أكاديمية التعلم
Academy Of Learning

موضوعات سيبرانية مختارة

الفهرس

- تنفيذ خطة الاستجابة للحادث
- الاستعادة و التعافي
- إعداد التقارير
- المتابعة / الدروس المستفادة

- المعايير و أفضل الممارسات
  - NIST
  - حل الوسط
  - جعل الأمن قابل للقياس
- استمرارية العمل
- التوعية و التدريب

الفصل الثالث

مقدمة في تطوير أمن البرمجيات

- مفهوم عملية تطوير البرمجيات
- مفهوم البرمجيات الآمنة
- الثغرات البرمجية التي يرتكبها المبرمجين و كيفية تفاديها
- دورة حياة تطوير البرمجيات SDLC
- تحديات تطوير البرمجيات
- المبادئ التوجيهية لأمن البرمجيات
- الممارسات الأساسية لأمن البرمجيات
- اختبار الأمن

الفصل الأول

مقدمة في قواعد البيانات

- مفهوم قواعد البيانات ووظائفها
- أنظمة إدارة قواعد البيانات
- النموذج العلائقي لقواعد البيانات
- لغات قواعد البيانات
- مقدمة في لغة QSL
  - مفاهيم SQL الأساسية
  - أنواع بيانات SQL في أنظمة Oracle
  - إنشاء الجداول و تعديلها و إضافة القيود و حذفها
  - استخراج البيانات بجملة Select
  - استرجاع البيانات من أكثر من جدول
  - معالجة البيانات (إضافة - تعديل – حذف)

مقدمة في قواعد البيانات

مفهوم قواعد البيانات ووظائفها

قواعد البيانات هي مجموعة من البيانات المنظمة والمرتبطة بموضوع معين داخل جدول أو مجموعة من الجداول بغرض استرجاعها لاتخاذ القرارات المطلوبة

هي مجموعة من البيانات المرتبطة ببعضها البعض المخزنة ، والمنظمة بأسلوب يسمح باستخلاص المعلومات التي يتم الاحتياج لها بسهولة. وهي مجموعة من عناصر البيانات المرتبطة مع بعضها البعض بعلاقة رياضية.

مكونات قاعدة البيانات:

وتتكون قاعدة البيانات (Database) من جدول واحد أو أكثر من جدول.

ويتكون الجدول من سجل (Record) أو أكثر من سجل

ويتكون السجل من حقل (Field) أو أكثر من حقل.

الهدف منها هو الحاجة إلي إدارة حجم كبير من البيانات، والتعامل معها وإدارتها بهدف تحسين وتسهيل طرق التعامل مع حجم كبير من البيانات من قبل عدد كبير من المستخدمين.

عند جمع البيانات عن المتدربين في كلية معينة، فإنه من المعروف أن لكل متدرب بيانات مثل (رقم المتدرب الأكاديمي، اسم المتدرب، القسم الذي ينتمي إليه، الشعبة ...)، فلو تم جمع بيانات كل متدرب في بطاقة و تم تسميته (سجل المتدرب RECORD) وكل بيان من بيانات المتدرب في هذا السجل يسمي (حقل FIELD) معنى ذلك أ ن الحصول على سجلات للمتدربين. عند جمع هذه السجلات مع بعضها يتم الحصول على قاعدة بيانات للمتدربين تسمى DATABASE

وظائف قواعد البيانات:

١- تخزين البيانات

٢- تنظيم البيانات

٣- فهرسة البيانات بطريقة يمكن بها استرجاعها ومن ثم معالجتها بسهولة

٤- إعداد النماذج التي تسهل إدخال البيانات

٥- البحث والاستعلام عن بيانات مخزنة في قاعدة البيانات

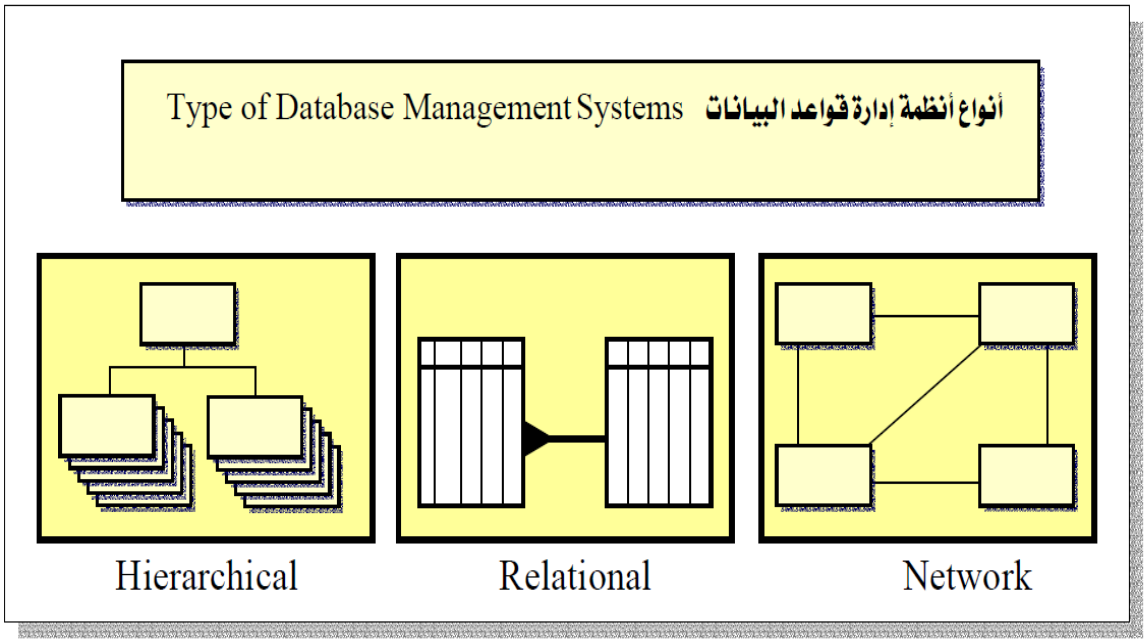
٦-عرض البيانات والمعلومات في صورة تقارير يمكن منها استخلاص المعلومات وتحليلها وتسهل دعم واتخاذ القرار

٧- إمكانية تبادل البيانات والمعلومات بين قواعد البيانات الأخري

أنظمة إدارة قواعد البيانات

نظام إدارة قاعدة البياناتDatabase Management System DBMS

هو البرنامج الذي يتم من خلاله استرجاع البيانات، أو الإضافة أو التعديل عليها، أو حذفها حيث يقوم البرنامج بالربط بين المستخدم وبين محرك قاعدة البيانات، لأداء تلك المهمة.



من الأمثلة على نظم إدارة قواعد البيانات

- قاعدة البيانات أكسس Access من شركة ميكروسوفت
- أوراكلOracle

أهم وظائف نظام إدارة قاعدة البيانات

١- نمذجة البيانات:

حيث تُنظم البيانات في صفوف وأعمدة وجداول، ويتم فهرستها لتسهيل الوصول إلى المعلومات ذات الصلة.

٢- إدارة تخزين البيانات:

يوفر نظام DBMS الحديث مساحة تخزين للبيانات، بالإضافة لنماذج إدخال البيانات، أو تعريفات الشاشة، وتعريفات التقارير، وقواعد التحقق من صحة البيانات.

٣- التحكم في التزامن:

لضمان الدقة والوصول المتزامن لقاعدة البيانات من قبل العديد من المستخدمين.

٤- أمن المعلومات:

يتم تحديد المستخدمين الذين يمكنهم الوصول إلى قاعدة البيانات، وعناصر البيانات التي يمكن الوصول إليها، وعمليات البيانات (الإضافة، الحذف، أو التعديل) التي يمكن للمستخدم إجراؤها.

٥- إدارة النسخ الاحتياطي والاسترداد:

هو أمر بالغ الأهمية للحفاظ على سلامة قاعدة البيانات، حيث يؤمن نظام DBMS النسخ الاحتياطي واستعادة البيانات من خلال أدوات مساعدة خاصة تسمح بتنفيذ إجراءات النسخ الاحتياطي والاستعادة الروتينية لقاعدة البيانات بعد حدوث عطل ما، مثل قطاع تالف في القرص أو انقطاع التيار.

النموذج العلائقي لقواعد البيانات

من أقوى أنظمة قواعد البيانات لقدرته الفائقة على استيعاب كميات كبيرة من البيانات دون التأثير على أدائه من حيث السرعة الفائقة والدقة، ولأن هذا النظام يتمتع بالسرية والأمان لاحتوائه على نظام إعطاء الصلاحيات والحقوق لمستخدميه ولسهولته في الاستخدام والفهم وسهولة برمجة تطبيقاته.

لقد صُمم نموذج البيانات العلائقية Relational Data Model في العام ١٩٧٠ بواسطةC.F. Codd ، وهو النموذج الأكثر استخدامًا في يومنا هذا، كما يُعَدّ الأساس لكل من:

- البحث العلمي في نظرية البيانات، والعلاقات، والقيود.

- العديد من منهجيات تصميم قواعد البيانات.

-لغة الوصول القياسية إلى قاعدة البيانات، حيث تسمى لغة الاستعلام المهيكلة - structured query language أي SQLاختصارًا

- جميع أنظمة إدارة قواعد البيانات التجارية الحديثة.

- يصف نموذج البيانات العلائقية العالم على أنه تجميعة من العلاقات والجداول المترابطة

لغات قواعد البيانات

لغة الاستعلام الهيكلية (SQL)

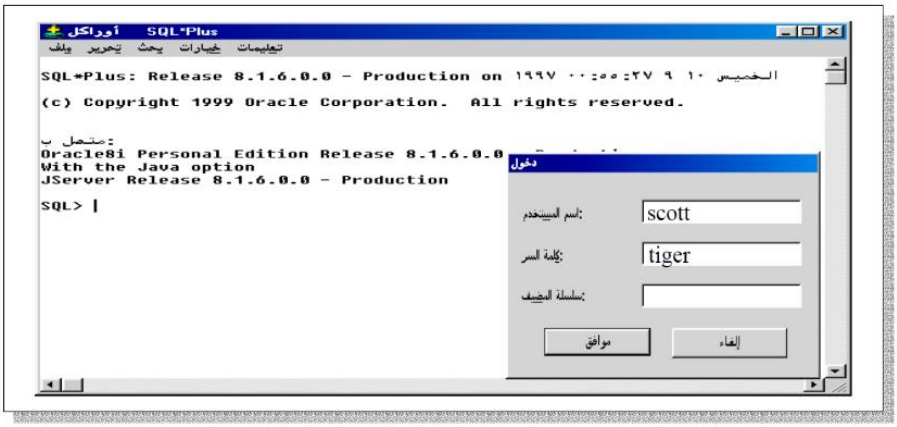
هي لغة تستخدم لإصدار جميع الأوامر التي تتعلق بقاعدة البيانات، وتنقسم هذه اللغة إلى خمسة أقسام رئيسة يمكن من خلالها إصدار الأوامر الخاصة بكل قسم، والجدول التالي يوضح الأقسام المختلفة من هذه اللغة ووصف الأوامر لكل قسم.

القسـم	الأمـر	وصـف الأمر
Data Retrieval	SELECT	أمر استرجاع البيانات من جدول أو كائن
(DML) Data Manipulation Language	INSERT	أمر إضافة بيانات إلى جدول أو كائن
	UPDATE	أمر التعديل في بيانات جدول أو كائن
	DELETE	أمر حذف بيانات جدول أو كائن
(DDL) Data Definition Language	CREATE	أمر إنشاء جدول أو كائن
	Alter	أمر التعديل في جدول أو كائن
	DROP	أمر إلغاء جدول أو كائن
	RENAME	أمر تغيير الاسم جدول أو كائن
	TRUNCATE	إلغاء جزء أو بتر جزء من جدول أو كائن
Transaction Control	COMMIT	تثبيت البيانات في الجدول
	ROLLBACK	الرجوع عن تثبيت البيانات
	SAVEPOINT	الرجوع لنقطة معينة
(DCL) Data Control Language	GRANT	إعطاء الصلاحيات للمستخدمين للدخول على البيانات
	REVOKE	سحب الصلاحيات من المستخدمين

مقدمة في لغة QSL

قامت شركة أوراكل باعتماد لغة تسمى لغة الاستفسارات (Structured Query Language SQL) للتعامل مع قواعد البيانات العلائقية وهي لغة سهلة تقوم بإنشاء الأشياء (Objects) الخاصة بقاعدة البيانات مثل الجداول والتعامل معها وتقوم بعمل جميع الاستفسارات اللازمة المطلوب معرفتها من قاعدة البيانات ويطلق عليها (SQL) ، كما قامت شركة أوراكل أيضاً بعمل تطبيق أو بيئة تستقبل الأوامر الخاصة بلغة الاستفسارات SQL وهذه البيئة تسمى محرر الـ (SQL*PLUS) ويمكن من خلال هذا المحرر استقبال الأوامر الخاصة بلغة SQL وتنفيذها وتعديل الأخطاء الموجودة في الأمر وجميع العمليات الأخرى

مفاهيم SQL الأساسية



محرر (بيئة) الـ SQL * PLUS

الشكل يبين شاشة الدخول على محرر SQL*PLUS حيث تقوم بكتابة اسم المستخدم وهو (SCOTT) وكلمة المرور وهي (TIGER) ثم الضغط على مفتاح (موافق) وذلك للدخول على المحرر الذي يستقبل أوامر لغة الاستفسارات SQL. علماً بأن اسم المستخدم وكلمة المرور يمكن أن تتغير وذلك على حسب المستخدم هل له صلاحية الدخول أم لا، فمن الممكن أن تدخل على المحرر باسم المستخدم (SYSTEM) وكلمة المرور (MANAGER) وفي هذه الحالة تدخل عليه وكأنك (مدير قاعدة البيانات) ، ويتكون محرر SQL * PLUS من قائمة تساعدك على تحرير الأمر والتعديل فيه وتنفيذه ومنها على سبيل المثال

- **SQL > EDIT**

يستخدم هذا الأمر لتعديل آخر أمر تم كتابته على محرر SQL*PLUS، وعند تنفيذ هذا الأمر ستظهر لك شاشة (المفكرة) وبها آخر أمر تم كتابته حيث يمكن من خلال هذه الشاشة التعديل في الأمر ثم حفظه وتنفيذه مرة أخرى من خلال محرر SQL*PLUS، ويمكن اختصار هذا الأمر فيكتب كالآتي

SQL > ED

- **SQL>RUN**

يستخدم هذا الأمر لإعادة تنفيذ آخر أمر تم كتابته في محرر SQL*PLUS ، ويمكن كتابة هذا الأمر بالشكل التالي : SQL>R.

- **SQL>SPOOL Filename**

يستخدم هذا الأمر عندما نريد حفظ كل ما تم عمله داخل محرر SQL*PLUS في ملف نصي بامتداد (LST) وذلك بغرض استرجاعها ومراجعتها، ومن الممكن أن نحصل على نسخة مطبوعة بواسطة الأمر التالي : **SQL>SPOOL OUT .**

- **SQL>SAVE file name**

يستخدم هذا الأمر لحفظ الأوامر في ملف وذلك لاسترجاعها مرة أخرى وتنفيذها وهنا لا بد من حفظ الملف بامتداد (sql) وذلك لنتمكن من تشغيله مرة أخرى. فإذا أردنا حفظ أمر ما داخل ملف اسمه test.sql نكتب الأمر التالي : SQL>SAVE test.sql.

- **SQL>GET filename**

يستخدم هذا الأمر لاسترجاع الأوامر التي تم حفظها بواسطة الأمر السابق، وذلك لتنفيذها مرة أخرى، فإذا أردنا استرجاع الأوامر من الملف test.sql نكتب الأمر التالي: SQL>GET test.sql.

- **SQL>START filename**

يستخدم هذا الأمر في تنفيذ الأوامر الموجودة في ملف تم حفظه بامتداد sql، فإذا أردنا تنفيذ الأوامر الموجودة في الملف test.sql مثلاً نقوم بكتابة الأمر التالي:

SQL>START test>sql

- **SQL>@ filename**

هذا الأمر مثل السابق تماماً.

- **SQL>LIST**

يستخدم هذا الأمر في استعراض سطور آخر أمر تم كتابته، ويمكن استعراض سطور معينة فمثلاً لو أردت استعراض السطور من ١ إلى ٣ نكتب الأمر كالتالي:

SQL>L 1 3

أنواع بيانات SQL في أنظمة Oracle

يوجد أنواع للبيانات التي تخزن داخل الجدول وهذه البيانات إما أن تكون بيانات حرفية أو عددية أو بيانات تاريخ أو بيانات أخرى والجدول التالي يبين أنواع البيانات المختلفة:

نوع البيانات	الوصف
Varchar 2 (الحجم)	تستخدم مع البيانات الحرفية المتغيرة الطول
Char(الحجم)	تستخدم مع البيانات الحرفية الثابتة الطول لا بد من تحديد طول البيانات الحرفية
Number(P,s)	الجزء Pتستخدم مع البيانات الرقمية ويمثل الحرف يمثل (s)الصحيح قبل العلامة العشرية، والحرف الجزء العشري بعد العلامة العشرية.
Date	تستخدم مع بيانات التاريخ والوقت
Long	تستخدم لتمثيل البيانات الكبيرة الحجم التي تصل إلى (٢) جيجابايت
CLOB-BLOB	تستخدم لتمثيل البيانات الكبيرة مثل الصور والرسومات التي تصل حجمها إلى أكثر من (٤) جيجا بايت
Bfile	تستخدم لتخزين الملفات الكبيرة والخارجية والتي يصل حجمها إلى أكثر من (٤) جيجابايت.

تستخدم الأنواع السابقة في تحديد نوع البيانات لكل عمود عند إنشاء الجدول.

إنشاء الجداول و تعديلها و إضافة القيود و حذفها

لغة تعريف البيانات (Data Definition Language) والتي عادة ما يرمز لها بـ (DDL)، وهذه اللغة هي التي تمكننا من إنشاء وتعديل وإلغاء أي كائن داخل قاعدة البيانات ، وكما هو معروف أن قاعدة البيانات تتكون من كائنات مختلفة وأهم هذه الكائنات هي الجداول (TABLES)

الكائن	وصف الكائن
Table	هو الوحدة الأساسية لمكونات قاعدة البيانات والتي نستخدمها في حفظ البيانات ويتكون من عدة صفوف وأعمدة.
View	المناظير: عبارة عن جزء مؤقت من جدول معين يتكون من عدة صفوف وأعمدة ويستخدم لغرض معين بشكل مؤقت.
Sequence	سلسلة: عبارة عن سلسلة تستخدم لتوليد أرقام متتالية بشكل معين دون تكرار لذلك يفضل استخدامها لتسجيل بيانات المفتاح الأساسي داخل الجدول
Index	فهرس: ويستخدم في عملية فهرسة بعض الأعمدة لتسهيل عملية البحث فيها عن معلومة معينة، وأيضاً لتقليل وقت الاستفسارات من الجداول.
Synonym	مرادفات: تستخدم لإعطاء أكثر من اسم على كائن معين.

الشروط التي يجب توافرها عند اختيار اسم الجداول أو أسماء الأعمدة:

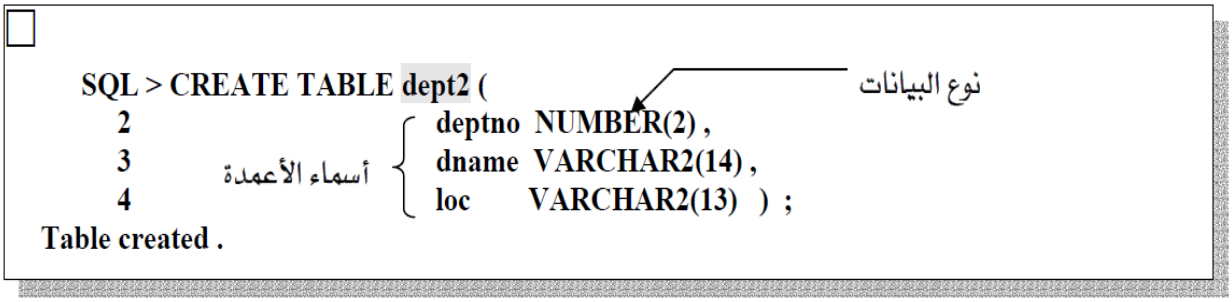
- يجب أن يبدأ اسم الجدول أو اسم العمود بحرف.
- يجب ألا يزيد طول الاسم عن (٣٠) حرفاً.
- من الممكن أن يتكون من حروف كبيرة وصغيرة وأرقام ورموز خاصة مثل (#, $, _).
- يجب ألا يتكرر اسم الجدول أكثر من مرة داخل قاعدة البيانات الواحدة.
- يجب ألا يتكرر اسم عمود أكثر من مرة داخل الجدول الواحد.
- يجب ألا يكون من الأسماء المحجوزة لأوراكل مثل (FROM, SELECT,.....)
- يفضل أن يكون اسم الجدول له معنى بحيث يعبر عن نوع بيانات الجدول

إنشاء الجداول Create tables

الصيغة العامة لإنشاء الجداول

```
SQL > CREATE table  أسم الجدول  (
                    العمود١  نوع البيانات ,
                    العمود٢  نوع البيانات ,
                    العمود٣  نوع البيانات  ) ;
```

مثال (١) : إنشاء جدول الإدارات (dept 2)



في المثال السابق تم إنشاء جدول الإدارات (dept2) والذي يتكون من ثلاثة أعمدة، العمود الأول نوعه رقمي وطوله حرفان ، والعمود الثاني نوعه حرفي وطوله (١٤ حرفاً)، وكذلك العمود الثالث نوعه حرفي وطوله (١٣ حرفاً)، وعندما نريد عرض البناء الداخلي للجدول الذي تم إنشاؤه نقوم بكتابة الأمر التالي:

```
SQL > DECRIBE dept2 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
  DEPTNO                                            NUMBER(2)
 DNAME                                             VARCHAR2(14)
 LOC                                               VARCHAR2(13)
```

التعديل في الجداول باستخدام ALTER TABLE

توفر لنا لغة الاستعلام أوراكل (SQL) إمكانية مهمة جداً وهي إمكانية التعديل في هيكل (البناء الداخلي) لجدول تم انشاؤه مسبقاً باستخدام الأمر ALTER TABLE، وعملية التعديل في الجداول تشتمل على ثلاث إمكانيات وهي إما إضافة أعمدة جديدة على الجدول أو التعديل في نوع بيانات عمود معين أو التعديل في نوع بيانات عمود معين أو إلغاء عمود معين. كما هو موضح بالجدول التالي:

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

مثال إضاف عمود جديد يسمى (REGION) إلى جدول الإدارات DEPT2

```
SQL> ALTER TABLE   dept2
  2  ADD        ( region VARCHAR2(20) );
Table altered .
```

في المثال السابق قمنا بإضافة عمود جديد يسمى (region) إلى جدول الإدارات وطبعاً هذا العمود لا يحتوي على أية بيانات ويمكن حفظ بيانات المناطق لكل إدارة، ويظهر هذا العمود كآخر عمود عند الاستعلام، وللتأكد من إضافة هذا العمود ننفذ الأمر التالي:

SQL > DECRIBE dept2 ;

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		V ARCHAR2(13)
REGION		V ARCHAR2(20)

مثال : التعديل في طول بيانات العمود (DNAME) ليصبح بطول ٢٠ بدلاً من ١٤

```
SQL> ALTER TABLE   dept2
  2  MODIFY     ( dname VARCHAR2(20) );
Table altered .
```

في المثال السابق تم التعديل في نوع بيانات العمود dname ليصبح طوله (20) VARCHAR بدلاً من (14) VARCHAR ، ويجب أخذ الملحوظات التالية في الاعتبار عند التعديل في أعمدة الجدول:

- يمكن زيادة حجم (طول) البيانات للأعمدة.
- يمكن تغيير نوع البيانات من نوع إلى آخر بحيث لا يؤثر ذلك في بيانات الأعمدة إذا كانت موجودة.
- لا يمكن تقليل حجم (طول) الأعمدة إذا كانت تحتوي على بيانات.

مثال : إلغاء العمود المسمى REGION من جدول الإدارات DEPT

```
SQL> ALTER TABLE   dept2
  2   DROP COLUMN    REGION  ;
Table altered .
```

في المثال السابق تم إلغاء العمود (REGION) من جدول الإدارات (DEPT2) ، ويجب أن نأخذ في الاعتبار الملحوظات التالية عندما نريد إلغاء أعمدة من جدول معين:

- يجب أن يكون العمود المراد إلغاؤه فارغاً من البيانات.
- لا يمكن إلغاء أكثر من عمود واحد فقط في الأمر الواحد.
- يجب أن يتبقى عمود واحد على الأقل بعد عملية الإلغاء داخل الجدول.
- لا يمكن استعادة العمود بعد إلغائه.

إلغاء جدول باستخدام الأمر DROP

عملية إلغاء الجدول هي عبارة عن إلغاء الجدول تماماً من قاعدة البيانات وحذف كل بياناته وكل القيود المتعلقة به، وبالتأكيد لا يمكن استعادته مرة أخرى.

مثال : إلغاء الجدول المسمى (DEPT 30)

```
SQL> DROP TABLE   dept 30 ;
Table dropped .
```

تغيير اسم جدول معين إلى اسم آخر باستخدام الأمر RENAME

لتغيير اسم الجدول (DEPT2) ليصبح اسمه مثلاً (DEPARTMENT) ونقوم بكتابة الأمر كما هو مبين في المثال التالي:

مثال تغيير اسم جدول الإدارات (DEPT 12) ليصبح باسم (DEPARTMENT).

```
SQL> RENAME   dept2   TO  deptartment
Table renamed.
```

في المثال السابق تم تغيير اسم جدول الإدارات ليصبح بالاسم (department)، ويجب أن يكون مالك الجدول هو الذي يقوم بتغيير الاسم، أي إنه لا يمكن لأحد من المستخدمين تغيير اسم الجدول إلا إذا كان هو مالك الجدول.

أنواع الجداول في بيئة قواعد البيانات أوراكل

يوجد في بيئة قواعد البيانات أوراكل نوعان من الجداول هما كالتالي:

- جداول تنشأ عن طريق المستخدمين USER TABLES، وهي مجموعة من الجداول ينشئها المستخدم ويتم التعامل معها من خلاله.
- جداول يتم إنشاؤها عن طريق خادم أوراكل (ORACLE SERVER)، وهي مجموعة من الجداول تسمى (DATA DICTIONARY)، يتم إنشاؤها بواسطة أوراكل وهي تحتوي على معلومات عن قاعدة البيانات.

وتنقسم إلى عدة فئات هي:

- الفئة (USER) وتحتوي على معلومات حول الكائنات الخاصة بالمستخدمين مثل الجداول.
- الفئة (ALL) : وتحتوي على معلومات عن كل الجداول والعلاقات التي يمكن للمستخدمين الدخول عليها.
- الفئة (_DBA): وتحتوي على معلومات خاصة بمدير قواعد البيانات (DBA) ولا يمكن لأحد الدخول عليها واستخدامها.

.(DATA DICTINARY)مثال (٩): عرض معلومات حول الجداول التي يملكها المستخدمون باستخدام

```
SQL> SELECT   *
  2  FROM     user_tables ;
```

مثال: عرض اسم ونوع الكائنات التي يمتلكها المستخدمون.

```
SQL> SELECT   object_name , object_type
  2  FROM     user_objects  ;
```

مثال: عرض أسماء ونوع الجداول والكائنات التي أنشأها المستخدم.

```
SQL> SELECT   *
  2  FROM     user_catalog  ;
```

القيود

هي عبارة عن شروط معينة توضع على الجداول لتنظيم العمليات المختلفة التي تتم على هذه الجداول مثل الإضافة والتعديل والحذف، فمثلاً عندما ننشئ جدولاً جديداً للموظفين ويتضمن هذا الجدول عموداً لتخزين رقم الموظف، وعندما نريد إضافة موظف إلى هذا الجدول، ربما نضيف بيانات موظف جديد ويكون رقمه مثلاً (١٠٠) ويكون هناك موظف مسجل بنفس الرقم داخل الجدول، في هذه الحالة يتم تخزين الرقم (١٠٠) في عمود رقم الموظف مرتين، أي أننا قمنا بتسجيل موظفين برقم واحد، وهذا غير منطقي أن يتم تسجيل موظفين أو أكثر برقم واحد، ولهذا فإننا نقوم بعمل قيد (Constraint) على العمود (رقم الموظف) حتى لا يقبل رقم مكرر.

أنواع القيود Constraint

الجدول التالي يبين الأنواع المختلفة من القيود ومعنى كل منها

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

إنشاء القيود Create a Constrait

تُنشأ القيود بطريقتين هما:

- عمل القيود أثناء إنشاء الجدول.
- عمل القيود بعد إنشاء الجدول.

وتطبق القيود على مستوى الأعمدة أو على مستوى الجدول

القيد PRIMARY KEY.

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول، ومعنى هذا القيد هو إنشاء مفتاح أساسي (Primary Key) داخل الجدول وذلك لتمييز عمود معين بحيث إن هذا العمود يكون له خاصيتان هما:

١- عدم قبول تكرار القيم داخله.
٢- عدم السماح بترك قيمته فارغة (NULL).

فمثلاً إذا أردنا تمييز المتدربين فإننا نميزهم عن طريق الرقم الأكاديمي فهذا الرقم يجب أن يكون رقماً وحيداً لا يتكرر فكل متدرب يحمل رقماً أكاديمياً وحيداً خاصاً به وأيضاً يجب أن يكون لكل متدرب رقم أكاديمي فليس من المعقول أن نسجل بيانات متدرب دون تسجيل رقمه، ولتحقيق ذلك نقوم بعمل عمود داخل جدول المتدربين ونطبق عليه القيد (primary Key).

مثال (١): إنشاء القيود أثناء إنشاء جدول الإدارات وتطبيقها على مستوى الأعمدة.

```
SQL > CREATE TABLE dept (
  5                    deptno  NUMBER(2)     PRIMARY KEY,
  6   أسماء الأعمدة    dname   VARCHAR2(14)     NOT NULL ,
  7                    loc     VARCHAR2(13)
  8                    ) ;
  9                                          القيود على مستوى الأعمدة
Table created .
```

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيدين على العمودين (deptno,dname)،

القيد الأول

هو (Primary Key) على العمود deptno لجعل هذا العمود (مفتاح أساسي) داخل الجدول، حيث إن رقم الإدارة يجب أن لا يتكرر داخل العمود وأيضاً عدم ترك قيمته فارغة.

القيد الثاني

(not null) على العمود dname يمنع ترك قيمة هذا العمود فارغة بدون بيانات فلا بد من إدخال اسم الإدارة عند إضافة إدارة جديدة إلى جدول الإدارات، والجدير بالذكر هنا أن القيد (NOT NULL) يطبق فقط على مستوى العمود أي إننا لا نستطيع تطبيقه على مستوى الجدول.

مثال(٢): إنشاء القيود أثناء إنشاء جدول الإدارات وتطبيقها على مستوى الجدول.

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

Table created .                                  القيد على مستوى الجدول
```

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (primary key) على مستوى الجدول كما هو واضح في السطر رقم (٥)، والفرق بين هذه الطريقة والطريقة المستخدمة في المثال رقم (١) هو أننا نستطيع حذف القيد إذا كان على مستوى الجدول ، ونلاحظ في هذا المثال أننا سمينا القيد بـ(dept_deptno_pk)

القيد UNIQUE KEY

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول، ومعنى هذا القيد هو عدم السماح بتكرار القيم داخل العمود.

مثال(٣) : إنشاء القيد (UNIQUE) أثناء إنشاء جدول الإدارات وتطبيقه على مستوى الجدول.

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

Table created .                                  القيد على مستوى الجدول
```

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (unique) على العمود (dname) وذلك بغرض عدم تكرار اسم الإدارات داخل الجدول، ونلاحظ أننا طبقنا القيد على مستوى الجدول ويمكن تطبيقه على مستوى العمود.

القيد FOREIGN KEY

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول، ويستخدم هذا القيد عندما نريد ربط جدولين ببعض، فمثلاً لربط جدول الموظفين بجدول الإدارات بغرض معرفة موظفي إدارة معينة، فإنه لا بد من وجود عمود (primary key) داخل جدول الإدارات ونفس هذا العمود يوجد في جدول الموظفين ويسمى (foreign key) ، كما في الشكل التالي:

EMP table

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

DEPT table

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

Foreign Key Primary Key

ولتنفيذ هذا الربط لا بد من تطبيق القيد (foreign key) على العمود (deptno) داخل جدول الموظفين كما هو واضح من المثالين التاليين:

مثال (٤): إنشاء القيد (foreign key) على العمود deptno في جدول الموظفين وتطبيقه على مستوى العمود.

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

Table created .
```

القيد على مستوى العمود

في المثال السابق تم إنشاء جدول الإدارات (emp) كما تم إنشاء قيد (foreign key) على العمود (deptno)، كلمة (REFERENCES) تشير إلى أنه تم تطبيق القيد (foreign key) على العمود (deptno) الذي يشير إلى العمود (deptno) داخل الجدول DEPT . وبهذه الطريقة تم تطبيق القيد على مستوى العمود.

أما تطبيق القيد على مستوى الجدول فسوف يبين في المثال التالي:

مثال(٥): إنشاء القيد (foreign key) على العمود deptno في جدول الموظفين وتطبيقه على مستوى الجدول.

```
SQL > CREATE TABLE emp (
  2         empno    NUMBER(4) ,
  3         ename    VARCHAR2(10)  NOT NULL ,
  4         job      VARCHAR2(9) ,
  5         mgr      NUMBER(4) ,
  6         hiredate DATE ,
  7         sal      NUMBER(7,2) ,
  8         comm     NUMBER(7,2) ,
  9         deptno   NUMBER(2) ,
 10      CONSTRAINT emp_deptno_fk  FOREIGN KEY (deptno)
 11                  REFERENCES dept(deptno)  ) ;

Table created .
```

القيد على مستوى الجدول

القيد CHECK

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول، ويستخدم هذا القيد عندما نريد تحديد مجال قيم معينة لعمود في الجدول، فمثلاً إذا أردنا أن نحدد القيم المدخلة للعمود deptno داخل جدول الإدارات بحيث تكون هذه القيم محصورة بين (٩٩ ، ١٠) فإننا نطبق هذا القيد، كما هو واضح من المثال التالي:

مثال(٦): إنشاء القيد (CHECK) على العمود DEPTNO في جدول الإدارات وتطبيقه على مستوى الجدول.

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

Table created .                                   القيد على مستوى الجدول
```

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (check) على العمود deptno بشرط أن تكون القيم المدخلة لهذا العمود محصورة بين ٩٠ و ١٠.

إضافة قيود على الجداول Adding Constrait

يتم إضافة القيود على الجدول بعد إنشائه باستخدام الأمر Alter Tableكما في الصيغة العامة التالية:

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

والأمثلة التالية توضح كيفية إضافة القيود على الجداول التي تم إنشاؤها سابقاً.

مثال: إضافة القيد (PRIMARY KEY) على العمود DEPTNO في جدول الإدارات.

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

Table altered .
```

في المثال السابق تم : إضافة القيد (PRIMARY KEY) على العمود DEPTNO في جدول الإدارات وذلك في حالة عدم إنشائه أثناء إنشاء الجدول.

إزالة القيود من الجداول DROP

يتم إزالة القيود من الجدول عن طريق استخدامنا لأمر Alter Table

استعراض القيود المطبقة على جدول معين:

أوراكل يقوم بإنشاء جداول لتسجيل التغيرات التي تتم في قاعدة البيانات وهذه الجداول تسمى Data Dictionary. ومن خلالها يمكن عرض القيود المطبقة على جدول معين كما هو واضح من المثال التالي: عرض القيود المختلفة المطبقة على جدول الموظفين

```
SQL > SELECT constraint_name , constraint_type
  2   FROM    user_constraints
  3   WHERE   table_name = 'EMP' ;
```

CONSTRAINT_NAME	C
SYS_C00674	C
SYS_C00675	C
EMP_EMPNO_PK	P
FK_DEPTNO	R

في المثال السابق عرض أسماء القيود وأنواعها المطبقة على جدول الموظفين، وكما نلاحظ أن ناتج عمود أنواع القيود (Constraint_type) عبارة عن حرف واحد يدل على نوع القيد ومعنى هذه الحروف هي كما يلي:

- الحرف C يعني أن نوع القيد هو CHECK.
- الحرف P يعني أن نوع القيد هو Primary Key.
- الحرف R يعني أن نوع القيد Foreign Key.
- الحرف U يعني أن نوع القيد هو UNIQUE.

أما نوع القيد NOT NULL فيظهر مثل القيد CHECK.

والمثال التالي يوضح كيفية عرض القيود المطبقة على الأعمدة.

مثال : عرض أسماء الأعمدة وأسماء القيود المطبقة عليها.

```
SQL > SELECT  constraint_name , column_name
  2     FROM  user_cons_column
  3     WHERE   table_name = 'EMP '  ;
```

استخراج البيانات بجملة Select

الصيغة (الشكل) العام لجملة SELECT.

```
SELECT      * or Columns [alias]
FROM        Table
WHERE       condition or conditions
ORDER BY    Column or Alias [ASC or DESC] ;
```

الصيغة العامة :

SELECT	تستخدم في بداية الأمر لاسترجاع بيانات من الجداول.
*	هذا الرمز يستخدم عند استرجاع جميع الحقول من الجداول.
Alises	الأسماء المستعارة للحقول.
FROM	تستخدم للإعلان عن اسم الجدول.
Table	اسم الجدول المراد استرجاع البيانات منه.
Where	تستخدم للإعلان عن الشرط أو الشروط.
Conditions	الشرط أو الشروط اللازمة لحصر البيانات الآتية من الجدول.
ORDERED BY	تستخدم للإعلان عن كيفية ترتيب البيانات المسترجعة من الجدول.
Column or Alies	اسم الحقل أو الحقول أو الأسماء المستعارة المراد الترتيب بها.
;	فاصلة منقوطة للإعلان عن نهاية الأمر.

متطلبات وإرشادات كتابة جمل SQL.

يوجد هناك بعض الإرشادات التي يجب مراعاتها عند كتابة جملة SQL لتكون الجملة صحيحة وقابلة للتنفيذ، وهذه الإرشادات هي:

١-يمكن كتابة مكونات جملة SQL بالأحرف الكبيرة أو الصغيرة فهذا لا يؤثر على سلامة الجملة وذلك لأن جملة SQL غير حساسة للحروف Not Case Sensitive.

٢-يفصل بين أسماء الحقول باستخدام الفاصلة (,).

٣-يمكن كتابة جملة SQL في عدة سطور فهذا لا يؤثر في صحة الجملة.

٤-لا يمكن فصل الكلمات المحجوزة للغة أو اختصارها، والكلمات المحجوزة تسمى Keywords وهي مثل (SELECT , FROM , WHERE, ORDERED BY).

٥-يفضل كتابة الجملة على أسطر ليسهل قراءتها وفهمها.

٦-لا بد من الإعلان عن نهاية الجملة بواسطة (;) .

ملحوظة: أوامر محرر SQL*PLUS لا يوضع بعدها الفاصلة المنقوطة (;) .

لتنفيذ جملة SQL:

لتنفيذ جملة SQL من الممكن استخدام إحدى الطرق التالية:

١- نضع الفاصلة المنقوطة (;) في نهاية الجملة.
٢- نضع علامة (/) في نهاية الجملة عند مؤشر SQL >.
٣- نكتب الأمر RUN عند مؤشر SQL >.

مثال (١): عرض جميع الحقول من الإدارات DEPT .

في هذا المثال نقوم بجمع

```
SQL> SELECT *
2 FROM       dept ;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

(DEPTNO , DNAME, LOC) الذي يحتوي على الأعمدة التالية DEPTالحقول والبيانات الموجودة في جدول الإدارات وذلك باستخدام الرمز (*) والذي يعني إظهار جميع حقول الجدول، لاحظ أن أسماء الحقول دائماً تظهر بالحروف الكبيرة.

مثال (٢): عرض حقول معينة من جدول الإدارات DEPT.

```
SQL> SELECT  deptno , dname
  2    FROM dept  ;
```

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

يوضح المثال السابق كيفية إظهار حقول معينة من الجدول، ونلاحظ هنا بأن أسماء الحقول يفصل بينها الفاصلة (,) ، فالمثال يقوم بعرض جميع أرقام الإدارات وأسمائها فقط من جدول الإدارات (DEPT).

استرجاع الحقول بأسماء مستعارة (Aliases):

نستخدم طريقة الأسماء المستعارة (Aliases) عندما نريد إظهار الحقل باسم غير اسمه الموجود في الجدول وذلك لتوضيح معنى الحقل مثلاً. وهناك ثلاثة طرق لإظهار الحقول بأسماء مستعارة :

١- استخدام كلمة (AS) بين اسم الحقل والاسم المستعار.
٢- استخدام المسافة (Space) بين اسم الحقل والاسم المستعار.
٣- استخدام علامة التنصيص المزدوجة التالية (" ") عندما يكون الاسم المستعار أكثر من كلمة. والمثال التالي يوضح ذلك.

مثال (٣) : عرض حقول بأسماء مستعارة من جدول الموظفين.

```
SQL> SELECT ename AS name , sal salary ,  job "employee job"
  2  FROM    emp  ;
```

NAME	SALARY	employee job
SMITH	800	CLERK
ALLEN	1600	SALESMAN
WARD	1250	SALESMAN
JONES	2975	MANAGER
MARTIN	1250	SALESMAN
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
SCOTT	3000	ANALYST
KING	5000	PRESIDENT
TURNER	1500	SALESMAN
ADAMS	1100	CLERK

يوضح المثال السابق كيفية استخدام الطرق المختلفة لإظهار الحقول بأسمء مستعارة، فنلاحظ هنا أن اسم الحقل ename قد ظهر في النتيجة باسم Name بحروف كبيرة وكذلك حقل الراتب SALARY، كما نلاحظ بأن الاسم المستعار الموجود بين علامتي التنصيص المزدوجة (" ") قد ظهر في النتيجة كما هو دون تحويله إلى الأحرف الكبيرة.

استخدام العمليات الحسابية وأولويات تنفيذها مع جملة SELECT:

من الممكن إجراء عملية حسابية على الحقول العددية للحصول على معلومة معينة فمثلاً إذا أردنا إظهار الموظفين ورواتبهم في السنة فإننا نقوم بضرب راتب كل موظف في العدد ١٢ بشكل التالي (SAL*12)، وكذلك عند إظهار إجمالي الراتب لكل موظف بعد إضافة ٥٠٠ ريال عليه (SAL+500)، لاحظ أن العمليات الحسابية على الحقول لا تؤثر على البيانات المخزنة داخل الجدول.

المعاملات الحسابية التي تستخدم في العمليات الحسابية Arithmetic Operators

١- الجمع (+).
٢- الطرح (-).
٣- الضرب (*).
٤- القسمة (/).

يمكن استخدام المعاملات الحسابية في جميع أجزاء جملة SQL ماعدا الجزء الخاص بFORM والأمثلة التالية توضح كيفية إجراء العمليات الحسابية على الحقول.

مثال (٤): عرض رواتب الموظفين السنوية من جدول الموظفين.

```
SQL> SELECT ename , sal , sal*12  "annual salary"
  2  FROM emp ;
  3
```

ENAME	SAL	annual salary
SMITH	800	9600
ALLEN	1600	19200
WARD	1250	15000
JONES	2975	35700

يبين المثال السابق كيفية استخدام العمليات الحسابية للحصول على رواتب الموظفين السنوية وذلك بضرب راتب كل موظف في ١٢ شهر.

أولويات تنفيذ العمليات الحسابية Operator Procedure

عند إجراء عملية حسابية كبيرة على حقل من الحقول لا بد أن تعرف كيفية حسابها ولمعرفة ذلك لا بد أن تعرف أولوية تنفيذ العوامل داخل جملة SQL فهي تنفذ بالترتيب التالي:

١- أولوية تنفيذ العمليات الحسابية للضرب والقسمة ثم للجمع والطرح.
٢- العمليات من نفس الأولوية تنفذ من اليسار إلى اليمين.
٣- عند وجود الأقواس في العمليات الحسابية يكون ما بداخلها له الأولوية وينفذ حسب الفقرة رقم(١).

لاحظ الفرق بين العمليتين التاليتين:

١- 100*(40+10)=100*50=5000
٢- (100*40)+10=4000+10=4010

والمثال التالي يوضح أولوية التنفيذ للعوامل الحسابية.

مثال (٥) : عرض رواتب الموظفين السنوية من جدول الموظفين.

```
SQL> SELECT  ename , sal , 12*sal+100
  2  FROM emp;
```

لاحظ أولوية التنفيذ :
العملية رقم (١) تنفذ أولاً ثم العملية رقم (٢) .

ENAME	SAL	12*SAL+100
SMITH	800	9700
ALLEN	1600	19300

مثال (٦) : عرض رواتب الموظفين السنوية من جدول الموظفين.

```
SQL> SELECT  ename , sal , 12*(sal+100)
  2  FROM emp;

ENAME          SAL      12*(SAL+100)
---------- ---------- --------------------
SMITH             800             10800
ALLEN            1600             20400
```

لاحظ أولوية التنفيذ :
العملية رقم (١) تنفذ أولاً لوجود الأقواس ثم
العملية رقم (٢) .

بملحوظة الفرق بين المثالين السابقين ، نجد أن النتيجة قد اختلفت تماماً وذلك لوجود الأقواس في مثال رقم ٦ فتم حساب ما بداخل الأقواس أولاً.

استخدام أداة الربط بين الحقول (||) Concatenation

لعمل سلسلة من الحقول نقوم بربط حقلين أو أكثر باستخدام أداة الربط (||) والتي تسمى Concatenation، ويكون ناتج الربط بين الحقول هو حقل واحد فقط، ومن الممكن أن نربط مع الحقول نص معين نضعه بين علامتي تنصيص فردية (' ') ، والمثال التالي يوضح ذلك.

```
SQL> SELECT ename, job , ename||job as "employees"
  2  FROM emp ;

ENAME      JOB          employees
---------- ---------    -------------------
SMITH      CLERK        SMITHCLERK
ALLEN      SALESMAN     ALLENSALESMAN
WARD       SALESMAN     WARDSALESMAN
JONES      MANAGER      JONESMANAGER
MARTIN     SALESMAN     MARTINSALESMAN
```

في المثال السابق تم الربط بين حقلي الإسم والوظيفة بأداة الربط (||) وقد ظهر هذان الحقلان كأنهم حقل واحد باسم مستعار employees يجمع بين الاسم والوظيفة

```
SQL> SELECT ename, job , ename||' is a '||job as "employees"
  2  FROM emp ;

ENAME      JOB          employees
---------- ---------    -------------------
SMITH      CLERK        SMITH is a CLERK
ALLEN      SALESMAN     ALLEN is a SALESMAN
WARD       SALESMAN     WARD is a SALESMAN
JONES      MANAGER      JONES is a MANAGER
MARTIN     SALESMAN     MARTIN is a SALESMAN
```

في المثال السابق تم ربط الحقلين الاسم والوظيفة وبينهما نص هو (is a) باستخدام أداة الربط (| |) ، فظهر الناتج كما هو موضح بالمثال.

استخدام عبارة DISTINCT لمنع تكرار السجلات:

```
SQL> SELECT deptno
  2  FROM emp ;

 DEPTNO
------------
        20
        30
        30
        20
        30
        30
```

لاحظ التكرار في أرقام الإدارات

عند إظهار محتويات الجدول نجد تكرار بعض القيم للحقل الواحد دون فائدة، فمثلاً لو طلب منك معرفة أرقام الإدارات التي ينتمي إليها الموظفين في جدول الموظفين (EMP)، فإنك بالطبع سوف تكتب هذا الآمر

```
SQL> SELECT  DISTINCT deptno
  2  FROM emp ;

  DEPTNO
------------
        10
        20
        30
```

وبالنظر إلى النتيجة سوف تجد أن هناك تكراراً في الأرقام دون فائدة كما هو واضح، ولمنع هذا التكرار نستخدم كلمة (distinct) مباشرة بعد كلمة SELECT كما هو مبين في المثال التالي:

لاحظ أن أرقام الإدارات لن تتغير وذلك لاستخدام كلمة DISTINC.

إظهار البناء الداخلي للجداول باستخدام الأمر DESCRIBE (DESC)

لإظهار معلومات حول أسماء الحقول وأنواعها الموجودة في جدول معين أي لإظهار البناء الداخلي للجدول نستخدم الأمر (DESCRIBE) والذي يمكن اختصاره إلى الأحرف التالية (DESC)

```
SQL> DESC emp ;

Name                                      Null?    Type
----------------------------------------- -------- ------------
EMPNO                                     NOT NULL NUMBER(4)
ENAME                                              VARCHAR2(10)
JOB                                                VARCHAR2(9)
MGR                                                NUMBER(4)
HIREDATE                                           DATE
SAL                                                NUMBER(7,2)
COMM                                               NUMBER(7,2)
DEPTNO                                             NUMBER(2)
```

في المثال السابق تم إظهار أسماء الحقول الموجودة في جدول الموظفين وأنواعها وبعض المعلومات الخاصة بكل حقل مثل: هل نوع الحقل نصي أو تاريخ أو رقم؟ وما طوله؟

التعامل مع القيمة NULL

ماذا تعني القيمة NULL، هذه القيمة تسمى قيمة غير معروفة أو قيمة خاوية بمعنى أنها لا تساوي الصفر ولا مسافة ولا أي رقم أو نص، فمثلاً هناك حقل في جدول الموظفين اسمه COMM هذا الحقل يخزن به قيمة تكليف الموظف بمهمة أو تكليفه بعمل إضافي يأخذ عليه أجر، فهناك موظفون يأخذون بدل تكليف وموظفون آخرون لا يمكن تكليفهم أي لا يأخذون بدل تكليف إطلاقاً وفي هذه الحالة يتم ترك حقل التكليف COMM خالياً ليس به أي قيمة ونطلق على القيمة الخالية هذه NULL.

المثال التالي يبين الموظفين الذين لا يكلفون بعمل إضافي أي لا يأخذون بدل تكليف

```
SQL> SELECT ename, job , sal , comm
  2  FROM emp ;

ENAME      JOB              SAL      COMM
---------- ---------- ---------- ----------
SMITH      CLERK            800
ALLEN      SALESMAN        1600        300
WARD       SALESMAN        1250        500
JONES      MANAGER         2975
MARTIN     SALESMAN        1250       1400
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
SCOTT      ANALYST         3000
KING       PRESIDENT       5000
TURNER     SALESMAN        1500          0
```

NULL

الموظف SMITH لا يأخذ بدل تكليف ولذلك تركت قيمة COMM خالية.

عند إجراء أي عملية حسابية عليها يكون النتائج دائماً NULL قيمة خالية، يمكنك أن تلاحظ ذلك في المثال التالي :

```
SQL> SELECT ename, job , sal , 12*sal+comm
  2  FROM  emp ;

ENAME      JOB              SAL      12*SAL+COMM
---------- ---------- ---------- ----------
SMITH      CLERK            800
ALLEN      SALESMAN        1600      19500
```

NULL

جملة الشرط (WHERE):

تُكتب هذه الجملة مباشرةً بعد جملة (FROM) وتستخدم في حصر البيانات على أساس شرط أو شروط معينة، ويتكون الشرط من طرفين بينهم معامل مقارنة Comparison Operator، وعند تحقق الشرط أي أن الشرط (TRUE) فإن جملة SELECT يكون لها ناتج، أما إذا كان ناتج الشرط غير متحقق (FALSE) فإن جملة SELECT لا يكون لها ناتج وتظهر رسالة No Row Selected ومعناها لم يتم تحديد أي صف.

مكونات جملة الشرط WHERE:

- أسماء الحقول Columns.
- معاملات مقارنةcomparison Operator
- قيم ثابتة سواء كانت عددية أو نصية.
- تعبيرات حسابية.

متطلبات وإرشادات كتابة جملة الشرط WHERE

يجب مراعاة الآتي عند كتابة جملة الشرط:

- عند استخدام قيم نصيّة أو قيم تُعبر عن تاريخ لا بد من وضعها داخل علامة التنصيص الفردية (' ') .
- في حالة استخدام القيم النصية لا بد من مراعاة حالة الأحرف كبيرة أم صغيرة.
- في حالة استخدام قيم تعبر عن تاريخ لا بد من مراعاة صيغة التاريخ المستخدمة (FORMAT) علماً بأن الصيغ الأساسية للتاريخ داخل لغة SQL هي كالتالي:
(DD-MON-YY) حيث أن (DD تعبر عن اليوم، و MON تعبر عن الشهر ، YY تعبر عن السنة)

جملة الترتيب (ORDERED BY):

تستخدم هذه الجملة لترتيب الصفوف الناتجة ترتيباً تصاعدياً أو تنازلياً ، وتكتب دائماً في نهاية جملة SELECT.

متطلبات وإرشادات كتابة جملة الترتيب ORDERED BY

يجب مراعاة الآتي عند كتابة جملة الترتيب

- يجب أن تُكتب في آخر جملة SELECT.
- تحتوي على أسماء الحقول Columns أو أسماء مستعارة Allies.
- للترتيب تصاعدياً اكتب (ASC) وهي اختصار لكلمة (Ascending) وهي القيمة الافتراضية للترتيب (Default).
- للترتيب تنازلياً لا بد من كتابة كلمة (DESC) وهي اختصار لكلمة Descending.

مثال (١) : عرض أسماء ووظائف وأرقام إدارات الموظفين الذين يعملون بوظيفة (CLERK)، مع ترتيب الناتج تصاعدياً حسب رقم الإدارة

```
SQL> SELECT ename , job , deptno
  2 FROM    emp
  3 WHERE   job = 'CLERK'
  4 ORDER BY  deptno
```

لاحظ
كتابة النص بين علامتي ' ' وبالأحرف الكبيرة .

ENAME	JOB	DEPTNO
MILLER	CLERK	10
SMITH	CLERK	20
ADAMS	CLERK	20
JAMES	CLERK	30

المثال رقم (!) يبين طريقة استخدام جملة الشرط WHERE، وذلك لعرض بيانات الموظفين الذين يعملون بوظيفة CLERK فقط، لاحظ أن جملة الشرط قد احتوت على طرفين بينهما معامل حسابي وهو (=) ، وكانت النتيجة كما هي واضحة في المثال وذلك لأن جملة الشرط قد تحققت وكان هناك موظفون يعملون بوظيفة CLERK، لاحظ أيضاً أن الطرف الثاني من جملة الشرط ('CLERK') قد استخدمنا فيه ثابتاً نصياً، ووضعنا هذا الثابت بين علامتي التنصيص الفردية، وأيضاً كتبنا الثابت النصي بالأحرف الكبيرة، وذلك لأن البيانات داخل جدول الموظفين مسجلة بالأحرف الكبيرة، ولذلك لا بد من مراعاة حالة أحرف البيانات داخل الجدول كبيرة أم صغيرة حسب ما هو موجود في الجدول. كما تم ترتيب الناتج تصاعدياً بواسطة ORDERED BY.

معاملات المقارنة المستخدمة في جملة الشرط WHERE Comparison Operator

تستخدم معاملات المقارنة التالية للمقارنة بين طرفي الشرط في جملة Where

المعامل	المعـــنى
=	يساوي
>	أكبر من
>=	أكبر من أو يساوي
<	أصغر من
<=	أصغر من أو يساوي
<> أو =!	لا يساوي

الصيغة العامة لجملة الشرط WHERE

SQL > **WHERE** تعبير OPERATOR قيمة

أمثلة مختلفة:

بشرط أن تاريخ التعيين يساوي (١ يناير ٩٥) WHERE hiredate = '01-JAN-95'

بشرط أن الراتب أكبر من أو يساوي ١٥٠٠ دولار WHERE sal >= 1500

بشرط أن اسم الموظف يكون SMITH WHERE ename = 'SMITH'

مثال (٢) عرض أسماء ووظائف ورواتب الموظفين الذين رواتبهم أكبر من أو تساوي ٣٠٠٠.

```
SQL> SELECT   ename , job , sal
  2  FROM     emp
  3  WHERE    sal >= 3000 ;
```

ENAME	JOB	SAL
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
FORD	ANALYST	3000

الناتج :
أكبر من أو يساوي (٣٠٠٠)

مثال (٣) عرض أسماء ورواتب وعمولة الموظفين الذين رواتبهم أقل من أو تساوي العمولة الخاصة بهم.

```
SQL> SELECT  ename , sal , comm
  2  FROM     emp
                3  WHERE     sal<= comm ;

ENAME        SAL       COMM
----------  ----------  ----------
MARTIN       1250        1400
```

في المثال السابق تم استخدام اسم العمود (comm) في طرف الجملة الشرط الأيمن، أي أننا من الممكن أن نستخدم أسماء الأعمدة للمقارنة مع أسماء أعمدة أخرى. وكانت النتيجة هي عرض بيانات الموظفين الذين رواتبهم أقل من أو تساوي العمولة الخاصة بهم.

معاملات مقارنة أخرى تستخدم في جملة الشرط WHERE

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

المعامل	المعـــنى
قيمة BETWEEN قيمة AND	حصر البيانات بين رقمين
IN (مجموعة من القيم)	حصر البيانات ضمن مجموعة من القيم
LIKE {% , _}	حصر البيانات حسب مطابقة النص أو الحروف
IS NULL	حصر البيانات الخالية NULL

مثال (٤) : عرض أسماء الموظفين ورواتبهم الذين تنحصر رواتبهم بين ١٥٠٠ و ٢٥٠٠.

```
SQL> SELECT  ename , sal
                2  FROM      emp
   3  WHERE    sal   BETWEEN 1500  AND  2500  ;
```

القيمة القيمة

ENAME	SAL
ALLEN	1600
CLARK	2450
TURNER	1500

لاحظ :
الراتب محصور بين ١٥٠٠ و ٢٥٠٠

في المثال السابق تم عرض بيانات الموظفين الذين تنحصر رواتبهم بين ١٥٠٠ و ٢٥٠٠ لاحظ أن القيمة ١٥٠٠ قد ظهرت في النتيجة لأن ناتج جملة BETWEEN يشتمل على القيمة الصغرى والقيمة الكبرى، ونؤكد هنا على أن القيمة الصغرى ينبغي أن تكون أقل من القيمة الكبرى، فإذا عكسنا القيمة الكبرى مكان الصغرى (BETWEEN 2500 AND 1500) فإن جملة الشرط لن تحقق أي تكون قيمتها (FALSE) وتظهر رسالة (No Row Selected) ومعناها لم يتم تحديد أي صف.

مثال (٥) : عرض رقم واسم وراتب ورقم المدير للموظفين الذين لديهم مديرين بالأرقام التالية (٧٩٠٢، ٧٥٦٦، ٧٧٨٨).

في المثال السابق تم استرجاع بيانات الموظفين الذين لديهم مديرين من بين الأرقام التالية: (٧٩٠٢، ٧٥٦٦، ٧٧٨٨). أما

```
SQL> SELECT  empno , ename , sal , mgr
  2  FROM       emp
  3  WHERE      mgr  IN  (7902,7566,7788,7839)  ;
```

EMPNO	ENAME	SAL	MGR
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788
7902	FORD	3000	7566

المدير رقم (٧٨٣٩) فليس لديه موظفين لذلك لا يوجد له نتيجة. وبهذا فإن جملة IN تقوم بالبحث عن الرقم من بين قائمة الأرقام الموجودة داخل الأقواس.

معامل LIKE{%,_}

يستخدم هذا المعامل للبحث عن نص معين داخل ثابت أو حقل نصي، حيث يتم مطابقة حروف النص المذكورة في جملة الشرط.

(%) هذا الرمز يعني أي حرف أو أحرف ، مثلاً التعبير ('A%') يعني مطابقة النصوص التي تبدأ بحرف A مهما كانت باقي الحروف التالية له. فهو يستخدم للبحث عن نص يبدأ بالحرف A.

والتعبير ('A%') فهو يستخدم للبحث عن النصوص التي تحتوي على الحرف A.

(_) هذا الرمز يعني مطابقة حرف واحد فقط، فمثلاً التعبير ('A%_') يعني أنه بغض النظر عن الحرف الأول، ويستخدم هذا التعبير عندما نريد البحث عن نص يكون الحرف الثاني فيه هو A. أما التعبير ('__A')، فيستخدم للبحث عن نص يكون الحرف الثالث فيه هو A.

ملحوظة :

يجب مراعاة حالة الأحرف هل هي كبيرة أم صغيرة عند استخدام المعامل LIKE.

مثال (٦) : عرض أسماء الموظفين الذين تبدأ أسمائهم بالحرف S.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename  LIKE  'S%'   ;

ENAME
----------
SMITH
SCOTT
```

مثال (٧) : عرض اسم وتاريخ تعيين الموظفين الذين تم تعيينهم في العام ١٩٨١ م

```
SQL> SELECT   ename , hiredate
 2  FROM      emp
 3  WHERE     hiredate LIKE '%81' ;

ENAME      HIREDATE
---------- ----------
ALLEN      20/02/81
WARD       22/02/81
JONES      02/04/81
MARTIN     28/09/81
BLAKE      01/05/81
CLARK      09/06/81
KING       17/11/81
TURNER     08/09/81
JAMES      03/12/81
FORD       03/12/81
```

مثال (٨) : عرض أسماء الموظفين الذين يكون الحرف الثاني في أسمائهم هو A

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename  LIKE  '_A%'  ;

ENAME
----------                    ( A ) الحرف الثاني
WARD
MARTIN
JAMES
```

في المثال السابق تم البحث عن الأسماء التي يكون الحرف الثاني فيها هو (A) ثم عرض هذه الأسماء.

مثال (٩) : عرض اسم ورقم المدير للموظفين الذين لا يوجد لديهم مدير.

```
SQL> SELECT  ename , mgr
  2  FROM    emp
  3  WHERE   mgr  IS NULL  ;

ENAME        MGR
----------   ----------
KING                    <-------  NULL
```

في المثال رقم (٩) تم استخدام المعامل IS NULL والذي يقوم بحصر البيانات الخالية، ففي المثال تم عرض بيانات الموظفين الذين لا يوجد لديهم مدير أي أن حقل المدير (MGR) لهذا الموظف خالي ليس به بيانات.

ملحوظة:

لا يمكن استخدام المعامل (=) مع القيم الخالية NULL ولكن لا بد من استخدام المعامل IS NULL، يمكن أن تجرب الأمر في المثال السابق بالشكل التالي لتعرف الفرق

```
SQL> SELECT  ename , mgr          <-------  الأمر هنا خطأ لاستخدام المعامل ( = )
  2  FROM    emp
  3  WHERE   mgr = NULL ;
```

المعاملات المنطقية في جملة الشرط WHERE

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

تستخدم المعاملات المنطقية التالية لتكوين أكثر من شرط في جملة WHERE، وهي معاملات تربط بين جملتين شرطيتين أو أكثر، وتكون النتيجة إما (TRUE) أ] تحقق الشرط أو (FALSE) لم يتحقق الشرط.

المعامل AND:

هذا المعامل يربط بين جملتين شرطيتين ويكون الناتج TRUE إذا كانت كلتا الجملتين TRUE والجدول التالي يوضح ناتج المعامل AND مع الحالات المختلفة لجملتي الشرط.

جملة الشرط الأولى	جملة الشرط الثانية	ناتج المعاملAND
True	True	True
True	False	False
False	False	False
True	Null	Null
False	Null	False
Null	Null	Null

بالتدقيق في الجدول السابق يمكن أن نستخلص النتائج التالية:

أولاً: ناتج المعامل AND يكون دائماً FALSE إلا في حالة أن الجملتين TRUE فقط.

ثانياً: عند استخدام القيمة NULL مع المعامل AND يكون الناتج دائماً NULL إلا في حالة أن إحدى الجملتين تكون NULL والأخرى تكون FALSE فقط.

مثال (١٠) : عرض رقم واسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من أو تساوي ١١٠٠ وفي نفس الوقت وظيفتهم CLERK.

```
SQL> SELECT   empno , ename , job , sal
  2  FROM     emp
  3  WHERE    sal >=1100  AND  job='CLERK' ;
                  T                T
                جملة             جملة
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

المثال السابق يبين أن جملتي الشرط sal >= 1100 و job = 'CLERK' قد تحققت أي أن نتيجة كلاً منهما كانت (TRUE) ولذلك فإن ناتج المعامل AND هو TRUE ولهذا قد تم عرض البيانات كما هو واضح من المثال.

مثال (١١): عرض اسم وراتب وعمولة الموظفين الذين يزيد راتبهم عن ١١٠٠ وفي نفس الوقت تقل عمولتهم عن ٥٠٠.

```
SQL> SELECT ename , sal , comm
  2    FROM   emp
  3   WHERE  sal>1100  AND comm<500  ;
```

ENAME	SAL		COMM
ALLEN	1600		300
TURNER	1500		0

المعامل OR:

هذا المعامل يربط بين جملتين شرطيتين ويكون الناتج TRUE إذا كانت إحدى الجملتين أو كلاهما TRUE والجدول التالي يوضح ناتج المعامل OR مع الحالات المختلفة لجملتي الشرط.

جملة الشرط الأولى	جملة الشرط الثانية	ناتج المعاملAND
True	True	True
True	False	True
False	False	False
True	Null	True
False	Null	Null
Null	Null	Null

بالتدقيق في الجدول السابق يمكن أن نستخلص النتائج التالية:

أولاً: ناتج المعامل OR يكون دائماً TRUE إلا في حالة أن الجملتين FALSE فقط.

ثانياً: عند استخدام القيمة NULL مع المعامل OR يكون الناتج دائماً NULL إلا في حالة أن إحدى الجملتين تكون NULL والأخرى تكون TRUE فقط فيكون الناتج TRUE.

مثال (١٢): عرض رقم واسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من (2500) أو تكون وظيفتهم MANAGER

```
SQL> SELECT   empno , ename , job , sal
  2  FROM     emp
  3  WHERE    sal >2500  OR   job='MANAGER'  ;
```

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

المثال السابق يبين أنه لا بد من تحقق أي من الجملتين حتى يتم استرجاع بيانات، وبالتدقيق في الموظف رقم (٧٧٨٢) نجد أنه بالرغم من أن راتبه يقل عن ٢٥٠٠ إلا أنه ظهر في النتيجة وذلك لتحقيق شرط الوظيفة (MANAGER) ، أي أنه يجب أن يتحقق أحد الشرطين لاسترجاع البيانات.

مثال (١٣): عرض اسم وراتب ورقم الإدارة للموظفين الذين رواتبهم أقل من (١٠٠٠) أو تكون إداراتهم رقم (١٠).

في المثال السابق تم عرض بيانات الموظفين الذين تقل رواتبهم عن ١٠٠٠ أو الموظفين المسجلين في الإدارة رقم (١٠).

```
SQL> SELECT   ename , sal , deptno
  2  FROM     emp
  3  WHERE    sal<1000  OR  deptno=10  ;
```

ENAME	SAL	DEPTNO
SMITH	800	20
CLARK	2450	10
KING	5000	10
JAMES	950	30
MILLER	1300	10

نلاحظ من المثال أن الموظفين المسجلين في الإدارة رقم ٢٠ يأخذون راتباً أقل من ١٠٠٠. والموظفين الذين يأخذون راتباً أكبر من ١٠٠٠ هم مسجلين في الإدارة رقم (١٠) وهذا يؤكد أنه لاسترجاع بيانات لابد من تحقق إحدى جملتي الشرط على الأقل.

المعامل NOT:

هذا المعامل يقوم بعكس ناتج جملة الشرط، أي أنه إذا كانت جملة الشرط (TRUE) فإن ناتج المعامل NOT يكون (FALSE) والعكس ، والجدول التالي يبين تأثير هذا المعامل على جملة الشرط.

جملة الشرط	ناتج المعاملNOT
TRUE	False
False	TRUE
Null	Null

يستخدم المعامل NOT أيضاً لنفي المعاملات الموضح بالجدول التالي:

المعامل	نفي المعامل	المعنى
BETWEEN ... AND	NOT BETWEEN ... AND	ليست بين رقمي ... و ....
IN (.....)	NOT IN (.....)	ليست ضمن القائمة ....
LIKE {% , _}	NOT LIKE {% , _}	ليست مطابقة
IS NULL	IS NOT NULL	ليست قيمة خيالية

أمثلة على استخدام المعامل NOT لعكس المعاملات سابقة الذكر.

WHERE job NOT IN ('CLERK' , 'MANAGER')

• WHERE sal NOT BETWEEN 1000 AND 1500

• WHREE ename NOT LIKE '%A%'

• WHERE comm IS NOT NULL

مثال (١٤) : عرض اسم ووظيفة الموظفين الذين ليست وظائفهم من ضمن الوظائف التالية :

(CLERK, MANAGER, ANALYST)

```
SQL> SELECT   ename , job
  2  FROM     emp
  3  WHERE    job  NOT  IN  ( 'CLERK' , 'MANAGER' , 'ANALYST' ) ;
```

ENAME	JOB
----------	--------
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
KING	PRESIDENT
TURNER	SALESMAN

في هذا المثال تم نفي المعامل IN (...) ولذلك تم عرض بيانات الموظفين الذين لهم وظائف ليست من ضمن قائمة الوظائف التالية (CLERK, MANAGER, ANALYST)

مثال (١٥) عرض اسم ووظيفة وراتب الموظفين الذين لا تنحصر رواتبهم بين ١٠٠٠ و ٣٠٠٠

```
SQL> SELECT   ename , job , sal
  2  FROM     emp
  3  WHERE    sal NOT BETWEEN 1000 AND 3000 ;
```

ENAME	JOB	SAL
SMITH	CLERK	800
KING	PRESIDENT	5000
JAMES	CLERK	950

في المثال السابق تم استبعاد الموظفين الذين تنحصر رواتبهم بين ١٠٠٠ و ٣٠٠٠ وتم عرض باقي الموظفين، وبذلك قد تم نفي المعامل (BETWEEN .... AND).

مثال (١٦) عرض اسم ووظيفة وراتب وعمولة الموظفين الذين يأخذون عمولة

```
SQL> SELECT  ename , job , sal , comm
  2  FROM    emp
  3  WHERE  comm IS  NOT  NULL  ;
```

ENAME	JOB	SAL	COMM
ALLEN	SALESMAN	1600	300
WARD	SALESMAN	1250	500
MARTIN	SALESMAN	1250	1400
TURNER	SALESMAN	1500	0

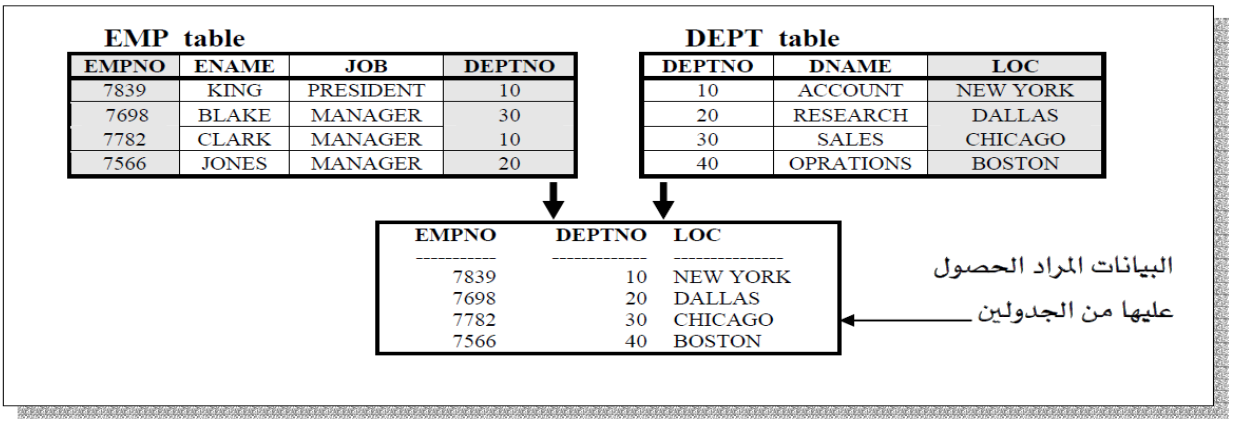
في المثال السابق تم عرض بيانات الموظفين الذين يأخذون عمولة ، أي الذين ليست عمولتهم خالية NULL.

وإذا أردنا ترتيب الناتج تنازلياً حسب الراتب نضيف السطر التالي على المثال السابق:

ORDERED BY sal DESC

استرجاع البيانات من أكثر من جدول

في بعض الأحيان نريد أن نقوم بعرض بيانات من أكثر من جدول لعمل تقارير مفيدة وشاملة، فمثلاً لو أردنا عرض رقم الموظف ورقم الإدارة التابع لها وموقع هذه الإدارة نجد أننا لا يد من الحصول على هذه البيانات من جدول الموظفين وجدول الإدارات لكون رقم الموظف موجود في جدول الموظفين ورقم الإدارة موجود في جدول الإدارات وأيضاً موجود في جدول الموظفين بينما موقع الإدارة موجود في جدول الإدارات، كما هو موضح بالشكل التالي:



وللحصول على تلك البيانات لا بد من عمل ربط بين الجدولين.

تعريف الربط : Join definition

هو عبارة عن ربط بين جدولين أو أكثر للحصول على بيانات من تلك الجداول.

أنواع الربط :Types of Joins

توجد عدة أنواع من الربط (Joins) وهي كالتالي:

- الربط بالتساوي Equijoin
- الربط بعدم التساوي Non-Equijoin.
- الربط الخارجي Outer Join
- الربط الداخلي في نفس الجدول Self Join.

ويتم عمل هذه الأنواع عن طريق جملة الاستفسار SELECT وبخاصة في جزء الشرط WHERE

الربط بالتساوي : Equijoin

في هذا النوع من الربط يتم ربط جدولين أو أكثر عن طريق عمودين متساويين، العمود الأول عادة ما يكون مفتاح أساس (Primary Key) في الجدول الأول والعمود الثاني يكون عبارة عن عمود ربط (Foreign Key) في الجدول الثاني.

والشكل التالي يبين الربط بالتساوي بين جدول الموظفين وجدول الإدارات عن طريق العمود (deptno) الموجود في كل منهما.

EMP table

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

Foreign Key

DEPT table

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

Primary Key

سوف يتم عمل الربط بين الجدولين باستخدام جملة SELECT عن طريق العمودين المشار إليهما بالسهم كما في المثال التالي: مثال (١):

```
SQL> SELECT emp.empno , emp.ename , emp.deptno ,
  2         dept.deptno , dept.loc
  3  FROM emp , dept
  4  WHERE emp.deptno=dept.deptno ;
```

شرط الربط بين الجدولين

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7369	SMITH	20	20	DALLAS
7499	ALLEN	30	30	CHICAGO
7521	WARD	30	30	CHICAGO
7566	JONES	20	20	DALLAS
7654	MARTIN	30	30	CHICAGO
7698	BLAKE	30	30	CHICAGO
7782	CLARK	10	10	NEW YORK
7788	SCOTT	20	20	DALLAS
7839	KING	10	10	NEW YORK
7844	TURNER	30	30	CHICAGO
7876	ADAMS	20	20	DALLAS

في المثال السابق تم عرض بيانات من جدولين عن طريق الربط باتساوي وسوف نقوم بشرح كل جزء من أجزاء جملة SELECT على حده كالتالي:

- في قائمة SELECT تم عرض رقم الموظفين وأسمائهم وأرقام إداراتهم من جدول الموظفين (EMP)، كما تم عرض رقم الإدارات وموقعها من جدول الإدارات (DEPT)، ونلاحظ هنا أننا حددنا من أين تأتي البيانات عن طريق كتابة اسم الجدول قبل اسم العمود ويفصل بينهما العلامة (.) كالتالي (emp.empno).
- في الجزء FORM تم كتابة أسماء الجداول التي ستأتي منها البيانات كالتالي (FROM emp,dept).

في الجزء WHERE تم كتابة شرط الربط بين الجدولين وهذا الشرط مهم جداً لإتمام عملية الربط، وبدون هذا الشرط سوف تكون النتيجة ليس لها معنى أو فائدة. استخدام الأسماء المستعارة للجداول:

يمكن استخدام الأسماء المستعارة للجداول لتسهيل عملية كتابة الأعمدة، فمثلاً نقوم باستبدال اسم الجدول (emp) بالحرف (e) ، واسم الجدول (dept) بالحرف (d) كالتالي:

```
SQL>SELECT e.empno,e.ename,e.deptno,
  2          d.deptno   , d.Loc
  3   FROM emp e , dept d
  4   WHERE e.deptno = d.deptno ;
```

عندما نريد عرض البيانات الموجودة في المثال رقم (١) ولكن للموظف KING فقط هنا لا بد من زيادة شرط على جملة SELECT مثال (٢):

```
SQL> SELECT e.empno , e.ename , e.deptno ,
  2         d.deptno , d.loc
  3  FROM emp  e , dept  d <-------- (e , d) الاسماء المستعارة
  4  WHERE e.deptno=d.deptno
  5  AND e.ename = upper('king') ;
```

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7839	KING	10	10	NEW YORK

```
SQL> SELECT e.ename , e.sal , s.grade
  2  FROM emp e , salgrade s
  3  WHERE  e.sal BETWEEN  s.losal  AND  s.hisal ;
```

شرط الربط بين الجدولين

ENAME	SAL	GRADE
SMITH	800	1
ADAMS	1100	1
JAMES	950	1
WARD	1250	2
MARTIN	1250	2
MILLER	1300	2
ALLEN	1600	3
TURNER	1500	3
JONES	2975	4
BLAKE	2850	4
CLARK	2450	4

معالجة البيانات (إضافة - تعديل – حذف)

وتتكون لغة التعامل مع البيانات (DML) من عدة جمل وهي كالتالي:

- جملة إضافة بيانات إلى الجدول INSERT INTO.
- جملة التعديل في بيانات الجدول UPDATE.
- جملة حذف بيانات من الجدول DELETE FROM.

إضافة سجل أو سجلات جديدة إلى جدول هي عملية إضافة بيانات جديدة إلى جدول معين عن طريق استخدام جملة الإضافة INSERT INTO.

الصيغة العامة لإضافة سجلات جديدة إلى جدول:

```
SQL > INSERT INTO جدول ( عمود١ , عمود٢ , ..... عمود٣ )
      VALUES        ( قيمة١ , قيمة٢ , .....قيمة٣ ) ;
```

شرح الشكل العام:

- جدول : اسم الجدول المطلوب إضافة سجلات فيه.
- (عمود١ ، عمود ٢، عمود ٣، .....) : أسماء الأعمدة المطلوب إدخال البيانات إليها.
- (قيمة ١، قيمة ٢، قيمة ٣، ....) : القيم المطلوب إضافتها إلى الأعمدة.

القواعد التي يجب التقيد بها عند الإضافة:

- يجب أن يكون عدد القيم التي سيتم إدخالها مساوياً لعدد الأعمدة المذكورة في جملة INSERT.
- يجب أن تكون القيم مرتبة بنفس ترتيب الأعمدة المراد إدخال القيم إليها، حيث إن القيمة ١ سوف تسجل في العمود ١ وهكذا. كما يجب أن تكون القيم أيضاً من نفس نوع بيانات الأعمدة.
- عند إدخال قيم التاريخ والنصوص لا بد من وضعها داخل علامتي تنصيص فرديتين ' '.
- يجب إدخال قيماً للأعمدة التي لا تقبل قيماً فارغة NULL مثل أعمدة المفتاح الأساسي (Primary Key) مثلاً (empno) في جدول الموظفين.

يجوز عدم ذكر أسماء الأعمدة في جملة INSERT وفي هذه الحالة لا بد من إدخال جميع قيم الأعمدة الموجودة في الجدول حسب ترتيب الأعمدة داخل الجدول مع مراعاة نوع البيانات لكل عمود.

مثال(١): إضافة سجل جديد إلى جدول الإدارات أي إضافة إدارة جديدة.

```
SQL> INSERT  INTO  dept (deptno , dname , loc )
  2   VALUES            ( 50 , 'DEVELOPMENT' , 'DETROIT')  ;
```

جدول الإدارات (DEPT) قبل الإضافة

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

جدول الإدارات (DEPT) بعد الإضافة

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

السجل (الصف) الذي تم إضافته

في المثال السابق تم إضافة إدارة جديدة برقم (٥٠) واسمها (DEVELOPMENT) وموقعها (DETROIT) ، ونلاحظ في هذا المثال أننا أخذنا في الاعتبار القيود التي يجب اتباعها عند الإضافة كما هو مذكور سابقاً، أي أننا أضفنا رقم الإدارة داخل العمود deptno الذي نوع بياناته number، وأضفنا اسم الإدارة وموقعها ووضعناهما داخل علامتي التنصيص الفردية (' ') وذلك لأن نوع بياناتهما حروف (Varchar2).

إضافة قيمة فارغة (NULL) إلى العمود:

يتم إضافة القيمة NULL إلى الأعمدة بطريقتين:

الأولى: عدم كتابة الأعمدة المراد تسجيل القيمة NULL بها في الجزء INSERT.

الثانية: أن تكتب الأعمدة ولكن تكتب قيمتها NULL داخل الجزء VALUES.

بشرط أن تقبل الأعمدة هذه القيمة أي إنها ليست عليها قيود مثل (primary key) كما في المثال التالي:

مثال (٢): إضافة سجل جديد إلى جدول الإدارات يحتوي هذا السجل على رقم الإدارة واسمها فقط.

```
SQL> INSERT  INTO  dept (deptno , dname )
  2   VALUES              ( 60 , 'MIS' ) ;
```

جدول الإدارات (DEPT) بعد الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	

قيمة فارغة NULL

في المثال السابق تم إضافة إدارة جديدة برقم (٦٠٩ واسمها (MIS) ، ولكن لم يتم إضافة موقعها ولذلك تم تسجيل القيم NULL داخل العمود (LOC) ومن الممكن أن نكتب هذا المثال كالتالي:

```
SQL> INSERT INTO dept (deptno , dname , loc )
٢       VALUES              ( 60 , 'MIS' , NULL );
```

إضافة قيم خاصة داخل الأعمدة:

إذا أردنا إضافة تاريخ اليوم الحالي (SYSDATE) إلى العمود (HIREDATE) الموجود داخل جدول الموظفين وذلك عند إضافة سجل جديد أي إضافة بيانات موظف جديد فيمكن عمل ذلك كما هو موضح من المثال التالي:

مثال(٣): إضافة بيانات موظف جديد إلى جدول الموظفين:

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

في المثال السابق تم إضافة بيانات موظف جديد إلى جدول الموظفين وتم وضع تاريخ التعيين له بحيث يكون تاريخ اليوم الحالي SYSDATE.

المثال السابق يمكن كتابته بحيث لا نذكر أسماء الأعمدة في الجزء INSERT، وهنا لا بد من كتابة كل القيم حسب ترتيب الأعمدة داخل الجدول كالآتي:

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

التعديل في بيانات سجل أو سجلات معينة داخل جدول (UPDATE):

التعديل في سجل أو سجلات معينة داخل جدول هي عملية تعديل بيانات عمود أو عدة أعمدة عن طريق استخدام جملة التعديل (UPDATE).

الصيغة العامة لتعديل البيانات داخل الجدول

```
SQL > UPDATE جدول
      SET   عمود١ = قيمة١ , عمود٢ = قيمة٢  .....
      WHERE   شرط   ;
```

شرح الشكل العام:

- جدول : اسم الجدول المطلوب تعديل السجلات فيه.
- عمود ١، عمود٢: أسماء الأعمدة المطلوب التعديل فيها.
- قيمة ١، قيمة ٢: القيم الجديدة المراد وضعها بدلاً من القيم القديمة.
- شرط: شرط لاختيار سجلات(صفوف) معينة للتعديل فيها، وبدون هذا الشرط فسوف يتم التعديل في جميع السجلات.

القواعد التي يجب التقيد بها عند التعديل:

- يجب أن يكون نوع البيانات الجديدة من نفس نوع بيانات الأعمدة المطلوب التعديل فيها.
- عند تعديل قيم التاريخ أو النصوص يجب وضع القيم الجديدة بين علامتي التنصيص الفردية (‘ ’).
- يجب أخذ الحذر عند كتابة الجزء WHERE في جملة التعديل لتحديد أي الصفوف التي سوف يتم التعديل فيها.

مثال تعديل اسم الإدارة رقم (٣٠) ليصبح (EDUCATION) بدلاً من (SALES)

```
SQL> UPDATE   dept
  2   SET dname='EDUCATION' ◄——— القيمة الجديدة لاسم الإدارة رقم (30)
  3   WHERE deptno=30 ;

1 row updated ◄———  رسالة تدل على أن التعديل تم في الإدارة رقم (30) فقط
```

في المثال السابق تم تعديل اسم الإدارة رقم (٣٠) من (SALES)إلى(EDUCATION) باستخدام أمر التعديل (UPDATE dept) في جدول الإدارات، ولاحظ اختيار رقم الإدارة ٣٠ باستخدام الجزء WHERE، ماذا يحدث لو لم نحدد الإدارة رقم (٣٠) ؟ أي ماذا يحدث لو ألغينا جملة الشرط WHERE من المثال السابق؟

عند عدم كتابة الشرط في عملية التعديل أي لم يتم تحديد الصف المراد التعديل فيه فإنه يتم التعديل في جميع الصفوف كما في المثال التالي:

مثالتعديل اسم الإدارة رقم (٣٠) ليصبح (EDUCATION) بدلاً من (SALES)

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

4 row updated ◄———  رسالة تدل على أن التعديل تم في جميع الصفوف
```

في المثال السابق تم تعديل اسم جميع الإدارات إلى (EDUCATION)، ولذلك لا بد أن من الحذر عند التعديل في صف معين فلا بد من تحديد الصف المراد التعديل فيه باستخدام الجزء (WHERE).

التعديل في أكثر من عمود:

في الأمثلة السابقة تم التعديل في عمود واحد فقط وهو عمود اسم الإدارات، فإذا أردنا التعديل في أكثر من عمود، فمثلاً لتعديل رقم الإدارة والوظيفة والموظف (BLAKE) ليصبح مثل الموظف (WARD)، فماذا نفعل؟ انظر المثال التالي:

مثال

```
SQL> UPDATE   emp
  2   SET (job , deptno) = (select  job , deptno  from emp  where ename='WARD')
  3   WHERE ename='BLAKE' ;
                                        (MANAGER    30)  نتيجة الاستعلام الفرعي
1 row updated
```

في المثال السابق تم عمل استعلام فرعي متعدد الأعمدة لإيجاد وظيفة ورقم الإدارة للموظف (WARD)، ومن ثم استخدامهما في الجزء SET لتعديل وظيفة ورقم الإدارة للموظف (BLAKE) وبهذا فقد تم التعديل في العمودين (job, deptnoللموظف (BLAKE).

حذف سجل أو سجلات معينة داخل جدول (DELETE FROM):

حذف سجل أو سجلات معينة داخل جدول هي عملية إلغاء بيانات عمود أو عدة أعمدة عن طريق استخدام جملة الحذف (DELETE FROM).

الصيغة العامة لحذف البيانات من الجدول

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

شرح الشكل العام:

- جدول: اسم الجدول المطلوب حذف سجل أو سجلات منه.
- شرط: شرط لاختيار سجلات (صفوف) معينة لحذفها، وبدون هذا الشرط فسوف يتم حذف جميع السجلات (الصفوف).

القواعد التي يجب التقيد بها عند الحذف:

- يجب الحذر عند كتابة الجزء WHERE في جملة الحذف لتحديد أي الصفوف التي سوف يتم حذفها، عندما لا تكتب جزء الشرط في جملة الحذف فإنه يتم حذف جميع صفوف الجدول كاملة.

مثال حذف الإدارة رقم (٤٠) من جدول الإدارات

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

1 row deleted.
```

في المثال السابق تم حذف الإدارة رقم (٤٠) من جدول الإدارات، أي إنه تم حذف صف كامل من الجدول.

مثال (١١): حذف جميع الموظفين من جدول الموظفين

```
SQL> DELETE FROM emp  ;

14 row deleted.
```

في المثال السابق تم حذف جميع الصفوف من جدول الموظفين وذلك لعدم تحديد الصف أو الصفوف المراد حذفها أي إننا لم نكتب جملة الشرط WHERE في المثال السابق.

(تنبيه: الرجاء عدم تنفيذ الأمر السابق للمحافظة على بيانات الجدول، وإذا كنت قد فعلت فاكتب الأمر (ROLLBACK لاسترجاع البيانات.

مثال حذف جميع الموظفين المسجلين في الإدارة التي لها اسم (SALES).

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

6  row deleted.
```

نتيجة الاستعلام الفرعي هو الإدارة رقم (30)

في المثال السابق تم همل الاستعلام الفرعي لإيجاد رقم الإدارة التي لها اسم (SALES) من جدول الإدارات وناتج الاستعلام هو (٣٠)، ومن ثم استخدام هذا الرقم لحذف جميع الموظفين المسجلين في الإدارة رقم (٣٠)، وبالتالي فقد تم حذف أكثر من صف من جدول الموظفين.

ماذا يحدث لو أردت حذف الإدارة رقم (١٠) من جدول الإدارات ؟

```
SQL> DELETE FROM  dept
  2    WHERE deptno= 10   ;

ERROR at line 1 :
ORA-02292 : integrity constraint ( USR.EMP_DEPTNO_FK)
Vaiolated –child record found
```

عند حذف الإدارة رقم (١٠) من جدول الإدارات ظهرت رسالة خطأ تبين أنه تم انتهاك قيد أو شرط ربط جدول الإدارات بجدول الموظفين ومعنى ذلك أنه لا يمكن حذف الإدارة رقم (١٠) وذلك بسبب أنه لا يوجد موظفون مسجلون في هذه الإدارة داخل الجدول (emp) وذلك لوجود ربط بين الجدولين عن طريق العمود (deptno) فهو بالنسبة لجدول الإدارات يعتبر مفتاح أساسي (Primary Key) وبالنسبة لجدول الموظفين يعتبر مفتاح ربط (Foreign Key).

عمليات قواعد البيانات (Database Transactions):

يوجد عملية تبدأ عند قيامنا بالإضافة أو التعديل أو الحذف في قاعدة البيانات هذه العملية تسمى Database transaction وهي عملية انتقال البيانات من مرحلة إلى أخرى، فمثلاً عندما نقوم بعمل إضافة سجل أو جدول ما فإن هذه الإضافة لا تتم بشكل نهائي إلا إذا قمنا بإصدار أمر من أوامر Database transaction وهو الأمر (COMMIT) وهذا الأمر يعني تثبيت أو حفظ البيانات بشكل نهائي سواء كنا نقوم بعملية إضافة أو تعديل أو حذف بيانات، فهو يعادل تماماً أمر حفظ (SAVE). أوامر Database transaction وهي كالتالي:

- COMMIT يقوم هذا الأمر بحفظ البيانات التي تم إجراء عمليات الإضافة أو التعديل أو الحذف عليها.
- ROLLBACK يقوم هذا الأمر بالتراجع عن عملية الإضافة والتعديل والحذف.

أمر حفظ البيانات بشكل نهائي COMMIT

هذا الأمر ينفذ بطريقتين:

الأولى: ينفذ هذا الأمر بمجرد كتابته مباشرة بعد عمليات الإضافة والتعديل والحذف كالتالي:

```
SQL> DELETE FROM emp
     2 WHERE deptno = 30 ;
6 row deleted.
SQL> COMMIT ;
```

في المثال السابق تم حذف الموظفين الذين يعملون بالإدارة رقم (٣٠) ولكن ليست بصورة نهائية وحتى يتم الحذف بشكل نهائي لا بد من إصدار أمر COMMIT

الفصل الثاني

إدارة المخاطر السيبرانية والاستجابة للحوادث الأمنية

- مفهوم إدارة المخاطر السيبرانية داخل المؤسسات
- مهام إدارة المخاطر السيبرانية داخل المؤسسة
- تحديد المخاطر الأمنية و تقييمها
- استيراتيجيات التخفيف من المخاطر
- الاستجابة للحادث
  - تحديد/اكتشاف الحادث
  - الاستجابة الأولية و التحقيق
  - صياغة خطة الاستيراتيجية للحادث
  - تنفيذ خطة الاستجابة للحادث
  - الاستعادة و التعافي
  - إعداد التقارير
  - المتابعة / الدروس المستفادة
- المعايير و أفضل الممارسات
  - NIST
  - حل الوسط
  - جعل الأمن قابل للقياس
- استمرارية العمل
- التوعية و التدريب

إدارة المخاطر

السيبرانية والاستجابة للحوادث الأمنية

مقدمة :

تعد إدارة المخاطر جزءًا لا يتجزأ من أي استراتيجية على مستوى المنظمة، يجب بإستمرار على المنظمات إجراء تقييم لمخاطر السيبرانية لتقليل الحوادث الأمنية المكلفة نظرا لأهمية إدارة المخاطر الأمنية والامتثال.

الخطر (Risk):

هو احتمال ان يستغل عامل التهديد وجود نقطة ضعف، أو حالة القابلية للإصابة، ومن ثم النفاذ من خلالها

التعرض (Exposure):

هو حالة تعرض لخسائر نجمت عن عامل تهديد.

المضادات (Countermeasures) (أو الحماية Safeguard):

تستخدم المضادات أو أنظمة الحماية لمجابهة الأخطار المحتملة، والتخفيف من آثارها، وقد يكون المضاد عبارة عن إعدادات معينة للبرامج، أو جهاز محدد يؤدي وظيفة محددة. أو إجراءات معينة تتبع لمجابه الخطر والتخفيف من آثاره. وعادة ما يقضي المضاد على نقاط الضعف. ويقلل احتمال استغلال عامل التهديد لنقاط الضعف الموجودة.

ومن الامثلة على المضادات مايلي:

- وجود إدارة صارمة لكلمات المرور.
- وجود حراس أمن للمناطق المهمة والحساسة.
- تطبيق آلية للتحكم بالوصول على مستوى نظام التشغيل.
- استخدام كلمات مرور على مستوى نظام الدخل والخرج الاساسي (BIOS)
- التدريب والتوعية بالأمن السيبراني

تحليل المخاطر:

الخطر هو احتمال استغلال أي ضعف موجود في نظام الحماية، ومن ثم احتمال حدوث ضرر ما. وإدارة المخاطر هي عملية تحديد الخطر، ثم تقويمه، ثم العمل على تقليله وتقليل الأثار الناتجة عنه إلى اقل مستوى ممكن (المستوى المقبول)، ثم تطبيق الأليات اللازمة للمحافظة عليه عند هذا المستوى، ولايمكن لإدارة المخاطر السيبرانية ان تكون فاعلة وذات فائدة المنظمة ما لم تقم على أساس تحليل المخاطر، وفق المنهج العلمي الصحيح.

عملية (تحليل المخاطر):

هي أداة من أدوات إدارة المخاطر السيبرانية، ويمكن تعريفها بانها (طريقة تحديد نقاط الضعف والتهديدات وتقدير الأضرار المحتملة لها، ومن ثم تحديد أماكن تطبيق أنظمة الحماية المجابهة لها) ومن ثم فإن تحليل الخطر هو أداة تساعد على إعطاء الأولوية المناسبة لكل خطر، ثم تحديد المتطلبات اللازمة لمجابهته، سواء أكانت المركزية أو إدارية أو مالية.

مفهوم إدارة المخاطر السيبرانية داخل المؤسسات

إدارة مخاطر الأمن السيبراني هى مجموعة خطوات تتُخذ بشكل دوري لمواجهة التهديدات الإلكترونية ومعالجتها من خلال رصدها وتحديدها وتقييمها، ومن أجل إدارتها بفاعلية وذلك يتطلب نظرة شاملة لهذه المخاطر وتعاون من كافة أفراد العمل، ليس فقط من أفراد إدارة المخاطر وإنما أفراد الإدارات الأخرى.

و هي أيضًا عملية مستمرة لتحديد وتحليل وتقييم ومعالجة تهديدات الأمن السيبراني التي تواجهها المنظمة .
في الحالات العادية، قد تستخدم كلمة (الخطر) ليقصد بها (التهديد)، أو العكس، وبمعنى آخر قد تحل أي من كلمتي (الخطر) و (التهديد) محل الأخرى دونما تغيير في المعنى بالنسبة للاشخاص غير المتخصصين في الأمن السيبراني ، وقد تستخدم المصطلحات (الضعف). والتهديد والخطر، والتعرض للتعبير عن نفس الشئ رغم ان لكل منها معنى مختلفاً في علم الأمن السيبراني . وأن كلا منها ليست بديلا للاخر. لذا يجب قبل ، تعريف كل مصطلح من هذه المصطلحات بشكل دقيق ومعرفة معناه والعلاقة التي تربطه بالمصطلحات الأخرى.

الضعف أو قابلية الإصابة (Vulnerability) ، هي ضعف الأجهزة أو البرامج أو الإجراءات في أنظمة الحماية، التي ينتج عنها نقاط ضعف أو ثغرات أو أبواب مفتوحة، يمكن للمهاجم النفاذ من خلالها إلى الشبكة والأجهزة، ومن ثم الوصول غير المصرح به إلى معلومات المنظمة ومواردها المختلفة، ليس هناك بيئة تقنية آمنة كاملة بنسبة (١٠٠٪) فكل بيئة يوجد بها نقاط ضعف وقابلية للاصابة ومخاطر إلى درجة معينة، والمطلوب من إدارة المخاطر السيبرانية هو تحديد هذه النقاط والتهديدات، ثم تقويم احتمال حدوث أي منها، وتقييم الضرر الناتج عن ذلك، ثم أخذ الخطوات والتدابير اللازمة لتقليل الخطر بشكل عام على المستوى المقبول الذي تحدده المنظمة.

يمكن للمخاطر السيبرانية ان تكون في أشكال متعددة. وليس بالضرورة ان تكون كلها ذات علاقة بالحاسب الألي، فقد يكون الموظف غير المدرب جيداً أحد التهديدات المهمة، وكذلك فإن وضع اشرطة النسخ الاحتياطي في مكان غير محصن وغير محمي بنظم حماية مادية إدارية أو المركزية هو تهديد لأمن المعلومات المخزنة في تلك الاشرطة.

يصعب قياس الأخطار الحقيقية، لكن يساعد تصنيفها إلى فئات محددة ثم ترتيبها وفق أولويات تتناسب مع تلك الفئات، وتحدد أي الأنواع يجب التعامل معه اولاً وفقاً لاولويته. ، فليس هناك بيئة تقنية آمنة، ومن الطبيعي ان توجد قائمة من التهديدات المحتملة، فلو أن لدى المسئول عن الأمن السيبراني قائمة بتلك التهديدات، ولديه ميزانية محددة لاتكفي لمواجهة جميع هذه التهديدات، فإن دور إدارة المخاطر السيبرانية سيكون ترتيب هذه التهديدات وفقاً لدرجة خطرها، ثم توجيه الميزانية المتوفرة لمواجهة اكثرها خطراً على المنظمة.

تتم إدارة المراحل على مرحلتين هما :

أولا " : مراحل عملية إدارة مخاطر الأمن السيبراني قبل وقوع المخاطر

تعتمد إدارة مخاطر الأمن السيبراني فيها على خمسة خطوات رئيسية والتي تشمل ما يلي:

١- تحديد الأصول وبيئة تكنولوجيا المعلومات

لا بد من تحديد الأصول قبل حمايتها، يتم تحديد جميع التطبيقات والخدمات والأجهزة، المُستخدمة في مختلف الأعمال، أو التي تدعم أهم عملياتها.

ويدخل في نطاق الأصول، الأجهزة المتصلة بالإنترنت والتي يمكن اختراقها، لتكون بمثابة أجهزة انطلاق لتنفيذ هجمات عديدة.

لا بد من الإلمام ببيئة تكنولوجيا المعلومات، والتي يشمل نطاقها جميع البيانات والأصول الرقمية الأخرى والشبكات والأنظمة ومكونات وخدمات الطرف الثالث والتكنولوجيا ونقاط النهاية، وما إلى ذلك.

٢- تحديد المخاطر وتقييمها

يتم تعيين مستوى تهديد لكل برنامج وجهاز حاسب و محمول وخادم وجهاز نقاط البيع وجهاز هاتف محمول، اعتمادًا على مدى تعرضه للتهديدات، والوقوف على مدى تأثير تلك التهديدات على الأداء العام للعملية الأساسية للأعمال.

ويمكن أن تتكون تلك التهديدات من الفيروسات أو الاختراقات أو قلة خبرة المستخدم أو ضعف سياسات التأمين أو الإصدارات القديمة من الحلول غير المطابقة.

٣- وضع استراتيجية قوية لإدارة مخاطر الأمن السيبراني

يتم وضع استراتيجية وخطة مدروسة وقوية لإدارة مخاطر الأمن السيبراني، تلك الاستراتيجية التي تحتاج إلى تطوير وتخطيط مناسبين، ومواصلة تحديثها من قِبل المنظمة.

وتشمل خطط إدارة مخاطر الأمن السيبراني خطط الاستجابة للحوادث، ودور الموظفين والشركاء، ولأن الإنسان هو الحلقة الأضعف فيما يخص التعرض لتهديدات الأمن السيبراني؛ فلا بد من إخضاع الموظفين لتدريب على أساسيات الأمن السيبراني، وكيفية تفادي الأخطاء الشائعة مثل النقر على رابط غير معروف أو الوصول إلى حساب الشركة على شبكة غير آمنة.

يجب جعل الأمن السيبراني مسؤولية الجميع، هو شيء لا غنى عنه ويجب إدراجه في استراتيجية إدارة المخاطر.

٤- تحديد الحلول للتغلب على مخاطر الأمن السيبراني:

هى خطوة تحديد الحلول، وتتمثل هذه الخطوة في الوصول إلى حلول مؤقتة (قصيرة الأجل)، وحلول دائمة (طويلة الأجل)، لمنع تهديدات الأمن السيبراني.

وتتمثل أمثلة الحلول في تصحيح البرامج، وتدريب المستخدمين، وتنفيذ سياسات جديدة لتكنولوجيا المعلومات، وتركيب مضادات الفيروسات، وتشديد التحكم في الوصول.

٥- تنفيذ الحلول ورصد الفاعلية

تنفيذ القرارات التي جرى اتخاذها في أقرب وقت ممكن، ومن ثم البدء في الحماية من التهديدات.

ومعظم الحلول البرمجية لمراقبة مخاطر الأمن السيبراني، تحتوي على لوحات معلومات تظهر مستويات التعرض للمخاطر، وذلك للتأكد من أن الحلول الُمقدمة تساعد بالفعل في حل التهديدات.

وفي حال وجود ثغرات في السياسات، أو دفاعات ضعيفة، أو تم تحديد مخاطر جديدة غير متوقعة؛ فإن العملية بالكامل تعود إلى الخطوة الأولى، وتبدأ عملية إدارة مخاطر الأمن السيبراني من جديد.

وهذه العملية بحاجة إلى تشغيل وتطوير مستمرين، للتأكد من عدم ظهور مشكلات جديدة في المستقبل.

ثانيا" :مراحل عملية إدارة مخاطر الأمن السيبراني بعد حدوث مخاطر سيبرانية:

١- تحديد المخاطر:

يعمل القائمون على إدارة المخاطر في تلك المرحلة على تحديد التهديدات في الوقت الحالي ، كما أن هذه المرحلة تتطلب معرفة وتحديد بيانات وبرامج وأجهزة المنظمة المتأثرة .

٢- الحماية من المخاطر

تستهدف هذه المرحلة حماية البيانات والبرامج والأجهزة الخاصة بالمنظمة، وذلك من خلال تطبيق وسائل الحماية من أبرزها استخدام برامج مكافحة الفيروسات.

٣- كشف المخاطر

تتطلب هذه المرحلة الكشف عن المخاطر عبر تنفيذ أنظمة رصد التهديدات الإلكترونية.

٤- الاستجابة للمخاطر

عقب الانتهاء من مرحلة رصد المخاطر تأتي مرحلة الاستجابة والتي تُطبق فيها استراتيجيات بالاستجابة للمخاطر والتي تتضمن الاتصال والتعافي والاستجابة للتهديدات.

٥- التعافي من المخاطر

العودة إلى الوضع الافتراضي قبل التعرض للتهديدات الإلكترونية هو ما تتطلبه هذه المرحلة من أجل سير العمل من جديد، كما أنها خطوة تستهدف الحد من حوادث الهجوم الإلكتروني فيما بعد من خلال تطوير البرامج الخاصة بالأمن السيبراني.

مهام إدارة المخاطر السيبرانية داخل المؤسسة

١- تحديد المخاطر وتحليلها

٢- التخفيف من آثار المخاطر

٣- تحديد مدى احتمالية حدوث المخاطر وآثارها،

٤- تطوير سياسات إدارة المخاطر

٥- إعداد تقارير تقييم المخاطر.

مهارات إدارة المخاطر

مهارات مدير المخاطر هي مجموعة متميزة من القدرات التي تؤهل المتخصصين في المخاطر للقيام بمختلف المهام المُكلفين بها، إذ يعتمد مديرو المخاطر على مهاراتهم التحليلية للكشف عن المخاطر المُحتملة، ومهارات الاتصال لشرح تلك المخاطر لإدارة المؤسسة، و المهارات المالية من أجل حساب المخاطر، وغيرها.

عناصر إدارة المخاطر

١- العصف الذهني.

٢- التحليل الخاص بالسلامة الوظيفية.

٣- السيناريو المتبع في تحقيق الهدف.

٤- السجل الخاص بالمخاطر السابقة.

٥- الجولات الميدانية.

٦- سؤال الخبراء

تحديد المخاطر الأمنية و تقييمها

نظرًا لاعتماد المؤسسات بشكل كبير على أنظمة المعلومات للقيام بالعملية الإنتاجية، تزيد احتمالية حدوث الخطر، مما يعني أن جميع المنظمات معرضة لخطر الهجوم السيبراني، بالتالي يدور تقييم مخاطر الأمن السيبراني حول تحديد المخاطر وإدارتها والتحكم فيها عبر المنظمة.

أنواع المخاطر السيبرانية

تتعرض المؤسسات لأشكال متعددة من المخاطر السيبرانية، ومن أشهر هذه الأنواع ما يلي:

١- التنصت

تتعرض الأنظمة الإلكترونية للمؤسسات للتنصت عند سرقة المعلومات الهامة اعتمادًا على حركة المرور التي تُخترق، وذلك نتيجة الوصول إلى المعلومات المرسلة والمتبادلة بين المواقع المضيفة والمستخدم.

٢- سرقة كلمة المرور

يعتمد المخترقون بشكل شائع على سرقة كلمة المرور في سرقة البيانات الهامة للمستخدم، فقد يحدث ذلك عبر التنصت أو من خلال التخمين، وكلما كانت كلمة المرور سهلة كلما سهل اختراقها.

٣- البرامج الضارة

لا تخلو المخاطر السيبرانية من البرامج الضارة التي تنشأ جراء وجود جزء ما داخل البرنامج المثبتة ويكون غير مألوف مما يلحق أضرار بالأجهزة، يستهدف المجرمون الإلكترونيون إطلاق البرامج الضارة بغرض سرقة البيانات الهامة للمستخدمين والتي قد تكون بيانات شخصية أو مصرفية.

٤- عمليات الاحتيال

يعتمد المجرمون الإلكترونيون على وسائل أخرى للاحتيال وسرقة بيانات المستخدم من خلال إرسال رسالة عبر البريد الإلكتروني تتطلب من المستخدم تعبئة بياناته، وكثيرًا ما يقع المستخدمون في هذا الفخ مما يجعلها من أكثر الوسائل الناجحة في المخاطر السيبرانية.

٥- الهجوم عبر المواقع

في بعض الأحيان عند الرغبة في سرقة بيانات المستخدمين أو إلحاق الأضرار بخدمات الموقع فإن المجرمين الإلكترونين يستهدفون المواقع غير المشفرة عبر أكواد محددة تعطل الموقع، ويرسلون هذه الأكواد إلى الموقع بمجرد الوصول إليه من أجل التعطيل أو سرقة المعلومات المطلوبة.

٦- الهندسة الاجتماعية

يعتمد المحتالون على الهندسة الاجتماعية باستخدام مواقع التواصل الاجتماعي وهي جزء من عمليات الاحتيال التي يُستهدف فيها الحصول على كلمة المرور، البيانات المصرفية، جهاز المستخدم.

تتطلب الهندسة الاجتماعية امتلاك مهارة الإقناع لدى المجرم الإلكتروني حتى يتمكن من الوصول إلى ما يريده، وعادةً ما يلجأ إلى حيلة التنكر في هوية شخص آخر كمسئول من جهة البنك أو الضرائب وغيرها من الجهات الرسمية لطلب البيانات من المستخدم، إلى جانب اللجوء إلى وسائل أخرى مثل إرسال رسائل مزيفة.

عوامل زيادة مخاطر الأمن السيبراني

عند البحث عن الأسباب التي تزيد من وقوع المخاطر الإلكترونية نجد أن بعضها يكون بسبب أخطاء تحدث من الأفراد العاملين بالمؤسسة ومشكلات تقنية، وفيما يلي نوضح لك هذه العوامل:

١- وقوع عطل في الشبكات ينجم عنه فقدان بيانات هامة.

٢- استخدام أجهزة المنظمة من أماكن بعيدة عند السفر أو من المنازل.

٣- عدم الاطلاع على سياسات الأمن السيبراني ومراجعتها خلال عام.

٤- إهمال تحديث كلمة المرور باستمرار.

٥- دخول أحد الموظفين غير المختصين على نظام الأمن السيبراني ووصوله إلى الخيارات الإدارية الخاصة به.

٦- استخدام أجهزة المنظمة في إجراء المعاملات البنكية مثل تحويل الأموال.

٧- الوصول إلى النظام الإلكتروني للأمن السيبراني من مواقع أخرى من قبل العملاء.

تقييم المخاطر السيبرانية

تُستخدم تقييمات المخاطر السيبرانية لتحديد وتصنيف المخاطر التي تتعرض لها العمليات والأصول التنظيمية الناتجة عن استخدام أنظمة المعلومات.

الغرض الأساسي من تقييم المخاطر السيبرانية هو تقديم ملخص تنفيذي لمساعدة صانعي القرار وفهم قيمة المعلومات المطلوب حمايتها.

يعد تقييم مخاطر الأمن السيبراني مهمة كبيرة ومستمرة، لذلك يجب توفير الوقت والموارد لتحسين الأمن، أو تحديد التهديدات المحتملة، أو توفير نموذج أو تجنب مشكلة توقف التطبيقات، يجب أن يتكرر مع ظهور تهديدات جديدة.

من الناحية المثالية، يفهم موظفو تكنولوجيا المعلومات في المنظمة كيفية عمل البنية التحتية الخاصة بها، بالإضافة إلى المديرين الذين يفهمون كيفية تدفق المعلومات.

الخطوات التي يجب اتخاذها لإكمال عمل تقييم شامل للمخاطر السيبرانية.

خطوات تقييم المخاطر السيبرانية

- الخطوة الأولى: تحديد قيمة المعلومات

فالمهمة الأولى هي تحديد البيانات ومعرفة البنية التحتية للمنظمة وقيمة هذه البيانات.

يساعد تلخيص هذه المعلومات في فهم المخاطر التي تواجه فرق الأمان لتحديد أفضل الممارسات لتجنب المخاطر.

لا تمتلك معظم المنظمات ميزانية كبيرة لإدارة المخاطر السيبرانية، لذلك تحتاج إلى تحديد نطاق على الأصول الأكثر أهمية للأعمال، وقد تكون هناك حاجة إلى طرف ثالث متخصص في تقييمات المخاطر لمساعدة المنظمة.

- الخطوة الثانية: تحديد أولويات الأصول

بعد تحديد الأصول وتقييم المخاطر، يتم تحديد أولويات الأصول التي يجب تقييمها، بالتالي إنشاء قائمة جرد الأصول، حيث إنها طريقة أفضل لتصور مسارات الترابط بين الأصول والعمليات.

هناك بعض التهديدات التي ستكون في كل تقييم للمخاطر، وتشمل أنواع التهديدات الشائعة الوصول غير المصرح به أو إساءة استخدام المعلومات أو تسرب البيانات.

- الخطوة الثالثة: تحليل المخاطر

تحديد احتمالية الخطر، والذي يشير إلى الضرر الذي يلحق بالمنظمة نتيجة التهديد باستغلال ثغرة أمنية، لذلك يجب أن تتوفر ضوابط أمان قوية لتكنولوجيا المعلومات بما في ذلك النسخ الاحتياطي للبيانات ووضع كلمات المرور وما إلى ذلك.

وهناك العديد من التهديدات الأخرى غير الاختراقات والبرمجيات الخبيثة مثل الكوارث الطبيعية، وفشل النظام، والخطأ البشري.

- الخطوة الرابعة: تحديد نقاط الضعف

الثغرة الأمنية هي تهديد يمكن استغلاله لإلحاق الضرر بالمنظمة أو سرقة بياناتها ويتم العثور عليها من خلال تحليل نقاط الضعف وتقارير منتجي البرمجيات وتحليل الأمان.

يمكن تقليل تهديدات البرامج من خلال إدارة التصحيح المناسبة عبر التحديثات الإجبارية التلقائية.

- الخطوة الخامسة: تحليل الضوابط

تحليل الضوابط الموجودة في المنظمة لتقليل الضعف أو تنفيذ الضوابط من خلال التشفير أو آليات كشف التسلل أو المصادقة الثنائية.

تحاول الضوابط الوقائية تطبيق التشفير أو مكافحة الفيروسات أو المراقبة الأمنية المستمرة، وتحاول الضوابط الاستقصائية اكتشاف وقت حدوث هجوم مثل الكشف المستمر عن البيانات.

- الخطوة السادسة: حساب تقييم المخاطر

بعد معرفة قيمة المعلومات ونقاط الضعف والضوابط، فإن الخطوة التالية هي تحديد احتمالية وتأثير هذه المخاطر السيرانية في حالة حدوثها.

يعود الأمر كله إلى معادلة بسيطة:

تقييم المخاطر =التأثير (إذا تم استغلاله) * الاحتمال (للاستغلال في بيئة التحكم المُقيّمة).

بعض الأمثلة على تقييمات المخاطر هي:

مرتفع:

يعني تهديد عاجل للمنظمة ويجب أن تقليص المخاطر فوريًا.

متوسط:

يعني أنه يجب إستكمال تقليل المخاطر في فترة معقولة.

منخفض :

يعني أن التهديدات طبيعية ومقبولة بشكل عام.

- الخطوة السابعة: التوثيق

من المهم توثيق جميع المخاطر المحددة في سجل المخاطر، ويجب تحديث ذلك بانتظام للتأكد من أن الإدارة لديها دائمًا حساب محدث لمخاطر الأمن السيبراني الخاصة بها، ووضع تقرير تقييم المخاطر لمساعدة الإدارة في اتخاذ القرارات بشأن السياسات والإجراءات.

يجب أن تتوافق عملية تقييم المخاطر الناجحة مع أهداف إدارة المخاطر حتى تساعد على تقليل المخاطر بشكل فعال من حيث التكلفة، ثم يمكن بعد ذلك إنشاء سياسة لتقييم المخاطر تحدد فيها ما يجب على المنظمة القيام به لمراقبة وضعها الأمني، وكيفية تقليل المخاطر، وكيفية تنفيذ تقييم المخاطر.

استيراتيجيات التخفيف من المخاطر

تعتمد إدارة مخاطر الأمن السيبراني على استراتيجيات تساعد على ترتيب أولويات المخاطر المطلوب معالجتها؛ لرصد التهديدات الأكثر ضررًا والمطلوب مواجهتها في الوقت المطلوب.

ويتولى فريق أمن تكنولوجيا المعلومات مسؤولية تنفيذ سياسات إدارة مخاطر الأمن السيبراني، وهو ما يفرض على أفراد الفريق إلمامهم بأحدث طرق الهجوم لكل نوع من الأجهزة على الشبكة، ومن ثم تحديث أساليبهم الدفاعية، مع مراقبة تلك السياسات لمعرفة ما إذا كانت تنفيذها يمنع التهديدات بشكل فعال أم لا.

لكن النهج القائم على المخاطر للأمن السيبراني مرن وقابل للتخصيص لتلبية الاحتياجات والمخاطر المحددة للمؤسسة.

كما يؤكد تحديد وترتيب أولويات مخاطر الأمن السيبراني الأكثر أهمية، يليه تطبيق الضوابط للتخفيف منها.

ويتضمن هذا النهج:

- المراقبة المستمرة وإعادة التقييم، لضمان أن تظل الضوابط فعالة وذات صلة في مواجهة التهديدات السيبرانية دائمة التطور.
- التركيز على أهم التهديدات ونقاط الضعف.
- استيراتيجية تعزيز ثقافة الأمن السيبراني الاستباقية من خلال التقييم المستمر للمخاطر ومعالجتها، وتقليل تأثير الحوادث السيبرانية.
- اتخاذ قرارات مستنيرة حول مكان تخصيص موارد الأمن السيبراني الخاصة بها، وتحديد أولويات جهود الأمن السيبراني بناءً على أصولها ونقاط ضعفها الأكثر أهمية

يوجد أربعة استراتيجيات:

- أ- العلاج:

وهي استراتيجية تعتمد على إيجاد أدوات أمنية وأفضل الممارسات لحل المشكلة التي تسبب الخطر، مثل تركيب جدران نارية وخوادم بالوكالة وأدوات مضادة.

ب-قبول الخطر:

يعني التعايش مع الخطر ولكن بشرط أن يكون ضمن المعايير المقررة لقبول المخاطر.

- جـ -الإنهاء:

أي قطع النظام أو البرنامج أو الأجهزة بالكامل، وإعادة تصميم العمليات المتأثرة لتشغيلها دونها.

- د- نقل الخطر:

أي تقليل تأثير الخطر عن طريق تقاسمه مع طرف آخر، مثل الاستعانة بمصادر خارجية.

الاستجابة للحادث

إن أي منظمة تعتمد في عملها على الحاسبات الألية وشبكات المعلومات لابد ان تضع في حسابها ان هناك مخاطر تحيط بمعلوماتها، يجب معرفتها، وأخذ التدابير اللازمة للحيلولة دون وقوعها، وقد يحدث ان تتعرض إلى حوادث حقيقية، سواءً كارثة طبيعية، أو تحول بعض الأخطار إلى كارثة حقيقية يجب معالجتها وهذا ما يسمى بالاستجابة للحادث.

تحديد/اكتشاف الحادث

يعمل القائمون على إدارة المخاطر في تلك المرحلة على تحديد التهديدات المحتملة سواء في الوقت الحالي أو في المستقبل، كما أن هذه المرحلة تتطلب معرفة وتحديد بيانات وبرامج وأجهزة المنظمة.

عند حدوث الحادثة لابد من إبلاغ الجهات والاشخاص المسئولين بذلك، والطلب منهم ابلاغ من يلزم إبلاغه، والحضور لموقع الحادثة، والقيام بالاعمال المطلوبة منهم، ويجب في هذه الحالة ابلاغ المديرين ومديري مراكز المعلومات والمشرفين والفنيين والاستشاريين، ويمكن استخدام (شجرة الإنذار) للاتصال بهؤلاء الأشخاص، بحيث تكون مهمة كل شخص الاتصال بمجموعة أشخاص محددين لايزيد عددهم على ثلاثة أشخاص مثلاً.

بهذه الطريقة يمكن توصيل الرسالة لإكبر عدد من الأشخاص في أقصر وقت ممكن، ولا يشترط ان تعكس هذه الشجرة أي ترتيب اداري أو تنظيمي في المنظمة. كما يجب ان تكون الرسالة المبلغة واضحة وشاملة. ويمكن إعداد نص مسبق لها يمكن استخدامه بعد تعديله وفقاً للظروف وقت وقوع الحادثة ونوعيتها.

وقد لايقتصر الوضع على إبلاغ أشخاص وجهات داخل المنظمة فقط، بل يجب إبلاغ جهات أخرى، كجهات توريد الأجهزة، من اجل تقديم العون، أو أخذ الحيطة والحذر، ومن ذلك:

- إبلاغ المواقع الأخرى البعيدة التابعة للمنظمة.
- إبلاغ الشركة الموردة، والشركة (أو الشركات) التي تتولى تقديم خدمات أخرى ذات علاقة، لاحتمال الحاجة إلى مساعدتهم في علاج آثار الحادثة
- إبلاغ الشركات أو الجهات التي تقوم بخدمات الدعم الفني والصيانة.
- إبلاغ أي جهة أخرى قد تكون عرضة للحادثة

الاستجابة الأولية و التحقيق

الاستجابة للحادث تكون بالوصول لموقع الحادثة و تأمينه بواسطة فريق التحقيق الجنائي الرقمي و يجب اتباع خطوات محددة لمعالجة هذه الحادثة والتعامل معها، بحيث يكون هدفها النهائي هو استعادة الوضع كما كان عليه قبل وقوع الحادثة. ويجب وضع خطة محكمة لذلك، تشمل جميع الخطوات المطلوبة، وما تحتاج اليه كل خطوة من: معدات وأجهزة وبرامج وأشخاص مدربين ومؤهلين لتنفيذها على أكمل وجه

يختص مركز الاستجابة الخاص بالتعامل مع الحوادث السيبرانية وطوارئ الانترنت داخل المنظمة ، وذلك من خلال التنبؤ مواجهتها والتخفيف من آثارها ومنع تكرار حدوثها، بالاعتماد على منظومة تقنية غير تقليدية للمراقبة والرصد الأمني، فضلاً عن تحليل الأدلة الرقمية والثغرات الأمنية الخاصة بالجرائم السيبرانية على مستوى المنظمة، للوقوف على أسبابها ومنع تكرار حدوثها في المستقبل، وذلك بالإضافة إلى التعامل مع البرمجيات الخبيثة

التحقيق و تقييم الأضرار:

في هذه الخطوة يقيم الوضع بشكل عام. وتحصر الأضرار الناتجة عن الحادثة، وتحدد الأجهزة والبرامج التي توقفت عن العمل، وتلك التي مازالت تعمل، وعلاقة كل منها بالاخر، إن التقييم الصحيح لوضع النظام بعد حدوث الحادثة يحدد جميع الخطوات التي تليه فمثلاً، لو جرى تقويم الوضع, ووجد أن هناك بعض مغذيات الطاقة الاحتياطية مازالت تعمل وأن هناك معلومات في وضع حرج جداً، فيجب البدء على الفور بأخذ نسخ فورية من هذه المعلومات قبل فقدها نهائياً، أو إيقاف تشغيل طبيعي للأجهزة وقواعد البيانات، يضمن عدم تعرض المعلومات والبرامج التطبيقية للخطر

فريق الاستجابة للحادث لابد من أن يكون ذو مهارات عالية في التحقيق الجنائي الرقمي و هو جزء من المنظمة و يكون التحقيق داخليا دون تدخل سلطات الدولة .

صياغة خطة الاستيراتيجية للحادث

يجب توثيق مايتم عمله مبدئياً وبطريقة لاتشكل عبئاً اضافياً على الأعمال الأساسية لمعالجة الحادثة، يتم توثيق الأشياء الرئيسية فقط (رؤوس الموضوعات) التي سيجري استكمالها لاحقاً.

تحديد الأعمال التي يجب القيام بها وتنفيذها وترتيب اولوياتها

في هذه الخطوة يتم تحديد الأعمال التي يجب القيام بها لمعالجة الحادثة، وتوزيعها على المنفذين بشكل متناسق

تنفيذ خطة الاستجابة للحادث

البدء الفعلي في تنفيذ الأعمال مع مراعاة اولوية كل منها في التنفيذ. وتختلف هذه الأعمال واولوياتها باختلاف نوع الحادثة. فلو كانت الحادثة هجوماً على شبكة الحاسب الألي المحلية مثلا. فمن الأفضل فصل الشبكة عن الإنترنت فوراً، وحفظ ملفات سجلات الأعمال لتعقب المهاجم. وفي بعد الاحالات الأخرى يتم قبول الضرر الاخف تفادياً لوقوع الضرر الأكبر، ومن الامثلة على ذلك إغلاق الأجهزة إغلاقاً سريعاً آمنا . وقطع الخدمة عن المستفيدين دون إبلاغ مسبق، قبل انتشار برنامج تدميري على الشبكة.

تُطبق فيها استراتيجيات الاستجابة للمخاطر والتي تتضمن الاتصال والتعافي والاستجابة للتهديدات.

الاستعادة و+التعافي

العودة إلى الوضع الافتراضي قبل التعرض للتهديدات السيبرانية هو ما تتطلبه هذه المرحلة من أجل سير العمل من جديد، كما أنها خطوة تستهدف الحد من حوادث الهجوم السيبراني فيما بعد من خلال تطوير البرامج الخاصة بالأمن السيبراني.

ويمكن استعادة الوضع كنتيجة لتنفيذ الأعمال التي تم تحديدها في الخطوة السابقة وربطها مع بعضها بعضاً في شكل منظومة متكاملة.

يمكن إعادة بناء النظام المعلوماتي كاملاً من النسخ الاحتياطية، إذا كانت هذه النسخ محفوظة بشكل جيد وتحدث باستمرار، وتغطي معلومات المنظمة والشبكة والمستخدمين وأنظمة التشغيل وقواعد البيانات كافة.

فلو تم تدمير الأجهزة والبرامج كافة، وكان يوجد نسخ احتياطية جيدة وشاملة، فإنه يمكن جلب أجهزة جديدة واستعادة النظام المعلوماتي كاملاً من هذه النسخ.

إعداد التقارير

بعد استعادة الوضع يجب توثيق جميع ما حدث في تقارير، بدءاً من الحادثة نفسها، وانتهاءً باستعادة النظام كاملاً يتم توثيق كل ما تم عمله من إجراءات، وما تم اكتشافه من اخطاء وملاحظات تفيد في اكتشاف مدى الضرر الذي حدث، كما تفيد ايضاً في توثيق القرائن والدلائل التي تدل على الجاني، وتساعد في ملاحقته، ومن أهم ما يجب توثيقه في التقارير مايلي:

- سجل الأحداث والوقائع، ويفضل إرفاق نسخة مطبوعة من ملفات سجلات الأعمال (Log Files)
- الاتصالات التي تمت، وتواريخها، وأوقاتها، وأطرافها، وملخص ماتم فيها.
- القرارات التي أتخذت.
- الأعمال التي أنجزت، والأشخاص أو الجهات التي أنجزتها، وأوقات البدء فيها والانتهاء منها وتواريخها.

المتابعة / الدروس المستفادة

لابد من اتخاذ الإجراءات التي تضمن عدم تكرار هذا الحادث ، المشكلات والأخطاء التي تسببت في حدوثها، وكيف كان حلها؟ الخلاصة لما حدث، والتوصيات المستقبلية لمنع تكرار ذلك أو محاولة اجتنابه أو التخفيف من آثاره.

تزويد الجهات التي جرى إبلاغها لالحادثة بالنتيجة، وتزويد الجهات الداخلية في المنظمة والأشخاص المعنيين بها بنسخة من التوثيق كاملاً، مع التركيز على التوصيات والتدابير والتعليمات المستقبلية التي أقرت و كيفية متابعتها و الدروس المستفادة من الحادثة .

المعايير و أفضل الممارسات

يمكن أن تساعد أطر ومعايير حوكمة الأمن السيبراني المؤسسات على حماية نفسها من الهجمات الإلكترونية.

توفر هذه الأطر والمعايير مجموعة من أفضل الممارسات التي يمكن للمؤسسات اتباعها لتحسين وضع الأمن السيبراني لديها منها .

NIST

المعهد القومي الأمريكي للمعايير و التقنية

، Technology (NIST) National Institute of Standards and

هو مختبر لمعايير القياس وهو وكالة غير تنظيمية تابعة لوزارة التجارة في الولايات المتحدة الأمريكية.

وضع المعهد الوطني للمعايير القياسية والتكنولوجيا في الولايات المتحدة (NIST) إطاراً لتحسين الأمن السيبراني من خلال :

أ- تحديد المخاطر الإلكترونية المحتملة "Identify" :

من خلال التعرف بشكل دقيق على الأصول المتاحة "Assets" وبيئة العمل، ومن ثم المخاطر الإلكترونية التي قد تتعرض لها.

ب- وضع أنظمة للحماية من المخاطر الإلكترونية المحتملة "Protect":

بمجرد تحديد المخاطر الإلكترونية المحتملة، فإنه يجب العمل على الحماية منها من خلال وضع وتنفيذ الضمانات المناسبة، وذلك للعمل على استمرارية تقديم خدمات البنية التحتية الحيوية وعدم تأثرها بالهجمات الإلكترونية.

ج- وضع أنظمة لاكتشاف الهجمات الإلكترونية "Detect":

أي تطوير وتنفيذ الأنشطة المناسبة لضمان اكتشاف الهجمات الإلكترونية فور حدوثها.

د- الاستجابة ومواجهة الهجمات الإلكترونية "Respond": من خلال تطوير أنظمة قادرة على مواجهة أي هجمات إلكترونية فور اكتشافها.

ه- إصلاح منظومة الحماية الإلكترونية "Recover": بمجرد مواجهة الهجمات الإلكترونية، فإنه يجب إصلاح منظومة الحماية لتفادي تكرار هذه الحوادث.

تنفيذ معايير وأطر حوكمة الأمن السيبراني

تحديد الإطار الصحيح أو المعيار المناسب للمنظمة. بمجرد تحديد إطار عمل أو معيار ، تحتاج إلى تطوير خطة للتنفيذ. يجب أن تتضمن هذه الخطة الخطوات التالية:

- تقييم الوضع الحالي للأمن السيبراني الخاص بالمنظمة
- تحديد الثغرات في ضوابط الأمن السيبراني
- تطبيق الضوابط اللازمة
- مراقبة وتحسين وضع الأمن السيبراني الخاص بالمنظمة

توفر هذه الأطر والمعايير مجموعة متنوعة من الفوائد ، بما في ذلك:

- زيادة الوضوح في مخاطر الأمن السيبراني
- تحسين عمليات إدارة المخاطر
- ضوابط الأمن السيبراني المحسنة
- انخفاض تكاليف الأمن السيبراني

حل الوسط

استراتيجية الأمن السيبراني القابلة للاستخدام هي استراتيجية قوية بما يكفي لمنع المتطفلين غير المصرح بهم على الشبكة، ولكنها متساهلة بما يكفي للسماح للموظفين و في العمل باستخدام المعلومات التي يحتاجون إليها بطريقة مبسطة.

جعل الأمن قابل للقياس

يجب جعل الأمن مُقاس و بذل الجهد لامتلاك مجموعة من المقاييس الأمنية لأن في حال عدم وجود مقاييس تعتمد عليها المنظمة سيكون هناك حالة من عدم اليقين و الشك و يوجد أربعة مقاييس ضرورية و من أهمها:

١- الامتثال التنظيمي :

يمكن أن يوضح أن السياسات الأمنية الموضوعة للمؤسسة قد تم إتباعها

٢- الفاعلية التنظيمية :

يمكن أن تستخدم لتثبت للإدارة العليا في المنظمة فاعلية البرنامج للأمن السيبراني

٣- الإدارة المالية :

يمكن أن تستخدم لتثبت للإدارة العليا في المنظمة أن الاستثمارات في الأمن السيبراني كان لها ما يبررها

٤- التشغيلية :

هي تقيس اهتمام قسم الأمن السيبراني و قيامه بمهامه

• استمرارية العمل

هناك جملة من الاستعدادات والتدابير الاحترازية التي يجب الأخذ بها لمواجهة أي خطر يهدد النظام المعلوماتي بالمنظة، وتساعد هذه التدابير على استمرارية العمل الناتج عن بعض الأخطار التي تحولت إلى كوارث أو الكوارث التي تفاجئ المنظمة والعاملين بها، خاصة الطبيعية منها، ومن هذه الاستعدادات والتدابير الاحترازية مايلي:

١- تجهيز البدائل والمعدات مسبقاً:

يجب توفير الحلول البديلة من أجهزة وبرامج ومعدات لاستخدامها في حال وقوع الخطر أو تحوله إلى كارثة، ويجب ان تكون هذه البدائل جاهزة، مع ضرورة التدريب الجيد على استخدامها، ومن اهم ما يجب توفيره كبدائل جاهزة مايلي:

- أجهزة خوادم رئيسية مثبت عليها جميع البرامج اللازمة.
- أجهزة حاسب إلى طرفية مثبت عليها جميع البرامج اللازمة.
- أجهزة الربط الرئيسية المركزية لشبكة الحاسب الألي.
- نسخ اصلية جاهزة للاستخدام من أنظمة التشغيل والبرامج التطبيقية العاملة مع ارقامها التسلسلية ومفاتيح حمايتها.
- اشرطة واقراص تخزين خالية ومهيأة للاستخدام الفوري.
- العدد والأدوات التي قد تظهر الحاجة اليها عند وقوع الخطر (أو الكارثة)، مثل عدد الصيانة والبطاريات ومصابيح الاضاءة (الكشافات).
- بعض الكابلات الاضافية ووصلات الكابلات.

٢- النسخ الاحتياطي:

أهم ما يتم الاعتماد عليه عند استعادة المعلومات المفقودة، أو إعادة تشغيل النظام المعلوماتي باستخدام أجهزة جديدة بعد حدوث خطر أو كارثة معلوماتية، هو وجود نسخ احتياطية سليمة وشاملة لجميع المعلومات المفقودة. لذلك يجب ان تكون خطة النسخ الاحتياطي سليمة ومعدة بعناية، كما يجب ان تختبر باستمرار، ومن اهم التدابير اللازمة لذلك.

- وضع جدول زمني للنسخ الاحتياطي، بحيث يكون هناك نسخ يومية، وأسبوعية، وشهرية، وسنوية.
- اخذ نسخ احتياطية كاملة (Full)، وأخرى للإضافات اليومية (Incremental) فقط، بحيث يشمل كل منهما جميع معلومات المنظمة، بما في ذلك جميع قواعد البيانات والملفات، ومعلومات حسابات المستخدمين.
- يجب مراعاة أن تكون المعلومات المنسوخة حديثة وبفترات زمنية متقاربة، حتى يمكن استعادة آخر تحديث على المعلومات عند الحاجة.
- تخزين أشرطة أو وسائط النسخ الاحتياطي في أماكن تخزين مناسبة ومعدة لهذا الغرض.
- القيام بعملية استرجاع (Restore) للمعلومات من وسائط النسخ الاحتياطية إلى أجهزة فعلية مشابهة تماماً للأجهزة العاملة دورياً: من اجل التأكد من صحة عملية الاسترجاع نفسها، وانه يمكن تنفيذها في أي وقت متى ما دعت الحاجة إلى ذلك، وكذلك للتأكد من سلامة النسخ الاحتياطية الموجودة.

٣- خرائط الموقع والشبكة ومخططاتها:

يجب تجهيز الخرائط والمخططات الأتية من اجل استخدامها عند الحاجة:

- خريطة الموقع بشكل عام.
- مخطط شبكة الحاسب الألي، بما في ذلك نقاط التجميع (الكبائن) الرئيسية، واجهزة الموزعات (Switches) والموجهات (Routers) ، والتمديدات الخارجية، وخطوط الالياف البصرية، وخطوط الربط مع الشبكة الواسعة (WAN) ، وأجهزة أمن المعلومات، مثل جدران النار، وأجهزة تخزين المعلومات.
- مخطط مركز البيانات (Data Center) (أو غرفة الأجهزة الرئيسة) بشكل تفصيلي وبجميع محتوياته.
- خريطة خطوط تغذية الطاقة الكهربائية.
- خريطة المداخل والمخارج العادية وأخرى لمخارج الطوارئ.

يجب وضع المعلومات الضرورية عن محتويات كل مخطط أو خريطة على المخطط أو الخريطة نفسها، وحفظ هذه المخططات والخرائط بطريقة يسهل الرجوع اليها عند الحاجة، وينصح بشدة عمل مخطط واحد كبير الحجم يحوي جل هذه المخططات والمعلومات، ووضعه في مكان مناسب وبشكل دائم.

يجب كذلك تجهيز المعلومات الضرورية عن كل برنامج أو نظام تشغيل يجري استخدامه بحيث تشمل مايأتي:

- وصف موجز للنظام والمهام التي يؤديها.
- الاصدار المستخدم، مع توضيح ذلك على الأقراص الخاصة به.
- الطريقة التفصيلية لتثبيت البرنامج أو نظام التشغيل على الأجهزة من جديد
- متطلبات البرنامج أو نظام التشغيل من أجهزة وذاكرة ومساحة تخزين فارغة على الأجهزة المراد تثبيته عليها. والبرامج التي يجب تثبيتها قبل تثبيته.

- **التوعية و التدريب**

تحتوي السياسات الأمنية والمعايير القياسية والإجراءات والتوجيهات الأوامر والتعليمات من إدارة المنشأة حول تطبيق الأمن السيبراني في المنشأة والأليات المنظمة . يمكن تحقيق توجهات إدارة المنظمة إذا كان العاملون فيها يعرفون تلك السياسات والمعايير والإجراءات والتوجيهاتو لكن لا يستطيعون تنفيذها، ولا التعامل معها. من هنا كان التدريب على الأمن السيبراني والتوعية به ضرورة ملحة.

الهدف الرئيسي من التدريب والتوعية هو إيصال مفهوم الأمن السيبراني والسياسات الامنية العامة لكل موظف في المنظمة، ثم بعد ذلك التأكد من أن كل من السياسات الامنية الموضوعية أو المختصة بأنظمة محددة والمعايير القياسية والإجراءات والتوجيهات قد وصلت بالصورة الصحيحة لكل شخص يتطلب عمله فهمها وتطبيقها والتعامل معها. وأنها وصلت كذلك إلى كل قسم أو نشاط في المنظمة يجب ان تحكمه وتحدد مساره. وبهذه الطريقة يمكن معرفة من يحتاج إلى التدريب أو التوعية وفي أي مجال، ويمكن كذلك معرفة النتائج والمسؤوليات المترتبة على عدم تطبيق ماتم التدريب عليه والعلم به.

يشتمل كل من التدريب والتوعية بأمن المعلومات، سواءً أكان يتم التعامل معهما كوحدة واحدة أم منفصلين عن بعضهما بعضاً، على ثلاثة مستويات:

١- المستوى الأعلى:

وهو مستوى عام شامل يحتوي مواد تدريبية وتوعوية، قصيرة المدة، عامة المفاهيم، لمعرفة الخطوط العريضة لكل من السياسات الامنية والمعايير القياسية والتوجيهات والإجراءات، دون الدخول في التفاصيل، ويستهدف المستويات العليا من إدارة المنظمة.

٢- المستوى المتوسط:

متوسط الشمولية، ويحتوى مواد تدريبية وتوعوية متوسطة المدة ومتوسط التفاصيل، ويستهدف: المهندسين والاستشاريين ورؤساء الأقسام.

٣- المستوى الأدنى:

وهو مستوى تفصيلي يحتوي مواد تدريبية وتوعوية طويلة المدة، تحتوي معلومات تفصيلية عن كيفية تطبيق السياسات الامنية والمعايير القياسية والتوجيهات والإجراءات خطوة بخطوة على ارض الواقع، ويستهدف الأفراد والجهات التنفيذية من فنيين ومستخدمين ومستفيدين.

يجب ان يكون كل من التدريب والتوعية بالأمن السيبراني مستمرين طوال العام، وبصفة دورية، وأن يتما على كل مستوى من المستويات الثلاثة لمرة واحدة على الأقل في كل سنة.

المدة الزمنية	قصيرة (بالساعات أو الأيام)	متوسط (بالأيام أو الأسابيع)
امثلة لطرق التعليم	• كورسات قصيرة • فيديو • نشرات إخبارية • لوحات إعلانية	• محاضرات • تدريب على عينات • حالات دراسة سابقة • تدريب مباشر حي

الفصل الثالث

مقدمة في تطوير أمن البرمجيات

- مفهوم عملية تطوير البرمجيات
- مفهوم البرمجيات الآمنة
- الثغرات البرمجية التي يرتكبها المبرمجين و كيفية تفاديها
- دورة حياة تطوير البرمجيات SDLC
- تحديات تطوير البرمجيات
- المبادئ التوجيهية لأمن البرمجيات
- الممارسات الأساسية لأمن البرمجيات
- اختبار الأمن

مقدمة في تطوير أمن البرمجيات

يوجد احتياج أكثر من أي وقت مضى إلى مطوري البرمجيات الذين يكتبون برامج تقوم بأغلب المهام التي يحتاجها كل من المنظمات والأفراد على السواء، لتقليل الجهد البشري المبذول في تلك المهام من ناحية، ولتقليل نسب الخطأ والمخاطر من جهة أخرى ، وهو الأمر الذي تتفوق فيه البرمجيات على الإنسان فيه بما أنها تعمل في بيئات لا تتأثر بالمخاطر التي يتأثر بها الإنسان، ولا تتعرض إلى السهو والنسيان أثناء تنفيذ المهام وقد توسع مجال تطوير البرمجيات وتطوير البرمجيات نفسها مباشرة بعد تطور تقنيات وأدوات برمجية .

مفهوم عملية تطوير البرمجيات

عملية تطوير البرمجيات software development تشمل الخطوات والمناهج المتبعة في بناء وتصميم البرمجيات وكتابتها واختبارها وتجميعها و تحتوي هذه العمليات على تفاصيل وأقسام كثيرة بداخلها .

تُصمم البرمجيات أولًا وفقًا لمتطلبات العميل الذي يحتاج إلى حل مشكلة لديه، ويضع المبرمج إن كان يعمل مستقلًا أو فريق التطوير داخل المنظمة مخططًا لحل هذه المشكلة، ومنهجية لتنفيذ ذلك الحل، ثم ينطلق في الخطوات التقنية لتنفيذ ذلك الحل باستخدام أدوات تطوير البرمجيات ولغات البرمجة وأطر العمل اللازمة.

بعد تصميم البرنامج، يُختبر البرنامج الناتج ليُرى إن كان يحقق المعايير التي طلبها العميل أم لا، إلى أن يتم الوصول إلى النسخة النهائية التي تدخل بيئة العمل مباشرة بعد تمام التأكد من خلوها من الأخطاء البرمجية والمشاكل التي قد تعطل عملها فيما بعد

يشار أحيانًا إلى تطوير البرمجيات بأسماء متبادلة مثل تطوير التطبيقات أو أحيانًا برمجة أنظمة تحكم لعتاد مخصص مثل أنظمة إنترنت الأشياء ويمتد حتى أنظمة التشغيل.

مفهوم البرمجيات الآمنة

البرمجيات الآمنة هي نتيجة لعمليات تطوير البرمجيات المدركة للأمان حيث يُضمن الأمان وبالتالي تُطور البرامج مع وضع الأمان في الاعتبار .يكون الأمان أكثر فاعلية إذا خُطط له وأُدير طوال كل مرحلة من دورة حياة تطوير البرمجيات (SDLC)، خاصةً في التطبيقات المهمة أو تلك التي تعالج المعلومات

فالبرامج الآمنة هي واحدة من أساسيات الحماية ، مما يضمن أن البرامج آمنة بما تتوافق مع مبدأ عدم التسبب بالضرر. ولضمان سلامة البرامج، ينبغي بناء قدرات الموظفين، وإجراء تقييمات للمخاطر، ووضع تدابير للتخفيف

الثغرات البرمجية التي يرتكبها المبرمجين و كيفية تفاديها

الثغرات الأمنية هي خطأ في التعليمات البرمجية أو منطق التشغيل في نظام التشغيل أو برنامج التطبيق. ولأن أنظمة التشغيل والتطبيقات الحالية معقدة للغاية وتتضمن الكثير من الوظائف، يصعب على فريق التطوير لدى الشركة المصنعة إنشاء برامج لا تحتوي على أخطاء.

تشمل الأخطاء الشائعة ما يلي:

١- عدم الانتباه للتعليمات البرمجية والمتطلبات الدقيقة.

٢-عدم ابتكار تصميم متكامل قبل البدء في البرمجة.

٣-عدم فحص الأخطاء أو العيوب بشكل دوري و دقيق.

٤- استخدام أساليب البرمجة المعقدة والمجردة لعمليات بسيطة.

٥- الاعتماد على الأكواد الثابتة التي قد تكون انتهت صلاحيتها.

٦-عدم تنظيم الأكواد المصدرية والتعليقات اللازمة لتجنب الخلط فيما بينها.

٧-عدم مراعاة أمان الأكواد بشكل دقيق لمنع تهديدات الأمن الإلكتروني والاختراق.

تفادي الأخطاء الشائعة البرمجية:

١-اتباع الممارسات الموثوقة:

استخدام ممارسات برمجية موثوقة مثل التعليقات الجيدة، وإنشاء وثائق توضيحية واضحة، واستخدام أسماء متعارف عليها للمتغيرات والدوال.

٢- اختبار الوحدات والاختبارات الآلية:

كتابة اختبارات وحدات للتحقق من صحة الأكواد وتجنب الأخطاء الشائعة.

٣- استخدام إدارة الإصدارات:

استخدام أنظمة إدارة الإصدارات لتتبع التغييرات في الأكواد وتسهيل التعاون بين الفريق.

٤-التعلم من الأخطاء:

استخدام تقنيات تحليل الأخطاء وتعلم من الأخطاء الماضية لتجنب تكرارها في المشاريع المستقبلية.

٥-تجريب الأكواد:

اختبار الأكواد بشكل متكرر ومنتظم للتحقق من أنها تعمل بشكل صحيح ولا تسبب أي مشاكل.

٦- التواصل والتعاون:

التواصل المستمر مع أعضاء الفريق والتعاون معهم لضمان تحقيق أعلى جودة ممكنة.

٧-استخدام أدوات التحليل الثابتة:

استخدام أدوات التحليل الثابتة (Static Analysis Tools) للكشف المبكر عن الأخطاء في الأكواد وتحسين الجودة.

٨-التوثيق الجيد:

كتابة وثائق توضيحية ووثائق مستخدم شاملة للتعريف بالبرنامج وطريقة استخدامه.

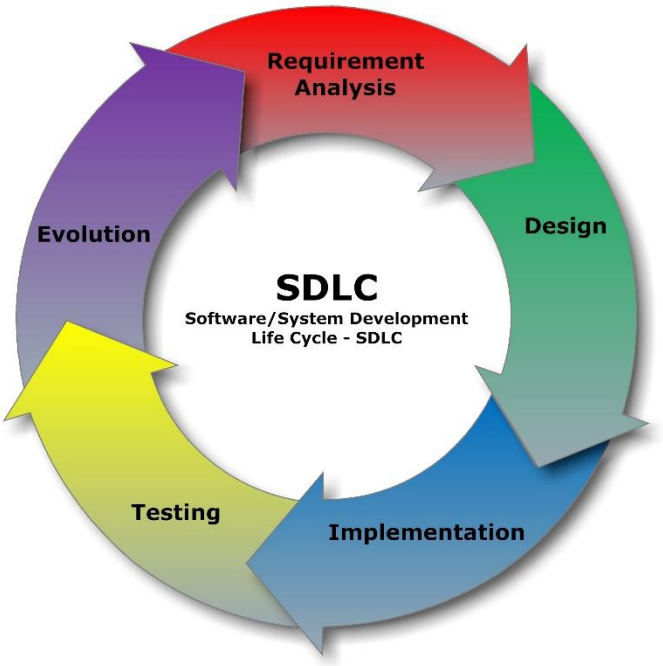
٩- استخدام تقنيات التصميم الجيدة:

استخدام تقنيات التصميم الجيدة مثل تصميم الكائنات وتقسيم الأكواد إلى وحدات صغيرة ومستقلة.

١٠-التدقيق الداخلي والتدقيق العملي:

تنظيم جلسات تدقيق الأكواد الداخلية والعملية للتحقق من الجودة وتحسينها.

دورة حياة تطوير البرمجيات SDLC



دورة حياة تطوير البرمجيات (SDLC) هي العملية الفعّالة من حيث التكلفة والموفِّرة للوقت التي تستخدمها فِرق التطوير لتصميم برمجيات عالية الجودة وتطويرها. الهدف من دورة حياة تطوير البرمجيات هو تقليل مخاطر المشروع من خلال التخطيط المستقبلي حتى يلبي البرنامج توقعات العملاء أثناء الإنتاج وبعده. وتوضِّح هذه المنهجية سلسلة من الخطوات التي تقسِّم عملية تطوير البرمجيات إلى مهام.

أسباب أهمية دورة حياة تطوير البرمجيات

قد يكون تطوير البرمجيات أمرًا يصعب إدارته بسبب المتطلبات المتغيرة، والترقيات التكنولوجية، والتعاون متعدد الوظائف. توفر منهجية دورة حياة تطوير البرمجيات (SDLC) إطار عمل إدارة منهجيًا مع عمليات تسليم محددة في كل مرحلة من مراحل عملية تطوير البرمجيات. لذلك، يتفق جميع أصحاب المصلحة على أهداف ومتطلبات تطوير البرمجيات مُقدمًا وتكون لديهم خطة أيضًا لتحقيق هذه الأهداف.

مزايا دورة حياة تطوير البرمجيات:

١- زيادة إمكانية رؤية عملية التطوير لجميع أصحاب المصلحة المعنيين

٢- كفاءة في التقدير والتخطيط والجدولة

٣- تحسين إدارة المخاطر وتقدير التكلفة

٤-تسليم البرمجيات بشكل منهجي وإرضاء العملاء بشكل أفضل

آلية عمل دورة حياة تطوير البرمجيات:

تُحدِّد دورة حياة تطوير البرمجيات (SDLC) العديد من المهام المطلوبة لإنشاء تطبيق برمجي. تمر عملية التطوير بعدة مراحل حيث يضيف المطورون ميزات جديدة ويصلحون الأخطاء في البرنامج.

وتختلف تفاصيل عملية دورة حياة تطوير البرمجيات باختلاف الفِرق. ومع ذلك، يوجد بعض مراحل دورة حياة تطوير البرمجيات الشائعة وهي:

١-الخطة

عادةً ما تتضمن مرحلة التخطيط مهام مثل تحليل التكلفة والعائد، والجدولة، وتقدير الموارد، والتخصيص. يجمع فريق التطوير المتطلبات من العديد من أصحاب المصلحة مثل العملاء، والخبراء الداخليين والخارجيين، والمديرين لإنشاء مستند مواصفات لمتطلبات البرنامج.

يحدد هذا المستند التوقعات، ويحدد الأهداف المشتركة التي تساعد في تخطيط المشروع. يقدِّر الفريق التكاليف، وينشئ جدولاً زمنيًا، وتكون لديه خطة مفصّلة لتحقيق أهدافه.

٢-التصميم

في مرحلة التصميم، يحلل مهندسو البرمجيات المتطلبات، ويحددون أفضل الحلول لإنشاء البرنامج وتحديد أدوات التطوير.

٣- التنفيذ

يكتب فريق التطوير التعليمات البرمجية للمنتج في مرحلة التنفيذ. ويحلل المتطلبات لتحديد مهام كتابة التعليمات البرمجية الصغرى التي يمكنه تنفيذها يوميًا لتحقيق النتيجة النهائية.

٤-الاختبار

يجمع فريق التطوير بين الأتمتة والاختبار اليدوي لفحص البرمجيات بحثًا عن الأخطاء. يشمل تحليل الجودة اختبار البرنامج بحثًا عن الأخطاء والتحقق مما إذا كان يفي بمتطلبات العميل. نظرًا إلى أن العديد من الفِرق تختبر التعليمة البرمجية التي تكتبها على الفور، فإن مرحلة الاختبار غالبًا ما تعمل بشكل متوازٍ مع مرحلة التطوير.

٥- النشر

عندما تطوِّر الفِرق البرامج، فإنها تُشفِّر نسخة مختلفة من البرنامج عن تلك التي يمكن للمستخدمين الوصول إليها وتختبرها. يُطلق على البرنامج الذي يستخدمه العملاء اسم الإنتاج، بينما يُقال إن النُسخ الأخرى توجد في بيئة الإنشاء أو بيئة الاختبار.

يضمن وجود بيئات إنشاء وإنتاج منفصلة أن العملاء يمكنهم مواصلة استخدام البرنامج حتى أثناء تغييره أو ترقيته. تتضمّن مرحلة النشر عدة مهام لنقل أحدث نسخة للإصدار إلى بيئة الإنتاج، مثل إنشاء الحِزم، وتكوين بيئة التطوير، والتثبيت.

٦-الصيانة

في مرحلة الصيانة، ومن بين المهام الأخرى، يُصلِّح الفريق الأخطاء ويحل مشكلات العملاء ويدير تغييرات البرمجيات الطارئة. بالإضافة إلى ذلك، يراقب الفريق أداء النظام العام والأمان وتجربة المستخدم لتحديد طرق جديدة لتحسين البرمجيات الحالية.

المقصود بنماذج دورة حياة تطوير البرمجيات

يقدم نموذج دورة حياة تطوير البرمجيات (SDLC) من الناحية النظرية دورة حياة تطوير البرمجيات بطريقة مُنظَّمة لمساعدة المؤسسات في تنفيذها. تُرتِّب النماذج المختلفة مراحل دورة حياة تطوير البرمجيات بترتيب زمني متغيّر لتحسين دورة التطوير. بعض نماذج دورة حياة تطوير البرمجيات الشائعة:

١- الانحداري

يُرتِّب النموذج الانحداري جميع المراحل بالتسلسل بحيث تعتمد كل مرحلة جديدة على نتيجة المرحلة السابقة. ومن الناحية النظرية، يتدفق التصميم من مرحلة إلى أخرى، مثل الشلال.

المميزات والعيوب

يوفر النموذج الانحداري الانضباط لإدارة المشروع ويمنح مخرجات ملموسة في نهاية كل مرحلة. لكن لا يوجد مجال للتغيير بعد اعتبار المرحلة مكتملة، حيث يمكن أن تؤثر التغييرات في وقت تسليم البرنامج وتكلفته وجودته. لذلك، يُعد هذا النموذج أكثر ملاءمةً لمشروعات تطوير البرمجيات الصغيرة، حيث يسهل ترتيب المهام وإدارتها ويمكن تحديد المتطلبات مسبقًا بدقة.

٢- التكراري

تقترح العملية التكرارية أن تبدأ الفِرق تطوير البرمجيات بمجموعة فرعية صغيرة من المتطلبات. بعد ذلك، تحسِّن الإصدارات بشكل متكرر بمرور الوقت حتى يصبح البرنامج الكامل جاهزًا للإنتاج. وينتج الفريق نسخة برمجية جديدة في نهاية كل تكرار.

المميزات والعيوب

من السهل تحديد المخاطر وإدارتها، حيث يمكن أن تتغير المتطلبات بين حالات التكرار. لكن يمكن أن تؤدي الدورات المتكررة إلى تغيير النطاق و التأثير على تقدير الموارد.

٣-الحلزوني

يجمع النموذج الحلزوني بين الدورات المتكررة الصغيرة للنموذج التكراري مع التدفق المتسلسل الخطي للنموذج الانحداري لتحديد أولويات تحليل المخاطر. يمكنك استخدام النموذج الحلزوني لضمان الإصدار والتحسين التدريجي للبرنامج من خلال إنشاء نماذج أولية في كل مرحلة.

المميزات والعيوب

النموذج الحلزوني مناسب للمشروعات الكبيرة والمعقدة التي تتطلب تغييرات متكررة. لكن يمكن أن يكون الأمر مكلفًا للمشروعات الصغيرة ذات النطاق المحدود.

٤- المرن

يُرتِّب النموذج المرن مراحل دورة حياة تطوير البرمجيات في عدة دورات تطوير. يكرّر الفريق المراحل المتتالية بسرعة، ولا يقدم إلا تغييرات صغيرة وتزايدية في البرمجيات في كل دورة. يقيِّم باستمرار المتطلبات والخطط والنتائج حتى يتمكّن من الاستجابة بسرعة للتغيير. يُعد النموذج المرن تكراريًا وتزايديًا، ما يجعله أكثر كفاءة من نماذج العمليات الأخرى.

المميزات والعيوب

تساعد دورات التطوير السريعة الفِرق في تحديد المشكلات ومعالجتها في المشروعات المعقدة في وقت مبكر وقبل أن تصبح مشكلات كبيرة. يمكن لهذه الفِرق أيضًا إشراك العملاء وأصحاب المصلحة للحصول على ملاحظات طوال دورة حياة المشروع. لكن قد يؤدي الاعتماد الزائد على ملاحظات العملاء إلى تغييرات مفرطة في النطاق أو إنهاء المشروع في منتصفه.

كيفية معالجة دورة حياة تطوير البرمجيات مسألة الأمان

في تطوير البرمجيات التقليدية، كان اختبار الأمان عملية منفصلة عن دورة حياة تطوير البرمجيات .(SDLC) ولا يكتشف فريق الأمان الثغرات الأمنية إلا بعد إنشاء البرنامج. أدى ذلك إلى ظهور عدد كبير من الأخطاء التي ظلت خفية بالإضافة إلى زيادة المخاطر الأمنية.

تم تدارك معظم الفِرق أن الأمان جزء لا يتجزأ من دورة حياة تطوير البرمجيات. حيث يمكن معالجة جانب الأمان في دورة حياة تطوير البرمجيات وإجراء تقييمات الأمان أثناء عملية دورة حياة تطوير البرمجيات بأكملها.

DevSecOpsهي ممارسة دمج اختبار الأمان في كل مرحلة من مراحل عملية تطوير البرمجيات. تتضمن الأدوات والعمليات التي تشجع على التعاون بين المطورين والمتخصصين في الأمان وفِرق التشغيل لإنشاء برمجيات يمكنها التغلب على التهديدات الحديثة. بالإضافة إلى ذلك، فإنها تحرص على كون أنشطة ضمان الأمان، مثل مراجعة التعليمة البرمجية، وتحليل البنية، واختبار الاختراق، جزءًا لا يتجزأ من جهود التطوير.

تحديات تطوير البرمجيات

يعد مجال صناعة التكنولوجيا للبرمجيات ضمن أحد أوسع المجالات الأكثر تعقيدا والأكثر ديناميكية على الإطلاق، لذلك كان لابد من أن تتأثر البرمجيات بالعديد من العوامل والتحديات الكثيرة على مر السنوات الماضية والقادمة، كتأثرها بظروف السوق، التحديات التي تتم مواجهتها أثناء تطويرالبرمجيات هي:

١-المنازعات داخل منظمات البرمجيات

٢- متطلبات البرامج الغير واضحة والمتغيرة باستمرار

٣- عدم القدرة على تحديد استخدام التكنولوجيا المناسبة

٤- التقدم السريع في عالم التقنيات

٥- سوء إدارة الجداول الزمنية للمشاريع

٦- ضمان الحفاظ على الجودة

٧- انقطاع التواصل بين فريق العمل

٨- التغيرات الثقافية لتطوير البرمجيات و صعوبة تكيف المطورين معها

٩- الأمن السيبراني فهو أخطر تحديات تطوير البرمجيات

١٠- الذكاء الاصطناعي و الأتمتة مثل تحديد متى يجب أتمتة العمليات واختبارها

١١- سرية المعلومات

١٢- تكاليف تطوير البرمجيات

المبادئ التوجيهية لأمن البرمجيات

١-توثيق الإجراءات الرسمية لتغيير نظم البرمجيات

٢- تقييم المخاطر و تحليل الآثار المترتبة على التغيرات

٣- الحصول على موافقة رسمية من الإدارة المسؤولة قبل إجراء أي تغيير

٤- اختبار إجراءات السلامة للتأكد من أنها لم تتأثر بأي تغييرات في نظام التشغيل

٥- إمكانية الحصول على التعديلات من المورد أو المُنتج الأصلي

٦- منع فرص تسرب المعلومات

٧- الاستعانة بمصادر خارجية لتطوير البرمجيات يجب أن يخضع لرقابة و إشراف مُنظم

٨- التأكد من صحة التراخيص و حقوق الملكية الفكرية عند التعامل مع جهات خارجية لتطوير البرمجيات

٩-تتضمن بعض التحديات من وجهة نظر أمان تطوير البرامج الفيروسات وأحصنة طروادة والقنابل المنطقية والديدان والوكلاء و البرامج الصغيرة كما يمكن أن تحتوي البرامج على ثغرات أمنية يمكن أن يقدمها مهندسو البرامج إما عن قصد أو بلا قصد.

١٠- تطوير البرمجيات يكون باستخدام لغات برمجة عالية المستوى والتي يمكن أن تكون لها آثارًا أمنية في حد ذاتها.

الممارسات الأساسية لأمن البرمجيات

تضمن أفضل ممارسات لأمن البرمجيات الآتي:

١- التحقق من صحة الإدخالات:

يجب التحقق من الإدخالات التي يتم إدخالها إلى التطبيق بحيث يتم تفادي الهجمات عن طريق إدخال بيانات خبيثة.

٢-تحديث البرامج المستخدمة:

يجب تحديث البرامج المستخدمة بشكل دوري لتجنب وجود ثغرات وتسوية المشاكل الأمنية.

٣- حماية البيانات :

يجب تشفير البيانات وتحديث تقنيات التشفير بين الفترات لضمان عدم اختراق حسابات المستخدمين أو البيانات الشخصية.

٤-إعدادات الأمان المنخفض:

يجب الحد من الخطورة وتطبيق إعدادات الأمان المنخفض للحد من فرص الاختراق.

٥-تطوير التطبيقات الآمنة :

يجب تصميم التطبيقات بأسلوب يضمن بدء تشغيلها واغلاقها بطريقة آمنة.

٦-التحقق المتعدد :

يجب تفعيل ميزة التحقق المتعدد من أجل تعزيز الأمان وتفادي الاختراق.

٧-التدريب المستمر :

يجب على الفرق البرمجية أن تحصل على تدريب دوري لتحديث المعرفة وتعزيز أفضل الممارسات الأمنية.

اختبار الأمن

هو عملية تهدف إلى الكشف عن العيوب في آليات الأمان لنظام المعلومات التي تحمي البيانات وتحافظ على وظائفها على النحو المنشود. نظرًا للقيود المنطقية لاختبار الأمان ، فإن اجتياز اختبار الأمان ليس مؤشراً على عدم وجود عيوب أو أن النظام يفي بمتطلبات الأمان بشكل كاف.

الأمن هو "حالة التحرر من الخطر أو التهديد"، أمن أنظمة البرمجيات على وجه الخصوص هو موضوع واسع، أمن البرمجيات هو تطبيق التقنيات التي تقيّم وتخفف وتحمي أنظمة البرمجيات من نقاط الضعف وتضمن هذه التقنيات استمرار البرنامج في العمل وأنه في مأمن من الهجمات ويتضمن تطوير البرامج الآمنة مراعاة الأمان في كل مرحلة من مراحل دورة تطوير البرامج لحمايتها من التهديدات السيبرانية المحتملة نظرا لأن الهجمات أصبحت معقدة بشكل متزايد ، فمن الضروري ضمان حماية البرامج . توفر خدمات اختبار أمان البرامج من SubRosa اختبار اختراق شامل للكشف عن نقاط الضعف .. فهو يحدد اختبار الاختراق الخاص بنقاط الضعف قبل أن يتمكن المهاجمون من استغلالها ، مما يقلل بشكل كبير من خطر حدوث اختراق أمني.

من خلال دمج اختبار الأمان طوال عملية التطوير ، يمكنك إنشاء تطبيقات أكثر أمانا من الألف إلى الياء.

تقنيات أمن البرمجيات:

يؤدي تطبيق تقنيات أمان البرمجيات على البرمجيات إلى إنتاج مستويات أعلى من الجودة، حيث تتمتع البرامج الأكثر أمانًا بسلوك صحيح ويمكن التنبؤ به، وهناك سبع ممارسات لتأمين البرامج:

١- مراجعة الكود باستخدام أدوات؛ للعثور على الأخطاء البرمجية ونقاط الضعف.

٢-تحليل المخاطر التصميم لتحديد العيوب.

٣- اختبار الاختراق.

٤-اختبار الأمان القائم على المخاطر.

٥-حالات لفحص كيفية تصرف النظام عند التعرض للهجوم

٦- متطلبات الأمن

٧- العمليات الأمنية

المراجع

١-	WM. Arthur Conklin,Greg White ,2018,Principles of Computer Security, maps to Compatia Security + ,Comptia
٢-	الإدارة العامة لتصميم وتطوير المناهج ،مقدمة قواعد بيانات أوراكل ، المؤسسة العامة للتدريب التقني و المهني
٣-	د.عدنان مصطفى البار،د.عيسى رفاعي السميري، ١٤٤٠هـ ،أساسيات الأمن السيبراني،مكتبة الملك فهد الوطنية
٤-	خالد عياد الحلبي ،٢٠١١م ، اجراءات التحري والتحقيق في جرائم الحاسوب والانترنت،دار الثقافة
٥-	د.محمد رضوان هلال ١٤٢٨ هـ ، دار العلوم للنشر
٦-	Dr.Jeetendra Pande ,Dr.Ajay Prasad ,2016, Digital Forensics,Uttarakhand Open University