ERD - Part 2

مسألة مشفى الأسد الجامعي

عملي مشترك

11/04/2023

RB Informatics;

قواعد معطيات 1

السلام عليكم و رحمة الله وبركاته...

تحدثنا في المحاضرة السابقة عن تعريف قواعد المعطيات ومفهومها و تحدثنا عن نماذجها وخصصنا الحديث عن نموذج ERD وأخذنا مثالاً عليه, وفي هذه المحاضرة سنقوم بحل مثال أصعب بعض الشيء لتتضح الفكرة بشكل أكبر.

لنبدأ على بركة الله...

مسألة تصميم نموذج ERD: مشفى الأسد الجامعي

لدى مشفى الأسد الجامعي عدد من المرضى، ويعمل فيه عدد من الأطباء، يتألف المشفى من عدد من الأقسام التخصصية ويتم قبول المريض في القسم المختص لحالته المرضية، كما يوجد في المشفى عدد من الأقسام المشتركة التي تقدم الخدمات لكافة الأقسام مثل (مخبر التحاليل الطبية، قسم التصوير الشعاعي.... الخ). ويخضع المريض خلال إقامته بالمشفى لعدد من الفحوصات وقد تُجرى له عملية جراحية واحدة أو أكثر، لكل مريض من المرضى المقيمين في المشفى في قسم معين طبيب مسؤول عن متابعته ويكون هذا الطبيب واحداً من الأطباء العاملين في المشفى.

والمطلوب:

- إعطاء مخطط كيان ارتباط لهذه القاعدة (قاعدة المعطيات).
- ماذا يحصل لدى انتقال المريض من قسم لآخر (كيف نمثل ذلك)؟
- كيف يمكن استرجاع السجل الطبي للمريض إذا راجع المشفى بعد فترة من خروجه؟

خطوات الحل:

أولاً: نحن نريد تصميم قاعدة بيانات لمشفى عام أو خاص أو...، فمن المنطق أنه يجب في البداية أن يكون لدينا ثلاث كيانات رئيسية لا غنى عنها وهي (الأطباء – المرضى – أقسام المشفى)، أي أن فكرة المشفى بحد ذاتها لا تقوم إذا فُقِدَ أحد هذه الكيانات.

ولكن السؤال الآن هو: ما نوع العلاقة بين هذه الكيانات؟

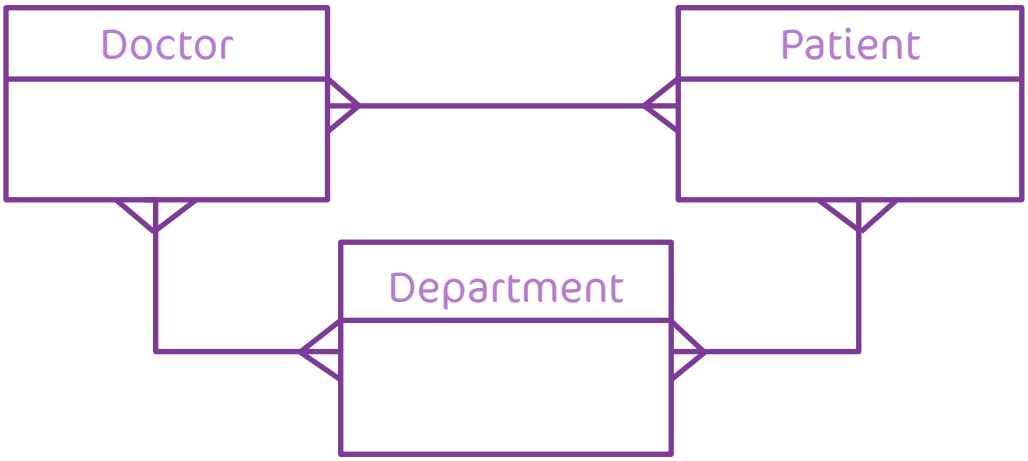
"Happiness is when what you think, what you say, and what you do are in harmony."

- ملاحظة: من المهم جداً فهم نص المسألة وذلك لمعرفة الطريقة التي سنربط بها الكيانات ببعضها البعض بشكل صحيح لتحقيق الطلبات المطلوبة في نص المسألة والخروج بأفضل تصميم Design ممكن لقاعدة المعطيات.

كما تحدثنا لدينا ثلاث كيانات رئيسية : (Doctor – Patient – Department). نعود لنص المسألة فكان من ضمن الطلبات كالتالي:

1. كل مريض سيدخل إلى المشفى يجب أن يكون ضمن قسم معين (حسب حالته المرضيّة).
   - إذاً هنالك حقاً علاقة بين المريض Patient والقسم Department.
   - ونوع هذه العلاقة هي many to many **وذلك لأنَّ:**
     - يمكن لقسم معيّن أن يحوي عدد من المرضى.
     - ويمكن لمريض معيّن بحد ذاته أن ينتقل بين أكثر من قسم خلال دخوله المشفى.
2. لكل مريض في قسم معين طبيب من القسم سيشرف عليه.
   - إذاً هنالك حقاً علاقة بين الطبيب Doctor والقسم Department.
   - ونوع هذه العلاقة هي many to many **وذلك لأنَّ:**
     - يمكن لقسم معيّن أن يحوي عدد من الأطبّاء.
     - ويمكن لطبيب معين أن يكون تابع لأكثر من قسم. على سبيل المثال: كان الطبيب يعمل في قسم الإسعاف ومن ثم بعد الاختصاص أصبح يعمل في قسم الهضميّة.
3. كل مريض سيشرف عليه طبيب.
   - إذاً أيضاً هنالك علاقة بين الطبيب Doctor والمريض Patient.
   - ونوع هذه العلاقة هي many to many **وذلك لأنَّ:**
     - يمكن للطبيب أن يعاين أكثر من مريض.
     - ويمكن أن يكون هنالك مريض تتم معاينته من قِبل أكثر من طبيب.

إذاً أصبح المخطط المبدئي كالتالي:



ولكن هذا المخطط يحتاج إلى إصلاح وكسر علاقات، فكما نعلم لا يمكننا وضع علاقات ال many to many بهذا الشكل وإنما تحتاج لجداول الكسر. ويتم ذلك من خلال مجموعة من الكيانات الضعيفة **والتي يمكن استنباطها من نص المسألة.**

فعلى سبيل المثال:

- ذُكر في نص المسألة أن المريض يتم قبوله في القسم المختص تبعاً لحالته المرضية.

أي أن هناك كيان للقبول **Admission**. وهو متمثّل واقعياً بورقة قبول تعطى للمريض عند دخوله المستشفى يملأ فيها بياناته من اسم وتاريخ الدخول وتاريخ الخروج و.... و.....، أي أن العلاقة التي كانت **many to many** بين المريض **Patient** والقسم **Department** تم كسرها من خلال كيان القبول **Admission**. ومن خلال ورقة القبول نستطيع تحديد القسم **Department** الذي سيتوجه إليه المريض.

- أيضاً ذُكر في نص المسألة أن لكل مريض في المشفى هنالك طبيب مسؤول عن متابعته.

أي أن هنالك كيان للمتابعة **Audit**, وهو متمثّل واقعياً بسجّل أو ب إضبارة تكون مُلازمة للمريض في المشفى وعند تنقله بين الأقسام ويُسجَّل فيها كل ما يتم إعطاؤه لهذا المريض من أدوية وإبر وتحاليل و...و...، أي أنَّ العلاقة التي كانت **many to many** بين المريض **Patient** والطبيب **Doctor** تم كسرها من خلال كيان المتابعة **Audit**.

- ذُكر أيضاً أنه يمكن للمريض خلال تواجده في المشفى أن يخضع لتحاليل معينة أو لعمليّة ما.

فنتيجة لل **Audit** أي نتيجةً لمتابعة المريض أصبح من الممكن ظهور كيان للعمليات **Operation** وستكون علاقتها كما ذكر في نص المسألة مع الأقسام المشتركة **Public**.

على سبيل المثال – الطبقي المحوري: يمكن لأي مريض من أي قسم أن يقوم بعمل طبقي محوري.

مثال آخر – غرف العمليّات: ففي غرفة العمليات نفسها يمكن عمل عمليات كل منها تابع لقسم مختلف.

"Your self-worth is determined by you. You don't have to depend on someone telling you who you are."

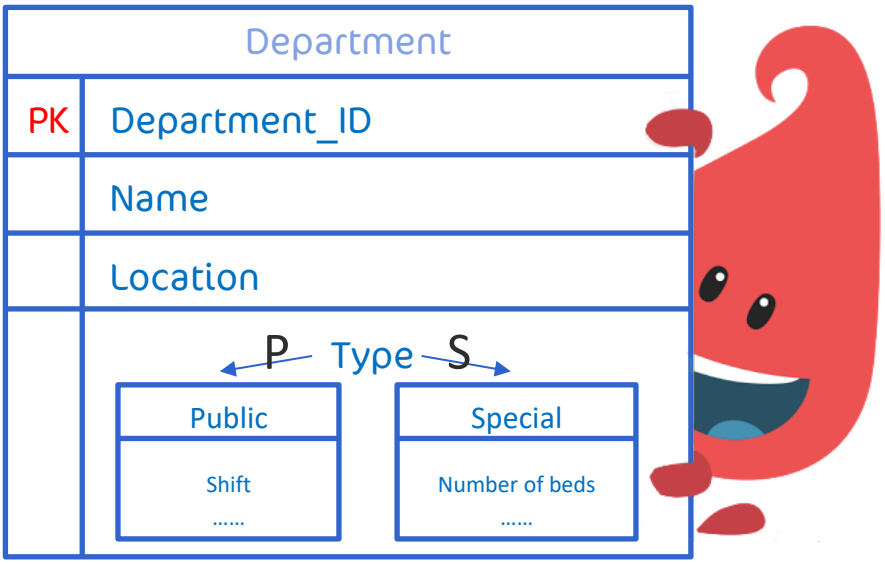
- سؤال: كيف سنتمكن من تطبيق هذه الفكرة أي فصل الأقسام بين خاص Special وعام مشترك Public في جدول القسم Department؟

- بدايةً فإنه يمكننا جعلها كيانين قائمين بحد ذاتهم وهذه فكرة صحيحة **ولكِن** وبسبب أن هنالك جزء من ال Attributes بينهما مشتركة مثل .... ,Name, Location ، ومع الأخذ بعين الاعتبار احتمال وجود attributes في أحد هذين الكيانين دون وجوده في الكيان الآخر، ولغاية عدم تعقيد التصميم Design **فإننا استخدمنا خاصية التجميع**.
- والمقصود بخاصية التجميع أننا نقوم بتجميع الattributes المشتركة بين هذين الكيانين ونضعها في كيان رئيسي واحد، ثم في داخل هذا الكيان الرئيسي يكون هنالك كيانين فرعيين هما خاص Special وعام مشترك Public.
- **عندها:** نقوم بوضع ال attributes الخاصة بال Public في كيان ال Public وال attributes الخاصة بال Special في كيان ال Special. وذلك من خلال أن يكون هنالك Attribute وليكن اسمها type فنقول على سبيل المثال:

إذا كانت قيمة type هي p: عندها نعلم بأن المعلومات التي سندخلها هي الAttributes الخاصّة بال **Public** وعندها تأخذ الAttributes الخاصة بالSpecial القيمة **Null**.

إذا كانت قيمة type هي s: عندها نعلم بأن المعلومات التي سندخلها هي الAttributes الخاصّة بال **Special** وعندها تأخذ الAttributes الخاصة بالPublic القيمة **Null**.

أي أنه سيصبح شكل ال Table الخاصة بال Department كالتالي على سبيل المثال:



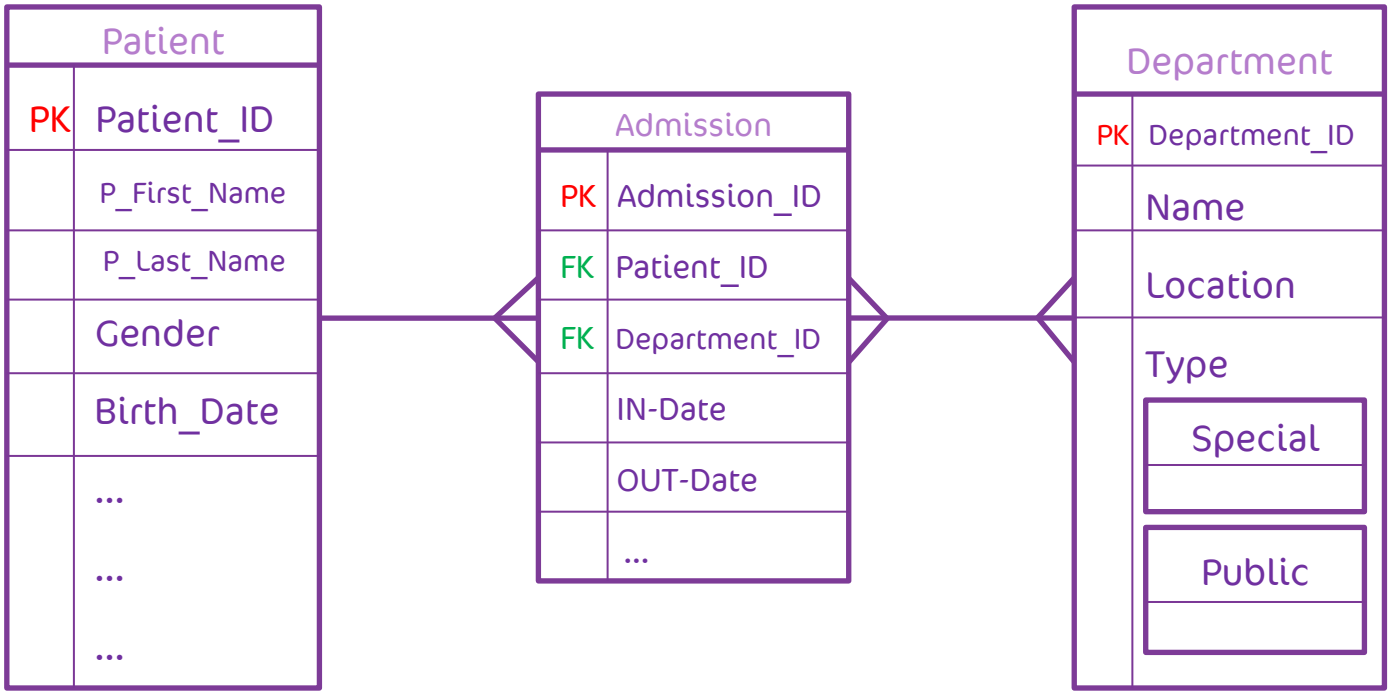
لنتابع الآن في إصلاح الجدول وكسر العلاقات: فكما ذكرنا إنَّ العلاقة بين المريض Patient والقسم Department تم كسرها من خلال كيان القبول Admission. وكيان ال Admission له العديد من ال Attributes مثل IN-Date و OUT-Date و... و...

- إذاً فالآن أصبحت العلاقة بين المريض Patient والقبول Admission هي علاقة one to many من طرف المريض Patient.
- أمّا عن العلاقة بين القسم Department والقبول Admission فإنها ليست كذلك وإنّما هي علاقة many to many

ولكن لماذا؟!

لأنّه من المحتمل لمريض معيّن أن ينتقل بين أكثر من قسم ضمن نفس القبول.

إذاً لنفرض أن هذه العلاقة أصبحت كالتالي:



ولكن الآن ماذا عن العلاقة بين القسم Department والقبول Admission؟!

لنأخذ مثالاً واقعياً ; على سبيل المثال:

- القبول رقم 100 والعائد للمريض ذو الID رقم 11 ← { والذي اسمه: أحمد، وجنسه: ذكر، وعمره: 30, ...} وتم قبوله في القسم ذو ال ID رقم 2 ← { والذي يمثل: قسم القلبيّة، وموقعه: في الطابق الثالث، ...} وتم هذا القبول بتاريخ 24/12/2020 عند الساعة 11:05 صباحاً. في اليوم ذاته عند الساعة 10:50 مساءاً تعرّض هذا المريض (أحمد) لجلطة دماغيّة، فتوجّب نقله للقسم ذو الID رقم 6 ← { والذي يمثل: قسم العناية المشدّدة، وموقعه: في الطابق الأرضي، ...} .

"Don't be afraid to give up the good to go for the great."

- والسؤال هو: أين سنجد هذه المعلومة (التي تدل على انتقال مريض من قسم إلى آخر)؟

- هل سننشأ New Record أي Admission جديد لنفس الشخص (أحمد) ونفس التاريخ عند الساعة 10:50 وتكون قيمة الDepartment_ID فيه 6؟

  بالتأكيد فإن هذا الحل خاطئ، فلا يمكننا أن ننشأ أكثر من Record لنفس الData .

- أو أننا نذهب إلى نفس الAdmission (ذو الرقم 100) ونقوم بتغيير ال Department_ID الخاص به من 2 إلى 6؟

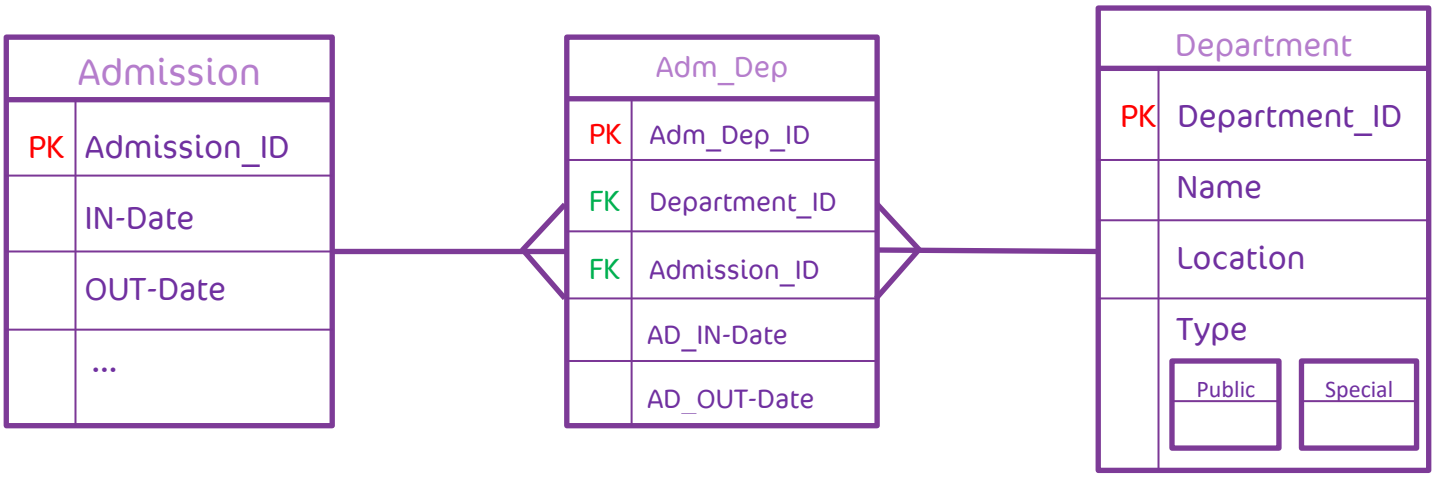
  بالتأكيد فإن ذلك أيضاً خاطئ، لأننا بهذه الحالة نكون قد تخليّنا عن معلومات وسجلّات في غاية الأهمية... فعند القيام بذلك مثلاً لن يعود بإمكاننا معرفة الأقسام التي انتقل بينها (أحمد) منذ دخوله المستشفى، وإنمّا يمكننا فقط معرفة آخر قسم دخلَهُ وذلك بالطبع ليس صحيحاً.

- ملاحظة: ممنوع ل Attribute معين أن تتغير قيمته على نفس ال Row, أي في ال Record الواحد فإن الattributes تكون قيمتها ثابتة.

- إذاً فالعلاقة بين القسم Department والقبول Admission هي many to many وتحتاج حقاً إلى جدول كسر. وجدول الكسر هذا سيكون عبارة عن كيان بسيط يدعى Adm_Dep.
- إن الكيان Adm_Dep بالتأكيد له primary key وهو Adm_Dep_ID. وبطبيعة الحال وبما أنه يستخدم لكسر علاقة بين كيانين فإن له 2 foreign keys هما Admission_ID, Department_ID.
- وكما تحدثنا في المحاضرة السابقة فإن أي كيان يجب أن يكون هنالك attributes خاصة به وتعبّر عنه، ولتكن هذه الattributes هي AD_IN-Date و AD_OUT-Date.

- ملاحظة: إنَّ هنالك فرق بين ال IN-Date, OUT-Date و بين ال AD_IN-Date, AD_OUT-Date. وهو أن ال IN-Date و الOUT-Date تشير إلى تاريخ دخول وخروج المريض من المستشفى بشكل عام, أما ال AD_IN-Date و ال AD_OUT-Date تشير إلى تاريخ دخوله وخروجه من قسم معين ضمن نفس القبول. أي أن تواريخ ال AD_IN-Date وال AD_OUT-Date يتوجب حتماً أن يكون زمنها بين ال IN-Date وال OUT-Date.

إذاً يصبح شكل العلاقة كالتالي:

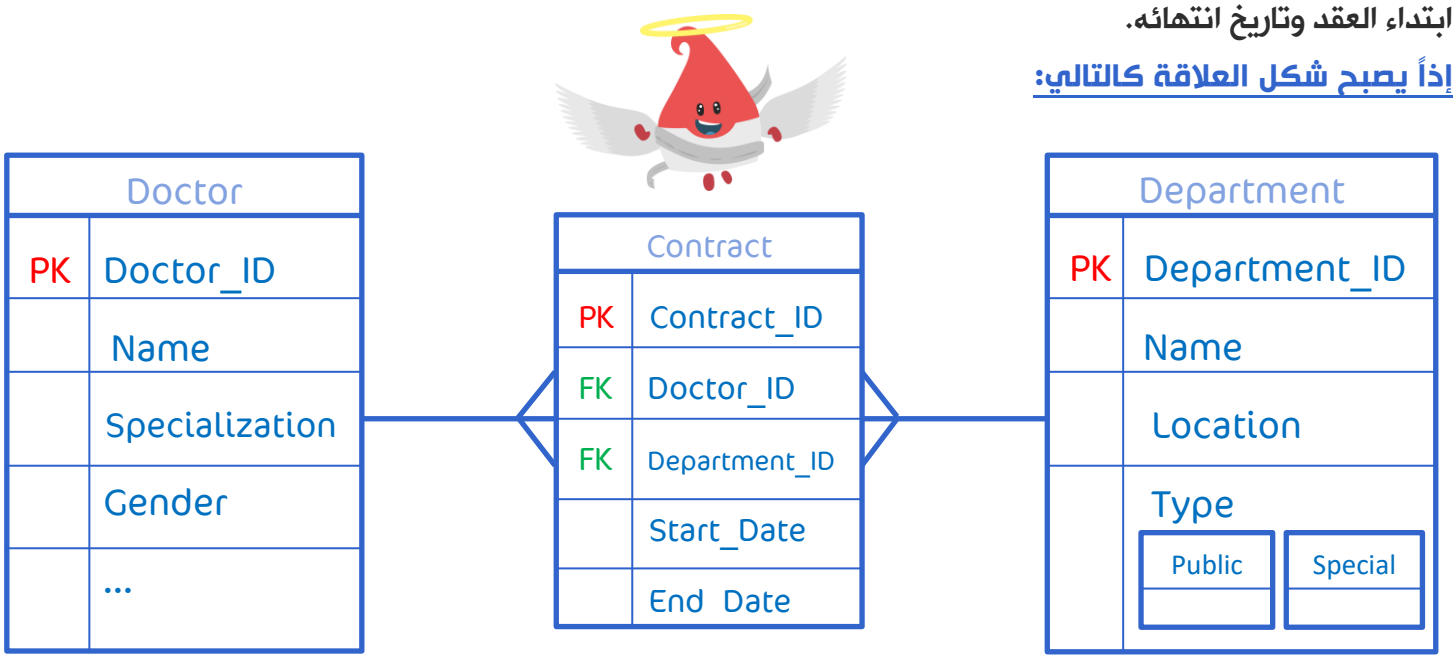


لنتابع في إصلاح باقي العلاقات: إن علاقة الطبيب Doctor مع القسم Department هي علاقة many to many وسنقوم بكسرها من خلال كيان جديد يدعى Contract يعبر عن العقد بين طبيب معين وقسم معين وسنأخذ مثالاً على ذلك مباشرةً:

- تعاقد قسم الإسعاف من المستشفى مع طبيب معين حديث التخرج لمدة سنة ... وبعد 5 سنوات وبعد أن قام هذا الطبيب بدراسة اختصاص الهضميّة على سبيل المثال عاد إلى هذه المستشفى ليقوم بتوقيع عقد ولكن ليس مع قسم الإسعاف وإنما الآن سيقوم بتوقيعه مع قسم الهضميّة.

إذاً بالنسبة للكيان Contract فسيحوي Contract_ID والذي هو نفسه ال primary key. وأيضاً فيه 2 foreign keys هما Doctor_ID, Department_ID. وأيضاً attributes خاصة به ولتكن Start_Date و End_Date دلالةً على تاريخ ابتداء العقد وتاريخ انتهائه.

إذاً يصبح شكل العلاقة كالتالي:



يتبقى لدينا كيانين وهما كيان المتابعة Audit وكيان العمليات Operation:

لنبدأ بكيان ال Audit, ولكن بداية يتوجب علينا الإجابة على سؤال وهو: من هم الParents الخاصين بكيان الAudit؟ أي من هما طرفا العلاقة؟

- ملاحظة: كما تحدثنا فإنه يمكن أن يكون هنالك أكثر من تصميم Design صحيح لقاعدة المعطيات Database ولكننا دوما نبحث عن الDesign الأخف كلفة من ناحية الProcessing والأقل تعقيداً.

فعلى سبيل المثال يمكننا جعل الParents لكيان الAudit هما الDoctor والPatient ولكن هذا ليس الحل الأمثل، ولنأخذ مثلاً على ذلك:

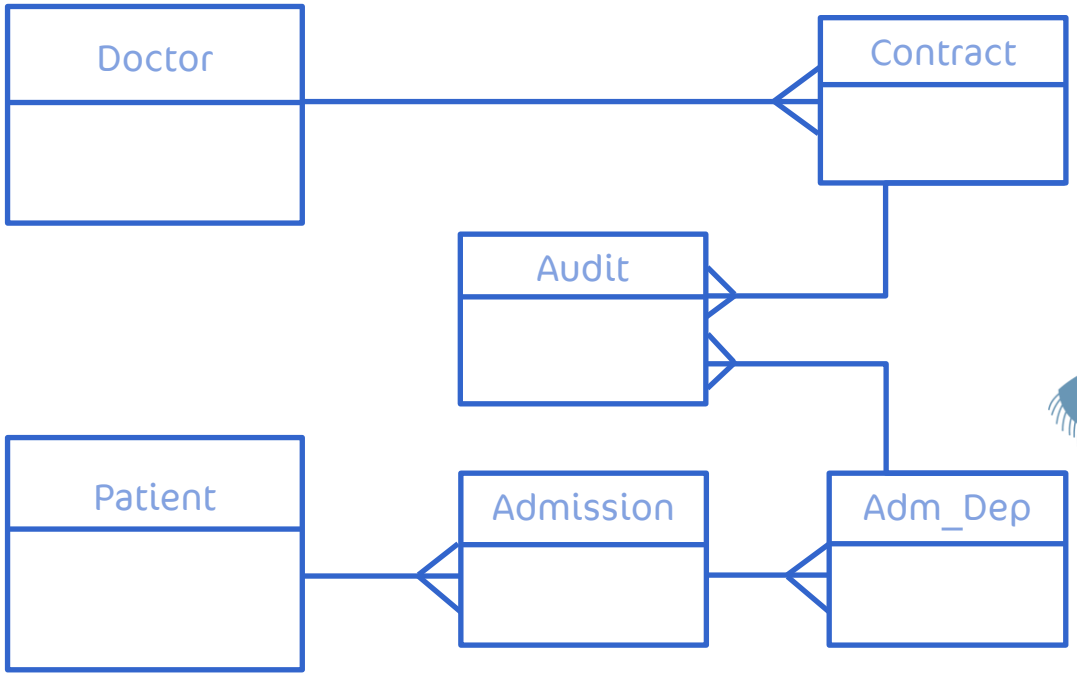
- ليكن هناك طبيب معين كان لديه 5 عقود مع هذه المستشفى متغيّرة مع الزمن، فعند قراءتنا لهذا التوصيف الخاص بهذه الAudit نستطيع معرفة أن هذا التوصيف قد قام بكتابته الطبيب (سامر) على سبيل المثال.
  ولكن ماذا لو أردنا معرفة فيما إذا كان الطبيب سامر مختصاً حين كتابته هذا التوصيف أم أنه كان لا يزال طبيباً عاماً؟

- عندها ستكون هذه العملية مكلفة أي أننا سنذهب لجدول الContract ونبحث ضمن التواريخ عن التاريخ الذي تم فيه كتابة هذا التوصيف ثم ننظر حينها فيما إذا كان الطبيب سامر مختصاً في ذلك الوقت أم لا.
- وكذلك الأمر لو جعلنا الPatient هو الParent الخاص بالAudit. فحينها يصبح الأمر صعب جداً في حال أردنا الحصول على Data معينة غير موجودة في هذين الكيانين.
- بينما لو جعلنا الParents لكيان الAudit هما الContract والAdm_Dep عندها ستكون عملية الوصول إلى الData عملية سهلة وغير مكلفة.

وذلك لأننا عندما نقوم بالربط مع الChild وليس مع الParent فعندها نكون ضمنياً قد قمنا بعمل join لكيان الChild مع كيان الParent ويصبح بإمكاننا ان نصل للمعلومات في كيان الParent .

فعلى سبيل المثال: عندما قمنا بربط الAudit مع الAdm_Dep فنحن فعلياً قمنا بعمل join بين الAdm_Dep والAdmission و الPatient.

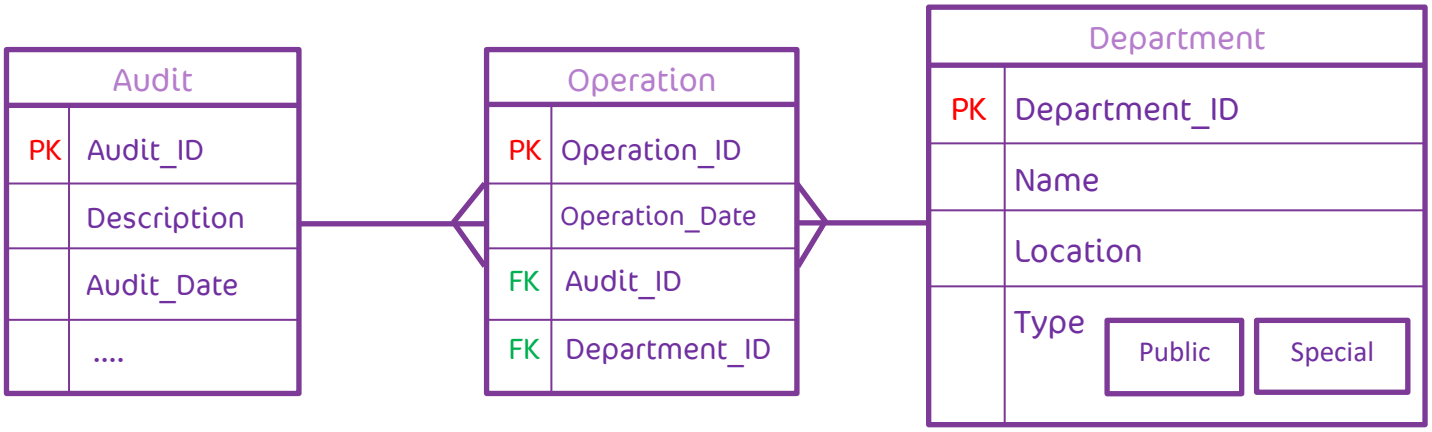
إذاً يصبح شكل العلاقة كالتالي:



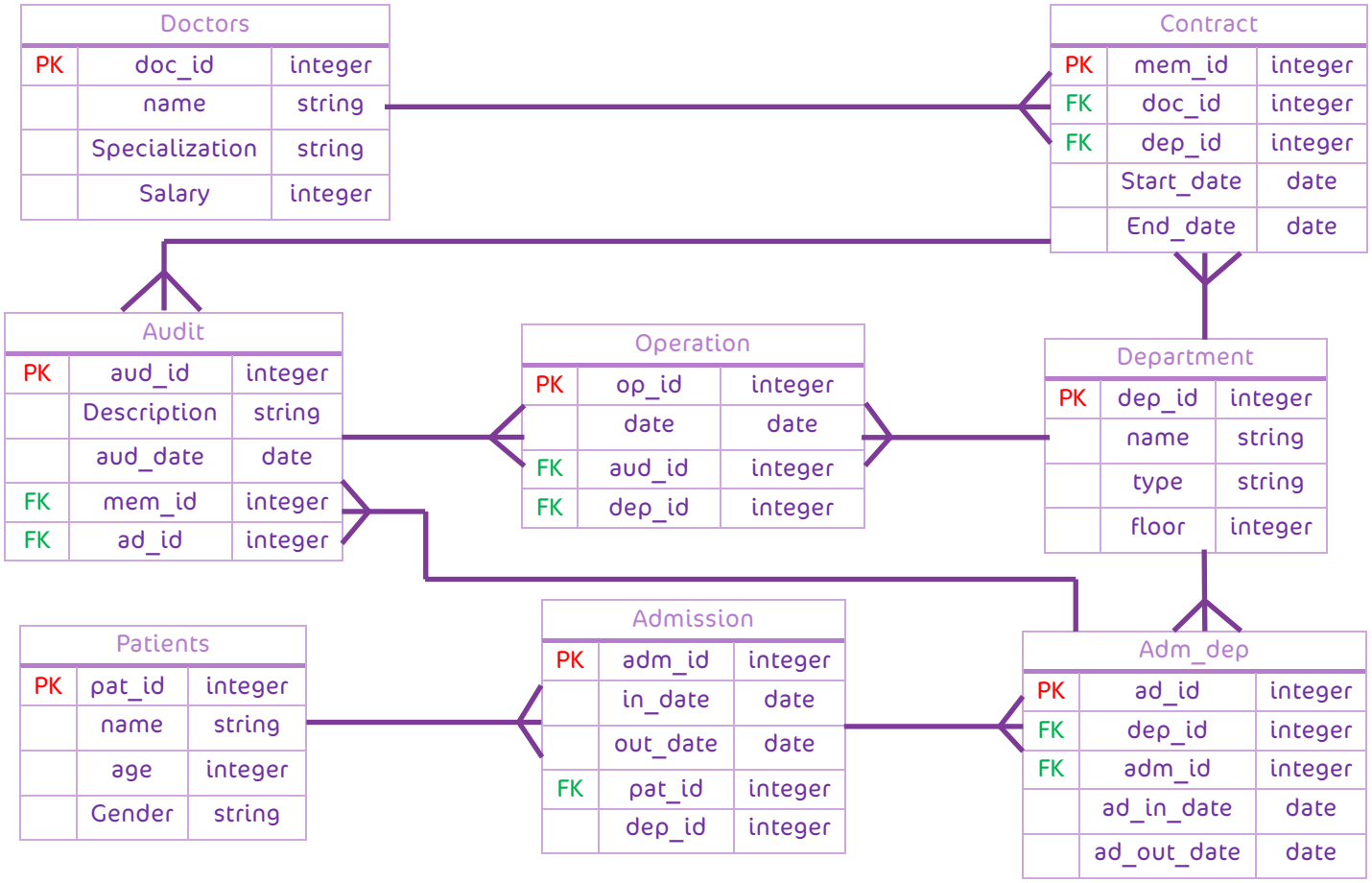
تبقى لدينا كيان الOperation :

- بالنسبة لكيان الOperation فإن له حتماً primary key هو Operation_ID وتاريخ العملية على سبيل المثال Operation_Date. الآن منطقياً فإن أحد الParents لهذا الكيان هو Audit; فبكيان الAudit تم التوصيف وما إلى هنالك.
- **ولكن** ليس بالضرورة أن يكون الطبيب الذي وصّف الحالة وطلب العملية هو نفسه الطبيب الذي قام بها، فمن الممكن أن يكون من قام بها هو طبيب آخر ومعه معاون وطبيب تخدير و....و... وكل ذلك فعلياً موجود في كيان الDepartment .

إذاً فتصبح العلاقة كالتالي:



وبعد أن انتهينا من تبسيط العلاقات وكسرها يكون شكل جدول الERD النهائي للمسألة السابقة كالتالي:



وبهذا نكون قد مثّلنا جدول المعطيات وشرحنا كيف يمكن لمريض أن ينتقل بين قسم وآخر. وحافظنا من خلال هذا التصميم على سجلّات المرضى بحيث تكون جميع بياناتهم موجودة في حال قام أحدهم بمراجعة المشفى بعد فترةٍ من خروجه.

نهاية المحاضرة