الفصل العاشر
ضوابط سلامة البيانات
Data Integrity

للتأكد من سلامة البيانات ودقتها داخل قاعدة البيانات **(Data Integrity)** *فإننا نقوم بعدد من الإجراءات والضوابط أثناء تصميم قاعدة البيانات والجداول كما سوف نحتاج إلى تفهم كامل للعلاقات بين الجداول ومدى اعتماد بعض الجداول على البعض الآخر، وهناك العديد من الضوابط* **Constraints** *التي يمكن استخدامها .*

في هذا الفصل سوف نتحدث عن الموضوعات التالية :

- *مقدمة عن الضوابط ولماذا نستخدمها*
- *أنواع الضوابط.*
- *غير خالى من البيانات* **Not Null**.
- *المفتاح الأساسى* **Primary Key**.
- *منع التكررات* **Unique**.
- *المفتاح الخارجى* **Foreign Key**.
- *التأكد* **Check**.
- *تنظيم الضوابط.*
- *الاستخدام الأمثل للضوابط.*

مقدمة عن الضوابط (Constraints) ولماذا نستخدمها

هناك عدد من الضوابط **Constraints** التي يمكن استخدامها للتأكد من أن البيانات تم إدخالها بشكل صحيح لضمان التعامل معها بدون أخطاء ، فمثلا عندما تضع برنامج لإدخال البيانات في جدول معين فانك تريد أن يتم إدخال البيانات في بعض الحقول دون تركها فارغة **NOT NULL** بينما إدخال البيانات في حقول أخرى يكون اختياريا **NULL** كما أن بعض الحقول يمكن أن تكون بياناتها متكررة مع حقول أخرى مثل النوع فممكن أن يكون **Male** لأكثر من موظف ولا يعتبر هذا تكرارا، بينما رقم البطاقة الشخصية لا يمكن تكراره كما أن بعض الضوابط سوف تمنع إدخال بيان في جدول فرعي لا يوجد له مقابل في الجدول الرئيسي، كأن تكتب بيان في جدول المرتبات لموظف ولا يوجد له بيان في جدول الأسماء وهكذا .

هذه الضوابط **Constraints** يتم وضعها أثناء تكوين الجداول وبعضها يمكن إضافته بعد تكوين الجداول ، والهدف هو المحافظة على سلامة البيانات ودقتها **Integrity** وذلك قبل إدخالها أو أثناء إدخالها في قواعد البيانات ، وهي تساعدك أثناء تصميم قاعدة البيانات والجداول في التأكد من أن موظف إدخال البيانات لن يستطيع إدخال البيانات بطريقة يصعب التعامل معها ، فمثلا لو هناك رقم التليفون كحقل داخل جدول كيف سيقوم موظف إدخال البيانات بإدخال رقم التليفون في الحقل؟ مثلا يقوم موظف بإدخال الرقم هكذا :

٠٠٢٠٢١٢٣٤٥٦٧ بينما يقوم موظف آخر بكتابته هكذا **+202-1234567** وموظف ثالث يمكن أن يكتبه هكذ **(00202) 123 4567** وموظف آخر يقول "سيبك مفيش داعي لرقم التليفون هو مين هايطلبهم أو يسأل فيهم"

تخيل أن المؤسسة التي تعمل لها النظام بها الف أو عدة آلاف من الموظفين، ففي الحالة السابقة هل يمكنك أن تتأكد من دقة بيانات حقل رقم التليفون وتتأكد أنه غير متكرر ؟

أنواع الضوابط Constraints

سوف نتعلم فيما يلي كيفية إنشاء واستخدام مجموعة من الضوابط داخل جداول قاعدة البيانات :

- غير خالي من البيانات **NOT NULL**
- المفتاح الأساسي**Primary Key**
- منع التكرارات **Unique**
- المفتاح الخارجي **Foreign Key**
- التأكد **Check**

استخدام المعيار NOT NULL

قمت بتصفح أحد المواقع على "النت" كما يقولون، ووجدت أن هناك فرصة مجانية للحصول على رحلة ثلاثة أيام دون أي تكلفة، تساءلت هل هؤلاء الناس مجانين، كل المطلوب مني هو أن أملأ طلب، قلت لم لا أجرب، بدأت في إدخال البيانات، وجدت أن بعض الحقول أمامها نجمة حمراء علامة على أنها ضرورية، قلت الرحلة تساوي، ولكن هناك كان حقل أمامه نجمة حمراء ويجب أن أكتب بياناته وهو رقم **Master Card** وأمامه عبارة "لضمان الجدية فقط فلا تخشى شيئا"، قلت لن أكتب هذا الرقم أبدا وتجاهلته ثم قمت بضغط زر **submit** فعادت الي الصفحة نفسها برسالة تقول لا بد من ملء بيانات الحقول التي أمامها نجمة حمراء ، قلت لن أذهب الى هذه الرحلة وعدت الى القاعدة الراسخة في نفسي "ليس هناك شيئا مجانيا" .

هذه الحقول ذات النجمة الحمراء أكيد تم تعريفها في قاعدة البيانات بأنها **NOT NULL** من البداية وبالتالي فان البرنامج الذي كنت أتعامل معه يتأكد أيضا من ضرورة إدخال بيانات لمثل هذه الحقول ! أما الحقول الأخرى فهي اختيارية يمكن وضع بيانات بها أو تجاهلها أي أنها يمكن أن تحمل قيمة **NULL** .

كما سبق القول فان NULL هي قيمة أو حقل بلا قيمة فهو ليس مسافة Blank وليس صفرا ٠ ، قيمته الحقيقية NULL .

وكما هو معروف فان أنواع البيانات لا تعدو في النهاية أن تكون حروف Alphanumeric أو أرقام NUMBERS أو تاريخ ووقت DATE/TIME وسوف نقوم بإنشاء جدول بصورته العادية التي تعلمناها.

```
create table emp_main (
emp_id      varchar(8), Not Null
name1 varchar(10), Not Null
name2 varchar(10), Not Null
name_l      varchar(10), Not Null
address     varchar(30), Not Null
city        varchar(15), Not Null
gov         varchar(15), Not Null
tel         varchar(9));
```

عندما تقوم بتجربة هذه الأمثلة تأكد بعد كل مرة أن تقوم بحذفها بأمر Drop Table بعد الانتهاء من تجربتها حتى تستطيع العمل بدون مشاكل فقد تعمل هذه الأمثلة ولكن قاعدة البيانات النشطة تكون غير التي تريدها، أيضا لا تنس استخدام أمر تنشيط قاعدة البيانات بأمر Use databasename.

نلاحظ أننا وضعنا NOT NULL لكل الحقول في الجدول فيما عدا حقل رقم التليفون TEL فهو اختياري رغم أنني مقتنع أن كل الموظفين لديهم تليفونات في منازلهم وفي جيوبهم أيضا !!! إذا لم تحدد القيمةNOT NULL ضمن مواصفات الحقل فإن قاعدة البيانات تستخدم تلقائيا القيمة NULL .

استخدام المفتاح الأساسي Primary Key

يستخدم هذا المفتاح ليكون الوسيلة التي يتم بها التعرف على السجل مثل "الرقم القومي" وهو رقم تعريفي للبيانات الموجودة في جدول، وممكن أن يتكون من حقل واحد أو عدة حقول، فيما يلي سوف نستخدم رقم الموظف emp_id ليكون هو المفتاح الأساسي Primary Key وسوف نعرف فيما بعد أن الجداول التابعة ستستخدم مفتاح

الجدول الأساسي للحصول على معلومات . في المثال التالي سوف نقوم بتخصيص حقل رقم الموظف لكي يستخدم كمفتاح أساسي **Primary Key** :

```
CREATE TABLE EMP_MAIN (
EMP_ID      CHAR(8)    NOT NULL,
NAME1       VARCHAR2(10)    NOT NULL,
NAME2       VARCHAR2(10)    NOT NULL,
NAME_L      VARCHAR2(10)    NOT NULL,
ADDRESS     VARCHAR2(30)    NOT NULL,
CITY        VARCHAR2(15)    NOT NULL,
GOV         VARCHAR2(15)    NOT NULL,
ZIP         NUMBER(5)    NOT NULL,
TEL         NUMBER(9),
PRIMARY KEY (EMP_ID));
```

لاحظ أن استخدام هذا المفتاح أيضا يمنع تكرار رقم الموظف بطريق الخطأ أو الصواب . كما أن من فوائد **Primary Key** أنه يحافظ على ترتيب البيانات داخل الجدول .

منع *التكرارات* Unique

المفتاح الأساسي **Primary Key** لا يتكرر ، ولكنه يحافظ على الترتيب ، بينما نحتاج الى عدم تكرار بيانات بعض الحقول ، مثلا تريد ألا يتم تكرار رقم التليفون بين اثنين أو أكثر من الموظفين وأن ينفرد كل موظف برقم تليفون واحد ، هنا يمكن استخدام المعيار **Unique** ، انظر المثال التالي وكيف يستخدم القيمة **UNIQUE** لمنع التكرارات في بيانات الصف (السجل) .

```
CREATE TABLE EMP_MAIN (
EMP_ID      CHAR(8)    NOT NULL,
NAME1       VARCHAR2(10)    NOT NULL,
NAME2       VARCHAR2(10)    NOT NULL,
NAME_L      VARCHAR2(10)    NOT NULL,
ADDRESS     VARCHAR2(30)    NOT NULL,
CITY        VARCHAR2(15)    NOT NULL,
GOV         VARCHAR2(15)    NOT NULL,
TEL         INT (9) UNIQUE,
PRIMARY KEY (EMP_ID));
```

استخدام مفتاح أجنبي Foreign Key

هو مفتاح أجنبي صحيح ولكنه غير مستورد ! في الحقيقة يتم استخدام هذا المفتاح داخل العائلة ، فكما سنعرف في البند التالي ، أن أهم علاقة داخل قاعدة البيانات هي علاقة الأب بولده وولد ولده حيث يكون الجدول الرئيسي هو الأب أو الوالد **Parent** والجداول الفرعية هي جداول أبناء **Child** من هنا فان الابن أيضا في مجموعة قد يصبح أبا لمجموعة أخرى وهكذا .

هذا المفتاح **Foreign Key** عبارة عن حقل في جدول فرعي **child** يشير الى مفتاح أساسي **Primary Key** في الجدول الرئيسي **Parent** ولنأخذ مثالا :

```
CREATE TABLE EMP_SALARY (
EMP_ID      CHAR(9)          NOT NULL,
TITLE       VARCHAR2(15)     NOT NULL,
SALARY      INT(6)       NOT NULL,
FOREIGN KEY (EMP_ID) REFERENCES EMP_MAIN (EMP_ID));
```

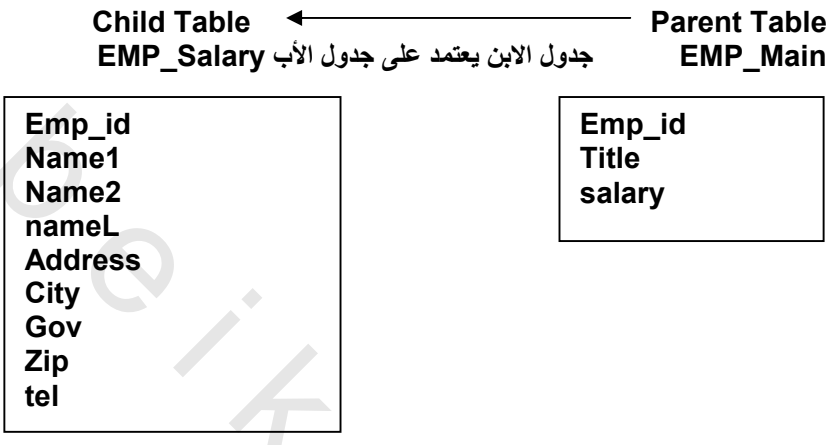
لاحظ أننا قمنا بتكوين جدول فرعي (الابن) **emp_salary** يحتوي على رقم الموظف ووظيفته والمرتب ولكن باقي بيانات الموظف موجودة في الجدول الرئيسي (الأب) **emp_main** العلاقة التي تربط بين الجدولين هي أن مفتاح الجدول الفرعي **Foreign Key** هو عبارة عن اشارة **references** الى المفتاح الرئيسي **Primary Key** في الجدول الرئيسي وهي علاقة وثيقة وبالتالي فانك لن تستطيع ادخال بيانات موظف في الجدول الفرعي وليس له بيانات في الجدول الرئيسي ، وهذا ما سوف نوضحه في القسم التالي .

العلاقة بين الوالد والابن في الجداول

توضيحا لموضوع **Foreign Key** كما قلنا ، فانه يمكن ايجاد علاقة حميمة بين الجداول وبعضها البعض تشبها بالعلاقات الحميمة بين البشر

في المثال السابق كان لدينا جدول رئيسي **emp_main** يحتوي على بيانات كل الموظفين هذا الجدول يسمى والد **parent** لأن كل الجداول الأخرى تشير اليه بينما جدول المرتبات فرعي يسمى **emp_salary** يحتاج دوما الى الجدول الوالد **parent table**

للحصول على اسم الموظف هذه العلاقة تسمى **Parent/Child** انظر الشكل ١٠-١



شكل ١٠-١ العلاقة بين الجداول **Parent/Child**

العلاقة التي تربط بين الجدولين هي رقم الموظف **emp_id** وكل موظف في كشف المرتبات أو جدول المرتبات **emp_salary** يجب أن يكون له سجل في الجدول الرئيسي ، والذي يربط بين الجدولين هو اضافة **Foreign Key** الى الجدول الفرعي ، هذا المفتاح سوف يربط رقم الموظف في الجدولين برباط وثيق .

لماذا يسمى هذا المفتاح ضابط أو **Constraint** ؟ في الواقع أنت لا تستطيع إدخال بيانات في الجدول الفرعي لأي موظف لا يوجد له بيانات في الجدول الرئيسي ، يجب إضافة بيانات الموظف كاملة في الجدول الرئيسي أولا ثم إضافة بيانات المرتب في الجدول الفرعي لأن الضابط **Foreign Key Constraint** يقوم بهذه المهمة ، انظر بيانات جدول **EMP_MAIN** وهو الجدول الأب وبيانات جدول **EMP_SALARY** وهو الجدول الإبن كما هو موضح بالشكل ١٠-٢ ومنه نلاحظ أن المفتاح الأساسي أو **PRIMARY KEY** الذي يربط سجل الموظف في كلا الجدولين هو حقل **EMP_ID**

EMP_MAIN TABLE		EMP_SALARY TABLE	
EMP_ID	NAME	EMP_ID	SALARY
1001	Mohammad	1001	10000
1002	Hamed	1002	15000
1003	Yakoub	1003	20000

شكل ١٠-٢

بفرض أننا قمنا بإدخال سجل موظف جديد إلى جدول **EMP_SALARY** هكذا :

```
Insert into emp_salary
Values(1004, 25000);
```

في هذا المثال عندما نقوم بإضافة بيانات موظف في جدول المرتبات ولم تجد قاعدة البيانات رقم الموظف في الجدول الأب فستعترض ولن تقبل لأنه حدث عدم تطابق مع الرقم الموجود في الجدول الأب (راجع أمر **INSERT** في الفصل التالي) وسوف تحصل على الرسالة التالية :

NOT MATCH

***استخدام معيار* Check**

يستخدم هذا الضابط لوضع شرط أو أكثر لحقل أو أكثر من حقول أي جدول ، فمثلا نريد أن نتأكد أن مرتب الموظف لا يقل عن ٥٠٠ جنيه كحد أدنى ، هنا نستخدم **Check Constraint** لوضع هذا الشرط كما في المثال التالي :

```
CREATE TABLE EMP_SALARY (
EMP_ID      CHAR(9)          NOT NULL,
TITLE       VARCHAR2(15)     NOT NULL,
SALARY      INT(6)           CHECK ( SALARY >= 500) );
FOREIGN KEY (EMP_ID) REFERENCES EMP_MAIN (EMP_ID),
```

في هذا المثال قمنا بوضع شرط ألا يقل مرتب أي موظف عن ٥٠٠ جنيه ، هذا بالإضافة طبعا الى القيد السابق والخاص بالمفتاح الأجنبي **Foreign Key** والذي سبق الحديث عنه في البند السابق. يمكنك استخدام **Check Constraints** في وضع أي شروط كما في المثال السابق.

الاستخدام الأمثل للضوابط Managing Constraints

هناك العديد من التساؤلات التي يجب أن تجيب عليها وأنت تقوم بتصميم قاعدة بيانات ومن ثم الجداول خاصة عندما تستخدم بعض الضوابط **Constraints** من هذه التساؤلات :

- هل تفهم جيدا العلاقة بين الجداول المختلفة داخل قاعدة البيانات ؟
- كيف يعتمد كل جدول على بيانات الجداول الأخرى ؟
- علاقة كل حقل في جدول بالحقول في الجداول الأخرى ؟
- هل بيانات الحقل اختيارية بحيث يمكن ترك الحقل بدون بيانات أحيانا **NULL** ؟أم أن بيانات الحقل لازمة وضرورية ولا يمكن ترك الحقل بدون بيانات **NOT NULL**؟
- كيف سيتم التعامل مع الجداول من التطبيقات المختلفة ؟

أيضا يجب أن تطلع على المعلومات المتوفرة داخل **System Catalog** حول قاعدة البيانات المستخدمة.

استخدام الترتيب الصحيح في التعامل مع الجداول

الموضوع ببساطة يحتاج إلى ترتيب الأفكار أولا، فمثلا عندما تريد إضافة بيان في جدول الابن يجب إضافة البيان المقابل في جدول الأب أولا والعكس صحيح عندما تريد حذف موظف من جدول الموظفين يجب حذفه أولا من جدول المرتبات لماذا ؟ لأن هناك علاقة تربط بين الجدولين مبنية على رقم الموظف هنا يجب أن نبدأ من حيث انتهينا .

أيضا عندما تريد أن تلغي استخدام **Primary Key** من جدول الموظفين يجب أن تلغي **Foreign Key** أولا وبذلك تنفصم العلاقة بين الجدولين بعدها يمكن أن تفعل ما تريد.

هذا ينبطبق على مناح كثيرة في الحياة، فمثلا عندما تريد أن تبني عمارة فانك تبدأ من الأسفل الى الأعلى، ولكنك عندما تريد أن تشرع في هدم عمارة فانك تبدأ من فوق السطح "وانت نازل" وهكذا فان الترتيب على جانب كبير من الأهمية .

فعندما تريد أن تضع **Foreign Key** فيجب أن تحدد **Primary Key** في الجدول

الرئيسي أولا ثم تحدد Foreign Key بعدها !

وللمزيد من الايضاح ، انظر الشكل ١٠-٣

EMP_MAIN TABLE		EMP_SALARY TABLE	
EMP_ID	NAME	EMP_ID	SALARY
1001	Mohammad	1001	10000
1002	Hamed	1002	15000
1003	Yakoub	1003	20000

عندما ترغب في حذف الموظف رقم ١٠٠٣ فانك تبدأ بحذفه من جدول المرتبات أولا

```
delete from emp_salary where emp_id = 1003
```

ثم بعد ذلك يمكنك حذف الموظف من جدول الموظفين

```
delete from emp_main where emp_id = 1003
```

شكل ١٠-٣ الترتيب في التعامل مع الجداول