الفصل الثاني عشر

إضافة محددات تضمن سلامة البيانات

Data Integrity

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

بالانتهاء من هذا الفصل ستكتسب المعارف وتتدرب على المهارات التى تجعلك قادرا على:

- *مقدمة عن الضوابط* (Constraints) *ولماذا نستخدمها*
- *استخدام المعيار* NOT NULL
- *استخدام المفتاح الأساسي* (Primary Key)
- *منع التكرارات* Unique
- *استخدام مفتاح أجنبي* Foreign Key
- *استخدام معيار التدقيق* Check
- *الاستخدام الأمثل للضوابط* Managing Constraints

مقدمة عن الضوابط (Constraints) ولماذا نستخدمها

هناك عدد من الضوابط Constraints التي يمكن استخدامها للتأكد من أن البيانات تم ادخالها بشكل صحيح لضمان التعامل معها بدون أخطاء ، فمثلا عندما تضع برنامج لادخال البيانات في جدول معين فانك تريد أن يتم ادخال البيانات في بعض الحقول دون تركها فارغة NOT NULL بينما ادخال 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 ادخال 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 اضافته بعد تكوين الجداول ، والهدف هو المحافظة على سلامة البيانات ودقتها **Integrity** وذلك قبل ادخالها أو أثناء ادخالها في قواعد البيانات ، وهي تساعدك أثناء تصميم قاعدة البيانات والجداول في التأكد من أن موظف ادخال البيانات لن يستطيع ادخال البيانات بطريقة يصعب التعامل معها ، فمثلا لو هناك رقم التليفون كحقل داخل جدول كيف سيقوم موظف ادخال البيانات بادخال رقم التليفون في الحقل؟ مثلا يقوم موظف بإدخال الرقم هكذا :

٠٠٢٠٢١٢٣٤٥٦٧ بينما يقوم موظف آخر بكتابته هكذا **+202-1234567** وموظف ثالث يمكن أن يكتبه هكذا **(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 و استخدام مجموعة من الضوابط داخل جداول قاعدة البيانات :

- غير خالي من البيانات **NOT NULL**
- المفتاح الأساسي**Primary Key**
- المفتاح الخارجي **Foreign Key**
- منع التكرارات **Unique**
- التدقيق **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 ادخال 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 بانها NOT NULL من البداية وبالتالي فان البرنامج الذي كنت أتعامل معه يتأكد ايضا من ضرورة ادخال 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 بانشاء جدول بصورتة العادية التي تعلمناها ، لاحظ أن Oracle SQL يستخدم varchar2 حتى الآن بينما MySQL يستخدم varchar :

```
SQL
CREATE TABLE EMP_MAIN (
```

```
EMP_ID      VARCHAR2(8),
NAME1       VARCHAR2(10),
NAME2       VARCHAR2(10),
NAME_L      VARCHAR2(10),
ADDRESS     VARCHAR2(30),
CITY        VARCHAR2(15),
GOV         VARCHAR2(15),
ZIP         VARCHAR2(5),
TEL         VARCHAR2(9));
```
TABLE CREATED

MySQL
```
>create table emp_main (
>emp_id      varchar(8),
>name1       varchar(10),
>name2       varchar(10),
>name_l      varchar(10),
>address     varchar(30),
>city        varchar(15),
>gov         varchar(15),
>zip         varchar(5),
>tel         varchar(9));
```
query ok, 0 rows affected (0.08)

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**

سوف نقوم بوضع **NOT NULL** للحقول التي يجب أن تحتوي على قيمة كما يمكننا أن نضع **NULL** للحقول التي ليس مهما أن تكون لها قيمة ان لم تدخل بها أي بيانات ، إذا تجاهلت وضع **NULL** أو **NOT NULL** فان نظام قواعد البيانات سوف يضع **NULL** من عنده أمام تلك الحقول . سنقوم الآن بوضع **NOT NULL** أمام بعض الحقول :

```
SQL
CREATE TABLE EMP_MAIN (
EMP_ID        VARCHAR2(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           VARCHAR2(5)    NOT NULL,
TEL           VARCHAR2(9));
TABLE CREATED
```

وبنفس الطريقة يتم التعامل مع MYSQL

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 القيمةNULL NOT 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 :

```
SQL
CREATE TABLE EMP_MAIN (
EMP_ID        CHAR(8)    NOT NULL  PRIMARY KEY,
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));
TABLE CREATED
```

وبنفس الطريقة يتم تعريف المفتاح الأساسي مع MYSQL

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 .

هناك طريقة أخرى لتخصيص أحد الحقول ليكون المفتاح الأساسي Primary Key وسوف نلاحظ ذلك في المثال التالي :

```
SQL
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));
TABLE CREATED
```

وبنفس الطريقة تستخدم مع MYSQL

عندما تقوم بتحديد نوع البيانات في نظام MySQL على أنه رقمي Numeric سوف يحجز لك عدد من الخانات الصحيحة وعلامة عشرية واحدة ، طبيعي في حالة ZIP, TEL فأنت لا تحتاج إلى علامة عشرية لذا فمن الأفضل أن تحدد النوع كالتالي

```
>zip int(5),
>tel int(9),  أو  tel intger(9),
```

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) .

```
SQL
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) UNIQUE,
PRIMARY KEY (EMP_ID));
TABLE CREATED
```

وبنفس الطريقة نكتب UNIQUE مع MYSQL

بهذا الضابط تأكد لنا أن رقم التليفون لن يتكرر مع أكثر من موظف ، أيضا استخدمنا مفتاح Primary Key مع حقل 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 :

SQL

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

MySQL

```
create table emp_salary (
emp_id              char(9)              not null,
title          varchar(15)          not null,
salary         numeric(6)           not null,
foreign key    emp_id_fk (emp_id) references emp_main (emp_id));
query ok, 0 rows affected (0.14)
```

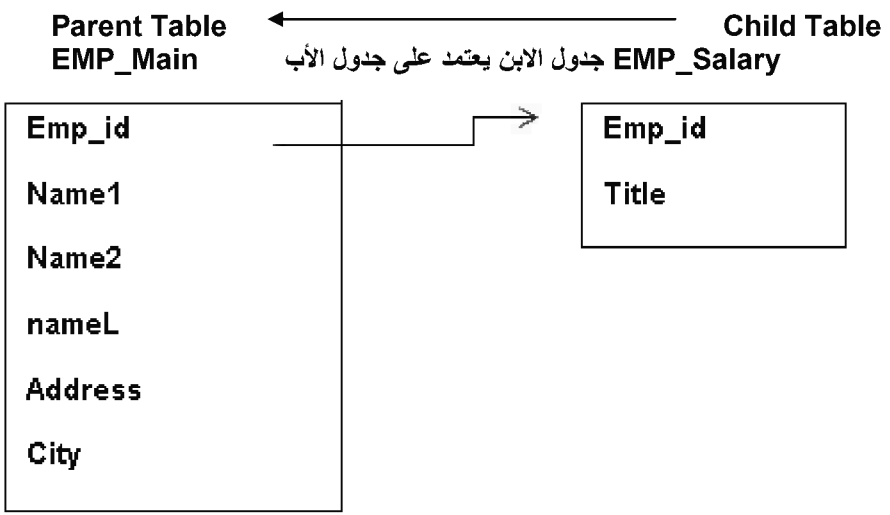
لاحظ أننا قمنا بتكوين جدول فرعي (الابن) **emp_salary** يحتوي على رقم الموظف ووظيفته والمرتب ولكن باقي بيانات الموظف موجودة في الجدول الرئيسي (الأب) **emp_main** العلاقة التي تربط بين الجدولين هي أن مفتاح الجدول الفرعي **Foreign Key** هو عبارة عن اشارة **references** الى المفتاح الرئيسي **Primary Key** 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 .

دائما تأكد من نظام قواعد البيانات الذي تستخدمه ، حيث أن هناك عدد قليل من الاختلافات التي يمكن أن تجعل **query** لا تعمل مثل **varchar2, number** التي لا تستخدم في نظام **MySQL** على سبيل المثال ويستخدم بدلا منها **varchar, numeric** .

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 ١١-١



شكل ١١-١ 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 . افرض أننا أنشأنا الجدول الابن دون

استخدام مفتاح خارجي . وتذكرنا الآن أننا يجب أن ننشئ علاقة بين الجدول الابنEMP_SALARY والجدول الأبEMP_MAIN ، هل يمكن اضافة Foreign Key بعد أن قمنا بتكوين الجدولين ؟ جرب هذه Query :

```
ALTER TABLE EMP_SALARY
ADD CONSTRAINT EMP_ID_FK FOREIGN KEY (EMP_ID)
REFERENCES EMP_MAIN (EMP_ID);
```

Table Altered

أصبح لدينا مفتاح جديد يربط رقم الموظف emp_id في كلا الجدولين هذا المفتاح هو Foreign Key وقد أعطيناه الاسم EMP_ID_FK وهو مرتبط بالجدول "الابن" وسوف يستخدم للحصول على بيانات من الجدول الرئيسي "الأب" .

لماذا يسمى هذا المفتاح ضابط أو Constraint ؟ في الواقع أنت لا تستطيع ادخال بيانات في الجدول الفرعي لأي موظف لا يوجد له بيانات في الجدول الرئيسي ، يجب اضافة بيانات الموظف كاملة في الجدول الرئيسي أولا ثم اضافة 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 ١١-٢ ومنه نلاحظ أن المفتاح الأساسي أو PRIMARY KEY الذي يربط سجل الموظف في كلا الجدولين هو حقل EMP_ID

EMP_MAIN TABLE

EMP_ID	NAME
1001	Mohammad
1002	Hamed
1003	Yakoub

EMP_SALARY TABLE

EMP_ID	SALARY
1001	10000
1002	15000
1003	20000

شكل ١١-٢

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
```

كما سبق القول تختلف نظم قواعد البيانات في طريقة كتابة Query لذلك عندما تشرع في كتابة Query تأكد من طريقة صياغة نصوص " Query Syntax " في نظام إدارة قواعد البيانات " RDBMS " الذي تستخدمه !!!

استخدام معيار التدقيق 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 :

SQL

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

MySQL

```
create table emp_salary (
emp_id              char(9)             not null,
title           varchar(15)         not null,
salary          numeric(6)          not null,
foreign key     emp_id_fk (emp_id) references emp_main (emp_id),
constraint chk_sal check ( salary >= 500) );
query ok, 0 rows affected (0.14)
```

في هذا المثال قمنا بوضع شرط ألا يقل مرتب أي موظف عن ٥٠٠ جنيه ، هذا بالاضافة 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 الآخرى ؟
- 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 الى ترتيب الأفكار أولا ، فمثلا عندما تريد اضافة بيان في جدول الابن يجب اضافة 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 ينطبق 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 ١١-٣

EMP_MAIN TABLE

EMP_ID	NAME
1001	Mohammad
1002	Hamed
1003	Yakoub

EMP_SALARY TABLE

EMP_ID	SALARY
1001	10000
1002	15000
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 ١١-٣ الترتيب في التعامل مع الجداول

طرق إضافة الضوابط

كما تعلمنا في هذا الفصل فاننا نقوم بتكوين الضوابط (Constraints) بأحد طريقتين :

- قم بإضافة ضوابط Constraint داخل Query التي تستخدمها لتكوين الجدول ، وهكذا فانك في مرحلة التصميم تكون الصورة العامة والعلاقات واضحة أمامك وفي ذهنك .
- أو قم بإضافة ضوابط Constraints لجداول موجودة فعلا وتم تكوينها دون اضافة Constraints لها ، وظهرت الحاجة الى ذلك خاصة بعد مرحلة ادخال بيانات وصعوبة اعادة التصميم من جديد ، في هذه الحالية يمكنك استخدام أمر Alter Table