جامعة فلسطين التقنية – خضوري – طولكرم

قسم تكنولوجيا المعلومات

تخصص البرمجيات وقواعد البيانات

مساق قواعد بيانات ٢

مدرس المساق : أ. عبير قشوع

٢٠١٨

الوحدة الأولى : مراجعة قواعد البيانات العلائقية

ما هي قواعد البيانات العلائقية: Relational Database

١. قاعدة البيانات العلائقية :

قاعدة البيانات العلائقية هي مجموعة من الجداول والتي يمكن من خلالها الوصول إلى البيانات أو إعادة تجميعها بالعديد من الطرق المختلفة دون الحاجة إلى إعادة تنظيم جداول قاعدة البيانات. وتستخدم لغة الاستعلام الهيكلية (SQL) كواجهة للمستخدم وواجهة لبرمجة التطبيقات (API). ويتم استخدام عبارات SQL للاستعلامات التفاعلية للحصول على معلومات من قاعدة البيانات العلائقية وجمع البيانات للتقارير.

تتكون الجداول من مجموعة من الصفوف تسمى سجلات (records) في حين ان كل سجل يتكون من مجموع من الأعمدة تسمى حقول (fields) ولا توجد هناك أهمية لترتيب الصفوف أو الأعمدة وتمثل الأعمدة الصفات لهذه الجداول (attributes) ويجب أن يكون لكل صفة مجال (domain) من القيم التي يمكن أن يحتويها هذا العمود وترتبط هذه الجداول مع بعضها البعض بواسطة روابط ويجب أن يكون لكل جدول مفتاح رئيسي (primary key) لتمييز الصفوف عن بعضها والنقطة التي تمثل تقاطع الصف مع العمود (الصفة) تمثل قيمة لهذا الصف.

٢. مخطط البيانات: عبارة عن مفاهيم تستخدم في وصف DBMS من حيث البيانات والعلاقات والقيود .

١- مخطط مفاهيمي: تسمى بمخططات المستوى العالي وذلك لأنها تصف المفاهيم القريبة من إدراك المستخدم لها كالكائنات Entities وهي الكائنات المادية والمعنوية

٢-مخطط مادي: تسمى بمخططات المستوى الأدنى لأنها تصف طريقه تخزين البيانات.

٣-مخطط قواعد البيانات : توضيح المخططات هذه من خلال رسم تخطيطي Diagram Schema

٣. مستويات وصف قواعد البيانات

ولها ثلاث مستويات :

١- مستوى خارجي External level

٢- مستوى مفاهيمي Conceptual level أو منطقي Logical

٣- مستوى داخلي Internal level

١) **المستوى الخارجي:** حيث يتم التعامل والتخاطب مع قاعدة البيانات والمعلومات ، ويمثل هذا المستوى المستخدم النهائي للبيانات ، ويستطيع كل مستخدم أن يتعامل مع جزء من البيانات وبذلك يمكن تصور كل مستخدم بأنه ينظر من خلال شباك ليرى منظر (view) قد يختلف عن المنظر الذ يراه مستخدم آخر. ويتم الربط بين وحدات البيانات في هذا المستوى وبين وحدات المستوى المنطقي عن طريق برمجيات نظام قواعد البيانات، وهذا التخاطب والتعامل يتم من خلال برامج تطبيقية وبرامج رسوم أو بطرق مباشرة من خلال لغة الاسترجاع SQL ويتم استخدام آليات تقوم باستقبال وإرسال البيانات من هذه الآليات :

أ. **لغة تعريف البيانات "DDL" Data Definition language** : وهي مجموعة أوامر تستخدم لتعديل تركيب قاعدة البيانات عن طريق إنشاء، استبدال ، تغيير أو حذف كائنات مثل الجداول ، الفهارس والمناظر.

ب. لغة التخاطب والاسترجاع " DML" Data Manipulation Language : وهي مجموعة أولمر تستخدم لاسترجاع ، حذف ، تعديل وإضافة البيانات أو القوائم.

ج. الأنظمة الحديثة تستخدم Interface Graphical User: لغة تعريف الأشكال VDL (View Definition Language) وتستخدم مع نظم إدارة قواعد البيانات DBMS التي تستخدم هيكل قواعد البيانات الثلاثي بطريقة حقيقية وذلك لتعريف مخطط البيانات في المستوى الخارجي.

٢) **المستوى المفاهيمي أو المنطقي :** يقوم بوصف قاعدة البيانات التي سوف يتعامل معها المستخدمون فهو يمثل المنظر العام المنطقي لجميع البيانات وترابطها مع بعضها البعض، ويمكن تصور هذا المنظر بأنه التصميم العام وخارطة شاملة لجميع وحدات وأجراء البيانات وعلاقتها مع بعضها البعض، وعن طريق هذا المستوى يمكن ربط الأجزاء المنطقية مع الأجزاء الفعلية والذي يقوم بهذا الربط هو برمجيات نظام قواعد البيانات.

يهتم هذا المستوى بما يلي :

- وصف الكيانات
- البيانات وتحديد نوعها
- العلاقة بين الكيانات
- عمليات المستخدم
- قيود البيانات لتحقيق الأمان

٣) **المستوى الداخلي:** وهو الذي يصف أسلوب تخزين البيانات فعليا على وحدات التخزين الخارجية وهو المستوى الأقرب إلى وحدات التخزين. إن المعلومات المتوفرة في هذا الجزء تساعد نظام التشغيل في اختيار الأسلوب المطلوب لحفظ البيانات وأسلوب الوصول إليها.

وظائف المستوى الداخلي :

- معرفه أماكن تخزين البيانات والفهارس
- حفظ البيانات وتشفيرها
- وصف تركيب البيانات المخزنة فيها

٤. مفاتيح الجداول (العلاقات):

تعتبر من أهم خصائص قواعد البيانات العلائقية حيث إنها تكون المميزة لجدول معين من جهة والرابط الذي يربط الجداول المختلفة مع بعضها من جهة أخرى

١- **المفتاح الأعظم : (Super Key)** هو أقل مجموعة من الصفات التي يمكن أن تميز الصف في الجدول عن بقية الصفوف ، أي مجموعة من حقل أو أكثر تحدد بشكل فريد الصفوف في الجدول، مثال:

جدول الموظف

رقم الموظف	رقم الهوية	اسم الموظف

احتمالات المفتاح الاعظم:

- رقم الموظف
- رقم الهوية
- رقم الموظف ، رقم الهوية
- رقم الموظف ، اسم الموظف
- رقم الهوية ، اسم الموظف
- رقم الموظف ، رقم الهوية ، اسم الموظف

٢- **المفتاح المرشح : (Candidate Key)** وهو الصفة (مجموعة الصفات) التي يمكن اختيارها كمفتاح رئيسي للجدول ويجب أن يكون هنالك أكثر من صف له نفس القيمة لهذه الصفة أو الصفات وكذلك يجب أن يكون له قيمة أي ليس NULL حيث رقم الموظف في المثال السابق يعتبر مفتاحا مرشحا ليكون مفتاحا رئيسيا

٣- **المفتاح الرئيسي : (Primary Key)** هو المفتاح الذي تم اختياره من مجموعة المفاتيح المرشحة ليكون محددا لكل صف في الجدول ، يمكن أن نختار رقم الموظف ليكون مفتاحا رئيسيا

٤- **المفتاح الثانوي: (Secondary Key)** هو عبارة عن صفة أو صفات تستخدم لغايات الاسترجاع فمثلا لو كان لدينا جدول يحتوي علي قائمة بالعملاء فالمفتاح الرئيسي هو رقم العميل ولكن إذا أردنا أن نسترجع رقم هاتف عميل معين (حيث من سيحفظ أرقام العملاء؟؟؟) ففي هذه الحالة يتم استخدام الاسم في عملية البحث وليس رقم العميل ومن هنا يتم اختيار اسم العميل كمفتاح ثانوي

٥- **المفتاح الأجنبي : (Foreign Key)** وهو صفة أو صفات تشير إلي مفتاح رئيسي أو قيمة غير مكررة (unique) في جدول آخر فمثلا الصفة رقم القسم في جدول الموظف مفتاحا أجنبيا لجدول الأقسام

٥. العلاقات في قواعد البيانات العلائقية

أولاً - العلاقة (واحد إلى واحد) أو (One To One)

هذا النوع من العلاقة يعني أن بيانات سجل واحد من جدول معين له بيانات سجل واحد فقط من جدول أخر. فإذا كان لدينا جدولين (الأقسام ورؤساء الأقسام) فالمتعارف عليه أنه لكل قسم رئيس قسم واحد فقط

ثانياً – العلاقة (واحد إلى كثير) أو (One To Many)

وتسمى أيضا واحد إلى متعدد ... والأهم أن نفهم العلاقة وليس الأهمية للتسمية ...وهذه العلاقة هي الأكثر استخداماً في قواعد البيانات وهي تعني أنه لكل سجل من سجلات الجدول الأول والذي يكون عادة يحتوي على (مفتاح أساسي) له أكثر من سجل أو عدة سجلات في الجدول الأخر الذي عادة (يحتوي مفتاح ثانوي)

ثالثا – العلاقة (كثير إلى كثير) أو(Many To Many)

وهي العلاقة الأكثر تعقيداً والأقل استخداماً وهي تستخدم عندما يكون لدينا جدولين ونريد أن يكون لكل سجل في الجدول الأول عدد من السجلات في الجدول الثاني وكذلك العكس أن يكون للجدول الثاني عدد من السجلات في

الجدول الأول (وهنا تكون العقدة لذا يجب عمل جدول (وسيط) يحتوي على (مفتاحين ثانويين) فقط ونقوم بربط كل (مفتاح رئيسي) من الجدولين (الأول و الثاني) بالجدول (الوسيط) بعلاقة (واحد إلى كثير) بين كل من الجدولين الرئيسين والجدول الوسيط وبذلك سنتمكن من ربط بيانات الجدولين الرئيسيين ويكون لكل سجل عدد من السجلات في الجدول الأخر، مثلا لو كان عندنا جدولين الأول (الأساتذة) والثاني (الصفوف) فمن المعروف أن كل أستاذ يدرس أكثر من صف وأن كل صف يقوم بتدريسه أكثر من أستاذ لذلك يجب عمل جدول (وسيط) لإتمام هذه العلاقة

٦. الكيانات Entities وخصائصها :

١- *Entity (المدخلة أو كينونة البيانات أو العينة)* :- وهو عنصر من الحياة الواقعية يتم تخزين معلومات عنه . مثل : الموظف أو المشروع ، وقد تعطى قيمة هذه الكينونة بطرق مختلفة كان تحدد قيمتها أو تعطى أسماء أو يحدد لونها أو طولها وهو نوعان :-

أ) هدف محدد ملموس : كالموظف أو القطعة أو المكان .

ب) هدف غير محدد غير ملموس : كالعنوان والرقم والاسم .

٢- *محددات البيانات (الصفة) Attribute* :- وهي كلمات تمثل مجموعة من الخصائص والصفات المهمة التي يتم تخزينها عن الكينونة كاسم الموظف والراتب .

الوحدة الثانية : معالجة البيانات ولغة SQL

لغة التساؤل البنيوى (SQL) (Structured Query Language) :-

لغة التساؤل عبارة عن لغة أعلى مستوى من لغات البرمجة العالية ، تستخدم في توجيه الأسئلة للحصول على المعلومات المطلوبة من قاعدة البيانات ، لذا فهي لغة شاملة إذ تضم أوامر لتعريف البيانات ، للتساؤل وللاستعلامات وللتحديث على قاعدة البيانات ، كما أنها تحتوي على إمكانيات لإنشاء المناظر Views في قاعدة البيانات وكذلك لفهرسة الملفات (الجداول) .

- **أقسام لغات التساؤل :-**

تنقسم لغات التساؤل إلى قسمين هما :-

أ- اللغات الخوارزمية :- حيث يبرمج المستفيد التعليمات للنظام والذي بدوره يستجيب لهذه التعليمات ويحسب النتائج المطلوبة .

ب- اللغات غير الخوارزمية :- يصف المستخدم الوظيفة المطلوبة بدون إصدار تعليمات معينة عن كيفية تنفيذها ، وتوفر معظم نظم إدارة قواعد البيانات لغات تساؤل تضم القسمين في وقت واحد .

إنّ من أشهر لغات التساؤل هي لغة التساؤل البنيوي SQL ، وهي من أقوى اللغات المتوفرة ، وقد تم تطوير عدة طرازات من هذه اللغة منذ أوائل عقد السبعينيات ، وفي سنة ١٩٨٦ م أعدت نسخة قياسية من لغة التساؤل البنيوي.

الاستعلامات في لغة SQL :-

تحتوي لغة SQL على جملة استعلامية واحدة أساسية لاسترجاع المعلومات من قاعدة البيانات وهي جملة Select ، ، ولجملة Select في SQL خيارات كثيرة جداً.

الصيغة العامة لجملة Select :-

Select أسماء الحقول المراد استرجاعها
From أسماء الجداول التي تحتوي الحقول المطلوبة
Where الشرط المراد تحققه حتى تعرض الحقول

ملاحظة :- يمكن إلغاء قسم Where وعندها يتم العرض لجميع صفوف الجدول لأنه لا يوجد شروط ، أما إذا كان لدينا جدول وأردنا عرض قيم جميع الأعمدة فإننا نكتفي بوضع إشارة (*) خلف كلمة Select بدلاً من كتابة أسماء جميع الحقول .

أمثلة على جملة Select :-

بالاعتماد على الجداول التالية :-

الموظفون Employee

SSN رقم الموظف	Fname الاسم الأول	Mint الاسم الأوسط	Lname اسم العائلة	Bdate تاريخ الميلاد	Address العنوان	Salary الراتب	Superssn رقم المشرف	Dno رقم القسم
1234567	Ali	Basem	Omar	9/1/175	Tulkarm	3000	3334455	5
3334455	Sami	Tamer	Saleh	8/12/1969	Tulkarm	4000	6668844	5
9988877	Rana	Jamal	Omar	19/7/1978	Nablus	2500	1234567	4
7654321	Huda	Sami	Abdullah	20/6/1971	Jenin	4300	3334455	4
6668844	Ramiz	Khaled	Salman	15/9/1972	Hebron	3800	3334455	5
4534534	Bader	Ahmad	Ibrahim	31/7/1982	Tulkarm	2500	9988877	5
8866655	Ahmad	Yousef	Sameer	29/3/1977	Tulkarm	2500	Null	1

القسم Department

Dname اسم القسم	Dnumber رقم القسم	MgrSSN رقم مدير القسم	Mgrstartdate تاريخ استلام المدير
Research	5	3334455	22/5/2016
Administration	4	9988877	1/1/2009
Accounting	1	8866655	19/6/2010

مواقع الأقسام Dept_locations

Dnumber رقم القسم	Dlocation الموقع
1	Tulkarm
4	Nablus
5	Hebron
5	Nablus
5	Tulkarm

المشروع Project

Pname اسم المشروع	Pnumber رقم المشروع	Plocation موقع المشروع	Dnum رقم القسم
ProductX	1	Tulkarm	5
ProductY	2	Nablus	5
ProductZ	3	Hebron	5
Computerization	10	Nablus	4
Reorganization	20	Tulkarm	1
New benefits	30	Nablus	4

يعمل في Works_on

ESSN رقم الموظف	Pno رقم المشروع	Hours عدد الساعات
1234567	1	32.5
1234567	2	7.5
6668844	3	40
4534534	1	20
4534534	2	20
3334455	2	10
3334455	3	10
3334455	10	10
3334455	20	30
9988877	30	10
9988877	10	35
8866655	20	10

الأقارب (المعالين) Dependent

ESSN رقم الموظف	Dependent_Name اسم القريب	Bdate تاريخ الميلاد	Relationship صلة القرابة
3334455	Samar	5/4/1996	Daughter
3334455	Basel	25/1/1999	Son
3334455	Sana'	3/5/1948	Parent
9988877	Ali	29/2/1945	Parent
1234567	Hani	1/1/2005	Son
1234567	Dalal	31/10/2010	Daughter
1234567	Iman	5/5/57	Parent

الأمثلة :-

١- استرجاع تاريخ الميلاد (Bdate) والعنوان (Address) للموظف الذي اسمه الأول (Fname) هو Ali واسمه الأوسط (Mint) هو Basem واسم العائلة له (Lname) هو Omar .

الحل:-

```
Select Bdate , Address
From Employee
Where Fname = 'Ali' and Mint = ' Basem ' and Lname = 'Omar'
```

٢- استرجاع الاسم الأول (Fname) ، اسم العائلة (Lname) والعنوان (Address) لكل الموظفين الذين يعملون في دائرة البحث (أي أن اسم الدائرة (Dname) هو Research).

```
Select Fname , Lname , Address
From Employee , Department
Where Dname = 'Research' and Dnumber = Dno
```

٣- استرجاع رقم المشروع (Pnumber) ، رقم القسم المستلم للمشروع (Dnum) ، الاسم الأخير لمدير القسم المستلم للمشروع (Lname) ، عنوان المدير (Address) ، وتاريخ ميلاد المدير (Bdate) وذلك لكل مشروع يقع في (Plocation) مدينة Nablus .

```
Select Pnumber , Dnum , Lname , Address , Bdate
From Project , Department , Employee
Where Plocation = ' Nablus ' and Dnum = Dnumber and Mgrssn = SSN
```

- **التعامل مع الحقول ذات الأسماء المتشابهة :-**

بما أن اسم الحقل قد يتشابه بين جدولين ، فإنه يجب التفريق بين هذه الحقول عند استخدامها في الاستعلامات وذلك بكتابة اسم الجدول متبوعاً بنقطة ثم اسم الحقل المطلوب لمنع التعارض بين أسماء الحقول.

مثال١:- استرجاع اسم القسم ، رقم القسم ورقم مدير القسم لكل قسم يقع في Tulkarm .

```
Select Dname , Dnumber , mgrssn
From Department , Dept_locations
Where Dlocation = 'Tulkarm' and Department.Dnumber = Dept_locations.Dnumber
```

لاحظ ان حقل Dnumber متشابه في الاسم بين جدولي Department و Dept_locations ، ولذا أضفنا اسم الجدول أمام كل منهما.

كما يظهر التكرار أيضاً في الاستعلامات التي تعود على نفس الجدول (تستخدمه) مرتين كالمثال التالي :-

- **عدم تحديد قسم Where واستخدام * :-**

عند إنشاء استعلام وعدم استخدام قسم Where فإن ذلك يعني عدم وجود شرط على اختيار السجلات وبالتالي فإن جميع السجلات في الجدول المذكور في جملة From ستظهر في النتيجة ، وإذا ظهر في قسم From أكثر من اسم جدول بدون وجود جملة Where فإن الضرب الكارتيزي للجدولين هو الناتج .

أمثلة :-

١- عرض أرقام جميع الموظفين .

```
Select SSN
From Employee
```

٢- اختيار كل المصاحبات بين رقم الموظف واسم القسم .

```
Select SSN , Dname
From Employee , Department
```

ملاحظة :- إذا أردنا استرجاع جميع الحقول (الأعمدة) لجدول أو أكثر فإننا فقط نضع (*) في قسم Select والتي تعني جميع الحقول .
أمثلة :-

١- استرجاع جميع الحقول لكل موظف يعمل في القسم رقم ٥ .

```
Select *
From Employee
Where Dno = 5
```

٢- عرض كل الحقول لكل موظف وكذلك كل الحقول للقسم الذي يعمل فيه هذا الموظف ، وذلك لكل موظف يعمل في قسم البحث Research .

```
Select *
From Employee , Department
Where Dname = 'Research' and Dno = Dnumber
```

*** ملاحظة :-**

إن لغة SQL لا تعامل الجدول كمجموعة رياضية (Set) ، إذ أن الصفوف (السجلات) قد تظهر أكثر من مرة (تتكرر) في نتيجة الاستعلام وبالتالي فإن الجدول الناتج يخالف المجموعة والتي لا تسمح بالتكرار لعناصرها .

إن لغة SQL لا تحذف التكرار بشكل تلقائي في نتائج الاستعلامات وذلك للأسباب التالية :-

١) إن حذف التكرار من الجداول هو عملية مكلفة ، وإحدى الطرق المستخدمة للإلغاء هي ترتيب السجلات أولاً ثم حذف التكرار .
٢) قد يكون المستخدم بحاجة لرؤية السجلات المتكررة في نتيجة الاستعلام .
٣) عند استخدام الاقترانات التجميعية (كالمعدل مثلاً) فإننا لا نريد حذف التكرار لأننا بحاجة إليه .

ولذا للتخلص من السجلات المكررة في لغة SQL ، فإننا نستخدم كلمة **Distinct** في قسم Select والتي تعني أن السجلات المميزة فقط هي التي ستظهر في النتيجة وبالتالي تصبح النتيجة جدول تنطبق عليه شروط المجموعة الرياضية .

مثال :- استرجاع قيمة الراتب لكل موظف.

```
Select Salary
From Employee
```

والتي ستظهر جميع رواتب الموظفين حتى المكررة منها ، ولإلغاء تكرار القيم نكتب :-

```
Select Distinct Salary
From Employee
```

- **بعض العمليات المعرفة على لغة SQL :-**

١- الاتحاد Union
٢- التقاطع Intersect
٣- الطرح (الفرق) Except

والناتج من هذه العمليات عبارة عن مجموعات من السجلات أي أن التكرار يلغى منها (إلا إذا كانت العملية متبوعة بكلمة All) وقبل إتمام أي من العمليات السابقة ، يجب التأكد أن الجدولين المراد تنفيذ العملية عليهما يحويان نفس الحقول والتي تظهر أيضاً بنفس الترتيب.

أمثلة :-

١- إنشاء قائمة بأرقام المشاريع التي يشتغل فيها موظف اسمه الأخير Omar سواء كموظف عادي أو كمدير للدائرة التي تتحكم في المشروع .

```
( Select Pnumber
  From Project , Department , Employee
  Where Dnum = Dnumber and mgrssn = Ssn and lname = 'Omar' )
Union
( Select Pnumber
  From Project , Works_on , Employee
  Where Pnumber = Pno and Essn = Ssn and lname = 'Omar' )
```

لاحظ ان الجزء الأول من الاستعلام يسترجع أرقام المشاريع التي تتضمن الموظف الذي اسم عائلته Omar كمدير للدائرة التي تدير المشروع ، بينما الجزء الثاني من الاستعلام يسترجع أرقام المشاريع التي تتضمن الموظف الذي اسم عائلته Omar كموظف عامل في المشروع .

- **تداخل (تتابع) الاستعلامات ومقارنة المجموعات :-**

إن بعض الاستعلامات تحتاج أن تفحص القيم الموجودة حالياً في قاعدة البيانات ثم تستخدم في شرط مقارنة ، ومثل هذا النوع من الاستعلامات تنفذ بشكل متداخل (متتابع) حيث نتبع قسم Where في الجملة الأولى بقسم Select في الجملة الثانية.

ملاحظة :- يمكن حل المثال السابق باستخدام فكرة التداخل كالتالي :-

```
Select Distinct Pnumber
From Project
Where Pnumber IN ( Select Pnumber
                   From Project , Department , Employee
                   Where Dnum = Dnumber and mgrssn = Ssn and lname = "Omar' )
              OR
    Pnumber IN ( Select Pno
                   From Works_on , Employee
                   Where Essn = Ssn and lname = "Omar' )
```

لاحظ ان الجزء الأول من الاستعلام يسترجع أرقام المشاريع التي تتضمن الموظف الذي اسم عائلته Omar كمدير للدائرة التي تدير المشروع ، بينما الجزء الثاني من الاستعلام يسترجع أرقام المشاريع التي تتضمن الموظف الذي اسم عائلته Omar كموظف عامل في المشروع .

مثال :-اختيار أرقام الموظفين الذين يعملون في نفس المشاريع التي يعمل فيها الموظف الذي رقمه (1234567)

```
Select Distinct Essn
From Works_on
Where Pno  In ( Select Pno
                From Works_on
                Where ssn = '1234567' )
```

ملاحظة :- عملية (IN) تقوم بمقارنة السجلات بين القوسين بمجموعة سجلات أخرى ، كما يمكن اختيار عمليات أخرى للمقارنة مثل :- < > , = , <= , >= , < , > <

مثال :- استرجاع الاسم الأول Fname ، اسم العائلة Lname ، لكل الموظفين الذين يحصلون على راتب أعلى من رواتب موظفي القسم رقم ٥ .

```
Select Fname , Lname
From Employee
Where Salary > All ( Select Salary
                     From Employee
                     Where Dno = 5 )
```

ملاحظة :- يمكن استخدام عبارة Exists لفحص فيما إذا كانت نتيجة استعلام داخلي فارغة (لا تحتوي على قيم) وتشبهها أيضاً عبارة Not Exists .

مثال : استرجاع أسماء الموظفين الذين ليس لديهم أقارب (أشخاص معتمدين) .

```
Select Fname , Lname
From Employee
Where Not Exists ( Select *
                   From Dependent
                   Where ESsn = Ssn  )
```

ويقوم هذا الاستعلام بما يلي :-لكل سجل موظف ، يقوم الجزء الداخلي من الاستعلام باختيار جميع سجلات المعالين التي يتطابق فيها حقل Essn مع حقل ssn في سجل الموظف ، وإذا كانت نتيجة هذا الجزء الداخلي فارغة فإن ذلك يعني عدم وجود معالين للموظف وبالتالي يتم اختيار هذا الموظف في النتيجة .

ملاحظة :-

- جملة Exists تعيد القيمة True إذا كان هناك على الأقل سجل واحد ينتج من الاستعلام خلفها وإلا فإنها تعيد القيمة False .
- جملة Not Exists تعيد القيمة True إذا لم يكن هناك أي سجل ينتج من الاستعلام خلفها وإلا فإنها تعيد القيمة False .

مثال٢ :- استرجاع أسماء المدراء (مدراء الأقسام) الذين لديهم على الأقل شخص معال (قريب) واحد .

```
Select Fname , Lname
From Employee
Where Exists ( Select *
        From Dependent
        Where Ssn = Essn )
  And Exists ( Select *
        From Department
        Where Ssn = mgrssn )
```

ويمكن كتابتها أيضاً كالتالي :-

```
Select Fname , Lname
From Employee
Where Exists ( Select *
        From Dependent , Department
        Where Ssn = Essn and Ssn = mgrssn )
```

- **استخدام المجموعة الضمنية في قسم Where وتعبير Null :-**

مثال : استرجاع أرقام الموظفين الذين يعملون في المشاريع ذات الأرقام (١ ، ٢ ، ٣)

```
Select Distinct Essn
From Works_on
Where Pno In ( 1 , 2 , 3 )
```

كما تسمح SQL للاستعلامات بفحص القيم الفارغة باستخدام (Null) وذلك بدلاً من استخدام = أو ≠ لمقارنة قيمة حقل بقيمة Null لذا فهي تستخدم (Is Null) أو (Is Not Null)

مثال : استرجاع أسماء الموظفين الذين ليس لديهم مشرف.

```
Select Fname , Lname
From Employee
Where Superssn Is Null
```

- **التجميع Grouping والاقترانات التجميعية Aggregat Functions :-**

هناك بعض العمليات التي يحتاجها المستخدم حيث قامت SQL باعتبارها وصممت لها اقترانات جاهزة مثل :-

- اقتران ايجاد العدد Count
- اقتران ايجاد المجموع Sum
- اقتران ايجاد أعلى قيمة Max
- اقتران ايجاد أدنى قيمة Min
- اقتران ايجاد المعدل Avg

مثال١ : استرجاع مجموع رواتب الموظفين ، أعلى راتب ، أدنى راتب ، معدل رواتب الموظفين.

```
Select Sum(Salary) , Max(Salary) , Min(Salary) , Avg(Salary)
From Employee
```

مثال٢ : استرجاع مجموع رواتب موظفي دائرة البحث Research .

```
Select Sum(Salary)
From Employee , Department
Where Dname = 'Research' and Dno = Dnumber
```

مثال٣ : استرجاع عدد موظفي الشركة.

```
Select Count(*)
From Employee
```

مثال٤ : استرجاع عدد موظفي دائرة البحث Research

```
Select Count(*)
From Employee , Department
Where Dname = 'Research' and Dno = Dnumber
```

ملاحظة :- (*)Count تعني عدد الصفوف في الجدول الناتج من الاستعلام.

مثال٥ : إيجاد عدد قيم الرواتب الفريدة (غير المتكررة)

```
Select Count ( Distinct Salary )
From Employee
```

مثال٦ : استرجاع أسماء الموظفين الذين لديهم معالَيْن (أقارب اثنين) أو أكثر.

```
Select Fname , Lname
From Employee
Where ( Select Count (*)
        From Dependent
        Where Ssn = Essn ) >=2
```

- هناك بعض الاستعلامات التي تحتاج لتنفيذ العملية التجميعية على سجلات معينة فقط ، لذا نحتاج إلى تجميع السجلات حسب حقل ما باستخدام جملة Group by ثم ننفذ العملية التجميعية على كل مجموعة.

مثال١ :- لكل قسم من أقسام الشركة يراد استرجاع رقم القسم ، عدد الموظفين في القسم ، معدل رواتبهم.

```
Select Dno , Count(*) , Avg(Salary)
From Employee
Group By Dno
```

مثال٢ : لكل مشروع ، يراد استرجاع رقم المشروع ، اسم المشروع وعدد الموظفين العاملين فيه.

```
Select Pnumber , Pname , Count(*)
From Project , Works_On
Where Pnumber = Pno
Group By Pnumber , Pname
```

- تضم لغة SQL عبارة Having والتي يمكن أن تظهر مرتبطة بجملة Group By ، ولذلك فإن جملة Having توفر شرط على مجموعة السجلات المصاحبة لكل قيمة من حقول التجميع وبالتالي فإن المجموعات المحققة للشرط فقط هي التي تظهر في النتيجة.

مثال١ : لكل مشروع يشتغل فيه أكثر من موظفين اثنين ، قم باسترجاع رقم المشروع ، اسم المشروع ، وعدد الموظفين العاملين فيه .

```
Select Pnumber , Pname , Count(*)
From Project , Works_On
Where Pnumber = Pno
Group By Pnumber , Pname
Having Count(*) > 2
```

مثال٢ : لكل مشروع ، قم باسترجاع رقم المشروع ، اسم المشروع ، وعدد الموظفين من قسم رقم ٥ الذين يعملون في هذا المشروع.

```
Select Pnumber , Pname , Count(*)
From Project , Works_On , Employee
Where Pnumber = Pno and Ssn = Essn and Dno = 5
Group By Pnumber , Pname
```

ملاحظة : يجب أن نكون على حذر كبير عندما يكون لدينا شرطان مختلفان (أحدهما على الاقتران التجميعي في جملة Select ، والآخر على الاقتران في جملة Having).

مثال : إيجاد عدد الموظفين الذين تزيد رواتبهم عن ٤٠٠٠ شيكل في كل قسم ، لكن فقط للأقسام التي يعمل فيها أكثر من ٥ موظفين.

الحل : هنا الشرط (الراتب أكبر من ٤٠٠٠) يطبق فقط على اقتران العدد Count في جملة Select ، لذا فالحل بالشكل التالي :-

```
Select Dname , Count(*)
From Department , Employee
Where Dnumber = Dno and Salary > 4000 and
    Dno In ( Select Dno
             From Employee
             Group By Dno
             Having Count(*) > 5 )
   Group By  Dname
```

- **استخدام عبارة Like :-**

في عمليات المقارنة أو البحث وعند عدم معرفة المستخدم بالقيمة المطلوبة كاملة فإنه يمكن استخدام عبارة Like متبوعة بالجزء المعلوم من القيمة مع استخدام إشارة (%) لتدل على عدد غير محدد من الأحرف المجهولة ، بينما إشارة (-) تدل على حرف واحد فقط مجهول .

مثال١ : استرجاع أسماء الموظفين الذين يحتوي عنوانهم على الحرف H .

```
Select Fname , Lname
From Employee
Where Address Like '%H%'
```

مثال٢ : استرجاع أسماء الموظفين الذين الحرف الثالث من اسمهم الأول هو L

```
Select Fname , Lname
From Employee
Where Fname Like '--L%'
```

- **استخدام العمليات الحسابية في الاستعلامات :**

يمكن استخدام جميع العمليات الحسابية كالجمع والطرح والضرب والقسمة لحساب أي قيمة شرط أن تطبق على قيم عددية.

مثال : لكل الموظفين العاملين في المشروع (ProductX) يراد زيادة رواتبهم بمقدار ١٠% ، قم بإنشاء استعلام يعيد اسم الموظف وقيمة راتبه بعد الزيادة.

```
Select Fname , Lname , 1.1 * Salary
From Employee , Works_On , Project
Where Ssn = Essn and Pno = Pnumber and Pname = ' ProductX'
```

- **ترتيب وفرز السجلات في لغة SQL :-**

تسمح لغة SQL للمستخدم بترتيب السجلات الناتجة من الاستعلام حسب قيمة حقل أو أكثر وذلك باستخدام عبارة (Order By)

مثال : استرجاع أسماء الموظفين وأسماء الأقسام التي يعملون بها وأسماء المشاريع التي يعملون فيها مرتبة حسب اسم القسم تصاعديا، وضمن كل قسم تكون الأسماء مرتبة أبجدياً حسب اسم العائلة ثم حسب الاسم الأول.

```
Select Dname , Lname , Fname , Pname
From  Department , Employee , Works_On , Project
Where Dnumber = Dno and Ssn = Essn and Pno = Pnumber
Order By Dname , Lname , Fname
```

والترتيب عادة يكون تصاعدي (Ascending) بشكل تلقائي لكن إذا أردنا تنازلي (Descending) نتبع اسم الحقل في جملة (Order By) بكلمة (Desc) أما إذا أُتبع بكلمة (Asc) فتدل على الترتيب التصاعدي.

مثال : عرض اسم الموظف وراتبه مرتبة تنازلياً حسب الاسم الأول .

```
Select Fname , Lname , Salary
From Employee
Order By `Fname Desc
```

- **جمل التحديث في لغة SQL (Update Statements) :-**

١- جملة الإدخال Insert :

وتستخدم لإضافة سجل جديد إلى قاعدة البيانات ، ويجب تحديد اسم الجدول المراد الإضافة إليه وقيم حقول السجل الجديد ، والقيم يجب أن تكون مرتبة بنفس الطريقة التي ذكرت فيها في الجدول.

مثال : إضافة سجل جديد لجدول Employee وبياناته كالتالي.

```
Insert into Employee
Values ('5679871' , 'Saleh' , 'Khalil' , 'Omar' , '02/04/52' , 'Nablus' , 3700 , '1234567' , 4 )
```

ملاحظة : يمكن تعبئة حقول معينة في جملة Insert وترك الحقول الأخرى فارغة لكن يجب في هذه الحالة تحديد أسماء الحقول التي سيتم إدخال قيمها.

مثال : إدخال سجل لموظف جديد معلوم فقط رقمه ، اسمه الأول ، اسم العائلة له .

```
Insert into Employee (Fname , Lname , Ssn)
Values ( 'Huda' , 'Adel' , '6565432')
```

ملاحظة : عند إدخال سجل جديد يجب مراعاة عدم ترك حقول المفاتيح أو الحقول المطلوبة فارغة دون قيم .

ملاحظة : يمكن استخدام جملة Insert واحدة لإدخال عدة سجلات ، كما يمكن أن تكون السجلات المراد إدخالها ناتجة من استعلام.

مثال : إذا كان لدينا جدول اسمه Depts يحتوي على الحقول التالية :-

(Dname , No_of_Emps , Totalsal)

وأردنا بداخله وضع اسم كل قسم ، عدد الموظفين العاملين فيه ، مجموع رواتبهم .

```
Insert Into Depts (     Dname , No_of_Emps , totalsal )
                    Select Dname , Count(*) , Sum(Salary)
                    From Department , Employee
                    Where Dnumber = Dno
                    Group By Dname )
```

٢- جملة الحذف Delete :-

تستخدم لإزالة الصفوف من الجدول ، وتتضمن قسم Where الذي يستخدم في جملة Select وذلك لتحديد الصفوف المراد حذفها ، والصفوف تحذف تلقائياً من جدول واحد في نفس اللحظة وربما تحذف من جداول أخرى إذا تم تحديد ذلك في الشروط والقيود عند تعريف الجداول ، وبالاعتماد على عدد الصفوف التي ينطبق عليها شرط Where ، فإن (صفر) أو (١) أو عدة سجلات تحذف بواسطة جملة Delete واحدة ، وفي حالة عدم وجود قسم Where فإن ذلك يعني حذف جميع سجلات الجدول المذكور بلا استثناء ، بينما يبقى الجدول في قاعدة البيانات كجدول فارغ .

أمثلة :-

١- حذف سجلات الموظفين الذين يحملون الاسم الأخير Omar .

```
Delete From Employee
Where Lname = 'Omar'
```

٢- حذف سجل الموظف ذو الرقم ١٢٣٤٥٦٧

```
Delete From Employee
Where Ssn = '1234567'
```

٣- حذف سجلات جميع الموظفين العاملين في دائرة البحث Research .

```
Delete From Employee
Where Dno In ( Select  Dnumber
               From Department
               Where Dname = 'Research' )
```

٣- جملة التعديل Update :

وتستخدم لتعديل قيم الحقول لصف أو أكثر ، وكما في جملة Delete ، تحتوي جملة Update على قسم Where لاختيار السجلات التي سيتم تعديلها من جدول واحد ، والتعديل على قيمة المفتاح الرئيسي في جدول ما قد ينتقل إلى الجداول الأخرى التي تحتويه كمفتاح أجنبي إذا تم تحديد ذلك في الشروط والقيود عند تعريف الجداول ، وهناك قسم آخر جديد يسمى (Set) يحدد الحقول التي ستتغير قيمتها وقيمتها الجديدة.

الصيغة العامة :-

```
Update اسم الجدول المطلوب
Set  اسم الحقل المراد تعديله = القيمة الجديدة
Where الشروط التي ستتحقق ليتم التعديل
```

أمثلة :

١- تغيير الموقع ورقم القسم المشرف للمشروع رقم 10 ليصبح Hebron وَ 5 .

```
Update Project
Set Plocation = 'Hebron' , Dnum = 5
Where Pnumber = 10
```

- يمكن تغيير قيمة حقل ما في عدة سجلات بجملة Update واحدة.

مثال : منح كل موظفي دائرة البحث Research زيادة على رواتبهم بنسبة ١٠% من الراتب الأصلي .

```
Update Employee
Set Salary = Salary * 1.10
Where Dno In ( Select Dnumber
               From Department
               Where Dname = 'Research' )
```

- يمكن أيضاً تعيين قيمة (Null) أو قيمة (Default) كقيمة جديدة للحقل ، لاحظ ايضاً أن جملة Update تحدد جدول واحد فقط ، لتعديل عدة جداول ، يجب إصدار عدة جمل Update .

٤- جملة الإنشاء Create :-

تستخدم جملة Create Table لإنشاء جدول جديد بتحديد اسمه وحقوله والقيود عليه ، يتم تحديد الحقول أولاً حيث يعطى لكل حقل اسم خاص به ونوع بيانات لتحديد مجال قيمه وبعض القيود الأخرى .

٥- جملة الحذف Drop :-

وذلك لحذف قاعدة البيانات أو لحذف جدول من جداولها

مثال :

```
Drop schema company Cascade
```

يحذف قاعدة البيانات Company وكل جداولها ومجالات القيم وكل العناصر الأخرى وذلك لاستخدام عبارة Casade ، ويمكن بدلاً منها استخدام عبارة Restrict حيث تحذف قاعدة البيانات فقط إذا لم تحتوي على أي عنصر

أمثلة :-

١- حذف جدول Dependent وكل ما يخصه من تعريفات

```
Drop Table Dependent Cascade
```

٢- حذف جدول Dependent لكن فقط إذا لم يكن مرتبطاً بجداول أخرى .

```
Drop Table Dependent Restrict
```

الوحدة الثالثة :إدارة ومكونات قواعد البيانات

وظائف إدارة قواعد البيانات Database Administration Functions

يمكن تلخيص الوظائف الاساسية لوحدة إدارة قواعد البيانات بما يلي:

١- **الاحتفاظ بالتصميم المنطقي (المخطط) لقواعد البيانات Logical Schema وإجراء أي تعديل عليه**. إن هذا التصميم يتم تعريفه بواسطة لغة الوصف DDL ويقوم برنامج الترجمة الخاص بتلك اللغة بتحويل هذا الوصف إلى مجموعة من الجداول مع الوصف الخاص بكل منها وحفظ هذا الوصف في قاموس البيانات.

٢- **وصف تركيب البيانات الحقيقي (الفيزيائي) Storage Structure :** وهذا يشمل اختيار الاسلوب المناسب لتخزين البيانات في وحدات التخزين وأسلوب الوصول لهذه البيانات واسترجاعها. ويتم ذلك باستخدام لغة الوصف DDL الخاصة بذلك أيضا. وبعد إجراء عملية الترجمة على هذه اللغة يتم تكوين وحجز أماكن على وحدات التخزين خاصة بقواعد البيانات والجداول اللازمة لها بناءً على التعريفات التي وردت في لغة الوصف DDL.

٣- **إجراء التعديلات والتغييرات اللازمة على كل من التصميم المنطقي والتصميم الفيزيائي الوارد ذكرهما في النقطتين السابقتين**. والتعديلات عادة تكون من حين لآخر حسب الحاجة اليها. ويتم تطبيق هذه التعديلات بواسطة لغة الوصف DDL الخاصة بذلك. وبعد إجراء عملية الترجمة اللازمة يجب إعادة تنظيم قواعد البيانات الموجودة لكي يتم تحويلها الى التركيب الجديد، علما بأن التعديل على التركيب الحقيقي للبيانات نادر الحصول ولكن ممكن ان يتم ذلك وخاصة عند تغيير وحدات التخزين أو إجراء تعديل على تصميم قواعد البيانات وكل هذه التعديلات تنعكس على محتويات قاموس البيانات.

٤- **تحديد صلاحيات استخدام قواعد البيانات للمستخدمين**: ويعتبر هذا الواجب من أهم واجبات وحدة إدارة قواعد البيانات، حيث يتم إجراء دراسة خاصة لكل مستخدم، وبناءً عليه تحدد صلاحية الاستخدام لكل منهم ويتم بعدها مراقبة ومراجعة هذه الصلاحيات واستخدامها، وكذلك مراقبة أمن وسرية البيانات.

٥- **العمل على حفظ دقة البيانات وصحتها ووضع وتطبيق المواصفات الخاصة بذلك Integrity Constraints**: ويتم تعريف هذه المواصفات وتخزينها في مكان محدد ومن ثم الرجوع اليها عند إجراء أي تحديث أو تعديل على قواعد البيانات، ومن واجب وحدة إدارة قواعد البيانات مراقبة هذه الشروط والالتزام بتطبيقها، والعمل على مراقبة صحة ودقة تطابق البيانات من حين إلى آخر.

٦- **العمل على مراقبة وتحسين أداء نظام إدارة قواعد البيانات DBMS Performance:** ويتم التحضير لتحسن أداء النظام إثناء مرحلة التصميم ولكن من حين لآخر وإثناء التطبيق الفعلي للنظام يتم قياس ومراقبة الأداء وبالتالي العمل على تحسينه إذا لزم الأمر. وأداء النظام يعتمد على عوامل كثيرة منها:

- عدد المستخدمين
- حجم البيانات
- عدد الطرفيات
- سرعة الجهاز
- سعة الذاكرة
- شبكة الاتصالات

- طبيعة الحركات
- تركيب البيانات
- طبيعة استخدام البيانات
- الفهارس المستخدمة في النظام

٧- **وضع التعليمات والإجراءات اللازمة لاستخدام قواعد البيانات:** ويتم وضع أسلوب استخدام موحد ليستخدمه العاملون جميعا في المؤسسة.

٨- **القيام بالإجراءات الخاصة بالنسخ الاحتياطي Backup والاسترجاع Recovery:** وذلك من أجل استعادة قاعدة البيانات عند حدوث خلل فيها.

مكونات نظام إدارة قواعد البيانات DBMS Components

يتم تخزين **قواعد البيانات** هي **وقاموس البيانات** في وحدات التخزين، ويتم التحكم بالوصول لهذه البيانات عادة عن **طريق نظام التشغيل Operating System**، وعندما تخزن البيانات في الذاكرة يتم التعامل معها بواسطة نظام إدارة قواعد البيانات DBMS.

ويقوم **المترجم للغة الوصف DDL** بمعالجة الجمل الخاصة بتركيب البيانات ويقوم بتخزين هذا الوصف في قاموس البيانات الذي يحوي أسماء الملفات وحقول تعريفها والفهارس الخاصة بها وغيرها. ثم يقوم **معالج الطلبيات Query Processor** باستقبال الطلبات من المستخدمين ومن ثم يتأكد من خلوها من الأخطاء ومطابقتها لقواعد اللغة. وبعدها يتم تمريرها إلى **مترجم لغة المعالجة DML** الذي يقوم بترجمة أوامر اللغة إلى **اللغة التنفيذية لمعالج الحاسوب Object Code** لتصبح أوامر تنفيذية.

عند تنفيذ هذه البرامج يتم التعامل مع قواعد البيانات عن طريق **مدير قواعد البيانات DB Manager** الذي يتعامل بدوره مع الجزء الخاص بنظام التشغيل وهو **مدير الملفات File Manager.**

وهناك عدة أجزاء مهمة في نظام قواعد البيانات تقوم بمراقبة البيانات والتأكد من صحتها أهمها:

- **جزء الأمن والسرية Security**
- **جزء تحكم المشاركة Concurrency Control**
- **جزء النسخ الاحتياطي Backup**
- **جزء استرداد البيانات Recovery**

قاموس البيانات Data Dictionary

قاموس البيانات هو عبارة عن قاعدة بيانات مصغرة عن قاعدة البيانات الكبيرة أي بيانات عن البيانات، *والهدف الأساسي لقاموس البيانات* **هو الاحتفاظ بجميع المعلومات اللازمة لتركيب البيانات وأسلوب الوصول اليها وما يتعلق بشروط تحديثها والحفاظ على دقتها وسلامتها.**

المعلومات التي يتكون منها قاموس البيانات هي:

١. جدول بأسماء الملفات أو الجداول Tables المستخدمة في قاعدة البيانات مع المعلومات الخاصة بكل منها.

٢. جدول بأسماء الحقول Attributes المستخدمة في كل جدول.

٣. جدول بأسماء مجالات الحقول Domains.

٤. جدول بأسماء المناظر Views المستخدمة من قبل المستخدمين وتعريفها.

٥. جدول بالفهارس المستخدمة Indices

٦. شروط الدقة Integrity Constraints في كل جدول بما فيها المفاتيح.

٧. جدول بأسماء المستخدمين وصلاحياتهم.

٨. المعلومات الخاصة بمراقبة استخدام قواعد البيانات Accounting.

٩. المعلومات الإحصائية عن قواعد البيانات.

١٠. المعلومات الخاصة بأسلوب تخزين واسترجاع الجداول.

اذن هذا القاموس يعطي خارطة كاملة للبيانات مع علاقتها ببعضها البعض وتركيبها ويجب الرجوع له في كل استرجاع أو تخزين أو تحديث لهذه البيانات ، ويستخدمه الجميع سواء المستخدم العادي أو المبرمج أو إدارة قواعد البيانات.

الوحدة الرابعة :أمن وحماية قواعد البيانات

تعتبر قواعد البيانات ذات أهمية كبيرة لتخزين البيانات ومعالجتها ومن ثم إخراجها بالشكل المطلوب وأيضا نظرا لاعتماد معظم الجهات الحكومية والمنشآت الخاصة عليها في حفظ بياناتها واسترجاعها، وبالمقابل فقد زاد انتشار جرائم السرقة للمعلومات في قواعد البيانات لأهداف مختلفة إما لاستخدامها في الابتزاز وكسب المال من بيع المعلومات الحساسة مثل بيع أرقام بطاقات الائتمان أو لبيان الاحتجاج على أمر معين أو لأهداف شخصية أخرى.

ولذا يجب حماية البيانات من المخاطر التي قد تواجهها إما عن قصد أو غير قصد.

أمثلة عن تعرض البيانات للخطر عن *غير قصد*:

١. تعطل النظام أثناء عملية التحديث Crash .

٢. عدم تحديث جميع البيانات بسبب أسلوب المشاركة Concurrency .

٣. عدم تحديث جميع البيانات بسبب توزيعها في عدة أماكن Distribution .

٤. وجود بعض الأخطاء المنطقية التي تخالف شروط الدقة ولم تأخذها المؤسسة بالاعتبار.

أمثلة عن تعرض البيانات للخطر *عن قصد*:

١. سرقة البيانات أي قراءتها من قِبل أشخاص معادين للمؤسسة .

٢. تغيير البيانات من قِبل أشخاص غير ذوي الصلاحية .

٣. مسح البيانات من قِبل أشخاص غير ذوي الصلاحية.

لا بد أن ننوه بأن أكثر الأخطاء تأتي من داخل المؤسسة نفسها كما أنه من المستحيل توفير الحماية الكاملة لأي بيانات فجميع طرق الحماية ودرجة قوتها هي نسبية وهناك دائمات محاولات لكسر هذه الحماية ودائما هناك معركة مستمرة بين الجهات الأمنية التي من واجبها توفير أفضل الحماية للبيانات وبين الجهات المعادية التي تحاول جهدها كسر أمن هذه البيانات.

هناك ثلاثة مصطلحات يجب التفريق بينها في هذا المجال هي:

- **الأمن Security** : ويعني توفير الحماية لمنع تعرض البيانات لجهات معادية.
- **الخصوصية Privacy**: وتعني حماية البيانات من التعرض لجهات ليست صاحبة صلاحية وقد تكون غير معادية .
- **الدقة Integrity :** وتعني حماية البيانات من التعرض للتعديل والتحديث بشكل خاطئ.

من المستويات المستخدمة لتوفير الحماية للبيانات:

١) **الاحتياطات العملية**: وذلك بتوفير مكان آمن لأجهزة الحاسوب والبيانات كأن توضع في أدوار تحت الأرض وتوفير الأقفال والاحتياطات اللازمة لحمايتها.

٢) **الاحتياطات البشرية:** بحيث تتحرى المؤسسة الدقة عند اختيار المستخدمين والعاملين فيها كما يجب ان تمنح كل شخص الصلاحيات اللازمة له فقط بالإضافة لمراقبة سلوك أفرادها.

٣) **الاحتياطات عن طريق نظام التشغيل:** يلعب نظام التشغيل المستخدم دورا مهما في توفير الحماية اللازمة للنظام، وغالبا ما يتم استخدام أسلوب أرقام الحسابات Account Numbers وكلمات السر Passwords وتوزيع الصلاحيات، كما أن هناك بعض نظم التشغيل تعطي احتياطات على مستوى الطرفيات ووقت وأسلوب استخدامها.

٤) **الاحتياطات المتوفرة عن طريق نظام قواعد البيانات:** حيث يوفر نظام إدارة قواعد البيانات غالبا مستوى آخر من الأمن للبيانات عن طريق أرقام الحسابات وكلمات السر وكذلك درجة الصلاحيات الممنوحة للمستخدمين ومراقبة استخدام البيانات.

٥) **الاحتياطات المتوفرة عن طريق البرامج التطبيقية:** حيث يمكن توفير درجة أعلى من الأمن للبيانات عن طريق تحضير برمجيات خاصة واستخدامها في أثناء تطبيق نظام إدارة قواعد البيانات.

٦) **تشفير البيانات:** وهي عملية تحويل المعلومات التي تكون بشكل نص بسيط عند التخزين بحيث تصبح غير مقروءة لأحد باستثناء من يملك معرفة خاصة أو مفتاح خاص لإعادة تحويل النص المشفر إلى نص مقروء. ويعد هذا الأسلوب مرتفع التكلفة وصعب التطبيق لذا لا يتم استخدامه إلا للبيانات مرتفعة الأهمية.

توفر لغة SQL بعض الأوامر المستخدمة لتوزيع صلاحيات استخدام البيانات على المستخدمين كما يمكن مراقبة هذا الاستخدام عند تنفيذه، كما يمكن تحديد مستوى المراقبة لتصل إلى حقل الجدول أو الجدول نفسه أو المنظر View **وفيما يلي بعض الصلاحيات المستخدمة:**

أ. صلاحية قراءة البيانات Select .

ب. صلاحية إضافة بيانات جديدة Insert .

ج. صلاحية حذف بيانات موجودة Delete.

د. صلاحية تحديث بيانات موجودة Update .

أما الأمر المستخدم لمنح الصلاحيات في لغة SQL فهو الأمر ***Grant*** وصيغته كالتالي:

{ قائمة بأسماء المستخدمين } **To** { اسم الجدول المطلوب} **On** { قائمة الصلاحيات المطلوبة } **Grant**

في حين أن الأمر المستخدم لإلغاء الصلاحيات في لغة SQL فهو الأمر ***Revoke*** وصيغته كالتالي:

{ قائمة بأسماء المستخدمين } **From** { اسم الجدول المطلوب} **On** { قائمة الصلاحيات المراد ازالتها } **Revoke**

مثال١: منح صلاحية إدخال سجلات جديدة للجدول Books للمستخدمين usr1, usr2

Grant Insert **On** Books **To** usr1, usr2

مثال٢: إلغاء صلاحية القراءة والتحديث عن الجدول Books من المستخدم usr4

Revoke Select , Update **On** Books **From** usr4

ماذا تحمي في قواعد البيانات؟

هدفنا الرئيسي من حماية قواعد البيانات هو حماية المعلومات الموجودة فيها ولحمايتها يجب حماية الصفات الثلاث الرئيسية التي ترتكز عليها المعلومات وهي (السرية، السلامة، التوفر).

١) **السرية** :وهي ضمان أن الأطراف المسموح لها فقط يمكن لها عرض البيانات، ويمكن تطبيق السرية من خلال تشفير المعلومات المخزنة في قاعدة البيانات، يتم التشفير على مستويين مختلفين: التشفير على مستوى البيانات العابرة أو التشفير على مستوى البيانات الثابتة.

- **التشفير على مستوى البيانات العابرة:** وهو للمعلومات الحساسة التي تتحرك داخل الشبكة ويمكن للمهاجم الوصول إليها من خلال التنصت .
- **التشفير على مستوى البيانات الثابتة:** وهو للمعلومات الحساسة المخزنة في قاعدة البيانات التي من الممكن للمهاجم اختراقها.

٢) **السلامة:** وهي ضمان صحة البيانات وأنه لا يمكن لأي شخص غير مصرح به أو البرامج الحاسوبية الضارة تغييرها. ولضمان السلامة يجب إتباع هذه الخطوات:

- تغيير كلمة المرور حالما يتم الانتهاء من تثبيت قاعدة البيانات.
- إزالة حسابات المستخدمين التي ليست قيد الاستخدام.
- وضع سياسات لتعيين كلمات مرور قوية، مثل إجبار الموظفين على تغيير كلمة المرور بداية كل شهر.
- السماح للمستخدم فقط استخدام الوظائف المصرح له باستخدامها.

٣) **التوافر:** على الرغم من حماية الحاسوب لتقييد محاولات الوصول من قبل الأشخاص غير المصرح بهم، يجب أن لا تزال البيانات متاحة للأشخاص المصرح لهم. ولضمان التوفر يجب إتباع هذه الخطوات:

أ. تقييد مقدار مساحة التخزين التي تمنح لكل مستخدم في قاعدة البيانات.

ب. أخذ نسخ احتياطية للبيانات بشكل دوري لضمان استعادة البيانات في حالة فقدانها.

ج. تأمين قاعدة البيانات ضد الثغرات الأمنية.

كيف نحمي المعلومات؟

يمكننا حماية الصفات الرئيسية للمعلومات عن طريق استخدام ضوابط الأمن:

١. **المنتجات** وهي الضوابط المادية التي توفر الأمن اللازم برصد ومراقبة البيئة في مكان العمل ومرافق الحوسبة التي تحوي على قاعدة البيانات، هذه المنتجات من الممكن أن تكون بسيطة كأقفال الأبواب أو معقدة كأنظمة كشف التطفل وجدار الحماية، وبذلك وفرنا الأمن المادي للبيانات.

٢. **الناس**: فبدون تنفيذ الناس واستخدام المنتجات الأمنية بشكل صحيح لا يمكن أبدا أن تكون البيانات محمية.

٣. **الإجراءات**، وهي عبارة عن الرقابة الإدارية وتشمل الخطط والسياسات التي تضعها المنظمة لضمان أن الناس يستخدمون المنتجات بشكل صحيح.

وهذه الطبقات الثلاث تتفاعل مع بعضها البعض لتحقيق الأمن للمعلومات.

بعض أنواع التهديدات اتجاه امن قواعد البيانات واساليب التصدي لها:

١- **إساءة استخدام الامتيازات**:عندما يتم توفير قاعدة بيانات المستخدمين مع الامتيازات التي تتجاوز متطلبات وظائفهم يوما بعد يوم، قد يساء استخدام هذه الامتيازات عمدا أو عن غير قصد. على سبيل المثال مسؤول قاعدة البيانات في مؤسسة مالية، ماذا سيحدث إذا قام بإيقاف عمليات المراجعة وإنشاء حسابات وهمية، وبذلك سوف يكون قادر على تحويل الأموال من حساب إلى آخر وبالتالي استغلال امتياز مفرط عمدا.

ويمكن منع إساءة الاستخدام عن طريق محدودية الوصول، فالمعلومات المتاحة للشخص هي فقط المعلومات التي يحتاجها.

٢- **الثغرات في نظام التشغيل:**وجود ثغرات أمنية في أنظمة التشغيل الأساسية مثل ويندوز ولينكس ويونيكس وما إلى ذلك والخدمات التي ترتبط بقواعد البيانات قد تؤدي إلى الوصول غير المصرح به وهذا قد يؤدي إلى هجوم تعطيل الخدمة .(Denial of service) **ويمكن منع هذا عن طريق تنزيل تحديثات نظم التشغيل المتعلقة بالأمن عندما تصبح متاحة.**

٣- **حقن SQL injection:** في هجوم حقنSQL المرتكب عادة يتم إدخال أو حقن بيانات قاعدة بيانات غير مصرح بها إلى قناة بيانات SQLالضعيفة أو غير المحمية، وعادة يتم استهداف قنوات البيانات التي تشمل الإجراءات المخزنة ومدخلات تطبيق الويب، ثم يتم تمرير هذه البيانات المحقونة إلى قاعدة البيانات حيث يتم تنفيذها باستخدام حقنSQL ، وبالتالي المهاجم ربما كسب وصول غير مقيد إلى قاعدة البيانات بأكملها.

ومن الممكن مكافحة حقن SQL عن طريق الجمع بين ثلاث تقنيات فعالة: أنظمة منع الاختراق (IPS) التي تتفقد حركة قاعدة البيانات وتحدد الهجمات التي استهدفت والثغرات، والتحكم بالوصول على مستوى الاستعلام (query-level access control) الذي يقوم بتحديد الامتيازات للحد الأدنى من العمليات والمعلومات المطلوبة، وأخيرا ترابط الحدث.

٤- **ضعف المصادقة:** نماذج ضعف المصادقة تسمح للمهاجمين بتوظيف استراتيجيات مثل الهندسة الاجتماعية (Social engineering) ويقوم المهاجم من خلالها باستغلال الميل الطبيعي للإنسان للثقة وحب مساعدة الآخرين لمعرفة معلومات تسجيل الدخول، أو باستخدام القوة الوحشية (Brute force) التي من خلالها يقوم المهاجم بإدخال توليفات لاسم المستخدم وكلمة المرور لمرات عديدة حتى يجد واحدا منهم يعمل ومن الممكن أن يستخدم برامج آلية لتقوم بعملية التوليف، أو أخيرا عن طريق سرقة معلومات التفويض مباشرة وبالتالي الحصول على معلومات تسجيل الدخول لقاعدة البيانات لمستخدمين مصرح لهم. **ولمنع هجمات المصادقة ينبغي استخدام أقوى تقنيات المصادقة (الرموز، الشهادات، الصفات الحيوية وغيرها) وأقوى السياسات واستخدام المصادقة المتبادلة والمصادقة متعددة العوامل.**

٥- **ضعف التعقب:** ضعف آلية تعقب تسجيل الدخول للحسابات في خادم قاعدة البيانات يمثل خطر حرج للمنظمة خاصة في مجال التجارة والرعاية الصحية والمالية وغيرها من صناعات الامتثال التنظيمية الصارمة، ولوائح القوانين مثل (PCI, SOX, and HIPAA) تتطلب تسجيل واسع النطاق للنشاطات لإعادة حدث في مرحلة سابقة من الزمن في حالة وقوع حادث، ولذلك يجب أن يتم تسجيل

المعاملات الحساسة أو العادية التي تتم في قاعدة البيانات بطريقة آلية لتسوية الحوادث، والتعقب بمثابة خط الدفاع الأخير لقاعدة البيانات فيمكن من خلاله الكشف عن وجود انتهاك وتتبع الانتهاك لنقطة معينة من الزمن ولمستخدم معين.

٦- **التعرض للنسخة الاحتياطية للبيانات:** عادة ما تكون النسخة الاحتياطية للبيانات المخزنة غير محمية من الهجوم، ونتيجة لذلك هناك عدة انتهاكات أمنية شملت سرقة أشرطة النسخ الاحتياطية والأقراص الصلبة لقاعدة البيانات.ولمنع التعرض للنسخ الاحتياطية للبيانات ينبغي أن تكون جميع النسخ الاحتياطية لقاعدة البيانات مشفرة، وقد اقترح بعض المتخصصين أنه مستقبلا قد لا تدعم منتجات نظم إدارة قواعد البيانات (DBMS)إنشاء نسخ احتياطية غير مشفرة.

الوحدة الخامسة : استرداد البيانات

هناك العديد من الأعطال التي قد تصيب نظام الحاسوب بسبب أمور كثيرة مثل انقطاع الكهرباء وتعطل أجهزة التخزين نفسها أو اندلاع حريق في الأجهزة أ حصول أخطاء في البرمجيات أو بسبب نوع من أنواع التخريب المتعمدة. في جميع هذه الحالات تتعرض البيانات للخراب أو الضياع. ولكي يتم إعادة البيانات إلى وضعها الصحيح السابق للخراب يتطلب الأمر القيام بإجراءات تساعد على ذلك. من هذه الإجراءات تجميع بعض المعلومات في أثناء سير العمل الاعتيادي، وفي حالة حدوث الخراب يتم الرجوع لهذه المعلومات لاسترجاع وإعادة هذه البيانات إلى سابق حالتها.

وهناك عدة احتمالات لضياع المعلومات في الحاسوب. منها تعطل الذاكرة RAM أو تعطل وحدات التخزين Disk لكن معظم حالات ضياع البيانات يكون بسبب تعطل الذاكرة.

إن أنظمة الحاسوب التي لا تستخدم أسلوب الاتصال المباشر On-Line ولا تعدد المستخدمين Multi-user أي نظم أسلوب تجميع العمل Batch يسهل فيها استرجاع البيانات ولا يحتاج إلى تعقيد وذلك لأن الاسترجاع يعتمد على إعادة تنفيذ البرامج التي تم خلالها ضياع للبيانات فقط. أما النظم الحديثة التي تعتمد على الاتصال المباشر وتعدد المستخدمين في آن واحد Concurrent فيصبح استرجاع البيانات فيها صعبا لأنه لا يمكن معرفة البرامج والبيانات التي قام كل مستخدم بتنفيذها على الحاسوب، وبذلك فإن أسلوب الاسترجاع يحتاج إلى احتياطات وبرمجيات خاصة لتقوم بهذا الواجب.

مثال على ذلك: نظام أحد البنوك الذي يحوي على مئة فرع جميعها متصلة بالحاسوب المركزي للبنك وبينما كان أحدها يقوم بإجراء صرف شيك وآخر ايداع نقد، وآخر يقوم بإجراء تحويل وغيرها من العمليات البنكية المختلفة، وفي أثناء العمل ولسبب ما، حصل ضياع للمعلومات الموجودة في ذاكرة الحاسوب والمطلوب إعادة قاعدة البيانات إلى الوضع الذي كانت عليه قبل حدوث هذه المشكلة أو إعادتها إلى حالة أقرب ما تكون إلى الوضع المذكور. ونظرا لعدم معرفة الوضع الذي كانت فيه قاعدة البيانات ولتعدد المستخدمين وتعدد عملهم يصبح من الصعوبة بمكان تحقيق هذا الطلب.

هناك أسلوبان يمكن استخدامهما لمعالجة هذه المشكلة، الأسلوب الأول يعتمد على فكرة إعادة المعالجة Reprocessing والثاني يعتمد على فكرة العودة إلى الخلف ثم الأمام Roll back / Roll forward .

- **الاسترجاع بأسلوب إعادة المعالجة Reprocessing**

بما أنه من يصعب العودة بالبيانات إلى نقطة محددة صحيحة ، فإنه يمكن العودة إلى أحسن وضع ممكن ومن ثم إعادة المعالجة من تلك النقطة. إن أبسط أشكال هذا الأسلوب هو عمل نسخة من البيانات الصحيحة على فترات محددة وبعدها يتم تجميع الحركات التي تمت بعد عمل كل نسخة. وفي حالة العطل يتم استرجاع آخر نسخة من البيانات ومن ثم إعادة معالجة الحركات المجمعة بعد آخر نسخة. وستكون نتيجة هذه الإجراءات إعادة البيانات إلى أفضل نقطة صحيحة ممكنة.

هناك صعوبات لتنفيذ هذا الأسلوب هي:

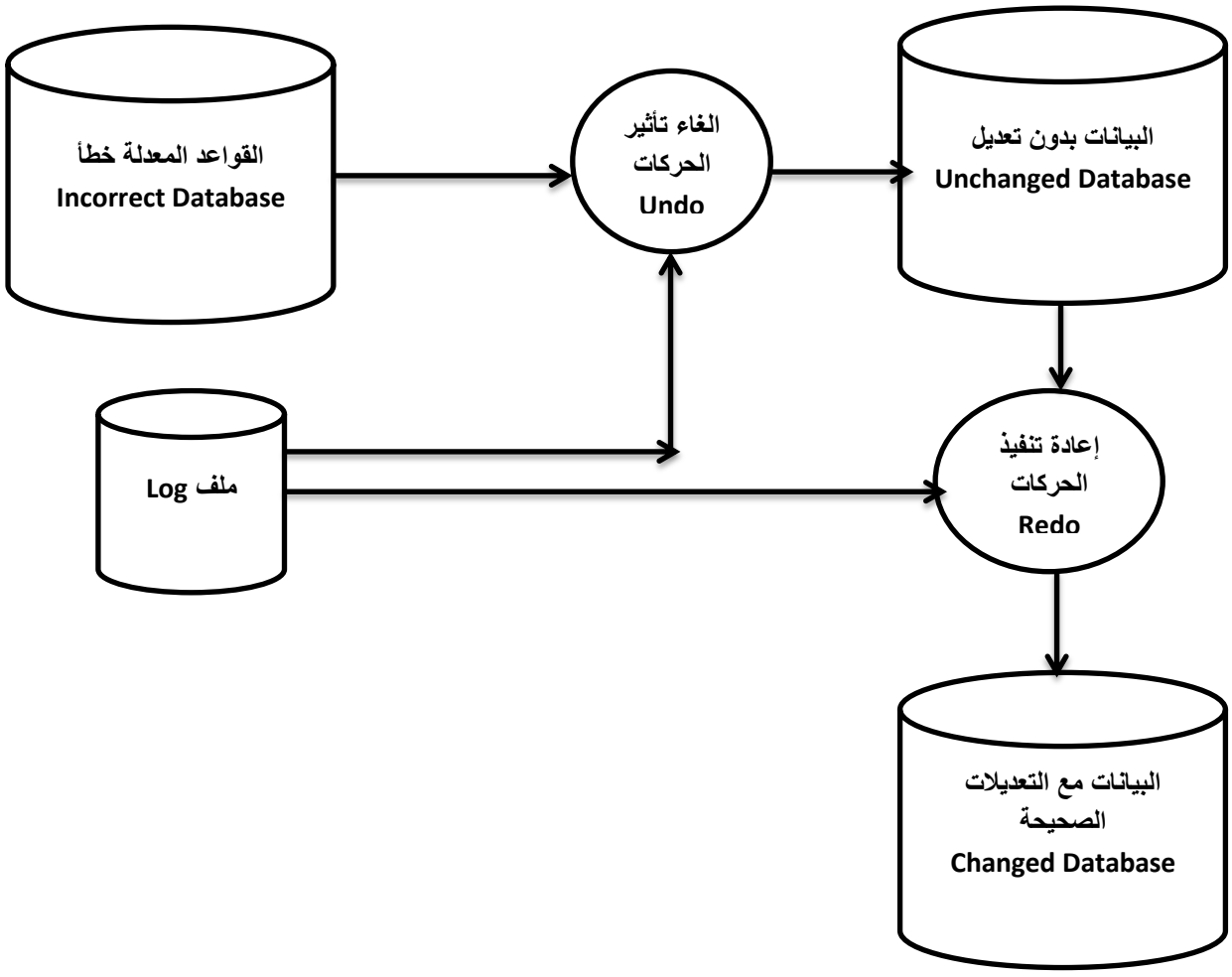
أ- الحاجة لتسجيل نسخ عن البيانات من فترة لأخرى وهذا يحتاج إلى سعة تخزين كبيرة ووقت كبير.

ب- يحتاج إلى معالجة الحركات بعد إخراج آخر نسخة وهذا يحتاج إلى وقت حتى وإن كان النظام من النوع البسيط فإنه يصعب إعادته إلى حالته الأولى.

ت- الحركات التي تتم على البيانات هي حركات ذات طبيعة ديناميكية أي أنها تحصل كلها مع بعضها البعض دون ترتيب مسبق Concurrent.

- **الاسترجاع بأسلوب العودة إلى الخلف / الأمام (Roll back / Roll forward)**

هذا الأسلوب يعتمد على فكرة أخذ نسخة (صورة) من البيانات على فترات أيضا والاحتفاظ في الوقت نفسه بملف خاص يسمى Log لأخذ التعديلات وتنفيذها على قواعد البيانات أي العودة إلى الأمام Rollforward بدلا من إعادة تنفيذ الإجراءات. أو يمكن استخدام ملف Log لإلغاء تأثير الحركات الخاطئة Undo عن قواعد البيانات أي العودة إلى الخلف Rollback وبعد ذلك يتم إجراء إعادة تنفيذها من جديد Redo . وفي كلا الحالتين يجب استخدام ملف Log كما تتطلب الإجراءات بعض الوقت لإعادة قواعد البيانات إلى حالتها الصحيحة. الشكل ٥-١ يوضح مبدأ عمل هذا الأسلوب:



الشكل ٥-١: مبدأ عمل الاسترجاع بأسلوب العودة إلى الخلف / الأمام

ملف Log

يستخدم هذا الملف في جميع أنواع قواعد البيانات تقريبا حيث يخزن به جميع ما يجري على البيانات من تعديل أو تبديل في أثناء العمل الاعتيادي ومن ثم يمكن الرجوع إليه في حالة حدوث أي عطل لاسترجاع الحالة الصحيحة لهذه البيانات. ويتركب السجلات في ملف Log كما يأتي:

- في بداية كل إجراء Transaction يوضع على هذا الملف سجل يحوي رقم الإجراء بالإضافة لكلمة توضح البداية < Ti , start > .
- في حالة تعديل البيانات من قبل أي إجراء، يوضع على الملف سجل يبين رقم الإجراء واسم الحقل المنوي تعديله والقيمة الجديدة لهذا الحقل < Ti , Dj , V_1 , V_2 > .
- في حالة انتهاء الإجراء طبيعيا يوضع على الملف سجل يحوي رقم الإجراء بالإضافة لكلمة توضح النهاية < Ti , End > .
- يحوي الملف أيضا علامة مميزة توضح لحظة التأكد من صحة البيانات وهذه العلامة تسمى نقطة الفحص < Check point > .

النسخ الاحتياطية Backups

إن أبسط أنواع الطرق لاسترجاع البيانات في حالة العطل هو أخذ نسخ احتياطية Backups من قواعد البيانات ويمكن وضع برنامج ثابت لتنفيذ هذا الاسلوب، فبعض المؤسسات تحتاج لأخذ نسخ يومية (نهاية اليوم) أو أسبوعية أو شهرية وبعضها سنوية أو يمكن عمل نسخ مختلفة يومية وأسبوعية معا. بعدها يمكن الرجوع إلى هذه النسخ لإعادة البيانات إلى أقرب وضع صحيح ممكن ومن ثم تنفيذ الإجراء المطلوب لاسترجاع البيانات إلى الوضع السابق كما سبق.

الوحدة السادسة : قواعد البيانات الموزعة

مقدمة عن قواعد البيانات الموزعة Distributed Database

لقد كان حديثنا في السابق مرتكزا على أنماط متنوعة من قواعد البيانات لكنها تشترك جميعا في أنها مركزية، ولكن نظرا للتقدم الكبير الذي حصل بشبكات الحاسوب وانتشار أجهزة الحاسوب فقد أصبح بالإمكان تخزين قواعد البيانات في أماكن عدة حسب استخداماتها بدلا من تخزينها في مكان واحد رئيس وكبير. وتعد نظم المعالجة الموزعة شكلا حديثا ومتطورا من أشكال المعالجة اللامركزية. ويتم بناء أنظمة المعالجة الموزعة من خلال شبكات اتصال حاسوبية تتقدم خدمة ربط الحواسيب مع بعضها البعض كما تقدم خدمة تراسل المعلومات حيث يتم الاتصال ما بين هذه الحواسيب عن طريق إرسال الرسائل Messages فيما بينها، وتحتوي هذه الرسائل إما على تعليمات أو على بيانات، والعامل الأساسي في الوقت اللازم للحصول على البيانات هو الوقت الذي يتطلبه نقل البيانات من خلال الشبكة.

إن الربط بين هذه الحواسيب يكون على عدة أشكال حسب التصميم الذي ترغب فيه المؤسسة ، فمنها ما يكون على مستوى محلي Local ويشمل مساحة جغرافية صغيرة مثل جامعة أو مدينة صغيرة في موقع محدد. ويستخدم من أجل ذلك شبكات التوزيع المحلية Local Area Network (LAN) ، وقد يكون أكبر من ذلك ليشمل مواقع موزعة على مستوى مدينة كبيرة أو بلد أو قارة أو على مستوى عالمي بين القارات ونطلق عليه التوزيع الواسع أو الشامل Global ويستخدم من أجل ذلك شبكات التوزيع العالمية الشاملة أو الواسعة Wide Area Network (WAN)، وقد ازداد الطلب على المعالجة الموزعة لقواعد البيانات مع كثرة انتشار الشركات الكبرى ذات الهيكلية الموزعة والمتفرعة مثل البنوك وتوزيع هذه الفروع على مدن كثيرة أو حتى في المدينة الواحدة، ورغبة هذه الشركات في توزيع المعالجة والتعامل مع قواعد البيانات لتسهيل اتصال الفروع بعضها ببعض، والحاجة إلى إجراء العمليات بأسهل وقت ممكن وأسرعه.

إن البرمجيات التي تقوم بتنظيم عمل هذا النظام وتمكن المستخدمين من تخزين واسترجاع قواعد البيانات بصورة سهلة تسمى برمجيات نظام قواعد البيانات الموزعة ، ومن أهم وظائف هذه البرمجيات توفير ما يسمى بشفافية النظام Transparency **والمقصود بالشفافية هي استطاعة المستخدم التعامل مع النظام الموزع وكأنه نظام مركزي دون إشغاله بطريقة توزيع النظام أو أسلوب عمله** ويمكن تقسيم هذه الشفافية إلى ما يلي:-

١. شفافية مكان التخزين Location Transparency : يستطيع المستخدم الحصول على قواعد البيانات دون الحاجة إلى معرفة مكان تخزينها وبحيث يظهر النظام وكأن جميع بياناته مخزونة في مكان واحد محلي مع أنها فعليا موزعة في أماكن مختلفة. إن عمل هذا النظام يشابه عمل النظام المركزي فيما عدا الوقت اللازم للحصول على هذه البيانات الذي غالبا ما يكون أطول. وهذه الشفافية تعتبر من أهم أغراض برمجيات قواعد البيانات الموزعة.
٢. شفافية التكرار Replication Transparency: في حالة توزيع قواعد البيانات مع تكرار بعضها في عدة أماكن يجب أن يخفي النظام هذا التكرار عن المستخدم. إن التكرار في تخزين البيانات في عدة أماكن يحسّن من كفاءة الطلبات للمستخدمين حيث يصبح بالإمكان الحصول على البيانات من أقرب نسخة من هذه البيانات، ولكن من جهة أخرى تصبح عمليات التحديث صعبة ومكلفة. ويضمن النظام أن تكون جميع النسخ مطابقة لبعضها البعض ويجب أن تتوقف قراءة هذه البيانات إلى حين الانتهاء من تحديث جميع النسخ. تعتبر هذه المشكلة من أهم وأصعب مشاكل قواعد البيانات الموزعة. ويقوم النظام غالبا بتطبيق بعض البروتوكولات

للقيام بهذه المهمة ولكن لا يجب أن لا يعلم بها المستخدم بحيث يم التعامل مع النظام وكأنه نظام مركزي وهذا هو المقصود بشفافية التكرار.

٣. شفافية تجزئة البيانات Fragmentation Transparency : يتم تقسيم قواعد البيانات ما بين مراكز الحواسيب المختلفة، فمثلا يقوم بنك ما بتخزين حسابات زبائنه كل في فرع يكون أقرب لمكان سكن الزبون. إن هذا الأسلوب هو تقسيم التصميم المنطقي Logical Schema إلى أجزاء صغيرة كل منها موجود في مكان، ويجب على النظام أن يخفي هذه التجزئة عن المستخدم وأن يقدم قواعد البيانات إليه وكأنها نظام مركزي.

حسنات وسيئات نظام قواعد البيانات الموزعة

أ- الحسنات:

١) **ملاءمة هذا الأسلوب لطبيعة بعض التطبيقات بشكل أفضل**: مثل شركات التأمين الكبرى التي لها مكاتب تتوزع على عدة مدن، لذا فطبيعة النظام هو الذ يفرض استخدام هذا الأسلوب الذي يقلل قدر الإمكان من نقل البيانات من خلال شبكة الاتصالات بحيث تكون متوفرة في مكان استخدامها مع إمكانية الحصول على بعضها من الأماكن الأخرى في الحالات الضرورية. في هذه الطريقة يصبح بالإمكان التحكم بالبيانات والحفاظ عليها بشكل أفضل.

٢) **أسباب اقتصادية وهيكلية**: فبدلا من التمسك بتنظيم هيكلي مركزي للمؤسسة، ومركز معلومات مركزي، يمكن توزيع عبء العمل على مواقع متعددة مما يسهل الوصول للمعلومات بسرعة إذا تم التوزيع للبيانات حسب التنظيم الوظيفي.

٣) **قابلية الدمج والربط بين قواعد البيانات المختلفة العاملة حاليا** مما يسهل وسائل الاتصال ونقل المعلومة آليا بواسطة الحاسوب.

٤) **سهولة الربط والتوسع المستقبلي في أجهزة الحاسوب**: مقارنة مع النظام المركزي، إن إضافة أجهزة جديدة للنظام الموزع أسهل بكثير من النظام المركزي، حيث يتم التوسع في النظام المركزي القديم باستبدال الأجهزة القديمة في حالة الزيادة في قواعد البيانات. أما في حالة النظام الموزع فيتم التوسع بإضافة وحدات جديدة في الأماكن التي يحصل فيها التوسع وربطها مع الشبكة بدلا من استبدال جميع الأجهزة القديمة، مثال ذلك: افتتاح فرع جديد لأحد البنوك حيث يمكن تنفيذه بسهولة ضمن بيئة قواعد البيانات الموزعة.

٥) **تقليل تكلفة الاتصالات ونقل المعلومات وزيادة أداء النظام**: عند استخدام المعالجة الموزعة فإن كل فرع يحوي البيانات الخاصة به ويعالج تلك البيانات محلياً، لذا تقلّ الحاجة لنقل البيانات ومعالجة المعلومات بين المراكز الموزعة، ويؤدي ذلك لزيادة السرعة مما يزيد ويحسن أداء النظام.

٦) **تحسين اعتمادية النظام والثقة به Reliability:** في حالة تعطل جهاز في النظام المركزي فإن جميع قواعد البيانات تصبح معطلة وغير قابلة للاستخدام، أما في النظام الموزع فإنه إذا تعطل أحد مراكز النظام فإن هذا الجزء من قواعد البيانات يمكن الحصول عليه من المراكز الأخرى عن طريق الاتصالات.

٧) **تحسين توفر النظام Availability :** حيث أن توزيع البيانات على المراكز التي تستخدمها تحسن سرعة الوصول إليها وتصبح متوفرة بشكل أفضل للمستخدم كما أنه بتكرار البيانات في مراكز مختلفة تصبح هذه البيانات أقرب إلى المستخدم وتقل تكلفة الوصول إليها.

ب- السيئات والمشكلات:

١) **صعوبة تحديث البيانات المكررة:** تعتبر هذه المشكلة من أهم المشكلات التي تواجه مصممي قواعد البيانات الموزعة. إذ يصعب تحديث جميع النسخ من هذه البيانات في الوقت نفسه، ولذا يجب اتباع بروتوكول خاص للقيام بهذه المهمة، وهناك كلفة عالية لتنفيذ هذه العملية بسبب نقل رسائل التحديث ما بين مراكز البيانات مع

عدم استخدام هذه البيانات خلال هذه المدة اللازمة لتحديث جميع النسخ حيث يتم اقفالها لحين الانتهاء أو استخدام فكرة ترتيب الإجراءات في جميع المراكز أو اعتماد نسخة واحدة من البيانات كنسخة رئيسية يتم الاعتماد عليها قبل البدء بالتحديث.

٢) **صعوبة إجراء الطلبيات المعقدة:** إن معالجة الطلبيات في حالة النظام الموزع أصعب بكثير منها في حالة النظام المركزي حيث يجري تبادل البيانات خلال الشبكة ما بين المراكز ففي هذه الحالة يجب اختيار أفضل الطرق لتنفيذ هذه الطلبيات بحيث تكون كلفة هذه الطلبية من الوقت أقل ما يمكن.

٣) **الكلفة العالية في تصميم وبرمجة أنظمة القواعد الموزعة:** يحتاج العمل في هذه البيئة إلى مبرمجين ومصممين ذوي كفاءة عالية وخبرة ممتازة أكثر بكثير من النظام المركزي مما يزيد من كلفة تطوير وتطبيق هذه الأنظمة.

تجزئة البيانات Data Fragmentation

يتم تصميم قواعد البيانات الموزعة بحيث توضع البيانات في أماكن استخدامها، وبذلك يمكن تجزئة هذه البيانات ما بين المراكز حسب أسلوب استخدامها وهناك طريقتان لتجزئة البيانات هما:

١- **التجزئة الأفقية Horizontal Fragmentation :** وفي هذه الطريقة يمكن تقسيم أي ملف أفقيا وتوزيع هذه الأجزاء على المراكز في النظام ويتم التقسيم غالبا حسب شرط معين.

مثال١ : بنك يحفظ بيانات زبائنه في جدول اسمه **Deposit** والذي يتكون كالتالي:

اسم الزبون C-Name	رمز الفرع B-Name	رقم الحساب Acct-No.	المبلغ Balance
أحمد صالح	BJ	5560	55 $
محمود علي	BA	5700	5000 $
منى علي	BK	6500	10 $
ماهر عواد	BJ	6600	100 $
صلاح محمد	BA	6890	1100 $
رولا أحمد	BK	9890	58 $

يمكن تجزئة هذا الجدول **أفقيا** بحيث نضع جميع الحسابات التي تخص **كل فرع** في جدول خاص بها ويصبح شكل الجداول الناتجة كالتالي:

جدول الفرع BJ

اسم الزبون C-Name	رمز الفرع B-Name	رقم الحساب Acct-No.	المبلغ Balance
أحمد صالح	BJ	5560	55 $
ماهر عواد	BJ	6600	100 $

```
Select C-Name, B-Name, Acct-No. , Balance

From Deposit

Where B-Name = 'BJ'
```

جدول الفرع BA

اسم الزبون C-Name	رمز الفرع B-Name	رقم الحساب.Acct-No	المبلغ Balance
محمود علي	BA	5700	5000 $
صلاح محمد	BA	6890	1100 $

```
Select C-Name, B-Name, Acct-No. , Balance

From Deposit

Where B-Name = 'BA'
```

جدول الفرع BK

اسم الزبون C-Name	رمز الفرع B-Name	رقم الحساب.Acct-No	المبلغ Balance
منى علي	BK	6500	10 $
رولا أحمد	BK	9890	58 $

```
Select C-Name, B-Name, Acct-No. , Balance

From Deposit

Where B-Name = 'BK'
```

٢- التجزئة العمودية Vertical Fragmentation : حيث يتم تقسيم جدول البيانات عموديا ما بين مراكز الحاسوب المختلفة.

مثال١: باستخدام الجدول السابق Deposit وإذا أردنا تجزئة الجدول عموديا بحيث نضع حقول : اسم الزبون ، رقم الفرع ورقم الحساب في مكان واحد ، في حين نضع حقول: رقم الحساب والمبلغ في مكان آخر فإنه يصبح شكل الجداول كالتالي:

جدول ١

اسم الزبون C-Name	رمز الفرع B-Name	رقم الحساب.Acct-No
أحمد صالح	BJ	5560
محمود علي	BA	5700
منى علي	BK	6500
ماهر عواد	BJ	6600
صلاح محمد	BA	6890
رولا أحمد	BK	9890

```
Select C-Name, B-Name, Acct-No.

From Deposit
```

جدول ٢

Acct-No.رقم الحساب	Balance المبلغ
5560	55 $
5700	5000 $
6500	10 $
6600	100 $
6890	1100 $
9890	58 $

Select Acct-No. , Balance

From Deposit

الوحدة السابعة : مشاكل التزامن

١- مشكلة التعديل المفقود Lost Update

وتحدث عندما يقوم إجراءان يتعاملان مع نفس عناصر قاعدة البيانات حيث يتم تنفيذهما معا بطريقة تجعل قيمة بعض عناصر قاعدة البيانات خاطئة.

مثال: لو كان لدينا إجراءان T_2 , T_1 وتم البدء بتنفيذهما معا على النحو التالي:

T_2		T_1	
		اقرأ قيمة X	Read_item(x);
		اطرح N من X	X:=X– N;
اقرأ قيمة X	Read_item(x);		
أضف M إلى X	X:=X+ M;		
		خزن قيمة X	Write_item(x);
		اقرأ قيمة Y	Read_item(Y);
خزن قيمة X	Write_item(x);		
		أضف N إلى قيمة Y	Y:=Y+N;
		خزن قيمة Y	Write_item(Y);

في المثال السابق تصور أن X هي مثلا عدد المقاعد المحجوزة في الطائرة وكانت قيمتها في البداية مثلا (80) ، حيث تمت قراءتها في الإجراء T_1، ومن ثم الغي حجز مقاعد منها بمقدار N من المقاعد ، فإذا كانت (N=5) فسيبقى (80-5=75) مقعد أي قيمة X=75، وبناء عليه سيستلم الإجراء T_2 القيمة 75 للمتغير X ليضيف إليها القيمة M ولتكن (M=4) فستصبح (X=75+4 = 79)، ولكن بسبب تزامن تنفيذ الإجراءات معا ، فإن T_2 قرأ قيمة X من البداية على أنها 80 وذلك قبل أن يقوم T_1 بطرح N وهي 5 منها، ولذا فستخرج قيمة X من T_2 تساوي (80 + 4 = 84) بدلا من (79) بسبب فقدان التعديل في الجملة (X:=X – N) أي (X:= 80 – 5 = 75).

٢- مشكلة التعديل المؤقت:

وتحدث عندما يقوم إجراء ما بتعديل قيمة عنصر ما ثم يفشل (يتعطل) هذا الإجراء لسبب ما ثم يقوم إجراء آخر باستدعاء هذا العنصر قبل أن تتم إعادته إلى قيمته الأصلية.

مثال:

T_2		T_1	
		اقرأ قيمة X	Read_item(x);
		اطرح N من X	X:=X– N;
اقرأ قيمة X	Read_item(x);	خزن قيمة X	Write_item(X);
أضف M إلى X	X:=X+ M;		
خزن قيمة X	Write_item(x);		
		اقرأ قيمة Y	Read_item(Y);
		توقف T_1 قبل أن يكمل عمله	

لاحظ أن T_1 قام بتعديل قيمة X ثم توقف قبل ان يتم عمله لذا فإن النظام يجب ان يعيد قيمة X لقيمتها الأصلية قبل التعديل ولكن قبل قيامه بذلك كان الإجراء T_2 قد قرأ القيمة المؤقتة للمتغير X والتي لم يتم تخزينها بشكل دائم (ثابت) في قاعدة البيانات بسبب تعطل T_1، وتسمى قيمة X التي تمت قراءتها من قبل T_2 (بيانات ملوثة) لأنها نتجت من إجراء لم يكتمل لذا تسمى هذه المشكلة أحيانا بمشكلة (القراءة الملوثة).

٣- مشكلة التلخيص الخاطئ:

وتحدث عندما يقوم إجراء ما بحساب تلخيص تجميعي لعدد من السجلات في حين يقوم إجراء آخر بتحديث بعض هذه السجلات، لذا فإن الاقتران التجميعي ربما يحسب بعض القيم قبل التعديل والبعض الآخر بعد التعديل.

مثال:

T_3		T_1	
اجعل المجموع صفر	Sum:=0		
اقرأ قيمة A	Read_item(A);		
أضف قيمة A للمجموع	Sum:=Sum + A		
		اقرأ قيمة X	Read_item(x);
		اطرح N من X	X:=X– N;
		خزن قيمة X	Write_item(x);
اقرأ قيمة X	Read_item(X);		
أضف قيمة X للمجموع	Sum:=Sum + X		
اقرأ قيمة Y	Read_item(Y);		
أضف قيمة Y للمجموع	Sum:=Sum + Y		
		اقرأ قيمة Y	Read_item(Y);
		أضف N إلى قيمة Y	Y:=Y+N;
		خزن قيمة Y	Write_item(Y);

تصور أن الإجراء T_3 يحسب عدد المقاعد المحجوزة في كل الرحلات في نفس الوقت الذي كان ينفذ فيه T_1 لذا فستصبح القيمة الناتجة من T_3 خاطئة لأنه قرأ قيمة X بعد طرح N من المقاعد منها أي بعد تعديلها لكنه قرأ قيمةY قبل تعديلها أي قبل إضافة N من المقاعد إليها.

٤- مشكلة القراءة غير المتكررة:

تحدث عندما يقوم إجراء ما بقراءة قيمة عنصر ما مرتين ويتم تغيير قيمة هذا العنصر من إجراء آخر بين القراءتين، لذا فسيستلم الإجراء قيم مختلفة في المرتين.

القضاء على مشاكل التزامن

١- استخدام الأقفال Locks :

القفل عبارة عن متغير يلحق بعنصر البيانات في قاعدة البيانات ويصف حالة العنصر بالنسبة للعمليات المحتمل تطبيقها عليه، ولكل عنصر بيانات في قاعدة البيانات قفل خاص به.

أنواع الأقفال:

١) ***القفل الثنائي Binary lock :*** لهذا القفل حالتان هما

أ- مقفل locked وتكون قيمته 1 .

ب- غير مقفل unlocked وتكون قيمته 0

ولكل عنصر قفل خاص به، فإذا كانت قيمة القفل 1 فإنه لا يمكن لأي عملية الوصول للعنصر والعمل عليه، ولكن إذا كانت قيمة القفل 0 فإنه يمكن لأية عملية تريد العنصر أن تصل اليه.

لذا يصبح لدينا عمليتان عند استخدام هذا النوع من الأقفال هما:

أولا: إجراء إقفال العنصر

```
Lock_item(x)
B :   if lock(x) =0 Then
            Lock(x):=1
      Else
      Begin
            Wait until (lock(x) =0)
            Go to B
      End;
```

تفسير الكلام السابق: عندما يريد إجراء ما استخدام العنصر x فإنه يصدر العملية السابقة التي تفحص قيمة القفل للعنصر x هل تساوي صفر ؟، فإذا وجدته صفر أي غير مقفل فإنها تجعله 1 (أي تغلقه) وتقوم بالعمل على هذا العنصر، أمل إذا وجدت أن قيمة القفل تساوي 1 أي أنه مغلق فإنها تضطر لانتظاره حتى يصبح متاحا ، وعندما يفرغ الإجراء من استخدام العنصر x فإنه يصدر العملية الغاء إقفال العنصر x والتي تجعل قيمة القفل صفر وبالتالي يصبح العنصر متاح لأي عملية تريده.

ثانيا: إجراء إلغاء إقفال العنصر

```
Unlock_item(x)
lock(x) =0
If any transactions are waiting then wake up one of the waiting transactions
```

٢) الأقفال المشتركة (المقتصرة) Shared (Exclusive) Locks :

بما أن القفل الثنائي محدود جدا لأنه يسمح فقط لإجراء واحد بحمل القفل على عنصر ما لذا فإننا نريد السماح لأكثر من إجراء بالوصول لنفس العنصر x إذا كان المطلوب هو قراءته، لكن إذا كان المطلوب الكتابة عليه او التحديث فإن الوصول يجب أن يكون مقتصرا على الاجراء الذي يمتلكه وهذا هو المقصود بالقفل المشترك.

وهنا نحتاج إلى ثلاث عمليات هي:

١. قفل العنصر x للقراءة read_lock (x)
٢. قفل العنصر x للكتابة write_lock (x)
٣. إلغاء إقفال العنصر x unlock (x)

أي أن القفل له ثلاث حالات : مقفل للقراءة – مقفل للكتابة – غير مقفل

- قفل القراءة يسمى قفل مشترك لأنه يسمح للإجراءات الأخرى بقراءة العنصر.
- قفل الكتابة يسمى قفل مقتصر لأنه فقط إجراء واحد يحمل القفل.

(١)إجراء إقفال العنصر للقراءة:

```
Read_lock(x)
B :     if lock (x) = "unlocked"  Then
        Begin
              Lock(x) = "read_locked";
              No-of-reads(x)=1
        End
        Else
        if lock (x) = "read_locked"  Then
              No-of-reads(x)= No-of-reads(x)+1
              Else
                    Begin
                          Wait until (lock(x) ="unlocked");
                          Go to B
                    End;
```

تفسير ذلك: افحص قيمة القفل للعنصر x فإذا كان غير مغلق قم بإغلاقه للقراءة واجعل عدد الإجراءات القارئة =1 ، أما إذا كان القفل مغلق فافحص فيما إذا كان مغلق للقراءة فإذا كان كذلك قم بالوصول للعنصر ليزداد عدد الإجراءات القارئة أي أضف 1 لعداد الإجراءات القارئة، أما إذا كان القفل مغلق للكتابة عندها يجب الانتظار حتى ينتهي الإقفال.

(٢)إجراء إقفال العنصر للكتابة:

```
Write_lock(x)
B :     if lock (x) = "unlocked"  Then
        Begin
```

```
        Lock(x) = "write_locked";
    Else
    Begin
        Wait until (lock(x) ="unlocked");
        Go to B
    End;
```

تفسير ذلك: عندما يريد الإجراء الكتابة والتحديث على العنصر xفإنه يفحص قيمة القفل فإذا وجده غير مقفل فإنه يقوم بإغلاقه ويؤدي عمله على العنصر x ، أما إذا وجده مقفل فسيضطر للانتظار حتى يصبح العنصر x متاح ويأتي دوره في العمل على العنصر x.

(٣)إجراء إلغاء إقفال العنصر:

Unlock_item(x)

```
if lock (x) = "write_locked"  Then
        Begin
            Lock(x) = "unlocked";
            wake up one of the waiting transactions, if any
        End
Else
        Begin
            if Lock (x) = "read_locked"  Then
            begin
                No-of-reads(x)= No-of-reads(x)- 1;
                If  No-of-reads(x)=0 Then
                begin
                    lock(x) ="unlocked";
                    wake up one of the waiting transactions, if any
                End;
            End;
```

تفسير ذلك: عند انتهاء الإجراء من استخدام العنصر x فإنه يفحص فيما إذا كان القفل للكتابة فإنه يقوم بفتحه (إلغاء الإغلاق)، ويقوم نظام التشغيل بفحص فيما إذا كانت هناك اجراءات أخرى تنتظر هذا العنصر فإنه يسلمه لأي منها، أما إذا كان القفل مغلق للقراءة فإنه يترك هذا العنصر ، وبالتالي يقل عدد الإجراءات التي تنتظر هذا العنصر بمقدار 1 ، ثم يفحص فيما إذا بقي إجراءات أخرى تنتظر هذا العنصر ليعطيها اليه وإلا فإنه يقوم بإلغاء القفل عنه.

المراجع:

١. قاعدة البيانات وإدارتها. د. خميس علي عمر ، د. عبد ذياب العجيلي ، جامعة القدس المفتوحة ٢٠٠٩.

٢. قاعدة البيانات المتقدمة. د. منيب قطيشات ، د. محمد قاسم عجاجة ، جامعة القدس المفتوحة ٢٠١١.

٣. Fundamentals of database systems / Ramez Elmasri, Shamkant B. Navathe

٤. صفحات الانترنت