# Структура таблиц базы данных

В базе данных выделяются следующие 15 таблиц, которые хранят в себе данные для некой минимальной работы с этой базой данных:

* Таблица POSITIONS – данная таблица хранит в себе все возможыне должности сотрудников университета;
* Таблица FACULTY – данная таблица хранит в себе информацию о факультетах университета;
* Таблица PULPIT – данная таблица хранит в себе информацию о кафедрах университета;
* Таблица DISCIPLINE – данная таблица хрнаит в себе информацию о преподаваемых дисциплинах;
* Таблица EMPLOYEE – данная таблица хранит в себе информацию о сотрудниках университета;
* Таблица PULPIT\_MANAGER – данная таблица хранит в себе информацию о заведующих кафедрами;
* Таблица FACULTY\_MANAGER – данная таблица хранит в себе информацию о деканах факультетов;
* Таблица SPECIALITY – данная таблица хранит в себе информацию о возможных специальностях на факультете;
* Таблица GROUP – данная таблица хранит в себе информацию о группах специальностей;
* Таблица STUDENT – данная таблица хранит в себе информацию о студентах университета;
* Таблица ADDRESS – данная таблица хранит в себе информацию об адресах корпусов;
* Таблица BUILDING – данная таблица хранит в себе информацию о корпусах университета;
* Таблица AUDITORIUM\_TYPE – данная таблица хранит в себе информацию о типах аудиторий;
* Таблица AUDITORIUM – данная таблица хранит в себе информацию об аудиториях университета;
* Таблица MARK – данная таблица хранит в себе данные об успеваемости студентов.

# Индексы

В базе данных выделяются следующие некластеризованные индексы:

* Индекс STUDENT\_NAME\_INDEX индексирует таблицу STUDENT по столбцу NAME;
* Индекс EMPLOYEE\_NAME\_INDEX индексирует таблицу EMPLOYEE по столбцу NAME;
* Индекс DISCIPLINE\_NAME\_INDEX индексирует таблицу DISCIPLINE по столбцу DISCIPLINE\_NAME;
* Индекс PULPIT\_NAME\_INDEX индексирует таблицу PULPIT по столбцу PULPIT\_NAME;
* Индекс ADDRESS\_INDEX индексирует таблицу ADDRESS по столбцам REGION, CITY, STREET и BUILDING\_NUMBER;
* Индекс AUDITORIUM\_TYPE\_CAPACITY\_INDEX индексирует таблицу AUDITORIUM по столбцам AUDITORIUM\_TYPE и CAPACITY.

# Процедуры

Выделим следующие процедуры:

* ENROLL\_STUDENT\_IN\_GROUP – эта процедура зачисляет студента в группу. В качестве входных параметров процедура принимает идентификатор студента и идентификатор группы. Процедура обновляет таблицу STUDENT и устанавливает для студента с заданным идентификатором идентификатор группы, указанный в качестве входного параметра;
* EXPEL\_STUDENT\_FROM\_GROUP – эта процедура отчисляет студента из группы. В качестве входного параметра процедура принимает идентификатор студента. Процедура обновляет таблицу STUDENT и устанавливает для студента с заданным идентификатором значение NULL в столбце GROUP\_ID;
* SET\_NEW\_FACULTY\_DEAN – эта процедура устанавливает нового декана факультета. В качестве входных параметров процедура принимает название факультета и идентификатор сотрудника. Процедура обновляет таблицу FACULTY\_MANAGER и устанавливает для факультета с заданным названием идентификатор сотрудника, указанный в качестве входного параметра;
* SET\_NEW\_PULPIT\_MANAGER – эта процедура устанавливает нового заведующего кафедрой. В качестве входных параметров процедура принимает название кафедры и идентификатор сотрудника. Процедура обновляет таблицу PUPLIT\_MANAGER и устанавливает для кафедры с заданным названием идентификатор сотрудника, указанный в качестве входного параметра.

# Функции

Выделим следующие функции:

* GET\_FACULTY\_BY\_GROUP\_ID – эта функция возвращает название факультета по идентификатору группы. В качестве входного параметра функция принимает идентификатор группы. Функция выполняет соединение таблиц SPECIALITY и GROUP по столбцу SPECIALITY\_CODE. Затем функция возвращает название факультета из таблицы SPECIALITY;
* GET\_STUDENTS\_AVERAGE\_MARK – эта функция выполняет соединение таблиц STUDENT, GROUP, MARK и SPECIALITY по столбцам GROUP\_ID, STUDENT\_ID и SPECIALITY\_CODE. Затем функция группирует строки по названию факультета и полному имени студента, а также вычисляет среднюю оценку для каждой группы;
* GET\_EXCELLENT\_STUDENTS – эта функция выполняет соединение таблиц STUDENT и MARK по столбцу STUDENT\_ID. Затем функция группирует строки по полному имени студента и вычисляет среднюю оценку для каждой группы. Функция возвращает только строки, в которых средняя оценка превышает 4,5.

# Представления

Выделим следующие представления:

* FACULTIES\_DEAN – это представление содержит информацию о деканах факультетов. Представление объединяет таблицы FACULTY, FACULTY\_MANAGER и EMPLOYEE по столбцам FACULTY, FACULTY и EMPLOYEE\_ID соответственно;
* PULPITS\_MANAGER – это представление содержит информацию о заведующих кафедрами. Представление объединяет таблицы PULPIT, PULPIT\_MANAGER и EMPLOYEE по столбцам PULPIT, PULPIT и EMPLOYEE\_ID соответственно;
* PULPITS\_EMPLOYEES – это представление содержит информацию о сотрудниках кафедр. Представление объединяет таблицы PULPIT и EMPLOYEE по столбцам PULPIT и PULPIT соответственно.

# Триггеры

Выделим следующие триггеры:

* TR\_STUDENT\_DELETE – этот триггер срабатывает после удаления строки из таблицы STUDENT. Триггер удаляет все оценки студента из таблицы MARK.
* TR\_DELETE\_EMPLOYEE – этот триггер срабатывает после удаления строки из таблицы EMPLOYEE. Триггер удаляет все записи о сотруднике из таблиц FACULTY\_MANAGER и PULPIT\_MANAGER.

# Вывод

Проанализировав имеющееся состояние базы данных университет ьыли выделены на создание объекты, которые были описаны выше. Все эти объекты были описаны для обеих СУБД: SQL Server 2022 и Oracle 19C.