آن واحد. غالباً ما تكون الحقول المشاركة في الربط الطبيعي مفاتيح رئيسية في أحد الجداول ومفاتيح أجنبية (ثانوية) في الجدول الآخر، ولكن هذا ليس شرطاً لازماً لاستخدام الربط الطبيعي.

Employee Relation

N°	Name	Service N°
23	Toto	17
25	Loulou	15
35	Jojo	17
23	Toto	14

Service Relation

Service N°	Service Name
14	Bay
15	Inf
17	Comp
28	Sell

نتيجة الربط الطبيعي على الشكل الآتي:

Employee&Sevice Relation

N°	Name	Service N°	Service Name
23	Toto	17	Comp
25	Loulou	15	Inf
35	Jojo	17	Comp
23	Toto	14	Bay

نلاحظ من الجدول السابق أن الخدمة رقم 28 (Sell) والتي ليس لها مقابل في العلاقة الأولى ليست موجودة في العلاقة الناتجة عن عملية الربط. يُعد الربط الطبيعي من أهم عمليات الجبر العلائقي.

2.9.3. الربط الخارجي (⟕ Outer Join)

لاحظنا في الربط الطبيعي أنه في بعض الأحيان لا يكون لجميع الأسطر في أحد الجداول ما يُربَط معه في الجدول المقابل. أي إن قيم حقول الربط في أحد الجداول ليس لها قيمة مقابلة تحقق شرط الربط الطبيعي في الجدول المقابل. قد نحتاج إلى إظهار تلك الأسطر التي ليس لها ما يُربط بها في إحدى العلاقتين أو كليهما. يمكن الوصول إلى هذه النتيجة باستخدام الربط الخارجي. الفائدة من الربط الخارجي هي المحافظة على المعلومات التي يمكن ألّا تظهر في الأنواع الأخرى من الربط.

يسمى الربط الخارجي من اليمين أو من اليسار حسب تموضع العلاقة المراد استرجاع كامل أسطرها بالنسبة لإشارة العملية. فعلى سبيل المثال الربط الخارجي من اليسار هو الربط الذي تظهر في نتيجته الأسطر من العلاقة الأولى التي لا يوجد لها ما يُربط معها في العلاقة الثانية بالإضافة إلى باقي الأسطر التي تحقق شرط الربط.

يمكن أن نُعرف الربط الخارجي الكامل Full Outer Join حينما نريد استرجاع جميع أسطر كلا العلاقتين كما في المثال الآتي:

Employee Relation

N°	Name	Office N°
23	Toto	17
25	Loulou	15
35	Jojo	22
23	Toto	14

Office Relation

Office N°	Building
14	A1
42	B2
17	A1
62	C7

نتيجة الربط الخارجي على الشكل الآتي:

Employee&Office Relation– Outer Join

N°	Name	Office N°	Building
23	Toto	17	A1
25	Loulou	15	⊥
35	Jojo	22	⊥
23	Toto	14	A1
⊥	⊥	42	B2
⊥	⊥	62	C7

2.9.4. الربط الجزئي (⋉ Semi Join ⋊)

الربط الجزئي يُعطينا جميع السجلات الموجودة في العلاقة الأولى (بما فيها السجلات التي ليس لها مقابل) دون أن نُظهر حقول العلاقة الثانية التي ليس لها مقابل من العلاقة الأولى. و بالتالي عملية الربط الجزئي هي عملية ربط خارجي من اليمين أو من اليسار، فهي إذن عملية غير تبديلية.

نتيجة الربط الجزئي على الشكل الآتي:

Employee&Office Relation– Semi Join

N°	Name	Office N°	Building
23	Toto	17	A1
25	Loulou	15	⊥
35	Jojo	22	⊥
23	Toto	14	A1

Damascus University

أسئلة البحث:

1. عرف ما يأتي: الإسقاط، الإتمام، الفرق، القسمة، الربط الخارجي.
2. بين الفائدة من الربط الذاتي.
3. بين الفرق بين الربط الطبيعي والربط المتساوي.
4. ليكن لدينا العلاقتان R,S المجاورتان. المطلوب أوجد ما يأتي:

- $\sigma_{A=C}(R \times S)$
- $\sigma_{D=10*B}(R \times S)$
- $\Pi_{C,E}(S)$

R	
A	B
X	1
Y	2

S		
C	D	E
X	10	U
Y	10	U
Y	30	V
Z	10	V

5. ماهي العمليات الأحادية في الجبر العلائقي .
6. ماهي العمليات الثنائية في الجبر العلائقي .

الفصل الرابع
تنظيم قاعدة البيانات

1. مفهوم التنظيم (Normalization)

يُقصد بالتنظيم (أوالتسوية Normalization) تصميم جداول قاعدة البيانات بحيث نتحكم بتكرار البيانات ونتجنب حالات الشذوذ التي يمكن أن تنتج عن عمليات الإضافة والحذف والتعديل على تلك البيانات.

سنقوم بدايةً بدراسة بعض الإشكاليات التي يُعاني منها التصميم السيئ لقاعدة البيانات، ومن ثم سندرس مستويات التنظيم وخصائص كل منها.

1.1. إشكاليات التصميم السيئ لقاعدة البيانات

يوجد هناك مشاكل أساسية ينبغي علينا تجنبها خلال عملية تصميم قاعدة البيانات.

من هذه المشاكل:

1. التكرار غير المفيد في تخزين البيانات (Redundancy Anomaly)
2. االإضافة الخاطئة(Insertion Anomaly)
3. الحذف الخاطئ (Deletion Anomaly)
4. التعديل الخاطئ (Update Anomaly)

في الواقع إن استخدام طريقة منظمة لتصميم جداول قاعدة البيانات يقلل من هذه المشاكل إلى حد كبير. سنتناول بعض الأمثلة التي توضح الشذوذ التي يمكن أن تؤثر على تصميم قاعدة البيانات وبعض المشاكل المحتمل حدوثها.

مثال1: لنفرض لدينا الجدول الآتي:

Name	Project	Profession	Salary	Budget
Mhd	DB	Assistant	7000	10000
Samer	UML	Designer	1000	20000

Samer	XML	Designer	1000	60000
Ali	XML	Manager	3000	60000
Ali	UML	Consulter	3000	20000
Ali	DB	Consulter	3000	10000
Sami	WEB	Employee	4000	3300
Majd	DB	Manager	500	10000
Majd	XML	Designer	500	60000
Yasser	XML	Designer	15000	60000
Yasser	DB	Designer	15000	10000

المشاكل التي يمكن أن نلاحظها في الجدول يمكن أن نُبيّنها فيما يلي.

1. التكرار غير المفيد:

- إن المعلومات المتعلقة بالراتب لكل موظف مكررة عدة مرات بلا فائدة.
- إن المعلومات المتعلقة بالميزانية لكل مشروع مكررة عدة مرات بلا فائدة.
- قيم الحقل الاسم (Name) مكررة وبالتالي لا يمكن اعتباره مفتاح للعلاقة.
- أيضاً قيم الحقل المشروع (Project) مكرر وبالتالي لا يمكن اعتباره مفتاح.
- إن المفتاح لهذا الجدول هو التركيب (Name, Project).

2. الإضافة الخاطئة:

ولما كان المفتاح هو (Name, Project) فإنه ليس بالإمكان إدراج موظف بدون تحديد المشروع الذي يعمل فيه لأنه لا يجوز أن يكون المفتاح عديم القيمة (Null). كذلك لا يمكن إدراج مشروع بدون تحديد على الأقل اسم موظف يعمل في هذا المشروع.

3. الحذف الخاطئ:

بفرض أن موظفاً ما قد أنهى عمله في الشركة أو أن موظفاً ما قد تمّ إعفاؤه من الخدمة في مشروع. بفرض أن هذا الموظف هو الموظف الوحيد في المشروع (Sami). إن عملية حذف سجل ذلك الموظف ستؤدي إلى فقدان معلومات عن المشروع الذي يعمل فيه.

4. التحديث الخاطئ:

إن التكرار في تخزين البيانات سيؤدي إلى تكرار في تحديث تلك البيانات. فمثلاً في الجدول السابق هناك تكرار في تخزين الراتب لكل موظف. إذا ما أردنا تحديث راتب موظف ما فإن ذلك يتطلب تحديثه في كل سجلات الجدول. إن هذا يقلل من فعّالية قاعدة البيانات. بالطريقة نفسها فإن تحديث ميزانية مشروع ما سيؤدي إلى المشكلة نفسها.

مثال2: بفرض أن شركة ما تضم مجموعة من مسؤولي المبيعات الذين يعملون في عدة مواقع. ترغب الإدارة بتخزين البيانات المتعلقة بموظفيها وبيانات مستودعاتها في قاعدة بيانات. المحاولة الأولى لبناء هذه القاعدة يمكن أن تكون وفق الجدول الآتي:

ID	Name	Address	Role	Store N°	Store Address	Phon Store
ID 1	Jane	Add1	Sales	S1	Sa1	011-1234567
ID 2	Fred	Add 2	Sales	S1	Sa1	011-1234567
ID 3	Edwar	Add 3	Manager	S1	Sa1	011-1234567
ID 4	Ann	Add 4	Sales	S2	Sa2	021-7891234
ID 5	Jhone	Add 5	Sales	S2	Sa2	021-7891234
ID 6	Smith	Add 6	Manager	S2	Sa2	021-7891234

يتضمن الجدول السابق بيانات الموظفين والمستودعات، ومن الواضح وجود تكرار غير مفيد في تلك البيانات نبينها فيما يأتي:

- إذا أردنا إدخال بيانات موظف جديد فيجب أيضاً إدخال بيانات المستودع الذي يعمل فيه، وللحفاظ على عدم تناقض البيانات في القاعدة يجب إدخال بيانات المستودع بدقة وبشكل مطابق للقيم المدخلة سابقاً (لأحد الموظفين السابقين في المستودع نفسه)، هنا تظهر مشكلة تكرار البيانات بالإضافة طبعاً إلى حجم التخزين المهدور.

- بفرض أننا نريد إدخال بيانات مستودع جديد قبل توظيف أحد فيه، هذا يتطلب إدخال قيم Null في بيانات الموظف ومن ضمنها الحقل المفتاح (ID) وهذا يخرق شرط تكامل البيانات (Integrity). هذا ما يدعى بإشكالية الإدخال (Anomaly Insertion).
- بفرض أننا حذفنا سجل الموظف الأخير في أحد المستودعات، سيؤدي ذلك إلى فقدان بيانات مستودع موجود في الشركة. هذا يرتبط بإشكالية الحذف (Deletion Anomaly).
- إذا أردنا تغيير رقم هاتف أحد المستودعات، عندها يجب تغيير رقم الهاتف في كل سجلات موظفي ذلك المستودع، فإذا تم تغيير جزء من هذه السجلات دون جزء آخر سيؤدي ذلك إلى بيانات متناقضة في قاعدة البيانات، وهذا ما يرتبط بإشكالية التعديل (Update Anomaly).

هذه الأخطاء والمشاكل عادةً لا تظهر إذا قام مصمم قاعدة البيانات باتباع الطرق الصحيحة في عملية تصميم جداول تلك القاعدة مع العلم أنه يمكن أن تظهر هذه المشاكل عند عدم تحديد المتطلبات المختلفة للنظام بشكل صحيح وكامل. يمكن مبدئياً حل المشاكل السابقة من خلال تقسيم الجدول السابق إلى جدولين، أحدهما يحمل بيانات الموظفين ويستورد مفتاح الجدول الثاني الذي يحمل بيانات المستودعات كما يأتي:

ID	Name	Address	Role	Store N°
ID1	Jane	Add1	Sales	S1
ID 2	Fred	Add 2	Sales	S1
ID 3	Ed	Add 3	Manager	S1
ID 4	Ann	Add 4	Sales	S2
ID 5	Jhone	Add 5	Sales	S2
ID 6	Smith	Add 6	Manager	S2

Store N°	Store Address	Store Phone
S1	Sa1	011-1234567
S2	Sa2	021-7891234

الآن عند إدخال بيانات موظف جديد، يتم تسجيل رقم المستودع الذي يعمل فيه، أما بقية بيانات المستودع فهي مخزنة في جدول مستقل وبالتالي لن تظهر بيانات متناقضة في قاعدة البيانات، وإذا أردنا إدخال بيانات مستودع جديد فيمكن إدخالها بغض النظر عمن يعمل في المستودع، وهذا يُجنبنا إشكالية الإدخال. وعند حذف الموظف الأخير في مستودع، تبقى بيانات تلك المستودع في جدول مستقل. وعند تعديل رقم هاتف مستودع ما، يتم التعديل على سجل واحد في جدول المستودعات.

تُعد عملية تقسيم الجدول السابقة من عمليات تنظيم قاعدة البيانات، التي سنتعرف عليها بتفصيل أكثر، ولكن بعد عرض مجموعة من المفاهيم الأساسية في هذا الخصوص من خلال الفقرات اللاحقة.

1.2. التبعية الوظيفية

(Functional Dependency, FD)

يُقال عن الواصفة X إنها تُحدد (Determine) الواصفة Y أو إن Y تعتمد اعتماداً وظيفياً (أو نقول إنه يوجد ارتباطاً تابعياً) على X إذا كانت كل قيمة من X تُحدد قيمة واحدة فقط من Y، ويُرمز لذلك بالتبعية الوظيفية الآتية $X \rightarrow Y$. يمكن التعبير عن ذلك رياضياً كما يأتي:

إذا كان هناك سطران T1 وT2 في مجموعة الأسطر R وكان T1.X=T2.X وأن $T1,T2 \in R$ فإن T1.Y=T2.Y.

لنتأمل العلاقة الموضحة في الجدول الآتي:

EmpN°	EmpName	EmpAdress
112	Samer	EA1
652	Salem	EA2
214	Hadi	EA3
432	Salem	EA4

نلا حظ من الجدول أن رقم الموظف (EmpN°) يُحدد اسمه (EmpName)، وكل قيمة لـ EmpN° تقابلها قيمة وحيدة لـ EmpName. لذلك يُقال إن EmpName تتبع وظيفياً لـ EmpN°، أو يُقال إن EmpN° تُحدد EmpName. نقول إن هناك ارتبطاً وظيفياً بين EmpN° و EmpName لأنه لا يمكننا أبداً إدراج الزوج $< 214, Sami >$ بدون حذف الزوج $< 214, Hadi >$. بالمقابل لا يوجد ارتباط وظيفي بين اسم الموظف (EmpName) وعنوان الموظف (EmpAdress). لذلك نكتب $EmpName - / \rightarrow EmpAdress$.

مثال آخر: إذا كان لدينا جدول يمثل طلاب الجامعة ويحتوي على خاصية رقم الطالب (Number) وكذلك اسمه (Name) ومعدله (Average) كما في الجدول الآتي:

Number	Name	Average
10611205	Samer	67
10711311	Majd	75
10854325	Sami	67
10612312	Hamd	83
10723141	Salem	75

نلاحظ من الجدول السابق أن الواصفة رقم الطالب تحدد الواصفة اسم الطالب لأن كل قيمة من الواصفة رقم الطالب تحدد قيمة واحدة من الواصفة اسم الطالب ونكتب $Number \rightarrow Name$. بينما لو نظرنا إلى المعدل فإنه لا يحدد رقم الطالب كما أنه لا

يحدد اسم الطالب وذلك لأنه من الممكن أن نجد قيمة للمعدل مُرتبطة بأكثر من قيمة من رقم الطالب وكذلك من اسم الطالب. أي إنه من الممكن أن يكون نفس المعدل لأكثر من طالب.

1.2.1. التبعية الوظيفية والمفاتيح

انطلاقاً من مفهوم التبعية الوظيفية ومن تعريف مفتاح العلاقة الذي هو واصفة أو أكثر تحدد صف واحد داخل الجدول (بدون تكرار)، يمكن أن نستنتج أن المفتاح هو الواصفة (أو مجموعة الواصفات) التي نستطيع من خلالها تحديد وظيفياً الواصفات الأخرى الموجودة كافة في الجدول. أي إن المفتاح هو واصفة (أو أكثر) من واصفات العلاقة بحيث تكون مصدراً لجميع التبعيات الوظيفية اللواتي هدفها باقي واصفات تلك العلاقة.

1.2.2. الإعتمادات الوظيفية التامة والاعتمادات الوظيفية المباشرة

نقول إن الواصفة Y تعتمد وظيفياً بشكل تام (أو كلي) على واصفة مركبة (Fully Functional Dependency)، إذا كانت تلك الواصفة المركبة تُحدد Y، و Y لا تعتمد وظيفياً على جزء منها. أي إن الاعتماد الوظيفي $A,B,C \to D$ هو اعتماد وظيفي تام إذا كان $A,B -/\to D$ و $A,C -/\to D$ و $B,C -/\to D$ و $A -/\to D$ و $B -/\to D$ و $C -/\to D$.

نقول عن الاعتماد الوظيفي $A \to B$ إنه اعتماد وظيفي مباشر إذا تعذر وجود واصفة أخرى C تحقق $A \to C$ و $C \to B$.

1.2.3. بديهيات الارتباطات الوظيفية

لتكن لدينا الواصفات X, Y, Z, U, V في الجدول العلائقي R. يمكن بسهولة برهان صفات الارتباطات الوظيفية الآتية:

1. الإنعكاسية (Reflexivity): إذا كانت الواصفة X مجموعة جزئية من الواصفة Y في جدول معين ($X \subseteq Y$) فإن X تعتمد اعتماداً وظيفياً على Y ويمكن كتابة ذلك بالشكل $Y \to X$.

2. الازدياد (Augmentation): إذا كانت $X \to Y$ وكان U, V واصفتين ما في R بحيث $U \subseteq V$ فإن $X, V \to Y, U$.

3. المتعدية (Transitivity): إذا كان الارتباط الوظيفي $X \to Y$ وكذلك الارتباط الوظيفي $Y \to Z$ فإن $X \to Z$.

4. قاعدة شبه التعدي (Pseudo-Transitivity): إذا كانت الاعتماد الوظيفي $X \to Y$ وكذلك الاعتماد الوظيفي $Y, U \to Z$ فإن $X, U \to Z$.

5. قاعدة الاجتماع (Union): إذا كانت الاعتمادات الوظيفية $X \to Y$ و $X \to Z$ فإن $X \to Y, Z$.

6. قاعدة التحليل (Decomposition): إذا كان الارتباط الوظيفي $X \to Y, Z$ فإن $X \to Y$ و $X \to Z$.

1.3. مستويات التنظيم

كما سبق وعرفنا، التنظيم بأنه عملية تسوية وتنظيم لجداول قاعدة البيانات العلائقية بهدف التقليل من تكرار البيانات وتقليص حجم التخزين المطلوب والحد من إشكاليات الإدخال والحذف والتعديل على القاعدة.

بشكل عام يتم تنظيم قاعدة البيانات من خلال إخضاعها لمجموعة اختبارات وتعديل هذه القاعدة لتحقق مجموعة معايير. من أجل ذلك نعرف مجموعة من مستويات التنظيم.

مستويات التنظيم الأساسية هي ثلاث:

- الشكل النظامي الأول

(First Normal Form, 1NF)

- الشكل النظامي الثاني

(Second Normal Form, 2NF)

- الشكل النظامي الثالث

(Third Normal Form, 3NF)

يُضاف لهذه المستويات الثلاثة الشكل المقترح من قبل R.Boyce والذي يُطلق عليه BCNF (Boys Cood Normal Form)، والشكلين النظامي الرابع (Forth Normal Form, 4NF) والشكل النظامي الخامس (Fifth Normal Form, 5NF). فيما يأتي سنقوم بدراسة هذه الأشكال.

1.3.1. الشكل النظامي الأول

(First Normal Form, 1NF)

يُقال عن جدول في قاعدة البيانات إنه من الشكل النظامي الأول إذا كانت كل واصفة لا يمكن أن تظهر أكثر من مرة في العلاقة، وكذلك لا يمكن تقسيم أي واصفة إلى مجموعة من الواصفات الأخرى (بمعنى آخر لا تشكل بحد ذاتها علاقة). أي إذا كان تقاطع كل سطر وعمود في الجدول يتضمن قيمة وحيدة غير قابلة للتجزئة.

لنتأمل الجدول الآتي:

Employee

Employee Name	Children		Obtained Diplomas		
	First Name	Birth Day	Type	Exam	
				Place	Date
Sami	Jim	1975	Bac	Damascus	1999
	Loulou	1977	License	Aleppo	1994
	Sophi	1972	Engineer	Damascus	1993
Paul	Jhon	1973	License	Damascus	1999
	Rimi	1977	Doctorate	Homs	1991

نلاحظ من الجدول السابق أن الواصفة Children مقسمة إلى واصفتين First Name و Birth Day في نفس العلاقة. كذلك الواصفة Type من أجل Sami تظهر 3 مرات: Bac, License, Engineer فهذه العلاقة ليست في الشكل النظامي الأول (1NF).

كمثال على علاقة في الشكل النظامي الأول يمكن أن نورد العلاقة Request الآتية:

Request(Client N°,Product N°, Client Name, Product Name)

وضع العلاقة (أو الجدول) في الشكل النظامي الأول لا يحل مشاكل التصميم السيئ لقاعدة البيانات إنما هو خطوة في طريق حلها، فعلى الرغم من أن العلاقة السابقة في الشكل النظامي إلا أنها لا تزال تعاني من مشكلة تكرار البيانات، وإشكاليات الإضافة والتعديل والحذف. السبب في ذلك هو فقط الارتباط الوظيفي $\text{client N°, product N°} \rightarrow \text{client Name}$ وكذلك الارتباط الوظيفي $\text{client N°, product N°} \rightarrow \text{product Name}$ يمكن أخذهما بعين الاعتبار، وبالتالي يمكننا إدراج الأسطر الخاطئة (زبونين يحملان الرقم نفسه أو مادتين لهما الرقم نفسه) الآتية بدون أي مشكلة:

$$< 52, p3, Alice, chair >$$
$$< 52, p5, Sami, Table >$$
$$< 27, p3, Sophi, Table >$$

بهدف أخذ جميع الارتباطات الوظيفية بعين الاعتبار من المفيد إذاً تقسيم العلاقة السابقة إلى 3 علاقات وهي:

Client(client N°, client Name)

Product(product N°, product Name)

Request(client N°, product N°)

1.3.2. الشكل النظامي الثاني

(Second Normal Form, 2NF)

يُقال عن علاقة إنها من الشكل النظامي الثاني إذا حققت ما يأتي:

1. هي من الشكل النظامي الأول.

2. كل الارتباطات الوظيفية التي مصدرها مفتاح العلاقة وهدفها باقي واصفات العلاقة هي ارتباطات تامة. بمعنى آخر كل الواصفات التي لا تشكل جزءاً من المفتاح، تعتمد وظيفياً بشكل تام على تلك المفتاح.

يمكن أن نستنتج بسهولة أن كل علاقة من الشكل النظامي الأول ومفتاحها يقتصر على واصفة واحدة فقط هي من الشكل النظامي الثاني، لكن العكس غير صحيح بالضرورة.

لنأخذ مثلاً العلاقة Client الآتية:

Client(client N°, client Name, merchantman N°, merchantman Name)

هذه العلاقة أيضاً تُبرز مشكلة لأن الارتباط الوظيفي

merchantman N° → merchantman Name

لم يتم أخذه بعين الاعتبار وبالتالي يمكننا إدراج الأسطر الخاطئة الآتية (الرقم نفسه لأكثر من تاجر):

< 73, Jhon, m17, Loulou >

< 125, Sami, m17, Alice >

إذاً من المناسب تقسيم العلاقة السابقة إلى علاقتين على الشكل الآتي:

Client(client N°, clientName, merchantman N°)

Merchantman(merchantman N°, merchantman Name)

وضع العلاقة في الشكل النظامي الثاني يبدأ بتحديد جميع الارتباطات الوظيفية بين الواصفات، ومن ثم تجزئة العلاقة إلى مجموعة علاقات (Decomposition) تضمن اعتماد جميع الواصفات التي لا تشكل جزءاً من المفتاح الأساسي بشكل تام على تلك المفتاح.

1.3.3. الشكل النظامي الثالث

(Third Normal Form, 3NF)

تكون العلاقة في الشكل النظامي الثالث إذا حققت ما يأتي:

1. هذه العلاقة من الشكل النظامي الثاني.
2. جميع الارتباطات الوظيفية التي تخرج من المفتاح (مصدرها المفتاح) هي ارتباطات مباشرة. بمعنى آخر لا توجد في العلاقة واصفات لا تشكل جزءاً من المفتاح وتعتمد بالتعدّي على المفتاح.

لنأخذ مثلاً العلاقة:

Employee(Emp N°, Emp Name, Emp Adress)

وكذلك العلاقة:

Request(request N°, product N°, requered quantity, delivered quantity)

إن كلا العلاقتين في الشكل النظامي الثالث.

في حال كانت العلاقة في الشكل النظامي الثالث فإنه يكون لدينا:

1. لا يوجد أي ارتباط وظيفي مصدره جزء من مفتاح العلاقة.
2. لا يوجد أي ارتباط وظيفي بحيث يكون مصدره واصفة غير مفتاحيه وهدفه واصفة أخرى أيضاً غير مفتاحيه (ولم تكن جزءاً من المفتاح).

الشكل النظامي الثالث يحل إشكاليات التعديل والحذف والإضافة، بالإضافة إلى حل مشكلة تكرار البيانات. عندما يُطلب تنظيم قاعدة بيانات فيقصد بذلك عادةً وضعها في الشكل النظامي الثالث (على الأقل).

1.3.4. الشكل النظامي لـ Boyce Codd

الشكل النظامي لـ Boyce Codd هو حالة خاصة من الشكل النظامي الثالث. في الواقع وضع العلاقة في الشكل النظامي الثالث قد يُبقي على بعض المشاكل، ولاسيما عندما تكون واصفة غير مفتاحية مصدراً لارتباط وظيفي بحيث يكون هدفه جزءاً (مجموعة جزئية) من مفتاح العلاقة.

لنفرض مثلاً أنه لدينا في أحد المعاهد القيود الآتية:

- من أجل مادة معينة ومن أجل طالب معين لدينا علامة فصلية.
- من أجل مادة معينة ومن أجل طالب معين لدينا مدرس ما.

- كل مدرس يقوم بتدريس مادة واحدة فقط.

هذه القيود تُتَرجَم من خلال الارتباطات الوظيفية الآتية:

$$student, subject \rightarrow mark$$
$$student, subject \rightarrow professor$$
$$professor \rightarrow subject$$

وبالتالي لدينا العلاقة الآتية:

$$R(\underline{student, subject}, professor, mark)$$

من الواضح أن هذه العلاقة الأخيرة هي من الشكل النظامي الثالث، لكنها ليست من الشكل النظامي لـ Boyce Codd.

المشكلة تكمن في أنه من خلال العلاقة السابقة لا يمكننا أن نضمن حقيقية أن كل مدرس يقوم بتدريس مادة واحدة فقط. فمثلاً من الممكن إدراج الثلاثية الخاطئة $< Loulou, math, Jhon >$ وكذلك الثلاثية الخاطئة $< Lili, English, Jhon >$ ضمن القاعدة.

في الواقع هذه العلاقة يمكن تقسيمها على النحو الآتي:

لدينا $professor \rightarrow subject$ وكذلك $student, subject \rightarrow mark$ ومن قاعدة شبه التعدي يكون $student, professor \rightarrow mark$.

نتيجتاً لذلك أصبحت لدينا العلاقتان:

$$Result(\underline{student, professor}, mark)$$
$$Teaching(\underline{professor}, subject)$$

مع ذلك فإن الارتباطين الوظيفيين $student, subject \rightarrow professor$ و $student, subject \rightarrow mark$ لم يتم أخذهما بعين الاعتبار وذلك يقود أيضاً إلى نوع من الشذوذ الذي لا يستطيع نظام إدارة قواعد البيانات اكتشافه.

النتيجة: قمنا بتقسيم العلاقة الأساسية ولكننا لم نستطع تحقيق القيود المفروضة عليها كافة.

الآن نعرف الشكل النظامي لـ Boyce Codd بأنه الشكل الموافق لحقيقة كون مصدر كل ارتباط وظيفي موجود بين واصفات العلاقة هو مفتاح تلك العلاقة. وبشكل عام تكون العلاقة من الشكل النظامي لـ Boyce Codd إذا كان مصدر كل ارتباط وظيفي موجود بين واصفات العلاقة هو مفتاح تلك العلاقة.

الخلاصة هي أنه من أجل جميع الأشكال النظامية السابقة فإن كل ارتباط وظيفي يجب ترجمته بعلاقة بحيث يكون مفتاح هذه العلاقة هو مصدر تلك الارتباط الوظيفي.

1.3.5. الشكلين النظاميين الرابع والخامس

1.3.5.1. الارتباطات متعددة القيم

لقد تحدثنا عن مراحل التنظيم السابقة بدءاً من مرحلة التنظيم الأولى وحتى مرحلة التنظيم Boyce–Codd. إن هذه المراحل تعتمد بمفهومها على الارتباطات الوظيفية. المرحلة التي نحن بصدد دراستها في هذه الفقرة تعتمد على ما يُسمى بالارتباطات التعددية أو الارتباط متعدد القيم (Multi–Valued Dependency, MVD).

تحدث التبعية متعددة القيم (غير التافهة) في جدول عندما:

1. يكون لدينا على الأقل ثلاث واصفات.
2. عندما تُحدد قيمة حقل مجموعة قيم لحقول أخرى.

مثال:

في معهد ما لدينا طلاب يدرسون عدة مواد ويمارسون عدة أنواع من الرياضة كما يبين الجدول الآتي:

Student Relation

Student Name	Subject	Sport
Loulou	Mathematics	Tennis
Loulou	Physics	Ski
Loulou	Mathematics	Ski
Loulou	Physics	Tennis
Sami	English	Foot
Sami	Mathematics	Foot

الارتباطات {Student Name, Subject} و {Student Name, Sport} هي ارتباطات متعددة القيم. كل طالب يمكن أن يدرس عدة مواد ويمارس عدة ألعاب رياضية ونكتب $\text{student Name} \rightarrow\rightarrow \text{subject} / \text{sport}$.

في الواقع يوجد ارتباط متعدد القيم لأنه من أجل كل طالب Stud الذي يدرس المادة Sub والذي يمارس الرياضة Sp، فإن الثلاثية <Stud, Sub, Sp> موجودة في العلاقة.

رياضياً نُعّرف الارتباط متعدد القيم $A \rightarrow\rightarrow B/C$ وفق الصيغة الآتية $[A_i, B_j] \wedge [A_i, C_k] \Rightarrow [A_i, B_j, C_k]$. يمكن ترجمة هذه العلاقة في مثالنا من خلال الجملة الآتية: إذا كان لدينا طالب ما A_i ويدرس مادة ما B_j ويمارس رياضة ما C_k، عندئذٍ الثلاثية $[A_i, B_j, C_k]$ يجب أن تكون موجودة في العلاقة.

في الجدول السابق نلاحظ ما يأتي:

Loulou studies mathematics

Loulou plays ski

الثلاثية (Loulou, mathematics, ski) تظهر في العلاقة.

في الواقع يوجد ارتباط متعدد القيم لأنه لا يمكن إدراج الثلاثية (Sami, mathematics, foot) و (Sami, informatic, swimming) من دون وجود

بالضرورة كل من الثلاثيتين (Sami, mathematics, swimming) و (Sami, informatic, foot).

إن مفتاح العلاقة الممثلة بالجدول السابق هو الواصفات الثلاث Student Name, Subject, Sport مجتمعة.

1.3.5.2. قواعد الارتباطات متعددة القيم

لتكن لدينا الواصفات الآتية X, Y, Z, U, V في الجدول العلائقي. يمكن بسهولة برهان صفات الارتباطات متعددة القيم الآتية:

1. الإتمام (Complementation): إذا كان $X \rightarrow\rightarrow Y$ فإن $X \rightarrow\rightarrow Z$.
2. الانعكاسية (Reflexivity): إذا كان $Y \subseteq X$ عندئذٍ $X \rightarrow\rightarrow Y$.
3. الازدياد (Augmentation): إذا كان $U \subseteq V$ و $X \rightarrow\rightarrow Y$ فإن $X \cup V \rightarrow\rightarrow Y \cup U$.
4. المتعدية (Transitivity): إذا كان $X \rightarrow\rightarrow Y$ وكذلك $Y \rightarrow\rightarrow Z$ فإن $X \rightarrow\rightarrow Z$.
5. قاعدة شبه التعدي (Pseudo-Transitivity): إذا كان $X \rightarrow\rightarrow Y$ وكذلك $Y, U \rightarrow\rightarrow Z$ فإن $X, U \rightarrow\rightarrow Z$.
6. قاعدة الاجتماع (Union): إذا كان $X \rightarrow\rightarrow Y$ و $X \rightarrow\rightarrow Z$ فإن $X \rightarrow\rightarrow Y \cup Z$.
7. قاعدة التحليل (Decomposition): إذا كان $X \rightarrow\rightarrow Y \cap Z$ فإن $X \rightarrow\rightarrow Z$ و $X \rightarrow\rightarrow Y$.

1.3.5.3. الشكل النظامي الرابع

(Fourth Normal Form, 4NF)

إن مرحلة جعل العلاقة بالشكل النظامي الرابع تعتمد على مبدأ إزالة الارتباطات التعددية كافة. وبالتالي تكون العلاقة من الشكل النظامي الرابع إذا كانت لا تحوي على أي ارتباط متعدد القيم.

لنأخذ على سبيل المثال العلاقة السابقة:

$$\text{student}(\underline{\text{student name}, \text{subject}, \text{sport}})$$

من الواضح أن هذه العلاقة من الشكل النظامي لـ Boyce Codd والذي كنا قد وجدنا سابقاً أنها تحوي بعض المشاكل (مشاكل التكرار مثلاً). من أجل التخلص من هذه المشاكل (أو على الأقل التخفيف من المشاكل) قمنا بتعريف الشكل النظامي الرابع.

تكون العلاقة من الشكل النظامي الرابع إذا تحقق ما يأتي:

1. هي من الشكل النظامي لـ Boyce Codd.
2. لا تملك أي ارتباط متعدد القيم.

تقتصر جميع العلاقات من الشكل النظامي الرابع على مفتاح العلاقة حصراً.

العلاقة $\text{student}(\underline{\text{student name}, \text{subject}, \text{sport}})$ السابقة ليست من الشكل النظامي الرابع لأن $\text{student Name} \rightarrow\rightarrow \text{subject} / \text{sport}$. من المناسب إذاً تقسيم تلك العلاقة إلى علاقتين $R_1(\underline{\text{student Name}, \text{subject}})$ و $R_2(\underline{\text{student Name}, \text{sport}})$.

إن كلاً من R_1 و R_2 من الشكل النظامي الرابع.

1.3.5.4. الشكل النظامي الخامس

(Fifth Normal Form, 5NF)

1.3.5.4.1. الاعتماد الارتباطي

(Join Dependency)

يعتمد الشكل النظامي الخامس على ما يُسمى بالارتباط الدّمجي أو الاعتماد الارتباطي (Join Dependency, JD). نقول عن علاقة R إنها تحقق عملية الارتباط الدمجي (نرمز له بـ $\otimes$) إذا تحقق $R(A,B,C) = R_1(A,B) \otimes R_2(B,C)$، علماً أن:

1. كل من R_1 و R_2 يمثل إسقاطاً للعلاقة R.
2. ربط العلاقتين R_1 و R_2 يساوي العلاقة R.

بشكل عام يكون $R(A,B,C) = R_1(A,B) \otimes R_2(A,C) \otimes R_3(B,C)$ بالتالي الربط الدمجي ينتج عن ربط لعلاقات جديدة (جزئية) ناتجة عن عمليات إسقاط للعلاقات الأصلية على بعض الواصفات.

لنأخذ العلاقة الممثلة بالجدول الآتي:

Relation R

Student Name	Faculty	Sport	Sport Professor
Loulou	Low	Ski	Sophie
Fifi	Management	Tennis	Sylvie
Riri	Leterater	Ski	Sami
Jhon	Sciences	Swimming	Sylvie
Toto	Low	Tennis	Sylvie
Jojo	Management	Ski	Sophie

الإسقاط على الواصفات الثلاث الأولى من العلاقة R يعطينا العلاقة A ، والإسقاط على الواصفتين الأخيرتين يعطينا العلاقة B .

A B

Student Name	Faculty	Sport
Loulou	Low	Ski
Fifi	Management	Tennis
Riri	Leterater	Ski
Jhon	Sciences	Swimming
Toto	Low	Tennis
Jojo	Management	Ski

Sport	Sport Professor
Ski	Sophie
Tennis	Sylvie
Ski	Sami
Swimming	Sylvie

الآن الربط $S = A \otimes B$ يعطينا:

Student Name	Faculty	Sport	Sport Professor
Loulou	Low	Ski	Sophie
Loulou	Low	Ski	Sami
Fifi	Management	Tennis	Sylvie
……	…….	……..	……..

نلاحظ من الجدول السابق أن $S \neq R$ وعندها نقول إن الربط مع ضياع (بالمعلومات).

في الواقع الربط $S = A \otimes B$ يقودنا إلى معلومات خاطئة. ففي مثالنا نلاحظ أن لدينا معلومة تفيد أن الطالب Loulou يمارس الـ Ski مع المدرس Sami، في حين أن هذه المعلومة غير موجودة في العلاقة الأساسية R .

الآن لنتعرف على الشكل النظامي الخامس. تكون العلاقة R من الشكل النظامي الخامس إذا كانت من الشكل النظامي الرابع ولا تملك أي ارتباط دمجي (Join Dependency).

لنأخذ العلاقة Anti الآتية:

Illness(I)	Germ(G)	Antibiotic(A)
M1	G1	A2
M2	G2	A2
M3	G2	A2
M2	G2	A4
M3	G1	A3
M3	G1	A2

نلاحظ من الجدول السابق إنه لا يوجد ارتباط متعدد لأنه لدينا $<A_2,G_1,M_1>$ و $<A_2,G_2,M_2>$ ولكن لا يوجد $<A_2,G_2,M_1>$.

يوجد ارتباط دمجي (Join Dependency) لأن:

$$R(illness, germ, antibiotic) = R_1(illness, germ) \otimes R_2(germ, antibiotic) \otimes R_3(illness, germ)$$

أي إنه إذا كان الفيروس G يظهر في المرض I، وإذا كان المضاد A يصلح لمكافحة الفيروس G والمضاد A مناسباً للقضاء على المرض I فإن الثلاثية $<I,G,A>$ يجب أن تظهر في العلاقة. بشكل رياضي يمكن أن نكتب:

$$<I,G> \wedge <G,A> \wedge <I,A> \Rightarrow <I,G,A>$$

R_1

Illness	Germ
M1	G1
M2	G2
M3	G2
M3	G1

R_2

Germ	Antibiotic
G1	A2
G2	A2
G2	A4
G1	A3

R_3

Illness	Antibiotic
M1	A2
M2	A2
M3	A2
M2	A4
M3	A3

ربط العلاقتين R_1و R_2 يعطينا الجدول الآتي:

Illness	Germ	Antibiotic	
M1	G1	A2	
M1	G1	A3	*
M2	G2	A2	
M2	G2	A4	
M3	G2	A2	
M3	G2	A4	*
M3	G1	A2	
M3	G1	A3	

إن عملية ربط الجدول السابق مع الجدول الموافق للعلاقة R_3 تقوم بحذف السجلات الموافقة لرمز الـ * في الجدول السابق ويعطينا العلاقة الأصلية R . نلاحظ أن العلاقة المدروسة من الشكل النظامي الرابع، لكنها ليست من الشكل النظامي الخامس. من المناسب إذاً تقسيمها إلى ثلاث علاقات على الشكل الآتي:

$R_1(\text{Illness}, \text{Germ})$

$R_2(\text{Germ}, \text{Antibiotic})$

$R_3(\text{Illness}, \text{Antibiotic})$

كمثال أخر لنأخذ العلاقة الآتية:

ACP Relation

Agent	Company	Product
A1	C1	P1
A1	C1	P2
A1	C2	P1
A1	C2	P2
A2	C1	P1

نقوم أولاً بتقسيم هذه العلاقة كما يأتي:

R_1

Agent	Company
A1	C1
A1	C2
A2	C1

R_2

Agent	Product
A1	P1
A1	P2
A2	P1

R_3

Company	Product
C1	P1
C1	P2
C2	P1
C2	P2

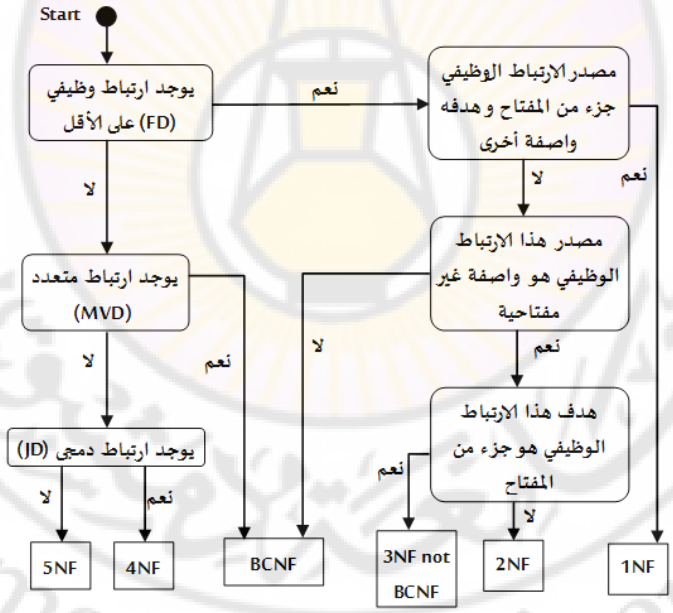
لنحسب الجُداء:

$$R_1(\text{Agent}, \text{Company}) \otimes R_2(\text{Agent}, \text{Product}) \otimes R_3(\text{Company}, \text{Product})$$

Agent	Company	Product
A1	C1	P1
A1	C1	P2
A1	C2	P1
A1	C2	P2
A2	C1	P1

نلاحظ أن جداء العلاقات الثلاث يُعطي العلاقة الأساسية.

بعد أن تعرفنا على الأشكال النظامية للعلاقة نبين في الشكل 1 مخططاً يساعدنا في تحديد الشكل النظامي المناسب لها.



الشكل1: خوارزمية تحديد الشكل النظامي لعلاقة ما

أسئلة البحث:

1. بيّن المقصود من تنظيم قاعدة البيانات وبين الفائدة منه.
2. اشرح أهم الإشكاليات التي يعاني منها التصميم السيء لقواعد البيانات.
3. بيّن الشكل النظامي لكل من العلاقات الآتية مع التعليل:
   A. $R_1(\underline{A, B})$
   B. $R_2(\underline{A}, B, C)$ علماً أن $B \rightarrow C$
   C. $R_3(\underline{A, B}, C, D)$ علماً أن $\begin{cases} A \rightarrow C \\ B \rightarrow D \end{cases}$
   D. $R_4(\underline{A, B, C})$ علماً أن $A \rightarrow\rightarrow B/C$
   E. $R_5(\underline{A, B}, C, D)$ علماً أن $C \rightarrow B$
4. بين مع التعليل الشكل النظامي للعلاقة $R(\underline{A, B}, C, D, E)$ في الحالات الآتية:
   A. $A \rightarrow B$
   B. $B \rightarrow C$
   C. $D \rightarrow E$
   D. $E \rightarrow B$
   E. من دون أي شرط.

الفصل الخامس

لغة الاستعلامات البنيوية (المهيكلة) SQL

1. مقدمة

تمكنك لغة (SQL) اختصاراً للكلمات (Structured Query Language) من إدارة قواعد البيانات بشكل كامل وإجراء جميع العمليات القياسية كإنشاء الجداول وتعبئتها بالبيانات، أو إجراء الاستعلامات عليها وكذلك الربط بين الجداول المختلفة، ولكن كيف يمكننا عمل ذلك بها؟

كما وضحنا سابقاً فإن قاعدة البيانات عبارة عن مكان أو مستودع كبير لتخزين البيانات المختلفة. تريحك قاعدة البيانات من عناء تخزين بياناتك في ملفات منفصلة وكما أنك تحتاج إلى إجراء عمليات البحث والتصنيف لهذه البيانات عن طريق خوارزميات البحث المعقدة التي تسهل لك ذلك، حيث إنها تعطيك واجهة سهلة للتعامل مع البيانات، وتكون البيانات في قاعدة البيانات مخزنة في عدة جداول Tables وكما نعلم فالجدول يتكون من صفوف Rows وأعمدة Columns هكذا:

الجدول السابق يتكون من ثلاثة صفوف وثلاثة أعمدة، وفي قواعد البيانات فإننا نسمي الصفوف بالسجلات Records ونسمي الأعمدة بالحقول Fields، فما هي الحقول وما هي السجلات ؟

يقوم الحقل الواحد في الجدول بتخزين معلومة معينة، فمثلاً عندما يكون لدينا قاعدة بيانات بأرقام الهواتف فإننا سنحتاج إلى حقلين (أو عامودين)، واحد للاسم، والثاني لرقم الهاتف. أما السجلات (أو الصفوف) فيحتوي كل منها على مجموعة من الحقول. لاحظ أن السجل الواحد في مثالنا يحتوي على معلومتين مختلفتين هما الاسم ورقم الهاتف، لذلك يمكننا القول بأن قاعدة البيانات تتكون من الجداول، والجداول تتكون من

السجلات والسجلات تتكون من الحقول، وكل حقل يحتوي على معلومة واحدة. أي إن الحقل هو أصغر وحدة قاعدة البيانات.

2. عبـارات SQL

تنقسم عبارات SQL إلى ثلاث فئات رئيسية وهي كالآتي:

2.1. لغة تعريف البيانات DDL

هي مجموعة من أوامر SQL نستطيع من خلالها إنشاء وتعريف الكائنات في قاعدة البيانات، وتقوم هذه الأوامر بإنشاء أو إسقاط أو تغيير كائن في قاعدة البيانات، وهي اختصار للكلمات (Data Definition Language). من أهم أوامرها Create, Alter, Drop, Rename, Truncate.

2.2. لغة معالجة البيانات DML

هي مجموعة من العبارات التي نستطيع من خلالها معالجة البيانات الموجودة في قاعدة البيانات، من إدراج البيانات وتحديثها أو تحديد أو حذف البيانات، وهي اختصار للكلمات (Data Manipulation Language)، ومن أهم أوامرها Insert, Update, Delete.

2.3. لغة التحكم والصلاحيات DCL

هي مجموعة من أوامر SQL نستطيع من خلالها منح أو سحب امتيازات استخدام كائن في قاعدة البيانات، وهي اختصار للكلمات (Data Control Language)، ومن أهم أوامرها Grant, Revoke.

وإيجازاً للقول فإن الجدول الآتي يوضح الفئات الرئيسية لعبارات SQL:

Insert Update Delete	لغة معالجة البيانات DML
Create Alter Drop Rename Truncate	لغة تعريف البانات DDL
Grant Revoke	**لغة التحكم والصلاحيات**
Commit Rollback Savepoint	**لغة معالجة البيانات** **Transaction control**

3. جملة Select

الأمر Select يمكنك من استعادة البيانات من داخل قاعدة البيانات، وتأخذ جملة Select الشكل الآتي:

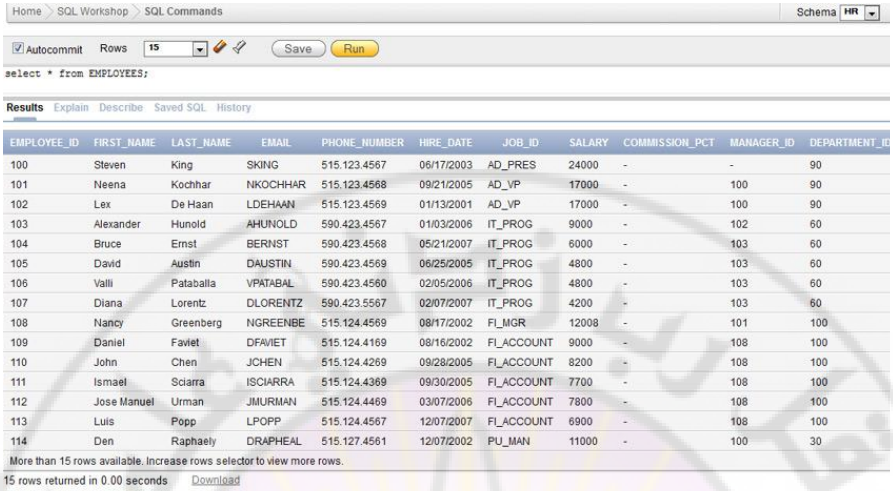
```
SELECT [DISTINCT] {*, column [alias],...}
FROM   table;
```

1) Select: تحدد ماهي الأعمدة المراد إظهارها.
2) From: تحدد المكان الذي نريد استعادة البيانات منه (جدول أو أكثر).
3) الفاصلة المنقوطة: تخبر SQL بأن العبارة كاملة وجاهزة للتنفيذ.
4) الأقواس [ ] تشير على أن ما بداخل هذه الأقواس اختيارياً.

3.1. اختيار الأعمدة كافة

تمثل عبارة * Select طريقة مختصرة لإخبار SQL بأننا نريد إظهار جميع أعمدة الجدول، فعلى سبيل المثال:

```
Select * from EMPLOYEES;
```


Home > SQL Workshop > SQL Commands

Schema HR

Autocommit Rows 15 Save Run

```
select * from EMPLOYEES;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

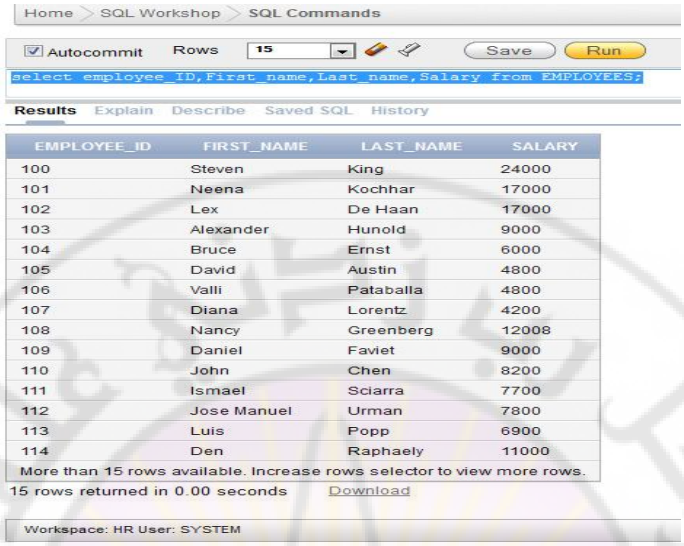
15 rows returned in 0.00 seconds Download

3.2. اختيار أعمدة محددة

يمكننا اختيار أعمدة محددة، وهو أمر في غاية السهولة حيث تقوم بكتابة أسماء الأعمدة بعد عبارة Select، مع وضع فاصلة بين أسماء الأعمدة، ثم تحدد الجدول بعد From.

مثال :

```
Select employee_ID, First_name, Last_name, Salary from EMPLOYEES;
```



- ✓ يمكن كتابة جملة SQL بالأحرف الكبيرة أو الصغيرة. أي إنها غير حساسة لحالة الأحرف
- ✓ يمكن كتابة جملة SQL في عدة أسطر، وتنتهي جملة Select بالفاصلة المنقوطة لكي تخبر SQL بأن الجملة كاملة وجاهزة للتنفيذ.
- ✓ تظهر الأعمدة المطلوبة بدءاً من اليسار إلى اليمين بترتيب كتابتها نفسه في عبارة Select

```
Select Department_Id , Department_name  from DEPARTMENTS;
```



3.3. التعابير الحسابية (Arithmetic Expressions)

ضع التعابير الحسابية بعد عبارة Select كأي عمود تريد استعادة بياناته، حيث تستخدم التعابير الحسابية مع تلك الأعمدة التي نوع معطياتها عددي مثل الأرقام والتاريخ.

يجب عليك الانتباه لأسبقية المعاملات(Operator Precedence) في التعابير الحسابية وفيما يأتي بيان بأسبقية المعاملات في التعابير الحسابية

1. ينفذ أولاً الضرب أو القسمة ، ومن اليسار إلى اليمين
2. بعد ذلك ينفذ الجمع أو الطرح لاحقاً، ومن اليسار إلى اليمين.

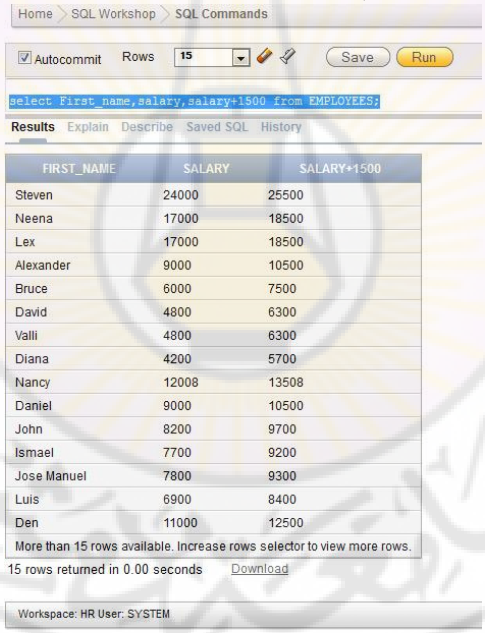
ملاحظة

✓ عند استخدام التعابير الحسابية يمكنك تغيير تسلسل أسبقية المعاملات الحسابية باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعابير الحسابية.

```
Select First_name, salary, salary+1500 from EMPLOYEES;
```

أمثلة على التعابير الحسابية

في هذا المثال نقوم باستخدام معامل الجمع لزيادة مرتبات جميع الموظفين بـ 1500، ثم نقوم بعرض الناتج في عمود جديد.




```
Select First_name, salary, 12*salary+1500 from EMPLOYEES;
```

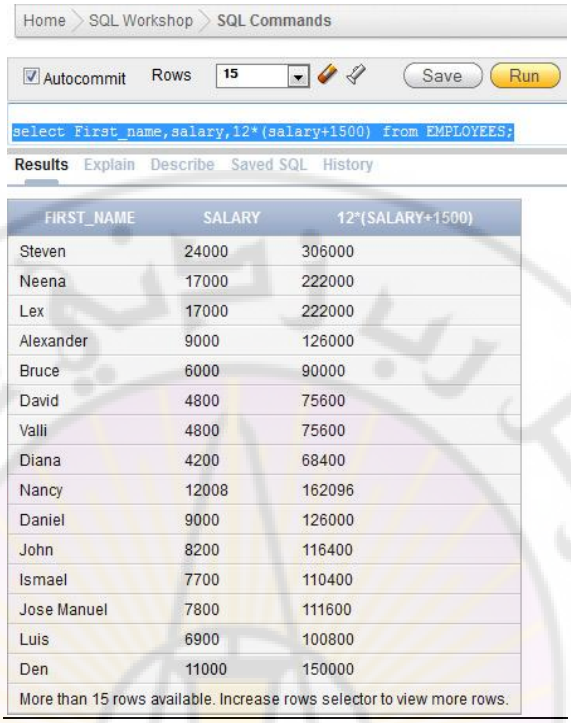
✓ تتجاهل SQL المسافات Blank Spaces قبل وبعد المعاملات الحسابية.

في هذا المثال سنقوم، طبقاً لأسبقية المعاملات، كما وضحنا سابقاً بحساب عملية الضرب أولاً وهي Salary *12، ثم إجراء عملية الجمع ثانياً وهي إضافة 1500.



- ✓ في المثال السابق كنا نريد زيادة مرتبات جميع الموظفين بـ 1500 ثم حساب المرتب السنوي بعد عملية الزيادة فكيف يمكننا ذلك؟
- ✓ كما تعلمنا سابقاً يمكنك تغيير تسلسل أسبقية المعاملات الحسابية باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعابير الحسابية، ويتم ذلك على النحو الآتي:

```
Select First_name, salary, 12*(salary+1500) from EMPLOYEES;
```



FIRST_NAME	SALARY	12*(SALARY+1500)
Steven	24000	306000
Neena	17000	222000
Lex	17000	222000
Alexander	9000	126000
Bruce	6000	90000
David	4800	75600
Valli	4800	75600
Diana	4200	68400
Nancy	12008	162096
Daniel	9000	126000
John	8200	116400
Ismael	7700	110400
Jose Manuel	7800	111600
Luis	6900	100800
Den	11000	150000
More than 15 rows available. Increase rows selector to view more rows.		

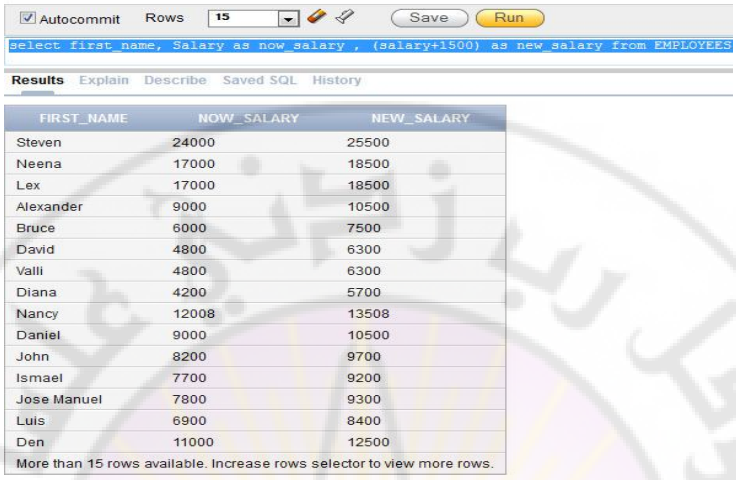
3.4. تعريف A Column Alias

يمكننا إعادة تسمية روؤس الأعمدة باستخدام A Column Alias، بمعنى إبدال الاسم الفعلي للحقل باسم اعتباري يتم تحديده بوساطة المستخدم ويكون ذلك مفيداً ولاسيما عند التعامل مع التعابير الحسابية، وفيما يأتي عدة اعتبارات لابد من مراعاتها عند إعادة تسمية روؤس الأعمدة:

1) يتم كتابة Alias في جملة Select يلي اسم العمود أو التعبير الحسابي المراد إعادة تسميته مع ترك مسافة.
2) يتم استخدام As بين اسم العمود والـ Alias، ويكون ذلك اختياري.
3) إذا احتوى Alias على مسافات أو أحرف وعلامات خاصة مثل (# or $)

```
Select first_name, Salary as now_salary , (salary+1500) as new_salary from
EMPLOYEES;
```

لابد من وضعه بين علامتي تنصيص (" ")Double Quotation.



FIRST_NAME	NOW_SALARY	NEW_SALARY
Steven	24000	25500
Neena	17000	18500
Lex	17000	18500
Alexander	9000	10500
Bruce	6000	7500
David	4800	6300
Valli	4800	6300
Diana	4200	5700
Nancy	12008	13508
Daniel	9000	10500
John	8200	9700
Ismael	7700	9200
Jose Manuel	7800	9300
Luis	6900	8400
Den	11000	12500

More than 15 rows available. Increase rows selector to view more rows.

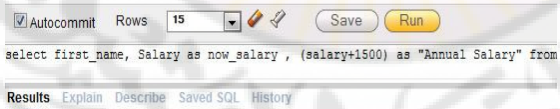
- ✓ في المثال السابق تم إعادة تسمية عمود Salary في جدول Employees بـ Now_Salary.

تم وضع كلمة As قبل Alias Name، وكما تعلمنا أن ذلك اختياري.

- ✓ يتم ظهور Alias Name بحالة أحرف كبيرة Uppercase.

مثال2:

```
Select First_name, salary,12*(salary+1500) as "Annual Salary" from EMPLOYEES;
```



FIRST_NAME	NOW_SALARY	Annual Salary
Steven	24000	25500
Neena	17000	18500
Lex	17000	18500
Alexander	9000	10500
Bruce	6000	7500
David	4800	6300
Valli	4800	6300
Diana	4200	5700
Nancy	12008	13508
Daniel	9000	10500
John	8200	9700

في هذا المثال ايضاً تم إعادة تسمية التعبير الحسابي Salary +1500 بـ Annual Salary، و كما نلاحظ تم وضع "Annual Salary" بين علامتي تنصيص لأنه يضم مسافات Spaces.

وهنا نلاحظ أنه تم إظهار العبارة كماهي دون تحويل إلى الأحرف الكبيرة.

يمكن استخدام أداة الربط || لدمج سلسلتي أحرف، حيث يكون ناتج الربط بين الحقول هو حقل واحد فقط. يمكن أن ندمج مع الحقول نصاً معيناً نضعه بين علامتي اقتباس ' ' كما يأتي:

```
Select ename, job, ename || ' is a ' || job as "Employees" from emp;
```

3.5. جملة الشرط Where Clause

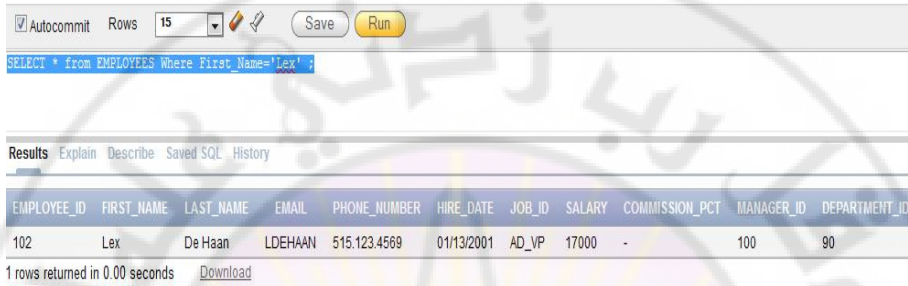
كان اختيارنا سابقاً للأعمدة يجري على الصفوف كافة التي يحتويها الجدول، أردنا إظهار صفوفاً تحتوي على قيم معينة، كإظهار الموظفين الذين تتجاوز مرتباتهم 3000 ليرة أو الذين يقطنون في مدينة ما، لفعل مثل ذلك سنحتاج إلى وضع كلمة Where ضمن عبارة Select، يأخذ الاختيار الشرطي الشكل الآتي:

```
SELECT      [DISTINCT] {*, column [alias], ...}
 FROM       table
[WHERE      condition(s)];
```

```
SELECT * from EMPLOYEES Where First_Name='Lex' ;
```

- ✓ تأتي جملة Where Clause بعد جملة From Clause.
- ✓ يلي Where شرطاً أوعدة شروط
- ✓ توجه عبارة Where الأمر إلى نظام إدارة قواعد البيانات أوراكل Oracle مثلاً بالبحث في قاعدة البيانات واستعادة الصفوف التي تتوافق مع شرط البحث.
- ✓ عند استخدام قيم محرفية أو قيم تعبر عن تاريخ يجب وضعها ضمن علامتي اقتباس أحادية ' '.

- ✓ في حال استخدام قيم محرفية يجب مراعاة حالة الأحرف صغيرة أم كبيرة.
- ✓ في حال استخدام قيم تعبر عن التاريخ يجب مراعاة صيغة (Format)



هذا التاريخ.

في هذا المثال نريد استرجاع جميع الأعمدة بجدول Employees، ولكن بشرط أن يكون اسم الموظف 'Lex'.

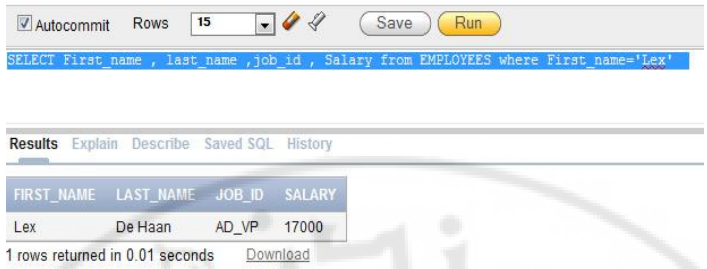
```
SELECT * from EMPLOYEES where HIRE_DATE='02/05/2006' ;
```



في هذا المثال نريد استرجاع جميع الأعمدة بجدول Employees، ولكن بشرط أن يكون تاريخ تعيين الموظف هو '02/05/2006'

- ✓ الأحرف النصية Character Strings، والتاريخ في جملة Where لابد من وضعها بين علامتي تنصيص (' ') Single Quotation.

```
SELECT First_name , last_name, job_id , Salary from EMPLOYEES where First_name='Lex'
```

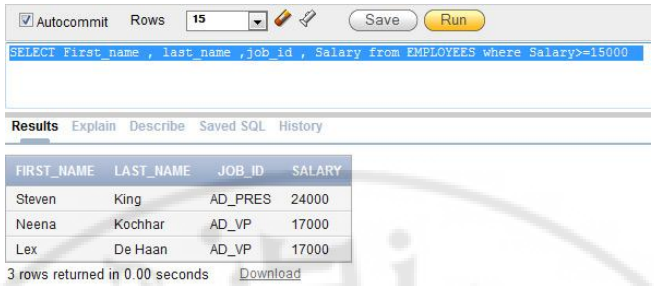


3.6. معاملات المقارنة الأساسية (Basic Comparison Operators)

تمنحنا SQL ستة عوامل للمقارنة نستطيع استخدامها مع Where، كما هو مبين في الجدول الآتي:

Operator المعامل	Meaning المعنى
=	يساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي
<> or ! =	لا يساوي

```
SELECT First_name , last_name, job_id, Salary from EMPLOYEES where Salary>=15000
```



في هذا المثال استخدمنا أحد معاملات المقارنة في جملة Where، حيث تم استعادة First_Name , Last_Name ,Job_Id , Salary عندما يكون Salary أكبر من أو يساوي 15000 ليرة .

مثال آخر:

```
SELECT First_name , last_name, job_id , Salary from EMPLOYEES where Salary<>15000
```

3.7. معاملات المقارنة الأخرى (Operator Comparison Other)

تتضمن SQL أربعة معاملات أخرى بالإضافة إلى المعاملات التي تم الإشارة إليها في الفقرة السابقة التي يمكن استخدامها مع جملة Where، كما هو مبين في الجدول الآتي:

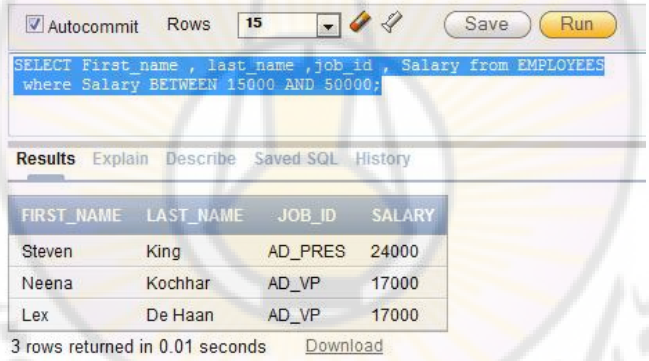
Operator المعامل	Meaning المعنى
Between..And...	بين قيمتين
In(List)	تتيح لك اختيار أي قيمة من List
Like {%, _}	يشابه أو يماثل
Is Null	فارغاً

3.7.1. المعامل Between

يمكنك إجراء عمليات البحث داخل نطاق معين من القيم وذلك باستخدام معامل Between مع جملة Where، فعلى سبيل المثل قد ترغب في استعادة بيانات جميع الموظفين الذين يتقاضون راتباً مابين 15000 و 50000 يمكنك القيام بهذا وذلك باستخدام العبارة الآتية:

```
SELECT First_name , last_name ,job_id , Salary from EMPLOYEES where Salary BETWEEN 15000 AND 50000;
```

في هذا المثال تم استخدام معامل Between مع جملة Where لاستعادة السجلات التي تقع ضمن نطاق معين من القيم.



3.7.2. المعامل In

يمكننا معامل In من القيام بعمليات البحث عن بنود داخل إحدى القوائم، حيث يمكنك استرجاع السجلات التي قيم عمود فيها (محدد في جملة Where) تساوي إحدى القيم المحددة ضمن مجموعة. على سبيل المثال لإظهار جميع السجلات الخاصة بالموظفين ذوي الأرقام 100 أو 110 أو 120 نقوم بكتابة العبارة الآتية:

```
SELECT * from EMPLOYEES where EMPLOYEE_ID In (100,110,120);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
120	Matthew	Weiss	MWEISS	650.123.1234	07/18/2004	ST_MAN	8000	-	100	50

3 rows returned in 0.04 seconds Download

في هذا المثال تم استرجاع السجلات التي فيها العمود Employee_Id يساوي إما 100 أو 110 أو 120، فنجد أن In تكافيء سلسلة مقارنات مربوطة بالمعامل or، حيث يمكننا الحصول على النتيجة نفسها في المثال السابق باستخدام المعامل or على النحو الآتي:

```
SELECT * from EMPLOYEES where EMPLOYEE_ID=100 or EMPLOYEE_ID=110 or
EMPLOYEE_ID=120
```

في هذا المثال سيتم استعادة جميع السجلات الخاصة بالموظفين الموجودين في القائمة التي تلي معامل In، حيث يتم استعادة جميع السجلات من جدول Employees عندما يكون First_Name يساوي إما'David' أو 'Nancy' أو'Lex' أو 'John'.

```
SELECT * from EMPLOYEES where First_name In
('Lex','David','Nancy','John')
```

فيما يأتي عدة اعتبارات لابد من مراعاتها عند استخدام معامل In مع جملة Where:

- ✓ وضع قيم المجموعة بين قوسين.
- ✓ فصل قيم المجموعة عن بعضها بفاصلة.
- ✓ يمكن وضع القيم بأي ترتيب.
- ✓ عندما تكون قيم المجموعة حرفية أو تاريخ لابد من وضعها بين علامتي تنصيص(' ').
- ✓ يمكن استخدام Not In لعكس الاختيار كما في العبارة الآتية:

```
SELECT * from EMPLOYEES where First_name not In ('Lex','David','Nancy','John')
```

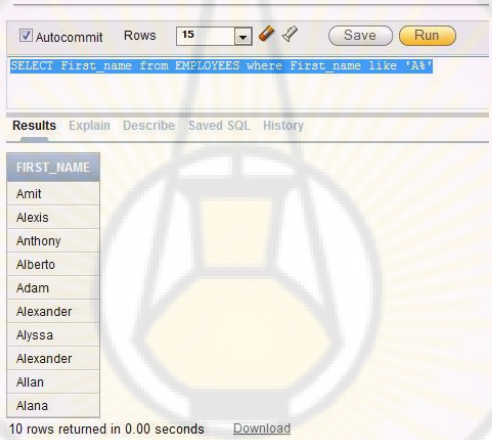
في هذه العبارة تمّ استعادة جميع السجلات من جدول Employees عندما يكون First_Name لا يساوي 'David' أو 'Nancy' أو 'Lex' أو 'John'.

3.7.3. المعامل Like

استخدام المعامل Like مع جملة Where يمكنك استعادة واسترجاع السجلات التي قيم عمود فيها تشابه سلسلة أحرف نبحث عنها. ليس هذا فحسب بل يمكننا من استرجاع السجلات التي تكون قيم عمود فيها مطابقة جزئياً لمجموعة أحرف نبحث عنها.

لاسترجاع جميع الأسماء التي تبدأ بحرف (A) يتم كتابة العبارة الآتية:

```
SELECT First_name from EMPLOYEES where First_name like 'A%'
```



للبحث عن الأسماء التي تشتمل على أحرف Ma نكتب العبارة الآتية:

```
SELECT * from EMPLOYEES where First_name like '%ma%'
```

لاسترجاع بيانات للموظفين الذي تم تعينهم خلال عام 2005 نكتب العبارة:

```
SELECT * from EMPLOYEES where HIRE_DATE like '%2005'
```

لاسترجاع جميع الأسماء التي يكون الحرف (A) فيها يأخذ الترتيب الثاني يتم كتابة العبارة الآتية:

```
SELECT First_name from EMPLOYEES where First_name like '_a%'
```



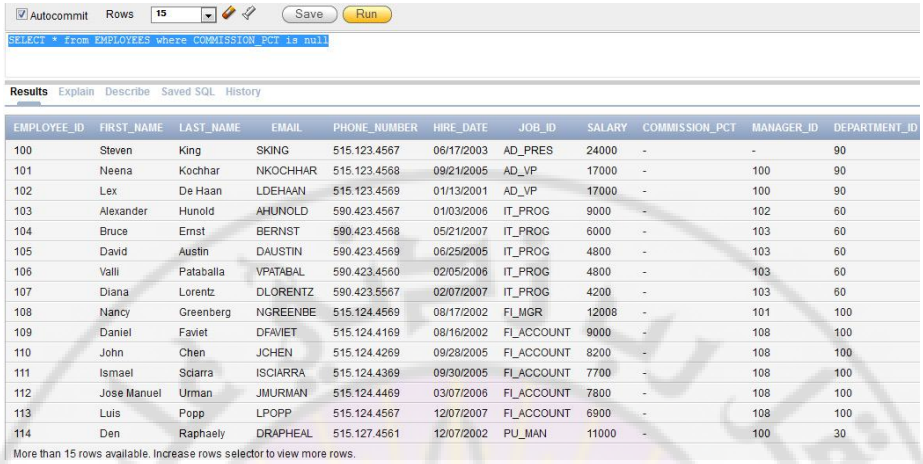
- ✓ تستخدم Like مع الأعمدة التي يكون نوع بياناتها حرفي.
- ✓ % اختصار لأي تتابع من سلسلة أحرف غير معينة مؤلفة من أي قيمة وبأي طول
- ✓ الرمز _ اختصار لأي حرف وحيد.

3.7.4. المعامل Is Null

قبل استخدام معامل Is Null مع Where لابد من الإجابة على ما هي القيمة Null ؟ تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات، حيث لا تعني القيمة Null صفر أو مسافة Blank Space .

لاسترجاع جميع بيانات الموظفين الذين لا يتقاضون حوافز Commission يتم كتابة الكود الآتي:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is null
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

لاسترجاع جميع بيانات الموظفين الذين يتقاضون حوافز Commission يتم كتابة الكود الآتي:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is not null
```

Damascus University

3.8. استخدام المعاملات المنطقية (Logical Operators)

بالإضافة إلى المعاملات التي تم الإشارة إليها سابقاً، تتضمن SQL ثلاث معاملات منطقية يمكن استخدامها مع جملة Where، كما هو مبين بالجدول الآتي:

Operator المعامل	Meaning المعنى
And	و
Or	أو
Not	النفي

3.8.1. المعامل and

يُستخدم المعامل and مع Where لاستعادة السجلات التي تحقق كافة الشروط المحددة في جملة Where، حيث يعني and أنه لاستعادة السجلات لابد من تحقق الشروط كافة.

لاسترجاع جميع بيانات الموظفين بالقسم It_Prog الذين يتقاضون مرتباً أكبر من 6000 نكتب العبارة الآتية:

```
SELECT * from EMPLOYEES where Salary > 6000 and JOB_ID='IT_PROG'
```



لاسترجاع جميع بيانات قسم 'It_Prog' الذين يتقاضون مرتباً أقل من 6000 نكتب العبارة الآتية:

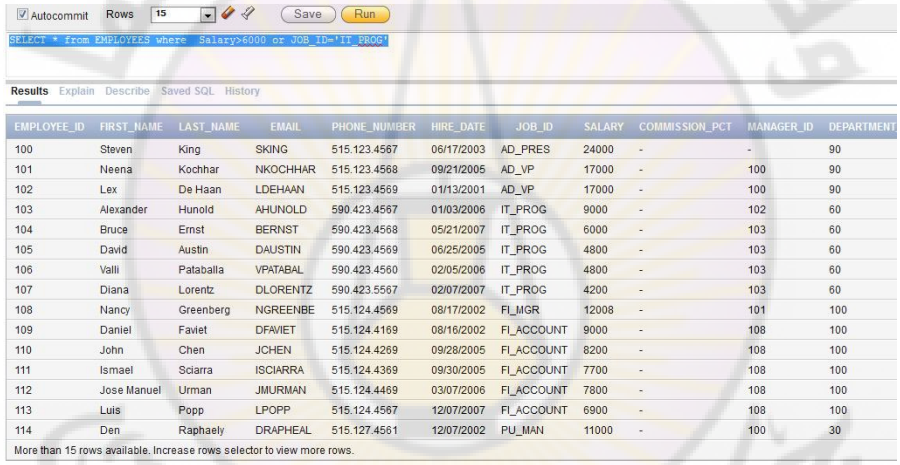
```
SELECT * from EMPLOYEES where  Salary <6000 and JOB_ID='IT_PROG'
```

3.8.2. المعامل or

نستخدم معامل or مع Where لاستعادة السجلات التي تحقق شرطاً واحداً من عدة شروط، حيث يعني المعامل or أن استعادة البيانات تتم إذا تحقق أي شرط من الشروط.

لاسترجاع بيانات الموظفين الذين يتقاضون مرتباً أكبر من 6000 ليرة أو يعملون بالوظيفة 'It_Prog'، نكتب العبارة الآتية:

```
SELECT * from EMPLOYEES where  Salary>6000 or JOB_ID='IT_PROG'
```



Autocommit Rows 15 Save Run

```
SELECT * from EMPLOYEES where  Salary>6000 or JOB_ID='IT_PROG'
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

✓ تم عرض بيانات الموظفين الذين يتقاضون مرتباً أكبر من 6000 أو يعملون في القسم It_Prog.

3.8.3. المعامل not

كما وجدنا سابقاً استخدام المعامل not يمكننا من استثناء سجلات معينة من اختيارنا، حيث يمكننا من استرجاع السجلات المعاكسة لاختيارنا، فعلى سبيل المثال لاختيار كافة السجلات من جدول Employees باستثناء التي يكون الـ Commission_Pct لها قيمته Null نكتب العبارة الآتية:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is not null
```

لاسترجاع بيانات الموظفين باستثناء الوظيفة Fi_Account, Pu_Man, Ad_Pres نكتب العبارة الآتية:

```
SELECT * FROM   EMPLOYEES
WHERE JOB_ID NOT IN ('AD_PRES', 'FI_ACCOUNT', 'PU_MAN');
```

- ✓ كما وجدنا سابقاً يمكن استخدم معامل Not مع المعاملات الأخرى مثل (Between, Like, Null)

- ✓ وإليك التلميح الآتي:

```
..... Where SALARY NOT BETWEEN 3000 AND 4000
..... Where FIRST_NAME NOT LIKE '%T%'
..... Where JOB_ID NOT IN('analyst','accountant')
..... Where COMMISSION_PCT IS NOT NULL
```

3.9. قواعد أسبقية المعاملات

Rules of Precedence

الجدول الآتي يوضح قواعد أسبقية المعاملات.

Order Evaluated الترتيب	Operator المعامل
1	جميع معاملات المقارنة
2	Not
3	and
4	Or

✓ كما تعلمنا سابقاً يمكننا تغيير تسلسل أسبقية المعاملات باستخدام الأقواس(Parentheses)، حيث تأخذ الأقواس الأسبقية في التعابير الحسابية

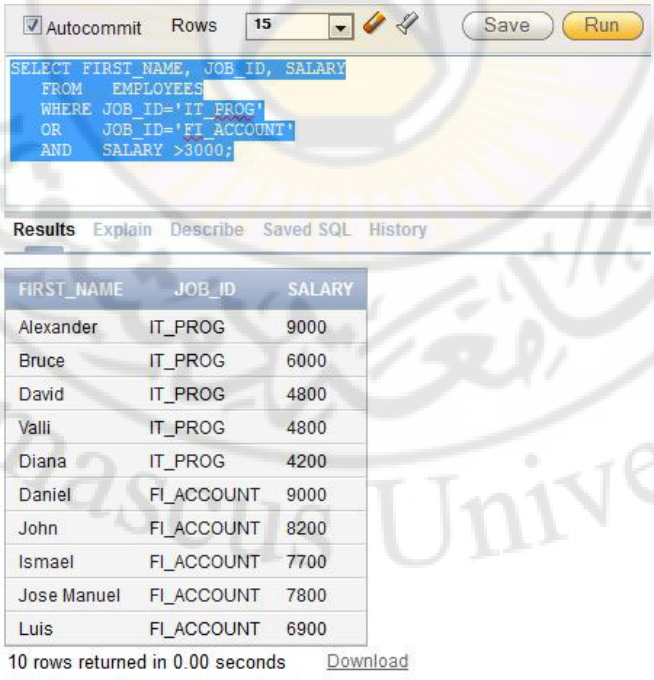
```
SELECT FIRST_NAME, JOB_ID, SALARY FROM    EMPLOYEES
  WHERE JOB_ID='IT_PROG'
  OR      JOB_ID='FI_ACCOUNT'
  AND    SALARY >3000;
```

✓ الشرط الأول هو

JOB_ID تكون'IT_PROG' و SALARY أكبر من 3000

✓ الشرط الثاني هو

JOB_ID تكون 'FI_ACCOUNT'



FIRST_NAME	JOB_ID	SALARY
Alexander	IT_PROG	9000
Bruce	IT_PROG	6000
David	IT_PROG	4800
Valli	IT_PROG	4800
Diana	IT_PROG	4200
Daniel	FI_ACCOUNT	9000
John	FI_ACCOUNT	8200
Ismael	FI_ACCOUNT	7700
Jose Manuel	FI_ACCOUNT	7800
Luis	FI_ACCOUNT	6900

10 rows returned in 0.00 seconds Download

3.10. استخدام عبارة Distinct لمنع تكرار السجلات

من أجل منع تكرار قيم الحقول نستخدم عبارة Distinct مباشرةً بعد كلمة Select كما في المثال الآتي :

```
SELECT Distinct deptno
From emp ;
```

3.11. ترتيب النتائج باستخدام Order by

يمكن ترتيب السجلات باستخدام Order By وفقاً لقيم العمود الذي نحدده ترتيباً تصاعدياً أو تنازلياً، ويمكن الترتيب بناءً على قيم عمود أو عدة أعمدة، وعند كتابة جملة Order By يجب اتباع الآتي:

- ✓ Order By تتبع جملة From{Table Name} مباشرةً في حالة عدم وجود Where {Condition(S)}
- ✓ Order By تتبع جملة Where {Condition(S)} مباشرةً.
- ✓ يوجد لدينا خياران عند استخدام Order By:

1) Order By {Column Name} Asc: يسمح هذا الخيار بالترتيب التصاعدي

2) Order By {Column Name} Desc: يسمح هذا الخيار بالترتيب التنازلي.

- ✓ يمكن الترتيب على أساس عدة أعمدة، ويجب الفصل بين أسماء الأعمدة بفاصلة (,).
- ✓ يوجد لدينا طريقتان لكتابة جملة Order By و هما:

1. عن طريق اسم العمود
2. عن طريق رقم العمود النسبي (موقعه في جملة Select)

الترتيب التصاعدي باستخدام Order By

لاستعادة بيانات جميع الموظفين مرتبة على أساس تاريخ التعيين، نكتب العبارة الآتية:

```
Select * from EMPLOYEES ORDER BY HIRE_DATE ASC
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
206	William	Gietz	WGIETZ	515.123.8181	06/07/2002	AC_ACCOUNT	8300	-	205	110
204	Hermann	Baer	HBAER	515.123.8888	06/07/2002	PR_REP	10000	-	101	70
203	Susan	Mavris	SMAVRIS	515.123.7777	06/07/2002	HR_REP	6500	-	101	40
205	Shelley	Higgins	SHIGGINS	515.123.8080	06/07/2002	AC_MGR	12008	-	101	110
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30
122	Payam	Kaufling	PKAUFLIN	650.123.3234	05/01/2003	ST_MAN	7900	-	100	50
115	Alexander	Khoo	AKHOO	515.127.4562	05/18/2003	PU_CLERK	3100	-	114	30
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
137	Renske	Ladwig	RLADWIG	650.121.1234	07/14/2003	ST_CLERK	3600	-	123	50
200	Jennifer	Whalen	JWHALEN	515.123.4444	09/17/2003	AD_ASST	4400	-	101	10
141	Trenna	Rajs	TRAJS	650.121.8009	10/17/2003	ST_CLERK	3500	-	124	50
184	Nandita	Sarchand	NSARCHAN	650.509.1876	01/27/2004	SH_CLERK	4200	-	121	50

More than 15 rows available. Increase rows selector to view more rows.

الترتيب التنازلي باستخدام Order By

لاستعادة بيانات جميع الموظفين مرتبة تنازلياً على أساس تاريخ التعيين، نكتب العبارة الآتية:

```
Select * from EMPLOYEES Order By HIRE_DATE DESC
```

ملاحظات:

- ✓ الترتيب الافتراضي في جملة Order By هو الترتيب التصاعدي مالم نصرح عكس ذلك بالكلمة Desc.
- ✓ يتم عرض القيم Null مؤخراً عند الترتيب التصاعدي، والعكس يتم ظهور القيم Null أولاً عند الترتيب التنازلي.
- ✓ يمكنك إجراء الترتيب باستخدام Order By على أساس قيم أعمدة ليست بجملة Select، كما في المثال الآتي:

```
Select FIRST_NAME ,SALARY  from EMPLOYEES ORDER BY HIRE_DATE
,SALARY DESC
```

FIRST_NAME	SALARY
Lex	17000
Shelley	12008
Hermann	10000
William	8300
Susan	6500
Daniel	9000
Nancy	12008
Den	11000
Payam	7900
Alexander	3100
Steven	24000
Renske	3600
Jennifer	4400
Trenna	3500
Nandita	4200
More than 15 rows available. Increase rows selector to view more rows.	

✓ يمكننا إجراء الترتيب باستخدام Order By على أساس قيم أكثر من عمود.

الترتيب باستخدام A Column Alias

يمكننا إجراء الترتيب في جملة Order By باستخدام A Column Alias كما في المثال الآتي:

```
Select FIRST_NAME ,SALARY*12 as total  from EMPLOYEES Order BY total DESC
```

FIRST_NAME	TOTAL
Steven	288000
Neena	204000
Lex	204000
John	168000
Karen	162000
Michael	156000
Nancy	144096
Shelley	144096
Alberto	144000
Lisa	138000
Den	132000
Gerald	132000
Ellen	132000
Eleni	126000
Clara	126000
More than 15 rows available. Increase rows selector to view more rows.	

3.12. القيمة Null وتأثيراتها

تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات بمعنى وجود قيم فارغة داخل الحقول ومن أشهر عيوب القيمة Null صعوبة إجراء أي عمليات حسابية على سجلات تحتوى بعضها على قيم فارغة.

استخدام القيمة Null

1) تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات، ولا تعني القيمة Null الصفر أو المسافة الفارغة Blank Space.
2) عند استخدام القيمة Null في التعابير الحسابية يكون الناتج Null.

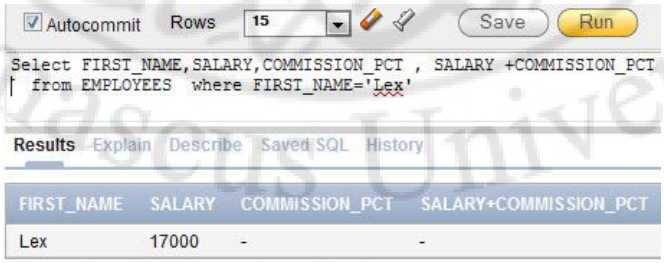
```
Select FIRST_NAME,SALARY,COMMISSION_PCT from EMPLOYEES
```

FIRST_NAME	SALARY	COMMISSION_PCT
Steven	24000	-
Neena	17000	-
Lex	17000	-
Alexander	9000	-
Bruce	6000	-
David	4800	-
Valli	4800	-
Diana	4200	-
Nancy	12008	-
Daniel	9000	-
John	8200	-
Ismael	7700	-
Jose Manuel	7800	-
Luis	6900	-
Den	11000	-

More than 15 rows available. Increase rows selector to view more rows.

في هذا المثال نجد أن العمود Commission_Pct يضم قيماً من النوع Null

```
Select FIRST_NAME,SALARY, COMMISSION_PCT , SALARY +COMMISSION_PCT
 from EMPLOYEES where FIRST_NAME='Lex'
```



في هذا المثال تم تناول العمليات الحسابية على القيم Null، حيث إن الـ Commission_Pct الخاص بالموظف 'Lex' يأخذ القيمة Null، وكما تعلمنا أن ناتج أي عملية حسابية طرفها القيمة Null يكون Null.

4. أنواع العلاقات

4.1. الربط بين جدولين أو أكثر

كما مر معنا سابقاً، كثيراً ما نحتاج إلى استرجاع بيانات من جدولين أو أكثر في استعلام واحد Query. إجراء مثل ذلك يتطلب منا إنشاء علاقة بين الجدولين أو الجداول، حيث تكمن القوة الحقيقية لقاعدة بيانات علائقية في قدرتها على استرجاع سجلات من جداول مختلفة في استعلام واحد. توجد لدينا العديد من الأسباب التي تدعونا إلى إنشاء علاقة بين الجداول منها:

1) دمج الأعمدة من جدولين أو أكثر.
2) اختيار أعمدة موجودة في جدول واحد بناءً على شرط يطبق على عمود أخر (Self Join).

4.2. استرجاع بيانات من جدولين أو أكثر

(Retrieving Data from Multiple Tables)

عند استرجاع بيانات من جدولين أو أكثر هناك عدة اعتبارات يجب مراعاتها:

1) وضع أسماء الأعمدة في جملة Select، على أن يتم الفصل بينها بالفاصلة.
2) وضع أسماء الجداول في جملة From، على أن يتم الفصل بينها بالفاصلة.
3) إضافة شرط الربط في جملة Where.
4) الصيغة العامة لاسترجاع بيانات من جدولين

Select	*table1.column, table2.column*
From	*table1, table2*
Where	*table1.column1 = table2.column2*;

```
Select  Emp.Empno, Emp.Ename, Dept.Dname
From    Emp, Dept
 Where Emp. Deptno=Dept.Deptno;
```

في هذا المثال تم استرجاع بيانات من جدولي Dept, Emp بينهما علاقة، حيث تم كتابة اسم الجدول قبل اسم العمود مع الفصل بينهما بنقطة في جملة Select، و كتابة الجداول مصدر البيانات في جملة From، مع كتابة شرط الربط في جملة Where.
في المثال السابق نجد أن أعمدة الربط لها الاسم Deptno نفسه، وهذا يسبب مشاكل مع نظام إدارة قواعد البيانات حيث لا يعرف لأي جدول تنتمي هذه الأعمدة ولذلك تم وضع اسم الجدول قبل اسم العمود، مع الفصل بينهما بنقطة.

4.3. أنواع الروابط

كما رأينا، في بعض الأحيان نرغب بعرض بيانات من أكثر من جدول. من أجل ذلك يجب علينا اجراء عملية ربط بين جدولين. سنقوم بدراسة أنواع مختلفة من الربط في الفقرات الآتية.

4.3.1. الرابطة المتكافئة Equijoin: لابد أن كل سجل في الجدول الأب يقابله سجل أو سجلات في الجدول الابن المرتبط به.

4.3.2. الرابطة غير المتكافئة Nonequijoin: لا تمتلك الجداول أعمدة وقيماً مشتركة.

4.3.3. الرابطة الخارجية Outerjoin: تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن السجلات المرتبطة يمكن أن تضم قيماً مشتركة ويمكن ألّا تضم ذلك، بمعنى أن الجدول الرئيس الأب يضم سجلاً أو سجلات لا يقابلها سجل أو سجلات في الجدول الابن المرتبط به.

4.3.4. الروابط الذاتية Self Join: يمكننا عمل رابطة ذاتية عند ربط الجدول مع نفسه.

4.3.5. الرابطة الديكارتية Cartesian Join: للحصول على الرابطة الديكارتية نقوم بحذف شرط الربط بين الجدولين في جملة Where، عندها سيقوم نظام إدارة قواعد البيانات بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني.

الرابطة المتكافئة Equijoin:

على سبيل المثال إذا أردنا تحديد اسم الإدارة لكل موظف فما علينا سوى المقارنة بين قيم deptno في كل من الجدولين emp و dept وفق الصيغة الآتية :

```
Select emp.empno, emp. ename, emp.deptno, Dept.deptno, dept.loc  From emp, dept
Where emp.deptno=dept.deptno
```

وهنا يكون الربط المتكافئ لكون كل سجل من جدول الموظفين emp يرتبط بسجل واحد أو أكثر من جدول الأقسام dept.

مع الملاحظة أن:

1. في هذا المثال نجد أن جدول Dept لا يوجد به إدارة ليس بها عاملاً في جدول Emp، حيث في هذا الاستعلام تم استرجاع البيانات لجميع الأقسام.
2. حيث إن عمود Deptno له الاسم نفسه في كلا الجدولين، تم وضع اسم الجدول قبل اسم العمود، مع الفصل بينهما بنقطة.
3. يطلق على العلاقة Equijoin أيضاً Simple Join أو Inner Join

الرابطة غير المتكافئة Nonequijoin

نادراً ما نحتاج إلى ربط سجلات من جدولين لا يمتلكان أعمدة و لا قيماً مشتركة. إن مثل تلك الرابطة تسمى رابطة غير متكافئة، كما في المثال الآتي:

Emp

EMPNO	ENAME	SAL
1	tarek	40000
2	ayman	28500
3	said	24500
4	ali	29980
5	ahmed	13500
6	helal	17000
7	ebrahim	16000

…

25 rows selected.

Salgrade

الراتب في جدول emp يقع بين أقل راتب و أكبر راتب في جدول SALGRADE

RADE	LOSAL	HISAL
1	8000	13000
2	13010	14000
3	14010	25000
4	25000	99990

- ✓ العلاقة بين الجدولين في عمود Sal في جدول Emp ينحصر بين عمودي Losal و Hisal في جدول Salgrade.

✓ في الروابط غير المتكافئة لا تستخدم معامل المساواة.

```
Select Emp.Ename, Emp.Sal, Salgrade.Grade
From   Emp, Salgrade
Where  Emp.Sal  Between  Salgrade.Losal and Salgrade.Hisal;
```

الرابطة الخارجية Outerjoin

تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن السجلات في الجداول المرتبطة يمكن أن تضم قيماً مشتركة ويمكن ألّا تضم. على سبيل المثال إذا احتوى جدول Departments على إدارة لا يوجد فيها عاملاً، معنى ذلك أن تلك الإدارة لن يتم اختيارها في استعلام يربط بين الجدولين. في الشكل التوضيحي الآتي نجد أنه لا يوجد موظفين في الإدارة Control And Credit

فإذا قمنا بكتابة العبارة الآتية:

```
Select  E.First_Name, D.Department_Id, D.Department_Name  From  Employees E,
Departments D
Where E.Department_Id = D.Department_Id
```

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
110	Accounting
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing
180	Construction
190	Contracting
200	Operations

FIRST_NAME	DEPARTMENT_ID
Steven	90
Neena	90
Lex	90
Alexander	60
Bruce	60
David	60
Valli	60
Diana	60
Nancy	100
Daniel	100
John	100
Ismael	100
Jose Manuel	100
Luis	100
Den	30
Alexander	30
Shelli	30
Sigal	30
Guy	30
Karen	30

نجد أن هناك سجلات لا تحقق شرط الربط:

(E.Department_Id = D.Department_Id)

بالتالي لن يتم استرجاعها في نتيجة الاستعلام، و حيث لا يوجد موظفون بالإدارة Credit Control And، وبالتالي لن تظهر في نتيجة الاستعلام طبقاً للـ Equijoin. لكي نعالج ذلك لابد من إنشاء رابطة خارجية وذلك بوضع إشارة الجمع (+) محاطة بقوسين وذلك بعد اسم محدد شرط الربط في جملة Where للجدول الذي يضّم السجلات التي لا يقابلها سجل أو سجلات في الجدول المرتبط.

```
Select  E.First_Name, D.Department_Id,
D.Department_Name From  Employees E,
Departments D Where E.Department_Id(+) =
D.Department_Id;
```

FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Jennifer	10	Administration
Pat	20	Marketing
Michael	20	Marketing
Sigal	30	Purchasing
Karen	30	Purchasing
Shelli	30	Purchasing
Den	30	Purchasing
Alexander	30	Purchasing
Guy	30	Purchasing
Susan	40	Human Resources
Kevin	50	Shipping
Jean	50	Shipping
Adam	50	Shipping
Timothy	50	Shipping
Ki	50	Shipping
Girard	50	Shipping

في هذا المثال القسم Control And Credit سيتم عرضها في نتيجة هذا الاستعلام، وذلك من خلال إنشاء رابطة خارجية بوضع إشارة الجمع (+) محاطة بقوسين بعد اسم محدد شرط الربط في جملة Where وذلك للجدول الذي يضّم السجلات التي لا يقابلها سجل أو سجلات في الجدول المرتبط وهو في مثالنا جدول Employees.

الرابطة الذاتية Selfjoin

يمكننا عمل الرابطة الذاتية عند ربط جدول ما مع نفسه، فعندما نريد ربط جدولاً مع نفسه. نكتب اسم الجدول نفسه عدة مرات في جملة Where، و بالتالي يتوجب علينا استخدام أسماء اعتبارية

في المثال الموضح بالشكل الآتي الجدول Employees يحتوي على عمود (Manager_Id) الذي يحدد رقم المدير. العمود Employee_Id يحدد رقم الموظف في جدول Employees الذي يعدّ المفتاح الرئيسي. لذلك توجد علاقة Selfjoin بين عمود Manager_Id وعمود Employee_Id.

مثال:

```
Select  Worker.First_Name,Manager.First_Name
From  Employees  Worker, Employees  Manager
 Where Worker.Manager_Id= Manager.Employee_Id
```

FIRST_NAME	FIRST_NAME
William	Gerald
Lisa	Gerald
Sundita	Gerald
Tayler	Gerald
Harrison	Gerald
Elizabeth	Gerald
Alexander	Lex
Clara	Alberto
Mattea	Alberto
David	Alberto
Danielle	Alberto
Amit	Alberto
Sundar	Alberto
Nandita	Adam
TJ	Adam
James	Adam

الرابطة الديكارتية Cartesian Joins

للحصول على الرابطة الديكارتية نقوم بحذف شرط الربط بين الجدولين في جملة Where، عندها سيقوم SQL بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني. هذا يجعل SQL يقوم بإنشاء ومعالجة كمية كبيرة جداً من البيانات لا قيمة لها، فعلى سبيل المثال إذا كان جدول Departments يضمّ 10 سجلات و جدول Employees يحتوي 30 سجل ،فإذا تم حذف شرط الربط من جملة Where في نتيجة الاستعلام يكون لدينا (10*30=300) سجل. لحساب عدد السجلات المسترجعة من رابطة ديكارتية للجدولين Employees, Departments يتم كتابة الاستعلام الآتي:

```
Select Count(*) From Departments, Employees;
```

تكون النتيجة Count(*)=300

4.4. استخدام أكثر من شرط في جملة Where

عند استرجاع بيانات من أكثر من جدول، يمكننا إضافة أكثر من شرط في جملة الشرط بعد شرط الربط، مثلاً لاسترجاع بيانات محللي النظم من رقم، اسم، الوظيفة، رقم الإدارة، موقع الإدارة، لابد من إضافة شرط في عبارة Where كما يأتي:

```
Select Empno, Ename, Emp.Job, Emp.Deptno, Loc
From Emp, Dept Where Emp.Deptno=Dept.Deptno
And Job='Analyst';
```

4.5. استخدام Table Alias في جملة Where

كما سبق وتعلمنا أنه يمكن استخدام اسم اعتباري للعمود، يمكننا أيضاً وضع اسم اعتباري للجدول. فعلى سبيل المثال يمكننا إجراء المثال السابق في جملة Where باستخدام الأسماء الاعتبارية للجداول Table Aliases كما في العبارة الآتية:

```
Select E.Empno, E.Ename, E.Deptno, D.Deptno, D.Loc
From Emp E, Dept D  Where E.Deptno=D.Deptno;
```

هناك عدة اعتبارات يجب مراعاتها عند استخدام الأسماء الاعتبارية:

1) الاسم الاعتباري للجدول لا يزيد عن 30 حرفاً.
2) لابد أن يستخدم الاسم الاعتباري أولاً في جملة Where.
3) يفضل أن يكون الاسم الاعتباري معبراً.

4.6. استرجاع بيانات من عدة جداول

(More than Two Tables)

في بعض الأحيان نريد استرجاع بيانات من أكثر من جدولين. على سبيل المثال نريد استرجاع بيانات الاسم ورقم الطلبية والأصناف وإجمالي الأصناف، إجمالي الطلبيات للعميل Tarek. يمكن استرجاع تلك البيانات من ثلاثة جداول هي Customer, Ord, Item بكتابة العبارة الآتية:

```
Select C.Name, O.Ord, I.Item, I.Itemtot, O.Total
From Customer C, Ord O, Item I Where C.Custid=O.Custid and O.Ordid=I.Ordid and
  C.Name='Tarek';
```

5. استخدام الدوال Functions

تمتلك SQL دوالاً Functions فعّالة و مفيدة عند استخدام عبارة Select. تنقسم هذه الدوال إلى نوعين:

- الدوال التجميعية لأكثر من صف(Multiple Row Functions): تقوم بالتعامل مع بيانات من أكثر من صف لإعطاء قيمة واحدة فقط.
- دوال الصف الواحد(Single Row Function): تقوم بالتعامل مع بيانات الصف الواحد لإعطاء قيمة وحيدة.

5.1. استخدام الدوال لأكثر من سطر Multiple Row Function

تزودنا SQL بعدة دوال تجميع Group Function يمكن أن تنفذ على مجموعة من السجلات وهي كما في الجدول الآتي:

الدالة	الاستخدام
Count	حساب عدد السجلات في جدول ما
Max	تحديد القيمة العظمى لعمود ما

Min	تحديد القيمة الدنيا لعمود ما
Avg	حساب المتوسط الحسابي لعمود ما
Stddev	حساب الانحراف المعياري لعمود ما
Sum	حساب المجموع الكلي لعمود ما
Variance	حساب التغاير لعمود ما

تقوم الدوال التجميعية Group Function بمعالجة قيم العمود المختار من الجدول وتقدم النتيجة في شكل قيمة وحيدة تخص العمود المختار. يجب علينا اتباع التعليمات الآتية عند استخدام Group Function :

1) وضع اسم العمود بين قوسين ( ) بعد الدالة مباشرة

```
Select Max(Salary)From Employees;
```

2) يمكن لجميع الأعمدة في جملة Select أن تكون كلها دوال تجميعية

```
Select Max(Salary), Min(Salary), Sum(Salary);
```

3) استخدام دالة تجميعية أخرى (أي تركيب الدوال التجميعية) غير مسموح به في SQL.

```
Select Max(Avg(Salary)) From Employees;
```

4) الدوال التجميعية تتجاهل القيم الفارغة Null values

5) يمكن استخدام الدوال التجميعية ضمن التعابير الحسابية.

5.1.1. استخدام Count

نستخدم الدالة Count لحساب عدد السجلات التي تضمّ قيماً غير فارغة (Not Null) في العمود المحدد في المجموعة. فمثلاً لحساب عدد السجلات في جدول Employees الذي يمتلك قيماً غير فارغة في العمود First_Name نكتب العبارة الآتية:

```
Select Count (First_Name) From Employees;
```

✓ تم وضع العمود المراد حساب عدد سجلاته بين قوسين بعد الدالة Count.

- ✓ لا يتم عد السجلات التي تضم قيماً فارغة Null.

5.1.2. استخدام Count(*)

عندما نستبدل اسم العمود بالنجمة (*) تقوم الدالة Count بحساب عدد السجلات بما فيها السجلات التي تحتوي على قيم فارغة (لماذا؟)، فعلى سبيل المثال:

```
Select   Count (*)
From     Employees
Where    Department_Id = 10;
```

5.1.2.1. استخدام Distinct مع Count

يمكننا استخدام كلمة Distinct مع Count، حيث تقوم باستبعاد السجلات المكررة. على سبيل المثال لحساب عدد السجلات في جدول Employees مع عدم استبعاد القيم الفارغة في العمود First_Name نكتب العبارة الآتية:

```
Select Count (Distinct (First_Name)) From Employees ;
```

5.1.3. استخدام الدالة Max

نستخدم الدالة Max لحساب أكبر قيمة في مجموعة ما، حيث يمكن استخدام الدالة Max مع القيم العددية أو النصية أو التاريخ والوقت. على سبيل المثال:

```
Select Max(Salary), Max(Hire_Date), Max(First_Name)
From Employees  ;
```

- ✓ يمكن استخدام Max مع أي نوع من البيانات.
- ✓ عند استخدام Max مع البيانات المحرفية فإنها تعيد أعلى قيمة Ascii، وعند استخدامها مع الأعمدة العددية تعيد أعلى قيمة جبرية، وعند استخدامها مع التاريخ والوقت تعيد القيمة الأحدث في العمود.
- ✓ تهمل الدالة Max القيم Null.

5.1.4. استخدام الدالة Min

```
SELECT MIN(SALARY)
FROM   EMPLOYEES
WHERE DEPARTMENT_ID =10;
```

تُعيد الدالة Min أصغر قيمة لا تساوي Null، كما أنها تُستخدم مع أي نوع من البيانات. على سبيل المثال نستخدم الدالة Min لتحديد أقل مرتب في القسم رقم 10.

5.1.5. استخدام الدالة Avg

نستخدم الدالة Avg لحساب المتوسط الحسابي لعمود ما، حيث تعيد Avg القيمة الوسطية لجميع القيم التي لا تساوي Null في العمود. هناك عدة اعتبارات يجب مراعاتها عند استخدام Avg :

```
SELECT      AVG(SALARY)  FROM   EMPLOYEES;
```

- ✓ يستخدم Avg مع القيم العددية فقط.
- ✓ يهمل Avg القيمة Null عند حسابه للقيمة الوسطى.
- ✓ يهمل Avg القيم المتكررة في العمود عند حسابه للقيمة الوسطية وذلك إذا صرحنا بكتابة Distinct قبل اسم العمود.

يمكننا حساب المتوسط الحسابي لعمود SALARY في جدول EMPLOYEES، وذلك بكتابة العبارة الآتية:

```
SELECT      MAX(SALARY), MIN(SALARY), AVG(SALARY) FROM
        EMPLOYEES;
```

يمكننا حساب أعلى قيمة وأدنى قيمة والمتوسط الحسابي لعمود SALARY بكتابة العبارة الآتية:

يمكننا حساب المتوسط الحسابي لعمود Commission_Pct في جدول Employees ، وذلك بكتابة العبارة الآتية:

```
Select   Avg(Commission_Pct)                    From
Employees
```

نلاحظ في هذا المثال أن Avg تتجاهل القيم Null.

5.1.6. استخدام الدالة Sum

نستخدم الدالة Sum لحساب المجموع الكلي لقيم عمود ما. هناك عدة اعتبارات يجب مراعاتها عند استخدام Sum:

- ✓ تُستخدم Sum مع القيم العددية فقط.
- ✓ تهمل Sum القيمة Null عند حساب المجموع الكلي.
- ✓ تهمل Sum القيم المتكررة في العمود عند حساب المجموع الكلي وذلك إذا صرحنا بكتابة Distinct قبل اسم العمود.

يمكننا حساب المجموع الكلي لعمود Salary في جدول Employees ، وذلك بكتابة العبارة الآتية:

```
Select Sum(Salary) From  Employees;
```

5.1.7. استخدام الدالة Nvl مع الدوال التجميعية Group Functions

```
SELECT AVG(NVL(COMMISSION_PCT,0))
FROM   EMPLOYEES;
```

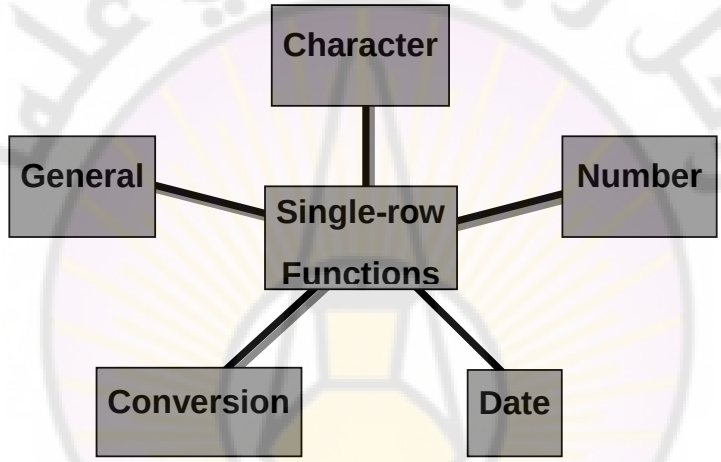
نستخدم الدالة Nvl مع الأعمدة التي تحتوي القيم الفارغة Null، حيث تقوم بمعالجة مشاكل القيم الفارغة باستبدالها بالقيمة الصفرية وبالتالي يمكننا استخدام الدوال التجميعية مع الأعمدة التي تضم قيماً فارغة لكي تشمل جميع السجلات.

5.2. استخدام الدوال لسطر واحد

(Single Row Functions)

تعالج Single Row Functions قيم العمود المختار من الجدول وتقدم النتيجة على شكل قيمة وحيدة لكل سجل، في حين تُستخدم Group Function في معالجة قيم العمود المختار من الجدول وتقدم النتيجة على شكل قيمة وحيدة تخص العمود المختار.

5.2.1. أنواع الدوال لسطر واحد



5.2.1.1. الدوال النصية Character Functions

يمتلك SQL عدداً من الدوال التي تستخدم لتعديل أنماط البيانات النصية (Char, Varchar)، وسوف نتعرض لأهم هذه الدوال النصية و هي كما في الجدول الآتي:

الدالة النصية	الاستخدام
Initcap	تغيير حالة الحرف الأول من كل سلسلة نصية إلى حرف كبير و تحويل حالة الأحرف التالية للحرف الكبير إلى

	حالة الأحرف الصغيرة.
Length	إعادة عدد الأحرف المكونة لسلسلة حرفية
Lower	تحويل كل حرف ضمن سلسلة حرفية إلى أحرف صغيرة
Upper	تحويل كل حرف ضمن سلسلة حرفية إلى أحرف كبيرة
Concat	دمج سلسلة حرفية مع سلسلة حرفية أخرى
Replace	استبدال سلسلة حرفية بسلسلة حرفية أخرى
Soundex	إيجاد الكلمات المتشابهة من حيث اللفظ الصوتي
Substr	نسخ الجزء المحدد من الكلمة ابتداءاً من حرف محدد
Instr	إعادة قيمة عددية تمثل موضع مجموعة أحرف ضمن سلسلة حرفية

سنقوم باستعراض أمثلة على الدوال النصية كما هو موضح في الجدول الآتي:

Function الدالة	Result النتيجة
Lower ('Jicc Center')	jicc center
Upper ('Jicc Center')	JICC CENTER
Initcap ('Jicc Center')	Jicc Center
Concat('Jicc'),('Center')	JiccCenter
Substr('String',1,3)	Str
Length('String')	6
Instr('String','T')	2

```
SELECT  FIRST_NAME, CONCAT (FIRST_NAME, JOB_ID),
LENGTH (FIRST_NAME),INSTR(FIRST_NAME, 'A')
FROM   EMPLOYEES;
```

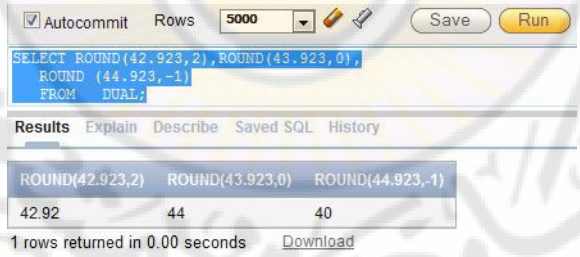
5.2.1.2. الدوال الرقمية Number Functions

تمتلك SQL عدداً من الدوال الرقمية التي تُستخدم مع أنواع البيانات الرقمية Numeric Data، وسوف نتعرض لأهم هذه الدوال الرقمية وهي موضحة في الجدول الآتي:

الدالة الرقمية	الاستخدام
Round	تقريب قيمة رقمية إلى عدد أرقام محدد بعد الفاصلة
Trunc	قطع أو قص القيم إلى دقة محددة
Mod	تحديد باقي القسمة الناتج عن عملية ما

سنقوم باستعراض أمثلة على الدوال الرقمية كما هو موضح فيما يأتي:

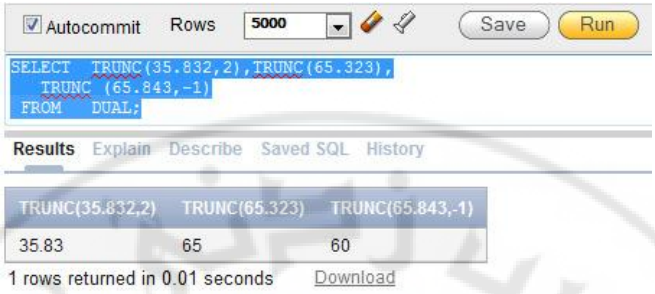
- Round (42.926,2) 42.93
- Trunc (44.926,2) 44.92
- Mod (1600,300) 100



مثال آخر:

```
SELECT TRUNC(35.832,2),TRUNC(65.323), TRUNC (65.843,-1)
FROM DUAL;
```

الجدول Dual هو جدول افتراضي موجود في لغة الـ SQL. يُستخدم هذا الجدول لإجراء العمليات التي لا يدخل فيها أي جدول من قاعدة البيانات.



5.2.1.3. التاريخ و دوال التاريخ

Date and Date Functions

تمتلك SQL عدداً من الدوال التي تُستخدم مع أعمدة التاريخ، حيث يمكن إظهار التاريخ بشكل حرفي. يمكننا استخدام التاريخ في عمليات الجمع والطرح. كما يمكن إضافة قيمة ما لتاريخ ما ونحصل بذلك على قيمة تاريخ جديدة. سوف نتعرض لأهم هذه الدوال وهي موضحة في الجدول الآتي:

الدالة	الإستخدام
Months_Between(Date1,Date2)	يحدد عدد الأشهر بين تاريخين
Next_Day(Date,'Day')	يحدد اليوم التالي من الأسبوع الذي يلي التاريخ المحدد
Add_Months(Date,Numberof Months)	يستخدم لحساب تاريخ جديد بناء على عدد محدد من الأشهر
Last_Day(Date1)	يحدد آخر يوم في الشهر الحالي أو لتاريخ محدد

سنقوم في المثال الآتي باستعراض أمثلة على دوال التاريخ كما هو موضح فيما يأتي:

Months_Between ('1-Sep-93','11-Jan-44')	19.6774194
Add_Months('11-Jan-96',6)	'11-Jul-96'
Next_Day('01-Sep-95','Friday')	'8-Sep-95'
Last_Day('01-Sep-95')	'30-Sep-95'

- ✓ Sysdate دالة SQL Function تُستخدم لاسترجاع التاريخ والوقت الحاليين.

```
Select Sysdate from  Dual;
```

5.3. الدوال التحويلية Conversion Functions

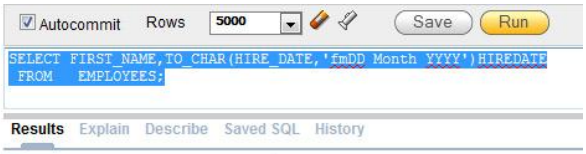
تستخدم SQL عدة دوال تحويلية. من أهم هذه الدوال التحويلية نذكر:

- ✓ To_Number : يُستخدم لتحويل نوع البيانات إلى نوع البيانات الرقمية. مثلاً يمكننا استخدام To_Number لتحويل السلسلة الحرفية السنة إلى حقل رقمي وذلك باستخدام أحد الدوال الرقمية على هذا الحقل مثل Mod.
- ✓ To_Char : يُستخدم لتحويل نوع البيانات إلى نوع البيانات النصية. مثلاً يمكننا استخدامه لتغيير تاريخ ميلاد موظف إلى حقل حرفي وذلك لتنسيقه ليظهر بتنسيق خاص. تعدّ هذه الدالة من أهم الدوال التحويلية في نظام لإدارة قواعد البيانات أوراكل Oracle مثلاً، ومن أهم استخداماتها تحويل نوع البيانات الرقمية إلى نوع البيانات النصية وذلك لإضفاء عليها تنسيق خاص مثل إظهار علامة بجوار الرقم والتحكم في رمز العلامة العشرية وعدد الأرقام بعد العلامة العشرية وهناك عدة رموز يمكننا استخدامها عند تحويل البيانات الرقمية إلى نوع البيانات النصية، كما هو موضح بالجدول الآتي :

الرمز	الاستخدام
9	يمثل رقم Number
0	يمثل الصفر Zero
$	يمثل علامة الدولار Dollar Sign
L	يمثل علامة العملة على النظام Local Currency Symbol
.	يمثل رمز العلامة العشرية Decimal Point
,	يمثل رمز تجميع الأرقام A Thousand Indicator

- ✓ To_Date : تُستخدم لتحويل نوع البيانات إلى نوع البيانات تاريخ.

في هذا المثال سنقوم بتحويل نوع بيانات عمود Hire_Date من نوع Date إلى نوع البيانات الحرفية باستخدام الدالة To_Char وذلك لعمل تنسيق خاص لحقل التاريخ.



Results | Explain | Describe | Saved SQL | History

FIRST_NAME	HIREDATE
Steven	17 June 2003
Neena	21 September 2005
Lex	13 January 2001
Alexander	3 January 2006
Bruce	21 May 2007
David	25 June 2005
Valli	5 February 2006
Diana	7 February 2007
Nancy	17 August 2002
Daniel	16 August 2002
John	28 September 2005
Ismael	30 September 2005
Jose Manuel	7 March 2006
Luis	7 December 2007
Den	7 December 2002
Alexander	18 May 2003
Shelli	24 December 2005
Sigal	24 July 2005

ملاحظات:

- ✓ تنسيق التاريخ لابد أن يوضع بين علامتي تنصيص مفردة ' '.
- ✓ نفصل بين التاريخ والتنسيق بفاصلة.

في هذا المثال سنقوم بتحويل نوع بيانات عمود Salary من نوع Number إلى نوع البيانات الحرفية وذلك باسخدام To_Char وذلك لعمل تنسيق خاص لحقل المرتب مع استخدام الرموز التي سبق الإشارة إليها:

```
SELECT TO_CHAR(SALARY,'$99,999') SALARY
FROM   EMPLOYEES;
```

SALARY
$24,000
$17,000
$17,000
$9,000
$6,000
$4,800
$4,800
$4,200
$12,008
$9,000
$8,200
$7,700
$7,800
$6,900
$11,000

5.4. الدوال العامة General Functions

من أهم الدوال العامة في Single Row Function دالتي Nvl و Decod. فيما يأتي توضيح لاستخدامات كلتا الدالتين في SQL.

5.4.1. الدلة NVL

تُستخدم الدالة Nvl لاستبدال القيمة الفارغة بقيمة أخرى، ويمكن استخدام Nvl مع نوع البيانات الرقمية وغير الرقمية، ومن أهم استخدامات تلك الدالة التغلب على مشكلة القيم الصفرية في البيانات الرقمية إذ تقوم Nvl باستبدال القيم الفارغة بقيم صفرية. هناك عدة اعتبارات يجب مراعاتها عند استخدام الدالة Nvl:

- ✓ تستخدم الدالة Nvl في تحويل القيم الفارغة إلى قيم فعلية.
- ✓ يمكنها معالجة أنواع البيانات الرقمية Numeric Data، والنصية Character String، والتاريخية Date.
- ✓ عند تحويل القيم الفارغة عليك بالالتزام بنوع البيانات نفسها Data Types في العمود الذي يضم القيم الفارغة فعلى سبيل المثال:

- Nvl(Commission_Pct,0)
- Nvl(Job_Id,'No Job Yet')
- Nvl(Hire_Date,'01-Jan-98')

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT,(
SALARY*12)+NVL(COMMISSION_PCT,0)
 FROM   EMPLOYEES;
```

FIRST_NAME	SALARY	COMMISSION_PCT	(SALARY*12)+NVL(COMMISSION_PCT,0)
Steven	24000	-	288000
Neena	17000	-	204000
Lex	17000	-	204000
Alexander	9000	-	108000
Bruce	6000	-	72000
David	4800	-	57600
Valli	4800	-	57600
Diana	4200	-	50400
Nancy	12008	-	144096
Daniel	9000	-	108000
John	8200	-	98400
Ismael	7700	-	92400
Jose Manuel	7800	-	93600
Luis	6900	-	82800
Den	11000	-	132000
Alexander	3100	-	37200
Shelli	2900	-	34800

6. استخدام Group by

فيما سبق كانت SQL تعالج الجدول أو الجداول بالكامل أو السجلات المحققة لشرط معين في جملة Where، إلا أننا في بعض الأحيان نريد إجراء عمليات منطقية على سجلات معينة مثل إظهار القيمة العظمى والدنيا والمتوسط الحسابي لكل قسم من الأقسام على حدة. لإجراء مثل هذا نحتاج إلى استخدام Group By، حيث تستخدم جملة Group By لتحديد الأعمدة التي يتم التجميع بناءً عليها أو عندما نريد تطبيق

الدوال التجميعية مثل Avg, Sum, Count, Max, Min على سجلات من الجدول معزولة على شكل مجموعات فرعية، حيث تميّز كل مجموعة بقيم المعطيات نفسها، بحيث تقسم Group By سجلات الجدول إلى مجموعات صغيرة متشابهة، ثم يمكننا استخدام Group Functions لاسترجاع معلومات تلخيصية لكل مجموعة من هذه المجموعات على حدة، والصيغة العامة لاستخدام Group By تأخذ الشكل الآتي:

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

- ✓ تقع جملة Group by بعد جملة Where أو بعد جملة From في حالة عدم وجود جملة Where.
- ✓ تقسم Group by سجلات الجدول إلى مجموعات صغيرة متشابهة، ثم يمكننا استخدام Group Functions لاسترجاع معلومات تلخيصية لكل مجموعة من هذه المجموعات على حدة.
- ✓ تتعامل Group by مع القيم الفارغة Null كمجموعة مميزة عن غيرها.
- ✓ تمكننا Group by من إنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى مثل: Group by Deptno,Sal

هناك عدة اعتبارات يجب مراعاتها عند استخدام Group by :

- ✓ يمكن أن تضم عدة أعمدة وتعابير.
- ✓ يتم وضع فواصل بين أسماء الأعمدة.
- ✓ يجب أن تحتوي على كل الأعمدة المكتوبة بجملة Select و التي لا تستخدمها دوال التجميع.

في هذا المثال نريد حساب المجموع الكلي لمرتبات كل قسم على حدة. لفعل ذلك يجب علينا استخدام عبارة Group By في جملة Select:

```
SELECT deptno, sum(sal)
FROM  emp
GROUP BY deptno;
```

- ✓ يجب أن تحتوي جملة Group by على كل الأعمدة المكتوبة بجملة Select والتي لا تستخدمها دوال التجميع (وهو في المثال السابق حقل Deptno الذي يراد التجميع بناءً عليه).

يمكننا إجراء المثال السابق وحساب المجموع الكلي لمرتبات كل قسم على حدة باستخدام عمود Deptno بدون كتابة عمود Deptno في جملة Select.

```
SELECT sum(sal)
FROM emp
GROUP BY deptno;
```

- ✓ يمكننا إنشاء مجموعات من السجلات باستخدام Group by بناءً على أعمدة ليست موجودة بجملة Select.

```
SELECT deptno,job,sum(sal)
FROM     emp
GROUP BY deptno, job;
```

في هذا المثال سنوضح إمكانية قيام Group by بإنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى.

- ✓ أولاً: يتم إنشاء مجموعات متشابهة بناءً على رقم القسم Deptno (First,The Rows Are Grouped By Department Number)

- ✓ ثانياً: في مجموعات الأقسام يتم عمل مجموعات متشابهة من السجلات بناءً على الوظيفة.

(Second, Within The Department Number Groups, The Rows Are Grouped By Job Title.)

عند إجراء المثال الآتي تظهر لنا رسالة الخطأ الآتية: (Ora-00934 : وظيفة المجموعة غير مسموحة هنا) والتي تعني أننا لا نستطيع استخدام الدوال التجميعية Group Function مع جملة Where.

```
SELECT deptno, MAX(sal)    FROM          emp
  WHERE MAX (sal) > 2000
  GROUP BY       deptno;
```

7. استخدام Having

تقييد الاختيار ضمن المجموعة (Restriction Group Selection)

قد لا نريد استرجاع دائماً جميع المجموعات الفرعية المختلفة في الجدول ضمن استعلام واحد. تقدم لنا SQL جملة Having لحذف المجموعات الفرعية من التي لا نريد استرجاعها في الاستعلام أو التقرير. تعدّ جملة Having مشابهة لجملة Where من حيث إنها تحدد فيما إذا كانت السجلات قابلة للاختيار أو لا، وتضمّ جملة Having محددات تستخدم لتقييم السجلات، حيث يمكننا وضع عدة شروط في جملة Having. كما يمكننا استخدام المعاملات المنطقية معها مثل and, or، وقد يتبادر إلى الذهن لماذا لا نستخدم عوضاً عنها جملة Where. لا يمكننا ذلك حيث يكمن الاختلاف بين جملة Having و جملة Where في أن محددات الجملة Having يجب أن تكون تابع تجميع، ويتم استخدام Having فقط عندما يتم حساب قيمة مجمعة بوساطة Select،

بينما عمل Where هو ترشيح السجلات التي سيطبق عليها لاحقاً جملة Group by، و ليس ترشيح المجموعات. الصيغة العامة لاستخدام Having موضحة كالآتي:

```
SELECT column, group_function(column)
FROM   table
[WHEREcondition]
[GROUP BY      group_by_expression]
[HAVING        group_condition]
[ORDER BY      column];
```

هناك عدة اعتبارات يجب مراعاتها عند استخدام Having:

- ✓ تقع جملة Having بعد فقرة Group by.
- ✓ تستخدم Having فقط مع Group By ولا يمكن استخدامها بدونها.
- ✓ نستخدم Having مع الدوال التجميعية الآتية فقط (Max,Min,Avg,Count,Sum).
- ✓ يمكن تحديد عدة شروط في جملة Having وذلك باستخدام المعاملات المنطقية and, or.
- ✓ يمكننا استخدام كلاً من Having و Where في جملة Select واحدة.

```
SELECT job, SUM(sal) PAYROLL
FROM      emp
WHERE   job NOT LIKE 'SALES%'
GROUP BY job
HAVING    SUM (sal)<6000
ORDER BY SUM (sal);
```

في المثال السابق نقوم باسترجاع الوظيفة والمجموع الكلي للمرتبات الشهرية لكل وظيفة على حده بشرط أن يكون المجموع الكلي للمرتبات الشهرية لكل وظيفة أقل من 6000

باستثناء وظيفة Salesman، ثم الترتيب التصاعدي بناءً على المجموع الكلي للمرتبات الشهرية.

8. استخدام وعمل استعلام فرعي (Sub Query)

تمكننا SQL من جعل الاستعلامات متداخلة ببعضها البعض بشكل نموذجي، بحيث ينتج عن الاستعلام الداخلي (Inner Query) قيماً تفحص في قسم الشرط للاستعلام الخارجي (Outer Query) لمعرفة فيما إذا كان الشرط سيتحقق عندها أم لا.

8.1. كيف تعمل الاستعلامات الفرعية؟

لنفترض أننا نريد كتابة استعلاماً لاسترجاع من يتقاضى مرتباً أكثر من محمد ؟ نجد أننا نحتاج إلى استعلامين:

الأول: ماهو مرتب محمد؟

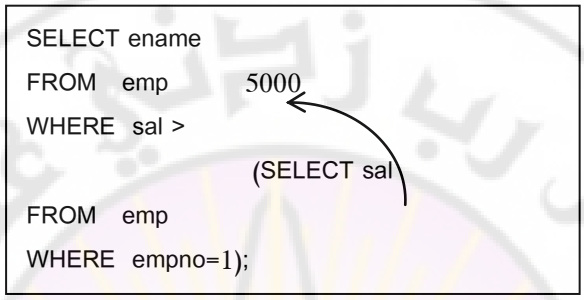
الثاني: البحث عن من يتقاضى مرتباً أعلى من مرتب محمد؟

لحل مثل تلك المشكلة لابد من دمج الاستعلامين معاً، بمعنى استخدام استعلام داخل استعلام آخر، حيث يعيد الاستعلام الداخلي (أو الفرعي) قيمة يستخدمها الاستعلام الرئيسي (أو الخارجي) لمعرفة فيما إذا كان سيتحقق الشرط عندها أم لا.

- ✓ ينفذ الاستعلام الفرعي أو الداخلي أولاً ثم ينفذ الاستعلام الرئيسي أو الخارجي.
- ✓ يولد أو يعيد الاستعلام الداخلي نتيجة يستخدمها الاستعلام الخارجي لمعرفة ما إذا سيتحقق الشرط عندها أم لا.
- ✓ الصيغة العامة لعمل واستخدام الاستعلامات الفرعية Sub Queries كالآتي:

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

في المثال الآتي نجد أن الاستعلام الداخلي يحدد مرتب الموظف الذي رقمه 1، يأخذ الاستعلام الخارجي (أو الرئيسي) هذه النتيجة ويستخدمها لاسترجاع الموظفين الذين يتقاضون مرتباً أعلى من هذا المرتب.



8.2. أنواع الاستعلامات الفرعية

Types of Sub Queries

1. الاستعلامات الفرعية التي تعيد سجل واحد

Single-Row Sub Queries

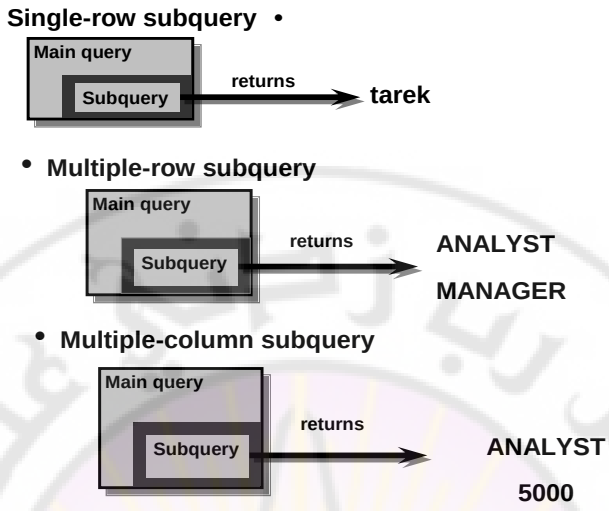
2. الاستعلامات الفرعية التي تعيد أكثر من سجل واحد

Multiple-Row Sub Queries

3. الاستعلامات الفرعية التي تعيد أكثر من عمود واحد

Multiple-Column Sub Queries

Damascus University



8.2.1. الاستعلامات الفرعية التي تعيد سجل واحد

كما سبق وتعلمنا أن (Single–Row Sub Queries) تعيد سجلاً واحداً فقط، ويستخدم ذلك النوع من الاستعلامات مع معاملات المقارنة الموضحة بالجدول الآتي :

Meaning المعنى	Operator المعامل
يساوي	=
أكبرمن	>
أكبرمن أو يساوي	>=
أصغر من	<
أصغر من أو يساوي	<=
لا يساوي	! = or <>

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) للموظفين الذين لهم الوظيفة نفسها للموظف رقم 555.

```
SELECT ename,job
FROM emp
WHERE job=
                    (SELECT        job
                FROM emp
          Where empno=555);
```

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) للموظفين الذين لهم الوظيفة نفسها للموظف رقم 7698 ومرتباتهم أكبر من راتب الموظف رقم 7934.

```
SELECT   ename, job
FROM     emp          MANAGER
WHERE    job =
              (SELECT  job
                FROM          emp
                  WHERE       empno = 7698)
  AND     sal >       1300
                (SELECT       sal
                FROM   emp
                WHERE empno = 7934);
```

- ✓ لابد من وضع الاستعلام الفرعي بين قوسين.
- ✓ لا تستخدم جملة Order By داخل الاستعلام الفرعي.

8.2.1.1. استخدام الدوال التجميعية في الاستعلام الفرعي

تتميز الدوال التجميعية بأنها تولد أو تعيد قيمة وحيدة مهما كان عدد السجلات المختارة. ينبغي علينا الانتباه إلى أن الدوال التجميعية التي هي دوال تجميع معرفة في جملة Group By ستولد وتعيد عدة قيم. لهذا السبب لا يسمح باستخدامها مع هذا النوع من

الاستعلامات الفرعية التي تعيد سجلاً واحداً، حتى لو استخدامنا Group By و Having لتوليد مجموعة وحيدة لخرج الاستعلام الفرعي. لهذا يجب علينا استخدام الدوال التجميعية المرغوبة في جملة Where و التي تحذف المجموعات غير المرغوبة.

في هذا المثال سنقوم باسترجاع بيانات الموظف (الاسم، الوظيفة، المرتب) الذي يتقاضى أعلى مرتب، حيث يمكننا تنفيذ هذا الاستعلام باستخدام Group Function داخل الاستعلام الفرعي كما يأتي:

```
SELECT ename, job, sal
FROM   emp
WHERE sal =          5000
                (SELECT          MAX(sal)
            FROM     emp);
```

8.2.1.2. استخدام (Group by و Having) في الاستعلام الفرعي

يمكن أن نستخدم الاستعلامات الفرعية في جملة Having كما يمكن استخدام الدوال التجميعية أو Group by و Having كما في المثال الآتي:

```
SELECT deptno, MIN(sal)
FROM   emp
GROUP BY       deptno
HAVING MIN(sal) >
                (SELECT          MIN(sal)
                FROM   emp
                WHERE deptno = 10);
```

في هذا المثال تم إنشاء مجموعات أصغر مرتبات بناءً على عمود رقم القسم، مع تقييد ذلك بشرط وهو أن يكون أقل مرتب في تلك الأقسام أكبر من أقل مرتب في القسم رقم 10.

- ✓ يجب علينا التأكد عند استخدام الاستعلامات الفرعية في القسم الشرطي (الذي فيه عامل المساواة أو عدم المساواة) من أن الاستعلام الفرعي سيعيد سجلاً واحداً فقط، وإلا سيصدر خطأ إذا أعاد أكثر من قيمة واحدة. لن يفشل الأمر إذا لم يعيد الاستعلام الفرعي أية قيمة ولكننا عندها لن نحصل على خرج من الاستعلام الفرعي.

8.2.2. الاستعلامات الفرعية التي تعيد أكثر من سجل واحد

تعيد الاستعلامات الفرعية متعددة السجلات أكثر من سجل، وتتطلب هذه الأنواع من الاستعلامات الفرعية معاملاً يمكن استخدامه لتقييم عدة قيم. على سبيل المثال لا الحصر يمكننا استخدام المعامل In مع الاستعلامات الفرعية متعددة الأسطر لأنه يقييم مصفوفة من القيم. الجدول الآتي يوضح المعاملات التي يمكننا استخدامها لتقييم الاستعلامات متعددة السجلات.

الوصف	المعامل Operator
يكون المحدد مساوياً لأي من القيم التي يعيدها الاستعلام الفرعي لهذا الشرط حتى يصبح محققاً (True)	In
يقارن المحدد بكل قيمة يعيدها الاستعلام الفرعي، ويكون الشرط محققاً (True) إذا كان أي عنصر من المجموعة يحقق الشرط	Any
يقارن المحدد بكل قيمة يعيدها الاستعلام الفرعي، ويكون الشرط محققاً	All

في هذا المثال نريد استرجاع بيانات الموظفين (رقم الموظف، الاسم، الوظيفة) ذوي المرتبات الأقل من أي من الموظفين الذين يعملون بوظيفة Clerk والذين لا يعملون في وظيفة Clerk

```
SELECT empno, ename, job    1300
FROM    emp                 1100
WHERE   sal < ANY           1300
                            (SELECT         sal     950
                            FROM    emp
                            WHERE job = ' CLERK ')
AND         job <> 'CLERK';
```

- ✓ Any <: تعني أقل من Minimum.
- ✓ Any >: تعني أقل من Maximum.
- ✓ Any =: تكافىء معامل In.

8.2.3. الاستعلامات الفرعية التي تعيد أكثر من عمود واحد

(Multiple-Column Sub Queries)

رأينا في الأمثلة السابقة استعلامات تستخدم أعمدة وحيدة كمحددات تقييم، وأحياناً لتنفيذ نوع من الشروط يجب علينا استعراض عدة أعمدة وكأنها محدد واحد، ويجب إحاطة تلك الأعمدة بين قوسين ووضع فواصل بين الأعمدة.

```
SELECT ename, deptno, sal, comm
FROM   emp
WHERE  (sal, NVL (comm,0)) IN
                              (SELECT sal, NVL (comm,0)
                       FROM   emp
                       WHERE deptno = 30);
```

في المثال السابق نريد استرجاع بيانات الموظف(الاسم، رقم القسم، المرتب، الحوافز)، لأي موظف يتماثل في المرتب والحوافز مع أي موظف في القسم رقم 30.

9. عبارة Insert: كيفية إضافة سجلات جديدة

تمتلك معظم أنظمة إدارة قواعد البيانات أدوات ممتازة لإدخال وتعديل وحذف البيانات من الجداول، وهي بلا شك أفضل وأسرع بكثير من إدخال البيانات بعبارات SQL، إلا أن ذلك يمّكن المبرمج من معالجة البيانات ضمن برامجه التي يكتبها.

نستطيع إضافة سجلات جديدة إلى الجدول باستخدام عبارة Insert، ويوجد لدينا صيغتان من عبارة Insert: الأولى تسمح بإضافة سجل واحد في نفس الوقت إلى الجدول، والثانية تسمح باختيار مجموعة سجلات من جدول وإدراجها في جدول آخر.

9.1. إضافة سجل واحد إلى الجدول

كما وضحنا سابقاً أنَّ عبارة Insert لها صيغتان الأولى والتي نحن بصددها تمكننا من إضافة سجل واحد إلى الجدول في الوقت نفسه، والصيغة العامة لها تأخذ الشكل الآتي:

مما سبق نجد أن الصيغة العامة لعبارة Insert تتألف مما يأتي:

1. Insert,Into,Values هي كلمات محجوزة في SQL.

```
INSERT INTO table [(column [, column...])]
VALUES      (value [, value...]);
```

2. Table ويشير إلى اسم جدول وحيد في قاعدة البيانات، وبالتحديد الجدول الذي نريد إدراج السجلات فيه.

3. Column: وتشير إلى أعمدة الجدول.
4. لائحة من القيم محاطة بين قوسين تقوم SQL بإدراجها في الأعمدة المقابلة لها على الترتيب.

```
5.  INSERT INTO table([(column [, column...])
6.  Select ([(column [, column...])])
7.  [where conditions]);
```

لإدراج سجل من البيانات إلى جدول Dept، نقوم بكتابة العبارة الآتية:

```
INSERT INTO    dept (deptno, dname, loc)
  VALUES               (50, 'DEVELOPMENT', 'cairo');
1 row created.
```

لإدراج سجل من البيانات إلى جدول Emp، نقوم بكتابة العبارة الآتية:

```
INSERT INTO    EMP (empno, ename, job,
                    mgr, hiredate, sal, comm,
                    deptno)
VALUES         (7369, 'ayman', 'programmer',
                    7372, SYSDATE, 4000, NULL,
                    50);
1 row created.
```

- ✓ يجب أن تتطابق الأعمدة والقيم المقابلة لها في نوع البيانات Data Types.
- ✓ لائحة القيم لابدّ أن تأخذ ترتيب الأعمدة نفسه في الجدول المراد إضافة السجلات إليه.
- ✓ لابد من وجود علاقة واحد إلى واحد بين الأعمدة ولائحة القيم، بمعنى أنه توجد قيمة واحدة من أجل كل عمود بالترتيب ذاته.

- ✓ قد نجهل قيمة أحد الحقول، وذلك بكتابة كلمة Null، بشرط ألا يكون العمود المقابل للقيمة معرفاً بالخاصية Not Null.
- ✓ عند إدخال قيم التاريخ والقيم المحرفية لا بد من وضعها بين علامتي اقتباس ' '.
- ✓ يجوز عدم ذكر أسماء الأعمدة في جملة Insert. في هذه الحالة لا بد من إدخال جميع قيم الأعمدة الموجودة في الجدول بحسب ترتيب الأعمدة داخل الجدول مع مراعاة نوع البيانات لكل عامود.
- ✓ قد تسبب Insert مشكلة في التجانس والتكامل المرجعي (وسيتم تناول ذلك عند إنشاء العلاقات بين الجداول لاحقاً في هذا الفصل).
- ✓ يمكن إضافة القيمة إلى الأعمدة بطريقتين:

1. عدم كتابة الأعمدة المُراد تخزين القيمة Null فيها في الجزء Insert.
2. كتابة الأعمدة ونكتب قيمتها Null داخل الجزء Values.

للتأكد من تنفيذ العبارة السابقة وإضافة السجل إلى جدول Emp قم بتنفيذ العبارة الآتية:

```
Select *  From Emp;
```

9.2. إضافة عدة سجلات إلى الجدول

نلاحظ أنه ليس من المنطقي أن تقوم بإدخال 20,000 سجل مثلاً في أحد الجداول سجلاً سجلاً باستخدام الصيغة الأولى من عبارة Insert. لإضافة عدة سجلات في الوقت نفسه تمكننا الصيغة الثانية من عبارة Insert من إضافة عدة سجلات إلى جدولنا مختارة من جدول آخر، والصيغة العامة لها هي:

```
INSERT INTO table([(column [, column...])
Select ([(column [, column...])])
[where conditions]);
```

مما سبق نجد أن الصيغة العامة لعبارة Insert في هذه الحالة تتألف مما يأتي:

1. Insert, Into هي كلمات محجوزة في SQL.
2. Table ويشير إلى اسم جدول وحيد في قاعدة البيانات، وبالتحديد الجدول الذي نريد إدراج السجلات فيه.
3. Column: وتشير إلى أعمدة الجدول.
4. عبارة Select: هي عبارة Select عادية تستخدم لانتخاب الأعمدة التي ستنسخ محتوياتها إلى الجدول المراد إضافة السجلات إليه. على سبيل المثال نريد إضافة سجلات أمان إلى الجدول Secemp من أجل كل سجل من سجلات جدول Emp كما يأتي:

```
INSERT INTO secemp (empno,name,job,mgr,sal,comm,deptno)
(Select (mpno,name,job,mgr,sal,comm,deptno)
 From emp);
```

9.3. اضافة سجلات عن طريق المتحولات البديلة

تمكننا لغة الـ SQL من استخدام متحولات تدعى بالمتحولات البديلة. هذه المتحولات عبارة عن مخزن مؤقت للبيانات و من خلالها يتم تخزين قيم معينة داخل المتحولات. أثناء تنفيذ عبارة SQL يتم استبدال هذه المتحولات بقيمتها و يتم تعريفها أثناء كتابة عبارة SQL وذلك بوضع الرمز & قبل اسم المتحول. كما تظهر رسالة تطلب قيمة هذه المتحولات أثناء التنفيذ. المثال الآتي يوضح كيفية إضافة سجل جديد باستخدام المتحولات البديلة:

```
INSERT
INTO dept (deptno,dname,loc)
Values (&dept_Id,'&dept_name','&dept_loc');
Enter value for dept_Id : 50
Enter value for dept_name : Informatics
Enter value for dept_loc : Damascus
```

في هذا المثال قمنا بإضافة بيانات لقسم جديد عن طريق استخدام المتحولات البديلة الآتية :

- &dept_Id: متحول يمثل رقم القسم وعند التنفيذ تم تخزين الرقم 50 فيه.
- &dept_name: متحول يمثل اسم القسم وعند التنفيذ تم تخزين القيمة Informatics فيه .
- &dept_loc: متحول يمثل عنوان القسم وعند التنفيذ تم تخزين القيمة Damascus فيه.

يمكن تكرار هذا الأمر بإضافة أكثر من قسم دون الحاجة إلى كتابة العبارة مرة أخرى.

بالنتيجة قمنا بإضافة قسم جديد إلى جدول الأقسام وكأننا كتبنا العبارة الآتية:

```
INSERT INTO dept (deptno,dname,loc)
Values (50,Informatics,Damascus);
```

يمكن استخدام المتحولات البديلة بدلاً من أسماء الجداول أو أسماء الأعمدة كما في المثال الآتي:

```
INSERT
INTO &dept_table (&dept_Id,dname,loc)
Values (50,Informatics,Damascus);
Enter value for dept_table : dept
Enter value for dept_Id : deptno
```

في هذا المثال قمنا باستخدام المتحول البديل &dept_table بدلاً من اسم الجدول dept و أيضاً المتحول البديل &dept_Id بدلاً من اسم العمود deptno.

10. حذف بيانات موجودة في قاعدة البيانات

(Deleting Record)

نستخدم عبارة Delete لحذف سجلات من جدول ما، وتأخذ الصيغة العامة لعبارة Delete الشكل الآتي:

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

مما سبق نجد أن الصيغة العامة لعبارة Delete تتألف مما يأتي:

1. Delete: هي أحد الكلمات المحجوزة في SQL.
2. From: هي أحد الكلمات المحجوزة في SQL ويليها اسم الجدول المراد حذف السجلات منه.
3. Where: هي أحد الكلمات المحجوزة في SQL ويليها شرط حذف السجلات، وهي عبارة اختيارية للحد من عدد السجلات المحذوفة. فإذا تم حذف عبارة Where من جملة Delete فإن هذا يعني حذف السجلات جميعها.

10.1. حذف سجلات محددة

(Deleting Sepecific Record)

لحذف سجلات الموظفين ذوي المرتبات الأكبر من 5000 من جدول Emp يتم كتابة العبارة الآتية:

```
Delete From     Emp Where Sal > 5000;
```

10.2. حذف السجلات كافة (Deleting all Record)

كما وجدنا يمكننا حذف السجلات كافة من جدول ما بحذف عبارة Where من جملة Delete. هذا يعني حذف السجلات جميعها، والصيغة العامة لها هي:

```
DELETE [FROM]   table;
```

لحذف السجلات كافة من جدول Emp يتم كتابة العبارة الآتية:

```
Delete From      Emp;
```

10.3. استخدام الاستعلام الفرعي في جملة Delete (Sub Queries In Delete Statements)

يمكننا استخدام الاستعلام الفرعي في جملة Delete، وذلك لحذف سجلات من جدول بناءً على قيم من جدول آخر. فعلى سبيل المثال إذا أردنا حذف سجلات الموظفين التابعين لقسم المبيعات يتم ذلك على النحو الآتي :

```
DELETE FROM   employee
WHERE         deptno =
                        (SELECT   deptno
                        FROM      dept
                        WHERE     dname='sales');
5 rows deleted.
```

- ✓ باستخدام Delete نستطيع حذف بيانات جدول واحد فقط في الوقت نفسه.
- ✓ تحذف Delete جميع السجلات التي تحقق الشرط في جملة Where.
- ✓ لا تحذف Delete الجدول.
- ✓ تحذف السجلات من الجدول بشكل مستمر، إلا أن الحذف الحقيقي للسجلات يعتمد على معالجة Commit المستخدمة التي سيتم تناولها في فقرة لاحقة في هذا الفصل.

- ✓ قد تسبب Delete مشكلة في التجانس والتكامل المرجعي (وسيتم تناول ذلك عند إنشاء العلاقات بين الجداول لاحقاً في هذا الفصل).
- ✓ تحذف Delete السجل أو السجلات بالكامل فلا حاجة للإشارة إلى أي عمود إلا في جملة Where عند استخدامها.

11. تعديل / تحديث البيانات الموجودة في قاعدة البيانات (Using the Update Command)

يمكننا استخدام Update لتعديل القيم الموجودة في جدول ما، والصيغة العامة لعبارة Update تأخذ الشكل الآتي:

```
UPDATE          table
SET column = value [, column = value]
[WHERE          condition];
```

مما سبق نجد أن الصيغة العامة لعبارة Update تتألف مما يأتي:

1. Update: من الكلمات المحجوزة في SQL وهو أمر بتحديث البيانات .
2. Table: يشير إلى اسم الجدول المراد التعديل في بياناته.
3. Set: تحتوي على سلسلة من عمليات التحديث، حيث نقوم بوضع اسم العمود الذي نقوم بتعديل بياناته كمعامل أول، ويحتوي المعامل الثاني على القيمة الجديدة، و يفصل بين المعاملين إشارة المساواة.
4. Where: وهي عبارة اختيارية يتم استخدامها لتحديد السجلات التي يراد التعديل بها، فإذا تم حذف عبارة Where فإن التعديل سيتم على كل سجلات الجدول.

11.1. تحديث السجلات كافة

(Update all Record)

يمكننا تحديث السجلات كافة من جدول ما بحذف عبارة Where من جملة Update فإن هذا يعني تحديث السجلات جميعها، والصيغة العامة لها هي:

```
UPDATE          table
SET             column = value [, column = value];
```

لتحديث السجلات جميعها من جدول Emp وذلك بوضع المرتب 3000 لكل الموظفين يتم كتابة العبارة الآتية:

```
UPDATE          emp
SET     sal = 3000;
14 rows updated.
```

11.2. تحديث سجلات محددة

(Update Specific Record)

يمكننا تحديث سجلات محددة والتعديل فيها بناءً على شرط أو عدة شروط، وذلك بتضمين عبارة Update فقرة Where. على سبيل المثال لتعديل مرتب الموظف رقم 555 من 3000 إلى 6000 يتم ذلك على النحو الآتي:

```
UPDATE          emp
SET     sal = 6000
WHERE           empno = 555;
```

11.3. استخدام الاستعلام الفرعي في جملة Update

(Sub Queries In Update Statements)

يمكننا استخدام الاستعلام الفرعي في جملة Update. على سبيل المثال إذا أردنا تعديل أو تحديث كلاً من الوظيفة والقسم للموظف رقم 555 لتماثل الوظيفة والقسم للموظف رقم 7489، سوف نحتاج إلى استخدام استعلام فرعي، ويتم كتابة عبارة Update على النحو الآتي:

```
UPDATE emp
SET     (job, deptno) =
                        (SELECT job, deptno
                FROM    emp
                WHERE   empno =7489)
WHERE   empno = 555;
1 row updated.
```

12. استخدام تعليمات التثبيت Commit والتراجع Rollback

يتطلب تنفيذ التغييرات في بيانات الجداول خطوتين:

1) تنفيذ إحدى العبارات الآتية Insert, Update, Delete.
2) تأكيد التغيير باستخدام Commit.

عندما يتم تنفيذ عبارة من عبارات DML (Insert, Update, Delete)، يقوم نظام إدارة قواعد البيانات أوراكل Oracle مثلاً بتغيير الجدول وحفظ نسخة من السجلات قبل إجراء التعديل، وسيبدو الأمر للمستخدم وكأن التغييرات قد تمت بشكل نهائي ودائم. إذا

قام المستخدم بالاستعلام في قاعدة البيانات، فإنه سيرى التعديلات موجودة. وإذا قام مستخدم آخر بالاستعلام في قاعدة البيانات فلن يرى تلك التعديلات، وذلك لأن التعديلات على البيانات لا تصبح مرئية من قبل المستخدمين الذين يُسمح لهم بالوصول إلى تلك الجداول، ولا تصبح دائمة حتى يتم تنفيذ تعليمة Commit. بمعنى أن تلك الخاصية تسمح للمستخدم بالتراجع عن أي تغيير قام به. يتم التراجع عن التعديلات باستخدام تعليمة Rollback، حيث تُعيد هذه التعليمة قاعدة البيانات إلى الحالة التي كانت عليها بعد تنفيذ آخر تعليمة Commit.

- ✓ يتم تنفيذ تعليمة Commit بشكل ضمني إذا قطع المطور(Developer) الاتصال مع نظام إدارة قواعد البيانات بشكل نظامي أو نفذ تعليمة من تعليمات DDL، حيث يؤدي ذلك إلى حفظ التغييرات.

```
SAVEPOINT save_point_name
ROLLBACK save_point_name
```

- ✓ إذا قُطع الاتصال مع نظام قاعدة البيانات بشكل غير نظامي قبل تنفيذ تعليمة Commit لن يتم حفظ التغييرات، وستعود قاعدة البيانات إلى النقطة التي تم عندها تنفيذ آخر تعليمة Commit.
- ✓ يمكن التراجع عن التغييرات بشكل جزئي، حيث تسمح تعليمة Savepoint للمستخدم حفظ التغييرات حتى جزء محدد من التغييرات. على سبيل المثال نفترض أننا قمنا بمجموعة من التعديلات على مجموعة من السجلات، ولكننا غير متأكدين من أننا أجرينا التعديلات بشكل صحيح. في حال رغبنا برؤية السجلات قبل تثبيت هذه التغييرات، يمكننا استخدام عدة تعليمات Savepoint، وبالتالي يمكننا التراجع عن التغييرات حتى أي نقطة من النقاط العلام الموضوعة كما هو موضح بالشكل الآتي، والصيغة العامة لكلٍ من نقطة العلام Savepoint والتراجع Rollback إلى نقطة علام تأخذ الشكل الآتي:

```
UPDATE…
SAVEPOINT jicc_ update;
Savepoint created.
INSERT…
ROLLBACK TO jicc_ update;
Rollback complete.
```

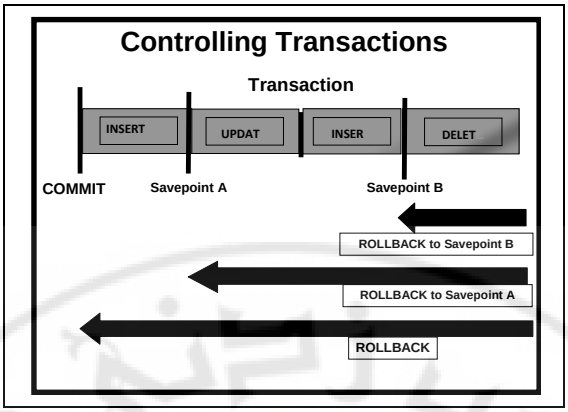
لتحديث كافة السجلات من جدول Emp وذلك بوضع المرتب 30000 لكل الموظفين يتم كتابة العبارة الآتية:

```
UPDATE          employee
SET     sal = 30000;
14 rows updated.
```

لحفظ التعديلات في قاعدة البيانات يتم كتابة العبارة الآتية Commit

لحذف السجلات كافة من جدول Emp ثم التراجع عن ذلك يتم بكتابة العبارات الآتية:

```
Delete  From    emp;
14 rows deleted.
ROLLBACK;
Rollback complete.
```



13. إنشاء وتعديل وحذف الجداول (كيفية إنشاء جداول وإضافة محددات من نوع Primary Key, Foreign Key)

مما سبق نجد أنه يوجد في لغة SQL ثلاث عبارات تسمى بعبارات تعريف البيانات وهي تسمح لنا بتعريف وحذف وتعديل الجداول في قاعدة البيانات، وهذه العبارات هي:

1. Create: تستخدم لتعريف جدول جديد في قاعدة البيانات.
2. Drop: تستخدم لحذف جدول موجود من قاعدة البيانات.
3. Alter: تستخدم لتغيير بنية الجدول الموجود.

قواعد البيانات العلائقية قادرة على تنفيذ كافة عبارات SQL في أي وقت من قبل أي مستخدم بشرط أن يكون له الحق في تنفيذ تلك الأوامر والعبارات. فلكي تقوم بإنشاء وحذف وتعديل الجداول لابد من أن تتأكد من أن مدير قاعدة البيانات يمنحك حق تعريف وحذف وتعديل الجداول.

13.1. إنشاء الجداول (Create Table)

تمكننا عبارة Create من إنشاء الجداول في SQL، وتأخذ عبارة Create الشكل الآتي:

```
CREATE TABLE   Table
        (column datatype [DEFAULT expr]);
```

مما سبق نجد أن الصيغة العامة لعبارة Create تتألف مما يأتي:

1- Create: من الكلمات المحجوزة في SQL، وتخبر DBMS بإنشاء كائن في قاعدة البيانات.

2- Table: من الكلمات المحجوزة في SQL، وتخبر DBMS بنوع الكائن المراد إنشائه في قاعدة البيانات.

3- Table Name: ويستخدم لتحديد اسم الجدول، ويجب أن يبدأ اسم الجدول بحرف نصي ويمكن أن يضمّ أحرفاً وأرقاماً، والشرطة_، ويجب ألا يتجاوز طوله 30 حرفاً، ولا يسمح بالفراغات، ولكن يسمح بالشرطة السفلية (Underscore).

4- Column Names: وتستخدم لتحديد أسماء الأعمدة المكونة للجدول المراد إنشاؤه، وتنطبق القواعد الخاصة بالتسمية نفسها على اسم الجدول على أسماء الأعمدة، ويجب أن تُحاط مجموعة تعريف الأعمدة بين قوسين.

5- Column Data Type: يحتوي تعريف الأعمدة على نوع البيانات، وطولها ودقتها كما هو موضح بالجدول اللاحق.

6- Constraints: وهي إعدادات اختيارية يمكن استخدامها لتمثل قيوداً للمحافظة على تكامل قاعدة البيانات.

هناك عدة اعتبارات يجب مراعاتها عند إنشاء الجداول وهي كالآتي:

1- يجب وضع أقواس تضم أسماء الأعمدة ونوع بياناتها.

2- يتم وضع فاصلة (,) بين تعريف أي عمود و آخر.

3- يجب أن يكون اسم العمود فريداً داخل الجدول.

4- لايمكن استخدام الكلمات المحجوزة كأسماء أعمدة في الجدول.

5- يتم استخدام الفاصلة المنقوطة لإخبار SQL أن الجملة انتهت وجاهزة للتنفيذ.

6- الجدول الآتي يوضح لنا أهم أنواع البيانات التي يمكن أن نستخدمها عند إنشاء الجداول في SQL.

نوع البيانات	الوصف
Char(N)	يعّرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (N) حرف، وجميع الحقول متساوية في الطول، حيث N=<255 .
Varchar2(N)	يعّرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (N) حرف، وتختلف الحقول في الطول وتتجاوز 255.
Number(N)	يعّرف عمود من نوع بيانات من نوع رقم، مع حد أقصى لعدد الأرقام (N)، ويمكن أن تأخذ (N) كحد أقصى 105.
Number(N,D)	يعّرف عمود من نوع البيانات Number، ويحتوي عدد أرقام قدره (N)، والتي تتضمن القيمة (D) التي توضح عدد الأرقام العشرية بعد الفاصلة.
Integer	يشبه نوع البيانات Number، ولكنه لا يأخذ غير القيم الرقمية الصحيحة، ولا يقبل أي أرقام عشرية بعد الفاصلة.
Long	يعّرف عمود من نوع البيانات الحرفية، وأقصى طول له 65.535، ولايمكن تعريف سوى عمود واحد من هذا النوع في الجدول، ولايمكن استخدام هذا العمود ضمن عبارات Where أو في الاستعلامات الفرعية أو الدوال أوالتعابير أو ضمن الفهارس.
Raw(N)	يعّرف هذا النوع عمود يحتوي سجل بيانات بطول قدره N
Smallint	يعّرف عمود يخزن أرقاماً موجبة وسالبة صغيرة تقع في

	النطاق (-32. 767 ، +32. 767)
Date	يعّرف عمود نوع بياناته تاريخ.

في العبارة الآتية نقوم بإنشاء جدول Department، يوضح أسماء أعمدته ونوع بياناتها.

```
CREATE TABLE department
        (deptno  NUMBER (2) not null,
                 dname   VARCHAR2 (15),
                 loc     VARCHAR2 (20));
Table created.
```

لعرض مواصفات وتعريفات الأعمدة نستخدم عبارة Describe أو اختصاراً Desc وتأخذ الشكل الآتي:

```
Describe Table Name;
```

13.1.1. إنشاء الجداول باستخدام الاستعلام الفرعي

يمكن إنشاء جدول باستخدام الاستعلام الفرعي كما في المثال الآتي:

نلاحظ وجود كلمة As بين اسم الجدول الجديد المراد إنشاؤه والاستعلام الفرعي. للتأكد

```
CREATE TABLE dept10
AS
SELECT   empno, ename, sal*12 ANNSAL,hiredate
FROM     emp
WHERE    deptno = 10;
Table created.
```

من مواصفات وتعريفات الأعمدة Column Definitions نستخدم الأمر Describe كما يأتي:

```
Describe Dept10 ;
```

13.2. تعديل الجدول (Altering Table)

يُستخدم Alter Table في تعديل بنية جدول موجود وتغيير محددات الجدول بعد إنشائه. فيما يأتي الخيارات التي يمكن استخدامها مع الأمر Alter:

```
ALTER TABLE table
    ADD (column datatype [DEFAULT expr]
                [, column datatype]...)
```

1) Add: يستخدم هذا الخيار لإضافة أعمدة جديدة أو شروط إلى الجدول ويأخذ الشكل الآتي:

```
ALTER TABLE dept10
ADD                 (job VARCHAR2(9));
Table altered.
```

2) Modify: يستخدم لتغيير محددات موجودة ويأخذ الشكل الآتي:

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expr]
                [, column datatype]...);
```

3) Disable: يُستخدم لإلغاء تفعيل شرط ما في الجدول.
4) Enable: يُستخدم لإعادة تفعيل شرط كنا قد قمنا بإلغاء تفعيله.
5) Drop: يُستخدم لإزالة شرط ما بشكل دائم من الجدول.
6) Set Unused: يُستخدم لتحديد عمود أو أكثر كأعمدة غير مستخدمة، بحيث يمكن حذف تلك الأعمدة من الجدول عندما تصبح مصادر النظام ملائمة لذلك.

7) Drop Column: يُستخدم لإزالة عمود ما من الجدول.

8) للتعديل في نوع البيانات Data Type الخاصة بعمود الـ Ename في جدول Dept30 نقوم بكتابة العبارة الآتية:

```
ALTER TABLE   dept30
MODIFY        (ename VARCHAR2(15));
Table altered.
```

13.3. حذف جدول (Dropping Table)

تستخدم SQL عبارة Drop لحذف الجداول من النظام. كل ما نحتاجه لحذف جدول هو أن نكتب اسمه بعد Drop Table فيتم حذف الجدول نهائياً بجميع سجلاته وفهارسه، ولايمنحنا النظام فرصة لإستعادة الجدول. تأخذ جملة Drop الشكل الآتي:

```
DROP TABLE table_name;
```

لحذف جدول Dept10 نقوم بكتابة العبارة الآتية:

```
DROP TABLE Dept10;
```

14. المحافظة على تكامل قاعدة البيانات

14.1. إضافة محددات إلى الجدول من نوع Primary/ Foreign Key

يوجد في نظام إدارة قواعد البيانات بعض الأدوات لحماية قاعدة البيانات. تسمى هذه الأدوات بالشروط Constraints. تقوم هذه الشروط بعدة وظائف مهمة منها:

1) التأكد من أن المفتاح الأساسي أو الرئيسي وحيد Unique.

2) فرض التكامل المرجعي: بمعنى التأكد من أن السجلات الأبناء الموجودة في جداول مرتبطة، تمتلك سجل أب.

3) التأكد من أن الأعمدة المرتبطة بالشروط تحتوي دائماً قيمة بداخلها Not Null.
4) التأكد من وضع القيمة الافتراضية Default Value في عمود ما.
5) التأكد من أن عموداً ما يحتوي على قيمة ما، وأن هذه القيمة موجودة ضمن نطاق محدد من القيم The Check Constraint.

14.2. الشرط Check (فحص وجود قيمة)

يُستخدم هذا الشرط للتأكد من أن قيمة ما لعمود ما تقع بين قيم مجموعة محددة. عندما نستخدم الشرط Check سيقوم نظام إدارة قواعد البيانات بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في المجموعة.

في المثال الآتي تم استخدام الشرط Check في تعريف جدول Employee. كما تم إعطاء اسماً للشرط Check وهو (Gender_Validation).

تسمية الشرط يكون اختيارياً، وإذا لم يتم تسمية الشرط فإن نظام إدارة قواعد البيانات سيولد اسماً خاصاً للشرط.

```
CREATE TABLE employee
        (emp_id NUMBER (2) not null,
                name     VARCHAR2 (15),
                gender   char (1)
Constraint gender_validation check (gender in('M','F')));
Table created.
```

نلاحظ أن نظام قواعد البيانات أوراكل Oracle مثلاً سيقوم بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في المجموعة ('M','F') فقط في العمود Gender بمعنى ألا يسمح إلا بهاتين القيمتين فقط.

14.3. خيار القيمة الافتراضية

(Default Value Option)

يستخدم هذا الخيار Default لإعطاء قيمة افتراضية لعمود ما عندما لايتم وضع قيمة في هذا العمود، ولا يمنع هذا الخيار من جعل القيمة Null في هذا العمود.

في هذا المثال تم استخدام الخيار Default للتأكد من أن العمود Comm تمّ إعطاؤه القيمة صفر عند عدم وضع أي قيمة له. يكون ذلك مفيداً في معالجة القيم الفارغة Null عند إجراء العمليات الحسابية.

```
CREATE TABLE employee
        (emp_id NUMBER (2) not null,
                name    VARCHAR2 (15),
                gender  char (1)
        sal     number
        comm    default (0),
Constraint gender_validation check (gender in('M','F')));
Table created.
```

14.4. الشرط Not Null (ليس فارغاً)

نستخدم الشرط Not Null للتأكد من أن العمود دوماً سيحتوي قيمة بداخله، وخاصة عندما يتم وضع هذا الشرط على عمود أو أعمدة المفتاح الأساسي للجدول.

لكي تقوم بوضع الشرط Not Null على عمود ما، قم بوضع العبارة Not Null بعد نوع بيانات العمود كما في المثال الآتي:

```
CREATE TABLE employee
        (emp_id NUMBER (2) not null,
                name    VARCHAR2 (15),
                gender  char (1)
        sal     number
        comm    default (0),
Constraint gender_validation check (gender in('M','F')));
Table created.
```

14.5. الشرط Unique (وحيد)

يستخدم الشرط Unique للتأكد من أن القيمة الموجودة بالعمود المشروط هي قيمة وحيدة في جميع سجلات الجدول. هذا يعني عدم السماح بتكرار القيم، حيث يقوم هذا الشرط بعمله عبر إنشاء فهرس وحيد (Unique Index) على هذا العمود، ويعدّ الشرط Unique أداة جيدة للمحافظة على هذه الخاصية في قاعدة البيانات. على سبيل المثال يمكن اعتبار عمود البريد الالكتروني الموجود في جدول الموظفين بانه لا يضم سوى قيمة وحيدة لا تتكرر، مع العلم بأن العمود " Emp_Id " هو المفتاح الأساسي في الجدول. أي إنه لابد أن يضمّ قيماً فريدة ووحيدة ولايسمح بالتكرار كما هو موضح كالآتي:

```
CREATE TABLE employee
        (emp_id NUMBER (2) not null
constraint unique emp_id unique,
         name   VARCHAR2 (15),
         gender char (1),
        email VARCHAR2(25),
          unique (email),
Constraint gender_validation check (gender in('M','F')));
Table created.
```

14.6. الشرط Primary Key (المفتاح الأساسي)

يستخدم الشرط Primary Key للمحافظة على تكامل عمود أو أعمدة المفتاح الأساسي، حيث يجعل هذا الشرط القيم الموجودة في المفتاح في عمود مشروط به تتمتع بالشرطين Not Null, Unique معاً. وعند تعريف هذا الشرط يتم ضمنياً إنشاء فهرساً وحيداً Unique Index وإنشاء شرطاً Not Null ضمنياً أيضاً على العمود أو الأعمدة المشروطة بشرط المفتاح الأساسي. يمكن اعتبار تعريف هذا الشرط كجزء من تعريف العمود أو جزء من تعريف الجدول، حيث إنه إذا تم تعريف هذا الشرط على عدة أعمدة (حالة مفتاح أساسي مركب) فإن تعريف هذا الشرط سيكون جزءاً من تعريف الجدول.

لكي نعرف الشرط Primary Key كجزء من تعريف عمود ما، نقوم بكتابة العبارة Primary Key بعد نوع بيانات الجدول كما في العبارة الآتية:

```
CREATE TABLE employee
        (emp_id NUMBER primary key,
    name        VARCHAR2 (15),
        Gender  char (1)
 Constraint gender_validation check (gender      in ('M','F')));
```

لكي نعرف الشرط Primary Key كجزء من تعريف الجدول، نقوم بكتابة العبارة Primary Key في أسفل تعريف الجدول، ويتم تعريفه بعد تعريف آخر عمود. ويجب وضع فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الأساسي كما في العبارة الآتية:

```
CREATE TABLE employee
        (emp_id NUMBER not null,
    F_name      VARCHAR2 (25),
         L_name         VARCHAR2 (15),
   Primary key(emp_id));
```

14.7. الشرط Foreign Key (المفتاح الغريب/ الأجنبي)

يُستخدم الشرط Foreign Key أداة للتأكيد أن الجداول المرتبطة لا تضمّ سجلات غير منتمية إلى سجل أب. يحتوي الشرط Foreign Key على خيار يسمح بحذف كل السجلات الأبناء المرتبطة مع سجل الأب عند حذف سجل الأب وهذا الخيار هو On Delete Cascade. يمكننا إنشاء مفتاح الربط الخارجي كجزء من تعريف العمود عند إنشاء الجدول، حيث يتم استخدام كلمة References ويليها اسم الجدول الذي يضمّ السجل الأب وذلك يكون بعد تعريف العمود. لسنا بحاجة لذكر اسم العمود من الجدول الأب، حيث إنه إذا لم تقم بكتابة اسم العمود الموافق من الجدول الأب يفترض نظام إدارة قواعد البيانات أنه هو المفتاح الأساسي.

في هذا المثال سنقوم بتعرف الشرط Foreign Key كجزء من تعريف عمود ما كما يأتي:

```
CREATE TABLE consultant_projects
 (emp_id number references consultant,
  project_name     varchar (25),
  complete_date    date);
Table created
```

لكي نعرف الشرط Foreign Key كجزء من تعريف الجدول، نقوم بكتابة العبارة Foreign Key في أسفل تعريف الجدول متبوعة باسم العمود المطبق علية الشرط ثم كلمة Reference ثم عمود المفتاح الرئيسي بالجدول الأب، ويتم تعريفه بعد تعريف آخر عمود. يجب وضع فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الخارجي كما في العبارة الآتية:

```
CREATE TABLE emp(
        empno   NUMBER(4),
        ename   VARCHAR2(10) NOT NULL,
        job     VARCHAR2(9),
        mgr     NUMBER(4),
        hiredate DATE,
        sal     NUMBER(7,2),
        comm    NUMBER(7,2),
        deptno  NUMBER(7,2) NOT NULL,
        CONSTRAINT emp_deptno_fk FOREIGN KEY(deptno)
        REFERENCES dept (deptno));
```

14.8. تعديل تعريف شرط ما (Modifying Constraints)

يمكننا إضافة شرط للجدول بعد إنشائه. كما يمكننا إزالة تلك الشروط أو إلغاء تفعيلها أو تفعيلها وذلك باستخدام Alter Table. يمتلك هذا الأمر عدة خيارات يمكن استخدامها مع الشروط وهي موضحة كالآتي:

Add (إضافة): يستخدم لإضافة شرطاً جديداً إلى الجدول.

Modify (تعديل): يستخدم لإضافة شرطاً إلى عمود موجود.

Drop (حذف) : يستخدم لحذف الشرط من الجدول.

Disable (إلغاء تفعيل): يسمح بإدخال البيانات دون النظر فيما إذا كانت تحقق الشرط أم لا، ولكن مع بقاء الشرط كما هو في قاموس البيانات.

Enable (تفعيل): يقوم بالتحقق من أن جميع البيانات التي يتم إدخالها والموجودة مسبقاً تحقق الشرط.

بعض الأمثلة اشرحها ؟!

```
ALTER TABLE     emp
  enable primary key;
 Table altered.
```

```
Alter table consultant_projects
Add primary key (emp_id,project_name);
```

```
ALTER TABLE     EMP
ADD CONSTRAINT emp_mgr_fk
FOREIGN KEY (mgr) REFERENCES emp (empno);
Table altered.
```

```
ALTER TABLE     emp
  2 disable primary key;
Table altered.
```

```
ALTER TABLE     emp
   drop primary key;
 Table altered.
```

Damascus University

أسئلة البحث:

1. اكتب جملة استعلام لعرض أسماء ووظائف وتواريخ تعيين وأرقام الموظفين بحيث يظهر رقم الموظف أولاً.
2. اكتب جملة استعلام لعرض وظائف الموظفين بدون تكرار.
3. اكتب جملة استعلام لعرض أرقام الموظفين وأسمائهم وظائفهم مع تغيير أسماء الأعمدة لتصبح (Employee-Num, Employee Name, Jobs).
4. اكتب جملة استعلام لعرض أسماء ورواتب الموظفين الذين رواتبهم أكبر من 5000، بحيث تكون النتيجة مرتبة تنازلياً بحسب الراتب.
5. اكتب جملة استعلام لعرض أسماء ورواتب الموظفين الذين رواتبهم ليست بين 1500 و 3000.
6. اكتب جملة استعلام لعرض أسماء ورواتب الموظفين الذين رواتبهم أكبر من 2000 و يعملون في القسم رقم 30، بحيث تكون السجلات مرتبة تنازلياً بحسب الراتب.
7. اكتب جملة استعلام لعرض أسماء الموظفين وعدد السنوات التي قضاها كل موظف في عمله.
8. اكتب جملة استعلام لتحويل السلسلة الحرفية FEBRUARY 23, 2016 إلى تاريخ.
9. اكتب جملة استعلام لعرض أعلى راتب والمتوسط الحسابي لرواتب الموظفين.
10. اكتب جملة استعلام لعرض الوظائف وعدد الموظفين في كل وظيفة.
11. اكتب جملة استعلام لعرض عدد المدراء.
12. اكتب جملة استعلام لعرض أسماء الوظائف ومكانها للموظفين الذين يعملون في القسم رقم 20 مع منع التكرار في الوظائف.
13. اكتب جملة استعلام لعرض أسماء وتواريخ تعيين الموظفين الذين يعملون في القسم نفسه الذي يعمل فيه الموظف Blake.

14. اكتب جملة استعلام لعرض أسماء الموظفين الذين يتقاضون رواتباً أكبر من المتوسط الحسابي لرواتب جميع الموظفين، مع ترتيب السجلات تنازلياً بحسب الراتب.
15. اكتب جملة استعلام لعرض أسماء ورواتب وأرقام الأقسام للموظفين الذين يعملون في القسم نفسه و لهم راتب الموظفين الذين يتقاضون العمولة نفسها.
16. اكتب جملة استعلام لعرض أسماء وتواريخ تعيين ورواتب الموظفين الذين لهم الراتب نفسه و عمولة الموظف Scott.

أضف بيانات الموظفين الآتية إلى جدول الموظفين، ثم تحقق من إتمام عملية الإضافة.

Empno	Ename	Job	MGR	Hiredate	Sal	COMM	Deptno
333	MhD	Eng	732	01-01-2016	4000	200	20
555	Jhon	Sales	754	21-01-2016	9000	300	30

17. قم بحفظ البيانات في الجدول السابق، ثم قم بتعديل راتب الموظف MhD ليصبح 17000.
18. قم بتعديل رواتب الموظفين الذين تقل رواتبهم عن 3000 لتصبح 10000، ثم تحقق من إتمام العملية بشكل جيد.
19. قم بحذف جميع بيانات الموظف رقم 555 (الذي تم إضافته في السؤال رقم 16).
20. قم بإنشاء جدول للأقسام يدعى Departments كما هو مبين فيما يأتي:

Column Name	ID	Name
Key Type		
Nulls/Unique		
FK		
Data Type	Number	Varchar2
Length	9	25

ثم قم باستعراض البناء الداخلي للجدول السابق.

21. قم بتعديل الجدول السابق ليصبح طول العامود Name 35 بدلاً من 25، ثم قم باستعراض البناء الداخلي للجدول.

22. قم بإنشاء جدول للأقسام باسم Departments يحتوي على رقم القسم، اسم القسم، عنوان القسم، رقم هاتف القسم، ثم قم بإضافة القيد primary key على عامود رقم القسم على مستوى الجدول.
23. قم بإنشاء جدول للموظفين باسم Employees يحتوي المعلومات اللازمة لتسجيل بيانات الموظفين، ثم قم بإنشاء القيد primary key على عمود رقم الموظف وذلك على مستوى الجدول و إضافة القيد unique على عمود اسم القسم و ذلك على مستوى العمود نفسه.
24. قم بإضافة القيد foreign key إلى جدول الموظفين بهدف ربطه مع جدول الأقسام.
25. أضف قيداً مناسباً لجعل رواتب الموظفين بين 3000 و 10000.
26. قم بعرض أسماء الأعمدة والقيود المطبقة على جدول الموظفين Employees.
27. قم بإنشاء جدول للموظفين باسم TheEmployees كما هو مبين فيما يأتي:

Column Name	Null	Type
Emp Num	Not Null	Number(3)
Emp Name		Varchar2(10)
Emp Job		Varchar2(9)
Emp MGR		Number(4)
Emp Hdate		Date
Emp Sal		Number(10,2)
Emp COMM		Number(10,2)
Emp Dept		Number(2)

الفصل السادس

كيفية تنصيب برنامج أوراكل اسكسبرس

Oracle Express Edition 11g Release2 Setup

تعدّ أوراكل اسكسبرس من أشهر منصات العمل بلغة SQL. يتم التنصيب من خلال تنزيل النسخة المضغوطة من موقع Oracle. بالدخول إلى موقع www.oracle.com نختار من القائمة Downloads ونجد Oracle Database Express Edition 11g Release 2 من قائمة Databases كما في الشكل الآتي:



بالنقر على رابط Oracle Database Express Edition تظهر الصفحة الموضحة في الشكل الآتي :



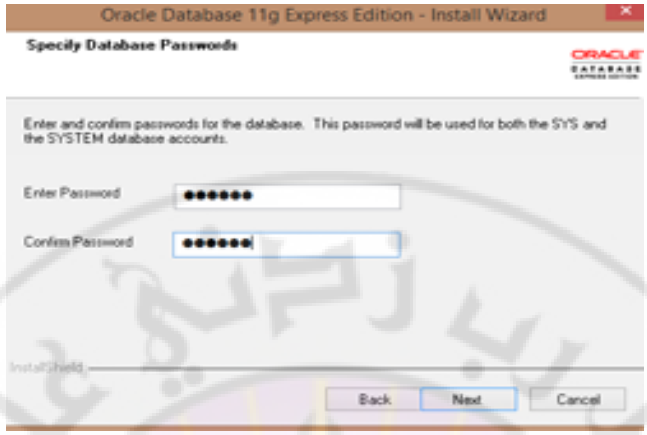
نختار الرابط الذي يوافق نظام التشغيل ومن ثم نقوم بفك ضغط الملف الذي تم تحميله ثم تشغيل الملف

Disk1/setup من المجلد OracleXE112_Win64

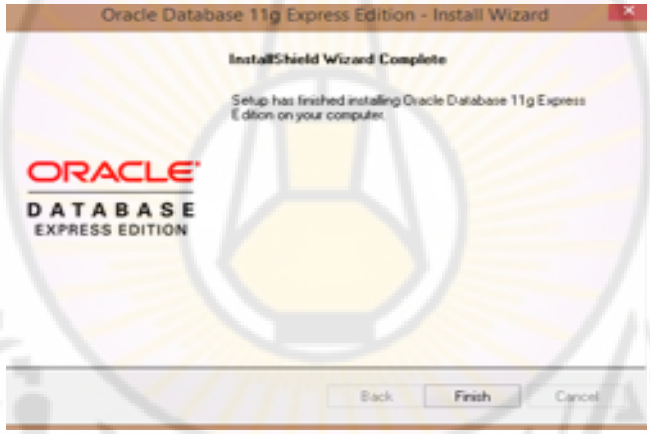


عندئذ يبدأ البرنامج بالتنصيب. نتابع مع توجيهات التنصيب حتى تظهر نافذة تطلب إدخال كلمة مرور خاصة بالنظام للمستخدم System .

نضع كلمة مرور خاصة بالنظام ولتكن system كما في الشكل الآتي :



بعد متابعة التوجيهات ينتهي التنصيب كما في الشكل الآتي :



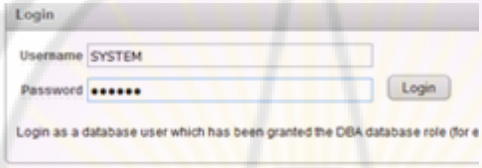
عند الانتهاء يتم إنشاء اختصار لملف تنفيذي كما في الشكل الآتي :



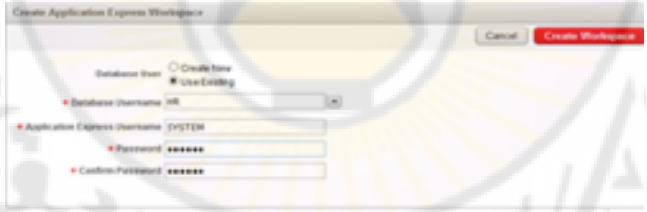
يتم تشغيل هذا الملف بالضغط المزدوج. عندئذ يتم فتح البرنامج على النافذة كما في الشكل الآتي :



نختار Application Express فتظهر نافذة الدخول إلى البرنامج كما في الشكل الآتي :

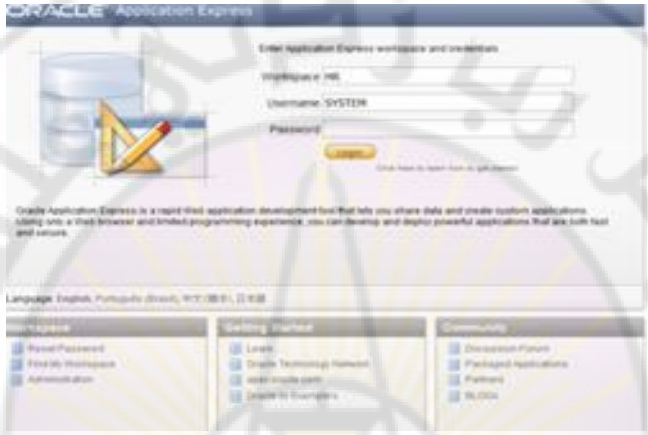


ندخل اسم المستخدم وكلمة المرور system ونختار اسم مستخدم وقاعدة بيانات موجودة مسبقاً ولتكن HR.



ندخل كلمة المرور ولتكن system[1] يتم الدخول إلى الصفحة الخاصة بقاعدة البيانات HR.

Enter Application Express workspace and credentials.



يتم الدخول إلى بيئة قاعدة البيانات كما في الشكل الآتي :



نختار SQL Workshop و منها نختار Commands SQL و من هنا يبدأ العمل كما في الشكل الآتي :

[1] نفس كلمة المرور التي فرضناها سابقاً.



هنا يتم تحرير تعليمات لغة الـ SQL. على سبيل المثال الشكل السابق يُظهر تحرير ثم تنفيذ العبارة :

```
Select * From Employees ;
```

Damascus University

المصطلحات العلمية

(عربي- إنكليزي)

الكينونة التجميعية	Associatetive Entity
مجال الواصفة	Attribute Domain
الواصفات	Attributes
العلاقة الثنائية	Binary Relationship
عدد الكينونات المشتركة بالعلاقة	Cardinality
عملية الجداء الديكارتي	Cartesian Product Operation
عملية الإتمام	Complement Operation
الارتباط الوظيفي التام	Complete Functional Dependency
قيد الإتمام	Completence Constraints
النموذج المفاهيمي	Conceptual Model
عملية الربط المشروط	Condiztion Join Operation
القيود	Constraints
لغات قواعد البيانات	Data Base Languages
نظام إدارة قواعد البيانات	**Data Base Management System (DBMS)**
نماذج أنظمة إدارة قواعد البيانات	Data Base Management System Models
مخطط قاعدة البيانات	Data Base Schema

Data Control Language (DCL)	لغة التحكم بالبيانات
Data Definition Language (DDL)	لغة تعريف البيانات
Data Diagram	مخطط البيانات
Data Manipulation Language (DML)	لغة التعامل مع البيانات
Data Model	نموذج البيانات
Database Object	كائن في قاعدة البيانات
Decomposition	تقسيم (تجزئة) العلاقات
Difference Operation	عملية الفرق
Direct Functional Dependency	الارتباط الوظيفي المباشر
Division Operation	عملية القسمة
Entity Type	نوع الكينونة
Equi Join Operation	عملية الربط المتساوي
Foreign Key	مفتاح العلاقة الثانوي
Functional Dependency	**الارتباط الوظيفي (التبعية الوظيفية)**
Generalization	التعميم
Identifier	المحدد
Inheritance	الوراثة
Integration	تكامل البيانات
Intersection Operation	عملية التقاطع
Join Dependency	الاعتماد الارتباطي

Join Operation	عملية الربط بين جدولين (الدمج)
Logical Model	النموذج المنطقي
Many-to-Many Binary Relationship	علاقة متعدد-إلى-متعدد
Multi Valued Dependency (MVD)	الارتباط متعدد القيم
Multiple-Row Subqueries	**الاستعلامات الفرعية من عدة صفوف**
Natural Join Operation	عملية الربط الطبيعي
Normalization	تنظيم قاعدة بيانات
One-to-Many Binary Relationship	علاقة واحد-إلى-متعدد
One-to-One Binary Relationship	علاقة واحد- إلى -واحد
Outer Join Operation	عملية الربط الخارجي
Physical Model	النموذج الفيزيائي
Primary Key	مفتاح العلاقة الرئيسي
Primary Key	مفتاح رئيس
Project Operation	عملية الإسقاط
Query	الاستعلام
Records (Occurrence)	السجلات
Recursive Relationship	العلاقة الدائرية
Redundancy	التكرار غير المفيد للبيانات
Referential Integrity	التكامل المرجعي
Relational Algebra	الجبر العلائقي

Relational Algebra Operations	عمليات الجبر العلائقي
Selection Operation	عملية الاختيار
Self Join Operation	عملية الربط الذاتي
Semi Join Operation	عملية الربط الجزئي
Single-Row Subqueries	**الاستعلامات الفرعية من صف واحد**
Specialization	التخصيص
Strong Entity	نوع كينونة قوي
Subclass	النوع الفرعي
Subqueries	الاستعلامات الفرعية
Superclass	النوع الأصلي
Ternary Relationship	العلاقة الثلاثية
The Boyce Codd Normal Form (BCNF)	الشكل النظامي لـ Boyce Codd
The Fifth Normal Form (5NF)	الشكل النظامي الخامس
The First Normal Form (1NF)	الشكل النظامي الأول
The Fourth Normal Form (4NF)	الشكل النظامي الرابع
The Group Functions	الدوال التجميعية
The Relational Model	النموذج العلائقي
The Second Normal Form (2NF)	الشكل النظامي الثاني
The Structure Query Language (SQL)	لغة الاستعلامات البنيوية
The Third Normal Form (3NF)	الشكل النظامي الثالث

Unary Relationship	العلاقة الأحادية
Union Operation	عملية الاجتماع
Weak Entity	نوع كينونة ضعيف



وقل رب زدني علما
جامعة دمشق
Damascus University

المراجع العلمية

[1] Fundamentals of Database Systems, 7th Edition, By Ramez Elmasri, Shamkant B. Navathe, 1990.

[2] Les Bases de Données Relationnelles, By André Flory, and Frédérique Laforest, 2000.

[3] Relational databases: Defining Relationships between Database Tables, By Susan Harkins, 2003.

[4] Oracle 11g: SQL by Joan Casteel, Cengage Learning 2009 .

[5] Database Systems Concepts, Designs and Application, By Shio Kumar Singh, 2ed Edition, 2011.

Damascus University

اللجنة العلمية

أ.م.د ريما القحة

م.د سمير جعفر

م.د غصون الجيرودي

المدقق اللغوي

د. نور الهدى حناوي

حقوق الطبع و الترجمة و النشر محفوظة لمديرية الكتب والمطبوعات

Damascus University Publications

Faculty of Science



MATHEMATICAL APPLIED

" INTRODUCTION TO DATABASE "

Dr.Mohamad Firas ALHALABI

and

Ayman Saleh MOHAMMAD

1438 – 1437

2017 – 2016

Damascus University