Базы данных

ЛЕКЦИЯ 3

Нормализация. Устранение аномалий и функциональных зависимостей

Алматы, 2023

На прошлой лекции …

- Ранее мы рассматривали анализ предметной области и
- проектирование базы данных с использованием ER-модели

На сегодняшней лекции

- Мы обсудим критерии, используемые для оценки дизайна (проектирования) базы данных.

- Мы сформулируем наше определение «хорошего» дизайна (проектирования) базы данных, используя концепцию функциональных зависимостей.

- Мы рассмотрим, как исправить «плохую» структуру базы данных с помощью процесса нормализации.

«Хороший» проект базы данных

- По вашему мнению какие критерии определяют «Хороший» проект базы данных?

«Хороший» дизайн (проектирование) базы данных

- Хороший дизайн (проектирование) базы данных демонстрирует следующие характеристики:
  - Таблицы содержат только связанные данные
  - Таблицы не хранят избыточных данных
  - Значения NULL сведены к минимуму

Таблицы содержат только связанные данные

- Например: Таблица «Преподаватель» должен содержать только данные о преподавателях

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803

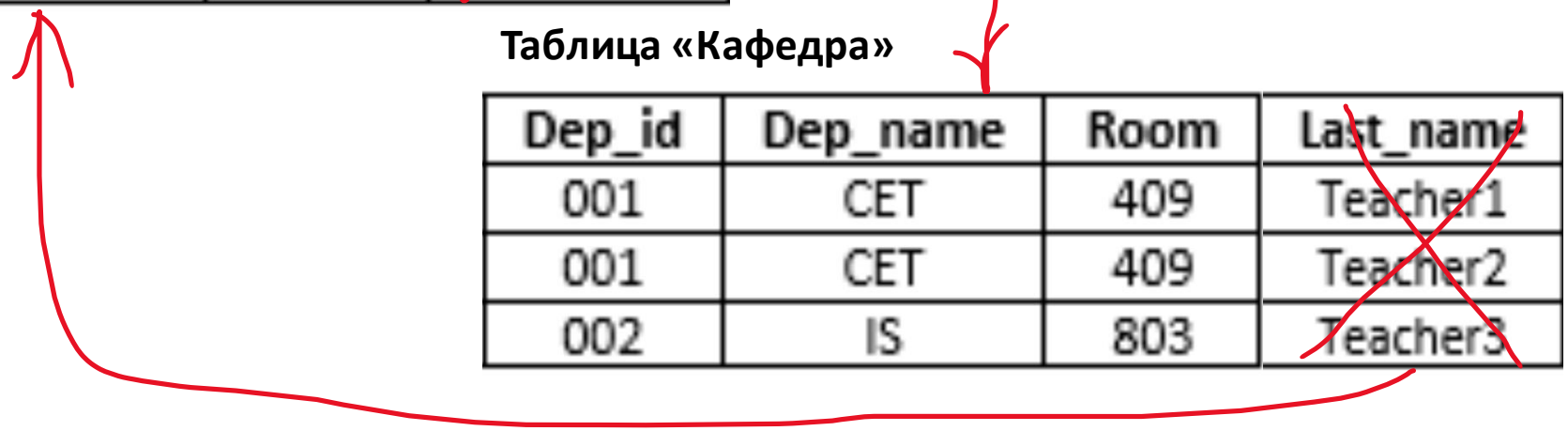
Таблицы не хранят избыточных данных

Таблица «Преподаватель»

Teacher_id	Last_name	Dep_id	Dep_name
001	Teacher1	001	CET
002	Teacher2	001	CET
003	Teacher3	002	IS

Таблица «Кафедра»

Dep_id	Dep_name	Room	Last_name
001	CET	409	Teacher1
001	CET	409	Teacher2
002	IS	803	Teacher3



Значения NULL сведены к минимуму

- Например: Таблица «Преподаватель» должен содержать только данные о преподавателях

Teacher_ID	Last_name	Dep_ID	Position
001	Teacher 1	001	Professor
002	Teacher 2	002	
003	Teacher 3	003	Assistant

Таблица «Преподаватель»

Teacher_ID	Last_name	Dep_ID
001	Teacher 1	001
002	Teacher 2	002
003	Teacher 3	003

Таблица «Должности»

Teacher_ID	Position
001	Professor
002	Tutor
003	Assistant

«Плохой» дизайн базы данных

- Итак, является ли следующая таблица, в которой хранится информация о преподавателях и о кафедрах университета «Плохим» дизайном базы данных?

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803

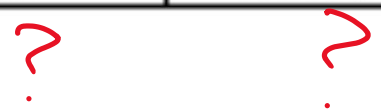
Аномалии обновления (Update anomalies)

- Очевидно, что эта таблица имеет плохое проектирование, потому что она хранит избыточные данные, что является одним из наших критериев при оценке проектирования базы данных.
- Избыточность данных появляется когда одни и те же данные хранятся в базе в нескольких местах, именно это и приводит к аномалиям.
- Избыточность данных, например, в этом отношении (таблице), приводит к трем типам аномалий обновления:
  - Аномалия вставки (Insertion anomaly)
  - Аномалия модификации (Modification anomaly)
  - Аномалия удаления (Deletion anomaly)

Аномалия вставки (Insertion anomaly)

- Аномалия вставки возникает, когда мы не можем вставить некоторые данные в таблицу до тех пор, пока не будут предоставлены другие данные.
- В данном примере возникают два типа аномалий вставки:
  - Чтобы вставить сведения о новом преподавателе в таблицу, мы должны включить сведения о кафедре, на которой должен работать преподаватель.
  - Чтобы вставить в таблицу сведения о новой кафедре, в которой в настоящее время нет преподавателей, необходимо ввести нулевые значения в атрибуты преподавателей, например, Teacher_id. Однако, поскольку Teacher_id является первичным ключом для таблицы, попытка ввести нули для Teacher_id нарушает целостность объекта и не допускается.

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803



Аномалия удаления (Deletion anomaly)

- Аномалия удаления возникает, когда удаление приводит к непреднамеренной потере данных.
- Если мы удалим запись (tuple) из таблицы, о преподавателе кафедры, сведения о кафедре, на которой он работает могут быть потеряны из базы данных.
- Например, если мы удалим кортеж Teacher_id=003, данные, относящиеся к кафедре IS, будут потеряны из базы данных.

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
~~003~~	~~Teacher3~~	~~002~~	~~IS~~	~~803~~

Аномалия модификации (Modification anomaly)

- При аномалии модификации, возможно, что не все данные, которые необходимо изменить в таблице, всегда будут изменены.
- Аномалия модификации обычно приводит к несогласованности данных из-за пропущенных обновлений.
- Например: Если мы хотим изменить значение одного из атрибутов той или иной кафедры в таблице, например номер кабинета для кафедры «СЕТ» мы должны обновить записи всех преподавателей, работающих на этой кафедре.
- Если это изменение не будет выполнено для всех соответствующих записей, база данных станет несогласованной.

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803

Функциональные зависимости

- «Хороший» дизайн (проектирование) таблицы базы данных содержит сгруппированные связанные данные.
- Предыдущий пример был «плохим» дизайном, поскольку в одной таблице хранились два набора связанных данных: информация о преподавателях и кафедрах.

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803

Функциональные зависимости

- Мы можем группировать таблицы, содержащей только связанные данные, используя функциональные зависимости.
- **Функциональные зависимости** описывают связи между атрибутами.
- Другими словами, функциональные зависимости сообщают, какие атрибуты однозначно определяются первичным ключом.

Функциональные зависимости

- Функциональные зависимости не ограничиваются значениями первичного ключа.
- Как правило, функциональная зависимость может использоваться для описания связи между любыми двумя атрибутами (столбцами) в отношении.
- Мы обозначаем функциональные зависимости как

X -> Y,

- когда значение X однозначно определяет значение Y, (где X,Y – набор из одного или нескольких атрибутов).

Степени функциональной зависимости. Классификация

- Различают следующие степени функциональной зависимости между атрибутами:
- **полная зависимость (Full dependency)**
- **частичная зависимость (Partial dependency)**
- **транзитивная зависимость (Transitive dependency)**

Полная функциональные зависимости

- **Полная функциональная зависимость** X -> Y существует в таблице, если нет атрибута A, который можно удалить из X, и при этом зависимость все еще будет сохраняться.
- Рассмотрим таблицу

  Студенты (student_id, last_name, birthdate, gpa).

- В этой таблице существует функциональная зависимость
- **(FD) : {student_id} -> {last_name, birthdate, gpa}**

  X У
- Это полная зависимость, потому что ни один атрибут не может быть удален с левой стороны X, и зависимость сохраняется.
- Так как student_id — это PK (первичный ключ), мы можем использовать его для уникальной идентификации каждого студента Фамилия, дата рождения и средний балл **зависят от PK.**

Частичные функциональные зависимости

- Противоположностью полной зависимости является частичная зависимость.

- Зависимость X -> Y является частичной зависимостью, если существует атрибут A, являющийся частью X, который можно удалить из X, и при этом зависимость все еще сохраняется.

Составной первичный ключ

- **Составной первичный ключ** — первичный ключ, состоящий из двух или более атрибутов.

- **Первичный ключ** определяется как ключ или столбец базы данных, который однозначно идентифицирует каждую строку в таблице базы данных.

Частичные зависимости

- Давайте рассмотрим таблицу «Преподаватели и дисциплины».

Teacher_id	Last_name	Course_id	Course_name	Credits
001	Teacher1	001	SDP1	3
001	Teacher1	002	SDP2	3
002	Teacher2	001	SDP1	3

- Предположим, что PK — это {teacher_id, course_id} – составной ключ.
-

Частичные зависимости

- Также предположим, что эта таблица имеет следующие зависимости:
- **FD1:** {teacher_id, course_id} -> {last_name, course_name, credits}
- **FD2:** {teacher_id} -> {last_name}
- **FD3:** {course_id} -> {course_name, credits}

- **FD1** является частичной зависимостью
- **FD2 и FD3** являются полными зависимостями.

Транзитивные зависимости

- Предположим, что X, Y и Z — столбцы в таблице R.
- Также предположим, что X -> Y и Y -> Z являются зависимостями в R.
- Эта зависимость является транзитивной.

- **Транзитивная зависимость** существует в таблице, если существуют множества одного или нескольких атрибутов X, Y и Z и существуют следующие зависимости: $X \rightarrow Y$ и $Y \rightarrow Z$.

Пример транзитивной зависимости

- Предположим, у нас есть следующая таблица, в которой хранится информация о студентах и их группах.
- PK — student_id

Student_id	Last_name	Group_id	Group_name
001	Student1	001	Group1
002	Student2	001	Group1
003	Student3	002	Group2

Пример транзитивной зависимости

- Таблица имеет следующие функциональные зависимости:
- **FD1:** {student_id} ->
  {last_name, group_id, group_name}
- **FD2:**{group_id} ->
  {group_name}
- Эта таблица содержит транзитивную зависимость
  {student_id} → {group_name}

потому что

{student_id} -> {group_id} -> {group_name}

Декомпозиция

- **Декомпозиция** – это процесс разбиения целого на части или элементы.
- Декомпозиция позволяет разбить таблицу на несколько таблиц в базе данных.
- Разбиение на подтаблицы должны происходить без потери данных, потому что информация в исходной таблице может быть точно восстановлена на основе декомпозированных таблиц.

Спасибо за внимание!