**БД 01.10.25**

**Нормализация БД**

Нормализация основана на анализе функциональных зависимостей между атрибутами отношения

Функциональная зависимость: X → Y

Подмножество атрибутов X – детерминант функциональной зависимости.

Пример:

Сотрудники (код, ФИО, пол, дата рождения).

Код → ФИО

Код → дата рождения

Код → пол

Код → (ФИО, дата рождения)

Каждому значению кода соответсвует одно значение. ФИО и дата рождения взаимно независимые.

**Математические свойства функциональных зависимостей (правила Армстронга):**

1. Рефлективность – если y являеться подмножеством x то x определяет y, при этом такая зависимость – тривиальная. Y c X => x → y

2. Дополнение – A → B => AC → BC

3. Транзитивность – A → B и B → C => A → C.

4. Самоопределение – X → X

5. Декомпозиция и композтция:

A → BC => A → B, A → C

A → B и C → D => AC → BA

6. Теорема всеобщего объединения – если A → B и C → D => A объединение (C – B) → BD

Множество всех функциональных зависимостей – **замыкание**.

**Непреводимым** – множество зависимостей в котором не одна зависимость не может быть выведена из другой.

**Свойства непреводимых зависимостей:**

1. В каждой части зависимости должн быть только один атрибут

2. Из левой части зависимости нельзя удалить ни одного атрибута без потери этой зависимости

**Нежелательные зависимости.** Некоторые зависимости входящие в фз язвляються не желательными, так как приводять к дублированию информации, что создает проблемы обновления данных и предстовляет угрозу целостности бд.

Пример:

Сотрудники (код, ФИО, пол, дата рождения, возрастная категория). Добовление **вк** приведет к новой зависимости.

(Дата рождения, пол) → Возрастная категория

Можно онтнести к нежелательным: много дубликатов, нужно выполнять перещет.

**Декомпозия без потерь (теорема Хеза)**

**Декомпозиция отношения** – это взятие одной или несколько проэкций исходного отношения, так чтобы эти проэкции содержали возможно с повторениями все атрибуты исходного отношения. В декомпозиции не должны быть потеряны данные.

Пусть дано отношение R(A, B, C) – если имееться функциональная зависимость атрибутов, то проэкции R1(A, B) и R2(A, C) образуют декомпозицию без потерь. При этом информация потеряна не будет.

**Нормализация** – обратимый пошаговый процесс декомпозиции отношений, с устранением нежелательных **фз**.

**Формы нормализации:** (каждая форма лучще предыдущей)

1. Отношение находиться в 1 форме если: Значение его всех атрибутов атомарны и остутсвуют повторяющиеся группы атрибутов.

Доходы сотрудников (код, фио, доход за январь…, доход за декабрь...)

2. Первичный ключь может быть составным атрибутом.

**Полная фз** – неключевой атрибут функционально полно зависит от составного ключа, если он он зависит от всего ключа в целом, но не находиться в зависимости от подмножества атрибутов составного ключа.

3. x → y, y → z, z транзитивно зависим от x. Отношение в 3 когда оно в 2 и каджый не ключевой атрибут нетразитивно зависит от первичного ключа

Нормальная форма (BCNF) – отношение в bcnf если оно в 3 форме, и в ней отсутвует зависимости атрибутов первичного ключа от неключевых атрибутов. Ситуация когда отношение в 3 форме но не в bcnf возникает при условии, что отношение имеет, 2 или более возможных ключа, которые состовные и имеют общие атрибуты.

4. Касаеться того где есть повторяющиеся наборы данных. Нарушение 4 формой в таблицах связках, в которых более 2 сущностей.

Пример:

Расписание (день недели, номера пары, аудитория, преподователь, предмет)

(день недели, пара, аудитория)

(день недели, пара, преподователь)

(день недели, пара, предмет)

Отношение в 4 форме если оно в bcnf и в нем отсутвуют многозначные зависимости.

5. Отношение в 5 форме тогда и только тогда, когда любая зависимость по соединению в нем определяеться только его потенциальными ключами.

**Денормализация**

Недостатки:

1. Неэкономное использование диска

2. Прблемы с обновлением данных в виду нарушения принциа каждый факт в одном месте

3. Проблемы контроля целостности.

Недостаток нормализации – замедление работы СУБД при выполнении запросов на выборку данных. В нормализованой бд практичиски каждый запрос требует соединения данных из нескольких таблиц. А соединение таблиц требует значительных ресурсов.

Денермализация – это процесс модификации структуры таблиц нормализованной бд с целью повышения производительности, засчет допущения некоторой управляемой избыточности данных.

**SQL**

DDL – работа со структурой бд, укладываються в:

CREATE

ALTER → объект <имя, объекта> [набор данных]

DROP

**Основные объекты дб**

1. База данных – контейнер который содержит, таблици и другие объекты.

2. Схема – часть базы данных хв пределах которой, все имена объектов должны быть уникальны.

3. Таблица – основной объект бд. В отличии от релеационной теории в стандарте sql, таблица определяеться как мультимножество строк.

4. INDEX – вспмогательный объект, для ускарения операции поиска данных, однако при этом замедляет операции вставки, удаления, обновления строк таблици. В качестве структуру данных для хранения индексов, используеться B-tree (сильно ветвляшееся дерево, B дерево). Для потенциальных ключей таблици, автоматически создаються индексы.

5. VIEW – представление, именной запрос на выборку, который храниться в бд, и выполняеться на сервере при любом обращении по имени, создавая виртуальную таблицу с отобранными данными.

6. Хранимые процедуры или функции – данные объекты, пишуться на процедурном расширении языка sql, который дополняет его такими управляющими консрукциями как ветвление и циклы и позволяет реализовать любые алгоритымы обработки данных.

7. Тригер( TRIGGER ) - особый вид хранимой процедуры, который срабатывает автоматически при наступлении определенных событий в бд. Основными событиями являються вставка обновление, удаление строк.

8. Последовательность ( SEQUENCE ) - объект для генерации уникальных значений, например сурогатных первичных ключей таблици.

9. USER, ROLE – пользователи, которые хранять пользователей и их права на выполнение действий в бд.

Пример:

CREATE table <имя таблици>

(список столбцов, [список ограничений таблици])

имя\_столбца

тип\_столбца

[DEFAULT значение]

[список ограничений столбца]

Имя столбца должно быть уникально.

**Типы данных:**

1. Числовые – 2 SMALLINT, 4 INTEGER (INT), 8 BIGINT

2. NUMERIC (P, S) – числа с фиксированной точкой, p – разряды, s – количество знаков после запятой.

3. FLOAT, REAL, DOUBLE , PRECISION – с плавающей точкой.

4. Строки – CHAR(n), VARCHAR(n), varchr байт столько сколько n, varchar память выделяеться автоматически, храниться под реальные значение текста + 4 байта под хранение размера текста.

5. TEXT

6. NCHAR (oracle) – строки в национальных кодеровках.

7. DATE, TIME, DATETIME, TIMESTAMP(с точностью до милесекунд).

8. BLOB – для хранения больших объектов, позволяет хранить в бд неструктурированную информацию.

9. JSON

**Ограничения**, позволяют создаватьдополнительные условия проверки вводимых данных которые субд проверяет автоматически. Механизм ограничений позволяет поддерживать данные, в не протеворечивом состоянии, соответсвующих бизнес правилам предметной области, ограничения могут разделяться на ограничения таблици и столбца.

**Ограничения столбца:**

1. NULL, NOT NULL – допустимо ли в значении неопределенные значения, по умалчанию NULL.

2. PRIMARY KEY – будет проверяться на уникальность, в таблице может быть только один.

3. UNIQUE – ограничение уникальности.

4. REFERENCE ON <имя таблици> – ограничение внешнего ключа, для значения ключа втоматически проверяеться существование главного первичного ключа главной таблици.

5. CHECK – создание кастомных ограничений столбца.

**Ограничение таблици:**

1. PRIMARY KEY – когда составной первичный ключ.

2. UNIQUE

3. FOREIGN KEY

4. CHECK

CREATE table as <запрос на выборку> - создание таблиц на основе других таблиц.

CREATE table copy-students as SELECT \* FROM students

INSERT

UPDATE

DELETE

SELECT

INSERT INTO <имя таблици> (список столбцов) VALUES (список значений)

INSERT INTO <имя таблици> (список столбцов) AS SELECT (запрос на выборку) – при выполнении команды ограничений проверяються для каждой вставляемой строки, и в случае их нарушения хотябы одной строк, все данные не будут вставлены.

TRUNCATE <имя таблици> - очистить таблицу.

UPDATE <имя таблици> SET <имя столбца1> <значение1> … <имя столбца n> <значение n>

WHERE <условие отбора строк>

SELECT – команда выборки, реализует все аперации реляционной алгебры, результатом всегда являеться новая таблица, при этом она может быть пустой, эта новая таблица являеться виртуальной, тоесть содержиться в оперативной памяти и при выполнении любой сколь угодно сложной команды выборки состояние базы данных не меняеться поскольку в результате выполнения команды, выполняеться толко операции чтения и обработки данных.

SELECT [DISTINCT] <список выражений> / \*

FROM <список таблиц / представление запросов>

[JOIN … ]

[WHERE <условие отбора строк> ]

[GROUP BY <список выражений для группировки>]

[HAVING <условие отбора групп с агригатными выражениями>]

[ORDER BY <список выражений для сортировки>] - нужен для сортировки, идет последним сортировку можно выполнять по всем типам данных кроме blob, сортировка может быть выполнена по нескольким столбцам. Строки сортируються в лексикографическом порядке.