تصميم مخطط كيان ارتباط ومدخل إلى الجبر العلاقاتي

د. مادلين عبود

13/04/2023

RB Informatics;

قواعد معطيات 1

مقدمة:

تحدثنا بالمحاضرة السابقة عن الارتباطات وأنواعها وعن التخصيص والتعميم والتجميع. وسنتابع حديثنا عن مخطط الكيانات والارتباطات، لندخل في الجبر العلاقاتي

↙ مثال عن كيفية تصميم مخطط كيانات ارتباطات ERD

I am the manager of a training company that provides instructor courses in management techniques.
We teach courses, each of which has a code and a name.
Introduction to UNIX and C programming (material) are two of our most popular courses.
Courses vary in length from one to four days. Said and Diana are two of our best teachers.
We need each instructor's name and phone number. The students can take several courses over time, and many do this we like to have each student's name and phone number.

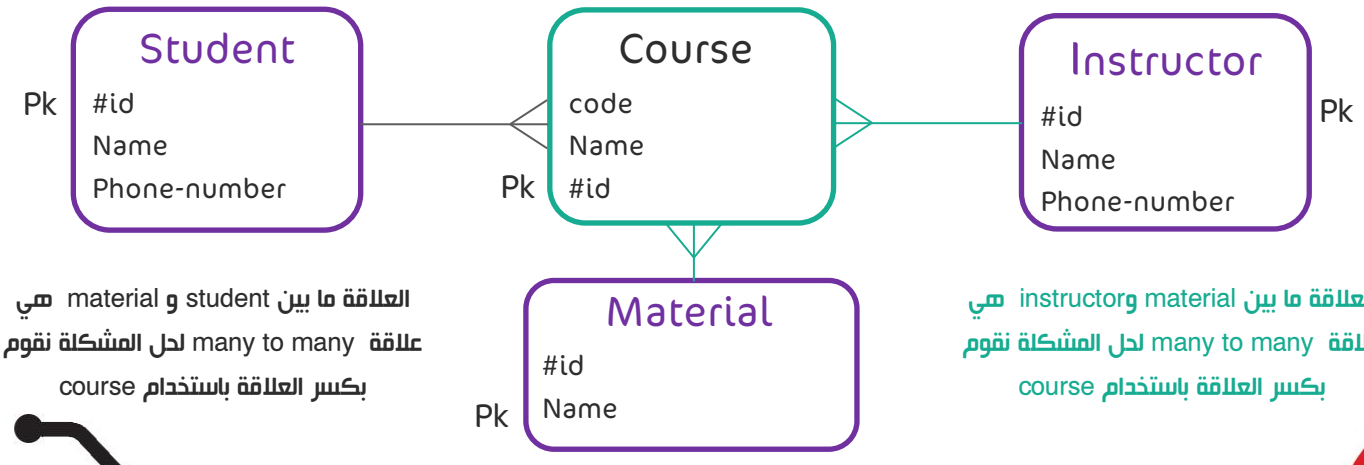
كيان

واصفات

لدينا طريقتين لتصميم المخطط

Chen notation Vs Crow's foot notation

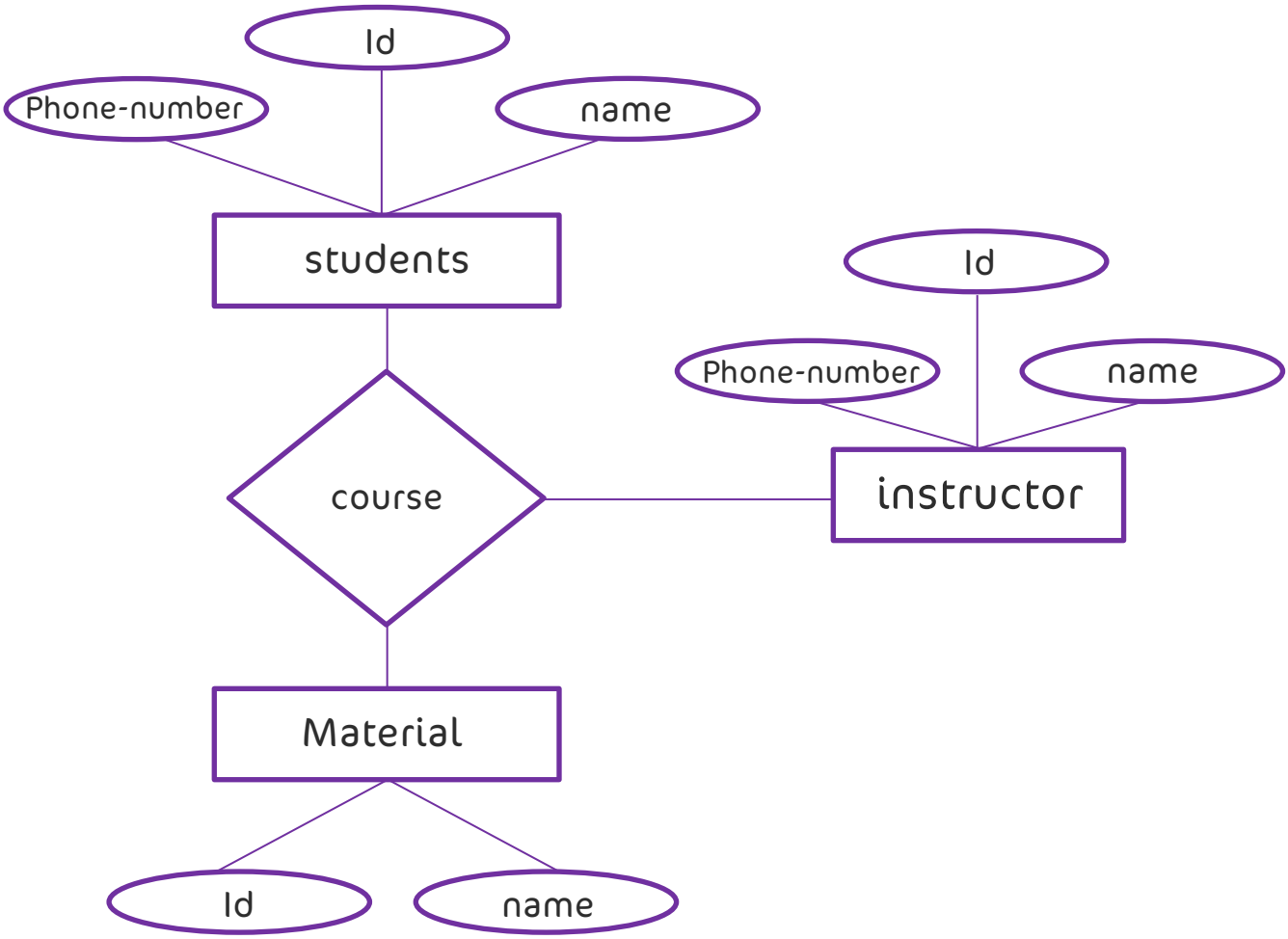
Crow's foot notation



العلاقة ما بين student و material هي علاقة many to many لحل المشكلة نقوم بكسر العلاقة باستخدام course

العلاقة ما بين instructorg material هي علاقة many to many لحل المشكلة نقوم بكسر العلاقة باستخدام course

Chen notation



أضف إلى معلوماتك:

- المخططات والحالات: تبدل المعطيات المخزنة في قاعدة المعطيات مع الزمن بسبب حركة المعلومات المضافة والمحذوفة منها وإليها.
ونسمي حالة قاعدة المعطيات Instance of the Database مجموعة المعلومات المخزنة في قاعدة المعطيات في لحظة معينة.
كما نسمي مخطط قاعدة المعطيات Database schema القالب العام لقاعدة المعطيات.
- استقلال المعطيات: يقصد باستقلال المعطيات إمكان تعديل تعريف مخطط من مخططات قاعدة المعطيات دون أن يؤثر هذا التعديل في تعريف المخطط من المستوى الأعلى في القاعدة.

Don't compare your life to others. There's no comparison between the sun and the moon, they shine when it's their time.

ثمة نوعان من استقلال المعطيات:

1. استقلال المعطيات فيزيائياً:

ويعني أن تعديل المخطط الفيزيائي (طريقة التخزين الفعلية وطرق الوصول إلى المعطيات) لا يتطلب إعادة كتابة البرامج التي تتعامل مع قاعدة المعطيات، والجدير بالذكر أننا نادراً ما نجري تعديلات في المستوى الفيزيائي، إلا في حالات معينة ويقصد تحسين الأداء.

2. استقلال المعطيات منطقياً:

ويعني إمكانية تعديل المخطط المفاهيمي لقاعدة المعطيات دون أن يتطلب ذلك إعادة كتابة البرامج التطبيقية، ويجري التعديل المخطط المفاهيمي عادةً عند تعديل البنية المنطقية لقاعدة المعطيات.

إن تحقيق الاستقلال المنطقي أصعب بكثير من تحقيق الاستقلال الفيزيائي وذلك لاعتماد البرامج التطبيقية إلى حد بعيد على البنية المنطقية للمعطيات المراد الوصول إليها.

Case method conventions (اتفاقيات طريقة الحالة)

1. Entity relationship modelling conventions

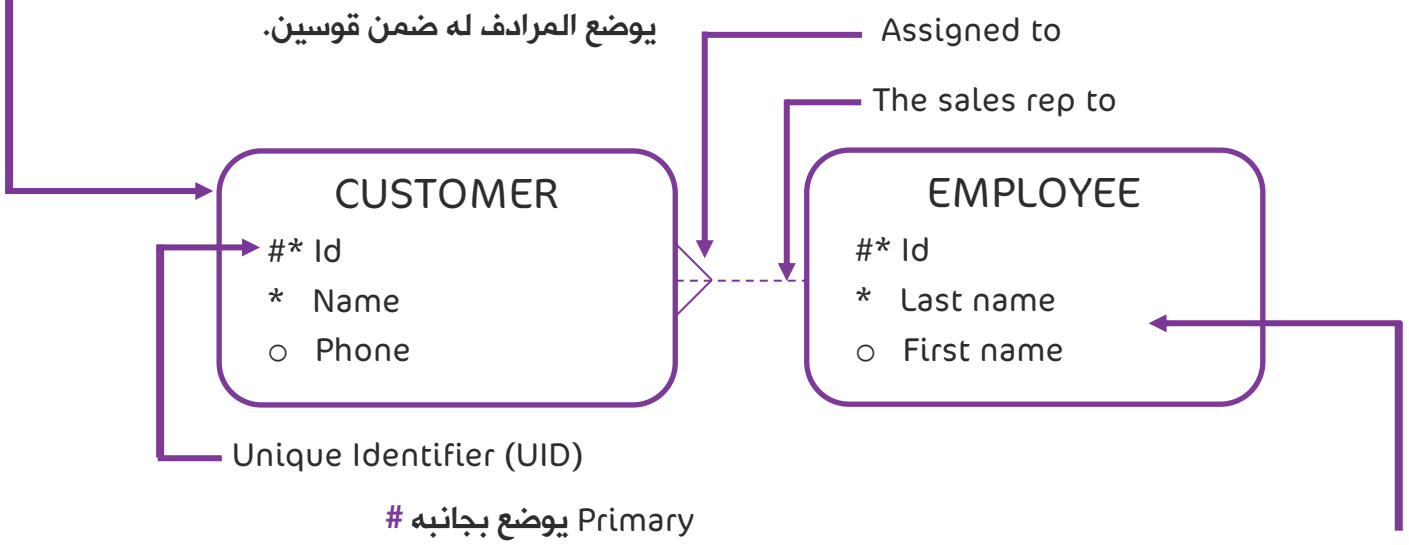
Entity:

شكله مستطيل دون زوايا قائمة (ناعم الحواف).

وحيد الاسم ومميز غير قابل للتكرار.

يكتب بأحرف كبيرة.

يوضع المرادف له ضمن قوسين.



Primary يوضع بجانبه #

Secondary يوضع بجانبه (#)

Attribute:

اسمها مفرد ويكتب بأحرف صغيرة.

اذا كانت الواصفة إلزامية يوضع بجانبها *

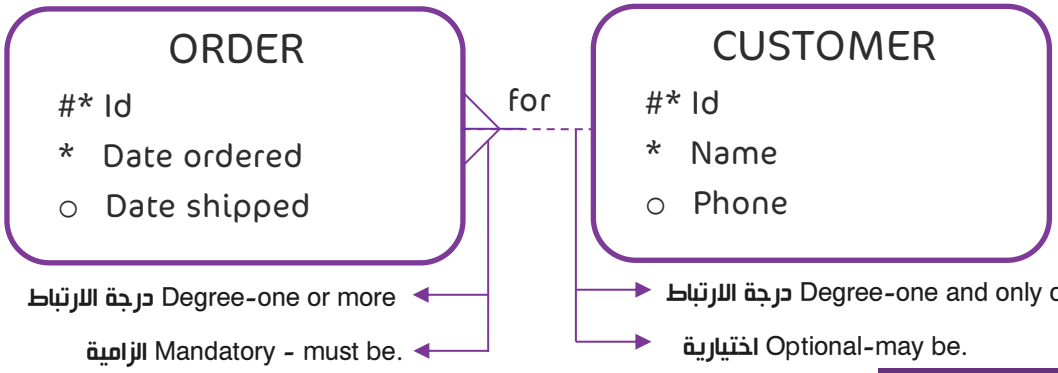
أما إذا كانت اختيارية يوضع بجانبها ○

مثال: ↙

Syntax

Each source entity {may be\ must be}.
Relationship name {one and only\one or more} Destination entity

EX
- Each ORDER must be for one and only one CUSTOMER
- Each CUSTOMER may be the client for one or more ORDERs

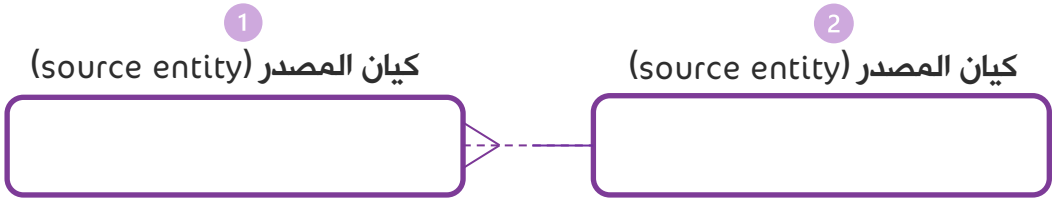


توضيح لما سبق:

قمنا برسم (تصميم) كل كيان مصدري.

(Must be) نعبر عنها بخط غامق أما (maybe) بـخط متقطع.

رسمنا علاقة الربط والتي هي من درجة many من طرف الكيان المصدري 1 (one or more) ومن درجة one من طرف الكيان المصدري 2 (one and only one).

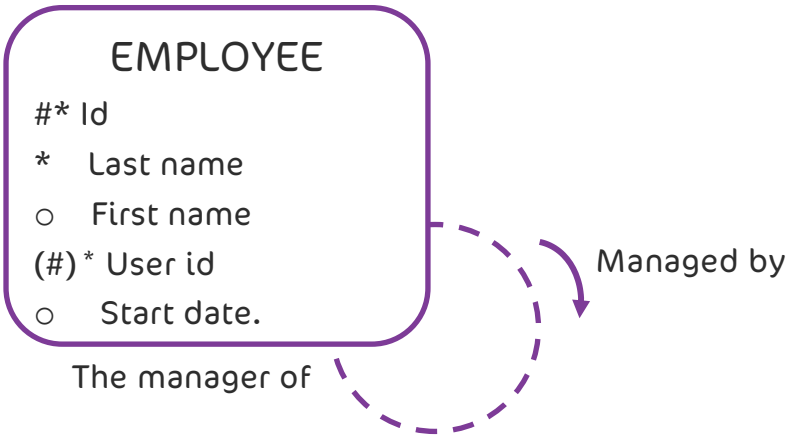


Relationship types

One To One	من درجة واحد-واحد (one and only one) في كلا الطرفين (الاتجاهين). علاقة نادرة.	مثال عليها: العلاقة بين الزوج والزوجة.
Many To One	من درجة واحد-كثير (one or more in one direction, one and only one in the other direction) علاقة شائعة جداً.	مثال عليها: العلاقة بين الركاب والطائرة.
Many To Many	من درجة كثير-كثير (one or more) في كلا الطرفين (الاتجاهين). يستخدم مع كيانات التقاطع.	مثال عليها: العلاقة بين الموظفين والمهارات.

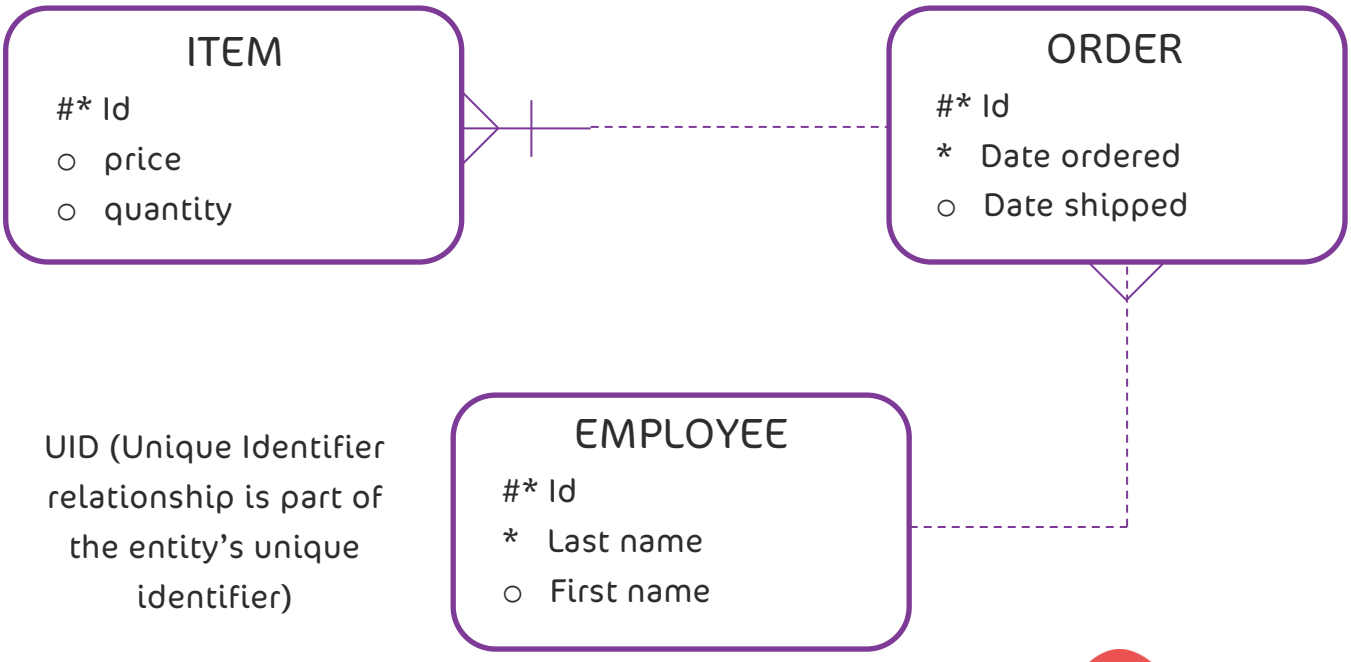
2. recursive relationship

تسمى العلاقة بين الكيان ونفسه بالعلاقة العودية recursive relationship وتمثل باستخدام "pig's ear"



3. weak entity sets

في مخطط الكيانات والارتباطات الممثل بطريقة crow's نضع خط ( ) للدلالة على الكيان الضعيف.

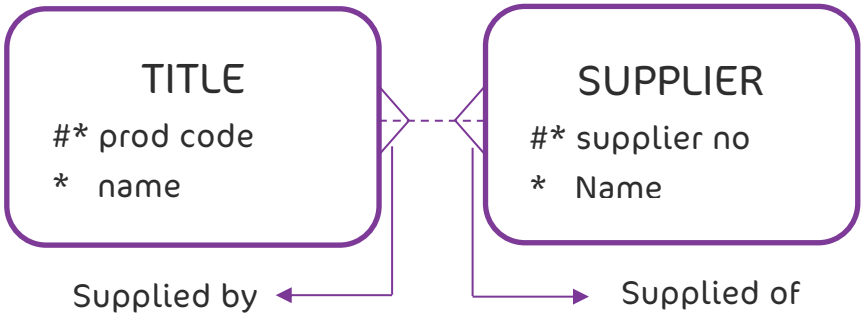


Everything you lose is a step you take.

Entity Relationship Diagram (ERD)

Identifying the problem in many-to-many relationship:

ليكن لدينا المخطط الـERD التالي ذو درجة الربط many-to-many



لنتناقش معاً:

هل يمكننا معرفة أي مزود يزود الدار البيضاء؟!

في أي كيان يمكننا تخزين واصفة purchase price؟!

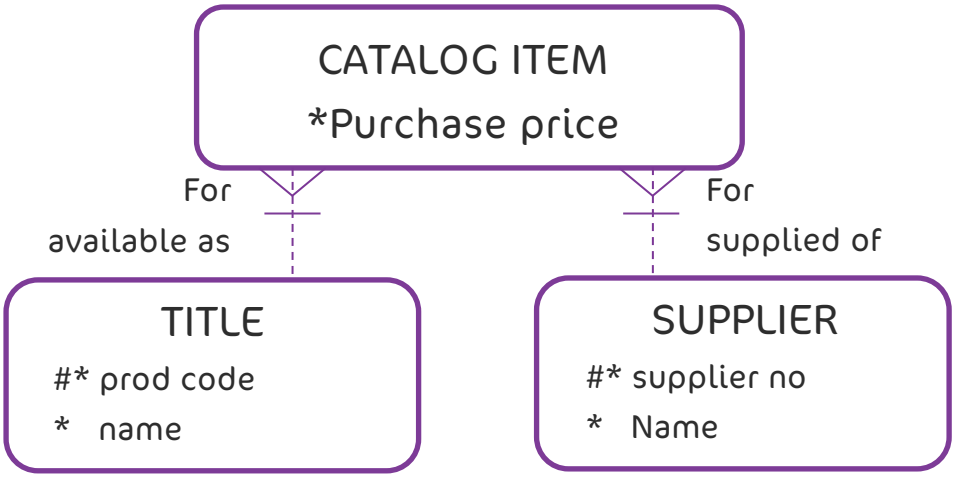
لكن نلاحظ أن هذه الواصفات لها صلة وعلاقة أكبر بعلاقة الارتباط بين هذه البيانات وليست ذات صلة مباشرة بهذه الكيانات، مثل واصفة (purchase price) بحيث إن سعر الشراء معتمد على العلاقة ما بين المزود والمنتج المراد شراؤه والذي يمكن أن يختلف من مزود لآخر ومن منتج لآخر وحسب الكمية وغيرها...

عندما تصف الواصفة العلاقة ما بين الكيانين نقوم بإنشاء كيان تقاطع ونضع عنده هذه الواصفة intersection entity.

ولحل مشكلة المثال السابق

نقوم بإنشاء كيان تقاطع intersection entity ونسميه الـCatalog item ونضع عنده الواصفة السابقة purchase price

كالآتي:



النموذج العلاقاتي Relational Model

البنى الأساسية

بنية قواعد المعطيات العلاقاتية

تتألف قاعدة المعطيات من مجموعة من الجداول لكل منها اسم وحيد مميز يتألف كل جدول بدوره من مجموعة من الأعمدة وعدد من الأسطر.

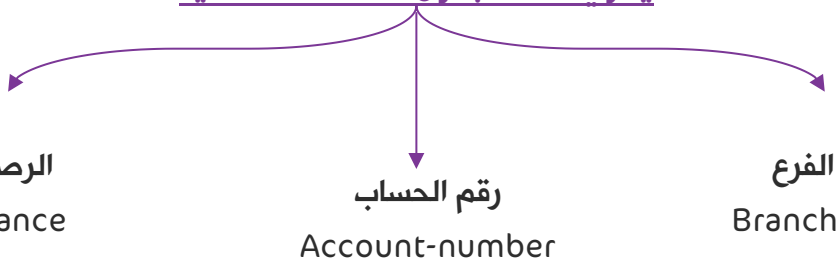
يمثل كل سطر من الجدول علاقة تربط مجموعة من القيم، لما كان الجدول يتألف من مجموعة من هذه الارتباطات، فهنالك تشابه شديد بين مفهوم الجدول ومفهوم العلاقة الرياضية ومن هنا أخذ النموذج العلاقاتي اسمه.

مثال لعلاقة:

Account-number	Branch-name	balance
A-101	Downtown	500
A-102	Perry ridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

يبين الشكل السابق جدول الحسابات account

يحوي هذا الجدول ثلاثة أعمدة هي:



وفق المصطلحات النموذج العلاقاتي تسمى عناوين الأعمدة واصفات attributes ، لكل واصف مجموعة من القيم تسمى مجال domain الواصف، فمثلاً مجال الواصف "اسم الفرع" هو مجموعة كل أسماء الفروع، ولنرمز إلى هذه المجموعة بـ D_1 ولنرمز بـ D_2 إلى مجال الواصف account-number وبـ D_3 إلى مجال الواصف balance.

إن لكل سطر من الجدول account يتألف من ثلاثية (V_1, V_2, V_3) حيث V_1 هو اسم الفرع وينتمي إلى المجال D_1 و V_2 هو رقم الحساب وينتمي إلى المجال D_2 و V_3 هو الرصيد وينتمي إلى المجال D_3

في الحالة العامة سوف يحوي جدول account مجموعة جزئية من الجداء $D_1 \times D_2 \times D_3 \longleftarrow$

وعموماً فإن جدولاً يحوي n واصفاً يجب أن يكون مجموعة جزئية من الجداء $D_1 \times D_2 \times D_3 \times \ldots \times D_n$

يعرف الرياضيون العلاقة بأنها مجموعة جزئية من الجداء الديكارتي لمجموعة من المجالات.

■ ملاحظة:

يتوافق هذا التعريف غالباً مع تعريفنا للجدول.

الفرق هو فقط في أننا نعطي أسماء للواصفات، على حين يربط الرياضيون أرقاماً بالأسماء، فيستخدمون الرقم (1) للواصف الذي يظهر مجاله أولاً، والرقم (2) للواصف الذي يظهر مجاله ثانياً وهكذا، وكما أن الجداول تعد من العلاقات فإننا سوف نستخدم المصطلحات الرياضية (علاقة، حدودية) بدلاً من المصطلحات (جدول، سطر).

ويكتب التعريف الرياضي كالآتي:

$$D_1, D_2, D_3, \ldots, D_n \ \ a\ relation\ r\ is\ a\ subset\ of\ D_1 \times D_2 \times D_3 \times \ldots \times D_n$$
$$thus\ a\ relation\ is\ a\ set\ of\ n-tuples$$
$$(a_1, a_2, \ldots, a_n)\ where\ a_i \in D_1$$

↙ مثال:

Customer-name={Jones, Smith, Curry, Lindsay }

Customer-street= {Main, North, Park}

Customer-city= {Harrison, Rye, Pittsfield}

Then

r = {(Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield)}

is a relation over customer-name × customer-street × customer-city

أنماط الواصفات:

- لكل واصف اسم مميز على مستوى العلاقة.
- لكل واصف مجال يحوي مجموعة القيم المسموحة للواصف.
- القيمة المعدومة null تنتمي إلى كل المجالات.

مخطط العلاقة Relation Schema

$R = (A_1, A_2, \dots, A_n)$ هي مخطط العلاقة

$A_n, \dots, A_2, A_1$ هي واصفات

Customer-Schema = (customer-name, customer-street, customer-city).

r(R) هي علاقة على مخطط العلاقة R

E.g: customer (Customer-Schema).

للتوضيح أكثر:

يجب علينا التمييز بين مخطط قاعدة المعطيات database schema أو التصميم المنطقي لقاعدة المعطيات وحالة قاعدة المعطيات database instance التي هي صورة لمعطيات قاعدة المعطيات في لحظة محددة.

يشابه مفهوم العلاقة مفهوم المتحولات في لغات البرمجة، على حين يشابه مفهوم مخطط العلاقة مفهوم تعريف الأنماط في لغات البرمجة.

يجب علينا معرفة ما يلي:

الاسم المعطى لمخطط العلاقة يبدأ بحرف كبير والاسم المعطى للعلاقة هو بأحرف صغيرة.

كالمثال السابق: customer (Customer-schema)

Customer-schema = (customer-name, customer-street, customer-city).

وبوجه عام نقول أن مخطط العلاقة يحوي قائمة من الواصفات ومجالاتها المعتبرة.

محتوى العلاقة Relation instance

يمكن تمثيل محتوى العلاقة في لحظة معينة بجدول. يمثل العنصر t من r بسطر في الجدول كالتالي:

Attributes

Tuples

customer name	customer street	customer city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

Customer

عند كتابة مثلاً (.... , المدينة , الاسم) لا يجوز فيما بعد وضع قيمة الاسم مكان المدينة: (.... , الاسم , المدينة)

وليس من الضرورة أن يوضع الPrimary key في أول حقل.

- الترتيب في العلاقة غير مهم
حيث يمكن أن يتم تخزين الTuples في ترتيب اعتباطي (عشوائي).

هذا أول مثال قمنا بذكره على تمثيل العلاقة.
نلاحظ أن تغيير ترتيب الـtuples لا يؤثر على تمثيل العلاقة.

account-number	branch-name	balance
A-101	Down Town	500
A-215	Mianus	700
A-102	Perry ridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Red wood	700
A-217	Brighton	750

قاعدة المعطيات

- تتألف قاعدة المعطيات من عدد من العلاقات، تحوي كل علاقة جزءاً من المعلومات حول المؤسسة.

مثال:

- account: يخزن معلومات عن الحسابات.
- depositor: يخزن معلومات حول العميل الذي يمتلك الحساب.
- customer: يخزن معلومات عن العملاء.

- تخزين جميع المعلومات كعلاقة واحدة. مثلاً البنك (account-number, balance, customer-name, ...)
- النتائج في تكرار المعلومات مثل (عميلان يملكان حساب واحد).
- الحاجة إلى Null Values مثل (عميل ليس له حساب).

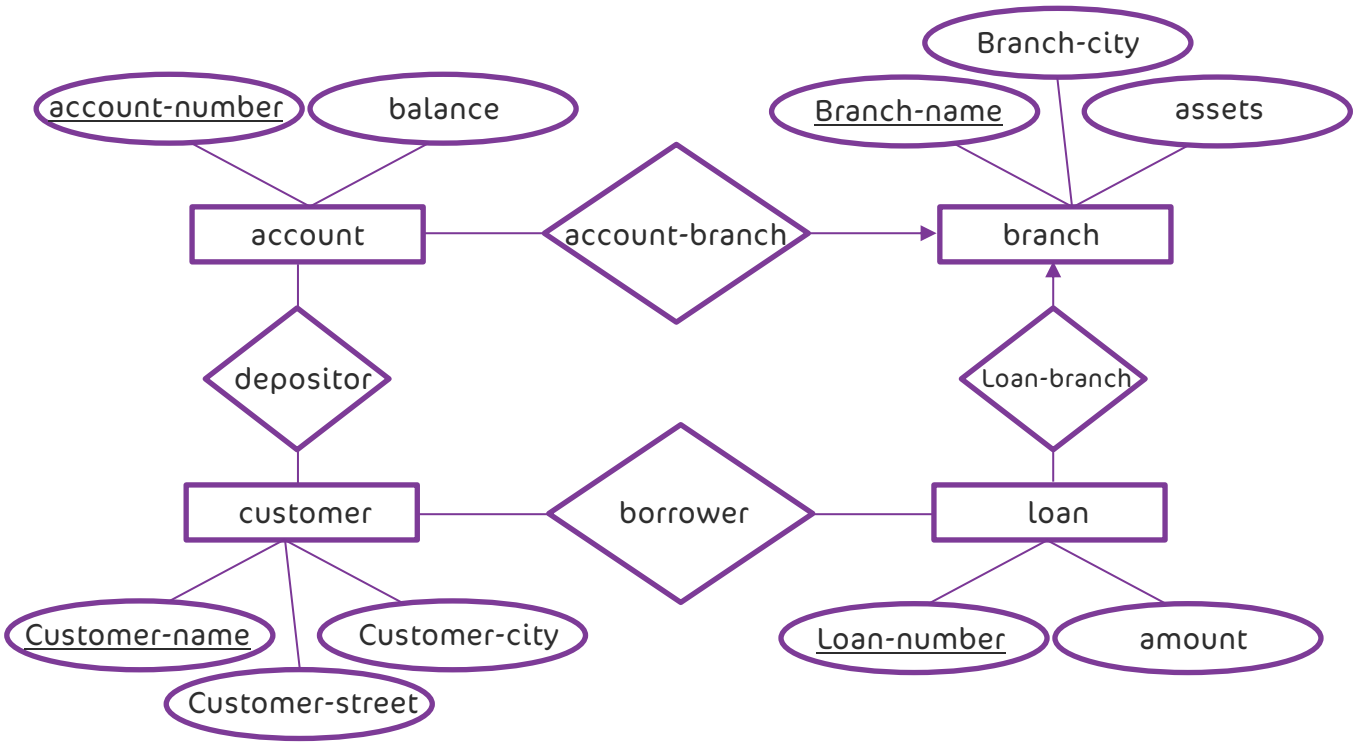
The Customer Relation:

Customer-name	Customer-street	Customer-city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

The Depositor Relation

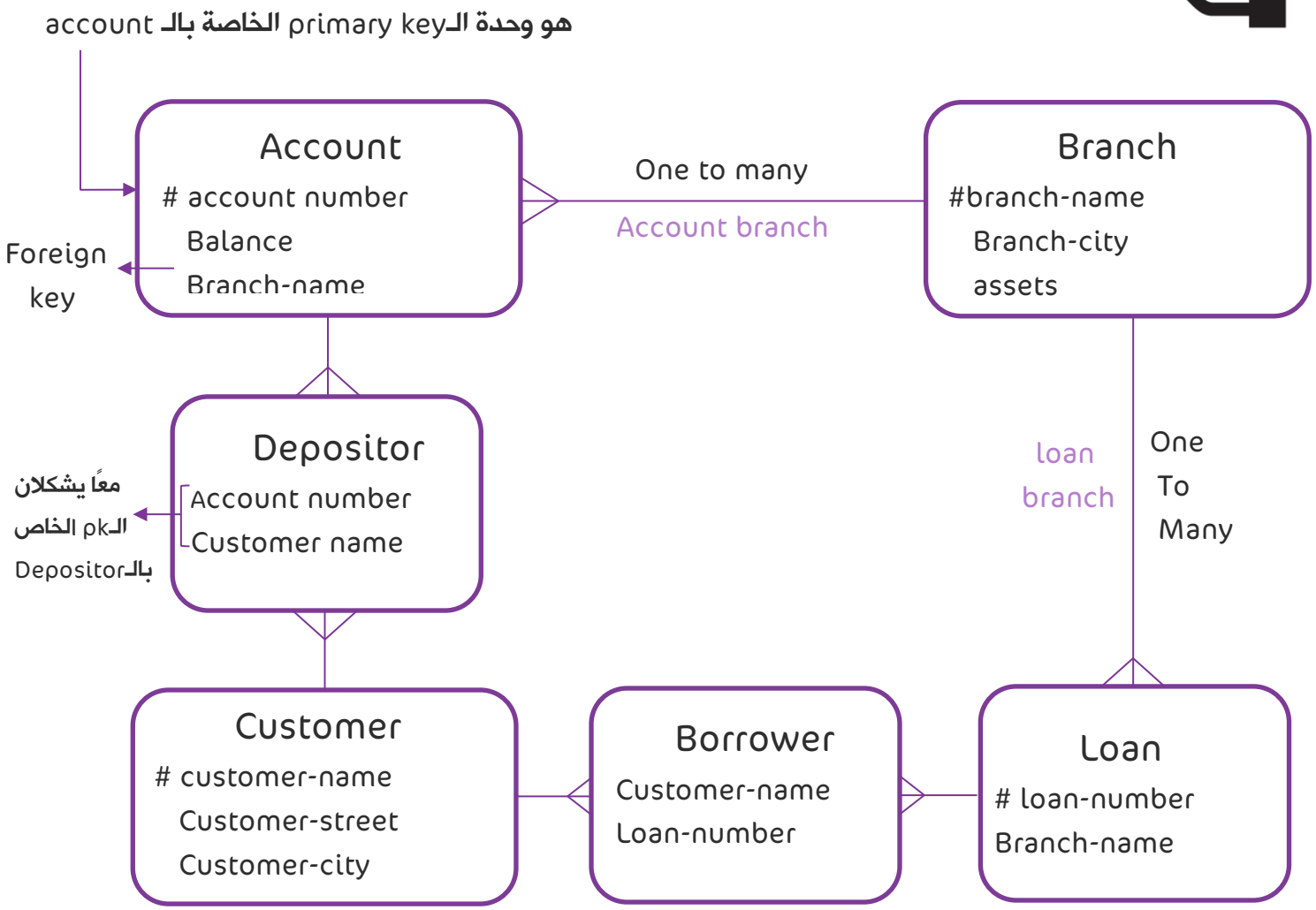
Customer-name	Account-number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindesay	A-222
Smith	A-215
Turner	A-305

ERD for The Banking Enterprise:



إن ما سبق هو ERD بطريقة Chen للتحويل إلى مخطط علاقاتي بطريقة crow's

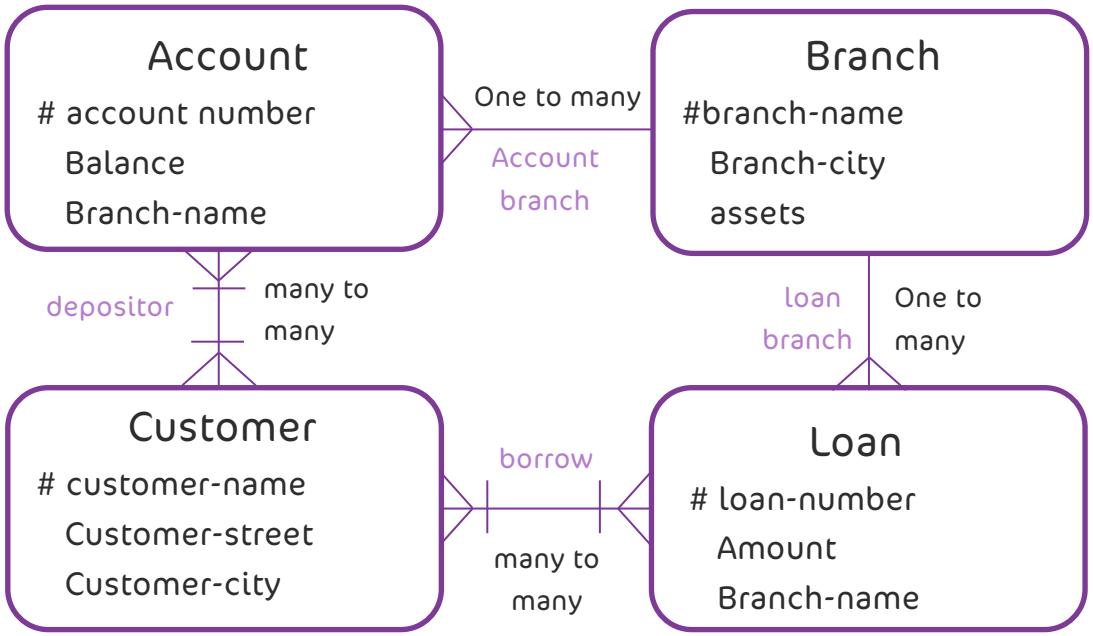
"Don't forget you've only read part of your story so far. You don't yet know how it ends."



إن الـDepositor هو كسر علاقة (many -to- many) الموجودة بين الكيانين account وcustomer.

عند تمثيل علاقة (one -to- many) نأخذ مفتاح الone ونضعه عند ال many.

ولكن التمثيل الأصح يكون شكله كالتالي:



البار (|) تدل على أن ال primary key للعلاقة بين الكائنين هي ال primary keys لكل منهما.

المفاتيح Keys

- لنفرض $K \subseteq R$

K هو super key من r إذا كانت قيم K كافية لتحديد Unique tuple من كل علاقة محتملة $r(R)$

من خلال " r الممكنة " فإننا نعني العلاقة التي يمكن أن توجد في المؤسسة التي ننظمها

Example:

{customer-name, customer-street} and {customer-name}

كلاهما super keys للـcustomer إذا لم يكن بإمكان اثنين أن يكون لهما نفس الاسم

بمعنى:

إذا كنا نعلم أنه لا يوجد لعميلان نفس الاسم في الشركة عندها يمكن اعتبار customer-name هو super key للـ customer.

أما في حال يمكن أنّ يوجد لعميلين أو أكثر نفس الاسم في الشركة عندها لا يمكننا اعتبار customer-name هو super key لوحده وإنما أصبحنا بحاجة واصفة أخرى للتمييز بين عميل وآخر مثل أخذنا ل customer-street مع customer-name يشكلان معاً ال super key لل customer.

- K هو Candidate key إذا كان K في الحد الأنى.

Example:

{customer-name}

هو candidate key للـ customer

نظراً لأنها super key { على افتراض أنه لا يمكن أن يكون لدى اثنين من العميل نفس الاسم } وليس هناك مجموعة فرعية منه تعتبر super key.

(تم شرح المفاتيح بشكل أكبر في المحاضرة السابقة)

تذكرة:

تحديد المفاتيح من مخطط ERD

الكيان القوي:	المفتاح الأولي للكيان القوي يصبح مفتاحاً أولياً في العلاقة.
الكيان الضعيف:	يتألف المفتاح الأولي للكيان الضعيف من اجتماع المفتاح الأولي للكيان القوي ومميز للكيان الضعيف.
الارتباطات:	يصبح اجتماع المفاتيح الأولية للكيانات المرتبطة مفتاحاً أعلى.

في حالة ارتباط ثنائي من نوع واحد-لكثير	يصبح المفتاح الأولي للكيان "كثير " مفتاحًا أوليًا للعلاقة.
في حالة ارتباط ثنائي من نوع واحد-لواحد	يمكن اختيار مفتاح أحد الكيانين ليصبح المفتاح الأولي للعلاقة.

لغات الاستفسار Query Languages

هي اللغات التي يعبر فيها المستخدم عن المعلومات المطلوب استرجاعها من قاعدة المعطيات.

أنواع اللغات:

- اللغات الإجرائية Procedural
- اللغات غير الإجرائية non-procedural

اللغات العلاقاتية الصافية "pure" Relational Languages:

- الجبر العلاقاتي Relational Algebra
- الحساب العلاقاتي Tuple Relational Calculus

▪ الجبر العلاقاتي Relational Algebra

هو لغة استعلام إجرائية.

يتألف من مجموعة من العمليات التي تأخذ علاقة أو اثنين كدخل وتعطي علاقة جديدة كخرج.

العمليات الأساسية في الجبر العلاقاتي:

- الاختيار Select
- الاسقاط project
- الاجتماع union
- الفرق set difference
- الجداء الديكارتي cartesian product
- إعادة التسمية rename

Not All Storms Come To Disrupt Your Life,
Some Come To Clear Your Path.

سوف نعرض فيما يلي العمليات الأساسية في الجبر العلاقاتي موضحين طريقة عملها بأمثلة على قاعدة المعطيات المتعلقة بالمصارف والمعرفة بالمخطط العلاقاتي التالي:

Loan-Schema (loan- number, amount, branch-name)
Customer-Schema (customer-name, customer-street, customer-city)
Borrower-Schema (customer-name, loan-number)
Employee-Schema (employee-name, phone-number)
Loan-Officer-Schema (banker-name, customer-name, loan-number)
Depositor-Schema (customer-name, account-number)
Account-Schema (account-number, balance, branch-name)
Branch-Schema (branch-name, branch-city, assets)

■ ملاحظة:

تسمى العمليات (اختيار, اسقاط, إعادة تسمية) عمليات أحادية لأنها تجرى على علاقة واحدة أما العمليات الثلاث الباقية هي ثنائية لأنها تجرى على زوج من العلاقات.

■ الاختيار Select Operation:

$Relation\ r$

A	B	C	D
α	α	1	**7**
α	β	5	**7**
β	β	12	**3**
β	β	23	**10**

$\sigma_{A=B \wedge D>5}(r)$

A	B	C	D
α	α	1	**7**
β	β	23	**10**

الرمز: $\sigma_{p(r)}$

p: اختيار مستند

تعرف كالتالي: $\sigma_{p(r)} = \{t | t \in r \ and \ p(t)\}$

نقوم باختيار مجموعة من الحدوديات التي تحقق شرطاً معيناً من العلاقة.

سنرمز إلى العملية كما يلي: $\sigma_{condition}(relation)$

حيث condition هو شرط يجب أن تحققه الحدوديات المختارة.

↙ مثال:

لنأخذ مخطط العلاقة التالية:

Loan-Schema (branch-name, loan-number, amount)

لاختيار مجموعة الحدوديات القروض للفرع branch-name = "perryridge"

نكتب العبارة التالية:

$$\sigma_{branch-name="perryridge"}(loan)$$

يمكن أن يحوي الشرط المطبق في عملية الاختيار عمليات مقارنة: $=, <, >, \leq, \geq$ ومعاملات منطقية not, and, or

ويمكن أن تطبق هذه المعاملات بين قيم الواصفات المكونة للعلاقة.

↙ مثال:

لاختيار القروض التي منحها الفرع branch-name = "perry ridge" والتي لا تقل عن 1200 نكتب:

$$\sigma_{branch-name=\text{perryridge} \wedge amount\ \Phi\ 1200}(loan)$$

لاختيار مجموعة الزبائن الذين يحملون نفس اسم المسؤول عن القرض الذي اقترضوه من علاقة loan-officer ذات المخطط التالي:

loan-officer (customer-name, banker-name, loan-number)

نكتب:

$$.\sigma_{customer-name=banker-name}(loan-officer)$$

- الإسقاط Project Operation:

$Relation\ r$

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A'	C'
α	1
α	1
β	1
β	2

A	C
α	1
β	1
β	2

وهي عملية وحيدة المعامل تسمح بانتظار بعض الواصفات من العلاقة. نرمز إلى هذه العملية بالشكل:

$$\Pi_{selected\ attributes}(relation)$$

الرمز: $\Pi_{A_1,A_2,\ldots,A_k}(r)$

حيث $A_1, A_2, \ldots, A_k$ هي واصفات و r هي اسم العلاقة.

يتم تعريف النتيجة على أنها اسم علاقة الأعمدة k التي تم الحصول عليها عن طريق محو الأعمدة غير المدرجة.

يتم إزالة الصفوف المكررة من النتيجة نظراً لأن العلاقات مجموعات.

↙ مثال:

لنفترض أننا نريد الحصول على أرقام القروض ومبالغها دون أن نهتم بالأسماء للأفرع.

يكتب الاستعلام السابق بالشكل:

$$\Pi_{loan-number,amount}(loan)$$

وبفرض أن علاقة القروض loan ممثلة بالجدول التالي:

loan-number	branch-number	amount

تكون العلاقة الناتجة من عملية الإسقاط من الشكل:

loan-number	amount

- الاجتماع Union Operation:

Relations r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$

A	B
α	**1**
α	**2**
β	**1**
β	**3**

الرمز : $r \cup s$

:defined as

$$r \cup s = \{t \mid t \in r \; or \; t \in s\}$$

نرمز إلى العملية ب ∪ وتجرى هذه العملية بين علاقتين متجانستين فنقول أنه يمكننا القيام بالعملية $r \cup s$ إذا تحقق الشرطان التاليان:

- **عدد الواصفات في العلاقين (s, r) هو نفسه.**
- **مجال الواصفة رقم i في r هو نفسه مجال الواصفة رقم i في s.**

يتعامل العمود الثاني من r مع نفس النوع من القيم كما يفعل العمود الثاني من s ويمكن بالطبع أن تكون العلاقتان (s, r) ناتجتين عن تعبير في جبر علاقاتي.

↙ مثال:

للحصول على جميع الزبائن الذين يتعاملون مع المصرف (الذين لديهم حساب أو اقترضوا قرضاً أو للحصول على كلا الفريقين)

نحتاج إلى إيجاد مجموعة الزبائن الذين لديهم حساب في المصرف وهي معلومات موجودة في العلاقة depositor، وإلى إيجاد مجموعة الزبائن الذين اقترضوا من المصرف وهي معلومات موجودة في علاقة borrower، ثم إلى إجراء عملية الاجتماع بين المجموعتين.

ومن ثم يكون التعبير الناتج:

$$\prod_{customer-name}(depositor) \cup \prod_{customer-name}(borrower).$$

"One day you will tell your story of how you overcame what you went through and it will be someone else's survival guide."

▪ الفرق Set difference Operation:

Relations r, s:

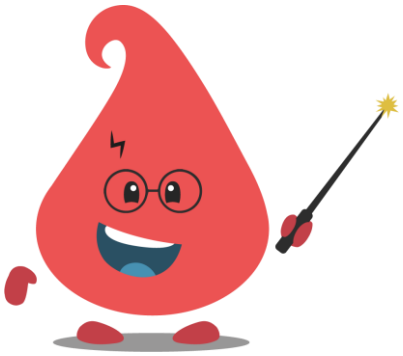
A	B
α	**1**
α	**2**
β	**1**

r

A	B
α	**2**
β	**3**

s

$r - s$



A	B
α	**1**
β	**1**

الرمز: $r - s$

defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

نرمز إلى العملية بـ " – " وتسمح بإيجاد الحدوديات التي تنتمي إلى علاقة الحد الأول ولا تنتمي إلى علاقة الحد الثاني.

↙ مثال:

لإيجاد مجموعة الزبائن الذين لديهم حساب مصرفي ولم يقترضوا من المصرف. نكتب:

$$\prod_{customer-name}(depositor) - \prod_{customer-name}(borrower)$$

وكما ذكرنا في عملية الاجتماع لكي تجري عملية الفرق بين علاقتين يجب أن يتحقق الشرطان المذكوران فيما سبق.

▪ الجداء الديكارتي Cartesian-Product Operation

Relations r, s:

A	B
α	**1**
β	**2**

r

C	D	E
α	10	***a***
β	10	***a***
β	20	***b***
γ	10	***b***

s

$r \times s$:

A	B	C	D	E
α	1	α	10	**a**
α	1	β	19	**a**
α	1	β	20	**b**
α	1	γ	10	**b**
β	2	α	10	**a**
β	2	β	10	**a**
β	2	β	20	**b**
β	2	γ	10	**b**

الرمز : $r \times s$

defined as:

$$r \times s = \{t\, q |\ t \in r \ and\ q \in s\}$$

نرمز إلى العملية ب $\times$ **وتسمح بتجميع معلومات من علاقتين ونفرض أن واصفات** $r(k)$ **و** $s(S)$ **مفككة أي ليس لها علاقة ببعضها. (هذا يعني** $R \cap S = \emptyset$**).**

إذا كانت هذه الواصفات $r(k)$ **و** $s(S)$ **ليست مفككة أي لها علاقة ببعضها إذا علينا القيام بإعادة التسمية** (remaining).

مثال:

ليكن لدينا R_1 **و** R_2 **بحيث** $R = R_1 \times R_2$

R

A	B	C	D
a_1	b_1	c_1	$\mathbf{d_1}$
a_1	b_1	c_2	$\mathbf{d_2}$
a_2	b_2	c_1	$\mathbf{d_1}$
a_2	b_2	c_1	$\mathbf{d_2}$

R_1

A	B
a_1	$\boldsymbol{b_1}$
a_2	$\boldsymbol{b_2}$

R_2

C	D
c_1	$\mathbf{d_1}$
c_2	$\mathbf{d_2}$

إذا أردنا الحصول على جميع الزبائن الذين اقترضوا من المصرف الفرع "Perryridge"

للحصول على هذه المعلومات نحتاج إلى المعلومات الموجودة في كلتا العلاقتين borrowerg loan **وتعطي عملية اختيار الحدوديات المتعلقة بالفرع المطلوب من جداء العلاقتين، معلومات عن الزبائن والقروض المأخوذة من الفرع المطلوب، ولكن ليست المعلومات المطلوبة، وللحصول على المعلومات المطلوبة نكتب:**

$$\Pi_{customer-name}(\sigma_{borrower.loan-number=loan.loan-number}(\sigma_{branch-name="perryridge"}(borrower \times loan)))$$

تركيب العمليات العلاقاتية Composition of operation:

إن نتيجة العمليات العلاقاتية هي علاقة.

هذا مايسمح لهذه العمليات أن تجتمع لتؤلف عبارات الجبر العلاقاتي.

$r \times s$:

A	B	C	D	Σ
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	r	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	r	10	b

Example:

$\sigma_{A=C}\ (r \times s)$.

A	B	C	D	Σ
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

مثال:

لإيجاد أسماء الزبائن الذين يعيشون في مدينة "Horrison" نكتب:

$$\Pi_{customer-name}\left(\sigma_{customer-city=\text{Horrison}}\ (customer)\right)$$

ونقصد بها تركيب عمليتين:

- اختيار الحدوديات التي تحقق الشرط (مدينة = Horrison)
- وإسقاط العلاقة الناتجة عن العملية السابقة على العمود $Customer\_name$

إعادة التسمية Rename operation:

- يسمح بإعطاء اسم جديد لناتج عملية مكتوبة بالجبر العلاقاتي.
- يسمح بإعطاء أكثر من اسم لعلاقة واحدة.

↙ مثال:

$$P_x(E)$$

يعيد ناتج العملية E باسم x.

في الحالة العامة:

إذا كانت العملية E تعيد علاقة فيها n واصف، فإن العملية:

$$P_{x(A_1,A_2,\dots,A_n)}(E)$$

تعيد الناتج ضمن علاقة اسمها x وأسماء واصفاتها هي $A_1, A_2, \dots, A_n$.

↙ مثال:

لإيجاد الرصيد الأعلى في المصرف نستخدم الاستراتيجية التالية:

1. إيجاد مجموعة الأرصدة غير العظمى للحسابات في المصرف.
2. طرح المجموعة الناتجة من مجموعة الأرصدة في المصرف، فنحصل على الرصيد الأعظم المطلوب.

مجموعة الأرصدة الموجودة في المصرف هي:

$$\Pi_{balance}(account)$$

مجموعة الأرصدة غير العظمى هي:

$$\Pi_{account.balance}\left(\sigma_{account.balance<d.balance}\left(account \times p_d(account)\right)\right)$$

والنتيجة ستكون ناتج طرح الثاني من الأول، أي:

$$\Pi_{balance}(account) - \Pi_{account.balance}\left(\sigma_{account\,.\,balance<d\,.\,balance}\left(account \times P_d(account)\right)\right)$$

↙ لنوضح مدى فهمنا للعمليات الأساسية بمثال:

قاعدة معطيات مصرف

branch (branch_name, branch_city, assets).
customer (customer_name, customer_street, customer_only).
account (account_number, branch_name, balance).
loan (loan_number, branch_name, amount).
depositor (customer_name, account_number).
borrower (customer_name, loan_number).

The branch relation

branch − name	branch − city	assets
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perry ridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

The borrow relation

customer − name	loan − number
Adams	L − 16
Curry	L − 93
Hayes	L − 15
Jackson	L − 14
Jones	L − 17
Smith	L − 11
Smith	L − 23
Williams	L − 17

The loan relation

loan − number	branch − name	amount
L − 11	Round Hill	900
L − 14	Downtown	1500
L − 15	Perryridge	1500
L − 16	Perryridge	1300
L − 17	Downtown	1000
L − 23	Redwood	2000
L − 93	Mianus	500

Example (1):

Find all loans of over 1200$.

$$\sigma_{amount>1200}(loan)$$

Find the loan number for each loan of an amount greater than 1200$.

$$\Pi_{loan-number}(\sigma_{amount>1200\$}(loan))$$

Example (2):

Find the names of all customers who have a loan, an account, or both, from the bank.

$$\Pi_{customer-name}(borrower) \cup \Pi_{customer-name}(depositer)$$

Find the name of all customers who have a loan and an account at bank.

$$\Pi_{customer-name}(borrower) \cap \Pi_{customer-name}(depositer)$$

Example (3):

Find the names of all customers who have a loan at the perryridge branch but don't have an account at any branch at the bank.

$$\Pi_{customer-name}(\sigma_{branch-name = perry\ ridge}(\sigma_{borrower\ .\ loan-number = loan\ .\ loan-number}(borrower \times loan))) - \Pi_{customer-name}(depositer)$$

Example (4):

Find the names of all customers who have a loan at the perryridge branch.

Query 1:

$$\Pi_{customer-name}(\sigma_{branch-name = perryridge}(\sigma_{borrower\ .\ loan-number = loan\ .\ loan-number}(borrower \times loan)))$$

Query 2:

$$\Pi_{customer-name}(\sigma_{borrower\ .\ loan-number = loan\ .\ loan-number}(\sigma_{branch-name = perryridge}(borrower \times loan)))$$

Example (5):

Find the largest account balance Rename account relation as d>

The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account\ .\ balance<d\ .\ balance}(account \times P_d(account)))$$

The End.

Remember, you are capable of far more than you think.