Relational model as of 2010-09-13

- **SCORE-Relational-Model-2010-09-13-Revised.pdf**
- SCORE-Relational-Model-2010-09-13.pdf

Conventions in the document:

- PRIMARY KEYs are bold
- FOREIGN KEYs are italic
- Entities are Capitalized and have a dark header
- relations start with a lower case letter and have a beige header
- blue attributes are solely for reasons of provenance (http://en.wikipedia.org/wiki/Provenance#Computers_and_law)

I'm going to update the description of entites so that it is better understandable. However, some general concepts in the draft are:

I try to avoid multi-value primary keys. If an entity is referred to by another (= there is a relationship between them) and would have a multi-valued key otherwise, I introduce an artificial key. If human language is involved in any way (e.g. a name which might be changed later on) I introduce an artificial key instead of using that possibly changing name in order to avoid update anomalies or overhead (even if it leads to redundancy in the database, but I think that storage is even saved and performance improved when a foreign key references a number instead of a string).

Some entities can possibly have more attributes as stated in the model. Such attributes are mostly dependent on a university (e.g. in some universities lecturers do have their own office, which is not necessarily a requirement in most schools where most teachers do not have their own office; it might be desirable to save an e-mail or postal-adress along with a member of the staff or a student, and so on, and so on). Therefore the entity "Attribute" is defined which may be defined to be available for persons with a given role via "roleHasAttribute". The actual value of an attribute for a given person is stored via "hasAttribute".

Permissions are handled in a similar way, where "impliesPermission" is the same thing as "roleHasAttribute" (maybe for consistency either the one should be named "roleHasPermission" or the other one "impliesAttribute") and "hasPermission" is analogous to "hasAttribute".

CourseAttribute is a similar mechanism for Courses, but less complicated since Courses do not act in certain roles. Things like "description", "levels of expectations" (german: Erwartungshorizont) go here. Since all of these are CRUDable they support scalability of our application.

For reasons of simplicity all user-definable attributes such as Attribute, Permission and CourseAttribute are stored as String (in SQL this might be a VARCHAR or TEXT with 1024 characters length or whatsoever). A type may be defined in terms of XML Schema, which can than be checked by the GUI at time of entering/editing a value.

FAQ

> what's the difference between the relation 'Attribute' and 'CourseAttribute?', they got the same syntax/schema. --*andré*

Attribute applies to Person, CourseAttribute? to Courses. There may be two kinds of Attributes applying to different kinds of entities, named the same but having a different meaning.

> Room: Statt number vielleicht eher number_of_seats oder direkt seats? Sprechende Namen helfen besser, name ist glaube ich ohnehin ein postgres Keyword und wird escaped, was wir nach meinem Stand vermeiden wollten. --*konrad*

number ist nicht die Anzahl der Sitze, die Anzahl der Sitze ist ein Feature. "providesFeature" (Spalte 1, Tabelle 5) ist eine attribuierte Relation zwischen "Room" und "Feature" die den Sachverhalt number_of_seats modellieren kann. Number ist, übrigens im Namen konsistent gehalten mit der verbalen Beschreibung, die Raum-Nummer, die, wie in der Beschreibung festgehalten, eindeutig in einem Gebäude ist (es kann einen R103 in Taku- 9 als auch in

Arnimallee 22 geben), daher gibt es dafür eine eigene UNIQUE-Constraint (Spalte 1, Tabelle 1, 7. Zeile).

> RoomType?: Wie sinnvoll ist dies als eigene Entität zu modellieren? Wäre z.B. isComputerLab nicht auch einfach ein Feature? Zumindestens aus Scheduler Sicht oder ist es ohnehin so das RoomType? sowieso ein hard constraint wird? *--konrad*

Ich sehe das prinzipiell auch so, prinzipiell sind RoomType? und CourseType? des generischen Ansatzes Features und Attribute zuzuordnen überflüssig. Dass ich sie eingeführt habe ist meinerseits eigentlich nur ein Relikt aus vorigen Versionen; ich weiß dass ich in Gesprächen mit André beides kurz ersonnen habe, aber wie gesagt, es gibt den generischen Ansatz, alles andere ist Überfluss. Ich werde es, sofern kein anderer stichhaltige Gründe FÜR RoomType? oder CourseType? hat beides aus dem Modell entfernen.

> Permission: Unser Rechtemanagement läuft ja auf kein hierarchisches hinaus, seh ich das richtig? *--konrad*

Richtig. Es ist flach.

> Year als eigene Entität halte ich für Redundanzfreiheit auf die Spitze getrieben oder was ist die Idee davon? *--konrad*

Year ist sowas wie "1. Fachsemester Informatik" oder "3. Fachsemester Philosophie". Ein Kurs findet vorzugsweise in einem solchen Year statt, vgl. Alp I im 1. Fachsemester der Informatiker. Dies ist eine besonders wichtige Entität bzgl. des Scheduling-Prozesses, denn Kurse die im selben Year stattfinden dürfen sich nicht überschneiden. Alp 1 und TI 1 sollen nicht zeitgleich stattfinden, können aber durchaus zum Zeitpunkt einer Veranstaltung des 1. Fachsemesters Philosophie stattfinden. Deswegen gibt es diesen, auf den ersten Blick like overkill aussehenden, Ausbund an Komplexität hier.

> Ansonsten kann name als primary key gewählt werden, der ohnehin UNIQUE ist. Du meintest doch künstliche Schlüssel um multi-valued primary keys zu vermeiden. *--konrad*

Und "whenever human language is involved".

> Permission, siehe 5. Wobei ich es hier schon wieder schöner finde, die Zugriffsrechte gesondert in einer Tabelle zu haben. *--konrad*

Was heißt das?

> Stehen in timeslot die timeslots für alle Departsment drin? *--konrad*

Nein, das sind lediglich Dinge wie "1. Stunde". Es sind keine TimeSpaceSlots?, nur definierte Punkte in der Zeit. Timeslots gehören zu einem Tag und modellieren das Zeitraster auf dem wir arbeiten.

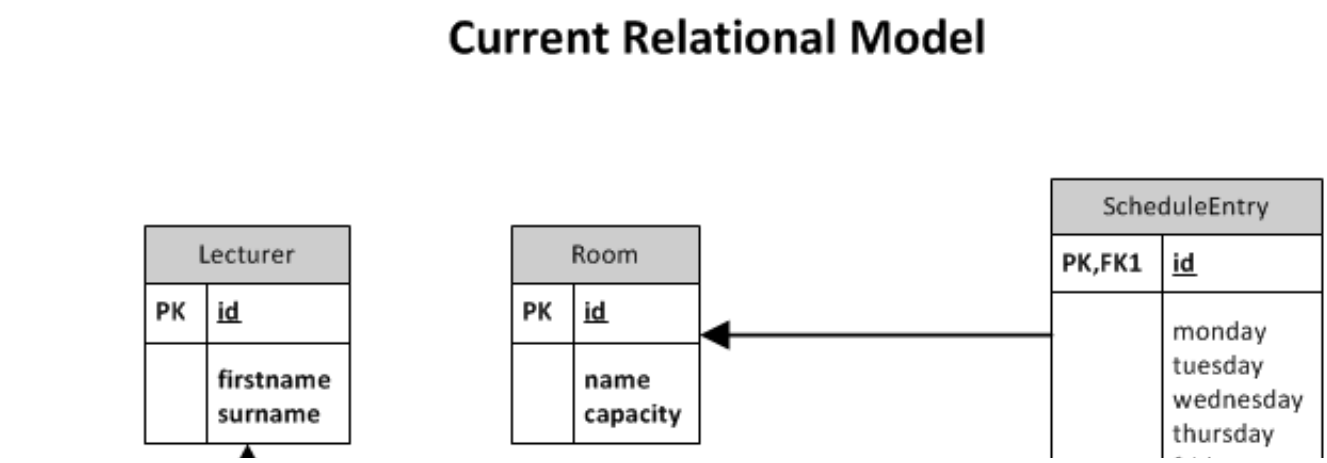
> Wie sieht es aus mit dem Stundenplan? [...] vielleicht sogar VIEW [...] *--konrad*

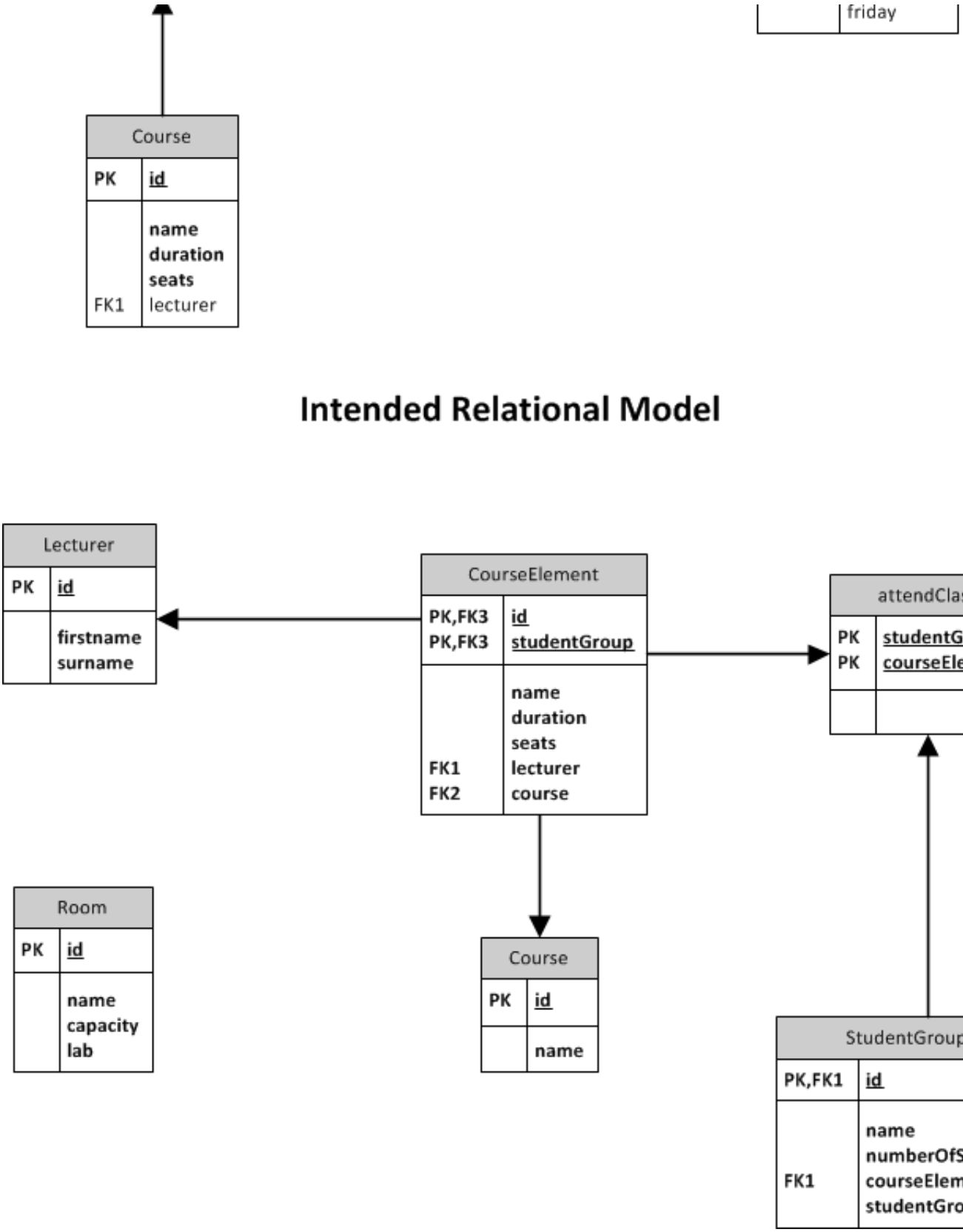
Wir haben das mit dem View in einer Open Question zu den UseCases (siehe Protokoll) abgehandelt. Die Informationen im relationalen Modell sind dafür bereits vorhanden.

> Es fehlt noch die Relation [Student] isEnrolledIn [CourseElementInstance?] damit für den Student sein persönlicher Timetable berechnet werden kann oder? *--konrad*

enrolledInCourse, Zeile 3, Tabelle 11. Die zweite Relation nach Period.

Relational Model as in ju-scheduler and konrad-scheduler





▶ **Attachments**