Реализация темпоральнсоти.

При редактировании аттрибутов студента, изменения сохраняются в виде новых записей по аттрибутам в базе данных, без удаление или редактировании предыдуших. Связь студента с предыдущими значениями фиксируется в **ChangeList** который содержит в себе такие поля как тип данных, характериющие название поля для студента, **old_id** - старый айди записи в таблице, **new_id —** id нового значения аттрибута студента сохраненного в таблице, **created_at —** поле типа **Data,** хранящее дату создания данной записи. **student_id** — id студента в системе. По **student_id** выбираются записи, изменения, относящиеся к данному студенту, **m_type —** имя таблицы в которой храниться аттрибут.

При просмотре данных о студенте по дате актуализации идет запрос к таблице Student на проверку существования данного студента в системе. В случае успеха следующий запрос обращаяется к таблице **ChangeList.** Производится выборка записей по **student_id** и **created_at.** Это означает, что выбиаются все записи принадлежащие данному студенту и сгруппированые по дате создания для таблицы, но являющимися датой изменения значений аттрибутов для объекта. Тем самым получаем список изменений для данного студента с интересующей нас датой актуализации.

Как уже говорилось выше, в таблице имеются такие поля как **created_at, student_id, new_id, old_id, m_type**. Поговорим о том, как производится работа над этими данными. При запросе к **ChangeList** возращается ответ в ввиде списка, каждый элемент которого содержит такие поля. Соответственно по **m_type** производятся запросы к определенным таблицам по **new_id** и **old_id**, тем самым подгружая те элементы, значения которых были изменены в определенный момент времени (**new_id** — новое значение **old_id** — старое значение). Так как данные в Ruby on Rails хранятся в виде моделей данных, то внутри каждого из них содержится такой метод как **chage_display,** который сравнивает поля аттриутов и разнице возращает в виде массива.

К примеру, допустим, студент за время периода обучения сменил фактический адрес проживания. Для этого может быть множество причин, такие как высока стоимость оплаты, при съемном жилье, а может и просто на новом месте путь, затрачиваемый на путь к месту обучения меньше. Данные адреса содержат такие поля как **person_id-** id человека являющимся студентов (связь таблиц persons к student смотрите в диаграмме классов) , **a_type** — тип адреса (прописка, фактический, регистрация), **address** — сам адрес. Соответственно при их изменении при редактировании студента, на сервер поступает массив данных, который обрабатывается. По этим данным, создается новый объект типа **Address** в который записываются поступившие значения. В тиблице **ChangeList** создается запись с **id** предыдущего адреса, а так же с новым и названием таблице. В поле **created_at** выставлется время создания записи. Ниже приводится диаграмма для наглядного отображения процесса (Рис.).

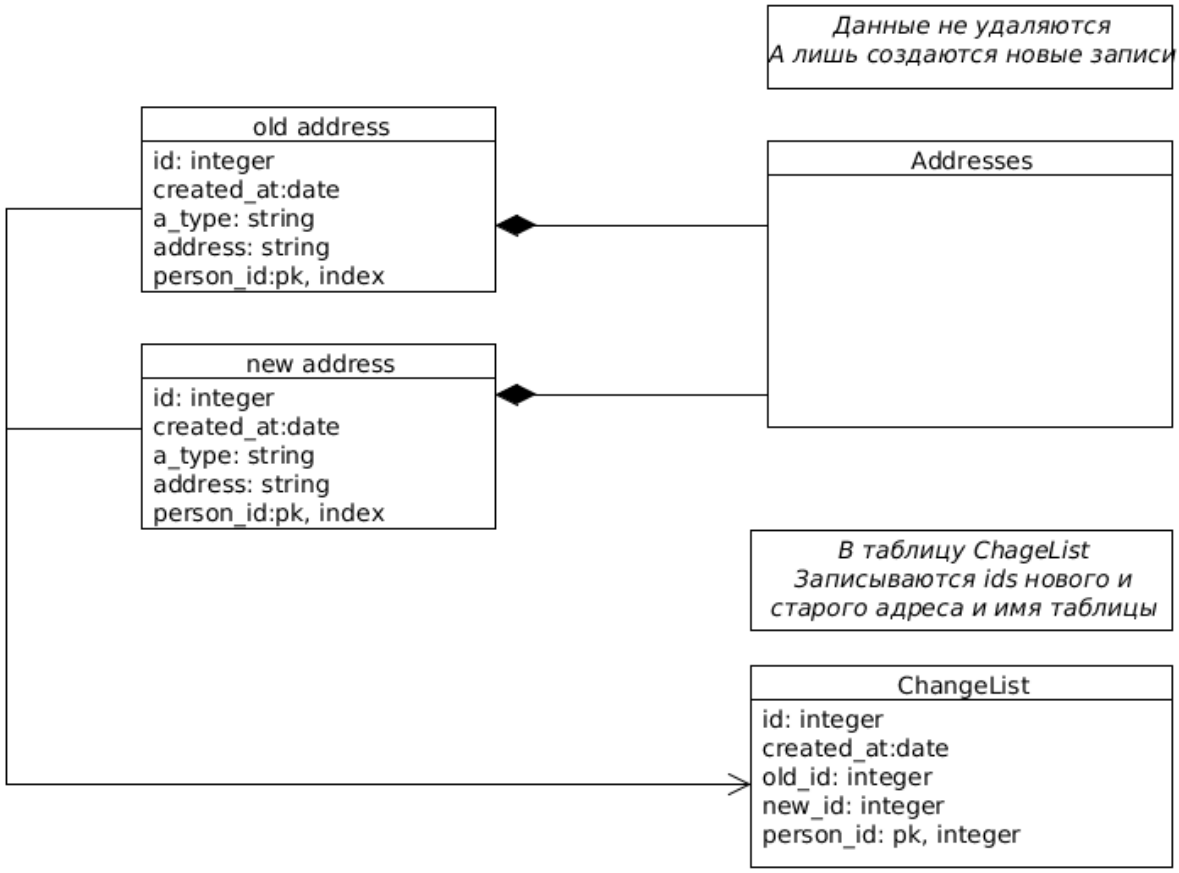

Рис.

Для рассмотренного примера необходимо подметить, что при считывании актуальных данных для студента, обращения к **ChangeList** не происходит. А лишь, к таблице адресов, так как, последняя сохраненная запись, с выборкой по типу и является актуальной, так как именно последняя сохраненая запись, является последней созданной для студента. Ниже приведено примерно ход работы с применением запроса в приложении (Рис.):

```
Address.where(:person_id => @student.person.id).where(:a_type=>params[:a_type]).last
```

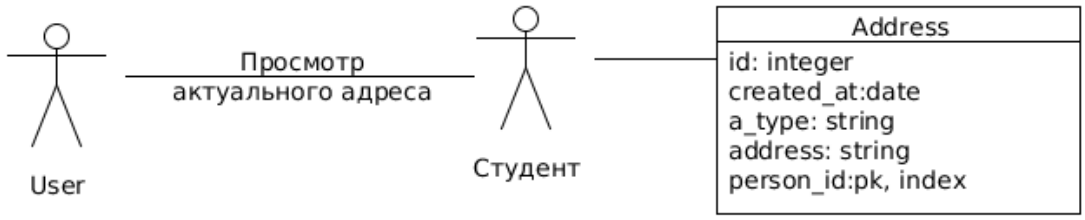

Рис.

Примерно такие же запросы производятся к каждому аттрибуту студента. Тем самым можно представить процесс работы приложения для просмотра актуальных данных.

Теперь рассмотрим выборку по дате актуализации. Для наглядности рассмотрим выборку для адреса (Рис.).

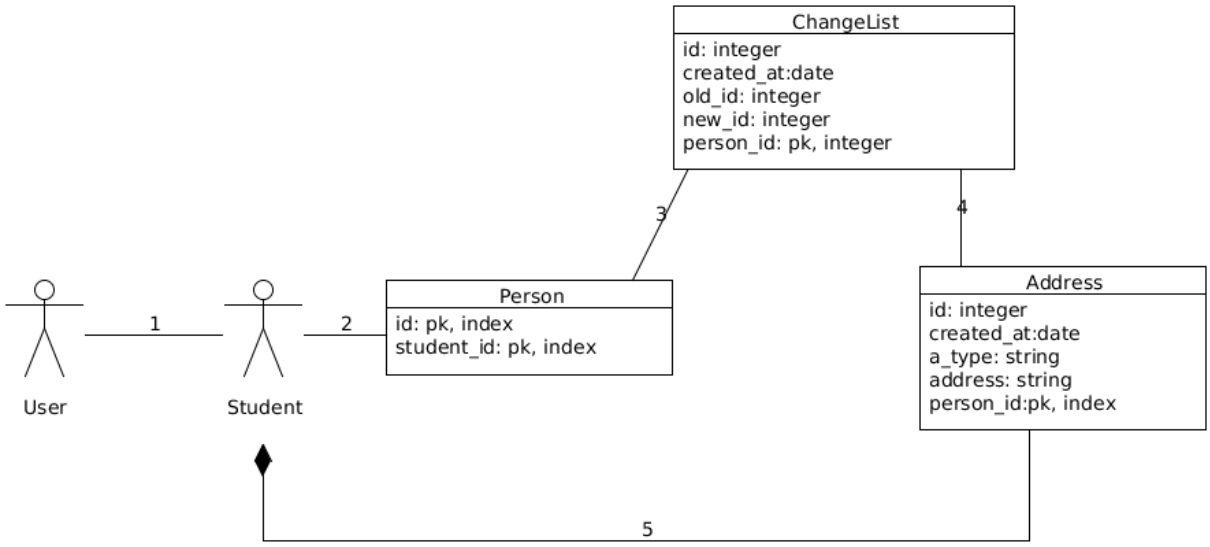

Рис.

Определение:

1. Запрос к студенту по дате актуализации;
2. Узнаем **person_id**, ведь через него студент связан личными данными как человек.
3. Запрашиваем у **ChangeList** данные с учетом типа интересующего типа данных (address), даты актуализации и принадлежности к человеку.
4. По полученному **old_id** запрашиваем элементы из таблицы addresses.
5. Возвращяем значение. Теперь студент иммет адрес соответствующий дате актуализации.

Необходимо заметить, что это всего лишь абстрактный пример демонстрирующий работу алгоритма, заложенного в проект. По факту в системе реализована комплексный анализ данных по дате актуализации, и данные из **ChangeList** получают сразу списком. Как уже говорилось раннее, запрос происходит по **person_id** и **created_at** и результирующий список составляют из запросов полученных к таблицам по **old_id** (Рис.)**.**

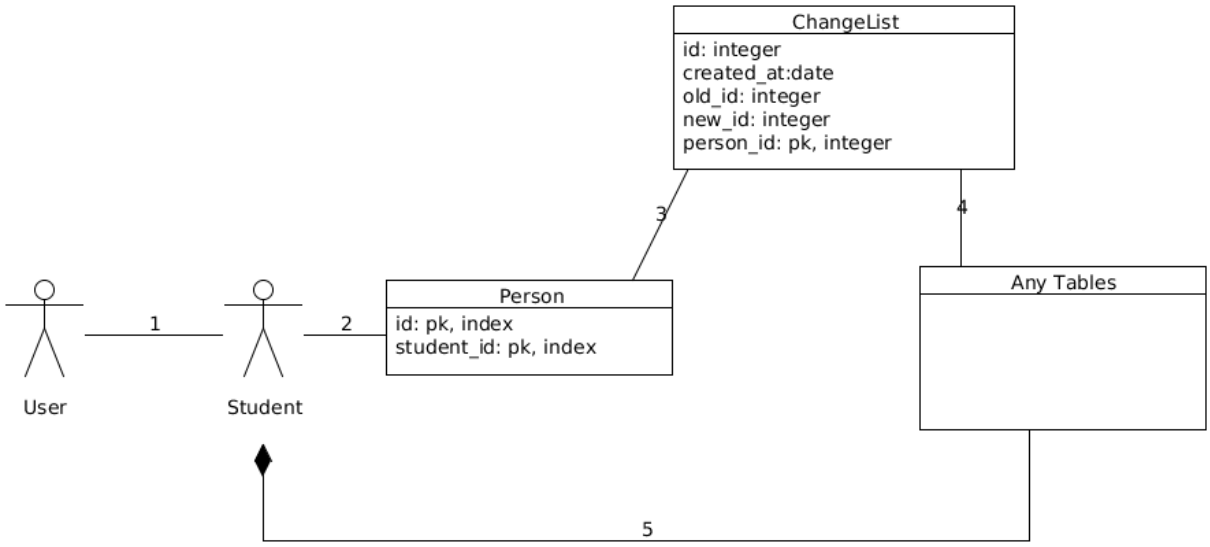

Рис.

Определение:

6. Запрос к студенту по дате актуализации;
7. Узнаем **person_id**, ведь через него студент связан личными данными как человек.
8. Запрашиваем у **ChangeList** данные с учетом типа интересующего типа данных, даты актуализации и принадлежности к человеку.
9. По полученному **old_id** запрашиваем элементы из таблиц.
10. Возращяем значения. Теперь студент содержит значения аттруботов по дате актуализации.

На этом принцып темпоральности данной системы изложен. Далее разберем работу формирования списка изменений для студента в **GUI,** благодаря которому легче ориентироваться по **lineHistory.**