

# الأكاديمية العربية الدولية

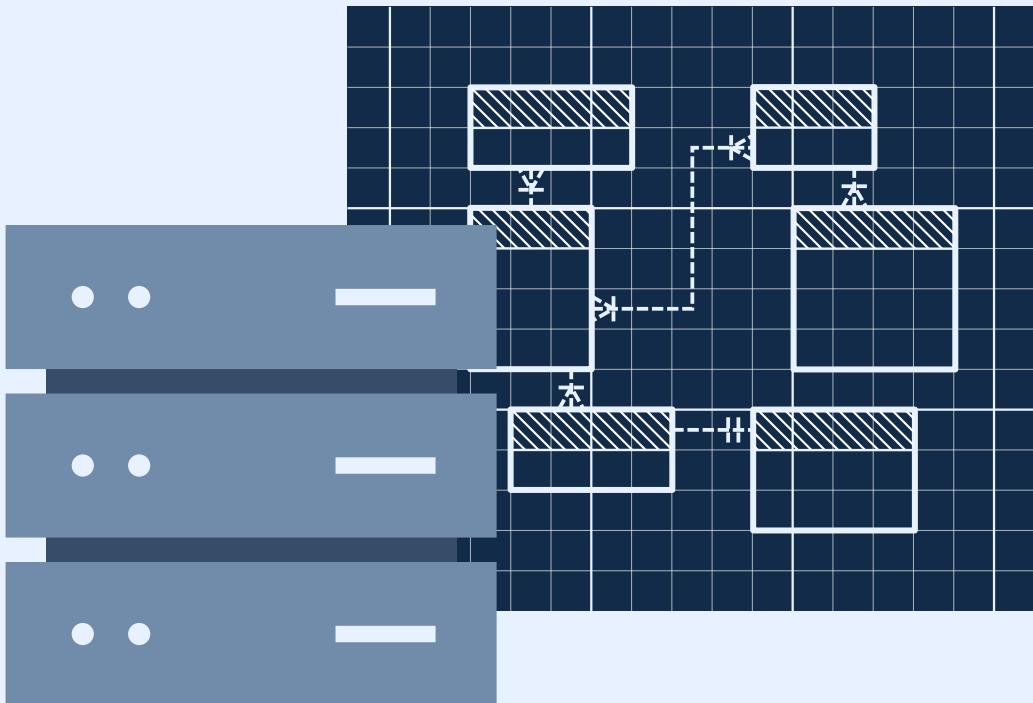


الأكاديمية العربية الدولية  
Arab International Academy

---

## الأكاديمية العربية الدولية المقررات الجامعية

---



# تمهیم ټوابع البيانات

---

إیدرین وات  
نیلسون انج

# تصميم قواعد البيانات

مراجع سريع إلى عملية تصميم قواعد البيانات وتطبيقاتها

تأليف

إيدريين وات

نيلسون إنج

ترجمة

أيمن طارق القاضي

علا عباس

تحرير

آيات البيطقالان

إعداد وإشراف

جميل بيلوني

جميع الحقوق محفوظة © 2022 أكاديمية حسوب  
النسخة الأولى v1.0

هذا العمل مرخص بعوجب رخصة المشاع الإبداعي: تُسبِّب المُصنَّف - غير تجاري - الترخيص  
بالمثل 4.0 دولي



# عن الناشر

أُنْتَجَ هَذَا الْكِتَاب بِرِعايَةِ شَرْكَةِ حَسُوبِ وَأَكَادِيمِيَّةِ حَسُوبِ.



تهدف أكاديمية حسوب إلى توفير دروس وكتب عالية الجودة في مختلف المجالات وتقدم دورات شاملة لتعلم البرمجة بأحدث تقنياتها معتمدةً على التطبيق العملي الذي يُؤهِّل الطالب لدخول سوق العمل بثقة.



حسوب مجموعة تقنية في مهمة لتطوير العالم العربي. تبني حسوب منتجات ترتكز على تحسين مستقبل العمل، والتعليم، والتواصل. تدير حسوب أكبر منصتي عمل حر في العالم العربي، مستقل وخمسات ويعمل في فيها فريق شاب وشغوف من مختلف الدول العربية.

# جدول المحتويات

9	تمهيد
11	<b>1. ما قبل ظهور أنظمة قواعد البيانات</b>
11	1.1 النظام القائم على الملفات
12	1.2 عيوب النظام القائم على الملفات
13	1.3 نظام قواعد البيانات
14	1.4 معنى البيانات
15	1.5 مصطلحات أساسية
15	1.6 تمارين
17	<b>2. مفاهيم قواعد البيانات الأساسية</b>
17	2.1 ما هي قاعدة البيانات؟
18	2.2 خصائص قاعدة البيانات
19	2.3 أنواع مستخدمي قاعدة البيانات
20	2.4 نظام إدارة قواعد البيانات وتصنيفاتها
24	2.5 مصطلحات أساسية
25	2.6 تمارين
26	<b>3. خصائص قواعد البيانات والمزايا التي تقدمها</b>
27	3.1 خصائص قواعد البيانات
28	3.2 دعم عدة واجهات عرض للبيانات
30	3.3 مصطلحات أساسية
30	3.4 تمارين
31	<b>4. نمذجة البيانات وأنواعها</b>
32	4.1 أنواع نماذج البيانات
34	4.2 مدى تجريد البيانات
36	4.3 طبقات تجريد البيانات
37	4.4 تخطيطات قاعدة البيانات <i>Schemas</i>
38	4.5 استقلالية البيانات المنطقية والمادية
38	4.6 مصطلحات أساسية

40	التمارين	4.7
<b>41</b>	<b>RDM</b>	<b>5. نموذج البيانات العلاقة</b>
41	المفاهيم الأساسية في نماذج البيانات العلاقة	5.1
45	خصائص الجدول	5.2
45	المفاهيم الأساسية	5.3
46	المصطلحات الأساسية	5.4
46	تمارين	5.5
<b>47</b>	<b>نماذج الكيان وال العلاقة ER و تمثيل البيانات</b>	<b>6. نماذج الكيان وال العلاقة ER و تمثيل البيانات</b>
48	الكيان ومجموعة الكيان ونوع الكيان	6.1
49	ارتباط الوجود	6.2
49	أنواع الكيانات	6.3
51	السمات	6.4
54	المفاتيح	6.5
61	مصطلحات أساسية	6.6
63	تمارين	6.7
<b>68</b>	<b>قواعد السلامة وقيودها لضمان سلامة البيانات</b>	<b>7. قواعد السلامة وقيودها لضمان سلامة البيانات</b>
68	سلامة النطاق Domain Integrity	7.1
71	قيود المؤسسة Enterprise Constraints	7.2
72	قواعد العمل Business Rules	7.3
73	أنواع العلاقات	7.4
78	مصطلحات أساسية	7.5
79	تمارين	7.6
<b>82</b>	<b>نماذج الكيان العلقي ER عند تصميم قواعد البيانات</b>	<b>8. نماذج الكيان العلقي ER عند تصميم قواعد البيانات</b>
82	التصميم العلقي Relational Design والتكرار Redundancy	8.1
83	حالة الإدخال الشاذة Insertion Anomaly	8.2
84	حالة التحديث الشاذة Update Anomaly	8.3
84	حالة الحذف الشاذة Deletion Anomaly	8.4
86	كيفية تجنب الحالات الشاذة	8.5
87	مصطلحات أساسية	8.6

87	8.7 تمارين
89	<b>9. الاعتمادات الوظيفية Functional Dependencies</b>
90	9.1 قواعد الاعتمادات الوظيفية
91	9.2 قواعد الاستدلال Inference Rules
94	9.3 مخطط الاعتمادية Dependency Diagram
95	9.4 مصطلحات أساسية
96	<b>10. فهم عملية التوحيد Normalization</b>
96	10.1 ما هو التوحيد؟
97	10.2 النماذج الموحدة Normal Forms
97	10.3 النموذج الموحد الأول أو 1NF First Normal Form
98	10.4 النموذج الموحد الثاني أو 2NF Second Normal Form
99	10.5 النموذج الموحد الثالث أو 3NF Third Normal Form
101	10.6 نموذج بويس-كود الموحد BCNF
103	10.7 التوحيد وتصميم قواعد البيانات
104	10.8 المصطلحات الأساسية والاختصارات
104	10.9 تمارين
107	<b>11. عملية تطوير قواعد البيانات</b>
107	11.1 دورة حياة تطوير البرمجيات - نموذج الشلال Waterfall
109	11.2 دورة حياة قاعدة البيانات Database Life Cycle
111	11.3 جمع المتطلبات Requirements Gathering
111	11.4 التحليل Analysis
112	11.5 التصميم المنطقي Logical Design
114	11.6 التطبيق Implementation
115	11.7 تحقيق التصميم Realizing the Design
115	11.8 ملء قاعدة البيانات Populating the Database
116	11.9 إرشادات لتطوير مخطط ER
116	11.10 المصطلحات الأساسية
117	11.11 تمارين

118	<b>12. لغة الاستعلامات الهيكلية SQL</b>	
119	إنشاء قاعدة بيانات	12.1
123	قيود الجدول	12.2
128	الأنواع التي يُعرفها المستخدم	12.3
129	مصطلحات أساسية	12.4
129	تمارين	12.5
131	<b>13. لغة معالجة البيانات DML الخاصة بلغة SQL</b>	
132	SELECT	13.1
138	INSERT	13.2
141	UPDATE	13.3
141	DELETE	13.4
142	الدوال المبنية مسبقا	13.5
146	ضم الجداول	13.6
149	مصطلحات أساسية	13.7
150	تمارين	13.8
154	<b>14. الملحق أ: مثال عملي عن تصميم قاعدة بيانات لجامعة</b>	
155	عملية التصميم	14.1
159	<b>15. الملحق ب: أمثلة عملية عن إنشاء مخططات ERD</b>	
159	التمرين الأول: شركة تصنيع	15.1
160	Car Dealership	15.2
162	<b>16. الملحق ج: حل تمرين باستخدام لغة SQL</b>	
162	الجزء الأول: استخدم لغة DDL	16.1
165	الجزء الثاني: إنشاء عبارات لغة SQL	16.2
167	الجزء الثالث: الإدخال والتعديل والحذف والفالهارس	16.3

# تمهيد

تسبق كل عملية تنفيذية عملية تصميمية تخطيطية مثل عملية تخطيط أي بناء دراسته وتحليله ورسمه على المخططات قبل البدء ببنائه وتنفيذها على الأرض وكذلك الحال مع قواعد البيانات فقبل تنفيذها برمجياً، يكون هنالك مرحلة تصميمية يجري فيها تصميم قواعد البيانات وتخطيط جداولها والأعمدة التي تحتويها وأنواع البيانات فيها والروابط الرابطة بينها والقيود المقيدة لها وهكذا، فلعملية التصميم تلك أهمية كبيرة لأي قاعدة بيانات، ولا يكاد يخلو أي نظام برمجي يتعامل مع البيانات من قاعدة بيانات لذا لابد من إجراء تصميم متين مناسب لقاعدة البيانات التي سيبني عليها النظام.

يعد موضوع تصميم قواعد البيانات مادة أساسية تدرّس في السنوات الأولى من تخصص علوم الحاسوب والتخصصات المتعلقة بـهندسة الحاسوب والبرمجيات في الجامعة ولابد من تعلم الموضوع دراسته قبل البدء بتعلم كيفية تنفيذ قاعدة بيانات برمجياً، فلا يمكن بناء قاعدة بيانات قوية متينة الارتباطات سليمة القيود موحدة عديمة التكرارات دون دراسة موضوع التصميم نظرياً وإنقاذه عملياً وذلك بتصميم قواعد بيانات نموذجية والإطلاع على تصميمات أخرى لقواعد بيانات صممها مهندسون أكفاء.

يشرح هذا الكتاب عملية تصميم قاعدة بيانات شرحاً موجزاً غير مخل، إذ يبدأ أولاً بتغطية كل المفاهيم المتعلقة بقواعد البيانات بدءاً من فكرة قاعدة البيانات بالأصل واختلافها عن أي نظام تخزين بيانات، وحتى العملية التصميمية وعمليات نمذجة البيانات وتمثيلها وعمليات الربط بين جداول البيانات وما يقوم عليها ثم يبني بعدها على ذلك كله شارحاً عملية تصميم قواعد البيانات وتطويرها حتى رسم مخطط ER النهائي لجدول قاعدة البيانات، وهو مخطط نموذج الكيان والعلاقة الواصف لقاعدة البيانات وجداولها وكل تفصيلة فيها، ثم يشرح بعدها كيفية تنفيذ المخطط الناتج عبر لغة SQL ولا يغيب عنه التطرق إلى أنظمة إدارة قواعد البيانات DBMS مع ذكر أشهرها.

هذا الكتاب مترجم عن كتاب "تصميم قاعدة بيانات - الإصدار الثاني" Database Design - 2nd Edition لصاحبيه إيدرين وات Adrienne Watt ونيلسون إنج Eng Nelson وهو مبني في الأصل على عدة مصادر ركيزتها كتاب Database Design للمؤلف الأول إيدرين وات، فالمصادر المبني عليها كل فصل مدرجة في نهايتها ويمكنك الرجوع إلى الكتاب الأجنبي الأصل إن أردت الإطلاع عليها، كما أن هذا الكتاب الأجنبي يعتمد في بعض الجامعات لتدريس مادة تصميم قواعد البيانات في فصول علوم الحاسوب وتصميم البرمجيات.

انتبه إلى أن هذا الكتاب لا يركز على عملية تفزيذ قواعد البيانات وبرمجتها بل يركز على عملية تصميم قاعدة بيانات وكيفية نمذجتها وإن كان يتطرق إلى لغة SQL سريعاً شارحاً باقتضاب كيفية إنشاء قاعدة بيانات والعمليات الأساسية للتعديل عليها، أما إذا أردت الاستزادة والتعمق في هذا الموضوع، فننصحك بالرجوع إلى كتاب ملاحظات للعاملين بلغة SQL وكتاب الدليل العملي إلى قواعد بيانات PostgreSQL وتوثيق لغة SQL من موسوعة حسوب وأي كتب ومصادر أخرى تصب في هذا المجال.

تجد في نهاية كل فصل من فصول الكتاب قائمة بالمصطلحات الأساسية التي ناقشها الفصل وتحدد عنها بالعربية وما يقابلها بالإنجليزية ليسهل عليك البحث عنها أو الرجوع إلى أي مصادر أجنبية كما تجد في نهايته تمارين تساعدك على ترسیخ ما تعلمت، لذا ننصحك بالاهتمام بها لتحقيق أقصى استفادة مما تعلمت من الفصل، وإن احتجت إلى أي مساعدة، فلا تتردد بطرح سؤالك في قسم **الأسئلة والأجوبة** في أكاديمية حسوب أو في مجتمع حسوب 10.

إذا كان لديك اقتراح أو تصحيح على النسخة العربية من الكتاب أو أي ملاحظة حول أي مصطلح من المصطلحات المستعملة، يرجى إرسال بريد إلكتروني إلى [academy@hsoub.com](mailto:academy@hsoub.com). إذا ضمّنت جزءاً من الجملة التي يظهر الخطأ فيها على الأقل، فهذا يسهل علينا البحث، وحباً إضافة أرقام الصفحات والأقسام.

جميل بيلوني

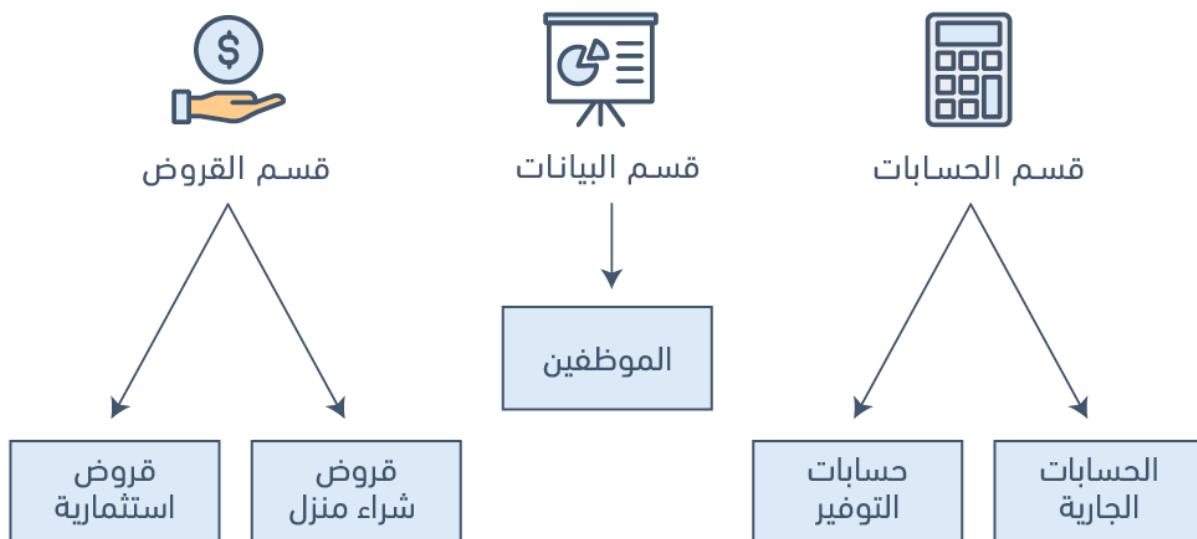
2022/02/20

# 1. ما قبل ظهور أنظمة قواعد البيانات

قطعت الطريقة التي تدير بها أجهزة الحاسوب البيانات شوًطاً طويلاً على مدار العقود القليلة الماضية، كما يأخذ مستخدمي الحاسوب المزايا العديدة الموجودة في نظام قواعد البيانات أمراً مسلّماً به على الرغم من عدم مرور وقت طويل على اعتماد أجهزة الحاسوب النظام القائم على الملفات File-based System والذي يُعدّ النهج الأقل في الأنقة والتكلفة لإدارة البيانات.

## 1.1 النظام القائم على الملفات

يُعدّ تخزين المعلومات في ملفات دائمة في الحاسوب إحدى طرق الحفاظ عليها، فمثلاً، يملك نظام الشركة عدداً من البرامج التطبيقية والمصممة لمعالجة ملفات البيانات تلك، ثم توليد المخرجات والنتائج في ملفات أخرى، حيث تُصمّم هذه البرامج بناءً على طلب المستخدمين في المؤسسة، كما تضاف برامج جديدة إلى النظام عند الحاجة إليها، ويسمى هذا النظام بالنظام القائم على الملفات File-based System، فمثلاً، يمكن استخدام النظام القائم على الملفات لإدارة بيانات نظام مصري تقليدي كما هو موضح في الشكل أدناه، حيث يوجد أقسام مختلفة في المصرف، ولكل منها برامج خاصة لإدارة ومعالجة ملفات البيانات المختلفة، كما يمكن استخدام برامج لأداء عمليات عديدة في الأنظمة المصرفية، مثل: الخصم من الحساب أو ائتمانه، وإنشاء كشف برصيد الحساب، وإضافة قروض عقارية جديدة، وإنشاء كشوف حسابات شهرية.



الشكل 1.1: مثال على نظام قائم على الملفات لبنك يستعمله لإدارة البيانات

## 1.2 عيوب النظام القائم على الملفات

يملك النظام القائم على الملفات والمستخدم لحفظ المعلومات التنظيمية العديد من العيوب، والتي دفعت فيما بعد لتطوير أنظمة جديدة أكثر كفاءة، نذكر منها التالي:

### 1.2.1 تكرار البيانات Data redundancy

غالبًا ما تنشأ ملفات البيانات الخاصة بالمؤسسة من طرف العديد من المبرمجين من أقسام مختلفة على مدى فترات طويلة من الزمن، مما يؤدي إلى تكرار البيانات عند تحديث أحد الحقول في أكثر من موضع واحد، حيث تسبب هذه العملية العديد من المشاكل، منها:

- عدم توحيد تنسيق البيانات.
- الاحتفاظ بالمعلومة نفسها في عدة أماكن مختلفة، أي ضمن ملفات مختلفة.
- عدم تناسق البيانات، وهو الموقف الذي تتعارض فيه النسخ المختلفة من البيانات نفسها، مما يهدى مساحة التخزين ويضاعف الجهد.

### 1.2.2 عزل البيانات

عزل البيانات هو الخاصية التي تحدد متى وكيف تصبح التغييرات التي يتم إجراؤها بواسطة عملية معينة مرئيةً للمستخدمين المتزامنين، والأنظمة المتزامنة الأخرى؛ ويؤدي حدوث أي مشكلة في مزامنة البيانات إلى صعوبة استرجاع البيانات المناسبة من قبل التطبيقات الأخرى التي تصل لنفس البيانات والتي ربما تكون مخزنة في عدة ملفات مختلفة.

### 1.2.3 مشاكل السلامة

تُعد مشاكل سلامة البيانات عيّناً آخرًا لاستخدام النظام القائم على الملفات، حيث تشير سلامة البيانات إلى صيانة البيانات، والتأكد من صحة وتناسق البيانات الموجودة في قاعدة البيانات، حيث يجب مراعاة العوامل التالية أثناء معالجة هذه المشكلة:

- يجب على قيم البيانات استيفاء قيود تناسق معينة ومحدّدة في برامج التطبيق.
- من الصعب إجراء تغييرات على برامج التطبيق من أجل فرض قيود جديدة.

### 1.2.4 مشاكل الأمان

يُعدّ الأمان مشكلةً في النظام القائم على الملفات للأسباب التالية:

- وجود قيود تتعلق بالحصول على الصلاحيات.
- تضاف متطلبات التطبيق إلى النظام بطرق مخصصة، لذلك من الصعب فرض القيود.

### 1.2.5 الوصول المتزامن

التزامن هو قدرة قاعدة البيانات على السماح لعدة مستخدمين بالوصول للسجل نفسه دون التأثير سلباً على معالجة المعاملات. يجب على النظام القائم على الملفات إدارة التزامن وضبطه باستخدام برمجية تطبيقية، حيث يُقفل الملف ويُمنع الوصول إليه عندما يفتحه تطبيق ما، مما يعني أنه لا يمكن لأي شخص آخر الوصول إلى ذاك الملف في الوقت نفسه آنذاك.

تُدير أنظمة قواعد البيانات عملية التزامن من خلال السماح لعدة مستخدمين من الوصول إلى السجل نفسه، ويعُدّ هذا فرق مهم بين قواعد البيانات وأنظمة القائمة على الملفات.

## 1.3 نظام قواعد البيانات

دفعت الصعوبات الناشئة عن استخدام النظام القائم على الملفات إلى تطوير نظام جديد لإدارة الكميات الكبيرة من المعلومات التنظيمية، والذي يُسمى بنظام قاعدة البيانات.

تلعب قواعد البيانات وتقنيتها دوراً مهماً في معظم المجالات التي تُستخدم فيها أجهزة الحاسوب، بما في ذلك الأعمال التجارية، والتعليم، والطب، وغيرها، وسنبدأ في هذا الفصل بتقديم بعض المفاهيم الأساسية المتعلقة بهذا المجال لفهم أساسيات أنظمة قواعد البيانات.

### 1.3.1 دور قواعد البيانات في إدارة الأعمال

يستخدم الجميع قواعد البيانات بطريقة أو بأخرى، حتى ولو كانت استخدامات بسيطة مثل تخزين معلومات عن أصدقائهم وعائلاتهم فقط، كما يمكن تدوين هذه البيانات أو تخزينها على جهاز حاسوب باستخدام برنامج

معالجة النصوص، أو يمكن حفظها على صورة جداول، ومع ذلك فإن أفضل طريقة لتخزين البيانات هي استخدام برامج إدارة قواعد البيانات، وهي أدوات برمجية قوية تسمح لك ب تخزين البيانات، ومعالجتها، واسترجاعها بعدة طرق مختلفة.

تتبع معظم الشركات معلومات العملاء من خلال تخزينها في قاعدة بيانات، وقد تشمل هذه البيانات العملاء، أو الموظفين، أو المنتجات، أو الطلبات، أو أي شيء آخر يساعد الشركة في تنفيذ مهامها.

## 1.4 معنى البيانات

تُعدّ البيانات Data معلومات واقعية، مثل: القياسات، أو الإحصائيات حول الأشياء والمفاهيم، كما تُستخدم للمناقشات، أو على أساس جزء من العمليات الحسابية، و يمكن أن تكون هذه البيانات شخصاً، أو مكاناً، أو حدثاً، أو إجراءً، أو أي شيء آخر، حيث تمثل كل معلومة أو حقيقة معينة عنصراً من عناصر البيانات .data element أي

إذا كانت البيانات معلومات، وكانت المعلومات هي ما نعتمد عليه في العمل، فيمكنك البدء في معرفة المكان الذي قد تخزن هذه البيانات فيه، فمثلاً، يمكن تخزين البيانات في:

- ملفات تخزين مخصصة
- جداول البيانات
- المجلدات
- الدفاتر
- القوائم
- الأوراق

تخزن كل هذه المواد المعلومات وكذلك قاعدة البيانات.

بسبب الطبيعة الميكانيكية لقواعد البيانات، نجد أن لها مقدرة كبيرة على إدارة و معالجة المعلومات المخزنة فيها، مما يجعل هذه المعلومات أكثر فائدة لعملك.

يمكننا البدء من خلال هذا الفهم للبيانات في رؤية كيف يمكن لقاعدة البيانات تخزين مجموعة من البيانات، وتنظيمها، وإجراء بحث سريع عليها، واسترجاعها، ومعالجتها. ويتضمن هذا الكتاب والفصول التالية الكثير من التفاصيل عن أنظمة إدارة قواعد البيانات وكيفية التعامل معها.

## 1.5 مصطلحات أساسية

- **التزامن Concurrency:** هو قدرة قاعدة البيانات على السماح لعدة مستخدمين من الوصول إلى السجل نفسه دون التأثير سلباً على معالجة المعاملات.
- **عنصر البيانات Data element:** حقيقة أو معلومة واحدة.
- **عدم تناقض البيانات Data inconsistency:** الحالة التي تتعارض فيها النسخ المختلفة للبيانات نفسها بعضها مع بعض.
- **عزل البيانات Data isolation:** الخاصية التي تحدد متى وكيف تصبح التغييرات التي تجري بواسطة عملية معينة مرئيةً للمستخدمين المتزامنين والأنظمة المتزامنة الأخرى.
- **سلامة البيانات Data integrity:** يشير إلى الصيانة والتأكد من أن البيانات في قاعدة البيانات صحيحة ومتّسقة.
- **تكرار البيانات Data redundancy:** حالة تحدث في قاعدة بيانات عندما يحتاج أحد الحقول إلى التحديث في أكثر من جدول.
- **نظام قاعدة البيانات Database approach:** وهو الذي يسمح بإدارة كميات كبيرة من المعلومات التنظيمية.
- **برامج إدارة قواعد البيانات Database management software:** أداة برمجية قوية تتيح لك تخزين البيانات، ومعالجتها، واسترجاعها بطرق مختلفة.
- **النظام القائم على الملفات File-based system:** وهو عبارة عن برنامج تطبيق مصمّم للتعامل مع ملفات البيانات.

## 1.6 تمارين

1. نقش كل من المصطلحات التالية:

- البيانات
- الحقل
- السجل
- الملف

2. ما هو تكرار البيانات؟

3. ناقش عيوب النظام القائم على الملفات.

4. اشرح الفرق بين البيانات والمعلومات.

5. استخدم الجدول أدناه للإجابة على الأسئلة التالية.

◦ كم عدد السجلات التي يحتوي عليها الملف؟

◦ كم عدد الحقول في كل سجل؟

◦ ما المشكلة التي قد تواجهها إذا أردت إنشاء قائمة مرتبة حسب المدينة؟

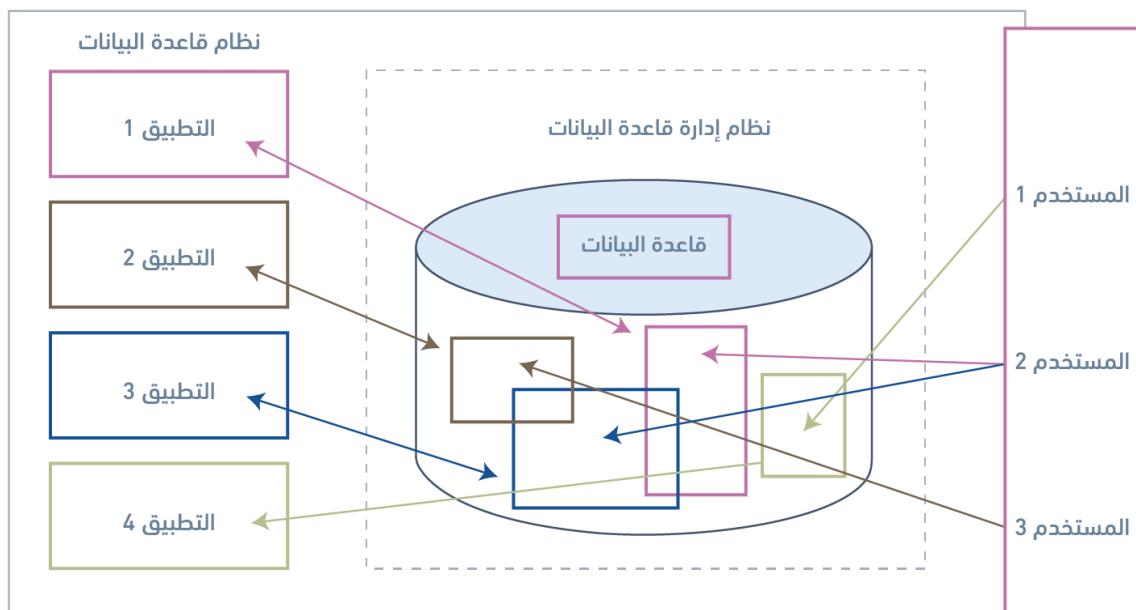
◦ كيف يمكنك حل هذه المشكلة عن طريق تعديل هيكلة الملف؟

PROJECT CODE	PROJECT MANAGER	MANAGER PHONE	MANAGER ADDRESS	PROJECT BID PRICE
21-5Z	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	\$16,833,460,00
25-2D	Jane D. Grant	615-898-9909	218 Clark Blvd., Nashville, TN 36362	\$12,500,000,00
25-5A	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	\$32,512,420,00
25-9T	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	\$21,563,234,00
27-4Q	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	\$10,314,545,00
29-2D	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	\$25,559,999,00
31-7P	0/11liam K. Moor	904-445-2719	216 Morton Rd., Stetson, FL 30155	\$56,850,000,00

## 2. مفاهيم قواعد البيانات الأساسية

ستتعرف في هذا الفصل على أهم المصطلحات والمفاهيم الأساسية في قواعد البيانات بدءاً من التعرف على مفهوم قاعد البيانات بحد ذاته ثم التعرف إلى الصفات التي تتصف فيها قواعد البيانات وأخيراً التعرف على مفهوم أنظمة إدارة قواعد البيانات وتصنيفاتها المنددرجة ضمنها.

### 2.1 ما هي قاعدة البيانات؟



الشكل 2.1: قاعدة البيانات هي مستودع للبيانات

تُعدّ قاعدة البيانات database تجميعة مشتركة من البيانات ذات الصلة، وتُستخدم لدعم أنشطة منظمة معينة، كما يمكن النظر إلى قاعدة البيانات على أساس مستودع للبيانات التي تُعرف مرةً واحدةً، ومن ثم يمكن الوصول إليها من مستخدمين مختلفين كما هو موضح في الشكل التالي.

## 2.2 خصائص قاعدة البيانات

تملك قاعدة البيانات الخصائص التالية:

- تُمثل بعض جوانب العالم الحقيقي، أو تجميعة من عناصر البيانات data elements -أو الحقائق facts- التي تُمثل معلومات مستقاة من الواقع.
- تُعدّ قاعدة البيانات منطقيةً، ومتماسكةً، ومُتسقةً داخلياً.
- صُممت قاعدة البيانات وُبُنيت وُمُلئت بالبيانات لخدمة غرض معين.
- يُخزن كل عنصر بيانات في حقل field.
- تُكُون مجموعة الحقول جدولًا table، فمثلاً، يحتوي كل حقل في جدول الموظف على بيانات حول موظف فردي.

يمكن أن تحتوي قاعدة البيانات على العديد من الجداول، فمثلاً، قد يحتوي نظام العضوية membership system على جدول عنوان، وجدول عضو فردي كما هو موضح في الشكل 2.

ت تكون منظمة عالم العلوم مثلاً من عدة أعضاء، وهم: أفراد individuals، ومنازل جماعية group homes، وأعمال تجارية businesses، وشركات corporations، حيث يملكون عضوية نشطة في هذه المنظمة، كما يمكن شراء العضوية لمدة سنة أو سنتين، وبعد ذلك يمكن تجديدها لمدة سنة أو سنتين أيضاً.

نلاحظ في الشكل أنّ ميني ماوس Minnie Mouse قد جددت عضوية العائلة في منظمة عالم العلوم Science World، كما نلاحظ أنّ كل شخص يملك المعرف رقم 100755 يعيش في العنوان التالي: Minnie Mouse 8932 Rodent Lane و Mickey Mouse، والأعضاء الأفراد كما يظهر في الشكل وهم: Moose Mouse كما هو ظاهر.

ID	100755	EXPIRY DATE	201503	Prev Exp	201402	Stat	A	Cat	FP	
Name	Mrs.	Minnie	Mouse	Res	222-2222					
Address	8982 Rodent Lane									
City	West Vancouver	Prov	BC	Country	Canada					
Notes										
Cards		2013/08/09		# Members		8	#Years		1	
	FirstName	LastName	YYMM	G	BARCODE	V	DATE	TIME	F	
*	Mickey	Mouse	0000	M	10000001	4	20130810	10:12:29	y	
	Mirrae	Mouse	0000	F	10000002	4	20130810	10:12:29	y	
	Mighty	Mouse	0000	M	10000003	4	20130810	10:12:29	y	
	Door	Mouse	0000	F	10000004	4	20130810	10:12:29	y	
	Tom	Mouse	0000	M	10000005	4	20130810	10:12:29	y	
	King	Rat	0000	M	10000006	4	20130810	10:12:29	y	
	Man	Mouse	0000	M	10000007	4	20130810	10:12:29	y	
	Moose	Mouse	0000	M	10000008	4	20130810	10:12:29	y	

Record: 1 of 1 No Filter Search

الشكل 2.2: نظام العضوية في Science World

## 2.3 أنواع مستخدمي قاعدة البيانات

يندرج مستخدمو قواعد البيانات ضمن أحد التصنيفات التالية:

1. **المستخدمون النهائيون End Users**: هم الأشخاص الذين تتطلب وظائفهم الوصول إلى قاعدة بيانات للاستعلام عن التقارير وتحديثها وإنشائها.

2. **مستخدم التطبيق user Application**: هو الشخص الذي يصل إلى برنامج تطبيقي موجود لأداء المهام اليومية.

3. **المستخدمون الخبراء Sophisticated users**: هم المستخدمون الذين لديهم طريقتهم الخاصة في الوصول إلى قاعدة البيانات. هذا يعني أنهم لا يستخدمون البرنامج التطبيقي المتوفر في النظام، فقد يحدّدون التطبيق الخاص بهم أو يصفون حاجتهم مباشرةً باستخدام لغات استعلام، يحتفظ هؤلاء المستخدمون المتخصصون بقواعد بياناتهم الشخصية باستخدام حزم البرامج الجاهزة التي توفر أوامر قائمةً على القوائم menu driven commands مثل برنامج MS Access.

4. **مبرمجو التطبيقات Application Programmers**: يطبق هؤلاء المستخدمون برماج تطبيقية محددة للوصول إلى البيانات المخزنة، حيث يجب أن يكونوا على دراية بنظم إدارة قواعد البيانات لإنجاز مهامهم بطريقة سلية.

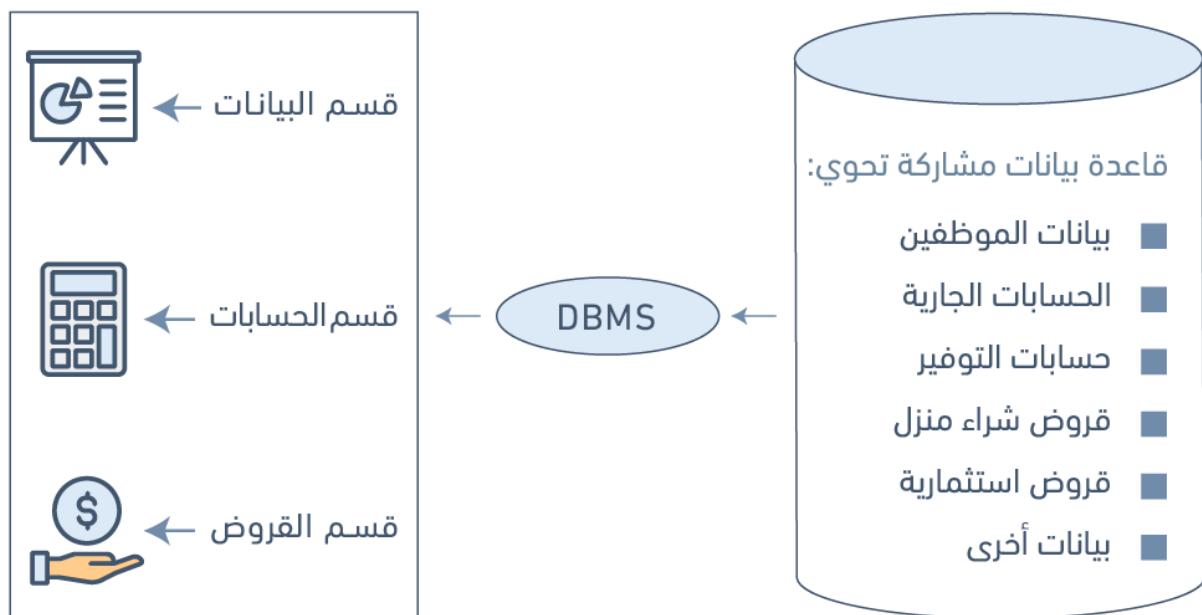
5. **مسؤولو قاعدة البيانات**: قد يكون مسؤول قاعدة البيانات DBA - أو Database Administrator - شخصاً أو مجموعة من الأشخاص في مؤسسة، المسؤولين عن إعطاء التصريح بالوصول إلى قاعدة البيانات ومراقبة استخدامها وإدارة جميع الموارد لدعم استخدام نظام قاعدة البيانات بأكمله.

## 2.4 نظام إدارة قواعد البيانات وتصنيفاتها

يُعدّ نظام إدارة قواعد البيانات system - أو DBMS اختصاراً- تجميعاً من البرامج التي تُمكّن المستخدمين من إنشاء قواعد البيانات databases، والحفظ عليها، والتحكم في جميع عمليات الوصول إليها، كما يُعدّ الهدف الأساسي لنظام إدارة قواعد البيانات هو توفير بيئه ملائمة وفعالة للمستخدمين لاسترجاع المعلومات وتخزينها.

يمكننا باستخدام نظام قواعد البيانات DBMS تمثيل النظام المصرفي التقليدي كما هو موضح في الشكل التالي، حيث يُستخدم في هذا المثال المصرفي نظام إدارة قواعد البيانات من قبل قسم شؤون الموظفين، وقسم الحسابات، وقسم إدارة القروض، للوصول إلى قاعدة البيانات المشتركة للشركة.

يمكن تصنيف أنظمة إدارة قواعد البيانات بناءً على عدة معايير، مثل: نموذج البيانات data model، وأعداد المستخدمين user numbers، وتوزيع قاعدة البيانات database distribution؛ وفيما يلي بيان تفصيلي لكل من هذه المعايير.



الشكل 2.3: نظام إدارة قواعد البيانات المصرفي

### 2.4.1 التصنيف على أساس نموذج البيانات

نموذج البيانات الأكثر انتشاراً المستخدم اليوم هو نموذج البيانات العلائقية relational data model، وذلك لأن جميع نظم إدارة قواعد البيانات، مثل: Oracle، MySQL، MS SQL Server، و DB2، تدعمه، ولا تزال النماذج التقليدية traditional models مثل نماذج البيانات الهرمية hierarchical data models الأخرى موجودة.

ونماذج بيانات الشبكة network data models مستخدمةً في الصناعة بصورة أساسية على منصات الحواسيب المركزية، ولكن نجدها محصورةً في استخدامات بسيطة بسبب تعقيدها، ويشار إليها على أنها نماذج تقليدية traditional models لأنها سبقت النموذج العلائقى relational model. وقد ظهرت في السنوات الأخيرة نماذج البيانات كائنية التوجه object-oriented data models، وهي نظام لإدارة قاعدة بيانات، حيث تمثل فيه المعلومات في شكل كائنات كما هو مستخدم في البرمجة كائنية التوجه.

تحتفل قواعد البيانات كائنية التوجه عن قواعد البيانات العلائقية relational databases، والتي تعتمد على الجدول أي تُعدّ جدولية التوجه table-oriented، كما تجمع أنظمة إدارة قواعد البيانات كائنية التوجه Object-oriented database management systems - وتخترق إلى OODBMS- بين إمكانيات قاعدة البيانات وإمكانيات لغات البرمجة كائنية التوجه.

ما زال انتشار قواعد البيانات كائنية التوجه ضعيف موازنة بقواعد البيانات العلائقية وذلك يرجع إلى عدم تعرّف المستخدمين عليها بعد، ويوجد بعض الأمثلة على نظم إدارة قواعد البيانات كائنية التوجه، وهي:

O2 •

ObjectStore •

Jasmine •

## 2.4.2 التصنيف على أساس أعداد المستخدمين

يمكن تصنيف نظم إدارة قواعد البيانات بناءً على عدد المستخدمين القادر على دعمهم، حيث من الممكن دعم مستخدم وحيد ويسمى نظام قواعد بيانات أحادي المستخدم ingle-user database system، أو دعم العديد من المستخدمين بصورة متزامنة ويسمى نظام قواعد بيانات متعدد المستخدمين multiuser database system.

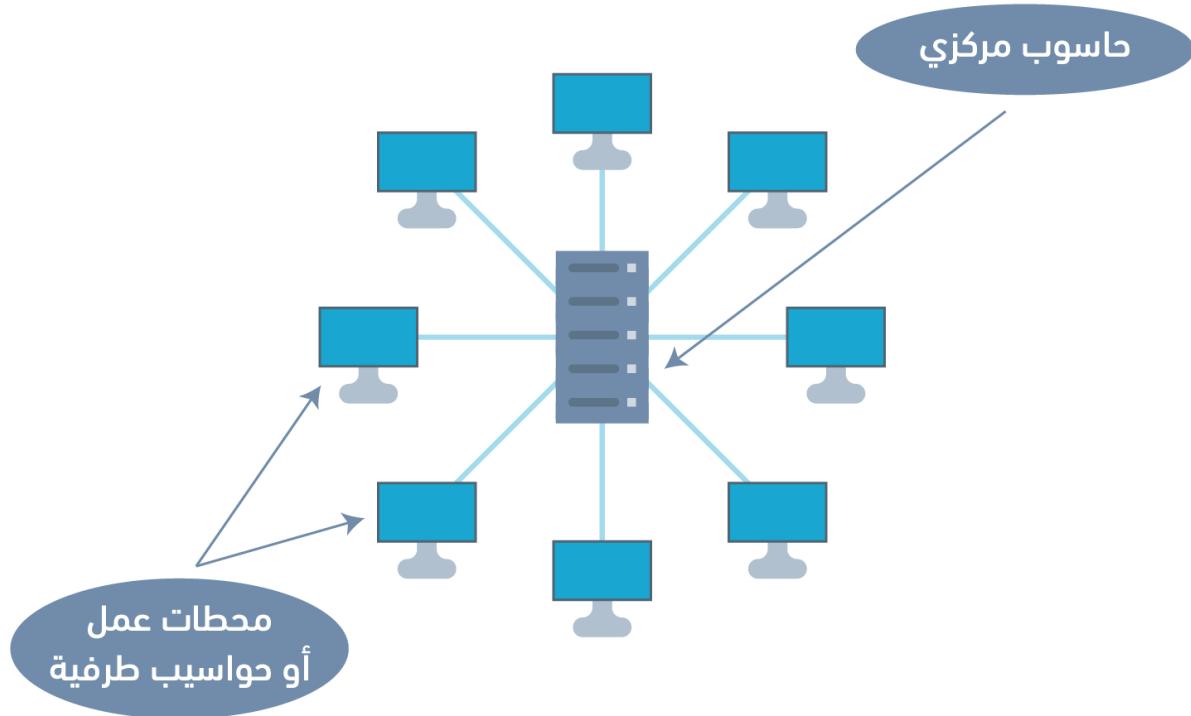
## 2.4.3 التصنيف على أساس توزيع قاعدة البيانات

توجد أربعة أنظمة توزيع رئيسية لأنظمة قواعد البيانات، ويمكن استخدامها لتصنيف قواعد البيانات حسب ما هو موضح أدناه.

### أ. الأنظمة المركزية Centralized systems

يُخَرَّن نظام إدارة قواعد البيانات DBMS وقاعدة البيانات database عند استخدام الأنظمة المركزية لقواعد البيانات centralized database system في موقع واحد تستخدمه أنظمة أخرى عديدةً كما هو موضح في الشكل التالي.

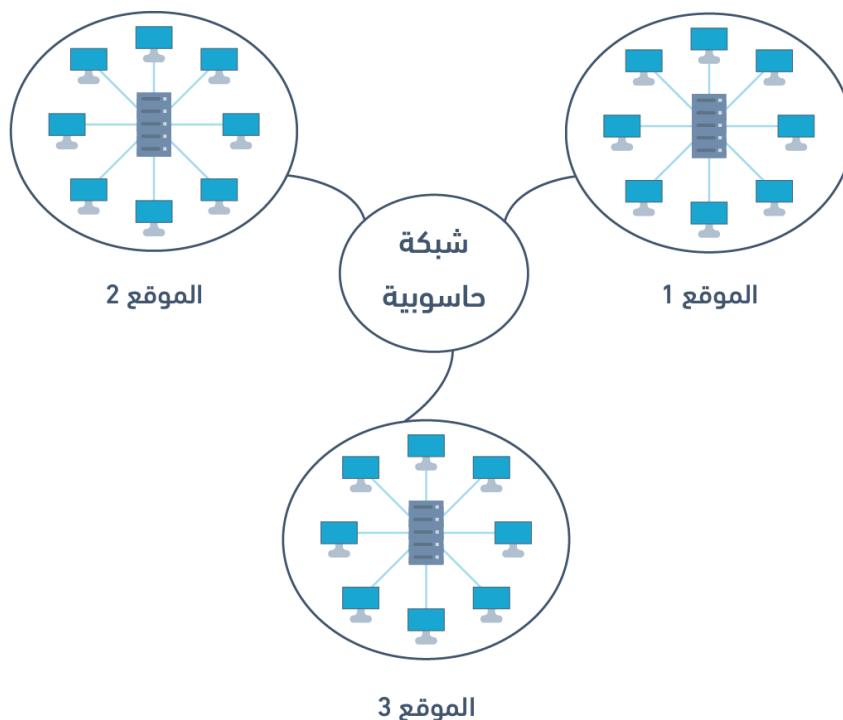
استخدمت العديد من المكتبات الكندية في أوائل الثمانينيات نظام 8000 GEAC لتحويل دليل أو فهرس البطاقات اليدوية إلى أنظمة فهرس مركبة يمكن قراءتها آلياً، حيث يحتوي كل فهرس على حقل باركود مشابه لذلك الموجود في منتجات المتاجر.



الشكل 2.4: مثال على نظام قاعدة بيانات مركبة.

### ب. نظام قاعدة البيانات الموزعة

يُوزَّع نظام إدارة قواعد البيانات DBMS وقاعدة البيانات database في نظام قاعدة البيانات الموزعة distributed database system من مواقع مختلفة متصلة بشبكة حاسوب، كما هو موضّح في الشكل التالي.



الشكل 2.5: مثال على نظام قاعدة بيانات موزعة.

### ج. أنظمة قواعد البيانات الموزعة المتجانسة

تستخدم أنظمة قواعد البيانات الموزعة المتجانسة Homogeneous distributed database systems برنامج إدارة قواعد البيانات نفسه من موقع متعددة، كما يمكن تبادل البيانات بين هذه المواقع المختلفة بسهولة، فمثلاً، تستخدم أنظمة معلومات المكتبات library information systems من البائع نفسه مثل نظام Geac Computer Corporation لإدارة قواعد البيانات نفسه، والذي يسمح بتبادل البيانات بسهولة بين مواقع مكتبة Geac المختلفة.

### د. أنظمة قواعد البيانات الموزعة غير المتجانسة

تستخدم موقع مختلف برنامج إدارة قواعد بيانات مختلف في نظام قاعدة البيانات الموزعة غير المتجانسة heterogeneous distributed database system، ولكن هناك برمج مشتركة إضافية تدعم تبادل البيانات بين هذه المواقع، فمثلاً، تستخدم أنظمة قاعدة بيانات المكتبات المختلفة تنسيق الفهرسة المقرورة آلياً MARC machine-readable cataloging -أي اختصاراً- نفسه لدعم تبادل بيانات تسجيلات المكتبة.

## 2.5 مصطلحات أساسية

- **عناصر البيانات data elements:** حقائق تمثل معلومات مستقاة من الواقع.
- **قاعدة البيانات database:** تجمیعه مشترکة من البيانات ذات الصلة، وُتُستخدم لدعم أنشطة منظمة معینة.
- **نظام إدارة قواعد البيانات DBMS أو database management system:** تجمیعه من البرامج التي تمكن المستخدیمین من إنشاء قواعد البيانات، والحفظ عليها، والتحكم في جميع عمليات الوصول إليها.
- **الجدول table:** مجموعة من الحقول .fields
- **نظام قاعدة البيانات المركزي centralized database system:** يخزن نظام إدارة قواعد البيانات DBMS وقاعدة البيانات database عند استخدام الأنظمة المركزية لقواعد البيانات centralized DBMS
- **نظام قاعدة البيانات الموزعة distributed database system:** يوزع نظام إدارة قواعد البيانات distributed database system في نظام قاعدة البيانات الموزعة distributed DBMS
- **نظام قاعدة البيانات الموزعة غير المتجانسة heterogeneous distributed database system:** تستخدم مواقع مختلفة ببرنامج إدارة قواعد بيانات مختلف، ولكن هناك برامح مشترکة إضافية تدعم تبادل البيانات بين هذه المواقع.
- **نظام قاعدة البيانات الموزعة المتجانسة homogeneous distributed database systems:** تستخدم برامح إدارة قواعد البيانات نفسه في مواقع متعددة.
- **نظام قاعدة بيانات متعدد المستخدیمین multiuser database system:** هو نظام إدارة قاعدة بيانات يدعم عدة مستخدیمین بصورة متزامنة.
- **نموذج البيانات كائنية التوجه object-oriented data model:** نظام لإدارة قواعد البيانات، حيث تمثل المعلومات فيه على صورة كائنات كما هو مستخدم في البرمجة كائنية التوجه.
- **نظام قاعدة بيانات أحدی المستخدیمین single-user database system:** نظام إدارة قاعدة بيانات يدعم مستخدم واحد فقط في كل مرة.
- **النماذج التقليدية traditional models:** هي نماذج البيانات التي سبقت النموذج العلائقی relational model

- **مبرمج التطبيق application programmer**: هو المستخدم الذي يطبق برمجياً تطبيقية محددة للوصول إلى البيانات المخزنة.
  - **مستخدم التطبيق application user**: يمكنه الوصول إلى برنامجٍ تطبيقي لأداء المهام اليومية.
  - **مسؤول قاعدة البيانات database administrator أو DBA**: هو الشخص المسؤول عن إعطاء التصريح بالوصول إلى قاعدة البيانات ومراقبة استخدامها وإدارة جميع الموارد لدعم استخدام نظام قاعدة البيانات بأكمله.
  - **المستخدم النهائي end user**: هو الشخص الذي تتطلب وظيفته الوصول إلى قاعدة بيانات للاستعلام عن التقارير وتحديثها وإنشائها.
  - **المستخدم الخبير sophisticated user**: هو الشخص الذي يستخدم طرقاً أخرى مختلفة عن البرنامج التطبيقي للوصول إلى قاعدة البيانات.
- ## 2.6 تمارين
1. ما هو نظام إدارة قواعد البيانات؟
  2. ما هي خصائص نظام إدارة قواعد البيانات؟
  3. اذكر ثلاثة أمثلة لقواعد بيانات مستقاة من الواقع مثل تحتوي المكتبة على قاعدة بيانات للكتب.
  4. اذكر ثلاثة أمثلة لقواعد البيانات العلائقية المستخدمة والأكثر شيوعاً.
  5. ما الفرق بين أنظمة قواعد البيانات المركزية والموزعة؟
  6. ما الفرق بين أنظمة قواعد البيانات الموزعة المتتجانسة وبين غير المتتجانسة؟

# 3. خصائص قواعد البيانات والمزايا التي تقدمها

تعني إدارة المعلومات معالجة المعلومات وتنظيمها لتوظيفها بما يخدمنا بطريقة نستفيد منها في تنفيذ مهامنا، كما منع نظام إدارة قواعد البيانات DBMS وجود الفوضى العرضية التي كانت تحدث للبيانات التي نجمعها ونضيفها إلى قواعد البيانات، حيث أصبح الوصول إليها أكثر سهولةً وتكاملاً مع بقية عملنا، كما تسمح لنا إدارة المعلومات باستخدام قاعدة بيانات أن نصبح مستخدمين استراتيجيين للبيانات التي نملكها.

غالباً ما نحتاج إلى الوصول إلى البيانات وإعادة فرزها لأغراض مختلفة تشمل ما يلي:

- إنشاء القوائم البريدية
- كتابة التقارير الإدارية
- توليد قوائم بالقصص الإخبارية المختارة
- تحديد احتياجات العملاء المختلفة

تملك قواعد البيانات قدرةً كبيرةً على معالجة البيانات، مما يسمح لها بإجراء العمليات التالية:

- الفرز Sort
- المطابقة Match
- ربط البيانات Link
- تجميع البيانات Aggregate
- تخطي الحقول Skip fields

## • إجراء العمليات الحسابية Calculate

## • ترتيب البيانات Arrange

تتعدد استخدامات قواعد البيانات وترتبط بـمجالات كثيرة، لذلك نجد من الممكن ربط قاعدة البيانات بكل من الأنظمة التالية:

## • موقع إلكتروني لتسجيل المستخدمين

## • تطبيقات الهواتف مثل تطبيق لتخزين بيانات عملاء منظمة تقدم خدمات اجتماعية

## • نظام السجلات الطبية لمنشأة رعاية صحية

## • دفتر العناوين address book الشخصية في عميل البريد الإلكتروني

## • تجميعة من الملفات النصية

## • نظام حجوزات الطيران

### 3.1 خصائص قواعد البيانات

يملك نظام قواعد البيانات عدداً من الخصائص والفوائد التي تميزه عن النظام القائم على الملفات file-based system، حيث سنذكر منها ما يلي:

#### 3.1.1 طبيعة الوصف الذاتي لنظام قاعدة البيانات

يُشار إلى نظام قاعدة البيانات على أنه ذاتي الوصف، وذلك بسبب احتوائه على قاعدة البيانات نفسها، وعلى بيانات وصفية metadata بحيث تُحدّد البيانات وتصفها، وال العلاقات بين الجداول في قاعدة البيانات، حيث تُستخدم هذه المعلومات بواسطة برامج أنظمة إدارة قواعد البيانات DBMS، أو مستخدمي قاعدة البيانات، ويعُد هذا الفصل بين البيانات ومعلوماتها أحد الفروقات الرئيسية التي تميز نظام قواعد البيانات عن النظام التقليدي القائم على الملفات، والذي يكون فيه تعريف البيانات جزءاً من برامج التطبيقات.

#### 3.1.2 العزل بين البرنامج والبيانات

تُحدّد هيكلة ملفات البيانات في النظام القائم على الملفات داخل برامج التطبيق، لذلك إذا أراد المستخدم تعديل هيكلة ملف معين، فعليه تعديل جميع البرامج التي تتصل بهذا الملف.

من الناحية الأخرى، تُخزن قواعد البيانات هيكلة البيانات في دليل catalog والنظام وليس في البرامج، لذلك كل ما هو مطلوب لتعديل هيكل ملف معين هو تعديل واحد فقط، ويسمى هذا بالعزل بين البرامج والبيانات أو الاستقلالية بين البرامج والبيانات program-data independence أيضاً.

## 3.2 دعم عدة واجهات عرض للبيانات

تدعم قاعدة البيانات استخدام عدة واجهات لعرض البيانات، حيث تُعدّ واجهة العرض view مجموعهً فرعيةً من قاعدة البيانات database، والتي تُعرف وتحُصّص لخدمة أغراض فئة محددة من مستخدمي النظام، وقد يملك مستخدمين متعددين واجهات مختلفة في النظام، حيث تحتوي كل منها على البيانات التي تهم مستخدم أو مجموعة من المستخدمين دون غيرهم.

### 3.2.1 مشاركة البيانات والنظام متعدد المستخدمين

صممت أنظمة قواعد البيانات لعدة مستخدمين، حيث تتيح لعدد من المستخدمين الوصول إلى قاعدة البيانات نفسها في الوقت نفسه، وذلك عن طريق استخدام استراتيجيات معينة تُسمى استراتيجيات التحكم المتزامنة concurrency control strategies، حيث تضمن هذه الاستراتيجيات صحة البيانات التي يتم الوصول إليها، كما تُحافظ على سلامة البيانات أيضًا.

يُعدّ تصميم أنظمة قواعد البيانات الحديثة المتعددة المستخدمين تحسنًا كبيرًا موازنًةً بتلك التي كانت في الماضي، والتي تقتصر على شخص واحد في كل مرة.

### 3.2.2 التحكم في تكرار البيانات

تُخزن البيانات في نظام قواعد البيانات - وفي الحالة المثالية - دون أي تكرار redundancy، أي أن كل عنصر بيانات موجود في مكان واحد فقط في قاعدة البيانات. ولكن يحدث في بعض الحالات تكرار للبيانات بفرض تحسين أداء النظام في أجزاء معينة، كما يُتحمّل في هذا التكرار عن طريق برمجة التطبيقات، وذلك بالمحافظة على الحد الأدنى منه عند تصميم قاعدة البيانات.

### 3.2.3 تشارك البيانات

يملك تكامل جميع بيانات المؤسسة داخل نظام قاعدة البيانات العديد من المزايا، حيث يسمح بمشاركة البيانات بين الموظفين وغيرهم من الذين يمكنهم الوصول إلى النظام، وكذلك يسمح للمستخدمين بتوليد المزيد من المعلومات من كمية معينة من البيانات أكثر مما سيكون بدون التكامل.

### 3.2.4 تطبيق قيود صارمة لضمان سلامة البيانات وصحتها

توفر أنظمة إدارة قواعد البيانات القدرة على تحديد وفرض قيود معينة على البيانات لضمان إدخال معلومات صحيحة من قبل المستخدمين، والمحافظة على سلامة البيانات، إذ تُعدّ قيود قاعدة البيانات database constraint قواعد لفرض ما يمكن إدخاله أو تعديله في جدول معين، مثل: الرمز البريدي باستخدام تنسيق معين، أو إضافة مدينة حقيقة في حقل المدينة.

هناك أنواع عديدة من القيود في قواعد البيانات، مثل: نوع البيانات Data type مثل تحديد نوع البيانات المسموح بها في الحقل مثل الأعداد فقط، أو تفرد البيانات Data uniqueness مثل المفتاح الأساسي والذي يضمن عدم إدخال أي تكرارات، كما يمكن أن تكون القيود بسيطة -بحيث تفرض على الحقل مباشرةً، أو معقدةً -أي برمجية.

### 3.2.5 تقييد الوصول الغير مصرح به

لا يحظى جميع مستخدمي نظام قاعدة البيانات بصلاحيات الوصول نفسها، فمثلاً، قد يكون لدى أحد المستخدمين صلاحيات القراءة فقط -أي القدرة على قراءة الملفات دون إجراء أي تعديلات عليها- بينما يكون لدى مستخدم آخر صلاحيات القراءة والكتابة -أي القدرة على قراءة الملفات والتعديل عليها-، ولهذا السبب يجب على نظام إدارة قاعدة البيانات توفير نظام أمان فرعي لإنشاء أنواع مختلفة من حسابات المستخدمين، والتحكم فيها، وتقييد الوصول الغير مصرح به.

### 3.2.6 استقلالية البيانات

يوجد ميزة أخرى لنظام إدارة قواعد البيانات، وهي الطريقة التي يسمح بها باستقلالية البيانات، بمعنى آخر، يتم فصل أوصاف بيانات النظام أو البيانات التي تصف البيانات -أي البيانات الوصفية metadata- عن برامج التطبيق، وهذا ممكن لأن نظام إدارة قاعدة البيانات يعالج التغييرات في هيكل البيانات، ولا تضمن هذه التغييرات في البرنامج نفسه.

### 3.2.7 معالجة المعاملات

يجب أن يتضمن نظام إدارة قواعد البيانات أنظمةً فرعيةً للتحكم في التزامن، حيث تضمن هذه الخاصية بقاء البيانات متسقةً وصالحةً أثناء معالجة المعاملات حتى وإن قام العديد من المستخدمين بتحديث المعلومات نفسها.

### 3.2.8 تقديم عدة واجهات عرض للبيانات

يسمح نظام إدارة قواعد البيانات DBMS للعديد من المستخدمين بالوصول إلى قواعد البيانات بصورة فردية أو بصورة متزامنة، كما ليس من المهم أن يعرف المستخدمون كيف وأين تخزن البيانات التي يصلون إليها.

### 3.2.9 النسخ الاحتياطي واسترجاع البيانات التالفة أو المفقودة

يعد النسخ الاحتياطي والاسترجاع طريقتين لحماية البيانات من الضياع، حيث يوفر نظام قواعد البيانات عمليةً منفصلةً عن عملية النسخ الاحتياطي للشبكة لنسخ البيانات احتياطياً واستعادتها، ويُعد النسخ الاحتياطي لقاعدة البيانات الطريقة الوحيدة لاستعادتها في حال فشل محرك الأقراص الثابتة وتعدّل الوصول إلى قاعدة البيانات المخزنة عليها.

إذا فشل نظام الحاسوب في منتصف عملية تحديث البيانات، فيكون النظام الفرعي للاسترجاع هو المسؤول عن التأكد من استعادة قاعدة البيانات إلى حالتها الأصلية، ويكون ما سبق فائدةً إضافيتين لنظام إدارة البيانات.

### 3.3 مصطلحات أساسية

- **استراتيجيات التحكم المتزامنة concurrency control strategies:** تسمح للعديد من المستخدمين بالوصول إلى عنصر البيانات نفسه في الوقت نفسه.
- **نوع البيانات data type:** يُحدّد نوع البيانات المسموح بها في حقل معين مثل يمكن أن يقبل الحقل أعداداً فقط.
- **تفرد البيانات data uniqueness:** يضمن عدم إدخال بيانات مكررة.
- **قيود قاعدة البيانات database constraint:** يُحدّد القيد ما يُسمح بإدخاله أو تعديله في جدول معين ومحدد.
- **البيانات الوصفية metadata:** تُحدّد وتصف البيانات وال العلاقات بين الجداول في قاعدة البيانات.
- **صلاحيات القراءة والكتابة read and write privileges:** القدرة على قراءة الملفات وتعديلها.
- **صلاحيات القراءة فقط read-only access:** القدرة على قراءة الملفات فقط دون تعديلها.
- **الوصف الذاتي self-describing:** يُشار إلى نظام قاعدة البيانات على أنه ذاتي الوصف، لأنّه يحتوي على قاعدة البيانات نفسها، بالإضافة إلى البيانات الوصفية التي تُحدّد وتصف البيانات وال العلاقات بين الجداول في قاعدة البيانات.
- **واجهة العرض:** مجموعة فرعية من قاعدة البيانات.

### 3.4 تمارين

1. ماذا يميّز نظام إدارة قاعدة البيانات DBMS عن النظام القائم على الملفات file-based system؟
2. ما هي استقلالية البيانات؟ وما أهميتها؟
3. ما هو الغرض من إدارة المعلومات؟
4. ناقش استخدام قواعد البيانات في بيئة العمل.
5. ما هي البيانات الوصفية؟

## 4. نمذجة البيانات وأنواعها

تُعد نمذجة البيانات Data Modeling الخطوة الأولى والأساسية عند تصميم أي قاعدة بيانات، كما تُعد هذه الخطوة مرحلة تصميم مجردة وعالية المستوى، كما يشار إليها باسم التصميم المفاهيمي conceptual design.

الهدف من هذه المرحلة هو إعطاء وصف واضح لكل من:

- البيانات الواردة في قاعدة البيانات، مثل الكيانات: طلاب، ومحاضرون، ودورات، ومواد.
- العلاقات بين عناصر البيانات data items، مثل: يشرف محاضرون على طلاب، ويدرسون محاضرون دورات.
- القيود المفروضة على البيانات، مثل: يتكون رقم الطالب من ثمانية خانات بالضبط، وتحتوي المادة الدراسية على أربع أو ست درجات فقط.

يُعبر في الخطوة الثانية عن عناصر البيانات، وال العلاقات، والقيود، باستخدام المفاهيم التي يوفرها نموذج البيانات عالي المستوى، ونظرًا لعدم وجود تفاصيل التنفيذ implementation details في هذه المفاهيم، فت تكون نتيجة عملية نمذجة البيانات تمثيلًا شبه رسمي لهيكل قاعدة البيانات، وهذه النتيجة سهلة الفهم، لذلك تُستخدم على أساس مرجع للتأكد من تلبية جميع متطلبات المستخدم.

الخطوة الثالثة هي تصميم قاعدة البيانات، حيث يكون لدينا خلال هذه الخطوة خطوتين فرعيتين، وهما: التصميم المنطقي لقاعدة البيانات logical design database، والتي تحدّد قاعدة البيانات في نموذج بيانات لنظام إدارة قواعد بيانات DBMS معين، والأخرى هي التصميم المادي لقاعدة

البيانات database physical design، والتي تحدد بنية تخزين قاعدة البيانات الداخلية، أو طريقة تنظيم الملفات، أو تقنيات الفهرسة، وتمثل هاتان الخطوتان الفرعيتان في التنفيذ الفعلي لقاعدة البيانات database implementation، وخطوات أساسية لبناء العمليات وواجهات المستخدم.

تُمثل البيانات في مراحل تصميم قاعدة البيانات باستخدام نموذج بيانات معين، حيث يكون نموذج البيانات data model مجموعة من المفاهيم، أو الصيغ التي تصف البيانات، وال العلاقات بينها، ودلائلها semantics، والقيود المفروضة عليها، كما تتضمن معظم نماذج البيانات أيضًا مجموعةً من العمليات الأساسية لمعالجة البيانات في قاعدة البيانات database.

## 4.1 أنواع نماذج البيانات

تُعد نماذج البيانات وسيلةً للتوصيف كيفية عرض البيانات وتخزينها، وسنناقش أنواعها في هذا القسم.

### 4.1.1 نماذج البيانات المفاهيمية عالية المستوى

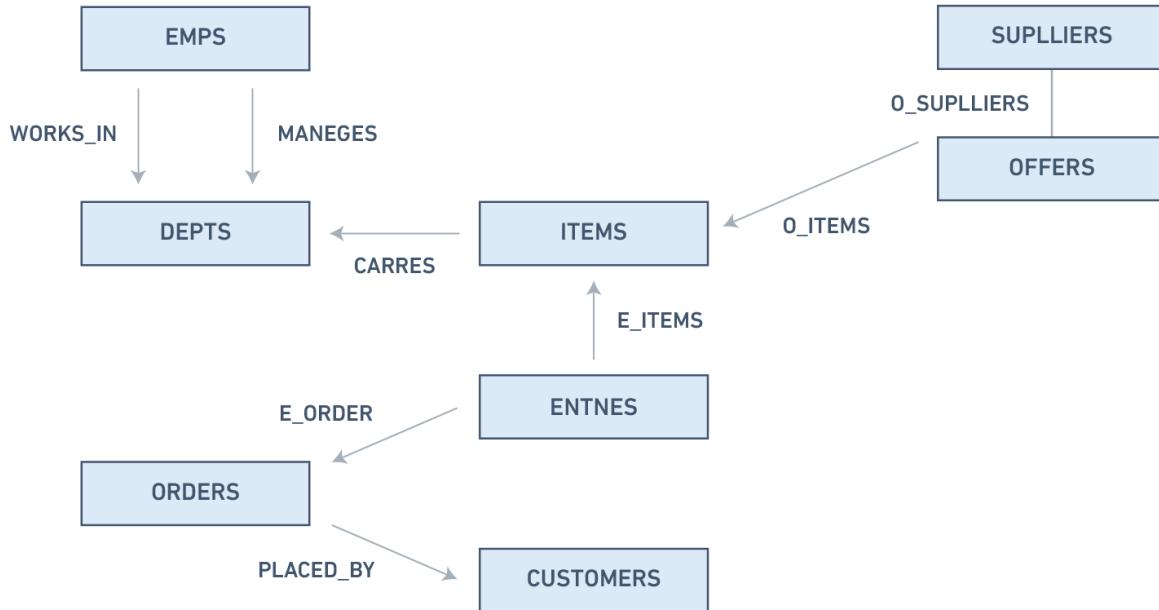
توفر نماذج البيانات المفاهيمية عالية المستوى High-level Conceptual Data Models مفهومًا لعرض البيانات بأساليب قريبة من الأسلوب الذي يدرك به الإنسان البيانات، والمثال النموذجي هو نموذج الكيان وال العلاقة Entity Relationship الذي يستخدم المفاهيم الرئيسية، مثل: الكيانات، والسمات، وال العلاقات، حيث يمثل الكيان كائناً واقعياً، مثل: موظف، أو مشروع، كما يحتوي على سمات تمثل خصائص، مثل: اسم الموظف، وعنوانه، وتاريخ ميلاده، وتمثل العلاقات كيفية ارتباط الكيانات مع بعضها البعض، فمثلاً، توجد علاقة بين الموظف وكل مشروع عندما يعمل الموظف في العديد من المشاريع.

### 4.1.2 نماذج البيانات المنطقية القائمة على السجلات

توفر نماذج البيانات المنطقية القائمة على السجلات Record-based Logical Data Models مفاهيم يمكن للمستخدمين فهمها واستيعابها كما أنها ليست بعيدةً جدًا عن الطريقة التي تخزن بها البيانات في الحاسوب. هناك ثلاثة نماذج معروفة من هذا النوع، وهي: نماذج البيانات العلائقية relational data models، ونماذج البيانات الشبكية network data models، ونماذج البيانات الهرمية hierarchical data models.

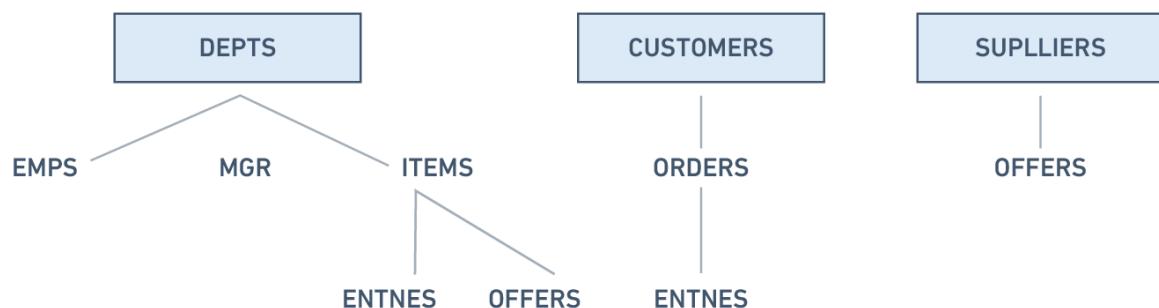
1. **يتمثل النموذج العلائقى relational model** البيانات على أساس علاقات أو جداول، فمثلاً، تضم العضوية في منظمة عالم العلوم World Science العديد من الأعضاء كما في الشكل 2 في فصل المفاهيم الأساسية في قواعد البيانات، ويُعد كل من معرف العضوية، وتاريخ انتهاء الصلاحية، ومعلومات العنوان، حقولاً في العضوية، ويكون الأعضاء أفراداً، مثل: Mickey، Minnie، و Mighty، Tom، و King， و Mang، و Moose، و Doorg.

2. يُمثل **النموذج الشبكي network model** البيانات على أساس أنواع سجلات، كما يُمثل أيضًا هذا النموذج نوعًا محددًا من علاقة واحد إلى متعدد one to many يسمى نوع المجموعة set type، كما هو موضح في الشكل التالي.



الشكل 4.1: مخطط النموذج الشبكي

3. يُمثل **النموذج الهرمي hierarchical model** البيانات على أساس هيكل شجرة هرمية، حيث يُمثل كل فرع من فروع التسلسل الهرمي عدًداً من السجلات ذات الصلة، ويوضح الشكل التالي هذا المخطط في صيغة النموذج الهرمي.



الشكل 4.2: مخطط النموذج الهرمي

## 4.2 مدى تجريد البيانات

ستلقي في هذا القسم نظرةً على عملية تصميم قاعدة البيانات من حيث تخصيصها لأداء وظائف معينة، فكما يبدأ أي تصميم بمستوى عالٍ من التجريد ثم ينتقل تدريجياً إلى التفاصيل الصغيرة، كذلك هو الحال عند تصميم قاعدة البيانات، فمثلاً، تبدأ عند بناء منزل بعدد غرف النوم والحمامات فيه، سواءً كان على مستوى واحد أو مستويات عدة، وتكون الخطوة التالية بتعيين مهندس معماري لتصميم المنزل تصميمًا أكثر تنظيماً ودقةً، حيث يصبح هذا المستوى أكثر تفصيلاً فيما يتعلق بأحجام الغرف، وكيف سيتم توصيل المنزل بالأسلامك، وأين ستُوضع تركيبات السباكة، وما إلى ذلك، والخطوة الأخيرة هي تعيين مقاول لبناء المنزل.

يتبع تصميم قاعدة البيانات يتبع طريقةً شبيهةً بهذه، حيث يبدأ بتحديد المستخدمين لقواعد العمل، ثم ينشئ مصممو ومحلو قاعدة البيانات تصميم قاعدة البيانات، وبعدها ينفرد مسؤول قاعدة البيانات التصميم باستخدام نظام إدارة قواعد البيانات DBMS.

تلخص الأقسام الفرعية التالية النماذج بترتيب تنازلي لمستوى التجريد.

### 4.2.1 النماذج الخارجية

- تمثل واجهة عرض قاعدة البيانات للمستخدم.
- تحتوي على عدد من واجهات عرض خارجية مختلفة.
- ترتبط ارتباطاً وثيقاً بالعالم الحقيقي الذي يراه المستخدم.

### 4.2.2 النماذج المفاهيمية

- توفر إمكانيات مرنة لهيكلة البيانات data-structuring.
- تقدم واجهة عرض مشتركة community view، وهي الهيكل المنطقي لقاعدة البيانات بأكملها.
- تحتوي على البيانات المخزنة في قاعدة البيانات.
- تظهر العلاقات بين البيانات بما في ذلك:
  - القيود.
  - المعلومات الدلالية مثل قواعد العمل.
  - معلومات الأمان والسلامة.
- تُعد قاعدة البيانات مجموعةً من الكيانات -أي الكائنات objects- من أنواع مختلفة.

- تمثل الأساس لتحديد وإعطاء وصف عالي المستوى لكائنات البيانات الرئيسية، كما أنها تتجاهل التفاصيل عموماً.
- تحدد هل قاعدة البيانات مستقلة بغض النظر عن قاعدة البيانات التي تستخدمها.

### 4.2.3 النماذج الداخلية Internal models

النماذج الثلاثة الأكثر شهرة من هذا النوع، هي: نموذج البيانات العلائقية relational data model، ونموذج البيانات الشبكية network data model، ونموذج البيانات الهرمي hierarchical data model، ومن السمات الرئيسية لنماذج البيانات الداخلية:

- تُعد قاعدة البيانات مثل تجميعه من السجلات ذات الحجم الثابت.
- تُعد أقرب إلى المستوى المادي أو بنية الملف.
- تمثل قاعدة البيانات كما يراها نظام إدارة قواعد البيانات DBMS.
- تطالب المصمم بمطابقة خصائص وقيود النموذج المفاهيمي مع خصائص نموذج التنفيذ المختار.
- يتضمن مقابلة الكيانات في النموذج المفاهيمي مع الجداول في النموذج العلائقى.

### 4.2.4 النماذج المادية Physical models

- هي التمثيل المادي أو الفيزيائي لقاعدة البيانات.
- تملك أدنى مستوى من التجريد.
- تحدد كيفية تخزين البيانات في قاعدة البيانات، وترتبط مباشرةً بكل من:
  - أداء قاعدة البيانات في وقت التشغيل Run-time performance.
  - تحسين التخزين وضغط الملفات.
  - تنظيم الملفات وطرق الوصول إليها.
  - تشفير البيانات.
- تحدد ما إذا كان نظام التشغيل operating system -أو OS اختصاراً- يدير المستوى المادي.
- تُقدم مفاهيم تصف تفاصيل كيفية تخزين البيانات في ذاكرة الحاسوب.

## 4.3 طبقات تجريد البيانات

يمكنك أن ترى في العرض التصويري كيف تعمل النماذج المختلفة معًا، لذلك دعنا نلقي نظرةً على هذا من أعلى مستوى، وهو النموذج الخارجي.

**النموذج الخارجي external model** هو كيفية عرض المستخدم النهائي للبيانات، فعادةً ما تكون قاعدة البيانات نظام مؤسسي يخدم احتياجات أقسام متعددة، كما لا يهتم أي قسم بروبية بيانات الأقسام الأخرى، فمثلاً، لا يهتم قسم الموارد البشرية human resources - أو HR اختصاراً- بعرض بيانات قسم المبيعات sales. وعليه تختلف طريقة عرض البيانات من مستخدم لآخر.

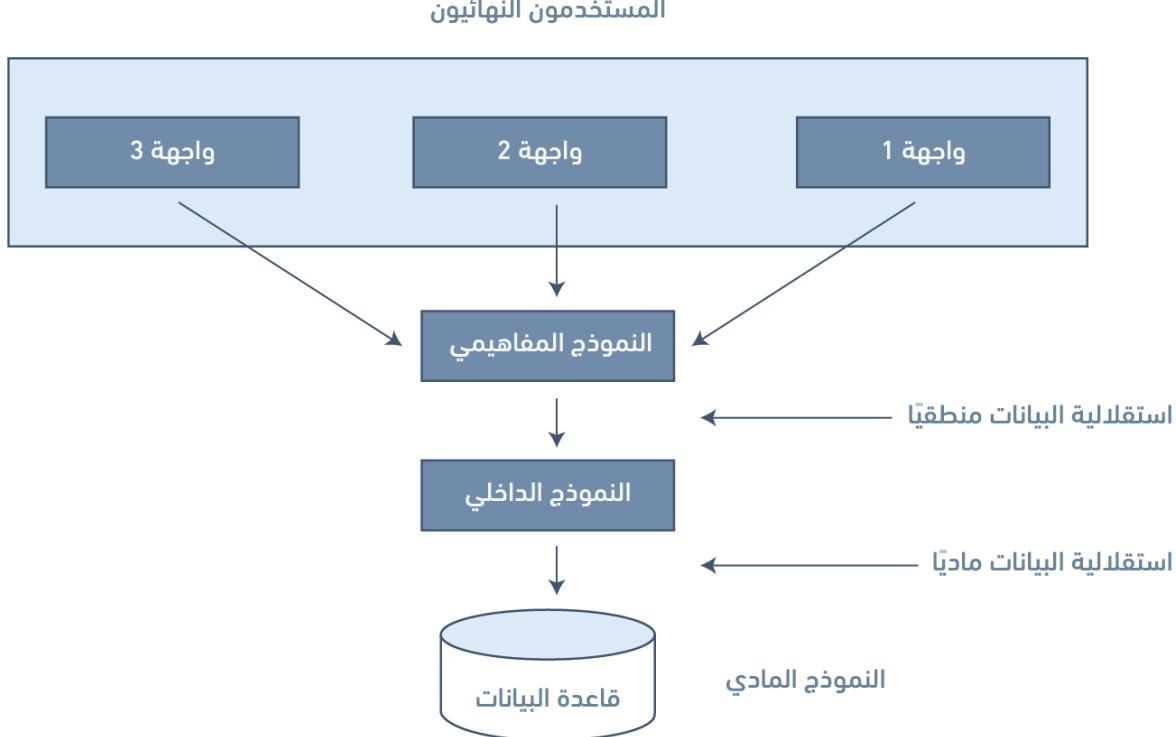
يتطلب النموذج الخارجي أن يقسّم المصمم مجموعة المتطلبات والقيود إلى وحدات وظيفية يمكن فحصها في إطار نماذجها الخارجية مثل تقسيم المؤسسة إلى قسم الموارد البشرية وقسم المبيعات.

تحتاج بوصفك مصمم بيانات إلى فهم جميع البيانات حتى تتمكن من إنشاء قاعدة بيانات على مستوى المؤسسة بناءً على احتياجات الأقسام المختلفة، لذلك يكون إنشاء **النموذج المفاهيمي conceptual model** هو الخطوة الأولى.

يكون النموذج المفاهيمي في هذه المرحلة مستقلًّا عن كل من البرامج software والعتاد hardware، ولا يعتمد على برنامج إدارة قواعد البيانات المستخدم في تنفيذ النموذج، ولا على العتاد المستخدم في ذلك، كما لا تؤثر التغييرات في العتاد أو برنامج إدارة قواعد البيانات على تصميم قاعدة البيانات على المستوى المفاهيمي.

بمجرد تحديد برنامج إدارة البيانات المراد استخدامه، يمكنك بعد ذلك تنفيذه، وهو ما يُسمى **بالنموذج الداخلي internal model**، حيث تقوم هنا بإنشاء جميع الجداول، والقيود، والمفاتيح، والقواعد، وما إلى ذلك، وغالبًا ما يشار إلى هذا باسم التصميم المنطقي logical design.

**النموذج المادي** ببساطة هو الطريقة التي تُخزّن فيها البيانات على القرص، وتختلف طريقة تخزين البيانات باختلاف نوع قاعدة البيانات المستخدمة.



الشكل 4.3: مستويات تجريد البيانات

#### 4.4 تطبيقات قاعدة البيانات Schemas

النطيط schema هو وصف عام وشامل لقاعدة البيانات، وعادةً ما يتم تمثيله بواسطة مخطط الكيان وال العلاقة entity relationship diagram .

غالباً ما تكون هناك العديد من التخطيطات الفرعية subschemas التي تمثل النماذج الخارجية المختلفة وبالتالي تعرض الواجهات الخارجية للبيانات.

فيما يلي قائمة بالعناصر التي يجب مراعاتها أثناء عملية تصميم قواعد البيانات:

- تخطيطات خارجية External schemas: من الممكن أن توجد عدة تخطيطات خارجية في قاعدة البيانات الواحدة.
  - تخطيطات فرعية متعددة Multiple subschemas: تعرض واجهات خارجية متعددة للبيانات.
  - تخطيط مفاهيمي Conceptual schema: يوجد تخطيط مفاهيمي واحد فقط لقاعدة البيانات الواحدة، يتضمن هذا التخطيط عناصر البيانات، والعلاقات، والقيود، وتمثّل بواسطة مخطط علاقة الكيان وال العلاقة ERD.
  - تخطيط مادي Physical schema: يوجد تخطيط مادي واحد فقط لقاعدة بيانات واحدة.

## 4.5 استقلالية البيانات المنطقية والمادية

يشير مفهوم استقلالية البيانات Data independence إلى حصانة تطبيقات المستخدم من التغييرات التي تطرأ على تعريفات البيانات وتنظيمها.

تكشف عمليات تحرير البيانات عن العناصر المهمة أو العناصر ذات الصلة بالمستخدم، وتكون التعقيد مخفياً عن مستخدم قاعدة البيانات.

تشكل استقلالية البيانات واستقلالية التشغيل معاً ميزة تحرير البيانات، وهناك نوعان من استقلالية البيانات، هما: استقلالية البيانات منطقياً، واستقلالية البيانات مادياً.

### 4.5.1 استقلالية البيانات منطقياً

يُعد التخطيط المنطقي logical schema تصميماً مفاهيمياً conceptual design لقاعدة البيانات، والذي يتم على الورق أو على لوح أبيض مثل الرسومات المعمارية للبيوت. تسمى القدرة على تغيير التخطيط المنطقي دون تغيير التخطيط الخارجي external schema، أو واجهة المستخدم باستقلالية البيانات منطقياً data independence، فمثلاً، يجب أن تكون إضافة أو إزالة كيانات جديدة، أو سمات، أو علاقات، إلى التخطيط المفاهيمي conceptual schema ممكناً دون الحاجة إلى تغيير التخطيطات الخارجية الحالية، أو إعادة كتابة برامج التطبيق؛ بمعنى آخر يجب ألا تؤثر التغييرات على التخطيط المنطقي على وظيفة التطبيق -أي طرق العرض الخارجية- مثل التعديلات على بنية قاعدة البيانات مثل إضافة عمود أو جداول جديد.

### 4.5.2 استقلالية البيانات مادياً

تشير استقلالية البيانات مادياً physical data independence إلى حصانة النموذج الداخلي ضد التغييرات في النموذج المادي، إذ يبقى التخطيط المنطقي دون تغيير على الرغم من إجراء تغييرات على تنظيم الملفات، أو هيكل التخزين، أو أجهزة التخزين، أو استراتيجية الفهرسة.

تعمل مرحلة استقلالية البيانات مادياً على إخفاء تفاصيل بنية التخزين من تطبيقات المستخدم، حيث لا ينبغي أن تتعامل التطبيقات مع هذه القضايا لعدم وجود فرق في العمليات الجارية على البيانات.

## 4.6 مصطلحات أساسية

- **النموذج الهرمي hierarchical model:** يمثل البيانات في هيكل الشجرة الهرمية.
- **النسخة instance:** سجل داخل جدول معين في قاعدة البيانات.
- **النموذج الشبكي network model:** يمثل البيانات على أساس أنواع سجلات.
- **العلاقة relation:** مصطلح آخر لوصف الجداول.

- **النموذج العلائقى relational model:** يُمثل البيانات على أساس علاقات أو جداول.
- **نوع المجموعة set type:** نوع محدد من علاقة واحد إلى متعدد one to many.
- **النموذج المفاهيمي conceptual model:** هو الهيكل المنطقي لقاعدة البيانات.
- **الخطيط المفاهيمي conceptual schema:** مرادف للخطيط المنطقي logical schema.
- **استقلالية البيانات data independence:** هي حصانة تطبيقات المستخدم من التغييرات التي تطرأ على تعاريفات البيانات وتنظيمها.
- **نموذج البيانات data model:** تجميعه من المفاهيم أو الصيغ المستخدمة لوصف البيانات، والعلاقات بينها، ودلائلها، والقيود المفروضة عليها.
- **نماذج البيانات data modeling:** هي الخطوة الأولى في عملية تصميم قاعدة البيانات.
- **التصميم المنطقي لقاعدة البيانات database logical design:** يُحدد قاعدة بيانات في نموذج البيانات الخاص بنظام إدارة قاعدة بيانات محدد.
- **التصميم المادي لقاعدة البيانات database physical design:** يُحدد بنية تخزين قاعدة البيانات الداخلية، أو تنظيم الملفات، أو تقنيات الفهرسة.
- **مخطط الكيان والعلاقة entity relationship diagram أو ERD:** يُعد نموذج بيانات، حيث يصف قاعدة البيانات، ويعرض الجداول، والسمات، والعلاقات.
- **النموذج الخارجي external model:** يُمثل واجهة عرض المستخدم لقاعدة البيانات.
- **الخطيط الخارجي external schema:** يُمثل واجهة المستخدم.
- **النموذج الداخلي internal model:** هو تمثيل قاعدة البيانات في الصورة التي يراها أو يتعامل معها نظام إدارة قواعد البيانات.
- **استقلالية البيانات منطقياً logical data independence:** هو القدرة على تغيير الخطيط المنطقي للبيانات دون تغيير الخطيط الخارجي.
- **التصميم المنطقي logical design:** هو الخطوة التي تُنشأ فيها الجداول، والقيود، والمفاتيح، والقواعد ... إلخ.
- **الخطيط المنطقي logical schema:** هو تصميم مفاهيمي لقاعدة البيانات، حيث يتم على الورق، أو الألواح البيضاء مثل الرسومات المعمارية لمنزل.
- **نظام التشغيل operating system أو OS:** هو المسؤول عن إدارة المستوى المادي للنموذج المادي.

- **استقلالية البيانات ماديًا** **physical data independence**: هو حصانة النموذج الداخلي ضد التغييرات في النموذج المادي.
- **النموذج المادي** **physical model**: هو التمثيل المادي لقاعدة البيانات.
- **الخطيط** **schema**: هو وصف عام وشامل لقاعدة البيانات.

## 4.7 التمارين

1. ما هو نموذج البيانات؟
2. ما هو نموذج البيانات المفاهيمي عالي المستوى؟
3. عرف المصطلحات التالية:
  - الكيان
  - السمة
  - العلاقة
4. اذكر وصف بإيجاز النماذج الشائعة لنماذج البيانات المنطقية القائمة على السجلات.
5. صِف الغرض من التصميم المفاهيمي.
6. ما هو الاختلاف بين التصميم المفاهيمي والتصميم المنطقي؟
7. عرف النماذج التالية:
  - ما هو النموذج الخارجي؟
  - ما هو النموذج المفاهيمي؟
  - ما هو النموذج الداخلي؟
  - ما هو النموذج المادي؟
8. ما هو النموذج الذي يتعامل معه مسؤول قاعدة البيانات؟
9. ما هو النموذج الذي يتعامل معه المستخدم النهائي لقاعدة البيانات؟
10. ما هو الاستقلال البيانات ماديًا؟
11. ما هو استقلال البيانات منطقيًا؟

# 5. نموذج البيانات العلائقية RDM

صمم نموذج البيانات العلائقية Relational Data Model في العام 1970 بواسطة C.F. Codd، وهو النموذج الأكثر استخداماً في يومنا هذا، كما يُعدّ الأساس لكل من:

- البحث العلمي في نظرية البيانات، وال العلاقات، والقيود.
- العديد من منهجيات تصميم قواعد البيانات.
- لغة الوصول القياسيّة إلى قاعدة البيانات، حيث تسمى لغة الاستعلام المهيكلة structured query language - أي SQL language.
- جميع أنظمة إدارة قواعد البيانات التجارية الحديثة.

يصف نموذج البيانات العلائقية العالم على أنه تجمّعه من العلاقات والجداول المتراكبة.

## 5.1 المفاهيم الأساسية في نماذج البيانات العلائقية

ستتعرف على المفاهيم الأساسية في نموذج البيانات العلائقية التي ترتكز ارتكازاً كبيراً على العلاقة والجدول وكل الخصائص المتعلقة بهما.

### 5.1.1 العلاقة

العلاقة relation - أو ما تعرف أيضًا باسم الجدول table أو الملف file -، وهي مجموعة فرعية من الناتج الديكارتي لقائمة من المجالات التي تتميز بالاسم، حيث يمثل كل صفات row ضمن الجدول الواحد مجموعةً من قيم البيانات ذات الصلة، ويُعرف الصفات أو السجل record باسم صفات tuple أيضًا، كما يُعدّ العمود في الجدول

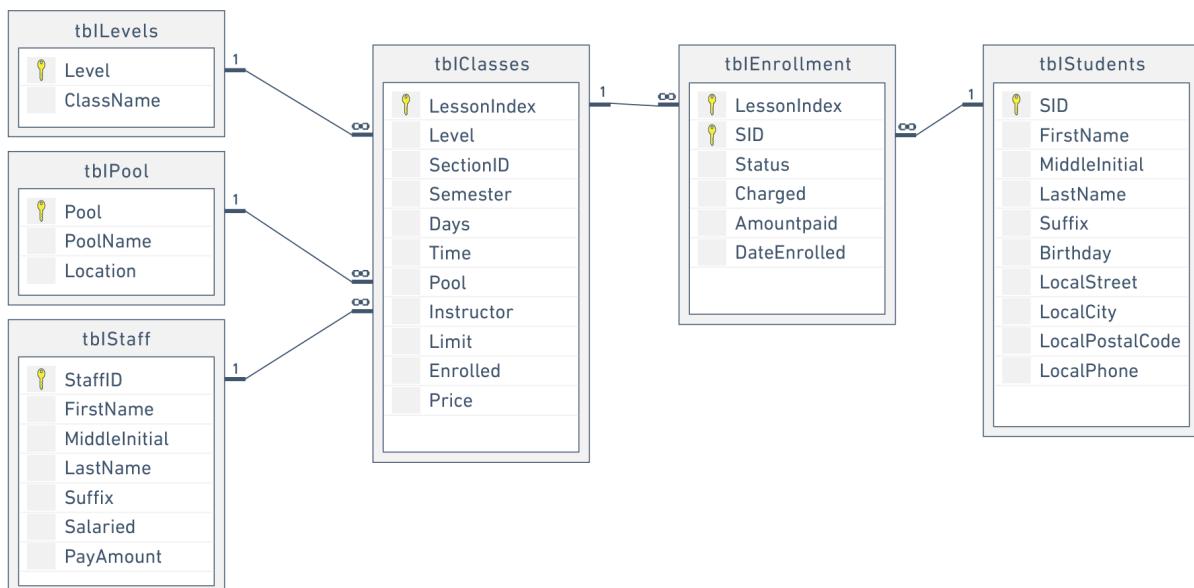
حقلًا، ويشار إليه باسم السِّمة attribute أيضًا، كما يمكنك النظر إلى الأمر بالطريقة التالية: تُستخدم السمات لتعريف السجلات، ويحتوي السجل على مجموعة من السمات.

توضّح الخطوات التالية المنطق بين العلاقة و مجالاتها:

- يُشار إلى عدد  $n$  من المجالات بالصورة:  $D_1, D_2, \dots, D_n$ .
- تُعدّ  $r$  علاقةً محدّدة على هذه المجالات.
- $r \subseteq D_1 \times D_2 \times \dots \times D_n$  ثم

## Table 5.1.2 الجدول

ت تكون قاعدة البيانات من عدة جداول، كما يحتوي كل جدول على بيانات، ويوضّح الشكل التالي قاعدة بيانات تحتوي على ستة جداول.



الشكل 5.1: قاعدة بيانات بستة جداول

## Column 5.1.3 العمود

تُخّزن قاعدة البيانات أجزاء المعلومات أو الحقائق بطريقة منظّمة، حيث يتطلّب الفهم الجيد لكيفية استخدام قواعد البيانات والاستفادة منها إلى أقصى حد فهم طريقة التنظيم هذه.

تسمى وحدات التخزين الرئيسية أعمدة columns، أو حقول fields، أو سمات attributes، وتضم هذه الوحدات المكونات الأساسية للبيانات التي يمكن تقسيم محتواها.

تحتاج عند تحديد الحقول المراد إنشاؤها إلى التفكير في البيانات التي لديك عمومًا، فمثلاً، استخلاص المكونات المشتركة للمعلومات التي ستخّزنها في قاعدة البيانات، وتجنب التفاصيل التي تميز عنصراً عن الآخر.

انظر الجدول التالي الذي يحوي مثلاً على معلومات حول بطاقة هوية:

Field Name	Data
First Name	Isabelle
Family Name	Whelan
Nationality	British
Salary	109,900
Date of Birth	15 September 1983
Marital Status	Single
Shift	Mon, Wed
Place of issue	Addis Ababa
Valid until	17 December 2003

## 5.1.4 Domain المجال

يُمثل المجال المجموعات الأصلية للقيم الدارية المستخدمة لنمذجة البيانات، وتعني القيمة الدارية **atomic value** أن كل قيمة في المجال غير قابلة للتجزئة بقدر ما يتعلق الأمر بالنمذج العلائقي، فمثلاً:

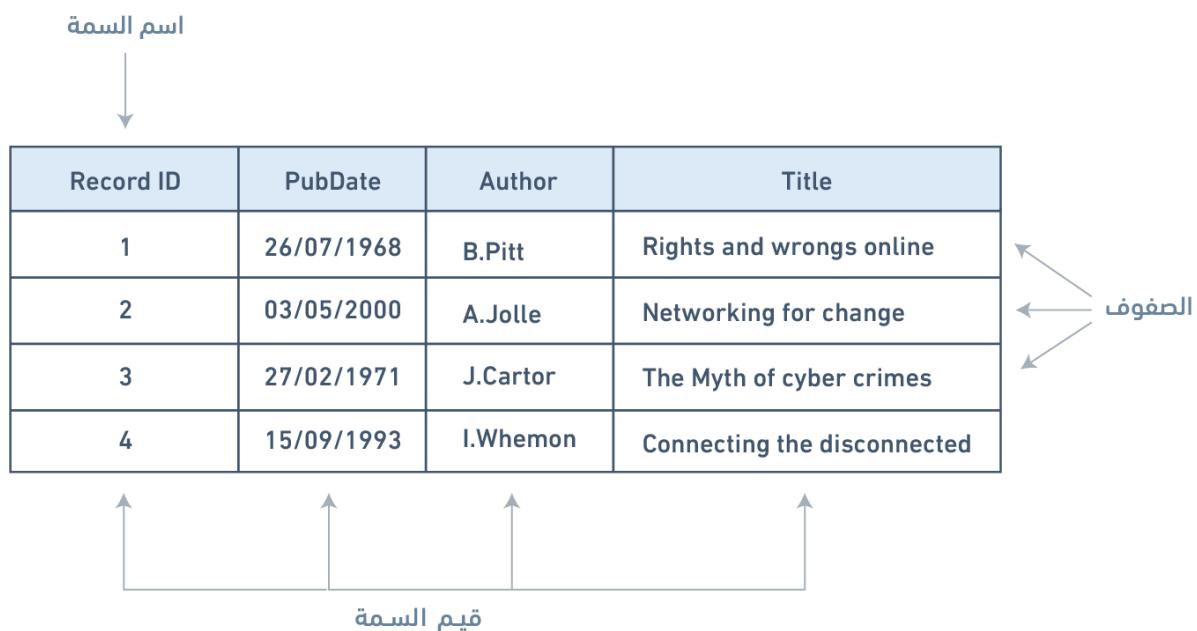
- يملك مجال الحالة الاجتماعية مجموعةً من الاحتمالات، وهي: متزوج، وأعزب، ومطلّق.
- يملك مجال أيام العمل مجموعةً من جميع الأيام الممكنة، وهي: الأحد، الاثنين، ...إلخ.
- مجال الراتب الشهري هو مجموعة جميع الأعداد الأكبر من 0 وأقل من 200000.
- مجال الاسم الأول هو مجموعة سلاسل الأحرف التي تمثل أسماء الأشخاص.

باختصار، المجال هو مجموعة من القيم المقبولة التي يُسمح للعمود باحتواها، كما يعتمد على الخصائص المختلفة ونوع بيانات العمود، وسنناقش أنواع البيانات في فصل آخر.

## 5.1.5 Records السجلات

مثلاً يحتاج محتوى أي مستند أو عنصر إلى تقسيمه إلى أجزاء صغيرة من البيانات للتخزين في الحقول، فيجب أيضاً أن تكون مترابطةً بحيث يمكن إعادةها إلى شكلها الكامل، ويتم ذلك عن طريق استخدام السجلات، إذ تحتوي السجلات على حقول مرتبطة، مثل: حقل العميل **customer**، أو الموظف **employee**، كما يُستخدم المصطلح **صف** أو **سطر tuple** أحياناً على أساس مرادف للسجل.

تشكل السجلات والحقول أساس جميع قواعد البيانات، وينحنا الجدول البسيط أوضح صورة عن كيفية عمل السجلات والحقول معاً في مشروع تخزين قاعدة بيانات.



الشكل 5.2: مثال على جدول بسيط

يوضح الجدول البسيط في الشكل السابق كيف يمكن للحقول الاحتفاظ بمجال واسع من مختلف أنواع البيانات، حيث يحتوي على:

- حقل ID (معرف السجل): هو عدد صحيح، ويُستخدم لترتيب البيانات في الجدول.

- حقل PubDate (تاريخ النشر): ويحتوي على تاريخ النشر ويكون في الصورة (يوم/شهر/سنة).

- حقل Author (المؤلف): يحتوي على اسم المؤلف، وهو حقل يحتوي على بيانات نصية.

- نص Title (حقل العنوان): يحتوي على أي نص يُمثل عنوان المؤلف.

من الممكن توجيه قاعدة البيانات للبحث في البيانات وتنظيمها بطريقة معينة، فمثلاً، يمكنك طلب مجموعة مختارة من السجلات محددة حسب التاريخ بطرق عديدة، وهي: كل البيانات المسجلة قبل تاريخ معين، أو كل البيانات المسجلة بعد تاريخ معين، أو كل البيانات المسجلة بين تاريخين، ويمكنك بالمثل أيضاً ترتيب السجلات حسب التاريخ.

نظرًا لإعداد الحقل أو السجل الذي يحتوي على البيانات على أساس حقل تاريخ، فلن تقرأ قاعدة البيانات المعلومات الموجودة في حقل التاريخ على أساس أعداد مفصولة بشرط مائلة، وإنما على أساس تواريخ، بحيث يجب ترتيبها وفقًا لنظام التقويم.

### Degree 5.1.6 الدرجة

الدرجة هي عدد السمات في الجدول، فدرجة الجدول في المثال السابق الموضح في الشكل هي 4.

## 5.2 خصائص الجدول

- لكل جدول في قاعدة البيانات اسم خاص به، ولا يتكرر الاسم الواحد في عدة جداول.
- يحتوي كل صفت في الجدول على بيانات فريدة، ولا يتكرر الصفت نفسه أكثر من مرة في الجدول.
- تكون المدخلات في الأعمدة ذريةً بحيث لا يحتوي الجدول على مجموعات مكررة أو سمات متعددة القيم.
- تكون المدخلات في الأعمدة من المجال نفسه بناءً على نوع بياناتها، فإما أن تكون أعداد - أي عدد صحيح، كسري ... الخ- أو سلسلة محارف، أو تاريخ، أو قيم منطقية - أي صح أو خطأ.
- غير مسموح بالعمليات التي تجمع بين أنواع البيانات المختلفة.
- كل سمة لها اسم فريد.
- ترتيب الأعمدة غير مهم.
- ترتيب الصفوف غير مهم.

## 5.3 المفاهيم الأساسية

- **القيمة الذرية value atomic**: تعني أن كل قيمة في المجال غير قابلة للتجزئة بقدر ما يتعلق الأمر بالنماذج العلائقية.
- **السمة attribute**: وحدة التخزين الأساسية في قواعد البيانات.
- **العمود column**: هو نفسه السمة attribute التي ذكرتها للتو.
- **الدرجة degree**: عدد السمات في الجدول.
- **المجال domain**: هو المجموعات الأصلية للقيم الذرية المستخدمة لنمذجة البيانات، وهو مجموعة من القيم المقبولة التي يُسمح للعمود باحتواها.
- **الحقل field**: يكافئ السمة attribute.
- **الملف file**: يكافئ العلاقة relation.
- **السجل record**: يحتوي على عدة حقول ذات صلة بعضها البعض، ويكافئ السطر tuple.
- **العلاقة relation**: مجموعة فرعية من الناتج الديكارتي لقائمة من المجالات التي تتميز بالاسم، وهي المصطلح التقني لكل من الجدول table ، أو الملف file.
- **الصف row**: يكافئ السطر tuple.

- لغة الاستعلام المهيكلة أي SQL: هي لغة الوصول القياسي إلى قاعدة البيانات.

- الجدول table: يكافئ العلاقة relation.
- السطر tuple: مصطلح تقني مرادف للصف أو السجل.

## 5.4 المصطلحات الأساسية

وردت العديد من المصطلحات المترادفة في هذا الفصل، ويرجى الرجوع إلى الجدول التالي بالإضافة إلى المفاهيم الأساسية السابقة، كما يحتوي العمود البديل الأول في هذا الجدول على المصطلحات الأكثر شيوعاً واستخداماً اليوم.

البديل الثاني	البديل الأول	المصطلح الرسمي - أي حسب العالم Codd
الملف File	الجدول Table	العلاقة Relation
السجل Record	الصف Row	السطر Tuple
الحقل Field	العمود Column	السمة Attribute

## 5.5 تمارين

استخدم الجدول التالي للإجابة على الأسئلة أدناه:

معرف سجل الموظف	اسم الموظف	تهيئة الموظف	كنية الموظف	رمز مهنة الموظف	البديل الثاني
123455	فريد	A.	عبد الله	12	
123456	علا	D.	عادل	18	
123457	فاتن	G.	علي	15	
123458	كريم	X.	إسماعيل	18	

1. حدد وصف جميع المكونات في الجدول باستخدام المصطلحات الصحيحة.
2. ما هو المجال المحتمل للحقل EmpJobCode؟
3. كم عدد السجلات المعروضة وكم عدد السمات المعروضة؟
4. اذكر خصائص الجدول.

# 6. نموذج الكيان والعلاقة ER وتمثيل البيانات

ظهر نموذج الكيان والعلاقة relationship entity - أو ER اختصاراً- لتمثيل البيانات منذ أكثر من 35 عام، وهو مناسب تماماً لنماذج البيانات للاستخدام مع قواعد البيانات، وذلك لأنه ذو طبيعة مجردة إلى حد ما، وسهل المناقشة والشرح.

تترجم نماذج الكيان والعلاقة للبيانات إلى علاقات بسهولة، كما تسمى تخطيط الكيان والعلاقة ER schema، حيث تُمثل بواسطة مخططات الكيان والعلاقة ER diagrams.

تعتمد عملية إنشاء نماذج الكيان والعلاقة للبيانات ER modelling على مفهومين هما:

- **البيانات Entities:** وتعُرف على أنها الجداول التي تحتوي على معلومات خاصة - أي بيانات.
- **العلاقات Relationships:** وتعُرف على أنها الارتباطات أو التفاعلات بين البيانات.

فيما يلي مثال على كيفية دمج هذين المفهومين في نموذج الكيان والعلاقة: يكون في قولنا "يدرس الأستاذ دورة أنظمة قواعد البيانات" الكيان هو كل من الأستاذ، ودورة أنظمة قواعد البيانات؛ أما العلاقة فهي الكلمة يدرس.

ستستخدم في هذا الفصل قاعدة بيانات تسمى الشركة COMPANY لتوضيح مفاهيم نموذج الكيان والعلاقة، حيث تحتوي على معلومات حول الموظفين employees، والأقسام departments، والمشاريع projects، كما يجب ملاحظة النقاط التالية:

- هناك عدة أقسام في الشركة، ولكل منها قسم مُعرّف فريد، واسم، وموقع للمكتب، وموظف معين يدير القسم.
- يتحكم القسم في عدد من المشاريع، ولكل منها اسم فريد، ورقم فريد، وميزانية.

- كل موظف له اسم، ورقمتعريف، وعنوان، وراتب، وتاريخ ميلاد، كما يُعين الموظف في قسم واحد، ويمكنه الانضمام لعدة مشاريع، كما تحتاج إلى تسجيل تاريخ بدء الموظف في كل مشروع، ومعرفة المشرف المباشر لكل موظف.

- نريد تتبع المعالين لكل موظف، حيث يملك كل مُعال اسم، وتاريخ ميلاد، وعلاقته بالموظفي.

## 6.1 الكيان ومجموعة الكيان ونوع الكيان

الكيان entity هو كائن في العالم الحقيقي له وجود مستقل، كما يمكن تمييزه عن الكائنات الأخرى، وقد يكون هذا الكيان:

- كائن له وجود مادي physical existence، مثل: محاضر، وطالب، و سيارة.
- كائن له وجود مفاهيمي conceptual existence، مثل دورة، ووظيفة، ومنصب.
- يمكن تصنيف الكيانات بناءً على قوتها، حيث يُعدّ الكيان ضعيفاً في الحالات التالية:
  - وجوده غير ممكن بدون علاقة مع كيان آخر.
  - مفاته الرئيسي مشتق من المفتاح الرئيسي للكيان الألب.
- يُعدّ جدول الزوج Spouse في قاعدة بيانات الشركة كياناً ضعيفاً لأن مفتاحه الرئيسي يعتمد على جدول الموظف، أي لن يكون سجل الزوج موجوداً إذا لم يتواجد سجل الموظف المقابل.
- يُعدّ الكيان قوياً إذا كان يمكن أن يوجد مستقلاً عن جميع الكيانات المرتبطة به.
- الأنوية Kernels هي كيانات قوية.
- يُعدّ الجدول كياناً قوياً إذا لم يحتوي على مفتاح خارجي key foreign، أو إذا احتوى على مفتاح خارجي يقبل القيم الفارغة null.

يجب معرفة مصطلح آخر، وهو نوع الكيان entity type الذي يحدّد تجميعة من الكيانات المتشابهة، كما تُعدّ مجموعة الكيان entity set تجميعة من نوع الكيان entity type في وقت معين.

يُمثل نوع الكيان في مخطط الكيان وال العلاقة ER- أي entity relationship diagram في صندوق، فمثلاً، نوع الكيان في الشكل التالي هو موظف EMPLOYEE.



الشكل 6.1: نوع الكيان هو الموظف

## 6.2 ارتباط الوجود

يعتمد وجود الكيان على وجود الكيانات ذات الصلة به، ويُعدّ الكيان ارتباطي الوجود Existence dependency إذا كان يحتوي على مفتاح خارجي إلزامي -أي سمة مفتاح خارجي لا يمكن أن تكون فارغة null-، فمثلاً، يكون في قاعدة بيانات الشركة COMPANY كيان الزوج Spouse معتمداً على وجود كيان الموظف.

## 6.3 أنواع الكيانات

يجب أيضاً أن تكون على دراية بأنواع الكيانات المختلفة بما في ذلك:

- الكيانات المستقلة independent entities
- الكيانات المعتمدة dependent entities
- الكيانات المميزة characteristic entities

### 6.3.1 الكيانات المستقلة

تُعدّ الكيانات المستقلة Independent entities -والتي يشار إليها أيضاً باسم الأنوية kernels- العمود الفقري لقاعدة البيانات، إذ تستند عليه الجداول الأخرى، ولها الخصائص التالية:

- هم اللبنات الأساسية لقاعدة البيانات.
- قد يكون المفتاح الرئيسي primary key بسيطاً، أو مركباً.
- لا يمكن أن يكون المفتاح الرئيسي مفتاحاً خارجياً.
- لا تعتمد على أي كيان آخر في وجودها.

إذا عدنا إلى قاعدة بيانات الشركة COMPANY الخاصة بنا، فتتضمن أمثلة الكيانات المستقلة جدول العميل Customer table، أو جدول الموظف Employee table، أو جدول المنتج Product table.

### 6.3.2 الكيانات المعتمدة

تستند الكيانات المعتمدة Dependent entities - على جداول أخرى حتى يكون لها معنى، ولها الخصائص التالية:

- تُستخدم الكيانات المعتمدة لربط نوافذ معاً.
- يعتمد وجودها على وجود جدولين أو أكثر في قاعدة البيانات، حيث لا يمكن وجود كيانات معتمدة في قاعدة بيانات تحتوي على جدول واحد فقط.
- تُنشئ علاقات (متعدد إلى متعدد) جداول ترابطية associative tables بمحاتين خارجيين على الأقل.
- قد تحتوي على سمات أخرى.
- يحدد كل مفتاح خارجي جدولًا مرتبطًا بالكيان نفسه.
- هناك ثلاثة خيارات للمفتاح الرئيسي:
  1. استخدام مزيجًا من المفاتيح الخارجية للجداول المرتبطة إذا كانت فريدة.
  2. استخدام مزيجًا من المفاتيح الخارجية وعمودًا مؤهلاً.
  3. أنشئ مفتاح رئيسي بسيط جديد.

### 6.3.3 الكيانات المميزة

توفر الكيانات المميزة Characteristic entities مزيدًا من المعلومات حول جدول آخر، ولهذه الكيانات الخصائص التالية:

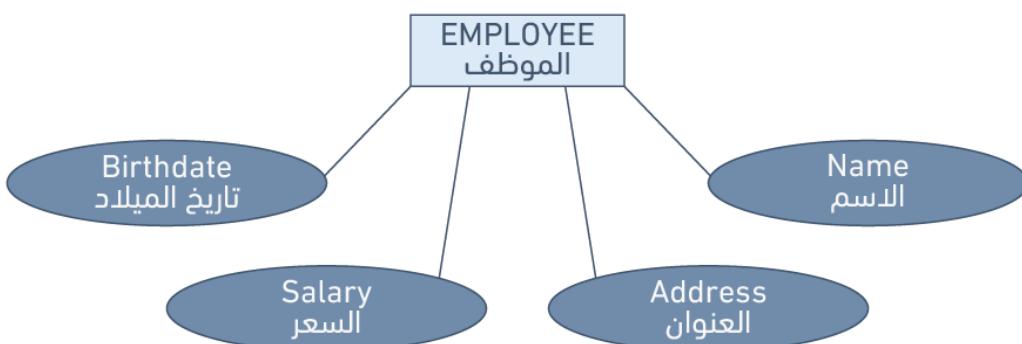
- تمثل سمات متعددة القيم.
- تصف كيانات أخرى.
- عادة ما يكون لديها علاقة علقة واحد إلى متعدد .one to many relationship.
- يُستخدم المفتاح الخارجي foreign key لتحديد الجدول المميز characterized table.
- خيارات المفتاح الرئيسي هي:
  1. استخدام مزيجًا من المفاتيح الخارجية وعمودًا مؤهلاً.
  2. أنشئ مفتاحًا رئيسيًا بسيطًا جديداً في قاعدة بيانات الشركة COMPANY، والتي قد تشمل:
    - جدول الموظف Employee (المعرف EID، الاسم، العنوان، العمر، الراتب) - EID هو المفتاح الرئيسي البسيط.

- جدول هاتف الموظف EmployeePhone (معرف الموظف EmployeeID، رقم الهاتف)، ID هنا هو جزء من مفتاح رئيسي مركب، وهو مفتاح خارجي مرتبط بجدول الموظف السابق.

## 6.4 السمات

يوضح كل كيان بمجموعة من السمات attributes، فمثلاً، يمكن وصف كيان الموظف بالسمات التالية: الاسم، أو العنوان، أو تاريخ الميلاد، أو الراتب. تملك كل سمة اسمًا محدّدًا، وترتبط بكيان معين، وب مجال من القيم المسموحة التي يمكن أخذها، ولكن لا تُعرض المعلومات حول مجال السمة في مخطط الكيان وال العلاقة ERD.

تمثّل كل سمة في مخطط الكيان وال العلاقة الموضح في الشكل التالي تمثيلًا بيضوياً مع اسم بداخله.



الشكل 6.2: تمثيل السمات في نموذج العلاقات الكائني للبيانات

### 6.4.1 أنواع السمات

هناك أنواع قليلة من السمات التي يجب أن تكون على دراية بها، ويجب ترك بعضها كما هي، لكن يحتاج بعضها الآخر إلى تعديل ليسهل تمثيلها في النموذج العلائقى relational model، وسيناقشه هذا القسم أنواع السمات؛ أما لاحقًا فسنناقش تعديل السمات لتلائم النموذج العلائقى بصورة صحيحة.

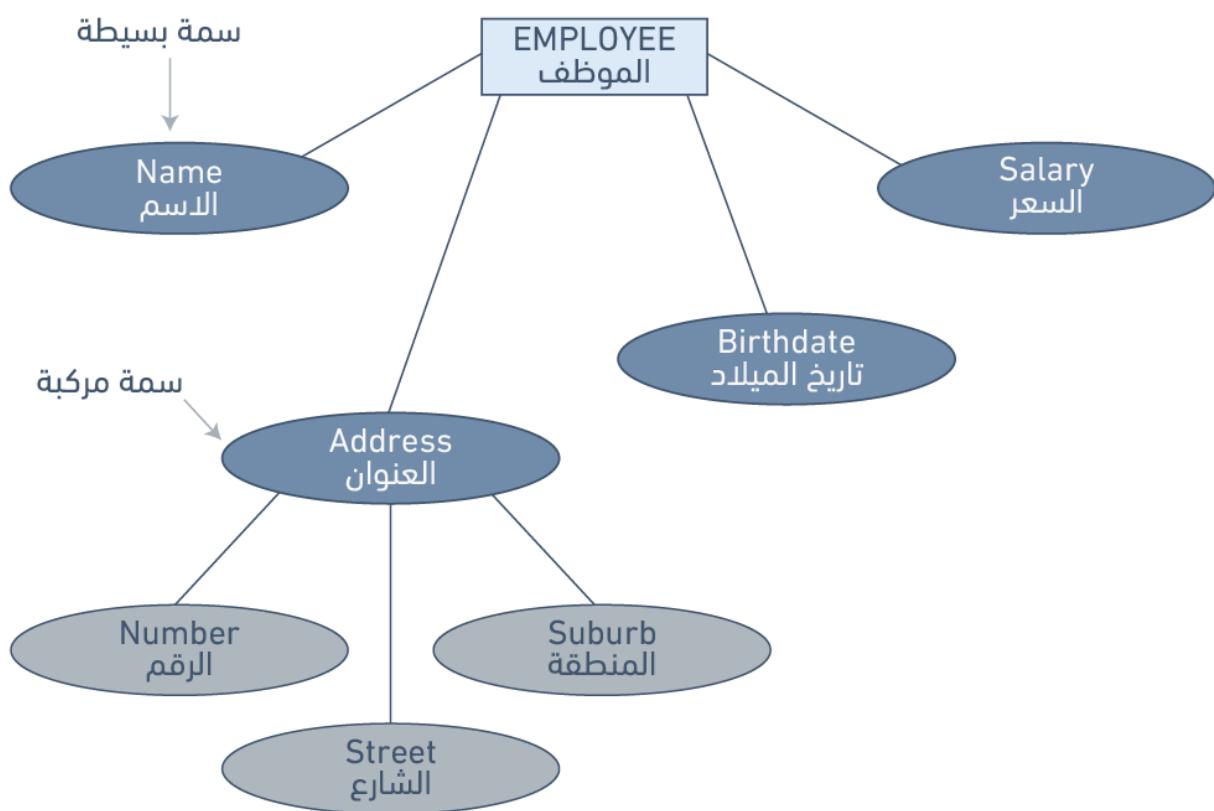
### 6.4.2 السمات البسيطة

السمات البسيطة Simple attributes هي السمات المستمدّة من مجالات القيمة الذريّة، ويطلق عليها أيضًا اسم السمات وحيدة القيمة single-valued، فمثلاً، تجد في قاعدة بيانات الشركة COMPANY أن الاسم والعنوان هما نموذجان للسمات البسيطة.

### 6.4.3 السمات المركبة

السمات المركبة Composite attributes هي التي تتكون من مجموعة متسلسلة هرميًّا من السمات، فمثلاً، قد يتكون العنوان باستخدام مثال قاعدة البيانات الشركة COMPANY والموضح في الشكل التالي، من

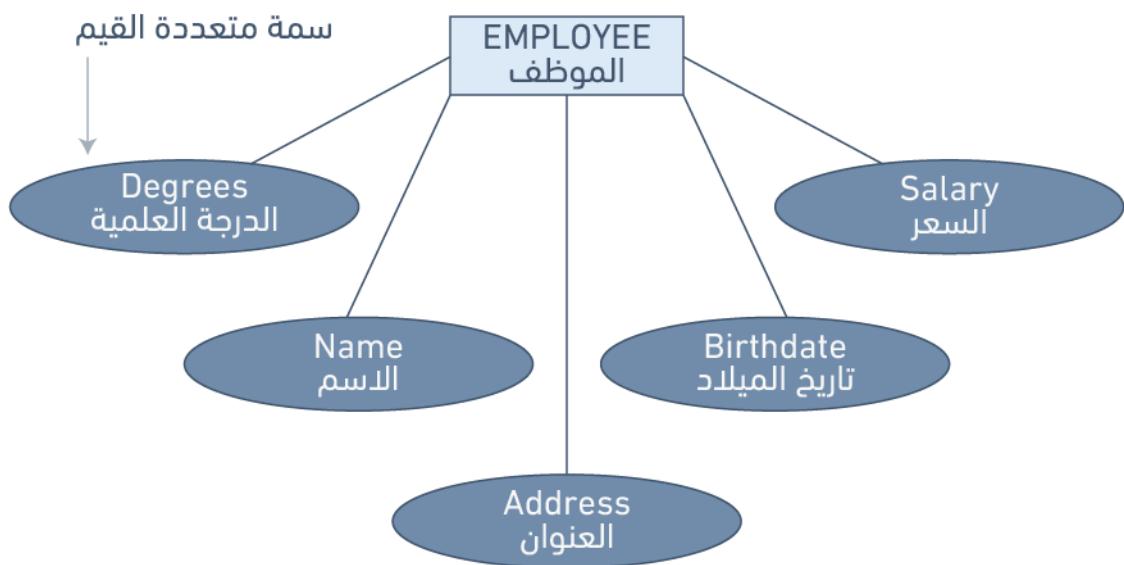
رقم الشارع، واسم الشارع، واسم الحي، حيث يُمثّل بالطريقة التالية: العنوان = {59 + "شارع خالد بن الوليد" + "حي القنوات" }.



الشكل 6.3: مثال للسمات المركبة

#### 6.4.4 السمات متعددة القيم

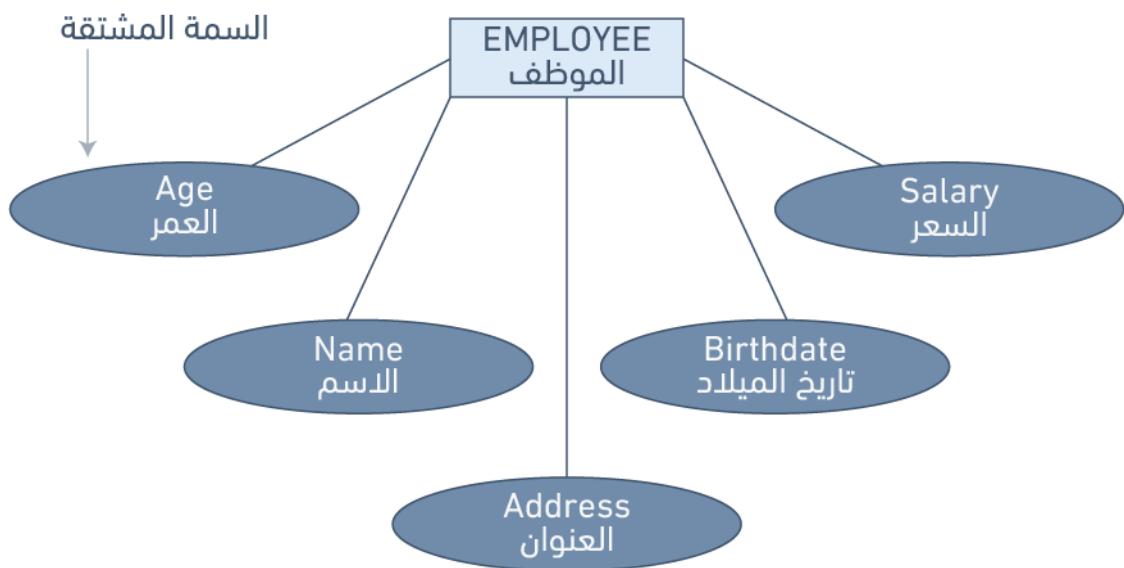
السمات متعددة القيم Multivalued attributes هي التي تحمل مجموعةً من القيم لكل كيان، فمثلاً، يمكن أن تحمل سمة الدرجات العلمية degrees لموظفي معين في قاعدة بيانات الشركة COMPANY العديد من القيم، مثل: دكتوراه PhD، الجامعة العربية للعلوم، إجازة في العلوم BSc كما هو موضح في الشكل التالي:



الشكل 4.4: مثال على السمات متعددة القيم

#### 6.4.5 السمات المشتقة

السمات المشتقة Derived attributes هي سمات تحتوي على قيم محسوبة من سمات أخرى، فمثلاً، يمكن اشتقاق العمر في الشكل التالي من تاريخ الميلاد، يسمى تاريخ الميلاد في هذه الحالة سمة مخزنة stored attribute، وهي التي تحفظ مادياً لقاعدة البيانات.



الشكل 4.5: مثال على السمات المشتقة

## 6. المفاتيح

يُعد المفتاح key أحد القيود المهمة التي يجب وجودها في جميع الكيانات، وهو عبارة عن سمة أو مجموعة من السمات التي تُستخدم قيمها لتعريف كيان منفصل individual entity تعرّفًا فريديًّا في مجموعة الكيانات.

### 6.5.1 أنواع المفاتيح

هناك عدة أنواع من المفاتيح، نذكر منها:

#### أ. المفتاح المرشح

يُعد المفتاح المرشح key مفتاحًا بسيطًا أو مركبًا، كما يكون فريديًّا وبسيطًا، وهو فريد لأنّه لا يمكن أن يكون لصفين المفتاح المرشح نفسه في الجدول في أيّ وقت، فمثلاً، تكون المفاتيح المرشحة الممكنة في كيان الموظف الموجود في قاعدة البيانات COMPANY، والذي يتكون من السمات التالية: معرف الموظف، الاسم الأول، اسم العائلة، رقم التأمين الاجتماعي SIN، العنوان، الهاتف، تاريخ الميلاد، الراتب، معرف القسم، هي ما يلي:

- رقم التأمين الاجتماعي SIN، أو معرف الموظف EID.
- الاسم الأول واسم العائلة، بافتراض عدم وجود شخصين في الشركة لهما الاسم نفسه.
- اسم العائلة ومعرف القسم، بافتراض عدم عمل شخصين لهما اسم العائلة نفسه في القسم نفسه.

#### ب. المفتاح المركب

يتكون المفتاح المركب key من سمتين أو أكثر، ويحسن الإبقاء على الحد الأدنى من السمات فيه. باستخدام المثال السابق نفسه، تكون المفاتيح المركبة الممكنة هي:

- الاسم الأول واسم العائلة، بافتراض عدم وجود شخصين في الشركة لهما الاسم نفسه.
- اسم العائلة ومعرف القسم، بافتراض عدم عمل شخصين لهما اسم العائلة نفسه في القسم نفسه.

#### ج. المفتاح الرئيسي

المفتاح الرئيسي primary key هو مفتاح مرشح key يُحدّد بواسطة مصمم قاعدة البيانات لاستخدامه على أساس آلية تعريف لمجموعة الكيانات بأكملها، كما يجب أن يُحدّد أسطر الجدول تحديديًّا، ولا يمكن تركه فارغاً.

يُشار إلى المفتاح الرئيسي في نموذج الكيان وال العلاقة ER عن طريق وضع خط تحت السمة التي تمثله.

- يُحدّد مفتاح مرشح بواسطة مصمم قاعدة البيانات لتحديد أسطر الجدول تحديداً فريداً، ولا يمكن تركه فارغاً.
- يختار مفتاح معين من قبل مصمم قاعدة البيانات لاستخدامه على أساس آلية تعريف لمجموعة الكيانات بأكملها، ويُشار إلى هذا المفتاح بالمفتاح الرئيسي primary key، كما يُشار إليه عن طريق وضع خط تحت السمة الممثلة له في نموذج الكيان وال العلاقة ER model.
- إذا أخذنا المثال التالي، يتكون كل صفت في جدول الموظف من (ID، الاسم الأول، اسم العائلة، SIN، العنوان، الهاتف، تاريخ الميلاد، الراتب، معرف القسم)، فإن المفتاح الرئيسي هو ID.

#### د. المفتاح الثانوي

المفتاح الثانوي secondary key هو سمة تُستخدم استخداماً صارماً لأغراض الاسترجاع، ويمكن أن يكون هذا المفتاح مركباً من عدة سمات مثل أن يتكون من الهاتف واسم العائلة معاً.

#### هـ. المفتاح البديل

المفاتيح البديلة keys Alternate هي المفاتيح المرشحة التي لم تُستخدم على أساس مفتاح رئيسي.

#### وـ. المفتاح الخارجي

يُعد المفتاح الخارجي foreign key - أو FK اختصاراً- سمة موجودة في جدول معين بحيث تشير إلى المفتاح الرئيسي في جدول آخر، أو يمكن تركه فارغاً، ويجب أن تكون كل من المفاتيح الخارجية والرئيسية من نوع البيانات نفسه، فمثلاً يُمثل مُعرف القسم DepartmentID المفتاح الخارجي ضمن قاعدة بيانات الشركة COMPANY، أي كما يلي:

- جدول الموظف Employee (معرف الموظف ID، الاسم الأول، اسم العائلة، رقم التأمين الاجتماعي SIN، العنوان، الهاتف، تاريخ الميلاد، الراتب، معرف القسم).

#### 6.5.2 القيم الفارغة Nulls

تُعد القيمة الفارغة Null رمزاً خاصاً ليس له علاقة بنوع بيانات محددة، مما يعني أنه إما غير معروف unknown أو غير قابل للتطبيق inapplicable، ولا يعني صفرًا أو فراغاً، ومن صفات هذه القيمة:

- لا توجد بيانات محددة لإدخالها.
- لا يمكن تواجدها في المفتاح الرئيسي.
- يجب تجنبها في جميع السمات الأخرى.
- يمكنها تمثيل ما يلي:

- قيمة سمة غير معروفة.
- قيمة سمة معروفة، ولكنها مفقودة.
- شرط "غير قابل للتطبيق".
- يمكنها تسبّب في العديد من المشاكل عند استخدام بعض الدوال، مثل: COUNT و AVERAGE و SUM.
- يمكنها تسبّب في مشاكل منطقية عند ربط الجداول العلائقية بعضها البعض.

تكون نتيجة عملية الموازنة القيمة الفارغة null عندما يكون أحد الحدود قيمةً فارغةً null، كما تكون النتيجة قيمةً فارغةً null في العمليات الحسابية إذا كان أحد الحدود قيمةً فارغةً null باستثناء الدوال التي تتجاهل هذه القيمة.

### 6.5.3 مثال لكيفية استخدام القيمة الفارغة null

استخدم جدول الرواتب Salary\_tbl لمعرفة كيفية استخدام القيمة الفارغة null.

emp#	JobName	Salary	Commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

على أساس خطة أولى، أبدأ بإيجاد جميع الموظفين في عمود الموظف # emp في قسم المبيعات Sales تحت عمود اسم الوظيفة jobName، ومن تزيد رواتبهم salary مع عمولتهم commission عن 30000.

```
SELECT emp# FROM Salary_tbl
WHERE jobName = Sales AND
      (commission + salary) > 30,000
```

يكون ناتج العملية أعلاه الموظفين E10، E12، إذ لا تتضمن هذه النتيجة الموظف E13 بسبب القيمة الفارغة null في عمود العمولة commission، حيث ستكون النتيجة القيمة الفارغة null عند جمع الراتب مع العمولة، لذا سنحتاج إلى إلقاء نظرة على الحقول بصورة منفصلة للتأكد من تضمين الصيغة الذي يحتوي على القيمة الفارغة null، كما هو مبين في الحل أدناه.

```
SELECT emp# FROM Salary_tbl
WHERE jobName = Sales AND
      (commission > 30000 OR
       salary > 30000 OR
       (commission + salary) > 30,000)
```

سيكون ناتج العملية أعلاه هو الموظفين E10 و E12 و E13 .

## 6.5.4 العلاقات

العلاقات Relationships هي الرابط الذي يربط الجداول بعضها البعض في قاعدة البيانات، وتُستخدم لربط المعلومات ذات الصلة بين الجداول.

تعتمد قوة العلاقة Relationship strength على كيفية تعريف المفتاح الرئيسي للكيانات المترابطة، إذ تُعد العلاقة ضعيفة weak، أو غير محددة non-identifying إذا كان المفتاح الرئيسي للكيان المرتبط لا يحتوي على المفتاح الرئيسي للكيان الأب parent entity. وتتضمن قاعدة بيانات الشركة Company بعض الأمثلة التالية:

• جدول العميل Customer يحوي الحقول التاليين:

◦ رقم العميل CustID

◦ اسم العميل CustName

• جدول الطلب Order يحوي الحقول التالية:

◦ رقم الطلب OrderID

◦ رقم العميل CustID

◦ تاريخ الطلب Date

يحتوي المفتاح الرئيسي للكيان المرتبط في العلاقة القوية أو المحددة على المفتاح الرئيسي للكيان الأب، مثل التالي:

• جدول الدورة التدريبية Course يحوي الحقول التالي:

◦ رمز الدورة CrsCode

◦ رمز القسم DeptCode

◦ وصف الدورة Description

• جدول الصف Class يحوي الحقول التالية:

◦ رمز الدورة CrsCode

◦ القسم Section

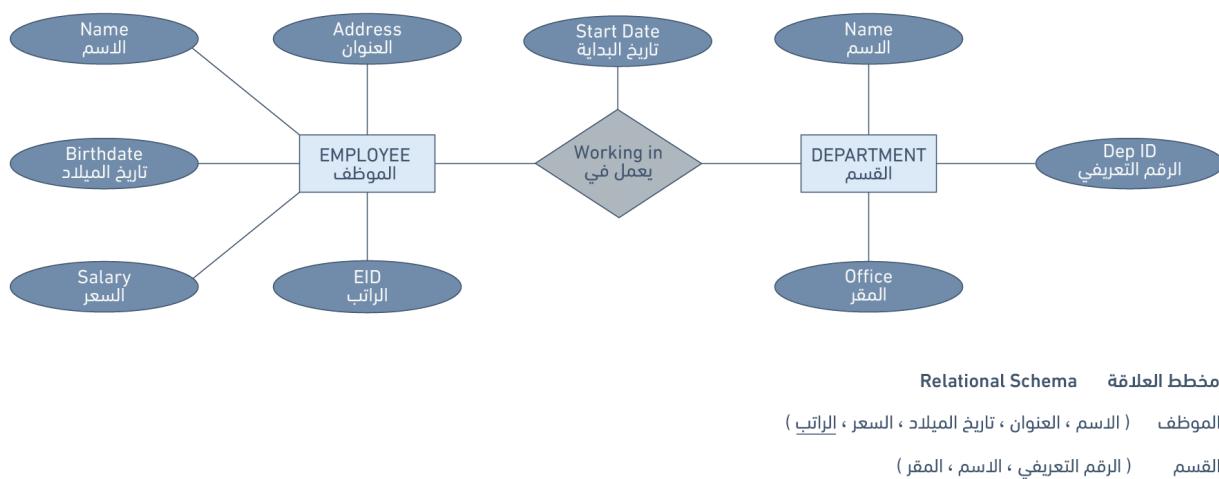
◦ وقت الصف ... إلخ ClassTime

## 6.5.5 أنواع العلاقات

هناك عدة أنواع من العلاقات منها:

### أ. علاقة واحد إلى متعدد

تُعدّ علاقة واحد إلى متعدد one to many - أو 1:M اختصاراً- الأساس في أي تصميم لقواعد البيانات العلاقة، وتوجد في جميع بيئات قواعد البيانات العلاقة، فمثلاً، يحتوي القسم الواحد على العديد من الموظفين، ويوضح الشكل التالي علاقة أحد هؤلاء الموظفين بالقسم.



الشكل 6.6: علاقة واحد إلى متعدد

### ب. علاقة واحد إلى واحد

تُعدّ علاقة واحد لواحد one to one - أو 1:1 اختصاراً- علاقة كيان واحد بكيان واحد آخر فقط، والعكس صحيح.

يُعدّ هذا النوع من العلاقات نوعاً نادراً جدًا في تصميم قواعد البيانات العلاقة، ومن الممكن أن يشير وجود هذه العلاقة إلى انتماء كيائين بالفعل إلى الجدول نفسه، فمثلاً، يكون الموظف في قاعدة بيانات الشركة COMPANY مرتبطاً بزوجة واحدة، وتكون الزوجة الواحدة مرتبطةً بموظف واحد.

### ج. علاقة متعدد إلى متعدد

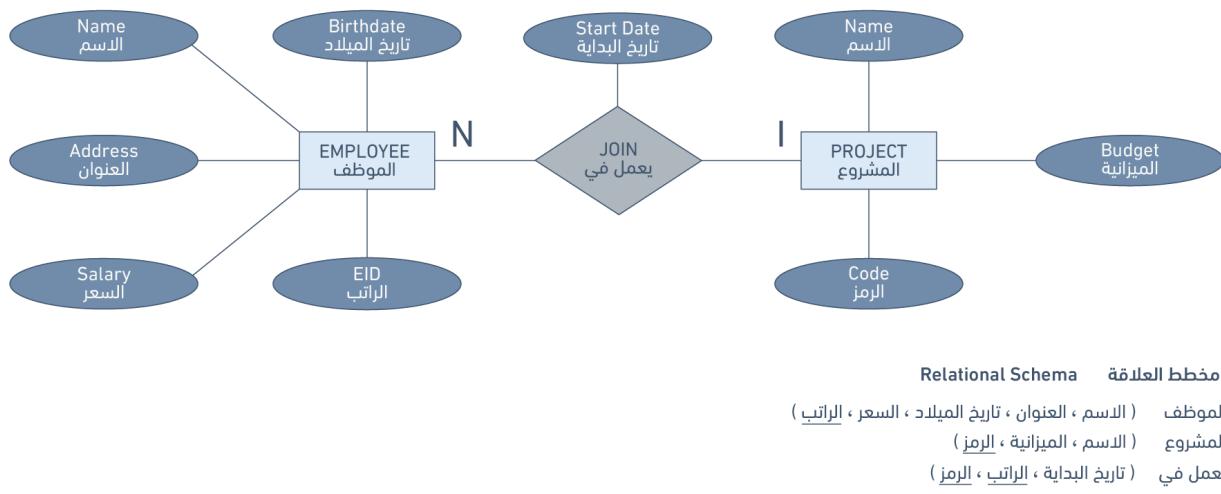
ضع في بالك النقاط التالية عند التعامل مع علاقة متعدد إلى متعدد many to many - أو M:N اختصاراً:-

- لا يمكن تمثيلها بهذه الصورة - أي متعدد إلى متعدد- في النموذج العلائقى relational model.
- يمكن تحويلها إلى علاقاتين من النوع واحد إلى متعدد.
- يمكن تنفيذها عن طريق كسرها لمجموعة علاقات من نوع واحد إلى متعدد.

- تنتهي على تنفيذ كيانات مركبة.
- تنشأ علاقاتين أو أكثر من النوع واحد إلى متعدد.
- يجب أن يحتوي جدول الكيان المركب على المفاتيح الرئيسية للجداول الأصلية على الأقل.
- يحتوي جدول الرابط على تكرارات متعددة لقيم المفتاح الخارجي.
- قد تُسند سمات إضافية حسب الحاجة.

يمكنك تجنب المشاكل الموجدة في علاقة متعدد إلى متعدد عن طريق إنشاء كيان مركب composite entity، أو كيان جسري bridge entity، فمثلاً، يمكن للموظف العمل في العديد من المشاريع، أو يمكن أن يعمل في المشروع الواحد العديد من الموظفين، اعتماداً على قواعد العمل؛ أو يمكن للطالبأخذ العديد من الدروس، كما يمكن للدرس الواحد أن يؤخذ بواسطة العديد من الطلاب.

يوضح الشكل التالي جائياً آخرًا من علاقة M:N، حيث يكون للموظف العديد من توارikh البداية المتعلقة بمشاريع مختلفة، لذلك نحتاج إلى جدول ربط JOIN بحيث يحتوي على معرف الموظف EID، والرمز Code، وتاريخ البداية StartDate.



الشكل 6.7: للموظف العديد من توارikh البدء المتعلقة بمشاريع مختلفة

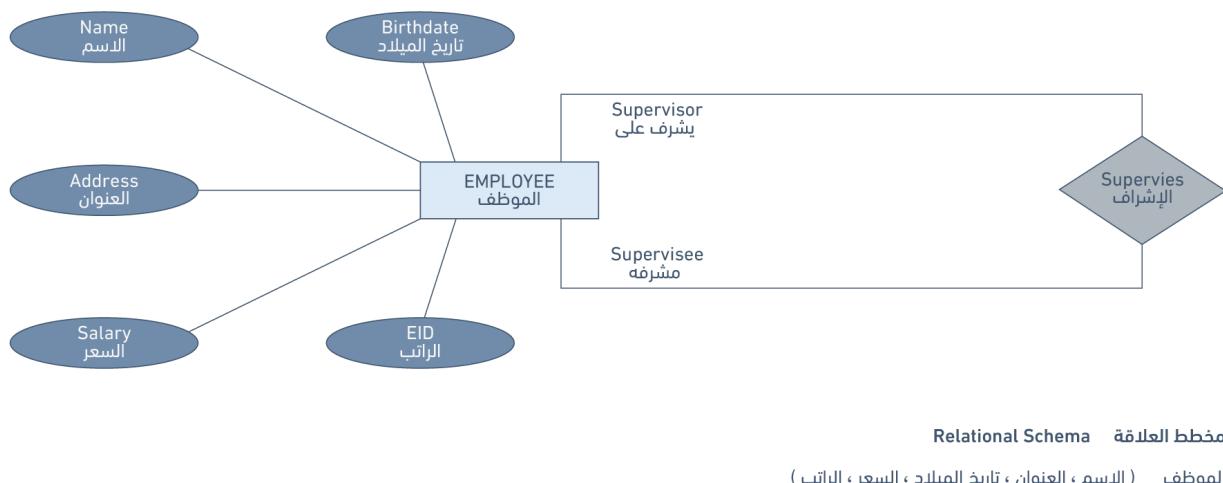
إليك مثال على تعيين علاقة ثنائية من نوع متعدد إلى متعدد M:N:

- حدد علاقاتين لكل علاقة ثنائية من نوع متعدد إلى متعدد، حدد علاقاتين.
- تمثل A، و B نوعين من الكيانات المشاركة في R.
- أنشئ علاقة جديدة S لتمثيل R.

- تحتاج S أن تتضمن المفاتيح الرئيسية الخاصة بـ A، و B، حيث يمكن أن تكون هذه مـا المفتاح الرئيسي في الجدول S، أو يمكن أن تضاف لها سمة بسيطة أخرى في الجدول الجديد R لتكوين المفتاح الرئيسي.
- تكون مجموعة المفاتيح الرئيسية لـ A، و B المفتاح الرئيسي لـ S.

#### د. العلاقة الأحادية -أو التكرارية-

العلاقة الأحادية Unary relationship -والتي تسمى العلاقة التكرارية recursive أيضاً- هي العلاقة التي توجد فيها علاقة بين تكرارات مجموعة الكيانات نفسها، ويكون في هذه العلاقة المفاتيح الرئيسي والخارجي متماثلين لكنهما يمثلان كيانين لهما أدوار مختلفة. تُعدّ مجموعة الكيان مجموعةً من نوع مماثل من الكيانات.



الشكل 6.8: العلاقات الأحادية (التكرارية)

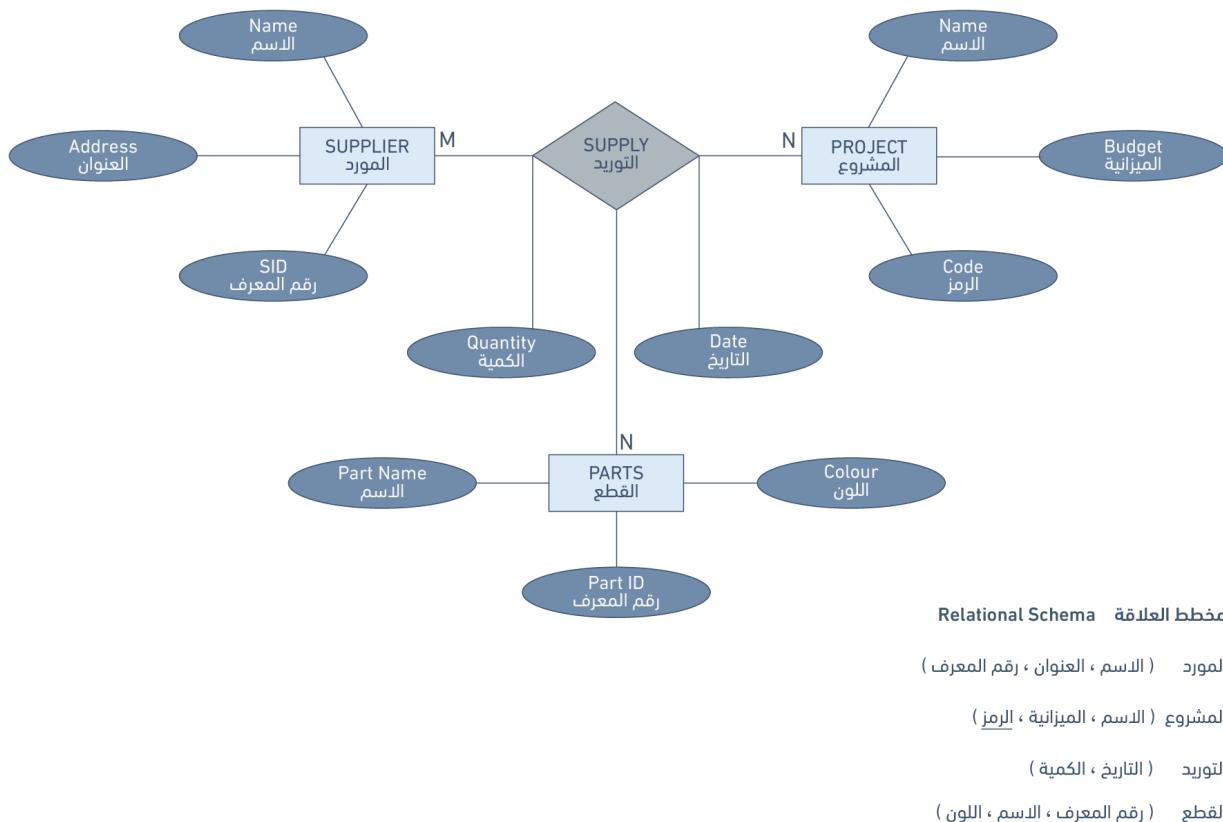
يمكن إنشاء عمود منفصل بحيث يشير إلى المفتاح الرئيسي لمجموعة الكيان نفسها في بعض كيانات العلاقة الأحادية.

#### ٥. العلاقة الثلاثية n-ary

العلاقة الثلاثية ternary relationship هي نوع من العلاقات يضمن إنشاء علاقة متعدد إلى متعدد إلى متعدد بين ثلاثة جداول، والشكل التالي هو مثال عن هذه العلاقة.

يشير مصطلح n-ary إلى جداول متعددة في العلاقة، وتذكر أنّ N تكافئ many أي  $N = \text{many}$ .

- لكل علاقة (n-ary) حيث  $2 < n$ ، أنشئ جدولًا جديداً لتمثيل تلك العلاقة.
- المفتاح الرئيسي للجدول الجديد هو مزيج من المفاتيح الرئيسية للكيانات المشاركة التي تمثل الجانب المتعدد N.
- تملك جميع الكيانات المشاركة في معظم حالات علاقة n-ary الطرف المتعدد من جانبها.



الشكل 6.9: العلاقة الثلاثية

## 6.6 مصطلحات أساسية

- **المفتاح البديل alternate key:** المفاتيح البديلة هي جميع المفاتيح المرشحة التي لم تُستخدم على أساس مفتاح رئيسي.
- **المفتاح المرشح candidate key:** هو مفتاح بسيط أو مركب يتصف بأنه فريد أي لا يمكن أن يتكرر وجوده في سطرين وأيضاً أدنى minimal أي وجوده ضروري ومطلوب في كل الأعمدة المشتركة فيه.
- **الكيانات المميزة characteristic entities:** هي الكيانات التي توفر مزيداً من المعلومات حول جدول آخر.
- **السمات المركبة composite attributes:** هي السمات التي تتكون من العديد من السمات المتسلسلة هرمياً.
- **المفتاح المركب composite key:** يتكون المفتاح المركب من سمتين أو أكثر، ويحسن الإبقاء على الحد الأدنى من السمات فيه.
- **الكيانات المعتمدة dependent entities:** هي الكيانات التي تعتمد على جداول أخرى حتى يكون لها معنى.

- **السمات المشتقة derived attributes**: هي التي تحتوي على قيم محسوبة من سمات أخرى.
- **البيانات المشتقة derived entities**: انظر الكيانات المعتمدة.
- **EID**: مُعرّف الموظف.
- **الكيان entity**: شيء ما أو كائن ما في العالم الحقيقي، وله وجود مستقل، ويمكن تمييزه عن الكائنات الأخرى.
- **نموذج الكيان وال العلاقة لتمثيل البيانات أو ER**: يسمى أيضاً تخطيط الكيان وال العلاقة، و يُمثل بواسطة مخططات الكيان وال العلاقة، والتي هي مناسبة تماماً لنمذجة البيانات للاستخدام مع قواعد البيانات.
- **تخطيط الكيان وال العلاقة entity relationship schema**: انظر إلى نموذج الكيان وال العلاقة لتمثيل البيانات.
- **مجموعة الكيان entity set**: تجميعة من الكيانات من النوع نفسه في وقت معين.
- **نوع الكيان entity type**: تجميعة من الكيانات المتشابهة.
- **المفتاح الخارجي أو foreign key**: هو سمة موجودة في جدول معين بحيث تشير إلى المفتاح الرئيسي في جدول آخر، أو يمكن تركه فارغاً null.
- **الكيان المستقل independent entity**: الكيانات المستقلة هي للبنات الرئيسية لبناء قواعد البيانات، ولا يعتمد وجودها على أي كيانات أخرى.
- **النواة kernel**: انظر الكيان المستقل independent entity.
- **المفتاح key**: سمة أو مجموعة من السمات التي تُستخدم قيمها لتعريف كيان منفصل individual entity تعريفاً فريداً في مجموعة الكيانات.
- **السمات متعددة القيم multivalued attributes**: هي التي لها مجموعة من القيم لكل كيان.
- **n-ary**: علاقـة بين جداول متعددة.
- **القيمة الفارغة null**: رمزاً خاصاً ليس له علاقة بنوع بيانات محدد، مما يعني أنه إما غير معروف unknown أو غير قابل للتطبيق inapplicable، ولا يعني صفرًا أو فراغاً
- **العلاقات relationships**: هي الرابط الذي يربط الجداول بعضها البعض في قاعدة البيانات، و تُستخدم لربط المعلومات ذات الصلة بين الجداول.
- **قوة العلاقة relationship strength**: تُحدّد قوـة العلاقة استناداً إلى كيفية تعريف المفتاح الرئيسي للبيانات ذات الصلة.

- **المفتاح الثانوي secondary key:** هو سمة تُستخدم استخداً مصارماً لأغراض الاسترجاع.
- **السمات البسيطة simple attributes:** مستمدّة من مجالات القيم الذرية.
- **SIN:** رقم التأمين الاجتماعي.
- **السمات أحادية القيمة single-valued attributes:** انظر السمات البسيطة.
- **السمة المخزنة stored attribute:** هي السمة التي تُحفظ مادياً في قاعدة البيانات.
- **العلاقة الثلاثية ternary relationship:** هي نوع من العلاقات التي تضمن إنشاء علاقة متعدد إلى متعدد بين ثلاثة جداول.
- **العلاقة الأحادية unary relationship:** هي العلاقة التي توجد فيها علاقة بين تكرارات مجموعة الكيانات نفسها وتدعى أيضاً علاقة تكرارية recursive relationship.

## 6.7 تمارين

1. ما المفهومان اللذان يعتمد عليهما نموذج الكيان وال العلاقة ER؟
2. تتكون قاعدة البيانات التالية من جدولين، لذلك أجب على الأسئلة التالية عبر استخدامها:
  - حدد المفتاح الرئيسي لكل جدول.
  - حدد المفتاح الخارجي في الجدول PLAY.
  - حدد المفاتيح المرشحة في كلا الجدولين.
  - ارسم نموذج الكيان وال العلاقة ER.
  - هل يحقق الجدول PLAY سلامًّا مرجعيةً؟ ولم؟ أو لم لا؟

الجدول :DIRECTOR

DIRNUM	DIRNAME	DIRDOB
100	J.Broadway	01/08/39
101	J.Namath	11/12/48
102	W.Blake	06/15/44

## الجدول :PLAY

PLAYNO	PLAYNAME	DIRNUM
1001	Cat on a cold bare roof	102
1002	Hold the mayo, pass the bread	101
1003	I never promised you coffee	102
1004	Silly putty goes to Texas	100
1005	See no sound, hear no sight	101

3. عرف المصطلحات التالية، حيث قد تحتاج إلى البحث في الإنترت.

- المخطط .schema
- لغة المضيف .host language
- اللغات الفرعية للبيانات .data sublanguage
- لغة تعريف البيانات .data definition language
- العلاقة الأحادية .unary relation
- المفتاح الخارجي .foreign key
- العلاقة الافتراضية .virtual relation
- الرابط .connectivity
- المفتاح المركب .composite key
- جداول الرابط .linking table

4. تضمن قاعدة بيانات شركة PRE الجداول الثلاثة أدناه، لذلك أجب عن الأسئلة التالية مستخدماً إياها:

- حدد المفتاح الرئيسي والمفتاح الخارجي في كل جدول.
- هل يحقق الجدول TRUCK سلاماً مرجعيةً؟ اشرح إجابتك.
- ما نوع العلاقة بين الجدولين TRUCK، و BASE.
- كم عدد الكيانات في الجدول TRUCK.
- حدد المفاتيح المرشحة في الجدول TRUCK.

## الجدول :TRUCK

TNUM	BASENUM	TYPENUM	TMILES	TBOUGHT	TSERIAL
1001	501	1	5900.2	11/08/90	as-125
1002	502	2	64523.9	11/08/90	ac-213
1003	501	2	32116.0	09/29/91	ac-215
1004		2	3256.9	01/14/92	ac-315

## الجدول :BASE

BASENUM	BASECITY	BASESTATE	BASEPHON	BASE MGR
501	Dallas	TX	893-9870	J. Jones
502	New York	NY	234-7689	K. Lee

## الجدول :TYPE

TYPENUM	TYPEDESC
1	single box, double axle
2	tandem trailer, single axle

5. لنفترض أنك تستخدم قاعدة البيانات أدناه والمكونة من جدولين، أجب عن الأسئلة الآتية:

- حدد المفتاح الرئيسي في كل جدول.
- حدد المفتاح الخارجي في الجدول BookOrders.
- هل هناك أي مفاتيح مرشحة في أي من الجدولين؟
- ارسم نموذج الكيان وال العلاقة ER.
- هل يحقق الجدول BookOrders السلامة المرجعية؟
- هل تحتوي الجداول على بيانات مكررة؟ ما هي؟

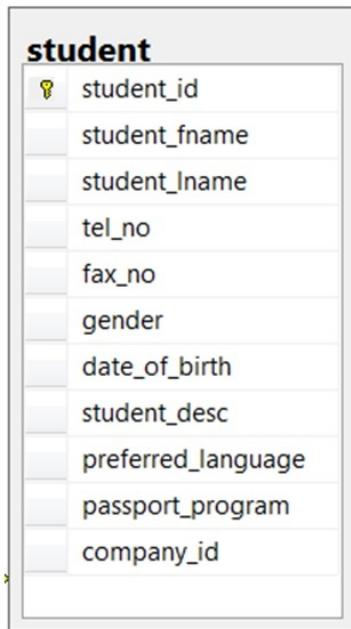
## الجدول :Customer

CustID	CustName	AccntNo
100	Joe Smith	010839
101	Andy Blake	111248
102	Sue Brown	061544

## الجدول :BookOrders

OrderID	Title	CustID	Price
1001	The Dark Tower	102	12.00
1002	Incubus Dreams	101	19.99
1003	Song of Susannah	102	23.00
1004	The Time Traveler's Wife	100	21.00
1005	The Dark Tower	101	12.00
1006	Tanequil	102	15.00
1007	Song of Susannah	101	23.00

6. بالنظر إلى جدول الطالب الموضح في الشكل أدناه، اكتب قائمة بجميع المفاتيح المرشحة الممكنة، واذكر سبب اختيارك لكل واحد من المفاتيح.

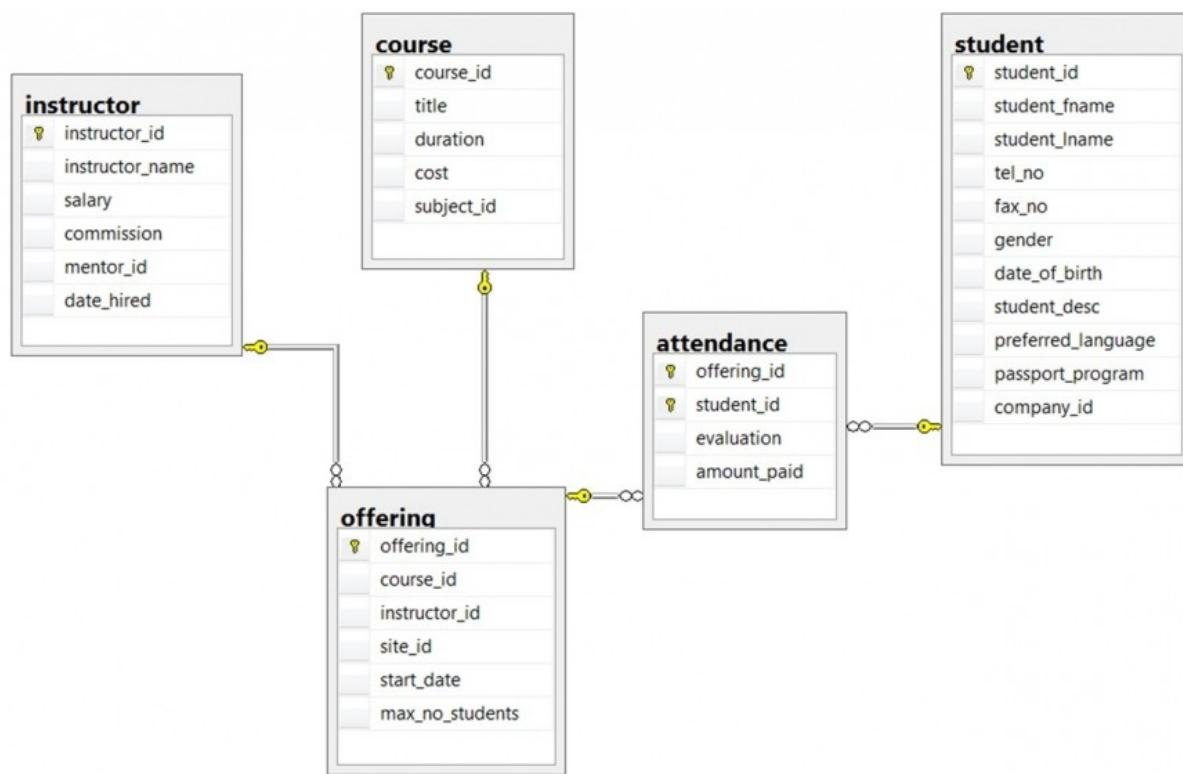


الشكل 6.10: السؤال السادس

7. أجب عن الأسئلة التالية مستخدماً مخطط الكيان وال العلاقة ERD لقاعدة بيانات المدرسة الموضحة في الشكل أدناه.

- حدد جميع الأنوية والكيانات المعتمدة والمميزة في مخطط الكيان وال العلاقة ERD.
- أي من الجداول لها علاقات ضعيفة، وأيها لديها علاقات قوية؟
- بالنظر إلى كل جدول من جداول قاعدة بيانات المدرسة، ما هي السمات التي يمكنها أن تأخذ القيمة الفارغة Null، ولماذا؟

- ما هي الجداول التي أنشئت على أساس نتيجة لعلاقات متعدّد إلى متعدّد؟



الشكل 6.11: السؤال السابع

# 7. قواعد السلامة وقيودها لضمان سلامة البيانات

تُعدّ القيود Constraints ميزةً مهمةً جدًا في النموذج العلائقى relational model الذي يدعم نظريةً محددةً للقيود المطبقة على السمات attributes أو الجداول tables، كما تُعدّ هذه القيود مفيدةً بسبب سماحها لمصمم قواعد البيانات بتحديد دلالات semantics البيانات، فهذه القيود هي القواعد التي تجبر نظم إدارة قواعد البيانات DBMSs اختصاراً- على التحقق من توافق البيانات مع Database management systems هذه الدلالات.

## 7.1 سلامة النطاق Domain Integrity

يُعدّ النطاق قيداً من قيود النموذج العلائقى، حيث يُقيّد قيم السمات في العلاقة، لكن هناك دلالات واقعية للبيانات التي لا يمكن تحديدها إذاً استخدمناها مع قيود النطاق فقط، لذلك نحتاج إلى طرقٍ أكثر تحديداً لبيان قيم البيانات المسموح بها أو غير المسموح بها والتنسيق المناسب لكل سمة، فمثلاً، يجب أن يكون معرف الموظف Employee ID أو EID اختصاراً- في قاعدة البيانات فريداً، أو يجب أن يكون تاريخ ميلاد الموظف Birthdate ضمن المجال [Jan 1, 1950 - Jan 1, 2000]، حيث توفر هذه المعلومات ضمن تعليمات منطقية تسمى قيود السلامة integrity constraints، ويوجد عدة أنواع من قيود السلامة كما هو موضح أدناه.

### 7.1.1 سلامة الكيان Entity integrity

يجب احتواء كل جدول على مفتاح رئيسي primary key لضمان سلامة الكيان، ولا يمكن احتواء المفتاح الرئيسي PK أو أي جزء منه على قيم فارغة null، حيث لا يمكننا تحديد بعض الصفوف rows عندما تكون قيم المفتاح الرئيسي فارغة، فمثلاً، لا يمكن أن يكون الهاتف Phone في جدول الموظف EMPLOYEE مفتاحاً رئيسياً نظراً لعدم امتلاك بعض الأشخاص أيّ هاتف.

## 7.1.2 السلامة المرجعية Referential integrity

تتطلب السلامة المرجعية وجود مفتاح رئيسي مقابل للمفتاح الخارجي *foreign key*, وإلا فيجب أن يكون فارغاً. ويُحدّد هذا القيد بين جدولين أي الجدول الأب والجدول الابن؛ حيث يحافظ على المطابقة بين الصنوف في هذين الجدولين، وهذا يعني أن المرجع من صفي في جدول إلى جدول آخر يجب أن يكون صالحًا.

فيما يلي مثال على قيود السلامة المرجعية في قاعدة بيانات العملاء/الطلبات Customer/Order الخاصة بالشركة Company:

• جدول العميل Customer يحوي الحقول التاليين:

◦ رقم العميل CustID

◦ اسم العميل CustName

• جدول الطلب Order يحوي الحقول التالية:

◦ رقم الطلب OrderID

◦ رقم العميل CustID

◦ تاريخ الطلب Date

يجب فرض السلامة المرجعية لضمان عدم وجود سجلات معزولة أو يتيمة orphan records، فالسجل المعزول هو السجل الذي تكون قيمة مفتاحه الخارجي FK غير موجودة في الكيان المقابل -أي الكيان الذي يحوي المفتاح الرئيسي PK، وتذكّر أنّ عملية الضم join تكون بين المفتاحين الرئيسي PK والخارجي FK.

ينص قيد السلامة المرجعية في المثال السابق على وجوب تطابق CustID في جدول الطلبات Order مع CustID صالح في جدول العملاء Customer.

تملك معظم قواعد البيانات العلاقة سلامة مرجعية تصريحية declarative referential integrity، أي يجري إعداد قيود السلامة المرجعية عند إنشاء الجداول.

فيما يلي مثال آخر على قاعدة بيانات صنوف دراسية/مقررات Course/Class:

• جدول الدورة التدريبية Course يحوي الحقول التالي:

◦ رمز الدورة CrsCode

◦ رمز القسم DeptCode

◦ وصف الدورة Description

• جدول الصف Class يحوي الحقول التالية:

ـ رمز الدورة CrsCode

ـ القسم Section

ـ وقت الصف ClassTime

ينص قيد السلامة المرجعية هنا على وجوب تطابق المفتاح الخارجي CrsCode في جدول Class مع مفتاح رئيسى CrsCode صالح في جدول Course، حيث لا يكفي في هذه الحالة أن تشكل السمتان و Section في جدول Class المفتاح الرئيسي PK، إذ يجب علينا فرض السلامة المرجعية.

يجب امتلاك المفتاح الرئيسي PK والمفتاح الخارجي FK أنواع البيانات نفسها، كما يجب أن يأتيا من نفس النطاق عند إعداد السلامة المرجعية، وإلا فلن يسمح نظام إدارة قاعدة البيانات العلائقية RDBMS بعملية [join](#).

يُعدّ نظام RDBMS نظام قاعدة بيانات شائع، حيث يعتمد على النموذج العلائقى الذى قدمه إدجار كود (E.F.) (Codd) من مختبر أبحاث سان خوسيه (San Jose) التابع لشركة IBM، كما تُعدّ أنظمة قواعد البيانات العلائقية أسهل في الاستخدام والفهم من أنظمة قواعد البيانات الأخرى.

### 7.1.3 السلامة المرجعية في نظام مايكروسوفت أكسس

يجري إعداد السلامة المرجعية في نظام مايكروسوفت أكسس MS Access من خلال ضم المفتاح الرئيسي PK الذي هو معرف العميل CustID في جدول العملاء إلى معرف العميل CustID في جدول الطلبات Order، ويوضح الشكل التالي طريقة عمل ذلك على شاشة تحرير العلاقات Edit Relationships في نظام مايكروسوفت أكسس:



### 7.1.4 السلامة المرجعية في الإصدار Transact-SQL من لغة SQL

تُضيّط السلامة المرجعية في الإصدار Transact-SQL (اختصاراً T-SQL) المستخدمة في خادم MS SQL Server عند إنشاء جدول الطلبات Order باستخدام المفتاح الخارجي FK، حيث تُظهر التعليمات المدرجة أدناه المفتاح الخارجي FK في جدول الطلبات Order الذي يكون مرجعاً إلى المفتاح الرئيسي PK في جدول العملاء Customer:

```
CREATE TABLE Customer
( CustID INTEGER PRIMARY KEY,
  CustName CHAR(35) )
```

```
CREATE TABLE Orders
( OrderID INTEGER PRIMARY KEY,
  CustID INTEGER REFERENCES Customer(CustID),
  OrderDate DATETIME )
```

### 7.1.5 قواعد المفاتيح الخارجية

يمكن إضافة قواعد مفاتيح خارجية إضافية عند ضبط السلامة المرجعية مثل ما نفعله بالصفوف الأبناء -في جدول Orders- عندما يُحذف أو يُغيّر -أي يُحذف- السجل وهو جزء من الجدول الأب -Customer- والذي يملك مفتاحاً رئيسيّاً PK، فمثلاً، تعرض نافذة تحرير العلاقات في MS Access في الشكل السابق خيارات إضافيين لقواعد المفتاح الخارجي FK، هما: التحديث المتسلسل أو التعاقبی Update Cascade، والحذف المتسلسل Cascade Delete، فإذا لم يُحدّد هذان الخياران، فسيمنع النظام حذف أو تحديث قيم المفتاح الرئيسي PK في جدول الأب -أي جدول العملاء Customer- في حالة وجود سجل ابن، فالسجل ابن هو أي سجل مع مفتاح رئيسي PK مطابق.

يوجد خيار إضافي في بعض قواعد البيانات عند تحديد خيار الحذف ويسمى Set to Null، حيث يُحذف صفات المفتاح الرئيسي PK في هذا الاختيار، ولكن يُضيّط المفتاح الخارجي FK في الجدول ابن على القيمة الفارغة NULL، فعلى الرغم من أنّ هذا يؤدي إلى إنشاء صفات يتيم، إلا أنه أمر مقبول.

## 7.2 قيود المؤسسة

يشار إلى قيود المؤسسة أحياناً بالقيود الدلالية semantic constraints، وهي قواعد إضافية يحددها المستخدمون أو مسؤولو قاعدة البيانات، كما يمكنها الاستناد إلى جداول متعددة، وفيما يلي بعض الأمثلة عنها:

- يمكن للصف الدراسي class ضم ثلاثة طالبين على أساس حد أقصى.

- يمكن للمدرس teacher تدريس أربعة صفوف في الفصل الواحد على أساس حد أقصى.
- لا يمكن للموظف employee المشاركة في أكثر من خمسة مشاريع.
- لا يمكن لراتب الموظف تجاوز راتب مديره.

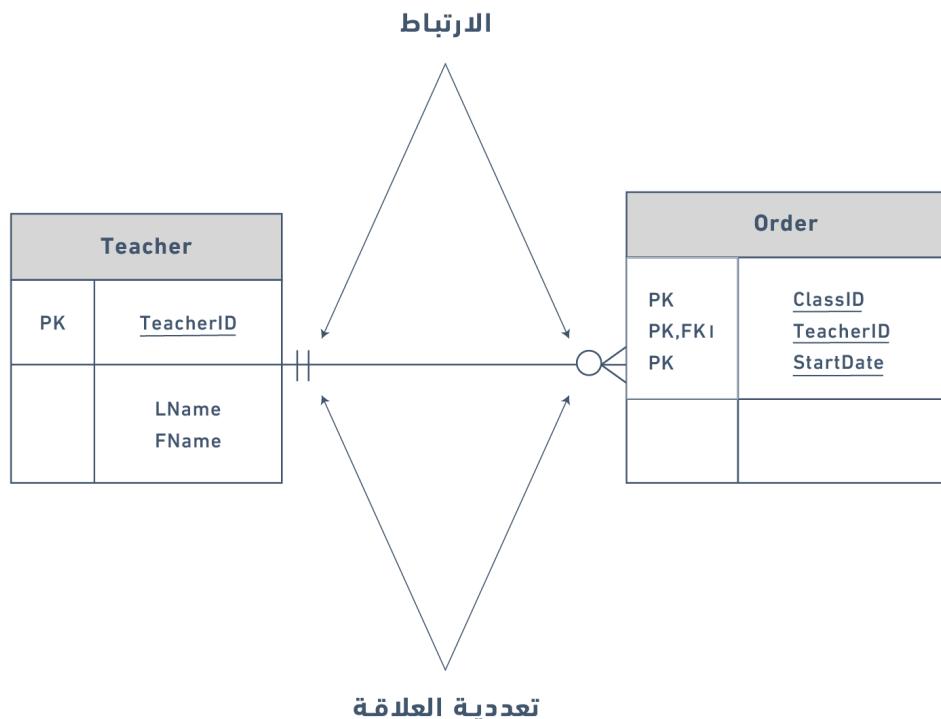
## 7.3 قواعد العمل

نحصل على قواعد العمل من المستخدمين عند جمع المتطلبات gathering requirements، كما تُعد عملية جمع المتطلبات عمليةً مهمةً للغاية، ويجب على المستخدم أن يتحقق من نتائجها قبل بناء تصميم قاعدة البيانات، فإذا كانت قواعد العمل غير صحيحة، فسيكون التصميم غير صحيح، وفي النهاية لن يعمل التطبيق على النحو الذي توقعه المستخدمون، وفيما يلي بعض الأمثلة عن قواعد العمل، وهي:

- يمكن للمدرس تدريس طلاب متعددين.
- يمكن للصف الدراسي امتلاك 35 طالبًا على أساس حد أقصى.
- يمكن تدريس المقرر course عدة مرات، ولكن يدرسه مدرس واحد فقط.
- لا يدرس جميع المدرسين صفوفًا دراسية.

### 7.3.1 تعددية العلاقة Cardinality والارتباط

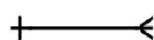
تُستخدم قواعد العمل لتحديد تعددية العلاقة والارتباط، حيث تصف تعددية العلاقة Cardinality العلاقة بين جدولين بيانات من خلال التعبير عن الحد الأدنى والحد الأقصى لعدد مرات حدوث الكيان المرتبط بحدوث كيان آخر ذي صلة، ويمكنك الشكل التالي من رؤية أن تعددية العلاقة ممثلة من خلال العلامات الداخلية على رمز العلاقة، حيث تكون درجة العلاقة Cardinality هي 0 على اليمين بينما هي 1 على اليسار.



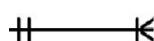
يمثل الرمز الخارجي لرمز العلاقة الارتباط Connectivity بين الجدولين، فالارتباط هو العلاقة بين جدولين مثل علاقة واحد إلى واحد one to one، أو واحد إلى متعدد one to many؛ والمرة الوحيدة التي يكون فيها الارتباط صفرًا هي عندما يكون للمفتاح الخارجي FK قيمة فارغة null.

يوجد ثلاثة خيارات للعلاقة بين هذه الكيانات عندما يتعلق الأمر بالمشاركة، وهي: 0، أو 1، أو متعدد many، فمثلاً، قيمة الارتباط Connectivity هي 1 في الشكل السابق على الجانب الخارجي الأيسر من هذا الخط، ومتعدد على الجانب الخارجي الأيمن.

يظهر الشكل التالي الرمز الذي يمثل علاقة واحد إلى متعدد one to many



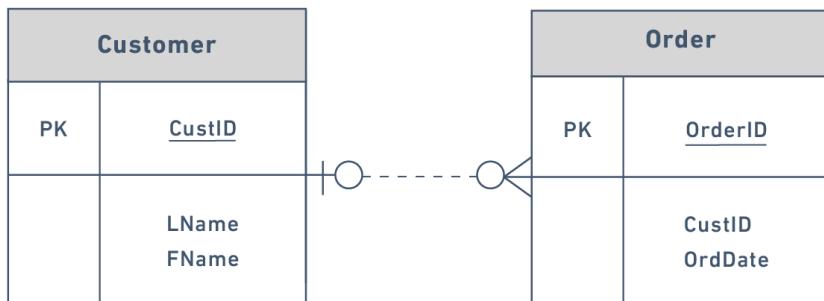
يعرض الشكل الآتي كلاً من العلامات الداخلية -التي تمثل عدديّة العلاقة Cardinality- والعلامات الخارجية -التي تمثل الارتباط Connectivity-، إذ يقرأ الجانب الأيسر من هذا الرمز على أن الحد الأدنى 1 والحد الأقصى 1، بينما يقرأ الجانب الأيمن على النحو التالي: الحد الأدنى 1 والحد الأقصى متعدد.



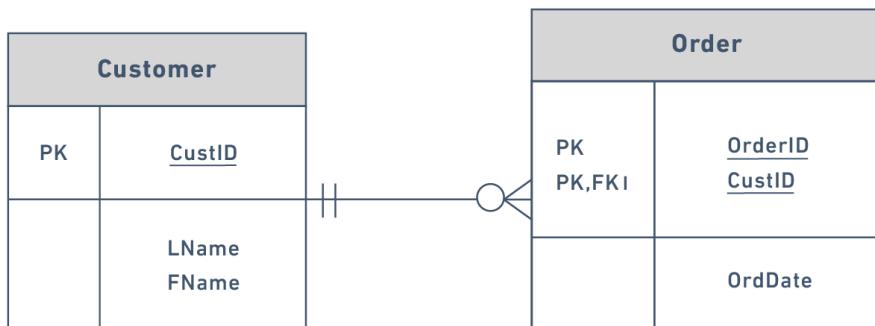
## 7.4 أنواع العلاقات

يشير السطر الذي يربط جدولين في مخطط الكيان وال العلاقة ER diagram -أو identifying relationship- إلى نوع العلاقة بين الجدولين؛ فهي إما وثيقة أو معروفة identifying أو غير وثيقة non-identifying.

العلاقة الوثيقة هي خط متصل بحيث يحتوي المفتاح الرئيسي PK على المفتاح الخارجي FK، كما يشار إلى العلاقة الغير وثيقة بخط متقطع مع عدم وجود المفتاح الخارجي FK ضمن المفتاح الرئيسي PK.



علاقة غير وثيقة



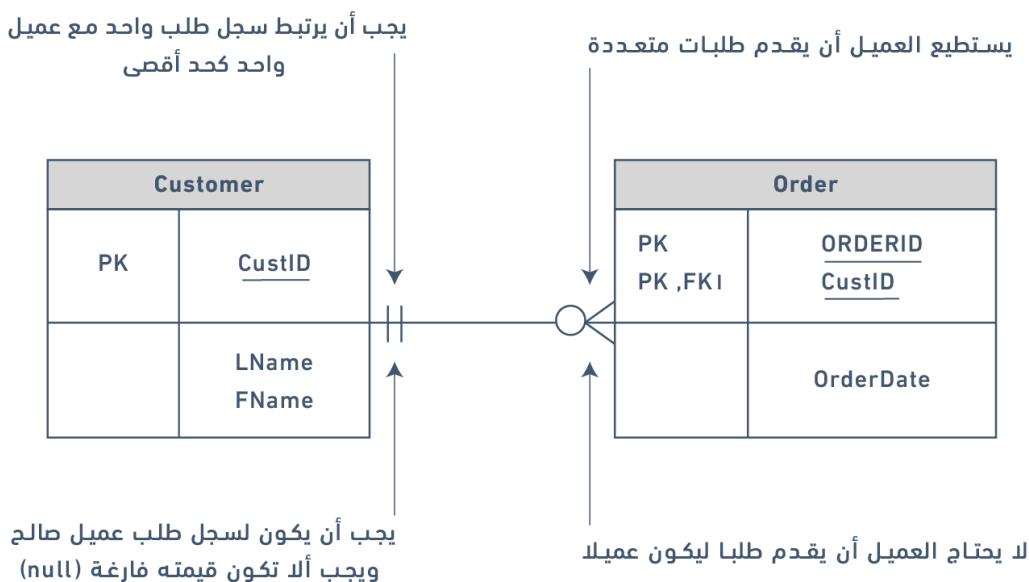
علاقة وثيقة

#### 7.4.1 العلاقات الاختيارية

يمكن أن يكون للمفتاح الخارجي FK قيمةً فارغةً في العلاقة اختيارية أو لا يحتاج الجدول الألب إلى وجود جدول ابن مطابق.

يوضح الرمز المبين في الشكل  نوعاً مكوناً من صفر وثلاث بروزات -تشير إلى متعدد- والذي يفسّر على أنه علاقة صفر أو متعدد zero OR many.

إذا نظرت إلى جدول الطلبات Order table على الجانب الأيمن من الشكل الآتي مثلاً، فستلاحظ عدم حاجة العميل customer إلى تقديم طلب ليكون عميلاً، أي أن الجانب المتعدد اختياري، ويوضح الشكل التالي المثال :zero to many عن كيفية استخدام رمز العلاقة اختيارية صفر إلى متعدد.

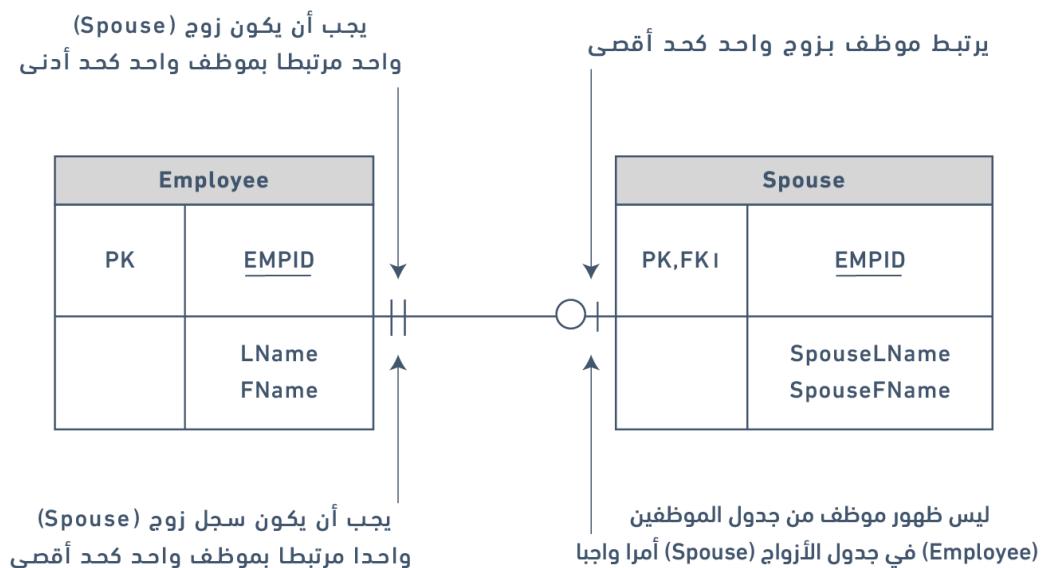


يمكن أيضًا قراءة رمز العلاقة في الشكل السابق على النحو التالي:

- الجانب الأيسر: يجب احتواء كيان الطلب order entity على كيان واحد مرتبط على أساس حد أدنى في جدول العميل Customer table، وكيان واحد مرتبط على أساس حد أقصى.
- الجانب الأيمن: يمكن للعميل عدم تقديم طلبات (أي صفر طلب) على أساس حد أدنى، أو تقديم طلبات متعددة على أساس حد أقصى.

يوضح الشكل  نوعاً آخرًا من رموز العلاقة اختيارية بصفة وواحد، أي علاقة صفر أو واحد zero OR one.

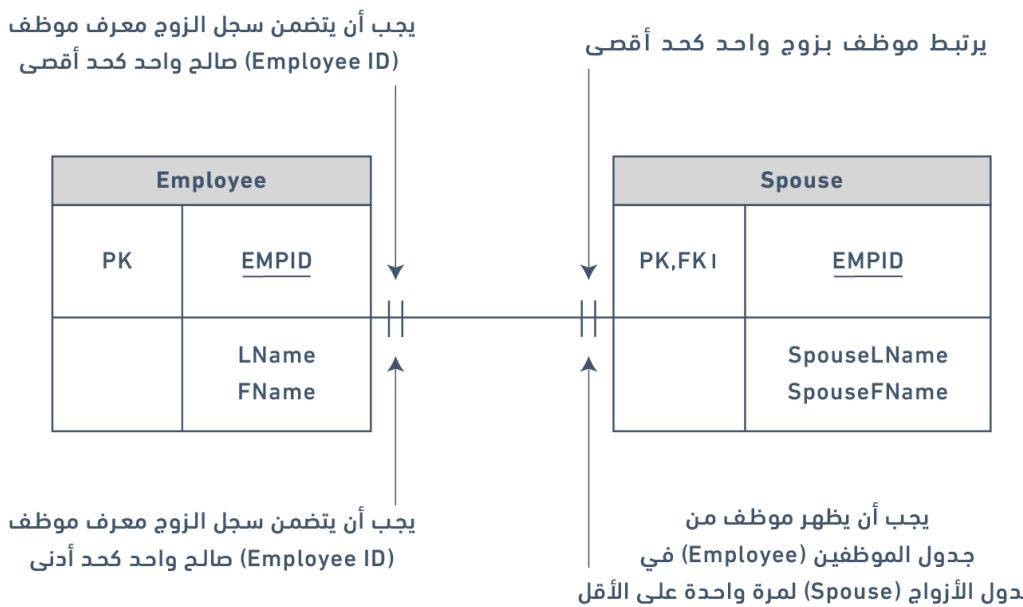
يوضح الشكل التالي مثلاً عن كيفية استخدام رمز العلاقة اختيارية صفر إلى واحد zero to one:



## 7.4.2 العلاقات الإلزامية

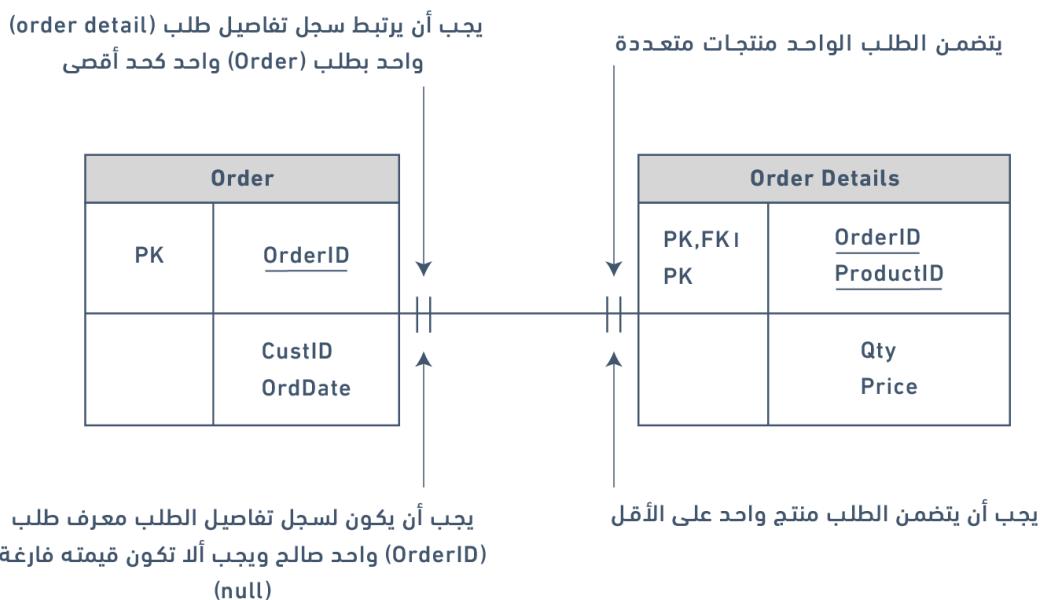
يتطلب حدوث كيان واحد حدوث كيان مقابل في العلاقة الإلزامية. يُظهر رمز هذه العلاقة علاقة واحد فقط one side كما في الشكل , أي أن الجانب واحد إلزامي.

يوضح الشكل مثلاً عن كيفية استخدام رمز العلاقة الإلزامية واحد فقط one and only one



يوضح الشكل  رمز علاقة واحد إلى متعدد one to many حيث يكون الجانب المتعدد many side

الزميًّا، ويوضح الشكل التالي مثلاً عن كيفية استخدام رمز العلاقة الإلزامية واحد إلى متعدد:

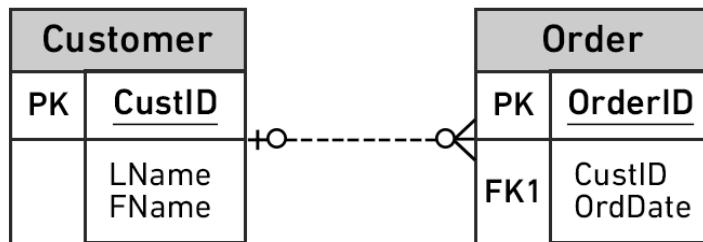


رأينا أن الجانب الداخلي من رمز العلاقة الموضح على الجانب الأيسر من الرمز في الشكل الآتي يمكن أن يكون له تعددية العلاقة cardinality قيمتها 0 وارتباط connectivity متعدد كما هو موضح على الجانب الأيمن من الرمز في الشكل التالي، أو ارتباط قيمته واحد وهو غير موضح في الشكل ولا يمكن أن يكون لديه ارتباط قيمته 0، كما هو في الشكل ، إذ يمكن أن يكون الارتباط 1 فقط.

تُظهر رموز الارتباط الحدود القصوى، فإذا أظهر رمز الارتباط على الجانب الأيسر القيمة 0، فلن يكون هناك ارتباط بين الجداول.

فيما يلي طريقة قراءة رمز العلاقة مثل الرمز الموجود في الشكل الآتي:

- يجب العثور على معرف العميل CustID في جدول الطلبات Order table وفي جدول العملاء Customer table أيًضا بحد أدنى 0 وبحد أقصى 1 مرة.
- تعني القيمة 0 أن معرف العميل CustID في جدول الطلبات Order table قد تكون قيمته فارغة null.
- تشير القيمة 1 الموجودة أقصى اليسار -أي قبل القيمة 0 مباشرةً التي تمثل الارتباط- إلى أنه إذا كان هناك معرف عميل CustID في جدول الطلبات Order table، فيمكن وجود هذا المعرف في جدول العملاء Customer table مرةً واحدةً فقط.
- يمكنك افتراض شيئين عندما ترى الرمز 0 لتعددية العلاقة cardinality: 1. يسمح المفتاح الخارجي FK في جدول الطلبات Order table بوجود القيم الفارغة. 2. ليس المفتاح الخارجي FK جزءاً من المفتاح الرئيسي PK لأنه يجب ألا تحتوي المفاتيح الرئيسية على قيم فارغة null.



## 7.5 مصطلحات أساسية

- **قواعد العمل business rules:** نحصل عليها من المستخدمين عند جمع المتطلبات، وستستخدم لتحديد عددي العلاقة cardinality.
- **عددي العلاقة cardinality:** تُعبر عن الحد الأدنى والحد الأقصى لعدد مرات حدوث الكيان المرتبط بحدث كيان ذي صلة.
- **الارتباط connectivity:** وهو يمثل العلاقة بين جدولين، مثل: علاقة واحد إلى واحد، أو علاقة واحد إلى متعدد.
- **القيود constraints:** تُمثل القواعد التي تجبر نظم إدارة قواعد البيانات DBMSs على التحقق من أن البيانات تتوافق الدلالات semantics.
- **سلامة الكيان entity integrity:** تتطلب وجود مفتاح رئيسي primary key لكل جدول، كما لا يمكن أن يحتوي المفتاح الرئيسي ولا أي جزء منه على قيم فارغة null.
- **العلاقة المعرفة identifying relationship:** حيث يحتوي المفتاح الرئيسي على المفتاح الخارجي foreign key، ويشار إليها في مخطط ERD بخط متصل.
- **قيود السلامة integrity constraints:** هي التعليمات المنطقية التي تحدد قيم البيانات المسموح بها أو غير المسموح بها، كما تحدد التنسيق المناسب للسمة attribute.
- **العلاقة الإلزامية mandatory relationship:** يتطلب حدوث كيان واحد حدوث كيان مقابل.
- **العلاقة الغير معرفة non-identifying relationship:** لا تحتوي على المفتاح الخارجي ضمن المفتاح الرئيسي، ويشار إليها في مخطط ERD بخط منقط.
- **العلاقة اختيارية optional relationship:** حيث يمكن أن يكون للمفتاح الخارجي FK قيمةً فارغةً، أو عندما لا يحتاج الجدول الألب إلى وجود جدول ابن مطابق.
- **السجل اليتيم orphan record:** هو السجل الذي لم يُعثر على قيمة مفتاحه الخارجي في الكيان المقابل أي في الكيان الذي يوجد به المفتاح الرئيسي.

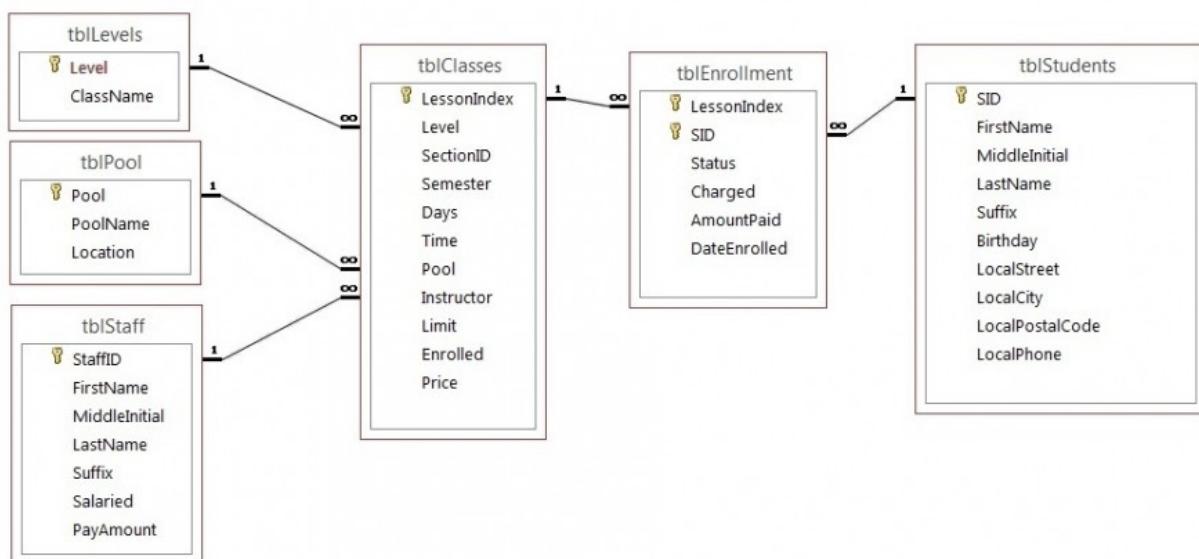
- **السلامة المرجعية referential integrity**: تتطلب أن يكون للمفتاح الخارجي مفتاحاً رئيسياً مطابقاً، وإلا فيجب أن يكون للمفتاح الخارجي قيمةً فارغةً.
- **نظام إدارة قواعد البيانات العلائقية RDBMS أو relational database management system** وهو نظام قاعدة بيانات شائع، حيث يعتمد على النموذج العلائقى الذى قدّمه إدgar Codd من مختبر أبحاث سان خوسيه San Jose التابع لشركة IBM.
- **نوع العلاقة relationship type**: هو نوع العلاقة بين جدولين في مخطط ERD، أي إما علاقة وثيقة أو معروفة identifying، أو غير وثيقة non-identifying، ويُشار إلى هذه العلاقة بخط مرسوم بين جدولين.

## 7.6 تمارين

اقرأ الوصف التالي ثم أجب عن الأسئلة:

صُممّت قاعدة بيانات نادي السباحة في الشكل الآتي لتحتوي على معلومات حول الطلاب students المسجلين enrolled في صفوف السباحة، حيث حُرِّزَت المعلومات التالية: الطالب students، والتسجيل enrollment، وصفوف السباحة swim classes، والمسابح pools التي تقام فيها الصفوف، ومدربو instructors صنفوف السباحة، والمستويات levels المختلفة من صنفوف السباحة.

استخدم الشكل التالي للإجابة على الأسئلة:



حدّدت المفاتيح الرئيسية primary keys، وعُرِّفت أنواع البيانات التالية في خادم SQL Server.

```

**tblLevels**
Level - Identity PK
ClassName - text 20 - nulls are not allowed

**tblPool**
Pool - Identity PK
PoolName - text 20 - nulls are not allowed
Location - text 30

**tblStaff**
StaffID - Identity PK
FirstName - text 20
MiddleInitial - text 3
LastName - text 30
Suffix - text 3
Salaried - Bit
PayAmount - money

**tblClasses**
LessonIndex - Identity PK
Level - Integer FK
SectionID - Integer
Semester - TinyInt
Days - text 20
Time - datetime (formatted for time)
Pool - Integer FK
Instructor - Integer FK
Limit - TinyInt
Enrolled - TinyInt
Price - money

**tblEnrollment**
LessonIndex - Integer FK
SID - Integer FK (LessonIndex and SID) Primary Key
Status - text 30
Charged - bit

```

```

AmountPaid - money
DateEnrolled - datetime
**tblStudents**
SID - Identity PK
FirstName - text 20
MiddleInitial - text 3
LastName - text 30
Suffix - text 3
Birthday - datetime
LocalStreet - text 30
LocalCity - text 20
LocalPostalCode - text 6
LocalPhone - text 10

```

طبق هذا المخطط في خادم SQL Server، أو باستخدام نظام access وعندما ستحتاج إلى اختيار أنواع بيانات قابلة للموازنة.

1. اشرح قواعد العلاقة relationship rules لكل علاقة مثل العلاقة بين الجدولين tblEnrollment وtblStudents التي تمثل إمكانية تسجيل الطالب في صنف سباحة متعددة.

2. حدد عدديّة cardinality كل علاقة بافتراض القواعد التالية:

- قد يكون أو لا يكون للمسبح pool صفة class للسباحة.
- يجب ارتباط جدول المستويات levels table دائمًا بصنف سباحة واحد على الأقل.
- قد لا يدّرس جدول فريق التدريب staff table أيّ صنف سباحة.
- يجب تسجيل جميع الطلاب في صنف سباحة واحد على الأقل.
- يجب احتواء صنف السباحة على طلاب مسجلين فيه.
- يجب امتلاك صنف السباحة على مسبح صالح.
- قد لا يعين مدرب لصنف السباحة.
- يجب ارتباط صنف السباحة دائمًا بمستوى موجود.

3. حدد الجداول الضعيفة، والجداول القوية التي شرحناها في مقال سابق.

4. حدد الجداول المعرفة identifying، والجداول غير المعرفة non-identifying.

# 8. نمذجة الكيان العلقي ER عند تصميم قواعد البيانات

تتضمن إحدى النظريات المهمة التي ظهرت لنموذج الكيان العلقي (ER) entity relational فكرة الاعتمادية الوظيفية functional dependency (اختصاراً FD)، والهدف من دراستها هو تحسين فهمك للعلاقات بين البيانات، واكتساب المنهجية الكافية لمساعدة في التصميم العملي لقاعدة البيانات.

تُستخلص الاعتمادات الوظيفية FD من دلالات semantics من نطاق التطبيق، أي مثل القيود constraints التي شرحناها في [الفصل السابق](#)، وتصف كيفية ارتباط السمات المنفصلة individual attributes بعضها البعض، كما تُعد الاعتمادات الوظيفية FD نوعاً من القيود بين السمات داخل علاقة، وتساهم في تصميم مخطط علقي جيد، حيث سنلقي في هذا الفصل نظرةً على:

- نظرية الاعتمادية الوظيفية الأساسية وتعريفها.
- منهجية تحسين تصميمات المخططات، وتسمى أيضاً التوحيد normalization.

## 8.1 التصميم العلقي Relational Design والتكرار

يجب احتواء التصميم الجيد لقاعدة البيانات العلائقية على جميع السمات والارتباطات الضرورية، إذ يقوم التصميم بذلك بأقل قدر ممكن من المعلومات المخزنة مع عدم وجود بيانات مكررة redundant.

يُعد التكرار أمراً غير مرغوب فيه في تصميم قاعدة البيانات، وذلك لأنه يسبب المشكلات في الحفاظ على التناسق بعد التحديثات، لكن يمكن أن يؤدي التكرار إلى تحسينات في الأداء في بعض الأحيان مثل إمكانية استخدام التكرار بدلاً من عملية الضم join لربط البيانات، حيث تُستخدم عملية الضم join عند الحاجة إلى الحصول على معلومات تستند إلى جدولين مرتبطين.

ضع في بالك الجدول الآتي الذي يوضح مثلاً عن التكرار المستخدم في الحسابات المصرفية Bank Accounts، وفروع المصرف Branches، حيث يظهر العميل رقم 1313131 مرتين، أي مرتين للحساب ذو الرقم A-101، ومرة أخرى للحساب رقم A-102؛ إذ لا يكون رقم العميل زائداً في هذه الحالة على الرغم من وجود حالات حذف شاذة deletion anomalies في الجدول، وسيحيل هذه المشكلة وجود جدول منفصل للعملاء.

إذا تغير عنوان الفرع branch address، فيجب تحريره في أماكن متعددة، وإذا ترك رقم العميل في الجدول كما هو، فلن تحتاج إلى جدول للفرع branch، ولن تكون هناك حاجة إلى عملية ضم، وبالتالي، سيتحسن الأداء.

accountNo	blance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Round Hill	Horseneck	8000000
A-201	900	9676543	brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000

## 8.2 حالة الإدخال الشاذة Insertion Anomaly

تحدث هذه الحالة عند إدخال معلومات متضاربة في جدول، حيث تحتاج إلى التتحقق من أن بيانات الفرع branch متوافقة مع الصفوف الموجودة عندما ندخل سجلاً جديداً مثل رقم الحساب A-306، كما في الجدول التالي:

accountNo	blance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Round Hill	Horseneck	8000000
A-201	900	9676543	brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000
A-306	800	1111111	Round Hill	Horseneck	8000800

## 8.3 حالة التحديث الشاذة **Update Anomaly**

إذا غير أحد فروع المصرف عنوانه مثل الفرع راوند هيل Round Hill في الجدول الآتي، فنحن بحاجة إلى تحديث جميع الصفوف التي تشير إلى هذا الفرع، حيث يسمى تغيير المعلومات الموجودة بصورة غير صحيحة بحالة تحدث شاذة.

accountNo	blance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Round Hill	Horseneck	8000000
A-201	900	9676543	brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000

## 8.4 حالة الحذف الشاذة **Deletion Anomaly**

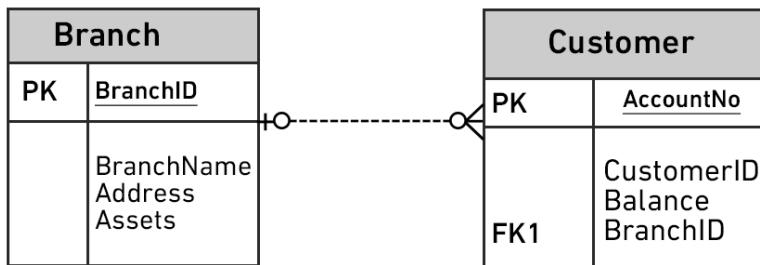
تحدث هذه الحالة عند حذف سجل قد يحتوي على سمات لا ينبغي حذفها، إذا أزلنا معلومات حول الحساب الأخير في أحد الفروع مثل الحساب رقم A-101 في الفرع داون تاون Downtown في الجدول التالي على سبيل المثال، فستختفي جميع معلومات ذلك الفرع.

accountNo	blance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Round Hill	Horseneck	8000000
A-201	900	9676543	brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000

مشكلة حذف الصف A-101 هي عدم معرفتنا مكان الفرع Downtown، وأننا سنفقد جميع المعلومات المتعلقة بالعميل 1313131.

نحتاج إلى تفكيك الجدول الأصلي إلى جداول أصغر متعددة بحيث يكون لكل جدول حدًّا أدنى من التداخل مع الجداول الأخرى، وذلك لتجنب هذه الأنواع من مشاكل التحديث أو الحذف.

يجب احتواء كل جدول حساب مصرفي على معلومات حول كيان entity واحد فقط، مثل: الفرع Branch، أو العميل Customer، كما هو موضح في الشكل التالي:



سيضمن اتباع هذه العملية أن إضافة معلومات الفرع أو تحريرها سيؤثر على سجل واحد فقط، لذلك لن تعدل معلومات الفرع عن طريق الخطأ، ولن تُسجل بصورة غير صحيحة، وذلك عند إضافة معلومات العميل أو حذفها.

#### 8.4.1 مثال تطبيقي: جدول مشروع-موظفي والحالات الشاذة

يوضح الجدول الآتي مثلاً لجدول مشروع-موظفي employee project table، كما يمكننا الافتراض من هذا الجدول أن:

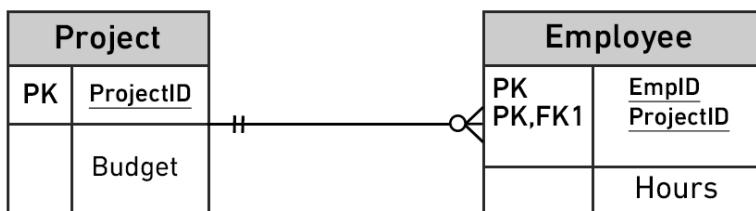
1. معرف الموظف EmpID، ومعرف المشروع ProjectID هما المفتاح الرئيسي المركب composite PK.
2. يحدد معرف المشروع الميزانية Budget، أي أن للمشروع P1 ميزانية لفترة 32 ساعة.

EmpID	Budget	ProjectID	Hours
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5
	17	P4	

فيما يلي بعض الحالات الشاذة anomalies المحتملة التي قد تحدث مع هذا الجدول خلال الخطوات التالية:

1. الإجراء: إضافة الصف Add row . {S85, 35, P1, 9}.
2. المشكلة: هناك صفات tuples بميزانيات متضاربة.
3. الإجراء: حذف الصف Delete tuple . {S79, 27, P3, 1}.
4. المشكلة: الخطوة رقم 3 تزيل ميزانية المشروع P3.

- .5. الإجراء: تحدیث الصف tuple من  $\{S75, 35, P1, 7\}$  إلى  $\{S75, 32, P1, 7\}$ .
- .6. المشكلة: تُنشئ الخطوة رقم 5 صَفَّين بقيم مختلفة لمیزانیة المشروع P1.
- .7. الحل: إنشاء جدول منفصل separate table لكل من المشاريع Projects والموظفيين Employees، أي كما هو موضَّح في الشكل التالي:



## 8.5 كيفية تجنب الحالات الشاذة

أفضل طريقة لإنشاء جداول بدون حالات شاذة هي التأكد من توحيد الجداول، ويتحقق ذلك من خلال فهم الاعتمادات الوظيفية، حيث تضمن الاعتمادية الوظيفية FD في جدول انتماء جميع السمات attributes إلى هذا الجدول، أي سترميز التكرار والحالات الشاذة.

### 8.5.1 مثال تطبيقي: جدول المشاريع والموظفيين المنفصلان

جدول المشروع Project

ProjectID	Budget
P1	32
P2	40
P3	27
P4	17

جدول الموظف Employee

Emp ID	Project ID	Hours
S75	P1	7
S75	P2	3
S79	P1	4
S79	P3	1
S80	P2	5

يكون من خلال الاحتفاظ بالبيانات منفصلة باستخدام جدول مشاريع وجدول موظفين ما يلي:

- لن تُنشأ حالات شاذة إذا تغيرت المیزانیة.

- ليس هناك حاجة إلى قيم وهمية للمشاريع التي لم يُعين لها موظفون.
- إذا حُذفت مساهمة موظف ما، فلن تُفقد بيانات مهمة.
- لن تنشأ حالات شاذة إذا أضيفت مساهمة موظف.

## 8.6 مصطلحات أساسية

- **حالة الحذف الشاذة deletion anomaly**: وتحدث هذه الحالة عند حذف سجل قد يحتوي على سمات لا ينبغي حذفها.
- **الاعتمادية الوظيفية أو FD**: تصف كيفية ارتباط السمات المنفصلة individual attributes بعضها البعض.
- **حالة الإدخال الشاذة insertion anomaly**: تحدث عند إدخال معلومات متضاربة في جدول.
- **عملية الصم join**: تُستخدم هذه العملية عندما تحتاج إلى الحصول على معلومات بناءً على جدولين متعلقين ببعضهما.
- **حالة التحديث الشاذة update anomaly**: تغيير المعلومات الموجودة بصورة غير صحيحة.

## 8.7 تمارين

أولاً: طبق التوحيد normalization على الجدول التالي:

Attribute Name	Sample Value	Sample Value	Sample Value
StudentID	1	2	3
StudentName	John Smith	Sandy Law	Sue Rogers
CourseID	2	2	3
CourseName	Programming Level 1	Programming Level 1	Business
Grade	75%	61%	81%
CourseDate	Jan 5th, 2014	Jan 5th, 2014	Jan 7th, 2014

**ثانيًا:** أنشئ مخطط ER منطقي لخدمة تأجير الأفلام عبر الإنترنت، حيث لا توجد علاقات من النوع متعدد إلى متعدد many to many، واستخدم الوصف التالي للعمليات التي يجب أن تستند عليها قواعد عملك:

تصنّف خدمة تأجير الأفلام عبر الإنترنت عناوين الأفلام وفقًا لنوعها إلى: الأفلام الكوميدية comedy، وأفلام الغرب الأمريكي western، وأفلام الكلاسيكية classical، وأفلام الخيال العلمي science fiction، وأفلام الرسوم المتحركة cartoon، وأفلام الحركة action، وأفلام الموسيقية musical، وأفلام المصدرة حديثًا new release.

يحتوي كل نوع على العديد من العناوين المحتملة، وتتوفر نسخ copies متعددة لمعظم العناوين داخل النوع، فمثلاً، لاحظ الملخص التالي:

TYPE TITLE .1

Musical My Fair Lady (Copy 1) .2

My Fair Lady (Copy 2) .3

Oklahoma (Copy 1) .4

Oklahoma (Copy 2) .5

Oklahoma (Copy 3) .6

.7 ...إلخ.

**ثالثًا:** ما هي الحالات الشاذة الثلاثة للبيانات التي من المحتمل تشكّلها نتيجةً لتكرار البيانات؟ وكيف يمكن القضاء على مثل هذه الحالات الشاذة؟

ستتطرق في فصل لاحق لمثال عملي حول تصميم قاعدة بيانات كاملة من الصفر كتدريب عملي.

# 9. الاعتمادات الوظيفية Functional Dependencies

الاعتمادية الوظيفية functional dependency -أو FD اختصاراً- هي علاقة بين سمتين attributes، حيث تكون عادةً بين المفتاح الرئيسي PK والسمات الأخرى التي ليست مفاتيحاً non-key attributes داخل الجدول، إذ تعتمد السمة Y وظيفياً على السمة X -والتي تُعد مفتاحاً رئيسياً PK-. في علاقة R إذا حددت قيمة السمة X قيمة السمة Y بصورة فريدة لكل نسخة صالحة من السمة X، ويشار إلى هذه العلاقة من خلال التمثيل التالي:

X → Y

يسمى الجانب الأيسر من مخطط الاعتمادية الوظيفية FD السابق بالمحدد determinant، ويسمى الجانب الأيمن بالاعتمادي dependent، وفيما يلي بعض الأمثلة على ذلك.

تحدد السمة SIN الاسم Name، والعنوان Address، وتاريخ الميلاد Birthdate في المثال الأول أدناه، حيث يمكننا تحديد أي من السمات الأخرى داخل الجدول باستخدام السمة SIN.

SIN → Name, Address, Birthdate

تحدد السمتان SIN، Course تاريخ الانتهاء DateCompleted في المثال الثاني، حيث يجب أن يعمل هذا أيضاً مع مفتاح رئيسي مركب composite PK.

SIN, Course → DateCompleted

يشير المثال الثالث إلى أن السمة ISBN تحدد العنوان Title.

ISBN → Title

## 9.1 قواعد الاعتمادات الوظيفية

انظر إلى جدول البيانات (R) لمخطط العلاقة (R)ABCDE التالي:

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	c2	d2	e1
a3	b2	c1	d1	e1
a4	b2	c2	d2	e1
a5	b3	c3	d1	e1

قد تسأل نفسك عند النظر إلى هذا الجدول: ما نوع الاعتمادات التي يمكننا ملاحظتها بين السمات في الجدول؟

بما أنّ قيم السمة A فريدة a1, a2, a3, etc اعتماداً على تعريف الاعتمادية الوظيفية FD أنّ:

$A \rightarrow B$ ,  
 $A \rightarrow C$ ,  
 $A \rightarrow D$ ,  
 $A \rightarrow E$

ويترتب على ذلك أيضاً أن  $BC \rightarrow A$ , أو أي مجموعة فرعية أخرى من المجموعة ABCDE.

يمكن تلخيص ذلك على أساس  $A \rightarrow BCDE$ .

تُعدّ السمة A مفتاحاً رئيسياً.

بما أن قيم السمة E هي نفسها دائمًا أي كلها لها القيمة e1، فهذا يعني أنّ:

$A \rightarrow E$ ,  
 $B \rightarrow E$ ,  
 $C \rightarrow E$ ,  
 $D \rightarrow E$

ولكن لا يمكننا تلخيص ما سبق باستخدام  $ABCD \rightarrow E$ , وذلك لأنّ:

$A \rightarrow E$ ,  
 $B \rightarrow E$ ,  
 $AB \rightarrow E$

### 9.1.1 ملاحظات أخرى

1. تركيبات BC فريدة، لذلك  $BC \rightarrow ADE$

2. تركيبات BD فريدة، لذلك  $BD \rightarrow ACE$

3. إذا كانت قيم السمة C متطابقة، فإن قيم السمة D متطابقة كذلك.

4. لذلك  $C \rightarrow D$

5. ولكن لا تحدّد قيم السمة D قيم السمة C.

6. لذلك لا تحدّد السمة C السمة D، ولا تحدّد السمة D السمة C.

يمكن مساعدة النظر إلى البيانات الفعلية في توضيح السمات التي هي سمات اعتمادية dependent والسمات التي هي سمات محدّدة determinant.

## 9.2 قواعد الاستدلال Inference Rules

تُعدّ بديهيّات أرمسترونغ Armstrong's axioms مجموعةً من قواعد الاستدلال المستخدمة لاستنتاج جميع الاعتمادات الوظيفية في قاعدة بيانات علائقية، حيث طوّر ويليام أرمسترونغ William W. Armstrong هذه البديهيّات.

لنفترض أنّ  $(U, R)$  هو مخطط علاقة لمجموعة سمات  $U$ ، حيث سنستخدم الأحرف  $X, Y, Z$  لتمثيل أي مجموعة فرعية، أو اتحاد union مجموعتين من السمات اختصاراً، بدلاً من استخدام  $Y \cup Z$ .

### 9.2.1 بديهيّة الانعكاس Axiom of reflexivity

تنص هذه البديهيّة على أنه إذا كانت  $Y$  مجموعة فرعية subset من  $X$ ، فإنّ  $X$  تحدّد  $Y$ ، كما في المعادلة:

$$\text{if } Y \subseteq X \text{ then } X \rightarrow Y$$

اففترض الاعتمادية الوظيفية  $\text{PartNo} \rightarrow \text{NT123}$  على سبيل المثال، حيث تترتب  $(\text{PartNo}, X)$  من معلومات متعددة، وهي:  $(\text{partID}(123), Y)$ ، و  $(\text{NT}, Z)$ .

### 9.2.2 بديهيّة الزيادة Axiom of augmentation

تنص بديهيّة الزيادة -والمعروفة باسم الاعتمادية الجزئية partial dependency على أنه إذا كانت  $X$  تحدّد  $Y$ ، فإنّ  $XZ$  تحدّد  $YZ$  مهما كانت  $Z$ :

$$\text{if } X \rightarrow Y \text{ then } XZ \rightarrow YZ \text{ for any } Z$$

تنص بديهية الزيادة على أن يجب على كل سمة ليست مفتاحاً non-key attribute الاعتماد على المفتاح الرئيسي PK بصورة كاملة، فمثلاً، تعتمد السمات التالية: اسم الطالب StudentName، العنوان Address، والمدينة City، وProv، وPC - أي الرمز البريدي postal code - على سمة رقم الطالب StudentNo فقط، ولا تعتمد على السمتين StudentNo، Grade معاً.

StudentNo, Course  $\rightarrow$  StudentName, Address, City, Prov, PC, Grade, DateCompleted

هذا الوضع غير مرغوب به لأنه يجب على كل سمة ليست مفتاحاً الاعتماد على المفتاح PK تماماً، ولكن تعتمد معلومات الطالب في هذه الحالة جزئياً فقط على المفتاح الرئيسي PK أي StudentNo.

يجب إصلاح هذه المشكلة من خلال تقسيم الجدول الأصلي إلى جداولين على النحو التالي:

- الجدول الأول 1: يحوي الحقول التالية:

StudentNo

Course

Grade

DateCompleted

- الجدول الثاني 2: ويحوي الحقول التالية:

StudentNo

StudentName

Address

City

Prov

PC

### 9.2.3 البديهية المتعددة Axiom of transitivity

تنص البديهية المتعددة على أنه إذا كانت X تحدّد Y، وY تحدّد Z، فإنّ X تحدّد Z أيضاً:

*if  $X \rightarrow Y \wedge Y \rightarrow Z$  then  $X \rightarrow Z$*

يحتوي الجدول أدناه على معلومات غير مرتبطة مباشرةً بالطالب، فيجب أن يكون لمعرف البرنامج ProgramID، واسم البرنامج ProgramName جدولًا خاصًا بهما، حيث لا يعتمد اسم البرنامج على رقم الطالب، وإنما يعتمد على معرف البرنامج.

StudentNo  $\rightarrow$  StudentName, Address, City, Prov, PC, ProgramID, ProgramName

هذه الحالة غير مرغوب بها بسبب اعتماد السمة التي ليست مفتاحًا -أي ProgramName- على سمة أخرى ليست مفتاحًا أيضًا -أي ProgramID-.

نحل هذه المشكلة من خلال تقسيم هذا الجدول إلى جدولين: جدول لمعلومات الطالب، والآخر لمعلومات البرنامج، أي كما يلي:

• الجدول 1:

StudentNo  $\rightarrow$  StudentName, Address, City, Prov, PC, ProgramID

• الجدول 2:

ProgramID  $\rightarrow$  ProgramName

لكن لا نزال بحاجة إلى مفتاح خارجي FK في جدول الطالب لنتتمكن من تحديد البرنامج الذي سُجّل الطالب به.

## 9.2.4 الاتحاد Union

تشير هذه القاعدة إلى أنه إذا كان جدولان منفصلان، والمفتاح الرئيسي PK هو نفسه، فقد ترغب في وضع الجدولين معًا إذ تنص هذه القاعدة على أنه إذا كانت X تحدّد Z، وX تحدّد Y، فيجب على X أن تحدّد Y وZ أيضًا:

$$\text{if } X \rightarrow Y \wedge X \rightarrow Z \text{ then } X \rightarrow YZ$$

افرض على سبيل المثال أنّ:

SIN  $\rightarrow$  EmpName

SIN  $\rightarrow$  SpouseName

قد ترغب في ضم هذين الجدولين في جدول واحد على النحو التالي:

SIN  $\rightarrow$  EmpName, SpouseName

قد يختار بعض مسؤولي قاعدة البيانات database administrators أو DBA اختصارًا -الاحتفاظ بهذه الجداول مفصولةً لسبعين، هما: السبب الأول هو أن كل جدول يصف كيانًا مختلفًا، لذلك يجب إبقاء الكيانات

منفصلةً عن بعضها البعض، والسبب الثاني هو إذا ترك اسم الزوج SpouseName فارغاً NULL في معظم الأوقات، فلا حاجة إلى تضمينه في نفس جدول اسم الموظف .EmpName

### 9.2.5 التفکك Decomposition

التفکك Decomposition هو عكس قاعدة الاتحاد Union، فإذا كان لديك جدولًا يبدو أنه يحتوي على كيانين يحدّدهما المفتاح الرئيسي PK نفسه، فستفترّق في تقسيمه إلى جداولين، حيث تنص هذه القاعدة على أنه إذا كانت  $X \rightarrow YZ$ ، فستكون  $X \rightarrow Y$  و  $X \rightarrow Z$  بصورة منفصلة:

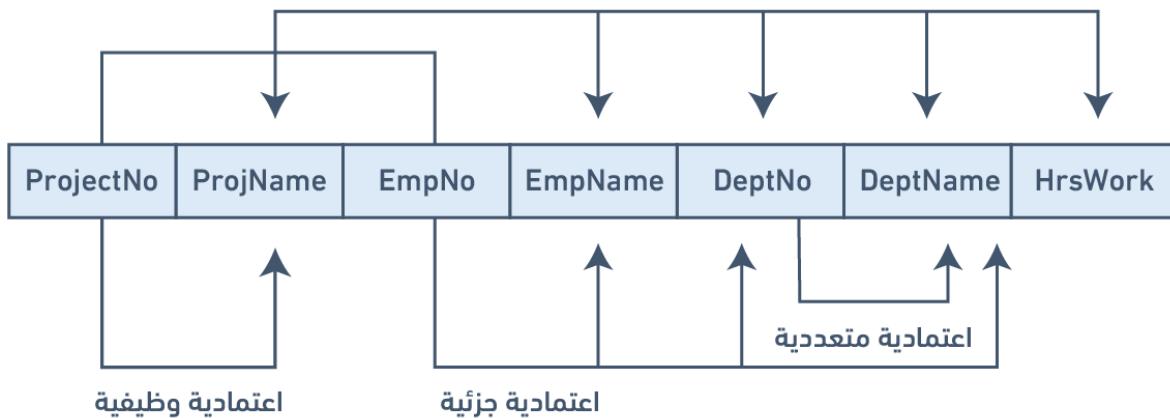
$$\text{if } X \rightarrow YZ \text{ then } X \rightarrow Y \wedge X \rightarrow Z$$

### 9.3 مخطط الاعتمادية Dependency Diagram

يوضح مخطط الاعتمادية والمبيّن في الشكل الآتي العديد من الاعتمادات المختلفة التي قد تكون موجودةً في جدول لم تطبّق عليه عملية التوحيد non-normalized table، أي الجدول الذي يحتوي على تكرار بيانات.

تُحدّد الاعتمادات التالية في هذا الجدول كما يلي:

- يتألف المفتاح الأساسي من مجموع .EmpNo و ProjectNo



- الاعتمادات الجزئية :PDs

$$\text{ProjectNo} \rightarrow \text{ProjName} \quad \circ$$

$$\text{EmpNo} \rightarrow \text{EmpName}, \text{DeptNo} \quad \circ$$

$$\text{ProjectNo, EmpNo} \rightarrow \text{HrsWork} \quad \circ$$

- الاعتمادية المتعددة Transitive Dependency

DeptNo —> DeptName ◦

## 9.4 مصطلحات أساسية

- **بديهيات أرمسترونغ Armstrong's Axioms:** هي مجموعة من قواعد الاستدلال المستخدمة لاستنتاج جميع الاعتمادات الوظيفية في قاعدة بيانات عائقية.
- **أي DBA:** أي database administrator، وتعني مسؤول قاعدة البيانات.
- **التفكك decomposition:** قاعدة تشير إلى أنه إذا كان لديك جدول يبدو أنه يحتوي على كيانين يحددهما المفتاح الرئيسي PK نفسه، ففكّر في تقسيمه إلى جدولين.
- **الاعتمادي dependent:** الجانب الأيمن من مخطط الاعتمادية الوظيفية.
- **المحدد determinant:** الجانب الأيسر من مخطط الاعتمادية الوظيفية.
- **الاعتمادية الوظيفية functional dependency أو FD:** علاقة بين سمتين، عادةً ما تكون بين المفتاح الرئيسي PK والسمات الأخرى التي ليست مفاتيح داخل جدول.
- **جدول غير موحد non-normalized table:** الجدول الذي يحتوي على تكرار بيانات فيه.
- **الاتحاد Union:** قاعدة تشير إلى أنه إذا كان جدولان منفصلان، ولهمما المفتاح الرئيسي PK نفسه، ففكّر في وضعهما معاً.

# 10. فهم عملية التوحيد

## Normalization

يجب أن يكون التوحيد جزءاً من عملية تصميم قاعدة البيانات، ولكن من الصعب فصل عملية التوحيد عن عملية نمذجة الكيان العلائقى ER modelling، لذلك يجب استخدام الطريقتين بالتزامن.

يُستخدم مخطط علاقات الكائنات entity relation diagram -أو ER اختصاراً- لتوفير الرؤية الكبيرة أو الشاملة لمتطلبات بيانات المؤسسة وعملياتها، إذ ينشأ ذلك من خالل عملية تكرارية تتضمن تحديد الكيانات المرتبطة، وسماتها، وعلاقاتها؛ بينما يركّز إجراء التوحيد على خصائص كيانات محددة، كما يمثل رؤيةً مُصغرّةً للكيانات داخل مخطط ERD.

### 10.1 ما هو التوحيد؟Normalization

التوحيد هو فرع من فروع النظرية العلائقية الذي يوفر رؤى التصميم، وهو عملية تحديد مقدار التكرار redundancy الموجود في الجدول.

أهداف التوحيد هي:

- القدرة على وصف مستوى التكرار في مخطط علائقى.
- توفير آليات لتغيير المخططات بهدف إزالة التكرار.

تعتمد نظرية التوحيد على نظرية الاعتمادات الوظيفية اعتماداً كبيراً، وتحدد هذه النظرية ستة نماذج موحدة normal forms -أو NF اختصاراً-، حيث يتضمن كل نموذج موحد مجموعةً من خصائص الاعتمادية التي يجب أن يفي المخطط بها، كما يعطي كل نموذج موحد ضمانات حول وجود / أو عدم وجود حالات تحديث شاذة update anomalies، وهذا يعني احتواء النماذج الموحدة الأعلى على عدد أقل من التكرار، وبالتالي، مشاكل تحديث أقل.

## 10.2 النماذج الموعودة Normal Forms

يمكن أن تكون جميع الجداول الموجودة في قواعد البيانات ضمن أحد النماذج الموعودة التي سناقشها الآن. نريد الحد الأدنى من التكرار بين المفتاح الرئيسي PK، والمفتاح الخارجي FK من الناحية المثالية، كما يجب اشتغال كل شيء آخر من جداول أخرى.

هـنـاك سـتـة نـمـاذـج مـوـعـودـة، وـلـكـنـا سـنـلـقـي نـظـرـة عـلـى النـمـاذـج الـأـرـبـعـة الـأـوـلـى فـقـطـ، وـهـيـ:

- النـمـوذـج الـمـوـعـودـ الـأـوـلـ First normal form، أو 1NF اختصاراً.
- النـمـوذـج الـمـوـعـودـ الثـالـثـ Second normal form، أو 2NF اختصاراً.
- النـمـوذـج الـمـوـعـودـ الـثـالـثـ Third normal form، أو 3NF اختصاراً.
- نـمـوذـج بـوـيـسـ-كـوـدـ الـمـوـعـودـ Boyce-Codd normal form، أو BCNF اختصاراً، وـهـوـ نـادـرـ الـإـسـتـخـدـامـ.

## 10.3 النـمـوذـج الـمـوـعـودـ الـأـوـلـ First Normal Form أو 1NF اختصاراً

يـسـمـحـ فـقـطـ بـقـيمـ غـيرـ مـكـرـرـةـ فـيـ النـمـوذـجـ الـمـوـعـودـ الـأـوـلـ عـنـ تـقـاطـعـ كـلـ صـفـ وـعـمـودـ، وـهـذـاـ يـؤـدـيـ إـلـىـ عـدـمـ وجودـ مـجـمـوعـاتـ مـكـرـرـةـ repeating groupـ، حـيـثـ يـجـبـ إـزـالـةـ الـمـجـمـوعـةـ الـمـكـرـرـةـ، وـتـشـكـيلـ عـلـاقـتـيـنـ جـدـيـدـتـيـنـ لـتـوـحـيـدـ عـلـاقـةـ تـحـتـوـيـ عـلـىـ مـجـمـوعـةـ مـكـرـرـةـ، وـيـكـوـنـ الـمـفـتـاحـ الرـئـيـسـيـ PKـ لـلـعـلـاقـةـ الـجـدـيـدـةـ هـوـ تـرـكـيـبـ مـنـ الـمـفـتـاحـ الرـئـيـسـيـ PKـ لـلـعـلـاقـةـ الـأـصـلـيـةـ بـالـإـضـافـةـ إـلـىـ سـمـةـ مـنـ الـعـلـاقـةـ الـمـنـشـأـةـ حـدـيـثـاـ لـلـحـصـولـ عـلـىـ تـعـرـيفـ فـرـيدـ.

### 10.3.1 عملية نـمـوذـجـ 1NF

سـنـسـتـخـدـمـ جـدـولـ تـقـرـيرـ درـجـاتـ الطـالـبـ Student\_Grade\_Reportـ أـدـنـاهـ، مـنـ قـاعـدـةـ بـيـانـاتـ المـدـرـسـةـ علىـ أـسـاسـ مـثـالـ لـشـرـحـ عمـلـيـةـ نـمـوذـجـ 1NFـ Schoolـ.

\*\*Student\_Grade\_Report\*\* (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- تكون المجموعة المكررة في جدول تقرير درجات الطالب Student\_Grade\_Report هي معلومات المقرر الدراسي course، إذ يمكن للطالبأخذ مقررات متعددة.
- أـزـلـ المـجـمـوعـةـ الـمـكـرـرـةـ، إـذـ إـنـ المـجـمـوعـةـ الـمـكـرـرـةـ فـيـ هـذـهـ الـحـالـةـ هـيـ مـعـلـومـاتـ المـقـرـرـ لـكـلـ طـالـبـ.
- حـدـدـ الـمـفـتـاحـ الرـئـيـسـيـ PKـ لـجـدـولـكـ الـجـدـيـدـ.
- يـجـبـ أـنـ يـحـدـدـ الـمـفـتـاحـ الرـئـيـسـيـ PKـ قـيـمـةـ السـمـةـ CourseNoـ وـStudentNoـ تـحـديـداـ فـرـيدـاـ.
- يـبـقـيـ جـدـولـ مـقـرـراتـ الطـالـبـ StudentCourseـ بـعـدـ إـزـالـةـ جـمـيـعـ السـمـاتـ الـمـتـعـلـقـةـ بـالـمـقـرـرـ وـالـطـالـبـ.

- أصبح جدول الطالب Student الآن بصيغة النموذج الموحد الأول مع إزالة المجموعة المكررة.
- الجدولان الجديدان موضحان كما يلي:

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

### 10.3.2 كيفية تحديث حالات نموذج 1NF الشاذة

بالنظر إلى الجدولين السابقين:

- تحتاج طالبًا لإضافة مقرر جديد.
- قد يكون لدينا تناقضات عندما تحتاج معلومات المقرر إلى تحديث.
- قد نحذف أيضًا معلومات هامة حول مقرر عند حذف طالب.

## 10.4 النموذج الموحد الثاني أو Second Normal Form

يجب أن تكون العلاقة أولاً بصيغة نموذج 1NF للانتقال إلى النموذج الموحد الثاني 2NF، حيث تكون العلاقة تلقائياً بصيغة نموذج 2NF إذا وفقط إذا اشتمل المفتاح الرئيسي PK على سمة واحدة.

إذا احتوت العلاقة على مفتاح رئيسي مركب composite PK، فيجب أن تعتمد كل سمةٍ لـPK على مفتاح الرئيسي PK بأكمله اعتماداً كاملاً، ولا تعتمد على مجموعة فرعية من المفتاح الرئيسي PK، أي لا يجب وجود اعتمادية جزئية partial dependency، أو ما يُسمى بالزيادة augmentation.

### 10.4.1 عملية نموذج 2NF

- يجب أولاً أن يكون الجدول بصيغة نموذج 1NF للانتقال إلى النموذج 2NF.
- جدول الطالب Student موجود بالفعل بصيغة نموذج 2NF لأنه يحتوي على عمود مفتاح رئيسي واحد فقط بالفعل.
- ليست كل السمات -وتحديداً جميع معلومات المقرر course information- معتمدةً بالكامل على المفتاح الرئيسي عند فحص جدول مقررات الطالب Student Course، إذ تكون السمة الوحيدة المعتمدة بالكامل على المفتاح الرئيسي هي الدرجة grade.
- عُرِّف الجدول الجديد الذي يحتوي على معلومات المقرر.

- عـرـفـ المـفـتـاحـ الرـئـيـسـيـ PKـ لـلـجـدـوـلـ الجـدـيـدـ.
- الـجـدـاوـلـ الـثـلـاثـةـ الجـدـيـدـةـ موـضـحـةـ أـدـنـاهـ.

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

CourseInstructor (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)

#### 10.4.2 كيفية تحديث حالات نموذج 2NF الشاذة

بالنظر إلى الجداول الثلاثة السابقة:

- نحتاج إلى مقرر course عند إضافة مدرس جديد.
- قد يؤدي تحديث معلومات المقرر إلى وجود تناقضات في معلومات المدرس.
- قد يؤدي حذف المقرر أيضاً إلى حذف معلومات المدرس.

#### 10.5 النموذج الموحد الثالث أو 3NF

يجب أن تكون العلاقة بصيغة النموذج الموحد الثاني 2NF للانتقال إلى النموذج الموحد الثالث 3NF، كما يجب إزالة جميع الاعتمادات المتعددة transitive dependencies أيًضاً، فقد لا تعتمد السمة التي ليست مفتاحاً اعتماداً وظيفياً على سمة أخرى ليست مفتاحاً.

##### 10.5.1 عملية نموذج 3NF

- أـزـلـ كـافـةـ السـمـاتـ الـاعـتـمـادـيـةـ dependent attributesـ فيـ الـعـلـاقـةـ،ـ أوـ فيـ الـعـلـاقـاتـ المتـعـدـدـةـ منـ كـلـ جـدـوـلـ منـ الـجـدـاوـلـ الـتـيـ لهاـ عـلـاقـةـ متـعـدـدـةـ.
- أـنـشـئـ جـدـوـلـاـ،ـ أوـ جـدـاوـلـ جـدـيـدـةـ معـ إـزـالـةـ الـاعـتـمـادـيـةـ.
- تـحـقـقـ مـنـ الـجـدـوـلـ أوـ الـجـدـاوـلـ الـجـدـيـدـةـ،ـ كـمـ تـحـقـقـ مـنـ الـجـدـوـلـ أوـ الـجـدـاوـلـ الـمـعـدـلـةـ أيـضاـ وـذـلـكـ لـلـتـأـكـدـ مـنـ اـحـتـواـءـ كـلـ جـدـوـلـ عـلـىـ مـحـدـدـ determinantـ وـمـنـ عـدـمـ وـجـودـ جـدـوـلـ يـحـتـوـيـ عـلـىـ اـعـتـمـادـيـاتـ غـيرـ مـنـاسـبـةـ.
- الـجـدـاوـلـ الـأـرـبـعـةـ الـجـدـيـدـةـ موـضـحـةـ أـدـنـاهـ.

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

Course (CourseNo, CourseName, InstructorNo)

Instructor (InstructorNo, InstructorName, InstructorLocation)

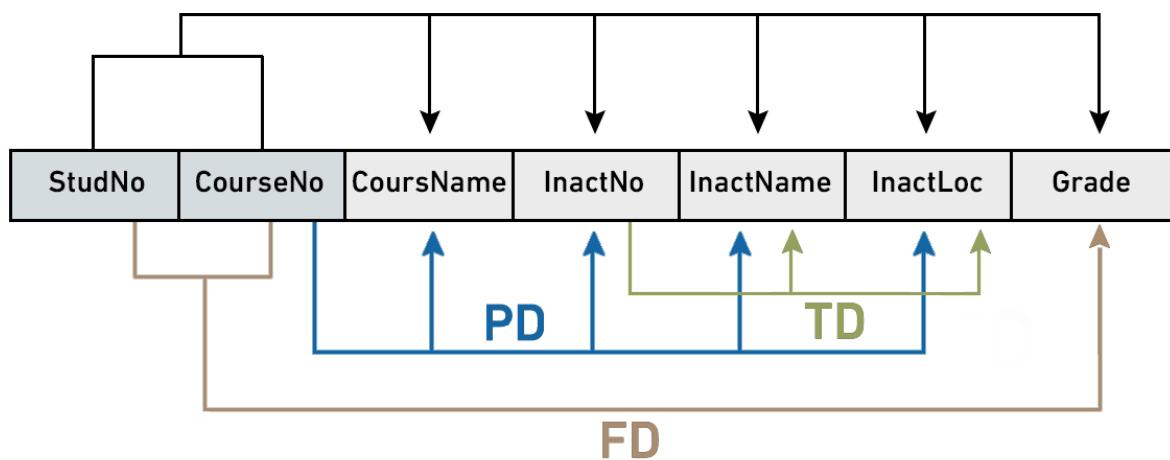
يجب ألا تكون هناك حالات شاذة في النموذج الموحد الثالث في هذه المرحلة.  
للتـقي نظرـةً على مخطط الاعتمـادـية المـوضـحـ في الشـكـلـ الآـتـيـ لـهـذـاـ المـثـالـ،ـ حيثـ تـتـمـثـلـ الـخـطـوـةـ الـأـوـلـيـ فيـ إـزـالـةـ الـمـجـمـوعـاتـ الـمـكـرـرـةـ.

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

تلـحـصـ الـاعـتمـادـيـاتـ الـمـوـضـحـةـ فيـ الشـكـلـ التـالـيـ عـلـىـ نـفـسـ الـفـرـقـةـ الـمـوـضـحـةـ الـتـيـ تـوـجـيـدـ قـاعـدـةـ بـيـانـاتـ الـمـدـرـسـةـ

:School database



الـاـخـتـصـارـاتـ الـمـسـتـخـدـمـةـ فيـ الشـكـلـ السـابـقـ هيـ كـمـاـ يـلـيـ:

• .partial dependency PD

• .transitive dependency TD

- FD: الاعتمادية الكاملة functional dependency، إذ يشير الاختصار FD إلى الاعتمادية الوظيفية full dependency على أساس احتصار الاعتمادية عادةً، ولكن استخدامنا لاحتصار FD على أساس احتصار الاعتمادية الكاملة full dependency في الشكل السابق فقط.

## 10.6 نموذج بويس-كود الموحد BCNF

قد تنتج حالات شاذة عندما يحتوي الجدول على أكثر من مفتاح مرشح candidate key على الرغم من أن العلاقة بصيغة نموذج 3NF، حيث يُعد نموذج بويس-كود الموحد BCNF أو Boyce-Codd normal form اختصاراً حالةً خاصةً من النموذج 3NF، وتكون العلاقة ضمن نموذج BCNF إذا وفقط إذا كان كل محدد candidate key مفتاحاً مرشحاً determinant.

### 10.6.1 المثال الأول عن نموذج BCNF

ضع في بالك الجدول التالي:

Advisor	Major	Student_id
Smith	Physics	111
Chan	Music	111
Dobbs	Math	320
White	Physics	671
Smith	Physics	803

القواعد الدلالية semantic rules - أي قواعد العمل المطبقة على قاعدة البيانات- لهذا الجدول هي:

1. يجوز لكل طالب Student التخصص في عدة مواد.

2. يكون لكل طالب معين مدرسًا واحدًا فقط لكل تخصص Major.

3. لكل تخصص عدة مدرسين.

4. يدرس كل مدرس تخصصاً واحداً فقط.

5. يدرس كل مدرس عدة طلاب في تخصص واحد.

الاعتمادات الوظيفية لهذا الجدول مذكورة أدناه، فالأولى هي مفتاح مرشح candidate key، والثانية ليست كذلك.

```
Student_id, Major → Advisor
Advisor → Major
```

تشمل الحالات الشاذة لهذا الجدول ما يلي:

1. الحذف Delete: مثل حالة حذف الطالب معلومات المدرس.
2. الإدخال Insert: مثل حالة احتياج المدرس الجديد إلى وجود طالب.
3. التحديث Update: مثل الحالات المتناقضة.

ليـست السـمة المـفردـة single attribute مـفتـاحـاً مـرـشـحاً.

يمـكـن أـن يـكـون المـفـتـاح الرـئـيـسي PK هو Student\_id, Advisor, أو Student\_id, Major، ويـمـكـنـك إـنـشـاء جـدـولـين جـدـيـدـين، لـتـقـلـيلـ الـعـلـاقـة St\_Maj\_Adv إـلـىـ النـمـوذـج BCNF كـمـاـ يـليـ:

St_Adv (Student_id, Advisor)
Adv_Maj (Advisor, Major)

• جـدول St\_Adv

Advisor	Student_id
Smith	111
Chan	111
Dobbs	320
White	671
Smith	803

• جـدول Adv\_Maj

Major	Advisor
Physics	Smith
Music	Chan
Math	Dobbs
Physics	White

## 10.6.2 المـثـالـ الثـانـيـ عـنـ نـمـوذـجـ BCNF

انـظـرـ الجـدولـ التـالـي Client\_Interview

RoomNo	StaffNo	InterviewTime	InterviewDate	ClientNo
G101	SG5	10.30	13-May-02	CR76
G101	SG5	12.00	13-May-02	CR56
G102	SG37	12.00	13-May-02	CR74

G102 SG5 10.30 1-July-02 CR56

FD1 - ClientNo, InterviewDate  $\rightarrow$  InterviewTime, StaffNo, RoomNo (PK)

FD2 - staffNo, interviewDate, interviewTime  $\rightarrow$  clientNO  
(candidate key: CK)

FD3 – roomNo, interviewDate, interviewTime  $\rightarrow$  staffNo, clientNo  
(CK)

FD4 – staffNo, interviewDate  $\rightarrow$  roomNo

تكون العلاقة بصيغة نموذج BCNF إذا وفقط إذا كان كل محدد determinant مفتاحاً مرشحاً.

نحو بحاجة إلى إنشاء جدول يتضمن أول ثلاثة اعتمادات كاملة FD -أي جدول2 Client\_Interview، وانشاء جدول آخر -أي جدول StaffRoom- للاعتمادية الكاملة FD الرابعة.

## • حدوٰ ج 2، Client Interview

ClientNo	InterviewDate	InterViewTime	StaffNo
CR76	13-May-02	10.30	SG5
CR56	13-May-02	12.00	SG5
CR74	13-May-02	12.00	SG37
CR56	1-July-02	10.30	SG5

• حدوٰر ، StaffRoom :

RoomNo	StaffNo	StaffNo
G101	13-May-02	SG5
G102	13-May-02	SG37
G102	1-July-02	SG5

## 10.7 التوحيد وتصعيم قواعد البيانات

تَأَكَّدُ أَثْنَاءَ عَمَلِيَّةِ تَوْحِيدِ تَصْمِيمِ قَاعِدَةِ الْبَيَانَاتِ مِنْ تَوَافُقِ الْكَيَانَاتِ الْمُقْتَرَحةَ لِلنَّمُوذِجِ الْمُوَحَّدِ الْمُطَلُّوبِ قَبْلَ إِنْشَاءِ تُبْنِيِ الْجَدُولِ.

ضممت العديد من قواعد البيانات واقعياً بصورة غير سليمة، أو أثقل كاھلها بحالات شاذة عند تعديلها بصورة غير سليمة خلا ، فتة زمنية.

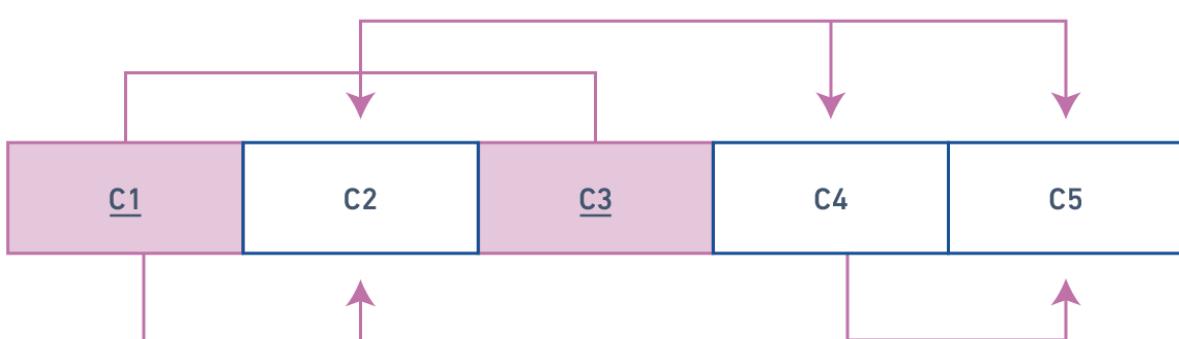
قد يُطلب منك إعادة تصميم قواعد البيانات الحالية، وتعديلها، كما يمكن أن يكون ذلك مهمةً كبيرةً إذا أجريت عملية التوحيد على الجداول بصورةٍ غير صحيحة.

## 10.8 المصطلحات الأساسية وال اختصارات

- نـمـوذـج بـويـسـكـودـ المـوـحـدـ أوـ BCNFـ: وهو حالة خاصة من نـمـوذـجـ .3NFـ
- الـنـمـوذـجـ المـوـحـدـ الـأـوـلـ أوـ 1NFـ: يـسـمـحـ بـقـيمـ غـيرـ مـكـرـرـةـ فـقـطـ عـنـدـ تـقـاطـعـ كـلـ صـفـ وـعـمـودـ، لـذـلـكـ لـاـ تـوـجـدـ مـجـمـوـعـاتـ مـكـرـرـةـ.
- الـتـوـحـيـدـ: عـمـلـيـةـ تـحـدـيـدـ مـقـدـارـ التـكـرـارـ الـمـوـجـودـ فـيـ الجـدـوـلـ.
- الـنـمـوذـجـ المـوـحـدـ الـثـانـيـ أوـ 2NFـ: يـجـبـ أـنـ يـكـوـنـ لـلـعـلـقـةـ صـيـغـةـ نـمـوذـجـ 1NFـ، كـمـاـ يـجـبـ اـشـتـهـارـ الـمـفـتـاحـ الرـئـيـسـيـ PKـ عـلـىـ سـمـةـ وـاحـدـةـ.
- الـقـوـاعـدـ الـدـلـالـيـةـ semantic rulesـ: قـوـاعـدـ الـعـلـمـ الـمـطـبـقـةـ عـلـىـ قـاعـدـةـ الـبـيـانـاتـ.
- الـنـمـوذـجـ المـوـحـدـ الـثـالـثـ أوـ 3NFـ: يـجـبـ أـنـ يـكـوـنـ لـلـعـلـقـةـ صـيـغـةـ نـمـوذـجـ 2NFـ، كـمـاـ يـجـبـ إـزـالـةـ جـمـيـعـ الـاعـتـمـادـيـاتـ الـمـتـعـدـيـةـ transitive dependenciesـ، فـقـدـ لـاـ تـعـتـمـدـ السـمـةـ الـتـيـ لـيـسـ مـفـتـاحـاـ non-key attributeـ اـعـتـمـادـاـ وـظـيـفـيـاـ عـلـىـ سـمـةـ أـخـرـىـ لـيـسـ مـفـتـاحـاـ أـيـضاـ.

## 10.9 تمارين

- ما هو التوحيد normalization؟
- متى يكون جدول ما في نـمـوذـجـ 1NFـ؟
- متى يكون جدول ما في نـمـوذـجـ 2NFـ؟
- متى يكون جدول ما في نـمـوذـجـ 3NFـ؟
- عـرـفـ وـنـاقـشـ كـلـ مـنـ الـاعـتـمـادـيـاتـ الـمـشـارـ إـلـيـهاـ فـيـ مـخـطـطـ الـاعـتـمـادـيـةـ الـمـوـضـحـ فـيـ الشـكـلـ التـالـيـ:



6. تـسـتـخـدـم كـلـيـة جـامـعـيـة college جـديـدة بـنـيـة الجـدـول المـوـضـحـة فيـ الجـدـول التـالـي، وـذـلـك لـتـبـعـ الطـلـابـ والمـقـرـراتـ، وـبـالـتـالـي، اـرـسـمـ مـخـطـطـ الـاعـتـمـادـيـة لـهـذـا الجـدـولـ.

Attribute Name	Sample Value	Sample Value	Sample Value
StudentID	1	2	3
StudentName	John Smith	Sandy Law	Sue Rogers
CourseID	2	2	3
CourseName	Programming Level 1	Programming Level 1	Business
Grade	75%	61%	81%
CourseDate	Jan 5 <sup>th</sup> , 2014	Jan 5th, 2014	Jan 7 <sup>th</sup> , 2014

7. اـعـرـضـ الجـدـاوـلـ بـصـيـغـةـ النـمـوـذـجـ المـوـحـدـ الثـالـثـ الـتـيـ سـتـنـشـئـهـاـ لـإـصـلـاحـ الـمـشـكـلـاتـ الـتـيـ وـاجـهـتـهـاـ باـسـتـخـدـامـ مـخـطـطـ الـاعـتـمـادـيـةـ الـذـيـ رـسـمـتـهـ لـلـتوـ،ـ ثـمـ اـرـسـمـ مـخـطـطـ الـاعـتـمـادـيـةـ لـلـجـدـولـ الثـابـتـ.

8. توـفـرـ الـوـكـالـةـ الـتـيـ تـسـمـيـ Coverـ Instantـ موـظـفـينـ بـدـوـامـ جـزـئـيـ أوـ مـؤـقـتـ لـلـفـنـادـقـ فـيـ اـسـكـلـنـدـاـ،ـ إـذـ يـوـضـحـ الشـكـلـ الـآـتـيـ الـوقـتـ الـذـيـ يـقـضـيـهـ موـظـفـوـ الـوـكـالـةـ فـيـ الـعـمـلـ فـيـ فـنـادـقـ مـخـلـفـةـ،ـ حـيـثـ يـكـوـنـ رقمـ التـأـمـيـنـ الـوطـنـيـ NINـ -ـأـيـ national insurance numberـ فـرـيـدـاـ لـكـلـ موـظـفـ.

استـخـدـمـ الجـدـولـ التـالـيـ لـلـإـجـابـةـ عـلـىـ السـؤـالـيـنـ الـآـتـيـيـنـ:

NIN	ContractNo	Hours	eName	hNo	hLoc
1135	C1024	16	Smith J.	H25	East Killbride
1057	C1024	24	Hocine D.	H25	East Killbride
1068	C1025	28	White T.	H4	Glasgow
1135	C1025	15	Smith J.	H4	Glasgow

1. هـذـاـ الجـدـولـ عـرـضـةـ لـحـالـاتـ تـحـدـيـثـ شـاذـةـ لـذـاـ قـدـمـ أـمـثـلـةـ عـلـىـ حـالـاتـ شـاذـةـ لـلـإـدـخـالـ وـالـحـذـفـ وـالـتـحـدـيـثـ.

2. ظـبـقـ عـمـلـيـةـ التـوـحـيـدـ عـلـىـ هـذـاـ الجـدـولـ لـيـصـبـحـ لـهـ صـيـغـةـ النـمـوـذـجـ المـوـحـدـ الثـالـثـ،ـ معـ التـأـكـدـ مـنـ ذـكـرـ أيـ اـفـتـرـاضـاتـ.

9. اـمـلـاـ الفـرـاغـاتـ:

- يـنـتـجـ عـنـ نـمـوـذـجـاـ مـوـحـدـاـ أـقـلـ.
- تـسـمـ السـمـةـ الـتـيـ تـحـدـدـ قـيـمـتـهاـ قـيـمـاـ أـخـرىـ دـاـخـلـ صـفـ .
- السـمـةـ الـتـيـ لـاـ يـمـكـنـ تـقـسـيمـهـاـ تـوـصـفـ بـأـنـهـ .

- يشير \_\_\_\_\_ إلى مستوى التفاصيل الذي تمثله القيم المخزنة في صف الجدول.
- يجب ألا يحتوي الجدول العلائقى على مجموعات \_\_\_\_\_.

# 11. عملية تطوير قواعد البيانات

يتمثل أحد الجوانب الأساسية لهندسة البرمجيات في تقسيم عملية التطوير إلى سلسلة من المراحل أو الخطوات، حيث ترکز كل مرحلة منها على جانب واحد من جوانب التطوير.

يشار أحياناً إلى مجموعة هذه الخطوات بدورة حياة تطوير البرمجيات - software development life cycle أو SDLC اختصاراً، حيث ينتقل المنتج البرمجي عبر مراحل دورة الحياة هذه - في بعض الأحيان بصورة متكررة أثناء ضبطه أو إعادة تطويره- حتى يتوقف استخدامه في النهاية، كما يمكن التحقق من كل مرحلة في دورة الحياة للتأكد من صحتها قبل الانتقال إلى المرحلة التالية في الحالة المثالية.

## 11.1 دورة حياة تطوير البرمجيات - نموذج الشلال Waterfall

لنببدأ بإلقاء نظرة عامة على نموذج الشلال waterfall model الذي هو أحد نماذج تمثيل دورة حياة عملية تطوير البرمجيات Software Development Life Cycle كما ستجده في معظم كتب هندسة البرمجيات.

يوضح هذا الشكل الشلالي الموجود في الشكل الآتي نموذج شلال عام يمكن تطبيقه على أية عملية تطوير لنظام حاسوبي، حيث يُظهر هذا النموذج العملية على أساس تسلسل صارم من الخطوات بأن يكون خرج خطوة واحدة دخلاً للخطوة التالية، كما يجب إكمال كل خطوة قبل الانتقال إلى الخطوة التالية.



يمكننا استخدام عملية نموذج الشلال على أساس وسيلة لتحديد المهام المطلوبة مع دخل وخرج كل نشاط activity، إذ المهم هنا هو مجالات الأنشطة التي يمكن تلخيصها على النحو التالي:

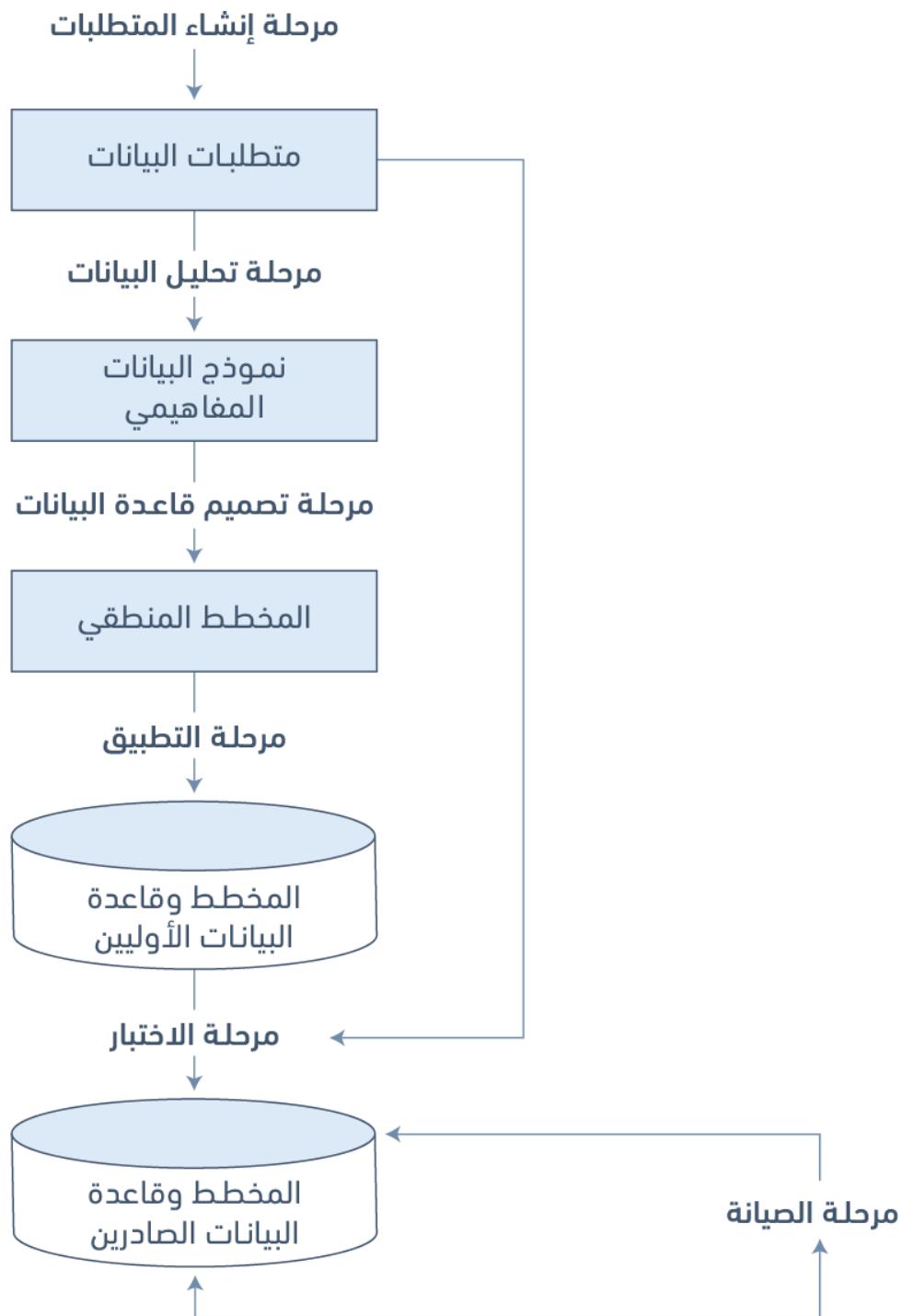
- تتضمن مرحلة إنشاء المتطلبات Establishing requirements التشاور والاتفاق مع أصحاب المصلحة حول ما يريدونه ويحتاجون إليه من النظام، والتي يُعبر عنها بما يسمى وثيقة المتطلبات وباللغة الإنجليزية statement of requirements.

- تبدأ مرحلة **التحليل Analysis** بالنظر في وثيقة المتطلبات وتنتهي من خلال إنتاج مواصفات النظام system specification، حيث تُعدّ المواصفات تمثيلاً رسمياً لما يجب على النظام فعله، ويعبر عنها عبارات مستقلة عن كيفية تطبيقها.
- تبدأ مرحلة **التصميم Design** بمواصفات النظام وينتج عنها وثائق التصميم، كما تقدم هذه المرحلة وصفاً تفصيلياً لكيفية بناء النظام.
- مرحلة **التطبيق Implementation** هي بناء نظام حاسوبي وفقاً لوثيقة تصميم معينة مع مراعاة البيئة التي سيعمل فيها النظام، مثل العتاد، والبرمجيات المتوفرة للتطوير؛ كما قد تُنفذ مرحلة التطبيق على مراحل باستخدام نظام أولي يمكن التحقق من صحته واختباره قبل إصدار النظام النهائي للاستخدام.
- توازن مرحلة **الاختبار Testing** النظام المُطبّق مع وثائق التصميم ومواصفات المتطلبات، وتنتج هذه المرحلة تقرير قبول، أو قائمةً بالأخطاء والزلات البرمجية bugs التي تتطلب مراجعة عمليات التحليل، والتصميم، والتطبيق لتصحيحها، أي تُعدّ مرحلة الاختبار عادةً المهمة التي تؤدي إلى تكرار نموذج الشلال خلال دورة الحياة.
- تتضمن مرحلة **الصيانة Maintenance** التعامل مع تغيرات المتطلبات، أو بيئه التطبيق، أو إصلاح الزلات البرمجية، أو نقل النظام إلى بيانات جديدة مثل ترحيل نظام من حاسوب مستقل إلى محطة عمل يونكس أو بيئه متصلة بالشبكة، كما سيعاد النظر في دورة حياة الشلال بصورة متكررة بسبب احتواء مرحلة الصيانة على تحليل التغيرات المطلوبة، وتصميم حل، وتطبيقه، واختباره على مدى حياة نظام برمجي جرت صيانته.

## 11.2 دورة حياة قاعدة البيانات Database Life Cycle

نستطيع استخدام دورة الشلال مثل أساس لنموذج تطوير قاعدة البيانات الذي يتضمن ثلاثة افتراضات هي:

1. يمكننا فصل تطوير قاعدة البيانات عن عمليات المستخدم التي تستخدم قاعدة البيانات، أي تحديد وإنشاء تخطيط schema لتعريف البيانات في قاعدة البيانات.
2. يمكننا استخدام معمارية التخطيطات الثلاثة three-schema architecture مثل أساس لتمييز الأنشطة المرتبطة بالتخطيط.
3. يمكننا تمثيل القيود constraints لفرض دلالات البيانات مرّةً واحدةً في قاعدة البيانات عوضاً عن فرضها على كل عملية مستخدم تستخدم البيانات.



يمكننا باستخدام هذه الافتراضات والشكل السابق رؤية أنّ هذا المخطط يمثّل نموذجاً لأنشطة وخرجها لتطوير قاعدة البيانات، فهذا المخطط ليس قابلاً للتطبيق على النهج العلائقى فقط وإنما يُطلق على أية صنف class من نظم إدارة قواعد البيانات DBMS أيضًا.

يُعَدّ تطوير تطبيقات قواعد البيانات عمليةً للحصول على متطلبات العالم الحقيقي real-world، وتحليل المتطلبات، وتصميم البيانات ووظائف النظام، ثم تطبيق العمليات في النظام.

## 11.3 جمع المتطلبات Requirements Gathering

تُعدّ مرحلة جمع المتطلبات requirements gathering الخطوة الأولى في نموذج الشلال، ويجب على مصممي قاعدة البيانات خلال هذه الخطوة إجراء مقابلات مع العملاء-أي مستخدمي قاعدة البيانات- لفهم النظام المقترن والحصول على البيانات والمتطلبات الوظيفية، وتوثيقها، كما تكون نتيجة هذه الخطوة وثيقة تتضمن المتطلبات التفصيلية التي قدمها المستخدمون.

تتضمن مرحلة إنشاء المتطلبات Establishing requirements التشاور والاتفاق بين جميع المستخدمين بشأن البيانات الثابتة persistent data التي يرغبون في تخزينها مع الاتفاق على معنى عناصر البيانات وتفسيرها، كما يلعب مسؤول البيانات دوراً رئيسياً في هذه العملية لأنّه يستعرض القضايا التجارية، والقانونية، والأخلاقية داخل المؤسسة التي تؤثّر على متطلبات البيانات.

تُستخدم وثيقة متطلبات البيانات data requirements document لتأكيد فهم المتطلبات مع المستخدمين، فلا ينبغي أن تكون رسميةً أو مشفرةً بمستوى عالٍ لضمان سهولة فهمها.

يجب أن تقدّم هذه الوثيقة ملخصاً موجزاً لمتطلبات جميع المستخدمين-أي ليس مجرد مجموعة من الأفراد فقط-، وذلك لأنّ الهدف هو تطوير قاعدة بيانات مشتركة واحدة.

يجب ألا تصف المتطلبات ككيفية معالجة البيانات، بل تصف عناصر البيانات، والسمات attributes التي تمتلكها، والقيود المطبقة، وال العلاقات التي تربط بين عناصر البيانات.

## 11.4 التحليل Analysis

تبدأ مرحلة تحليل البيانات Data analysis بوثيقة متطلبات البيانات، ثم ينتج عنها نموذج بيانات مفاهيمي conceptual data model. الهدف من التحليل هو الحصول على وصف تفصيلي للبيانات التي ستتناسب مع متطلبات المستخدم، بحيث يجري التعامل مع خصائص البيانات ذات المستوى العالي والمنخفض واستخدامها. تتضمن هذه الخصائص المجال المحتمل من القيم التي يمكن السماح بها للسمات، مثل: رمز مقررات الطالب student course code، وعنوان المقرر course title، ونقطة الائتمان credit points في قاعدة بيانات المدرسة على سبيل المثال.

يُوفّر نموذج البيانات المفاهيمي تمثيلاً رسمياً مشتركاً لما يجري توصيله بين العملاء والمطوريين أثناء تطوير قاعدة البيانات، فهذا النموذج يركز على البيانات في قاعدة البيانات، بغض النظر عن الاستخدام النهائي لتلك البيانات في عمليات المستخدم، أو تطبيق البيانات في بيئات حاسوبية محددة، لذلك يهتم نموذج البيانات المفاهيمي بمعنى البيانات وبنيتها، وليس بالتفاصيل التي تؤثّر على كيفية تطبيقها.

إذاً يُعدّ نموذج البيانات المفاهيمي تمثيلاً رسمياً للبيانات التي يجب أن تحتويها قاعدة البيانات، والقيود التي يجب على البيانات تلبيتها، كما يجب التعبير عن ذلك بمصطلحات مستقلة عن كيفية تنفيذ النموذج.

لذلك يرُكز التحليل على الأسئلة التي تحتوي عبارات مثل عبارة "ما هو المطلوب؟" وليس على الأسئلة التي تحتوي عبارات مثل عبارة "كيف يتحقق ذلك؟".

## 11.5 التصميم المنطقي Logical Design

تبدأ مرحلة تصميم قاعدة البيانات بنموذج بيانات مفاهيمي وينتج عنها مواصفات التخطيط المنطقي logical schema الذي سيحدد نوع نظام قاعدة البيانات المطلوب -أي سيحدد إن كان من نوع شبكي، أو علائقي، أو كائني التوجه.

لا يزال التمثيل العلائقي relational representation مستقلاً عن أي نظام إدارة قواعد البيانات DBMS، فهو نموذج بيانات مفاهيمي آخر.

يمكننا استخدام التمثيل العلائقي لنموذج البيانات المفاهيمي على أساس دخل لعملية التصميم المنطقي، وخرج هذه المرحلة هو مواصفات علائقية مفصلة أي تخطيط منطقي لجميع الجداول والقيود الازمة لتلبية وصف البيانات في نموذج البيانات المفاهيمي.

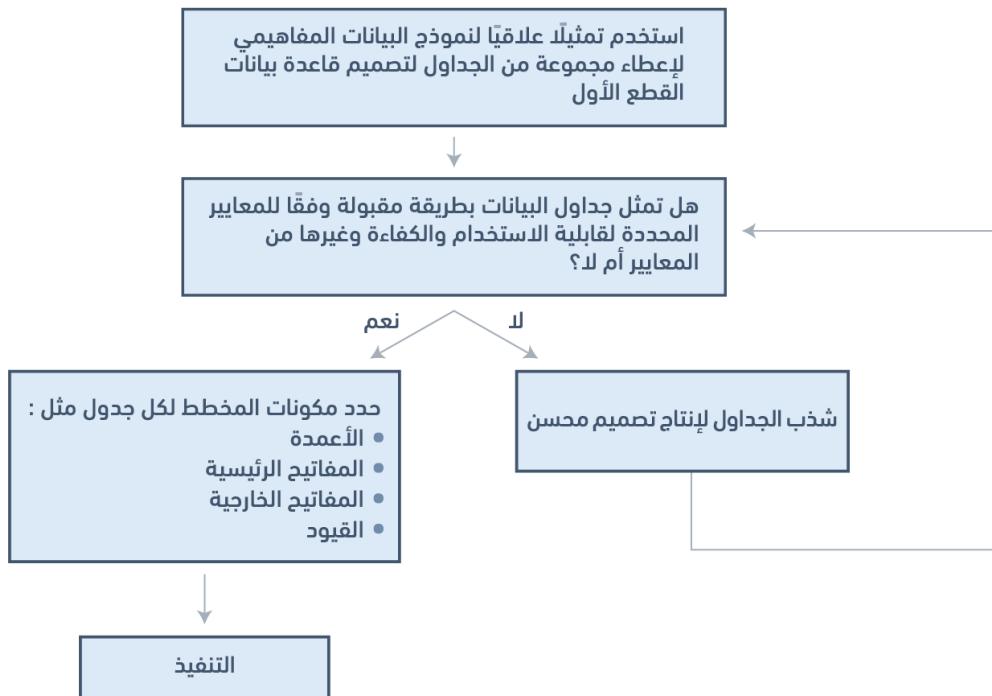
تختار الجداول الأكثر ملاءمة أثناء نشاط التصميم لتمثيل البيانات في قاعدة بيانات، ولكن يجب أخذ هذه الاختيارات في الحسبان معايير التصميم المختلفة بما في ذلك على سبيل المثال مرونة التغيير، والتحكم في التضاعف أو الاستنساخ duplication، وأفضل طريقة لتمثيل القيود. تحدد الجداول المحددة بالخطيط المنطقي البيانات المخزنة وكيفية معالجتها في قاعدة البيانات.

يتجه مصممو قواعد البيانات الملمون بقواعد البيانات العلائقية ولغة الاستعلامات الهيكلية SQL للذهاب مباشرةً إلى مرحلة التطبيق بعد إنتاج نموذج البيانات المفاهيمي لكن لا يؤدي مثل هذا التحول المباشر للتمثيل العلائقي إلى جداول SQL بالضرورة إلى قاعدة بيانات تحتوي على جميع الخصائص المرغوبة مثل الكمال completeness والسلامة integrity والمرنة flexibility والقابلية efficiency، لكن لا يعني هذا أن التحول المباشر إلى جداول SQL ينتج قاعدة بيانات جيدة تلقائياً.

ستمثل هذه الخطوة الأولى بدقة الجداول والقيود الازمة لتلبية وصف نموذج البيانات المفاهيمي، وبالتالي ستلبي متطلبات الكمال والسلامة، ولكنها قد تكون غير مرونة، أو قد تقدم قابلية استخدام ضعيفة، يُثنى flexed التصميم الأول بعد ذلك لتحسين جودة تصميم قاعدة البيانات، ويهدف مصطلح الثنوي Flexing إلىأخذ الأفكار المتزامنة من شيء مثني لغرض مختلف وتشذيب جوانب من هذا الشيء. أي الوصول إلى الغاية نفسها بطريقة وفكرة أخرى تحقق المقصود.

يلخص الشكل الآتي الخطوات التكرارية الموجودة في تصميم قاعدة البيانات بناءً على النظرة العامة المقدمة، كما يكون الغرض الرئيسي من هذا الشكل هو التمييز بين الهدف العام للجداول التي يجب استخدامها عن التعريف المفصل للأجزاء المكونة لكل جدول، حيث تُدرس هذه الجداول واحداً تلو الآخر رغم أنها ليست

مستقلةً عن بعضها البعض، كما سيؤدي كل تكرار يتضمن مراجعةً للجداول إلى تصميم جديد، ويشار إلى هذه التصاميم الجديدة معاً باسم تصاميم القطع الثاني *second-cut designs* حتى لو تكررت العملية لأكثر من حلقة واحدة.



أولاً، ليس من الضروري تلبية جميع متطلبات المستخدم التي يمثلها نموذج بيانات مفاهيمي معين بواسطة قاعدة بيانات واحدة، كما يوجد أسباب مختلفة لتطوير أكثر من قاعدة بيانات، مثل: الحاجة إلى عملية مستقلة في موقع مختلف، أو التحكم الإداري ببيانات قواعد البيانات، لكن إذا احتوت مجموعة قواعد البيانات على بيانات مضاعفة وكان المستخدمون بحاجة للوصول إلى البيانات في أكثر من قاعدة بيانات، وهناك أسباب محتملة لتلبّي قاعدة بيانات واحدة متطلبات متعددة، وإلا فيجب فحص المشاكل المتعلقة بمضاعفة البيانات وتوزيعها.

ثانياً، أحد الافتراضات حول تطوير قاعدة البيانات هو أنه يمكننا فصل تطوير قاعدة البيانات عن تطوير عمليات المستخدم التي تستفيد منها، ويستند ذلك إلى توقع تحديد جميع البيانات المطلوبة بواسطة عمليات المستخدم المحددة حالياً، وإمكانية الوصول إليها بمجرد تطبيق قاعدة البيانات، لكننا نطلب أيضاً المرونة للسماح بتلبية تغيرات المتطلبات المستقبلية، كما يمكن التنبؤ بالطلبات الشائعة التي ستُقدم إلى قاعدة البيانات عند تطوير قاعدة بيانات لبعض التطبيقات، وبالتالي يمكننا تحسين تصميمنا للطلبات الأكثر شيوعاً.

ثالثاً، تعتمد العديد من جوانب تصميم قاعدة البيانات وتطبيقاتها في المستوى التفصيلي على نظام إدارة قاعدة البيانات DBMS المستخدم، فإذا كان اختيار نظام إدارة قواعد البيانات ثابتاً أو أُجري قبل مهمة التصميم، فيمكن استخدام هذا الاختيار لتحديد معايير التصميم بدلاً من الانتظار حتى مرحلة التطبيق، أي يمكن دمج

قرارات التصميم لنظام إدارة قاعدة البيانات DBMS معين عوّضاً عن إنتاج تصميم عام، ثم تكييفه مع نظام إدارة قاعدة البيانات DBMS أثناء التطبيق.

ليس غريباً العثور على تصميم مفرد لا يمكنه تلبية جميع خصائص قاعدة البيانات الجيدة في الوقت نفسه، لذلك من المهم أن يعطي المصمم الأولوية لهذه الخصائص، ويكون ذلك عادةً باستخدام معلومات من مواصفات المتطلبات، مثل: تحديد ما إذا كانت السلامة أهم من الكفاءة، وما إذا كانت قابلية الاستخدام أهم من المرونة في تطوير معين.

ستحدّد تعليمات لغة تعريف البيانات SQL أو DDL اختصاراً- data definition language الخاصة بلغة التخطيط المنطقي في نهاية مرحلة التصميم، حيث تصف لغة DDL قاعدة البيانات التي يجب تطبيقها لتلبية متطلبات المستخدم.

## 11.6 التطبيق Implementation

تضمن مرحلة التنفيذ أو التطبيق Implementation بناء قاعدة بيانات وفقاً لمواصفات التخطيط المنطقي، والذي سيتضمن مواصفات تخطيط التخزين storage schema المناسب، وفرض الأمان، والتخطيط الخارجي، وما إلى ذلك، كما يتأثر التطبيق بشدة باختيار نظم إدارة قواعد البيانات المتاحة، وأدوات قواعد البيانات، وبيئة التشغيل.

هناك مهام إضافية تتجاوز مجرد إنشاء تخطيط قاعدة بيانات database schema وتطبيق القيود، إذ يجب إدخال البيانات في الجداول، ومعالجة القضايا المتعلقة بالمستخدمين وعمليات المستخدم، كما يجب دعم الأنشطة الإدارية المرتبطة بالجوانب الأوسع لإدارة بيانات الشركة.

نريد معالجة أكبر عدد ممكн من هذه القضايا الموضحة أدناه داخل نظام إدارة قواعد البيانات تماشياً مع نهج نظم إدارة قواعد البيانات.

يتطلب تطبيق التخطيط المنطقي عملياً في نظام إدارة قواعد البيانات DBMS معرفةً مفصلةً للغاية بالميزات والفوائد المحددة التي يجب تقديمها من قبل نظام إدارة قواعد البيانات.

ستشمل المرحلة الأولى من التطبيق انسجام متطلبات التصميم مع أفضل أدوات التطبيق المتاحة ثم استخدام تلك الأدوات للتطبيق، وذلك مثالياً وتماشياً مع الممارسة الجيدة لهندسة البرمجيات، كما قد يتضمن ذلك في قواعد البيانات على اختيار منتجات البائعين ذات متغيرات من نظام إدارة قواعد البيانات DBMS ولغة SQL الأكثر ملائمة لقاعدة البيانات التي تحتاج إلى تطبيقها، لكننا لا نعيش في عالم مثالى، كما سُتُّخذ في كثير من الأحيان قرارات اختيار العتاد والقرارات المتعلقة بنظام إدارة قواعد البيانات DBMS قبل النظر في تصميم قاعدة البيانات بوقت طويل، وبالتالي، يمكن أن يتضمن التطبيق ثنياً إضافياً للتصميم للتغلب على محدوديات البرمجيات أو العتاد.

## 11.7 تحقيق التصميم

نحتاج إلى إنشاء قاعدة بياناتنا بعد إنشاء التصميم المنطقي وفقاً للتعرifات التي أنتجناها، كما يُحتمل أن يتضمن التطبيق مع نظام إدارة قواعد البيانات DBMS العلائقى استخدام لغة SQL لإنشاء جداول وقيود تلبي وصف التخطيط المنطقي واختيار تخطيط التخزين المناسب -إذا كان نظام إدارة قواعد البيانات DBMS يسمح بهذا المستوى من التحكم.

تتمثل إحدى طرق تحقيق ذلك في كتابة تعليمات لغة SQL المناسبة في ملف يمكن لنظام إدارة قواعد البيانات DBMS تفويذه، بحيث يكون هناك سجل مستقل أو ملف نصي من تعليمات لغة SQL التي تعرّف قاعدة البيانات؛ أما الطريقة الأخرى فهي العمل تفاعلياً باستخدام أداة قاعدة بيانات مثل الأداتين Microsoft Access أو SQL Server Management Studio.

مهما كانت الآلية المستخدمة لتطبيق التخطيط المنطقي، فالنتيجة هي أن قاعدة البيانات -مع الجداول والقيود- معروفة ولكنها لن تحتوي على بيانات لعمليات المستخدم.

## 11.8 ملء قاعدة البيانات

يوجد طريقتان لملء الجداول بعد إنشاء قاعدة البيانات؛ إما من بيانات موجودة أو من خلال استخدام تطبيقات المستخدم المطورة لقاعدة البيانات.

قد تكون هناك بيانات موجودة من قاعدة بيانات أخرى وذلك بالنسبة لبعض الجداول فمثلاً توقع عند إنشاء قاعدة بيانات لمستشفى وجود بعض السجلات بالفعل لجميع الموظفين المراد تضمينهم في قاعدة البيانات، كما يمكن أيضاً إحضار البيانات من وكالة خارجية مثل قوائم العناوين التي تُجلب بصورة متكررة من شركات خارجية أو يمكن إنتاجها أثناً مئه إدخال بيانات كبيرة -أي يمكن إجراء تحويل السجلات اليدوية المطبوعة إلى ملفات حاسوبية بواسطة وكالة إدخال بيانات- ويُعدّ استخدام وسائل الاستيراد والتصدير الموجودة في نظام إدارة قواعد البيانات DBMS أبسط طريقة لملء قاعدة البيانات في مثل هذه الحالات.

توفر عادةً وسائل لاستيراد وتصدير البيانات بتنسيقات قياسية مختلفة، وتُعرف هذه الوظائف أيضاً في بعض الأنظمة باسم تحميل loading البيانات وتفريغها unloading، كما يتيح الاستيراد إمكانية نسخ ملف البيانات مباشرةً إلى جدول.

إذا جرى الاحتفاظ بالبيانات بتنسيق ملف غير مناسب لاستخدام عملية الاستيراد، فيجب إعداد برنامج تطبيقي يقرأ البيانات القديمة، ويحولها حسب الضرورة، ثم يدخلها في قاعدة البيانات باستخدام شيفرة لغة SQL الذي أُنتجه خصيصاً من أجل هذا الهدف.

يُسمى نقل كميات كبيرة من البيانات الموجودة إلى قاعدة بيانات بالتحميل المجمع **bulk load**، وقد يتضمن التحميل المجمع للبيانات كميات كبيرةً جدًا من البيانات المحمّلة أي تحميل جدول في نفس الوقت، لذلك قد تجد وسائل في نظام إدارة قواعد البيانات DBMS لتأجيل فحص قيد حتى نهاية التحميل المجمع.

## 11.9 إرشادات لتطوير مخطط ER

لاحظ أن هذه الإرشادات العامة ستتساعد في تطوير أساس قوي لتصميم قاعدة البيانات الفعلية أي النموذج المنطقي:

1. وّثّق جميع الكيانات المكتشفة خلال مرحلة جمع المعلومات.
2. وّثّق جميع السمات التي تنتمي إلى كل كيان، وحدّد المفاتيح المرشحة **candidate keys**، والمفاتيح الرئيسية **primary keys**، كما تأكّد من اعتمادية جميع السمات التي ليست مفاتيح **non-key attributes** لكل كيان بصورة كاملة على المفتاح الرئيسي.
3. طّور مخطط ER الأولي وراجعه مع الأشخاص المناسبين، وتذّرّأً أن هذه عملية تكرارية.
4. أنشئ كيانات -أي جداول- جديدة للسمات متعددة القيم والمجموعات المكرّرة، ثم ضمّن هذه الكيانات -أي الجداول- الجديدة في مخطط ER، وراجع ذلك مع الأشخاص المناسبين.
5. تحقّق من نمذجة الكيان العلائقى ER عبر تطبيق عملية التوحيد **normalizing** على الجداول.

## 11.10 مصطلحات أساسية

- **التحليل analysis:** تبدأ مرحلة التحليل من خلال النظر في وثيقة المتطلبات وتنتهي من خلال إنتاج مواصفات النظام.
- **التحميل المجمع bulk load:** هو نقل كميات كبيرة من البيانات الموجودة إلى قاعدة بيانات.
- **وثيقة متطلبات البيانات data requirements document:** وتنسخدم هذه الوثيقة لتأكيد فهم المتطلبات مع المستخدم.
- **التصميم design:** تبدأ مرحلة التصميم بمواصفات النظام، وينتّج عنها وثائق التصميم، كما يوّفر وصفًا تفصيليًّا لكيفية بناء النظام.
- **تحديد المتطلبات establishing requirements:** تتضمن هذه المرحلة التشاور والاتفاق مع أصحاب المصلحة على ما يريدونه من النظام، كما يعبّر عنها بوثيقة المتطلبات.
- **الثنبي flexing:** مصطلح يهدف إلىأخذ الأفكار المتزامنة من شيء مثني لغاية مختلفة وإضعاف جوانب من هذا الشيء عند ثبيه.

- **التطبيق implementation**: بناء نظام حاسوبي وفقاً لوثيقة تصميم معينة.
- **الصيانة maintenance**: تتضمن هذه المرحلة التعامل مع تغيرات المتطلبات، أو بيئة التطبيق، أو إصلاح الأخطاء، أو نقل النظام إلى بيئات جديدة.
- **جمع المتطلبات requirements gathering**: عملية يقابل خلالها مصمم قاعدة البيانات مستخدماً قاعدة البيانات لفهم النظام المقترن، وذلك للحصول على البيانات، والمتطلبات الوظيفية، وتوثيقها.
- **تصاميم القطع الثاني second-cut designs**: مجموعة التكرارات التي يتضمن كل منها مراجعة الجداول التي تؤدي إلى تصميم جديد.
- **دورة حياة تطوير البرمجيات SDLC أو software development life cycle**: سلسلة لخطوات عملية تطوير قواعد البيانات.
- **الاختبار testing**: توازن هذه المرحلة النظام المطبق مع وثائق التصميم ومواصفات المتطلبات، وينتج عنها تقرير قبول.
- **نموذج الشلال waterfall model**: يُظهر هذا النموذج عملية تطوير قاعدة البيانات مثل تسلسل صارم من الخطوات حيث يكون خرج خطوة دخلاً للخطوة التالية.
- **عملية نموذج الشلال waterfall process**: وسيلة لتحديد المهام المطلوبة لتطوير قاعدة البيانات، بالإضافة إلى دخل وخرج كل نشاط.

## 11.11 تمارين

1. اشرح نموذج الشلال، واذكر خطواته.
2. ماذا يعني الاختصار SDLC؟ وما الذي يمثله؟
3. ما الذي يجب تعديله في نموذج الشلال لاستيعاب تصميم قاعدة البيانات؟
4. اذكر الخطوات التكرارية الموجودة في تصميم قاعدة البيانات.

## 12. لغة الاستعلامات الهيكلية SQL

لغة الاستعلامات الهيكلية Structured Query Language -أو SQL اختصاراً- هي لغة قاعدة بيانات مصممة لإدارة البيانات الموجودة في نظام إدارة قواعد البيانات العلائقية.

طُورت شركة IBM لغة SQL في أوائل السبعينيات -عُرِفت بالإصدار 1986-، حيث صُمِّم الإصدار الأولي المسماً بلغة الاستعلامات الهيكلية الإنجليزية SEQUEL -اختصاراً للعبارة Structured English Query- لمعالجة واسترداد البيانات المخزنة في نظام خاص بشركة IBM وشبه علائقية لإدارة قواعد البيانات Language R. quasi-relational database management system.

قدّمت بعد ذلك شركة Oracle Corporation Relational Software Inc -والتي أصبحت الآن شركة Oracle- أول تطبيق متاح تجاريًّا للغة SQL والمسمى بـ Oracle V2 لحواسيب VAX في أواخر السبعينيات من القرن الماضي. تُستخدم العديد من أنظمة DBMS العلائقية المتاحة حالياً، مثل:

Oracle Database •

Microsoft SQL Server •

MySQL •

IBM DB2 •

IBM Informix •

Microsoft Access •

تُستخدم لغة قاعدة بيانات SQL في نظام DBMS من أجل:

- إنشاء بنى قواعد البيانات والجداول.
- إجراء الأعمال الأساسية لإدارة البيانات، مثل: الإضافة، والحذف، والتعديل.
- إجراء استعلامات معقدة لتحويل البيانات الأولية إلى معلومات مفيدة.

سوف نركز في هذا الفصل على استخدام لغة SQL لإنشاء بنى قواعد البيانات والجداول، باستخدام لغة SQL على أساس لغة تعريف بيانات DDL- أو data definition language - بصورة أساسية.

ستستخدم لغة SQL في فصل لاحق على أساس لغة معالجة بيانات data manipulation language أو DML لـ إدخال البيانات، وحذفها، و اختيارها، وتحديثها في جداول قاعدة البيانات.

## 12.1 إنشاء قاعدة بيانات

تتكون عبارات لغة SQL DDL الرئيسية من: عملية إنشاء قاعدة البيانات CREATE DATABASE، وعمليات الإنشاء CREATE والحذف DROP والتعديل ALTER على الجداول، إذ تُستخدم عبارة CREATE في لغة SQL لـ إنشاء بنى قواعد البيانات والجداول.

### 12.1.1 مثال عن إنشاء قاعدة بيانات

تُنشأ قاعدة بيانات جديدة تسمى SW باستخدام العبارة CREATE DATABASE SW بلغة SQL. الخطوة التالية بعد إنشاء قاعدة البيانات هي إنشاء جداول قاعدة البيانات.

التنسيق العام للأمر CREATE TABLE هو:

```
CREATE TABLE <tablename>
(
  ColumnName, Datatype, Optional Column Constraint,
  ColumnName, Datatype, Optional Column Constraint,
  Optional table Constraints
);
```

يكون Tablename اسم جدول قاعدة البيانات مثل جدول الموظف Employee، كما يتكون كل حقل من الأمر CREATE TABLE من ثلاثة أجزاء، هي:

1. اسم العمود ColumnName

2. نوع البيانات Data type

3. قيد عمود اختياري Optional Column Constraint

## 12.1.2 **ColumnName** اسم العمود

يجب أن يكون اسم العمود **ColumnName** فريداً في الجدول، وبعض الأمثلة على أسماء الأعمدة هي **LastName**، و **FirstName**.

## 12.1.3 **Data Type** نوع البيانات

يجب على نوع البيانات أن يكون نوع بيانات نظام أو نوع بيانات يعرّفه المستخدم، كما تملك العديد من أنواع البيانات حجماً، مثل: **CHAR(35)**، **Numeric(8,2)**، أو **Integer**.

- النوع **Bit**: بيانات أعداد صحيحة **Integer** لها قيمة 1 أو 0.
- النوع **Int**: بيانات أعداد صحيحة **Integer** لها القيم من  $-2^{31}$  إلى  $2^{31}-1$  أي  $2,147,483,648$ .
- النوع **Smallint**: بيانات أعداد صحيحة **Integer** لها القيم من  $-2^{15}$  إلى  $2^{15}-1$  أي  $32,768$ .
- النوع **Tinyint**: بيانات أعداد صحيحة **Integer** لها القيم من 0 حتى 255.
- النوع **Decimal**: بيانات ذات دقة ثابتة وقياس رقمي لها القيم من  $-10^{38}$  إلى  $10^{38}$ .
- النوع **Numeric**: مرادف لنوع **decimal**.
- النوع **Timestamp**: رقم فريد على مستوى قاعدة البيانات.
- النوع **Uniqueidentifier**: معرف فريد عالمياً أو **GUID** اختصاراً.
- النوع **Money**: تتراوح قيم البيانات النقدية من  $-922,337,203,685,477.5808$  إلى  $922,337,203,685,477.5807$  أي العدد 1 إلى  $2^{63}-1$  بدقّة تصل إلى واحد من عشرة آلاف من الوحدة النقدية.
- النوع **Smallmoney**: تتراوح قيم البيانات النقدية من  $-214,748.3647$  إلى  $214,748.3648$  بدقّة تصل إلى واحد من عشرة آلاف من الوحدة النقدية.
- النوع **Float**: بيانات أرقام ذات دقة عشرية تتراوح قيمها بين  $-1.79E+308$  و  $1.79E+308$ .
- النوع **Real**: بيانات أرقام ذات دقة عشرية قيمها تتراوح من  $-3.40E+38$  إلى  $3.40E+38$  حتى  $3.33$  ميلي ثانية.
- النوع **Datetime**: بيانات التاريخ والوقت تتراوح قيمها من تاريخ 1 يناير كانون الثاني 1753 إلى تاريخ 31 ديسمبر كانون الأول 9999 بدقّة تبلغ واحد إلى ثلاثة أجزاء من مئة من الثانية، أو 3.33 ميلي ثانية.

- النوع Smalldatetime: بيانات التاريخ والوقت تتراوح قيمها من تاريخ 1 يناير كانون الثاني 1900 حتى تاريخ 6 يونيو حزيران 2079 بدقة تبلغ دقة واحدة.
- النوع Char: بيانات ملحوظة ثابتة الطول وليس يوني코드 بطول أقصى 8000 حرف.
- النوع Varchar: بيانات متغيرة الطول وليس يوني코드 بحد أقصى 8000 حرف.
- النوع Text: بيانات متغيرة الطول وليس يوني코드 بطول أقصى يبلغ  $1 - 2^{31}$  أي 2,147,483,647 حرفاً.
- النوع Binary: بيانات ثنائية ذات طول ثابت بطول أقصى 8000 بايت.
- النوع Varbinary: بيانات ثنائية متغيرة الطول بطول أقصى يبلغ 8000 بايت.
- النوع Image: بيانات ثنائية متغيرة الطول بطول أقصى  $1 - 2^{31}$  أي 2,147,483,647 بايت.

#### 12.1.4 قيود العمود الاختيارية

قيود العمود الاختيارية هي NULL، و NOT NULL، و UNIQUE، و PRIMARY KEY، و DEFAULT، و تُستخدم لتهيئة قيمة لسجل جديد.

يشير قيد العمود NULL إلى أن القيمة الفارغة null مسموح بها، مما يعني أنه يمكن إنشاء صف بدون قيمة لهذا العمود، ويشير قيد العمود NOT NULL إلى وجوب توفير قيمة عند إنشاء صف جديد.

سيستخدم تعليمية لغة SQL للتوضيح والتي هي CREATE TABLE EMPLOYEES لإنشاء جدول موظفين يحتوي على 16 سمة أو حقل attributes .fields

```
USE SW
CREATE TABLE EMPLOYEES
(
    EmployeeNo           CHAR(10)      NOT NULL      UNIQUE,
    DepartmentName       CHAR(30)      NOT NULL      DEFAULT
    "Human Resources",
    FirstName            CHAR(25)      NOT NULL,
    LastName             CHAR(25)      NOT NULL,
    Category             CHAR(20)      NOT NULL,
    HourlyRate           CURRENCY     NOT NULL,
    TimeCard              LOGICAL     NOT NULL,
    HourlySalaried       CHAR(1)       NOT NULL,
    EmpType              CHAR(1)       NOT NULL,
    Terminated           LOGICAL     NOT NULL
)
```

```

ExemptCode          CHAR(2)          NOT NULL,
Supervisor          LOGICAL         NOT NULL,
SupervisorName      CHAR(50)         NOT NULL,
BirthDate           DATE            NOT NULL,
CollegeDegree       CHAR(5)          NOT NULL,
CONSTRAINT          Employee_PK PRIMARY KEY(EmployeeNo)
);

```

الحقل الأول هو EmployeeNo من النوع CHAR، ويبلغ طول هذا الحقل 10 محارف، ولا يمكن للمستخدم ترك هذا الحقل فارغاً NOT NULL، أما الحقل الثاني هو DepartmentName من النوع CHAR بطول 30، يستخدم قيد الجدول المعرف بواسطة الكلمة CONSTRAINT لإنشاء المفتاح الأساسي primary key (EmployeeNo)، وذلك بعد تعريف جميع أعمدة الجدول، أي كما يلي:

```
CONSTRAINT EmployeePK PRIMARY KEY(EmployeeNo)
```

يمكننا إنشاء جدول أقسام Department، وجدول مشاريع Project، وجدول مهام Assignment باستخدام SQL DDL، كما هو موضح في المثال التالي:

```

USE SW
CREATE TABLE DEPARTMENT
(
  DepartmentName Char(35) NOT NULL,
  BudgetCode     Char(30) NOT NULL,
  OfficeNumber   Char(15) NOT NULL,
  Phone          Char(15) NOT NULL,
  CONSTRAINT DEPARTMENT_PK PRIMARY KEY(DepartmentName)
);

```

أُنشئ جدول المشاريع project التالي بسبعة حقول هي: معرف المشروع ProjectID، واسم المشروع ProjectName، والقسم Department، والحد الأقصى للساعات MaxHours، وتاريخ البدء StartDate، وتاريخ الانتهاء EndDate.

```

USE SW
CREATE TABLE PROJECT
(
  ProjectID      Int  NOT NULL IDENTITY (1000,100),
  ProjectName    Char(50) NOT NULL,

```

```

Department      Char(35) NOT NULL,
MaxHours       Numeric(8,2) NOT NULL DEFAULT 100,
StartDate       DateTime NULL,
EndDate         DateTime NULL,
CONSTRAINT      ASSIGNMENT_PK PRIMARY KEY(ProjectID)
);

```

بينما أنشئ جدول المهام assignment بثلاثة حقول، هي: معرف المشروع ProjectID، ورقم الموظف EmployeeNumber، وساعات العمل HoursWorked.

يُستخدم جدول المهام لتسجيل الموظف باستخدام الحقل EmployeeNumber، ومقدار الوقت باستخدام الحقل HoursWorked الذي عمل فيه الموظف في مشروع معين باستخدام الحقل ProjectID، أي كما يلي:

```

USE SW
CREATE TABLE ASSIGNMENT
(
ProjectID      Int NOT NULL,
EmployeeNumber Int NOT NULL,
HoursWorked    Numeric(6,2) NULL,
);

```

## 12.2 قيود الجدول

تُعرف قيود الجدول بواسطة الكلمة المفتاحية CONSTRAINT ويمكن استخدامها لتطبيق العديد من القيود الموضحة أدناه.

### 12.2.1 القيد IDENTITY

يمكننا استخدام قيد العمود اختياري IDENTITY لتوفير قيمة فريدة تزايدية لهذا العمود، إذ تُستخدم أعمدة الهوية Identity مع قيود المفتاح الرئيسي KEY PRIMARY لتكون بمثابة معرف صف فريد للجدول، كما يمكن إسناد الخاصية IDENTITY إلى عمود له نوع بيانات tinyint، أو smallint، أو int، أو decimal، أو numeric، وهذا القيد:

- يُولد أرقاماً متسلسلةً.
- لا يفرض سلامة الكيان entity integrity.
- يمكن أن يحتوي عمود واحد فقط على الخاصية IDENTITY.
- يجب تعريفه على أساس نوع بيانات integer، numeric، أو decimal.

- لا يمكن تحدیث عمود له الخاصية IDENTITY.

- لا يمكن أن يحتوي على قيم فارغة NULL.

- لا يمكنه ربط الافتراضات والقيود الافتراضية بالعمود.

بالنسبة للقيد IDENTITY[(seed, increment)]:

- Seed: هي القيمة الأولية لعمود الهوية identity.

- Increment: هي القيمة المطلوب إضافتها إلى عمود الزيادة increment الأخير.

سنستخدم مثال قاعدة بيانات آخر لتوضیح عبارات لغة SQL DDL بصورة أكبر من خلال إنشاء الجدول Hotel في قاعدة بيانات الفندق HOTEL كما يلي:

```
CREATE TABLE  tblHotel
(
    HotelNo      Int          IDENTITY (1,1),
    Name          Char(50)     NOT NULL,
    Address       Char(50)     NULL,
    City          Char(25)     NULL,
)
```

## 12.2.2 القيد UNIQUE

يمنع القيد UNIQUE من إدخال قيم مكررة في عمود، حيث:

- يُستخدم القيدان PK، و UNIQUE لفرض سلامة الكيان.

- يمكن تعريف قيود UNIQUE متعددة للجدول.

- يجري دائمًا التحقق من صحة البيانات الموجودة عند إضافة قيد UNIQUE إلى جدول موجود.

- يمكن وضع القيد UNIQUE على الأعمدة التي تقبل القيم الفارغة، حيث يمكن أن يكون صُف واحد فقط NULL.

- ينشئ القيد UNIQUE دليلاً فريداً للعمود المختار تلقائياً.

الصيغة التالية هي الصيغة العامة للقيد UNIQUE:

```
[CONSTRAINT constraint_name]
UNIQUE [CLUSTERED | NONCLUSTERED]
(col_name [, col_name2 [.., col_name16]])
```

[ON segment\_name]

يستخدم المثال التالي القيد UNIQUE كما يلي:

```
CREATE TABLE EMPLOYEES
(
    EmployeeNo      CHAR(10)      NOT NULL      UNIQUE,
```

### 12.2.3 القيد FOREIGN KEY المفتاح الخارجي

يعرف القيد FOREIGN KEY -أو FK اختصاراً- عموماً، أو مجموعة من الأعمدة التي تتطابق قيمها مع المفتاح الرئيسي PRIMARY KEY -أو PK اختصاراً- لجدول آخر، بحيث:

- تحدث القيم في المفتاح الخارجي FK تلقائياً عند تحديث أو تغيير قيم المفتاح الرئيسي PK في الجدول المرتبط.
  - يجب أن تشير قيود المفتاح الخارجي FK إلى القيد المفتاح الرئيسي PK، أو القيد UNIQUE لجدول آخر.
  - يكون عدد أعمدة المفتاح الخارجي FK هو نفسه قيد المفتاح الرئيسي PK، أو قيد UNIQUE.
  - إذاً استخدم الخيار WITH NOCHECK، فلن يتحقق قيد المفتاح الخارجي FK من صحة البيانات الموجودة في الجدول.
  - لا يوجد دليل index للأعمدة التي تشارك في قيد المفتاح الخارجي FK.
- الصيغة التالية هي الصيغة العامة لقيد المفتاح الخارجي :FOREIGN KEY

```
[CONSTRAINT constraint_name]
[FOREIGN KEY (col_name [, col_name2 [..., col_name16]])]
[REFERENCES [owner].ref_table [(ref_col [, ref_col2 [..., ref_col16]])]]
```

يكون الحقل HotelNo في المثال التالي في الجدول tblRoom مفتاحاً خارجياً FK للحقل HotelNo في الجدول tblHotel الموضح سابقاً:

```
USE HOTEL
GO
CREATE TABLE  tblRoom
(
    HotelNo  Int      NOT NULL ,
    RoomNo   Int      NOT NULL ,
```

```

Type      Char(50)      NULL ,
Price     Money         NULL ,
PRIMARY KEY (HotelNo, RoomNo),
FOREIGN KEY (HotelNo) REFERENCES tblHotel
)

```

## 12.2.4 القيد CHECK

يقيّد القيد CHECK القيم التي يمكن إدخالها في جدول، بحيث:

- يمكن أن يحتوي على شروط بحث مشابهة لعبارة WHERE.
- يمكنه الربط بين الأعمدة في نفس الجدول.
- يجب العمل على تقييم قاعدة التحقق من صحة البيانات للقييد CHECK من خلال تعبير boolean expression بوليانى.
- يمكن تعريفه لعمود له قاعدة مرتبطة به.

الصيغة التالية هي الصيغة العامة للقييد CHECK:

```

[CONSTRAINT constraint_name]
CHECK [NOT FOR REPLICATION] (expression)

```

يقتصر حقل النوع Type في المثال التالي على الأنواع: Single أو Double أو Suite أو Executive.

```

USE HOTEL
GO
CREATE TABLE  tblRoom
(
    HotelNo  Int          NOT NULL ,
    RoomNo   Int          NOT NULL ,
    Type     Char(50)     NULL ,
    Price    Money        NULL ,
    PRIMARY KEY (HotelNo, RoomNo),
    FOREIGN KEY (HotelNo) REFERENCES tblHotel
    CONSTRAINT Valid_Type
    CHECK (Type IN ('Single', 'Double', 'Suite', 'Executive'))
)

```

يجب في المثال التالي أن يكون تاريخ تعيين الموظف قبل 2004, 1, January, أو يجب أن يكون الحد الأقصى للراتب 300 ألف دولار:

```
GO
CREATE TABLE SALESREPS
(
    Empl_num Int Not Null
    CHECK (Empl_num BETWEEN 101 and 199),
    Name      Char (15),
    Age       Int    CHECK (Age >= 21),
    Quota     Money  CHECK (Quota >= 0.0),
    HireDate  DateTime,
    CONSTRAINT QuotaCap CHECK ((HireDate < "01-01-2004") OR (Quota
    <=300000))
)
```

## 12.2.5 القيد DEFAULT

يُستخدم القيد DEFAULT لتوفير قيمة تُضاف تلقائياً لعمود ما إذا لم يوفرها المستخدم، بحيث:

- يمكن احتواء العمود على قيد DEFAULT واحد فقط.
- لا يمكن استخدام القيد DEFAULT في الأعمدة التي لها نوع البيانات timestamp، أو التي لها الخاصية identity.
- ترتبط القيود DEFAULT تلقائياً بعمود عند إنشائها.

الصيغة العامة للقيد DEFAULT هي:

```
[CONSTRAINT constraint_name]
DEFAULT {constant_expression | niladic-function | NULL}
[FOR col_name]
```

يُضبط المثال التالي القيمة الافتراضية default لحقل city field على القيمة "Vancouver":

```
USE HOTEL
ALTER TABLE tblHotel
Add CONSTRAINT df_city DEFAULT 'Vancouver' FOR City
```

## 12.3 الأنواع التي يُعرفها المستخدم

تعتمد الأنواع التي يُعرفها المستخدم دائمًا على نوع البيانات التي يوفرها النظام، فيمكن لهذه الأنواع فرض سلامة البيانات والسماح بالقيم الفارغة null. اختر الأنواع التي تكون تحت الكلمة "Programmability" في قاعدة البيانات الخاصة بك، لإنشاء نوع بيانات يُعرفه المستخدم في خادم SQL Server، ثم انقر بزر الفأرة الأيمن واختر المسار 'New'، أو نفذ إجراء النظام sp\_addtype data type المخزن system stored procedure، ثم اكتب ما يلي:

```
sp_addtype ssn, 'varchar(11)', 'NOT NULL'
```

سيؤدي هذا إلى إضافة نوع بيانات جديد عُرّفه المستخدم يسمى SIN بتسعة محارف.

سيستخدم الحقل EmployeeSIN نوع البيانات SIN الذي عُرّفه المستخدم في المثال التالي:

```
CREATE TABLE SINTable
(
    EmployeeID      INT Primary Key,
    EmployeeSIN     SIN,
    CONSTRAINT CheckSIN
    CHECK (EmployeeSIN LIKE
        ' [0-9][0-9][0-9] - [0-9][0-9] [0-9] - [0-9][0-9][0-9] ')
)
```

### 12.3.1 التعليةمة ALTER TABLE

يمكن استخدام تعليمات ALTER TABLE لإضافة وحذف القيود، بحيث:

- تسمح تعليمية ALTER TABLE بإزالة الأعمدة.

- يتحقق من جميع البيانات الموجودة عند إضافة قيد للتأكد من عدم وجود انتهاكات.

نستخدم في المثال تعليمية ALTER TABLE للخاصية IDENTITY في الحقل :ColumnName

```
USE HOTEL
GO
ALTER TABLE  tblHotel
ADD CONSTRAINT unqName UNIQUE (Name)
```

استخدم تعليمية ALTER TABLE لإضافة عمود مع الخاصية IDENTITY.

ADD

ColumnName int IDENTITY(seed, increment)

## 12.3.2 التعليةمme DROP TABLE

تزيل التعليةمme DROP TABLE جدوًّا من قاعدة البيانات، لذلك يجب عليك التأكد من تحديد قاعدة البيانات الصحيحة.

DROP TABLE tblHotel

سيؤدي تنفيذ عبارة DROP TABLE SQL إلى إزالة الجدول tblHotel من قاعدة البيانات.

## 12.4 مصطلحات أساسية

- **DDL**: اختصار لغة تعريف البيانات .data definition language
- **DML**: اختصار لغة معالجة البيانات .data manipulation language
- **SEQUEL**: اختصار لغة الاستعلامات الهيكلية الإنجليزية Structured English Query Language التي صُمِّمت لمعالجة واسترداد البيانات المخزَّنة في نظام شبه علائقى لإدارة قواعد البيانات، وخاص بشركة IBM، ويسمى نظام R.
- **لغة الاستعلامات الهيكلية أو SQL**: لغة قاعدة بيانات مصممة لإدارة البيانات الموجودة في نظام إدارة قواعد البيانات العلائقية.

## 12.5 تمارين

1. باستخدام المعلومات الخاصة بالتمرين الموجود في الفصل قواعد السلامة والقيود المطبقة عند تصميم قواعد البيانات، طبق التخطيط بلغة Transact SQL أي اعرض تعليمات SQL لكل جدول وطبق القيود.
2. أنشئ الجدول Employee الموضح أدناه في خادم SQL Server، واعرض التعليمات التي استخدمتها.

النوع	اسم الحقل (العمود)
CHAR(3)	EMP_NUM
VARCHAR(15)	EMP_LNAME
VARCHAR(15)	EMP_FNAME
CHAR(1)	EMP_INITIAL
DATE	EMP_HIREDATE
CHAR(3)	JOB_CODE

3. اكتب شيفرة لغة SQL لإدخال صفوف الجدول السابق، بعد إنشاء بنيته.

استخدم الجدول السابق للإجابة على الأسئلة من 4 إلى 10.

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arrough	June	E	01-Dec-96	500
104	Ramoras	Anne	K	15-Nov-87	501
105	Johnson	Alice	K	01-Feb-93	502
106	Smithfield	William		22-Jun-04	500
107	Alonzo	Maria	D	10-Oct-93	500
108	Washington	Ralph	B	22-Aug-91	501
109	Smith	Larry	W	18-Jul-97	501

4. اكتب شيفرة لغة SQL لتغيير رمز الوظيفة job code إلى 501 للموظف الذي رقمه 107، وافحص النتائج بعد الانتهاء من المهمة، ثم أعد ضبط رمز الوظيفة إلى قيمته الأصلية.

5. اكتب شيفرة لغة SQL لإعطاء قائمة بجميع السمات الخاصة برمز الوظيفة 502، بافتراض إدخال البيانات الموضحة في جدول الموظف Employee.

6. اكتب شيفرة لغة SQL لحذف الصف الخاص بالشخص الذي اسمه "William Smithfield" ، والذي وُظف في 22 June 2004، والذي تصنيف رمز وظيفته هو 500.

استخدم المعاملات المنطقية لتضمين جميع المعلومات الواردة في هذه المسألة

7. أضف السمتين PROJ\_NUM، وEMP\_PCT إلى جدول الموظف، بحيث تكون السمة EMP\_PCT هي نسبة المكافأة المدفوعة لكل موظف.

8. اكتب شيفرة لغة SQL باستخدام أمر واحد لإدخال رقم المشروع PROJ\_NUM = 18 لجميع الموظفين الذين تصنيف الوظيفة JOB\_CODE الخاص بهم هو 500.

9. اكتب شيفرة لغة SQL باستخدام أمر واحد لإدخال رقم المشروع PROJ\_NUM = 25 لجميع الموظفين الذين تصنيف الوظيفة JOB\_CODE الخاص بهم يساوي 502 أو أعلى.

10. اكتب شيفرة لغة SQL لتغيير رقم المشروع PROJ\_NUM إلى 14 للموظفين الذين تعينوا قبل 1994, 1، ورمز الوظيفة الخاصة بهم يساوي 501 على الأقل. (قد تفترض أن الجدول سيعاد إلى حالته الأصلية التي سبقت هذا السؤال).

# 13. لغة معالجة البيانات DML الخاصة بلغة SQL

تُستخدم لغة معالجة البيانات SQL أو Data Manipulation Language اختصاراً- الخاصة بلغة SQL للاستعلام عن البيانات في قاعدة البيانات وتعديلها، وسنشرح في هذا الفصل كيفية استخدام تعليمات أوامر لغة SQL DML والتي هي SELECT و INSERT و UPDATE و DELETE المعروفة كما يلي:

•: للاستعلام عن بيانات في قاعدة البيانات.

•: لإدخال بيانات في جدول.

•: لتحديث بيانات في جدول.

•: لحذف بيانات من جدول.

في تعليمات SQL DML:

• يجب بدأ كل شرط في عبارة بسطر جديد.

• يجب انتظام بداية كل شرط مع بداية الشروط الأخرى.

• إذا تألف شرط من عدة أجزاء، فيجب توضع هذه الأجزاء على سطور منفصلة، كما يجب إضافة مسافة بادئة لها تحت بداية الشرط لإظهار العلاقة.

• تُستخدم الأحرف الكبيرة لتمثيل الكلمات الممحوزة.

• تُستخدم الحروف الصغيرة لتمثيل الكلمات التي يُعرفها المستخدم.

## 13.1 تعلیمة SELECT

تسمح التعليمة أو الأمر SELECT للمستخدم باستخراج البيانات من الجداول، بناءً على معايير محددة، حيث تُعالج وفقاً للتسلسل التالي:

- اختيار عنصر أو مجموعة عناصر SELECT DISTINCT.
- FROM من جدول أو مجموعة جداول.
- WHERE يليها تعبير شرطي.
- GROUP BY يليها حقل أو مجموعة حقول.
- ORDER BY يليها مجموعة حقول.

يمكن استخدام تعليمة SELECT لإنشاء قائمة بهواتف الموظفين من جدول الموظفين Employees، انظر:

```
SELECT FirstName, LastName, phone
FROM Employees
ORDER BY LastName
```

سيعرض هذا الإجراء اسم عائلة last name، first name، واسمي الأول first name، ورقم هاتفه phone number من جدول الموظفين Employees كما في الجدول التالي:

Last Name	First Name	Phone Number
Hagans	Jim	604-232-3232
Wong	Bruce	604-244-2322

ستستخدم في المثال التالي جدول الناشرين Publishers table الذي يمثله الجدول الآتي، حيث ستلاحظ أنّ كندا Canada مكتوبة بطريقة خاطئة في حقل بلد الناشر Publisher Country المقابل لحقل اسم الناشر "ABC Publishing" ، ومدينة الناشر "Example Publishing" .

Publisher Name	Publisher City	Publisher Province	Publisher Country
Acme Publishing	Vancouver	BC	Canada
Example Publishing	Edmonton	AB	Cnada
ABC Publishing	Toronto	ON	Canda

استخدم تعليمة UPDATE لتصحيح الأخطاء وتوحيد حقل البلد ليصبح Canada، كما سنتكلم لاحقاً عن تعليمة UPDATE في هذا الفصل.

إذا أضفت اسم الناشر Publisher Name، مدينة الناشر Publisher City، فستستخدم تعليةة SELECT ويتبعها اسم الحقول التي يفصل بينها بفاصلة أجنبية comma، أي كما يلي:

```
SELECT PubName, city
FROM Publishers
```

سيؤدي هذا الإجراء إلى عرض اسم الناشر ومدينته من جدول الناشرين.

إذا أردت عرض حقل الناشر باسم حقل المدينة -أي تبديل اسم الحقل PubName ليصبح city-، فاستخدم تعليةة SELECT مع عدم وضع فاصلة أجنبية بين Pub\_Name و city، أي كما يلي:

```
SELECT PubName city
FROM Publishers
```

سيعرض تنفيذ هذا الإجراء فقط الحقل PUB\_NAME من جدول الناشرين، بحيث يكون له العنوان "city".  
سيفترض SQL Server أنك تريدين وضع اسم عمود جديد للحقل PUB\_NAME إذا لم تضمن الفاصلة الأجنبية.

### 13.1.1 تعليةة SELECT مع معيار WHERE

قد ترغب أحياناً في التركيز على جزء من جدول الناشرين، مثل الناشرين الموجودين في مدينة فانكوفر فقط، إذ ستستخدم في هذه الحالة عبارة SELECT مع معيار WHERE، أي كما يلي:

```
WHERE city = 'Vancouver'
```

يوضح المثالان الأوليان التاليان كيفية تحديد اختيار سجل مع المعيار WHERE باستخدام BETWEEN، إذ يعطي كل من هذين المثالين نتائج تخزين العناصر نفسها التي عددها بين 20 و 50 عنصر في المخزن.

يستخدم المثال رقم 1 الكمية التي قيمتها بين 20 و 50 عنصر مع تضمين العنصرين 20 و 50 بالصورة التالية: qty BETWEEN 20 and 50

```
SELECT StorID, qty, TitleID
FROM Sales
WHERE qty BETWEEN 20 and 50 -- ضمن العنصرين رقم 20 و 50
```

يستخدم المثال رقم 2 الشرط qty >=20 and qty <=50

```
SELECT StorID, qty, TitleID
FROM Sales
WHERE qty >= 20 and qty <= 50
```

يوضح المثال رقم 3 كيفية تحديد اختيار سجل مع المعيار WHERE باستخدام NOT BETWEEN

```
SELECT StorID, qty, TitleID
FROM Sales
WHERE qty NOT BETWEEN 20 and 50
```

يظهر المثالان التاليان طريقتين مختلفتين لتحديد اختيار سجل مع المعيار WHERE باستخدام IN مع النتائج نفسها.

يوضح المثال رقم 4 كيفية اختيار السجلات باستخدام حقل المقاطعة province من جدول Publishers على أساس جزء من تعليمة WHERE.

```
SELECT *
FROM Publishers
WHERE province = 'BC' OR province = 'AB' OR province = 'ON'
```

يوضح المثال رقم 5 كيفية اختيار السجلات باستخدام المقاطعة province مع IN على أساس جزء من تعليمة WHERE:

```
SELECT *
FROM Publishers
WHERE province IN ('BC', 'AB', 'ON')
```

يوضح المثالان الآخرين كيف يمكن استخدام NOT NULL لتحديد السجلات، ولكن سنستخدم في هذين المثالين جدول الكتب Books table الغير موضح هنا، والذي يحتوي على حقول، وهي: العنوان Title، والكمية Quantity، وسعر الكتاب Price، وكل ناشر لديه جدول كتب يعطي قائمةً بجميع كتب الناشر.

يستخدم المثال رقم 6 القيمة NULL:

```
SELECT price, title
FROM Books
WHERE price IS NULL
```

يستخدم المثال رقم 7 القيمة NOT NULL:

```
SELECT price, title
FROM Books
WHERE price IS NOT NULL
```

## 13.1.2 استخدام محارف البدل **wildcards** في شرط **LIKE**

يحدد الشرط **LIKE** الصنوف التي تحتوي على الحقول التي تطابق أجزاءً محددة من سلاسل محرفية، كما يُستخدم الشرط **LIKE** مع البيانات التي هي من النوع **char**, **text**, **varchar**, **datetime**, **smalldatetime**. يسمح محرف البدل **wildcard** للمستخدم ب Matching الحقول التي تحتوي على محارف معينة، حيث سيعطي محرف البدل **'N%** province = **'N%** يوضح الجدول أربعة طرائق لتحديد محارف البدل في تعليمات **SELECT** في صيغة التعبير المنتظم:

محرف البدل	wildcard
يمثل أي سلسلة تتتألف من صفر أو أكثر من المحارف	<b>%</b>
يمثل أي محرف واحد	<b>-</b>
يمثل أي محرف واحد ضمن مجال محدد مثل المجال <b>[a-f]</b> أو مجموعة محددة مثل المجموعة <b>[abcdef]</b>	<b>[ ]</b>
يمثل أي محرف واحد ليس ضمن مجال محدد مثل المجال <b>[^a-f]</b> أو مجموعة محددة مثل المجموعة <b>[^abcdef]</b>	<b>[^ ]</b>

تبحث التعليمية **'Mc%'** في المثال رقم 1 عن جميع أسماء العائلة **last names** التي تبدأ بالمحرفين **"Mc"** مثل **McBadden**

```
SELECT LastName
  FROM Employees
 WHERE LastName LIKE 'Mc%'
```

تبحث التعليمية **'%inger'** في المثال رقم 2 عن جميع أسماء العائلة التي تنتهي بالمحارف **"inger"**، مثل: **Stringer**, **Ringer**, **McBadden**

```
SELECT LastName
  FROM Employees
 WHERE LastName LIKE '%inger'
```

تبحث التعليمية **'%en%'** عن جميع أسماء العائلة التي تحتوي على المحرفين **"en"**، مثل: **McBadden**, **Green**, **Bennett**

```
SELECT LastName
  FROM Employees
 WHERE LastName LIKE '%en%'
```

### 13.1.3 تعليمات ORDER BY مع الشرط SELECT

يُستخدم الشرط ORDER BY لترتيب السجلات في القائمة الناتجة، ويمكنك استخدام ASC لترتيب النتائج تصاعديًا، و DESC لترتيب النتائج تناظريًا.

يستخدم المثال التالي:

```
SELECT *
FROM Employees
ORDER BY HireDate ASC
```

يستخدم المثال التالي:

```
SELECT *
FROM Books
ORDER BY type, price DESC
```

### 13.1.4 تعليمات GROUP BY مع الشرط SELECT

يُستخدم الشرط GROUP BY لإنشاء خرج هو عبارة عن صف واحد لكل مجموعة، وينتج قيمًا موجزةً للأعمدة المحددة، كما هو موضح أدناه:

```
SELECT type
FROM Books
GROUP BY type
```

يستخدم المثال التالي التعليمية السابقة:

```
SELECT type AS 'Type', MIN(price) AS 'Minimum Price'
FROM Books
WHERE royalty > 10
GROUP BY type
```

إذا تضمنت تعليمات SELECT معيار WHERE ليكون السعر price قيمةً غير فارغة not null كما يلي:

```
SELECT type, price
FROM Books
WHERE price is not null
```

فستكون التعليمية التي تحتوي على شرط GROUP BY كما يلي:

```
SELECT type AS 'Type', MIN(price) AS 'Minimum Price'
FROM Books
WHERE price is not null
GROUP BY type
```

### 13.1.5 استخدام GROUP BY مع COUNT

يمكننا استخدام COUNT لإحصاء عدد العناصر الموجودة في حاوية container، ولكن إذا أردت حساب عدد عناصر مختلفة في مجموعات منفصلة مثل رخام ذي ألوان مختلفة، فسنستخدم دالة COUNT مع GROUP BY.

توضح تعليمية SELECT أدناه كيفية حساب عدد مجموعات من البيانات باستخدام دالة COUNT مع الشرط GROUP BY.

```
SELECT COUNT(*)
FROM Books
GROUP BY type
```

### 13.1.6 استخدام GROUP BY مع SUMg AVG

يمكننا استخدام دالة AVG لتعطينا متوسط أي مجموعة، وتُستخدم الدالة SUM لاعطاء المجموع.

يستخدم المثال رقم 1 التالي دالة AVG مع الشرط GROUP BY type.

```
SELECT AVG(qty)
FROM Books
GROUP BY type
```

يستخدم المثال رقم 2 التالي دالة SUM مع الشرط GROUP BY type.

```
SELECT SUM(qty)
FROM Books
GROUP BY type
```

يستخدم المثال رقم 3 كلاً من الدالتين AVG، وSUM مع الشرط GROUP BY type في تعليمية SELECT.

```
SELECT 'Total Sales' = SUM(qty), 'Average Sales' = AVG(qty), stor_id
FROM Sales
GROUP BY StorID ORDER BY 'Total Sales'
```

### 13.1.7 تقييد الصنوف مع HAVING

يمكن استخدام الشرط HAVING لتقييد الصنوف، فهو يشبه شرط WHERE باستثناء أنه يتضمن دالة تجميع aggregate function مثل WHERE: إذ لا يستطيع الشرط HAVING فعلى ذلك، أي يتصرف الشرط HAVING مثل الشرط WHERE، ولكنه قابل للتطبيق على المجموعات.

نستخدم في هذا المثال الشرط HAVING لاستبعاد المجموعات التي مقاطعتها 'BC'.

```
SELECT au_fname AS 'Author"s First Name', province AS 'Province'
FROM Authors
GROUP BY au_fname, province
HAVING province <> 'BC'
```

## 13.2 تعلية INSERT

تضيف تعلية INSERT صفوأً إلى جدول، وأيضاً ما يلي:

- تحدد تعلية INSERT الجدول أو العرض view التي ستدخل البيانات فيه.
- تعرض Column List قائمةً بالأعمدة التي ستتأثر بتعلية INSERT.
- يجب توفير كل قيمة إذا حُذف عمود.
- يمكن وضع الأعمدة في قائمة ضمن أي ترتيب إذا ضممتها.
- تحدد الكلمة VALUES البيانات التي تريد إدخالها في الجدول، وتكون VALUES إلزامية.
- يجب عدم إدراج الأعمدة ذات الخاصية IDENTITY بصورة صريحة في column\_list أو .values\_clause.

صيغة تعلية INSERT هي:

```
INSERT [INTO] Table_name | view_name [column_list]
DEFAULT VALUES | values_list | select statement
```

تُطبق القواعد التالية عند إدخال صنوف باستخدام تعلية INSERT:

- يؤدي إدخال سلسلة فارغة ('') في عمود من النوع varchar، أو text إلى إدخال مسافة واحدة.
- تحسن جميع الأعمدة ذات النوع char على اليمين right-padded لتصل إلى الطول المحدد.
- تُزال جميع المسافات الزائدة من البيانات المدرجة في أعمدة من النوع varchar، باستثناء السلسلة التي تحتوي على مسافات فقط، إذ تُختصر هذه السلسلة إلى مسافة واحدة فقط.

- إذا أخللت تعليمية `INSERT` بالقييد، أو الافتراض، أو القاعدة، أو إذا كان نوع البيانات خاطئاً، فستفشل هذه التعليمية، وسيعرض خادم SQL Server رسالة خطأ.

يمكن حدوث أحد الأشياء التالية للأعمدة التي لا تحتوي على قيم عند تحديد قيم بعض الأعمدة في `column_list` فقط:

1. تدخل قيمة افتراضية إذا كان للعمود قيد `DEFAULT`، أو إذا كان الافتراض مرتبط بالعمود، أو إذا كان الافتراض مرتبط بنوع البيانات التي يعُرّفها المستخدم.
2. تدخل القيمة الفارغة `NULL` فقط إذا سمح العمود بالقيم الفارغة، ولا توجد قيمة افتراضية موجودة للعمود.
3. سوف تُعرض رسالة خطأ ويرفض الصف إذا عُرف العمود بأنه غير فارغ `NOT NULL`، ولا توجد قيمة افتراضية.

يستخدم المثال التالي تعليمية `INSERT` لإضافة سجل إلى جدول الكتب `Authors`:

```
INSERT INTO Authors
VALUES('555-093-467', 'Martin', 'April', '281 555-5673', '816 Market
St.,', 'Vancouver', 'BC', 'V7G3P4', 0)
```

يوضح المثال التالي كيفية إدخال صفات جزئي `row` في جدول الناشرين `Publishers` مع قائمة أعمدة.

يملك عمود الدولة `country` قيمة افتراضية هي `Canada`، لذلك لا يلزمك تضمينه في قيمك.

```
INSERT INTO Publishers (PubID, PubName, city, province)
VALUES ('9900', 'Acme Publishing', 'Vancouver', 'BC')
```

اتبع المثال التالي لإدخال صفوف في جدول مع عمود `IDENTITY`، ولا تعطي قيمةً للعمود `IDENTITY`، ولا قيمةً لاسم العمود ضمن قائمة الأعمدة.

```
INSERT INTO jobs
VALUES ('DBA', 100, 175)
```

### 13.2.1 إدخال قيم محددة ضمن عمود `IDENTITY`

لا يمكن إدخال البيانات مباشرة في عمود `IDENTITY` افتراضياً، ولكن إذا حُذف صف خطأً، أو إذا كانت هناك ثغرات في قيم عمود `IDENTITY`، فيمكنك إدخال صفات وتحديد قيمة العمود `IDENTITY`.

```
IDENTITY_INSERT option
```

يمكن استخدام `IDENTITY_INSERT` على النحو التالي للسماح بإدخال قيمة هوية identity محددة:

```
SET IDENTITY_INSERT jobs ON
INSERT INTO jobs (job_id, job_desc, min_lvl, max_lvl)
VALUES (19, 'DBA2', 100, 175)
SET IDENTITY_INSERT jobs OFF
```

### 13.2.2 إدخال صفوف باستخدام عبارة SELECT

يمكننا إنشاء جدول مؤقت صغير من جدول كبير، لذلك يمكننا إدخال صفوف مع تعليمة `SELECT` لا يوجد تحقق لصحة التفرد `uniqueness` عند استخدام هذا الأمر، وبالتالي، قد يكون هناك العديد من الصفوف بالمعرف `pub_id` نفسه في المثال التالي.

ينشئ هذا المثال جدول ناشرين `Publishers` مؤقت هو `tmpPublishers` أصغر باستخدام تعليمة إنشاء جدول `CREATE TABLE`، ثم تُستخدم تعليمة `INSERT` مع تعليمة `SELECT` لإضافة سجلات إلى جدول الناشرين المؤقت من جدول الناشرين `Publishers`.

```
CREATE TABLE dbo.tmpPublishers (
    PubID char (4) NOT NULL ,
    PubName varchar (40) NULL ,
    city varchar (20) NULL ,
    province char (2) NULL ,
    country varchar (30) NULL DEFAULT ('Canada')
)
INSERT tmpPublishers
SELECT * FROM Publishers
```

نسخ في هذا المثال مجموعةً فرعيةً من البيانات:

```
INSERT tmpPublishers (pub_id, pub_name)
SELECT PubID, PubName
FROM Publishers
```

تُنسخ بيانات الناشرين هنا إلى جدول `tmpPublishers` ويُضيّط عمود الدولة `country` إلى القيمة `Canada`:

```
INSERT tmpPublishers (PubID, PubName, city, province, country)
SELECT PubID, PubName, city, province, 'Canada'
FROM Publishers
```

## 13.3 تعلیمة UPDATE

تغییر تعلیمة UPDATE البيانات في الصفوف الموجودة إما بإضافة بيانات جديدة أو بتعديل البيانات الموجودة سابقاً.

يستخدم المثال التالي تعلیمة UPDATE لتوحید حقل الدولة country ليكون Canada لجميع السجلات في جدول Publishers:

```
UPDATE Publishers
SET country = 'Canada'
```

يزيد المثال التالي مبالغ حقوق المؤلف royalty التي قيمتها بين 10 و20 بنسبة 10%:

```
UPDATE roysched
SET royalty = royalty + (royalty * .10)
WHERE royalty BETWEEN 10 and 20
```

### 13.3.1 UPDATE ضمن عبارات فرعية subqueries

يُمنح الموظفون في جدول الموظفين Employees الذين وظفهم الناشر في عام 2010 ترقيةً إلى أعلى مستوى وظيفي حسب نوع عملهم كما يلي:

```
UPDATE Employees
SET job_lvl =
(SELECT max_lvl FROM jobs
WHERE employee.job_id = jobs.job_id)
WHERE DATEPART(year, employee.hire_date) = 2010
```

## 13.4 تعلیمة DELETE

تزيل تعلیمة DELETE صفوياً من مجموعة سجلات، كما تحدّد عبارات DELETE الجدول أو العرض view الذي يحوي الصفوف التي ستحذف، ويمكن إدراج جدول أو صف واحد فقط في الوقت نفسه.

يُعدّ الشرط WHERE المعيار الذي يحدّد السجلات المراد حذفها، وتكون صيغة تعلیمة DELETE كما يلي:

```
DELETE [FROM] { table_name | view_name }
[WHERE clause]
```

قواعد تعليمية `DELETE` هي:

- إذا حُذف شرط `WHERE` فستُزال جميع الصفوف الموجودة في الجدول باستثناء الفهارس `indexes`, والجدول، والقيود.
- لا يمكن استخدام عبارة `DELETE` بعرض `view` يحتوي على شرط `FROM` يسمّي أكثر من جدول واحد، فتتعلّمه `DELETE` يمكن أن تؤثّر على جدول أساسٍ فقط في الوقت نفسه.

فيما يلي ثلات تعليمات `DELETE` مختلفة يمكن استخدامها:

- حذف جميع الصفوف من جدول:

```
DELETE
FROM Discounts
```

- حذف صفات محدّدة:

```
DELETE
FROM Sales
WHERE stor_id = '6380'
```

- حذف صفات بناءً على قيمة ضمن استعلام فرعي:

```
DELETE FROM Sales
WHERE title_id IN
(SELECT title_id FROM Books WHERE type = 'mod_cook')
```

## 13.5 الدوال المبنية مسبقا

يوجد العديد من الدوال المبنية مسبقاً في SQL Server، مثل:

- دوال التجميع `Aggregate`: ترجع قيماً موجزة `.summary values`.
- دوال التحويل `Conversion`: تحول نوع بيانات معين إلى نوع آخر.
- دوال التاريخ `Date`: تعرض معلومات عن التواریخ والأوقات.
- الدوال الرياضية `Mathematical`: تجري عمليات على البيانات العددية.
- الدوال المتعلقة بالسلسل `String`: تجري عمليات على سلاسل المحارف، أو على البيانات الثنائية، أو على التعابير.
- الدوال المتعلقة بالنظام `System`: ترجع معلومات من قاعدة البيانات.

7. الدوال المتعلقة بالنصوص Text، والصور image: تجري عمليات على بيانات نصية، أو على بيانات الصور.

سنشرح أدناه الدوال الأربع الأولى شرحاً مفصلاً مع أمثلة عنها.

### 13.5.1 دوال التجميع Aggregate functions

تجري دوال التجميع حسابات على مجموعة من القيم، وترجع قيمةً واحدةً أو قيمةً موجزةً.

يعرض الجدول التالي هذه الدوال:

وصفها	الدالة FUNCTION
ترجع متوسط average جميع القيم، أو القيم المميزة فقط، ضمن التعبير DISTINCT	AVG
ترجع عدد القيم غير الفارغة في التعبير، وإذا استخدم التمايز DISTINCT فستجد الدالة COUNT عدد القيم غير الفارغة الفريدة	COUNT
ترجع عدد الصنوف، ولا تأخذ الدالة COUNT(*) عاملات، كما لا يمكن استخدام التمايز DISTINCT معها	COUNT(*)
ترجع القيمة العليا في التعبير، ويمكن استخدام الدالة Max مع الأعمدة ذات النوع العددي، والمحرفية، والأعمدة ذات النوع datetime، ولكنها لا تُستخدم مع الأعمدة ذات النوع bit، كما تعطي الدالة MAX مع الأعمدة المحرفية أعلى قيمة في تسلسل مرتب، وتتجاهل هذه الدالة القيم الفارغة	MAX
ترجع القيمة الدنيا في التعبير. يمكن استخدام الدالة MIN مع أعمدة عددية، ومحرفية، وذات النوع bit، datetime، ولكنها لا تُستخدم مع أعمدة لها النوع datetime، كما تعطي الدالة MIN مع الأعمدة المحرفية أعلى قيمة في تسلسل مرتب، وتتجاهل هذه الدالة القيم الفارغة	MIN
ترجع مجموع كل القيم، أو فقط القيم المميزة في التعبير، و يمكن استخدام الدالة SUM DISTINCT مع الأعمدة العددية فقط	SUM

سنعرض فيما يلي أمثلةً عن كل من دوال التجميع الموجودة في الجدول السابق.

- المثال الأول: الدالة AVG

```
SELECT AVG (price) AS 'Average Title Price'
FROM Books
```

- المثال الثاني: الدالة COUNT •

```
SELECT COUNT(PubID) AS 'Number of Publishers'
FROM Publishers
```

المثال الثالث: الدالة COUNT

```
SELECT COUNT(province) AS 'Number of Publishers'
FROM Publishers
```

- المثال الرابع: الدالة COUNT (\*) •

```
SELECT COUNT(*)
FROM Employees
WHERE job_lvl = 35
```

- المثال الخامس: الدالة MAX •

```
SELECT MAX (HireDate)
FROM Employees
```

- المثال السادس: الدالة MIN •

```
SELECT MIN (price)
FROM Books
```

- المثال السابع: الدالة SUM •

```
SELECT SUM(discount) AS 'Total Discounts'
FROM Discounts
```

## 13.5.2 دالة التحويل Conversion function

تحوّل دالة التحويل نوع بيانات معين إلى نوع بيانات آخر.

يُحّوّل السعر price الذي يحتوي ضمنه على تسعتين 99 إلى خمسة محارف في المثال الآتي، حيث تكون صيغة التعليمية بالصورة التالية:

```
SELECT 'The date is ' + CONVERT(varchar(12), getdate())
```

إليك مثال:

```

SELECT CONVERT(int, 10.6496)
SELECT title_id, price
FROM Books
WHERE CONVERT(char(5), price) LIKE '%99%'

```

تغيير دالة التحويل في المثال التالي البيانات إلى نوع بيانات بحجم مختلف:

```

SELECT title_id, CONVERT(char(4), ytd_sales) as 'Sales'
FROM Books
WHERE type LIKE '%cook'

```

### 13.5.3 دالة التاريخ Date function

تُنجز دالة التاريخ تاريخاً عن طريق إضافة فاصل زمني إلى تاريخ محدّد، والنتيجة هي قيمة لها نوع datetime، وتساوي التاريخ مضافاً إليه عدد أجزاء التاريخ date parts .

إذا كان معامل دالة التاريخ قيمةً من النوع smalldatetime، فستكون النتيجة قيمةً من النوع smalldatetime أيضاً.

تُستخدم الدالة DATEADD لإضافة وزيادة قيم التاريخ، وصيغة هذه الدالة هي:

```
DATEADD(datepart, number, date)
```

إليك مثال:

```

SELECT DATEADD(day, 3, hire_date)
FROM Employees

```

يستخدم المثال الآتي الدالة DATEDIFF(datepart, date1, date2)، ويعيد هذا الأمر عدد أجزاء التاريخ أو "الحدود" boundaries المتقاطعة بين تاريخين محددين.

تجعل طريقة حساب الحدود المتقاطعة النتيجة التي أعطتها الدالة DATEDIFF متوافقة مع جميع أنواع البيانات، مثل الدقائق، والثواني، والميلي ثانية.

```

SELECT DATEDIFF(day, HireDate, 'Nov 30 1995')
FROM Employees

```

يمكننا فحص أي جزء من تاريخ معين من السنة إلى الميلي ثانية.

يعرض الجدول التالي أجزاء التاريخ DATEPART، و اختصاراتها، وقيمها المقبولة التي يعترف بها SQL Server خادم

قيمة VALUES	اختصاره ABBREVIATION	جزء التاريخ DATE PART
1753-9999	yy	Year
1-4	qq	Quarter
1-12	mm	Month
1-366	dy	Day of year
1-31	dd	Day
1-53	wk	Week
1-7 (Sun.-Sat.)	dw	Weekday
0-23	hh	Hour
0-59	mi	Minute
0-59	ss	Second
0-999	ms	Millisecond

### 13.5.4 الدوال الرياضية Mathematical functions

تجري الدوال الرياضية عمليات على البيانات العددية، ويعرض المثال التالي السعر الحالي لكل كتاب يبيعه الناشر، كما يعرض ما سيكون عليه الأمر إذا ارتفعت جميع الأسعار بنسبة 10%:

```

SELECT Price, (price * 1.1) AS 'New Price', title
FROM Books
SELECT 'Square Root' = SQRT(81)
SELECT 'Rounded' = ROUND(4567.9876,2)
SELECT FLOOR (123.45)

```

## 13.6 ضم الجداول Joining Tables

يُعدّ ضم جداولين أو أكثر مثل عملية موازنة بيانات ضمن أعمدة محدّدة، واستخدام نتائج الموازنة لتشكيل جدول جديد من الصنف المطلوب لذلك.

تقوم عبارة الضم join بما يلي:

- تحدد عموداً من كل جدول.
- توازن القيم الموجودة في تلك الأعمدة صفاً صفاً.
- تدمج الصنف ذات القيم المطلوبة ضمن صنف جديد.

تكون الموازنة عادةً مساواةً - أي القيم التي تتطابق مع بعضها البعض تماماً، ولكن يمكن تحديد أنواع ضم أخرى أيضاً.

سنشرح جميع أنواع الضم المختلفة أدناه، مثل: الضم الداخلي `inner`, واليساري (الخارجي)، واليميني (الخارجي)، والضم المتقاطع `cross join`.

### 13.6.1 الضم الداخلي

يربط الضم الداخلي جدولين في عمود له نفس نوع البيانات، وينتج الصفوف التي تتطابق فيها قيم العمود فقط، حيث يجري تجاهل الصفوف التي لا مثيل لها.

المثال الأول:

```
SELECT jobs.job_id, job_desc
  FROM jobs
  INNER JOIN Employees ON emp
  loyee.job_id = jobs.job_id
 WHERE jobs.job_id < 7
```

المثال الثاني:

```
SELECT authors.au_fname, authors.au_lname, books.royalty, title
  FROM authors
  INNER JOIN titleauthor ON authors.au_id=titleauthor.au_id
  INNER JOIN books ON titleauthor.title_id=books.title_id
 GROUP BY authors.au_lname, authors.au_fname, title, title.royalty
 ORDER BY authors.au_lname
```

### 13.6.2 الضم اليساري الخارجي

ينتج عن الضم الخارجي اليساري كل الصفوف الخارجية اليسرى، إذ تضمن جميع الصفوف من الجدول الأيسر التي لا تتحقق الشرط المحدد في مجموعة النتائج، وتُضبط أعمدة الخرج من الجدول الآخر على القيمة الفارغة `NULL`.

يستخدم المثال التالي الصيغة الجديدة للضم اليساري الخارجي:

```
SELECT publishers.pub_name, books.title
  FROM Publishers
  LEFT OUTER JOIN Books On publishers.pub_id = books.pub_id
```

بينما يستخدم المثال التالي الصيغة القديمة للضم الخارجي اليساري:

```

SELECT publishers.pub_name, books.title
FROM Publishers, Books
WHERE publishers.pub_id *= books.pub_id

```

### 13.6.3 الضم الخارجي الأيمن

يتضمن الضم الخارجي الأيمن في مجموعة النتائج الخاصة به كافة الصفوف من الجدول الأيمن التي لا تتحقق الشرط المحدد، وتُضيّط أعمدة الخرج المقابلة من الجدول الآخر على القيمة الفارغة `NULL`.

يستخدم المثال التالي الصيغة الجديدة للضم الخارجي الأيمن:

```

SELECT titleauthor.title_id, authors.au_lname, authors.au_fname
FROM titleauthor
RIGHT OUTER JOIN authors ON titleauthor.au_id = authors.au_id
ORDER BY au_lname

```

بينما يوضح المثال التالي الصيغة القديمة المستخدمة للضم الخارجي الأيمن:

```

SELECT titleauthor.title_id, authors.au_lname, authors.au_fname
FROM titleauthor, authors
WHERE titleauthor.au_id =* authors.au_id
ORDER BY au_lname

```

### 13.6.4 الضم الخارجي الكامل

يحدد الضم الخارجي الكامل أنه في حالة عدم تطابق صفات من أي من الجدولين مع معايير التحديد، فسيُضمنَنَ الصفات في مجموعة النتائج، وتُضيّط أعمدة الخرج الخاصة به التي تتوافق مع الجدول الآخر إلى القيمة الفارغة `NULL`.

فيما يلي مثال عن ضم خارجي كامل:

```

SELECT books.title, publishers.pub_name, publishers.province
FROM Publishers
FULL OUTER JOIN Books ON books.pub_id = publishers.pub_id
WHERE (publishers.province <> "BC" and publishers.province <> "ON")
ORDER BY books.title_id

```

## 13.6.5 الضم المتقاطع Cross join

الضم المتقاطع هو ناتج دمج جدولين، وينتج عن هذا الضم صفوف حالة عدم استخدام الشرط WHERE نفسها، أي كما يلي:

```
SELECT au_lname, pub_name,
  FROM Authors CROSS JOIN Publishers
```

## 13.7 مصطلحات أساسية

- دالة التجميع **aggregate function**: تعيد قيماً موجزة.
- دالة الترتيب **ASC**: ترتيب تصاعدي.
- دالة التحويل **conversion function**: تحول نوع بيانات معين إلى نوع بيانات آخر.
- الضم المتقاطع **cross join**: ناتج دمج جدولين.
- دالة التاريخ **date function**: تعرض معلومات عن التواريخ والأوقات.
- تعليةمة **DELETE**: تزيل صفوفاً من مجموعة سجلات.
- دالة الترتيب **DESC**: ترتيب تناظري.
- الضم الخارجي الكامل **full outer join**: يحدد الحالة التي لا يتطابق فيها صف من أي جدول مع معايير الاختيار.
- دالة **GROUP BY**: وهي تُستخدم من أجل إنشاء صف خرج واحد لكل مجموعة، وتنتج قيماً موجزاً للأعمدة المختارة.
- الضم الداخلي **inner join**: يربط جدولين من خلال عمود له نوع البيانات نفسه.
- تعليةمة **INSERT**: يضيف صفوفاً إلى جدول.
- الضم الخارجي اليساري **left outer join**: يُنتج جميع الصفوف الخارجية اليسرى.
- الدالة الرياضية **mathematical function**: تجري عمليات على البيانات العددية.
- الضم الخارجي اليميني **right outer join**: يتضمن جميع الصفوف من الجدول الأيمن الذي لم يحقق الشرط المحدد.
- تعليةمة **SELECT**: تُستخدم لاستعلام عن البيانات في قاعدة البيانات.

- **الدالة المتعلقة بالسلسل string function**: تجري عمليات على سلسل المحارف، أو البيانات الثنائية، أو التعابير.
- **الدالة المتعلقة بالنظام system function**: تعيد معلومات من قاعدة البيانات.
- **الدوال المتعلقة بالنصوص والصور text and image functions**: تجري عمليات على البيانات النصية وبيانات الصور.
- **تعليمية UPDATE**: تغير البيانات في الصنوف الموجودة إما بإضافة بيانات جديدة، أو بتعديل البيانات الموجودة.
- **حرف البدل wildcard**: يسمح للمستخدم بمطابقة الحقول التي تحتوي على محارف معينة.

## 13.8 تمارين

استخدم نموذج قاعدة البيانات [PUBS](#) الذي أنشأته مايكروسوفت لحل لأسئلة التالية:

1. اعرض قائمةً بتواريخ النشر والعنوانين (الكتب) التي نُشرت في عام 2011.
2. اعرض قائمةً بعناوين الكتب المصنفة على أساس كتب طبع تقليدي، أو كتب طبع حديث باستخدام جدول الكتب `Books`.
3. اعرض جميع المؤلفين `authors` الذين تتألف أسماؤهم الأولى من خمسة أحرف.
4. اعرض من جدول الكتب: النوع `type`، والسعر `price`، والمعرف `Pub_id`، وعنوان `title` الكتب التي وضعها كل ناشر، ثم أعد تسمية عمود النوع `type` ليصبح فئة الكتاب `Book Category`، ورتبه حسب النوع `type` تنازلياً، ثم حسب السعر `price` تصاعدياً.
5. اعرض الحقل `title_id`، والحقول `pubdate`، والحقول `pubdate` مضافاً إليه ثلاثة أيام، باستخدام جدول الكتب.
6. حدد باستخدام الداللين `datediff`، `getdate` مقدار الوقت المنقضي (مقداراً بالأشهر) منذ نشر الكتب في جدول الكتب.
7. اعرض قائمةً بمعزفات العنوانين `ID`، `title`، وكمية `quantity` جميع الكتب التي بيع منها أكثر من 30 نسخة.
8. اعرض قائمةً بجميع أسماء عائلات المؤلفين الذين يعيشون في أونتاريو `Ontario` -أو اختصاراً، والمدن التي يعيشون فيها.

9. اعرض جميع الصفوف التي تحتوي ضمنها على القيمة 60 في حقل شروط الدفع payterms، واستخدم جدول المبيعات.Sales

10. اعرض جميع المؤلفين الذين تتألف أسماؤهم الأولى من خمسة حروف، أو تنتهي بالحرف O أو بالحرف A، وتبدأ بالحرف M أو بالحرف P.

11. اعرض جميع عنوانين الكتب التي تكلفتها أكثر من 30 دولاراً، وإما تبدأ بحرف T أو معرف ناشرها هو 0.0877

12. اعرض من جدول الموظفين Employees أعمدة الاسم الأول fname، واسم العائلة lname، ومعرف الموظف emp\_id، والمستوى الوظيفي job\_lvl للموظفين الذين مستواهم الوظيفي أكبر من 200، وأعد تسمية عنوانين هذه الأعمدة لتصبح "First Name" و "Last Name" و "IDENTIFICATION" و "Job Level" .

13. اعرض قيمة حقوق المؤلف royalty، وحقوق المؤلف مضافة إليها 50% مع تسمية هذا الحقل إلى "royalty plus" ، ومعرف العنوان title\_id، وذلك باستخدام جدول Royaltiesched .

14. أنشئ السلسلة "12xxxx567" من السلسلة "1234567" باستخدام الدالة STUFF .

15. اعرض أول 40 حرفًا من كل عنوان كتاب، إلى جانب متوسط المبيعات الشهرية لهذا العنوان حتى الآن أي 12 / ytd\_sales ، واستخدم جدول العنوانات .Title

16. اعرض عدد الكتب التي حُدّدت أسعارها.

17. اعرض قائمةً بكتب الطبخ مع متوسط تكلفة جميع الكتب لكل نوع باستخدام الأمر GROUP BY . إليك مجموعة أسئلة متقدمة باستخدام الاستعلامات Union و Minus و Intersect :

1. تعلم معاملات المجموعات العلائقية UNION، وINTERSECT، وMINUS بصورة صحيحة فقط إذا كانت العلاقات متوافقة مع الاتحاد union-compatible، فماذا يعني التوافق مع الاتحاد؟ وكيف يمكنك التحقق من هذا الشرط؟

2. ما هو الفرق بين UNION، وALL UNION؟ وكتب صيغة كل منها.

3. لنفترض أن لديك جدولين هما Employees وEmployees\_1، بحيث يحتوي الجدول Employees على سجلات لثلاثة موظفين، هم: Anne McDonald، John Cretchakov، Alice Cordoza، كما يحتوي الجدول Employees\_1 على سجلات الموظفين John Cretchakov، Mary Cheng، ما هو خرج الاستعلام UNION باستخدام المعلومات السابقة؟ اعرض قائمةً بخرج هذا الاستعلام.

4. استخدم معلومات الموظف في السؤال رقم 3 لمعرفة خرج الاستعلام ALL UNION، واعرض قائمةً بخرج هذا الاستعلام.

5. استخدم معلومات الموظف في السؤال رقم 3 لمعرفة خرج الاستعلام INTERSECT، واعرض قائمةً بخرج هذا الاستعلام.

6. استخدم معلومات الموظف في السؤال رقم 3 لمعرفة خرج الاستعلام EXCEPT، واعرض قائمةً بخرج هذا الاستعلام.

7. ما هو الضم المتقطع cross join؟ واعطِ مثلاً عن صيغته.

8. اشرح هذه الأنواع الثلاثة للضم:

1. الضم الخارجي اليساري left outer join

2. الضم الخارجي اليميني right outer join

3. الضم الخارجي الكامل full outer join

9. ما هو الاستعلام الفرعى subquery، وما هي خصائصه الأساسية؟

10. ما هو الاستعلام الفرعى المرتبط correlated subquery؟ واعطِ مثلاً على ذلك.

11. افترض أنّ جدول المنتجات Product يحتوي على سمتين هما PROD\_CODE، وCODE، وقيم السمة PROD\_CODE هي: ABC، وDEF، وGHI، وJKL، حيث يجري مطابقة هذه القيم مع قيم السمة VEND\_CODE التالية: 125، 124، 123، و124، و125 على التوالي، فمثلاً، تقابل قيمة السمة PROD\_CODE التي هي ABC قيمة السمة VEND\_CODE التي هي 125، كما يحتوي جدول البائعين Vendor على سمة واحدة هي VEND\_CODE التي لها القيم التالية: 123 و 124 و 125 و 126 حيث تُعدّ السمة VEND\_CODE في جدول المنتجات مفتاحاً خارجياً للسمة PROD\_CODE في جدول البائعين.

12. ما هو خرج الاستعلامات التالية باستخدام المعلومات الموجودة في السؤال رقم 11؟

1. الاستعلام UNION بناءً على هذين الجدولين.

2. الاستعلام UNION ALL بناءً على هذين الجدولين.

3. الاستعلام INTERSECT بناءً على هذين الجدولين.

4. الاستعلام MINUS بناءً على هذين الجدولين.

إليك أسئلة متقدمة باستخدام الضم joins:

1. اعرض قائمةً بجميع عنوانين الكتب، وأرقام المبيعات في جدول الكتب Books، والمبيعات Sales، بما في ذلك العنوانين التي لا تحتوي على مبيعات باستخدام عملية الضم join.
2. اعرض قائمةً بأسماء عائلات المؤلفين، وجميع عنوانين الكتب المنشورة، والمرتبطة بكل مؤلف، بحيث تكون مرتبةً حسب اسم عائلة المؤلف باستخدام الضم، ثم احفظ ذلك على أساس عرض view يدعى Published Authors.
3. استخدم استعلاماً فرعياً لعرض جميع المؤلفين الذين يحصلون على حقوقهم بنسبة 100% ويعيشون في ألبرتا Alberta، وذلك بإظهار الاسم الأول، واسم العائلة، والرمز البريدي، ثم احفظ ذلك على أساس عرض view بعنوان AuthorsView، وبعدها أعد تسمية اسم عائلة المؤلف، واسمه الأول بالصورة 'First Name'، و 'Last Name' عند إنشاء العرض.
4. اعرض المتاجر stores التي لم تُبع الكتب الذي عنوانه 'Is Anger the Enemy?'.  
5. اعرض قائمةً بأسماء المتاجر للمبيعات بعد 2013، حيث يكون تاريخ الطلب Order Date أكبر من 2013، وذلك بعرض اسم المتجر store name، وتاريخ الطلب order date.
6. اعرض قائمةً بعنوانين الكتب المباعة في المتجر الذي اسمه News & Brews، وذلك بعرض اسم المتجر، وعنوانين الكتب، وتاريخ الطلب.
7. اعرض قائمةً بإجمالي المبيعات حسب العنوان، وذلك بعرض عمودي الكمية الإجمالية والعنوان.
8. اعرض قائمةً بإجمالي المبيعات حسب النوع، وذلك بعرض عمودي الكمية الإجمالية والنوع.
9. اعرض قائمةً بإجمالي المبيعات  $qty * price$  حسب النوع، وذلك بعرض عمودي إجمالي قيمة الدولارات والنوع.
10. احسب العدد الكلي لأنواع الكتب لكل ناشر، وأظهر اسم الناشر، والعدد الإجمالي لأنواع الكتب لكل ناشر على حدة.
11. اعرض أسماء الناشرين الذين ليس لديهم أي نوع من الكتب، وذلك بعرض اسم الناشر فقط.

# 14. الملحق أ: مثال عملي عن تصميم قاعدة بيانات لجامعة

فيما يلي متطلبات البيانات لمنتج من أجل دعم تسجيل وتقديم المساعدة لطلاب جامعة تعليم إلكتروني وهمية. تحتاج جامعة تعليم إلكتروني إلى الاحتفاظ بتفاصيل طلابها وموظفيها، والمقررات التي تقدمها وأداء الطلاب الذين يدرسون فيها. تدار الجامعة في أربع مناطق جغرافية (إنجلترا واسكتلندا وويلز وأيرلندا الشمالية).

يجب تسجيل معلومات كل طالب في البداية عند التسجيل، ويتضمن ذلك رقمتعريف الطالب الصادر في الوقت والاسم وسنة التسجيل والمنطقة الموجود فيها الطالب. ليس الطالب ملزماً بالتسجيل في أي مقرر عند التسجيل، فيمكنه التسجيل في مقرر ما في وقتٍ لاحق.

يجب أن تتضمن المعلومات المسجلة لكل عضو في القسم التعليمي وقسم الإرشاد رقم الموظف والاسم والمنطقة التي يوجد بها. قد يعمل كل موظف كمرشد counselor لطالب أو أكثر، وقد يعمل كمدرس tutor طالب أو أكثر في مقرر أو أكثر. قد لا يُخصص لأحد الموظفين أي طالب كمدرس أو كمرشد في أي وقتٍ معين.

يملك كل طالب مرشدًا واحدًا يُخصص له عند التسجيل، ويقدم الدعم للطالب طوال حياته الجامعية. يُخصص للطالب مدرسٌ منفصلٌ لكل مقرر سجل فيه الطالب. يُسمح للموظف فقط العمل كمرشد أو كمدرس طالبٌ مقيم في نفس منطقته.

يجب أن يكون لكل مقرر متوفّر للدراسة رمز مقرر وعنوان وقيمة من حيث نقاط الائتمان، حيث يكون للمقرر إما 15 نقطة أو 30 نقطة. قد يكون للمقرر حصة quota لعدد الطلاب المسجلين فيه في أي عرض. لا يحتاج المقرر إلى أي طالب مسجل فيه (مثل المقرر الذي كُتب للتو ثم عُرض للدراسة).

يُقيّد الطلاب في عدد المقررات التي يمكنهم التسجيل فيها في نفس الوقت، فقد لا يأخذون المقررات في نفس الوقت إذا تجاوز مجموع النقاط المدمجة للمقررات المسجلين فيها 180 نقطة.

قد يكون للمقرر ذي العدد 15 نقطة ما يصل إلى ثلاثة وظائف لكل عرض، ويكون للمقرر ذي العدد 30 نقطة ما يصل إلى خمس وظائف لكل عرض. تُسجل درجة الوظيفة في أي مقرر كعلامة من 100.

قاعدة بيانات الجامعة التالية نموذج بيانات محتمل يصف مجموعة المتطلبات المذكورة أعلاه. يحتوي النموذج على عدة أجزاء، بدءاً من مخطط ERD ووصف لأنواع الكيانات والقيود والافتراضات.

## 14.1 عملية التصميم

1. الخطوة الأولى هي تحديد النوى والتي هي عادة أسماء: الموظفين Staff والمقرر Course والطالب Student والوظيفة Assignment.

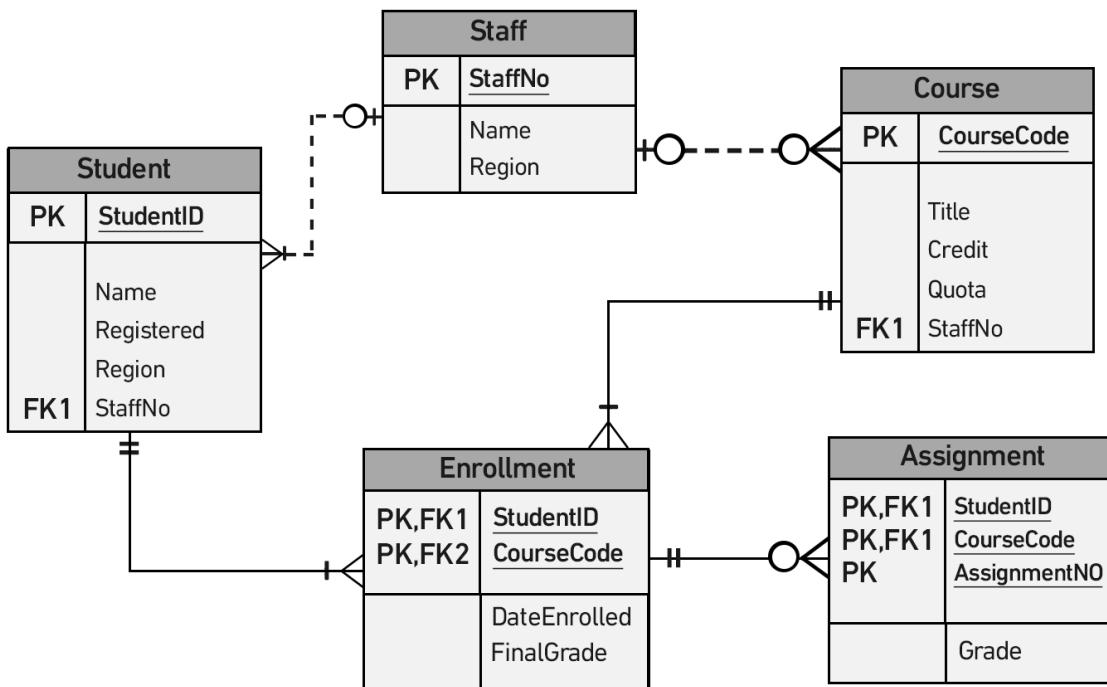
2. الخطوة التالية هي توثيق جميع السمات attributes لكل كيان entity. هذا هو المكان الذي تحتاج فيه إلى التأكيد من توحيد normalized جميع الجداول توحيداً صحيحاً.

3. أنشئ مخطط ERD الأولي وراجعه مع المستخدمين.

4. أجر تغييرات إن لزم الأمر بعد مراجعة مخطط ERD.

5. تحقق من نموذج ER مع المستخدمين لوضع اللمسات الأخيرة على التصميم.

يوضح الشكل التالي مخطط ERD للجامعة الذي يمثل نموذج بيانات لنظام سجلات الطلاب والموظفين:



## 14.1.1 الكيان Entity

Student (StudentID, Name, Registered, Region, StaffNo) •

Staff (StaffNo, Name, Region) • يحتوي هذا الجدول على مدربين وغيرهم من الموظفين

Course (CourseCode, Title, Credit, Quota, StaffNo) •

Enrollment (StudentID, CourseCode, DateEnrolled, FinalGrade) •

Assignment (StudentID, CourseCode, AssignmentNo, Grade) •

## 14.1.2 القيود Constraints

يجوز لأحد الموظفين أن يدرس أو يرشد الطلاب المتواجدين في نفس منطقتهم فقط. •

قد لا يسجل الطالب في مقررات لا تزيد قيمتها عن أكثر من 180 نقطة في نفس الوقت. •

للسمة Credit (ضمن المقرر Course) قيمة هي 15 أو 30 نقطة. •

قد يكون للمقرر الذي له 30 نقطة ما يصل إلى خمس وظائف، بينما يكون للمقرر الذي له 15 نقطة ما يصل إلى ثلات وظائف. •

للسمة Grade (ضمن الوظيفة Assignment) قيمة هي علامة من 100. •

## 14.1.3 الافتراضات Assumptions

يستطيع الطالب أن يسجل مرة واحدة للمقرر حيث تُسجل عمليات التسجيل الحالية فقط. •

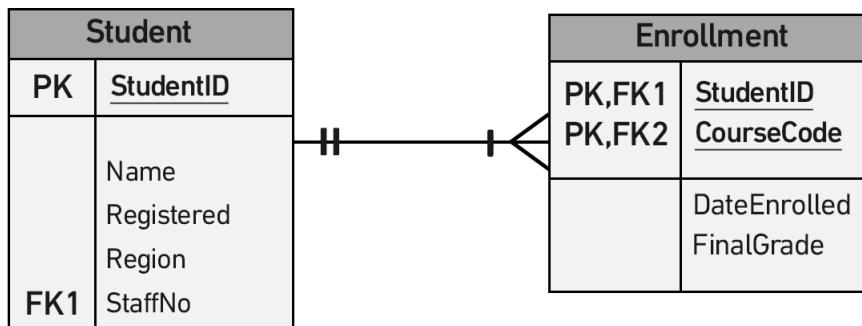
تُقدم الوظيفة مرة واحدة فقط. •

## 14.1.4 العلاقات Relationships (تشمل عدديّة العلاقة cardinality)

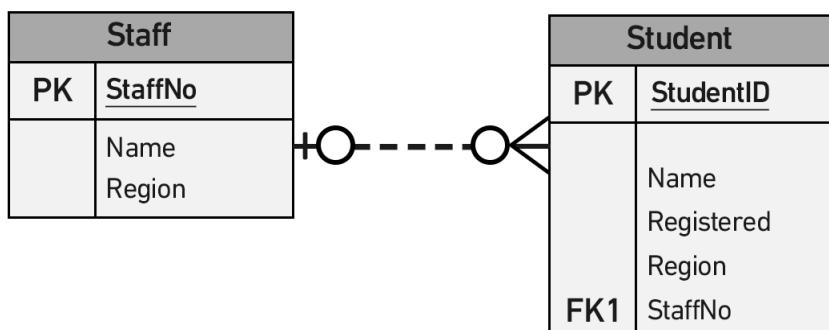
لاحظ في الشكل الآتي أن سجل الطالب مرتبط مع مقررات مُسجّلة بحد أدنى مقرر واحد إلى مقررات متعددة كحد أقصى.

يجب أن يكون لكل تسجيل enrollment طالب صالح.

بما أن معرف الطالب StudentID هو جزء من المفتاح الرئيسي PK، فلا يمكن أن يكون فارغا null، لذلك يجب وجود معرف طالب StudentID مدخل في جدول الطالب مرة واحدة على الأقل كحد أقصى، لأن المفتاح الرئيسي PK يجب ألا يتكرر.



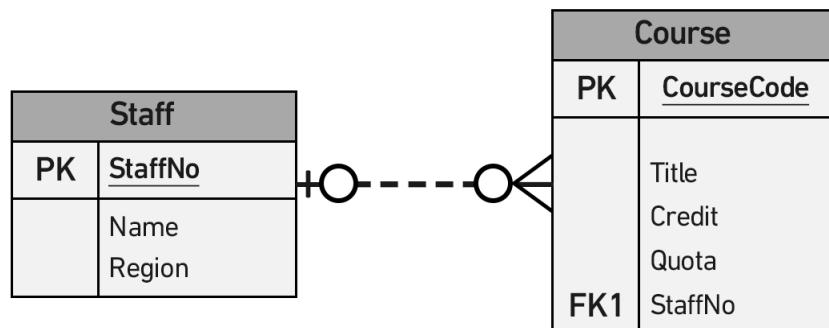
يوضح الشكل الآتي ارتباط سجل الموظفين (المدرس هنا) بحد أدنى 0 طالب وبطلاب متعددين كحد أقصى. قد يكون لسجل الطالب مدرس **tutor** أو قد يكون بدون مدرس.



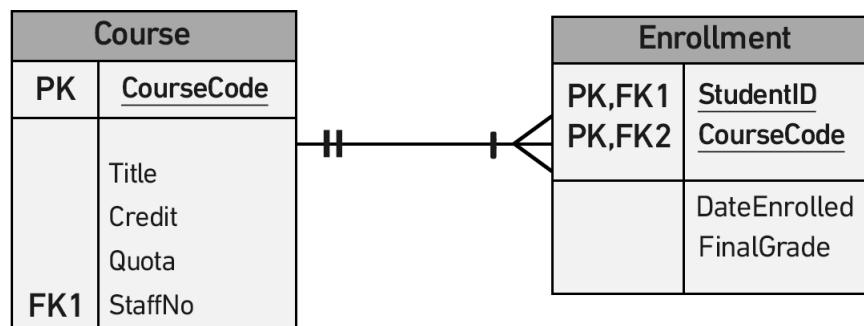
يسمح الحقل **StaffNo** الموجود في جدول الطلاب **Student** بالقيم الفارغة التي تُمثل بالقيمة 0 على الجانب الأيسر من الشكل السابق. لكن في حالة وجود الحقل **StaffNo** في جدول الطلاب **Student** في جدول **Enrollment**, فيجب أن يكون موجوداً في جدول الموظفين **Staff** بحد أقصى مرة واحدة (المُمثل بالقيمة 1 في الشكل السابق).

يرتبط سجل الموظفين **Staff** (المدرس هنا) بحد لا يقل عن 0 مقرّر كحد أدنى وبمقررات متعددة كحد أقصى. قد يكون المقرر **course** مرتبّطاً بمدرس **instructor** أو غير مرتبّط بمدرس.

الحقل **StaffNo** الموجود في جدول **Course** هو المفتاح الخارجي **FK** الذي يمكن أن يكون فارغاً، ويُمثل ذلك من خلال القيمة 0 على الجانب الأيسر من العلاقة في الشكل الآتي. إذا احتوى الحقل **StaffNo** على بيانات، فيجب أن يكون في جدول الموظفين **Staff** بحد أقصى مرة واحدة، ويُمثل ذلك بالقيمة 1 على الجانب الأيسر من العلاقة.

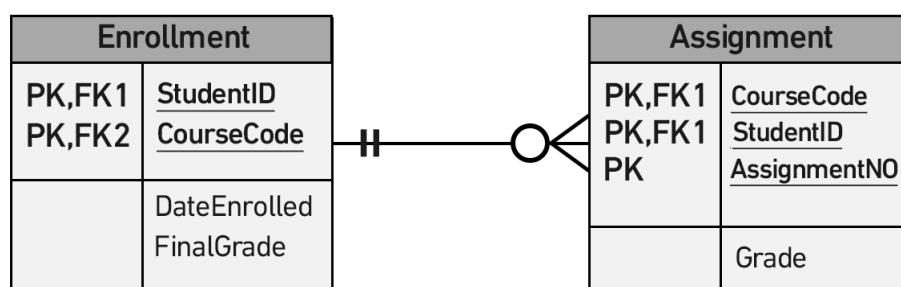


يجب توفير المقرر (في عملية التسجيل enrollment) مرة واحدة على الأقل ومرات متعددة كحد أقصى.  
يجب أن يحتوي جدول التسجيل Enrollment على مقرر واحد صالح على الأقل إلى مقررات متعددة  
كحد أقصى.



يمكن أن تحتوي عملية التسجيل على 0 مهمة كحد أدنى أو مهام متعددة كحد أقصى.  
يجب أن ترتبط الوظيفة assignment بتسجيل واحد على الأقل وبتسجيل واحد كحد أقصى.

يجب أن يحتوي كل سجل في جدول الوظائف على سجل تسجيل صالح، ويمكن ربط سجل تسجيل واحد بمهام متعددة.



## 15. الملحق ب: أمثلة عملية عن إنشاء

### مخططات ERD

#### 15.1 التمرين الأول: شركة تصنيع Manufacturer

تنتج شركة تصنيع منتجات، وتخزن معلومات المنتج التالية: اسم المنتج `product name` ومعرف المنتج `product ID` والكمية المتوفرة `quantity`. تتكون هذه المنتجات من مكونات متعددة، ويتوفر مورد أو أكثر كل مكون. تُحفظ معلومات المكون التالية: معرف المكون `component ID` واسم المكون `name` ووصف عنه `description`. الموردون `suppliers` الذين يوفرونها والمنتجات `products` التي تستخدم هذا المكون (استخدم الشكل الآتي لحل هذا التمرين).

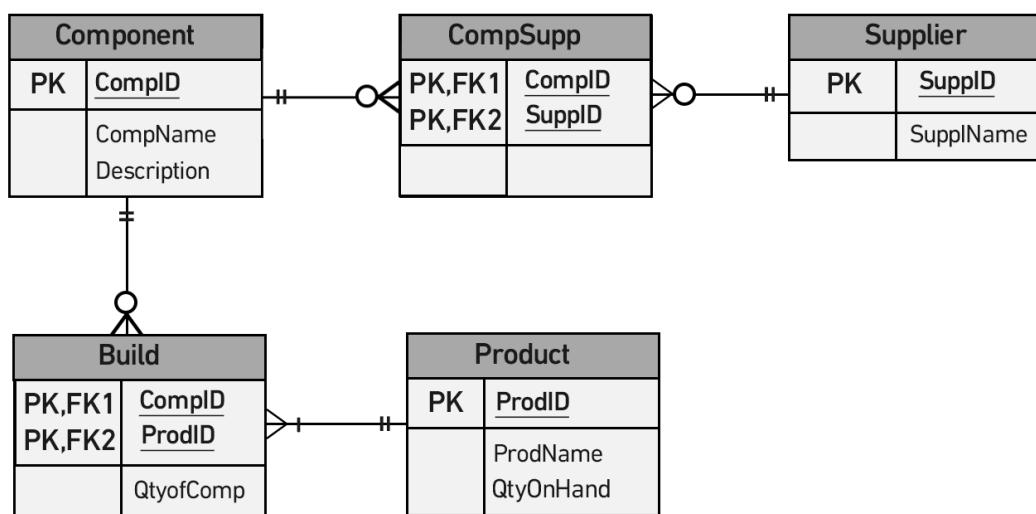
1. أنشئ مخطط ERD لإظهار كيفية تبع هذه المعلومات.
2. اعرض أسماء الكيانات `entity names` والمفاتيح الرئيسية `primary keys` وسمات `attributes` كل كيان وال العلاقات بين الكيانات وعددي العلاقة `.cardinality`.

#### 15.1.1 الافتراضات Assumptions

- يمكن وجود المورد دون أن يوفر مكونات.
- ليس واجباً أن يرتبط مكون بمحور.
- ليس واجباً أن يرتبط مكون مع منتج، فليست جميع المكونات مستخدمة في المنتجات.
- لا يمكن أن يوجد منتج بدون مكونات.

## 15.1.2 جواب مخطط ERD

- Component(CompID, CompName, Description) PK=CompID
- Product(ProdID, ProdName, QtyOnHand) PK=ProdID
- Supplier(SuppID, SuppName) PK = SuppID
- CompSupp(CompID, SuppID) PK = CompID, SuppID
- Build(CompID, ProdID, QtyOfComp) PK= CompID, ProdID



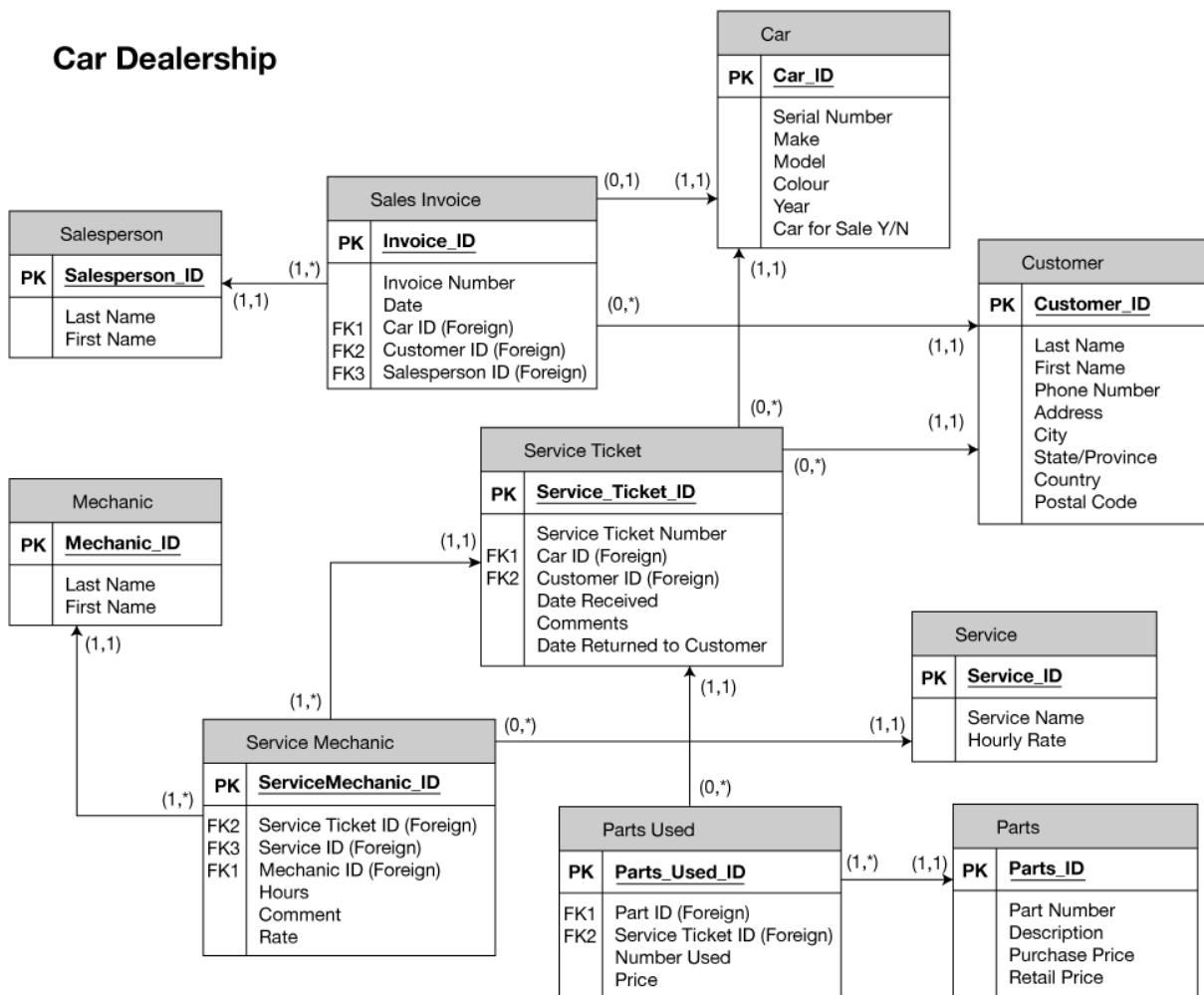
## 15.2 التمرين الثاني: وكيل سيارات

أنشئ مخطط ERD لوكيل سيارات، حيث يبيع هذا الوكيل كلاً من السيارات الجديدة والمستعملة، ويشغل قسمًا للخدمات. ابن تصميمك على قواعد الأعمال التالية:

- قد يبيع مندوب المبيعات salesperson سيارات متعددة، ولكن تُباع كل سيارة بواسطة مندوب مبيعات واحد فقط.
- يمكن أن يشتري العميل customer سيارات متعددة، ولكن تُشتري كل سيارة بواسطة عميل واحد فقط.
- يكتب مندوب المبيعات فاتورة invoice واحدة لكل سيارة يبيعها.
- يحصل العميل على فاتورة لكل سيارة يشتريها.
- قد يأتي العميل من أجل الحصول على خدمات لسيارته فقط، وهذا يعني أن العميل لا يحتاج إلى شراء سيارة لكي يُصنف كعميل.

- إذا جلب العميل سيارةً أو أكثر لإصلاحها أو للحصول على خدمة، فستكتب تذكرة خدمة service ticket لكل سيارة.
- يحتفظ وكيل السيارات بتاريخ خدمة لكل من السيارات المخدّمة، ويشير إلى سجلات الخدمة عن طريق رقم السيارة التسلسلي.
- يمكن أن يعمل على السيارة التي تُجلب للحصول على خدمة ميكانيكيون متعددون، وقد يعمل كل ميكانيكي على سيارات متعددة.
- قد تحتاج السيارة التي تحصل على خدمة إلى قطع أو قد لا تحتاج إلى قطع (مثل عملية ضبط المفّحّم أو تنظيف فوهة حاقد الوقود التي لا تتطلب توفير قطعٍ جديدة).

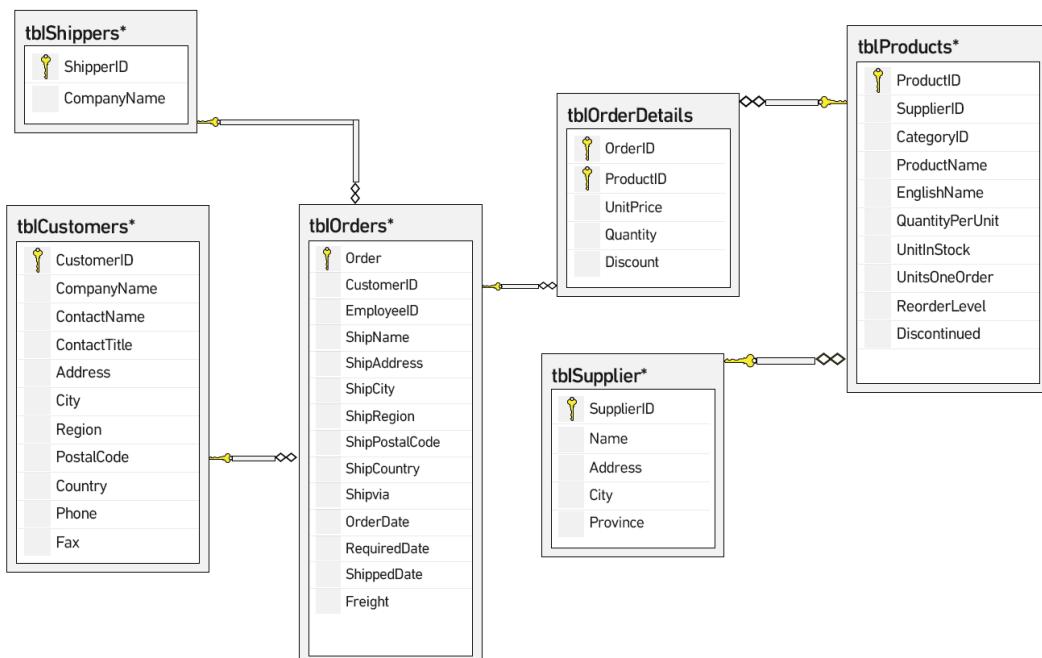
## 15.2.1 جواب مخطط ERD



# 16. الملحق ج: حل تمارين باستخدام لغة SQL

نُزّل السكريبت التالي: OrdersAndData.sql

## 16.1 الجزء الأول: استخدم لغة DDL



استخدم السكريبت orderData.sql الذي ينشئ جداول ويضيف بيانات مخطط ERD للطلبات والبيانات في الشكل السابق.

.1. أنشئ قاعدة بيانات تسمى Orders، وعدل السكريبت لدمج المفتاح الرئيسي PK والسلامة المرجعية مع التعديلات بما في ذلك القيود الموجودة في الخطوة 3.

.2. أضف القيود التالية:

Country (tblCustomers table: Country) قيمته الافتراضية هي

tblOrderDetails: Quantity – > 0

CompanyName (tblShippers table: CompanyName) يجب أن يكون فريداً

ShippedDate (tblOrders table: ShippedDate) يجب أن يكون أكبر تاريخ الطلب

```
CREATE DATABASE Orders
```

```
Go
```

```
Use Orders
```

```
Go
```

```
Use Orders
```

```
Go
```

```
CREATE TABLE [dbo].[tblCustomers]
```

```
[CustomerID] nvarchar(5) NOT NULL,  

[CompanyName] nvarchar(40) NOT NULL,  

[ContactName] nvarchar(30) NULL,  

[ContactTitle] nvarchar(30) NULL,  

[Address] nvarchar(60) NULL,  

[City] nvarchar(15) NULL,  

[Region] nvarchar(15) NULL,  

[PostalCode] nvarchar(10) NULL,  

[Country] nvarchar(15) NULL  

Constraint df_country DEFAULT 'Canada',
```

```
[Phone] nvarchar(24) NULL,  

[Fax] nvarchar(24) NULL,
```

```
Primary Key (CustomerID)
```

```
);
```

```
CREATE TABLE [dbo].[tblSupplier] (
```

```
[SupplierID] int NOT NULL,  

[Name] nvarchar(50) NULL,  

[Address] nvarchar(50) NULL,
```

```

[City]           nvarchar(50) NULL,
[Province]      nvarchar(50) NULL,
Primary Key (SupplierID)
);

CREATE TABLE [dbo].[tblShippers] (
[ShipperID]      int NOT NULL,
[CompanyName]    nvarchar(40) NOT NULL,
Primary Key (ShipperID),<
CONSTRAINT uc_CompanyName UNIQUE (CompanyName)
);

CREATE TABLE [dbo].[tblProducts] (
[ProductID]      int NOT NULL,
[SupplierID]     int NULL,
[CategoryID]     int NULL,
[ProductName]    nvarchar(40) NOT NULL,
[EnglishName]    nvarchar(40) NULL,
[QuantityPerUnit] nvarchar(20) NULL,
[UnitPrice]       money NULL,
[UnitsInStock]    smallint NULL,
[UnitsOnOrder]    smallint NULL,
[ReorderLevel]   smallint NULL,
[Discontinued]  bit NOT NULL,
Primary Key (ProductID),
Foreign Key (SupplierID) References tblSupplier
);

CREATE TABLE [dbo].[tblOrders] (
[OrderID]         int NOT NULL,
[CustomerID]     nvarchar(5) NOT NULL,
[EmployeeID]      int NULL,
[ShipName]        nvarchar(40) NULL,
[ShipAddress]     nvarchar(60) NULL,
[ShipCity]        nvarchar(15) NULL,
[ShipRegion]      nvarchar(15) NULL,
[ShipPostalCode]  nvarchar(10) NULL,
[ShipCountry]     nvarchar(15) NULL,
[ShipVia]         int NULL,

```

```

[OrderDate]           smalldatetime NULL,
[RequiredDate]        smalldatetime NULL,
[ShippedDate]         smalldatetime NULL,
[Freight]             money NULL
Primary Key (OrderID),
Foreign Key (CustomerID) References tblCustomers,
Foreign Key (ShipVia) References tblShippers,
Constraint valid_ShipDate CHECK (ShippedDate > OrderDate)
);
CREATE TABLE [dbo].[tblOrderDetails] (
[OrderID]      int NOT NULL,
[ProductID]    int NOT NULL,
[UnitPrice]    money NOT NULL,
[Quantity]     smallint NOT NULL,
[Discount]     real NOT NULL,
Primary Key (OrderID, ProductID),
Foreign Key (OrderID) References tblOrders,
Foreign Key (ProductID) References tblProducts,
Constraint Valid_Qty Check (Quantity > 0)
);
Go

```

## 16.2 الجزء الثاني: إنشاء عبارات لغة SQL

اعرض قائمة العملاء customers والطلبات orders المنشأة خلال عام 2014. أظهر الحقول order ID و order date و date ordered.

```

Use Orders
Go
SELECT CompanyName, OrderID, RequiredDate as 'order date', OrderDate
as 'date ordered'
FROM tblcustomers JOIN tblOrders on tblOrders.CustomerID =
tblCustomers.CustomerID
WHERE Year(OrderDate) = 2014

```

أضف حقلًّا جديداً (نشطاً) في جدول tblCustomer باستخدام عبارة ALTER TABLE، حيث تكون قيمته الافتراضية True.

```
ALTER TABLE tblCustomers
ADD Active bit DEFAULT ('True')
```

اعرض جميع الطلبات التي جرى شراؤها قبل 1 سبتمبر 2012 (اعرض الحقول company name و date ordered وكلفة الطلب الإجمالية (بما في ذلك تكلفة الشحن freight).

```
SELECT tblOrders.OrderID, OrderDate as 'Date Ordered',
sum(unitprice*quantity*(1-discount))+ freight as 'Total Cost'
FROM tblOrderDetails join tblOrders on tblOrders.orderID =
tblOrderDetails.OrderID
WHERE OrderDate < 'September 1, 2012'
GROUP BY tblOrders.OrderID, freight, OrderDate
```

اعرض جميع الطلبات المشحونة عبر شركة Federal Shipping (اعرض الحقول OrderID و ShipName و CustomerID و ShipAddress).

```
SELECT OrderID, ShipName, ShipAddress, CustomerID
FROM tblOrders join tblShippers on tblOrders.ShipVia =
tblShippers.ShipperID
WHERE CompanyName= 'Federal Shipping'
```

اعرض جميع العملاء الذين لم يشتروا في عام 2011.

```
SELECT CompanyName
FROM tblCustomers
WHERE CustomerID not in
(  SELECT CustomerID
FROM  tblOrders
WHERE Year(OrderDate) = 2011
)
```

اعرض جميع المنتجات التي لم تُطلب أبداً.

```
SELECT ProductID from tblProducts
Except
SELECT ProductID from tblOrderDetails
```

أو يمكن حل ذلك بالشكل التالي:

```
SELECT Products.ProductID,Products.ProductName
FROM Products LEFT JOIN [Order Details]
ON Products.ProductID = [Order Details].ProductID
WHERE [Order Details].OrderID IS NULL
```

اعرض معرفات الطلبات OrderID للزبائن الذين يقيمون في لندن باستخدام استعلام فرعي (اعرض الحقول OrderID و CustomerName و CustomerID)

```
SELECT Customers.CompanyName,Customers.CustomerID,OrderID
FROM Orders
LEFT JOIN Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Customers.CompanyName IN
(SELECT CompanyName
FROM Customers
WHERE City = 'London')
```

اعرض المنتجات التي يوفرها المورد A والمورد B (اعرض الحقول product name و supplier name)

```
SELECT ProductName, Name
FROM tblProducts JOIN tblSupplier on tblProducts.SupplierID =
tblSupplier.SupplierID
WHERE Name Like 'Supplier A' or Name Like 'Supplier B'
```

اعرض جميع المنتجات التي تأتي ضمن صناديق (اعرض الحقول product name و QuantityPerUnit)

```
SELECT EnglishName, ProductName, QuantityPerUnit
FROM tblProducts
WHERE QuantityPerUnit like '%box%'
ORDER BY EnglishName
```

## 16.3 الجزء الثالث: الإدخال والتعديل والحذف والفهارس

أنشئ جدول الموظفين Employee. يجب أن يكون المفتاح الرئيسي هو معرف الموظف EmployeeID. وهو حقل ترقيم تلقائي autonumber. أضف الحقول التالية:

LastName •

FirstName •

Address •  
City •  
Province •  
Postalcode •  
Phone •  
Salary •

استخدم عبارة إنشاء جدول `CREATE TABLE` وعبارات إدخال `INSERT` خمسة موظفين. ضم جدول `employee` إلى الجدول `tblorders`. اعرض السكريبت لإنشاء الجدول وإعداد القيود وإضافة الموظفين.

Use Orders

```

CREATE TABLE [dbo].[tblEmployee](
EmployeeID Int IDENTITY NOT NULL ,
FirstName varchar (20) NOT NULL,
LastName varchar (20) NOT NULL,
Address varchar (50),
City varchar(20), Province varchar (50),
PostalCode char(6),
Phone char (10),
Salary Money NOT NULL,
Primary Key (EmployeeID)
Go
INSERT into tblEmployees
Values ('Jim', 'Smith', '123 Fake', 'Terrace', 'BC', 'V8G5J6',
'2506155989', '20.12'),
('Jimmy', 'Smithy', '124 Fake', 'Terrace', 'BC', 'V8G5J7',
'2506155984', '21.12'),
('John', 'Smore', '13 Fake', 'Terrace', 'BC', 'V4G5J6', '2506115989',
'19.12'),
('Jay', 'Sith', '12 Fake', 'Terrace', 'BC', 'V8G4J6', '2506155939',
'25.12'),
('Jig', 'Mith', '23 Fake', 'Terrace', 'BC', 'V8G5J5', '2506455989',
'18.12');
Go

```

أضف حقلًّا يسمى `Totalsales` إلى جدول `Tblorders`. استخدم تعليمات لغة DDL وعبارة `ALTER TABLE`.

```
ALTER TABLE tblOrders
ADD Foreign Key (EmployeeID) references tblEmployees (EmployeeID)
```

استخدم عبارة `UPDATE` لإضافة مجموع مبيعات كل طلب بناءً على جدول `order details`.

```
UPDATE tblOrders
Set TotalSales = (select sum(unitprice*quantity*(1-discount))
FROM tblOrderDetails
WHERE tblOrderDetails.OrderID= tblOrders.OrderID
GROUP BY OrderID)
```

# من إصدارات أكاديمية حسوب

