**1 проект (кофепульт):** В первом проекте представлен достаточно классический пример работы реляционной базы данных. Есть 6 обычных таблиц: Admin, Coffee\_shop, Dishes, Order и Client. Также есть вспомогательная таблица DishOrd, необходимая для хранения списка блюд, соответствующих заказу. Проект соответствует 5 нормальным формам. Реляционные базы данных применены разумно (явных противоречий не обнаружено).

**2 проект (котомэйл):** В проекте разумно использованы реляционные БД для работы с администраторами, модераторами и котами. Однако при работе с письмами возникли ярко выраженные недостатки:

1. Для хранения текста письма необходимо использовать документо-ориентированные базы данных, такие как MongoDB и RavenDB. То есть не любая база данных позволяет подобное хранение, что является ограничением при разработке.
2. Коты смотрят письма только в отсортированном формате. Поэтому любой запрос на просмотр писем вызывает их сортировку, что тратит время и мощность сервера. Хранение писем в отсортированном формате в таблице нарушает 1 НФ.
3. Для хранения списка писем каждого пользователя создана дополнительная таблица MailAttachment. Для любого просмотра писем пользователя необходим запрос к этой таблице с условием на MailId.
4. Данные об электронных письмах являются рекурсивными: у каждого из писем-ответов может свой ответ. Поэтому для открытия цепочки писем потребуется множество запросов к таблице Mail. Это требует времени и нагрузки на сервер.

Разумным способом хранения электронных писем является хранение их в формате JSON. В данном случае письма изначально были бы отсортированы в порядке возрастания даты прибытия, и обращение к ним не будет тратить времени и мощности сервера. Также для хранения данных не придется использовать специальные базы данных.

**Вывод:** Не для всех данных имеет смысл использовать реляционные базы данных. Часто имеет смысл совмещать базы данных с более классическими способами хранения информации.