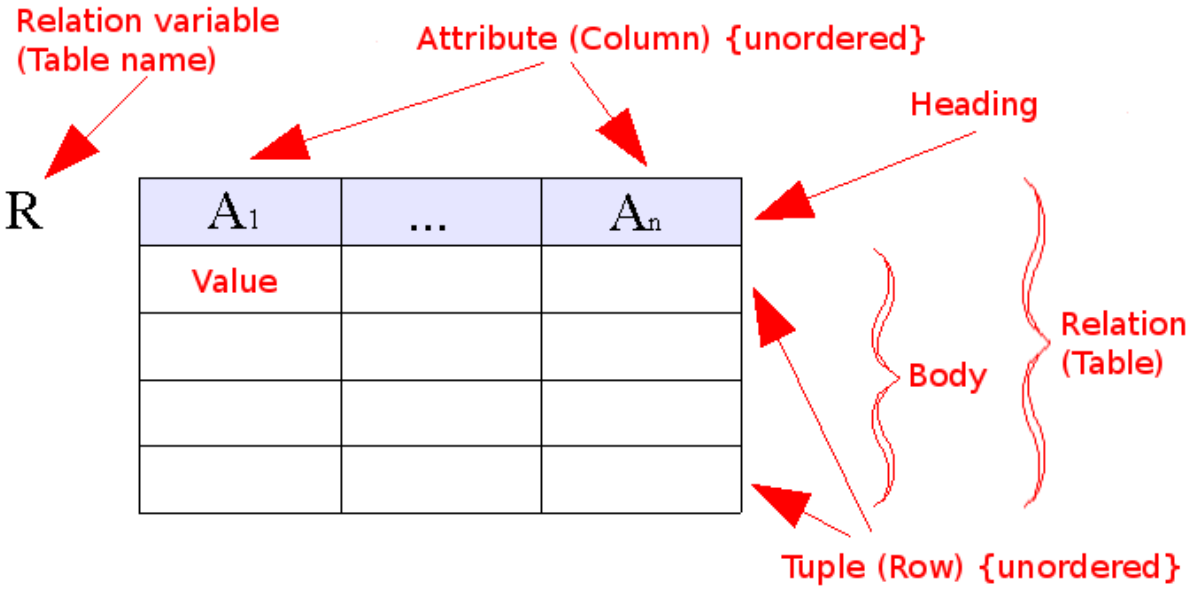
Базы данных (ЛР 1)

Совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения.

Реляционная модель

The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.



The fundamental assumption behind a relational model is that all data is represented as mathematical n-ary relations, an n-ary relation being a subset of the Cartesian product of n domains.

Database normalization

is the process of structuring a relational database in accordance with a series of so-called normal forms to reduce data redundancy and improve data integrity.

это процесс структурирования реляционной базы данных в соответствии с нормальными формами, которые уменьшают избыточность данных и улучшает целостности данных.

Normal Forms

Informally, a relational database relation is often described as "normalized" if it meets third normal form. Most 3NF relations are free of insertion, updating, and deletion anomalies.

Constraint (informal description in parentheses)	UNF (1970)	1NF (1970)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Unique rows (no duplicate records)[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Scalar columns (columns cannot contain relations or composite values)[5]	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-prime attribute has a full functional dependency on a candidate key (attributes depend on the *complete* primary key)[5]	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (attributes depend *only* on the primary key)[5]	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute (a stricter form of 3NF)	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	—
Every non-trivial functional dependency begins with a superkey (a stricter form of 3NF)	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	—
Every non-trivial multivalued dependency begins with a superkey	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	—
Every join dependency has a superkey component[8]	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	—
Every join dependency has only superkey components	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	—
Every constraint is a consequence of domain constraints and key constraints	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Every join dependency is trivial	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Реляционные базы данных представляют собой базы данных, которые используются для хранения и предоставления доступа к взаимосвязанным элементам информации. Реляционные базы данных основаны на реляционной модели — интуитивно понятном, наглядном табличном способе представления данных. Каждая строка, содержащая в таблице такой базы данных, представляет собой запись с уникальным идентификатором, который называют ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных.

ANSI-SPARC Architecture

is an abstract design standard for a database management system.

Three-level architecture

The objective of the three-level architecture is to separate the user's view:

- It allows independent customized user views: Each user should be able to access the same data but have a different customized view of the data. These should be independent: changes to one view should not affect others.
- It hides the physical storage details from users: Users should not have to deal with physical database storage details.
- The database administrator should be able to change the database storage structures without affecting the users' views.
- The internal structure of the database should be unaffected by changes to the physical aspects of the storage: For example, a changeover to a new disk.

The three levels are:

- External Level (User Views): A user's view of the database describes a part of the database that is relevant to a particular user. It excludes irrelevant data as well as data which the user is not authorised to access.
- Conceptual Level: The conceptual level is a way of describing what data is stored within the whole database and how the data is inter-related. The conceptual level does not specify how the data is physically stored. Some important facts about this level are:
    - DBA works at this level.
    - Describes the structure of all users.
    - Only DBA can define this level.
    - Global view of database.
    - Independent of hardware and software.
- Internal Level: The internal level involves how the database is physically represented on the computer system. It describes how the data is stored in the database and on the computer hardware.

Архитектура систем БД ANSI/SPARC описывает логическую организацию системы с точки зрения представления данных пользователям и включает три уровня: внутренний, концептуальный и внешний.

• *внутренний уровень* – уровень, наиболее близкий к физическому хранению;

• *внешний уровень* наиболее близок к пользователям, он связан со способами представления данных для отдельных пользователей;

• *концептуальный уровень* – промежуточный уровень между двумя первыми, на котором представлены все имеющиеся данные, но в более абстрактном по сравнению с внутренним уровнем виде.

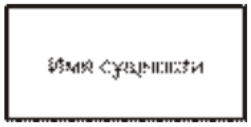
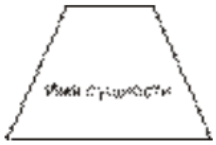
Сущность

Стержневая сущность.
Стержневая (сильная) сущность – независящая от других сущность. Стержневая сущность не может быть ассоциацией, характеристикой или обозначением (см. ниже).

Ассоциация

Ассоциативная сущность (или ассоциация) выражает собой связь «многие ко многим» между двумя сущностями. Является вполне самостоятельной сущностью. Например, между сущностями МУЖЧИНА и ЖЕНЩИНА существует ассоциативная связь, выражаемая ассоциативной сущностью БРАК.

Характеристика
Характеристическую сущность еще называют слабой сущностью. Она связана с более сильной сущностью связями «один ко многим» и «один к одному». Характеристическая сущность описывает или уточняет другую сущность. Она полностью зависит от нее и исчезает с исчезновением последней. Например, сущность Зарплата является характеристикой конкретных работников предприятия и не может в таком контексте существовать самостоятельно – при удалении экземпляра сущности Работника должны быть удалены и экземпляры сущности Зарплата, связанные с удаляемым работником.

Изображение	Комментарий
	Так на диаграмме изображается сильная сущность[1].
	Так на диаграмме изображается сущность, отличная от сильной сущности, без уточнения ее типа.
	Атрибут сущности. Вместо имени атрибута можно указать многоточие «…», что будет обозначать группу атрибутов.
	Атрибут сущности, являющийся первичным ключом.
	Обозначает связь, между двумя сущностями.
	Ассоциативная сущность (связь «многие ко многим»).
	Обозначение сущности вида «характеристика».
	Показывает сущность вида «обозначение».
	Прямая линия указывает на связь между сущностями, либо соединяет сущность и атрибут. Линия со стрелкой указывает направленную связь.

SQL

DDL

DDL, или Data Definition Language — это группа команд, которые используются для создания и изменения структуры объектов базы данных: таблиц, представлений, схем и индексов.

Наиболее известные команды SQL DDL — CREATE, ALTER, DROP.

DML

DML, или Data Manipulation Language — это группа операторов, которые позволяют получать и изменять записи, присутствующие в таблице.

DCL

DCL, или Data Control Language — это команды SQL, которые используют для предоставления и отзыва привилегий пользователя базы данных. При этом пользователь не может откатить изменения. Рассмотрим наиболее известные команды: GRANT и REVOKE.

TCL

TCL, или Transaction Control Language — одни из наиболее популярных команд SQL. Их используют для обеспечения согласованности базы данных и для управления транзакциями.

Транзакция — это набор SQL-запросов, выполняемых над данными, которые объединены в атомарную секцию. Это значит, что промежуточные результаты операции не видны для других конкурирующих транзакций — и вся секция будет либо выполнена, либо полностью отменена в случае ошибки.

Примеры команд: BEGIN/COMMIT, ROLLBACK.

Список литературы

https://al.cs.msu.ru/system/files/ER_method.pdf

https://picloud.pw/media/resources/posts/2018/02/20/%D0%9A%D0%B8%D1%80%D0%B8%D0%BB%D0%BB%D0%BE%D0%B2_%D0%92.%D0%92_%D0%93%D1%80%D0%BE%D0%BC%D0%BE%D0%B2_%D0%93.%D0%AE._-_%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2_%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_-_2008.pdf

https://www.cs.uct.ac.za/mit_notes/database/htmls/chp06.html

http://citforum.ru/database/dbguide/2-3.shtml

http://ssofta.narod.ru/bd/24.htm

ЛР2

Alter

Select

aggregate functions

MATERIALIZED VIEW/VIEW

group by

join

where

cast int to text (43::'42')

on delete/update cascade/restrict/default/set null

domains

enums

arrays[]

sequence

serial vs uuid

varchar max size

POINT

how to count distance (<->)

как обновить строчку фулл

как сделать строчку foreign key

как primary key

в каком порядке работают запросы (where before select as e.t.c.)

DML

DDL

DCL

сущность-связь

IF NOT EXISTS

ON CONFLICT

ASCENDING/DESCENDING

ORDER BY

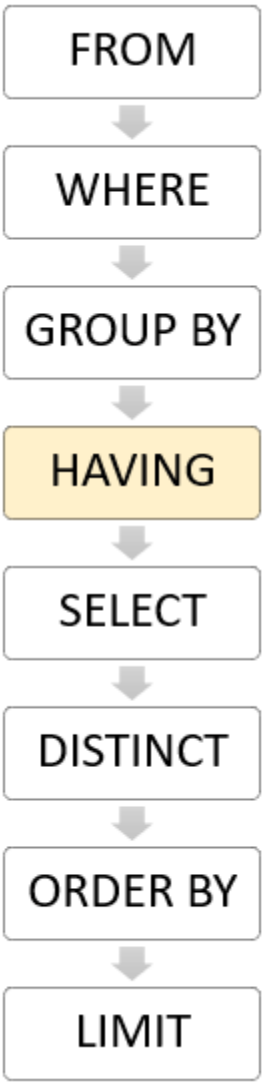
подзапросы, виды подзапросов, коррелирующие, скалярные, etc

те же views, работа с ними

left, right, cross join, etc

having, можно ли написать после селекта и почему

аналогично для where



Нормальные Формы

Нормализация отношений в базе данных — это процесс организации таблиц и связей между ними, чтобы минимизировать избыточность данных и устранить аномалии при обновлении, вставке и удалении записей. Это делается путем разделения таблиц на более мелкие и связывания их между собой посредством внешних ключей.

1NF (первая нормальная форма): в таблице нет повторяющихся групп значений, и каждая ячейка содержит только одно значение. Это означает, **что таблицы не могут содержать массивы или списки значений.**

2NF (вторая нормальная форма): таблица находится в 1NF и каждый столбец зависит только от первичного ключа. Если в таблице есть составной первичный ключ, то каждый столбец должен зависеть от всех столбцов в составном ключе.

каждый неключевой атрибут зависит от (каждого) её потенциального ключа.

3NF (третья нормальная форма): таблица находится в 2NF и каждый столбец не зависит от других неключевых столбцов. Если столбец зависит от неключевого столбца, то он должен быть вынесен в отдельную таблицу.

ни один неключевой атрибут R не находится в транзитивной функциональной зависимости от потенциального ключа R.

BCNF (нормальная форма Бойса-Кодда): таблица находится в 3NF, и каждая функциональная зависимость в таблице должна быть определена только ключом, а не другими неключевыми столбцами. (Частный случай 3NF когда первичный ключ составной)

левая часть (детерминант) должна быть потенциальным ключом отношения

for each relation X->Y X should be super key

4NF (четвертая нормальная форма): таблица находится в BCNF и **не имеет многозначных зависимостей**.

Многозначная зависимость возникает, когда в таблице **имеется некоторый столбец, который может иметь несколько значений для одного и того же значения первичного ключа**.

5NF (пятая нормальная форма): таблица находится в 4NF и не имеет зависимостей на основе многозначных фактов.

It is in 5NF if every non-trivial join dependency in that table is implied by the candidate keys.

Isolating semantically related multiple relationships

6NF Требование шестой нормальной формы заключается в том, что таблица должна удовлетворять всем нетривиальным зависимостям соединения. Из этого определения следует, что таблица находится в 6NF, когда она неприводима, то есть **не может быть подвергнута дальнейшей декомпозиции без потерь.**

Представления

View

Механизм, представляющий из себя именованный запрос. В момент каждого обращения к представлению, исполняется соответствующий ему SQL-запрос. При этом, представления могут использоваться как таблицы, на пример, в select ... from.

Materialized view

является «слепком» данных на определенный момент времени, который хранящимся физически в виде отдельной таблицы и не изменяется при изменении базовой версии.

Функциональные зависимости

это связь, которая может возникнуть между сущностями, хранящимися в базе данных. Если сущность **A** функционально определяет сущность **B**, то такую зависимость принято обозначать следующим образом:

$$A \rightarrow B$$

здесь

- **A** – детерминант отношения;
- **B** – зависимая часть.

для каждого из различных значений поля А обязательно существует только одно из различных значений поля Б.

Частичная зависимость
Эта зависимость может возникать в случаях, когда таблица содержит составной ключ. Составной ключ — это ключ таблицы, который состоит из нескольких атрибутов. Если ключ состоит из одного атрибута, то этот ключ является простым. При частичной зависимости один атрибут таблицы является зависимым от части ключа, то есть от отдельного атрибута, входящего в ключ отношения;

Полная зависимость
это случай, когда между двумя атрибутами A и B является прямая ($A \rightarrow B$) и обратная ($B \rightarrow A$) зависимость. При полной функциональной зависимости одному значению атрибута A соответствует только одно значение атрибута B. И, наоборот, одному значению атрибута B соответствует значение атрибута A.
Полная функциональная зависимость между двумя атрибутами A и B обозначается $A \leftrightarrow B$.

Транзитивная зависимость
Это зависимость, когда два атрибута связаны между собой через третий атрибут. Этот третий атрибут выступает посредником;

Многозначная зависимость
Это случай, когда одному значению одного атрибута соответствует несколько значений другого атрибута.