Базы данных

A10. Основы инженерии данных

Московский государственный технический университет
имени Н.Э. Баумана

Факультет ИБМ

Апр 2025 года

Москва

Артемьев Валерий Иванович © 2025

Инженерия данных

Инженерия данных (Data Engineering) наряду с анализом данных (Data Analysis) и исследованием данных (Data Science) не столько появилась, сколько **специализировалась**

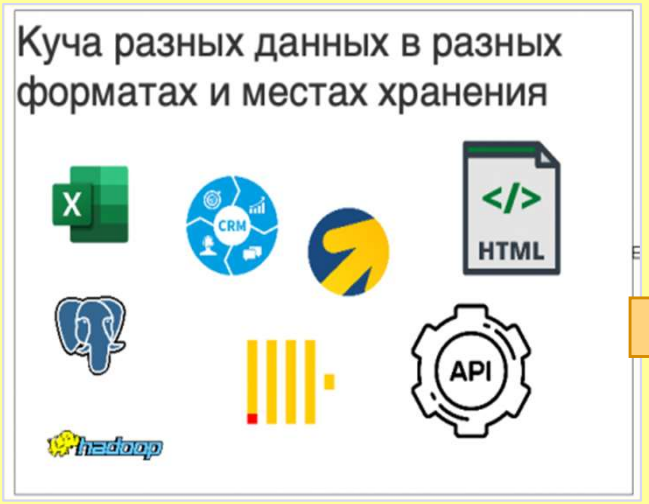
- с ростом потребности в *принятии решений на основе фактических данных высокого качества*,
- с появлением *возможности обрабатывать данные разной природы в разных форматах*,
- *с совершенствованием предсказательных способностей методов и средств искусственного интеллекта*,
- *с развитием оперативных аналитических систем*
- *и систем анализа больших данных.*

Инженерия данных *занимается вопросами подготовки данных* для их анализа и исследования, что может составлять *до 80% общей трудоёмкости*.

Роль инженера данных и его оттенки

- Роль инженера данных *в широком смысле – набор знаний и навыков специалиста ИТ*, который *удовлетворяет информационные потребности* аналитиков и исследователей данных в масштабах *корпорации*.
- *В узком смысле* это несколько *сокращённая роль представителя бизнеса* в контакте с аналитиками и исследователями данных от бизнеса в лаборатории (песочнице) данных *в режиме самообслуживания*.
- *Можно совмещать несколько ролей*, но этот случай не такой частый.

Что делает инженер данных?



Инженеры данных

- Профилирование источников данных
- Сбор / извлечение данных
- Обработка событий, потоков и очередей
- Преобразование данных
- Контроль и очистка данных
- Измерение качества данных
- Интеграция данных
- Создание витрин и аналитических наборов данных

КХД

Витрины данных

Аналитические наборы данных

Аналитики данных

Панели мониторинга и отчёты

Исследователи данных

Искусственный интеллект

Дорожная карта инженера данных от бизнеса

Архитектура данных:
- модели и потоки данных
- нотации моделей и потоков данных
- типовые структуры
- канонические модели

Хранение данных:
- файлы и форматы данных
- реляционные БД
- распределённые файловые системы
- нереляционные БД
- среда Big data

Программирование:
- языки и среды программирования
- языки запросов
- командные языки
- управление версиями

Интеграция данных:
- извлечение, преобразование загрузка данных
- конвейер данных
- управление справочниками и реестрами

Интер-операбельность:
- сбор данных
- коннекторы и адаптеры
- API и web-сервисы
- шины данных
- очереди сообщений

Управление качеством данных:
- профилирование источников данных
- контроль и очистка данных
- измерение качества данных

Дорожная карта навыков инженера данных от бизнеса

Архитектура данных:
- ER-диаграммы
- UML, Data Flow
- ROLAP снежинка и звезда
- EVA, Data Vault,
- Anchor
- ERwin

Хранение данных:
- CSV, XLS, DBF, PDF, HTML, XML, JSON
- БД SQL
- HDFS, Parket, AVRO
- MongoDB, HBase, Hive, Redis
- ArenaData

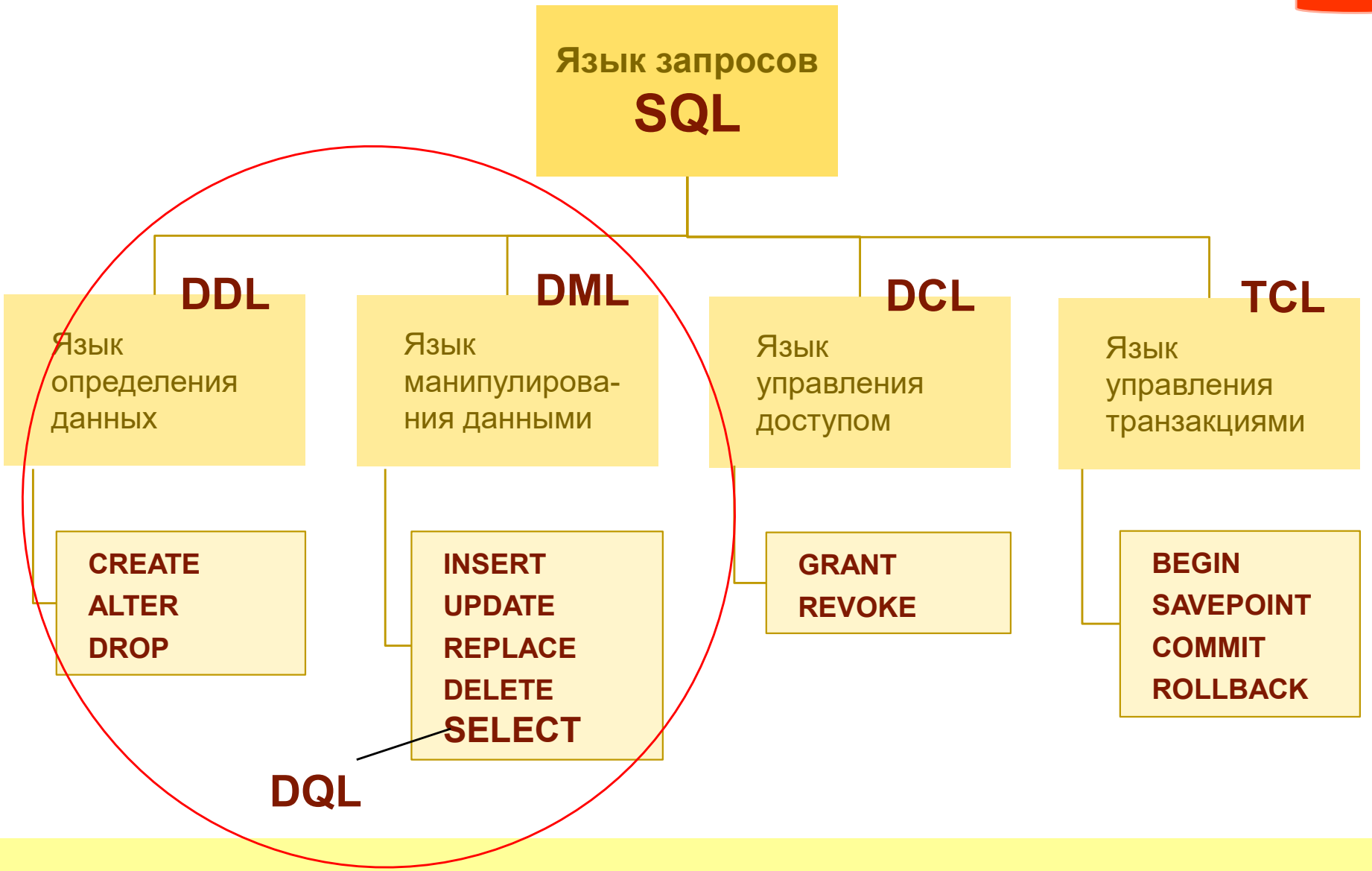
Программирование:
- Python
- Jupyter Hub
- QBE
- SQL, Dbt
- DAX
- Bash / PowerShell
- git hub / git lab

Интеграция данных:
- ETL / ELT-решения
- AirFlow
- MDM-решения
- Data Warehouse
- Date Lake
- витрины данных
- аналитические наборы данных

Интер-операбельность:
- ODBC
- JDBC
- REST API
- SOAP
- RabbitQ

Управление качеством данных:
- профилирование источников данных
- контроль и очистка данных
- измерение качества данных

Язык SQL для инженера данных



Инструкции CREATE для создания таблицы

Инструкции CREATE служит для создания объектов БД: таблиц (TABLE), представлений (VIEW) –виртуальных таблиц, а также индексов (INDEX) для ускорения выполнения операций. Их синтаксис отличается в зависимости от типа объекта.

CREATE TABLE *название_таблицы*

(*название_столбца1 тип_данных свойства_столбца1*,

название_столбца2 тип_данных свойства_столбца2,

.....

название_столбцаN тип_данных свойства_столбцаN,

свойства_уровня_таблицы

);

Пример создания таблицы

Создаём таблицу пользователей Users с четырьмя столбцами.
Поле id – первичный ключ с автоувеличением значения, столбец email объявлен уникальным:

```
 CREATE TABLE Users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
   LFname VARCHAR(50),
   BirthDay DATE,
   email VARCHAR(30) UNIQUE
);
```

Можно задать свойства первичного ключа и уникальности в виде свойств таблицы:

```
CREATE TABLE Users
(id INTEGER AUTOINCREMENT,
   LFname VARCHAR(50),
   BirthDay DATE,
   email VARCHAR(30) UNIQUE,
PRIMARY KEY (id),
UNIQUE (email)
);
```

Пример явного задания ограничений

Зададим ограничения NOT NULL и проверки CHECK :

```
 CREATE TABLE Users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
    LFname VARCHAR(50) NOT NULL CHECK(LFname !=''),
    BirthDay DATE NOT NULL CHECK(BirthDay<Now()),
    email VARCHAR(30) UNIQUE
);
```

Можно задать ограничения-проверки на уровне таблицы:

```
CREATE TABLE Users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
    LFname VARCHAR(50 NOT NULL,
    BirthDay DATE NOT NULL,
    email VARCHAR(30) UNIQUE,
CHECK ((LFname !='' AND (BirthDay<Now())
);
```

Пример задания именованных ограничений

Можно задать именованные ограничения-проверки на уровне таблицы:

```
CREATE TABLE Users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
    LFname VARCHAR(50 NOT NULL,
    BirthDay DATE NOT NULL,
    email VARCHAR(30) UNIQUE,
    CONSTRAINT users_pk PRIMARY KEY(id),
    CONSTRAINT user_email_uq UNIQUE(email),
    CONSTRAINT user_age_chk CHECK(BirthDay<Now()))
);
```

Впоследствии через эти имена мы сможем управлять ограничениями - удалять или изменять их.

Кстати, при создании таблицы можно поставить проверку, создана ли уже таблица, или проверку наличия таблицы с таким именем:

```
CREATE TABLE IF NOT EXISTS Users …
```

Объявление внешних ключей и ограничений ссылочной целостности

Для создания ограничения внешнего ключа после **FOREIGN KEY** указывается столбец таблицы, который будет представляет внешний ключ. А после ключевого слова **REFERENCES** указывается имя связанной таблицы, а затем в скобках имя связанного столбца, на который будет указывать внешний ключ. После выражения **REFERENCES** идут выражения **ON DELETE** и **ON UPDATE**, которые задают действие при удалении и обновлении строки из главной таблицы соответственно.

```
CONSTRAINT имя_ограничения
FOREIGN KEY (столбец1, столбец2, ... столбецN)
REFERENCES главная_таблица
(столбец_главной_таблицы1,
столбец_главной_таблицы2, ...
столбец_главной_таблицыN)
ON DELETE действие
ON UPDATE действие
;
```

Пример ограничений ссылочной целостности

Определим главную таблицу Companies и зависимую таблицу Users. Таблица Users связана с таблицей Companies, company_id является **внешним ключом**, который указывает на столбец id из таблицы Companies:

```
CREATE TABLE Companies
(   id INTEGER PRIMARY KEY AUTOINCREMENT,
    cname varchar(100) NOT NULL
);
CREATE TABLE Users
(   id INTEGER PRIMARY KEY AUTOINCREMENT,
    LFname VARCHAR(50) NOT NULL,
    BirthDay DATE NOT NULL,
    company_id INTEGER,
    FOREIGN KEY (company_id)  REFERENCES Companies (id)
);
```

Можно задать имя ограничения:

```
CREATE TABLE Users
(   …
CONSTRAINT users_companies_fk
    FOREIGN KEY (company_id)  REFERENCES Companies (id)
);
```

Действия при удалении и изменении связанной строки главной таблицы

С помощью выражений **ON DELETE** и **ON UPDATE** можно установить действия, которые выполняются соответственно при удалении и изменении связанной строки из главной таблицы:

- **CASCADE** – автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
- **SET NULL** – при удалении или обновлении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение **NULL**. (столбец внешнего ключа должен поддерживать NULL).
- **RESTRICT** – отклоняет удаление или изменение строк в главной таблице при наличии связанных строк в зависимой таблице.
- **NO ACTION** – то же самое, что и RESTRICT.
- **SET DEFAULT** – при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, заданное с помощью свойства DEFAULT.

Пример действий при удалении связанной строки главной таблицы

Каскадное удаление позволяет при удалении строки из главной таблицы автоматически удалить все связанные строки из зависимой таблицы.
Для этого применяется свойство **CASCADE**:

```
CREATE TABLE Users
(   id INTEGER PRIMARY KEY AUTOINCREMENT,
    LFname VARCHAR(50) NOT NULL,
    BirthDay DATE NOT NULL,
    company_id INTEGER,
     FOREIGN KEY (company_id)  REFERENCES Companies (id)
     ON DELETE CASCADE
);
```

Подобным образом работает и **ON UPDATE CASCADE**. При изменении значения первичного ключа автоматически изменится значение связанного с ним внешнего ключа. Однако поскольку первичные ключи изменяются очень редко, то на практике выражение **ON UPDATE** используется редко.

Инструкции ALTER для изменения описания таблицы

Инструкции ALTER TABLE служит для изменения описания таблицы – добавления, удаления, переименования полей, а также изменения их типов:

```
ALTER TABLE название_таблицы
оператор_изменения COLUMN название_столбца1
тип_данных  или свойства_столбца1,
...,
 свойства_уровня_таблицы
);
```

Операторы изменения описания таблиц в ALTER TABLE

С помощью операторов изменения описания таблиц можно выполнить следующие действия над столбцами таблицы:

- **ADD** – добавление нового столбца к таблице.
- **DROP**– удаление существующего столбца таблицы.
- **RENAME** – переименование существующего столбца таблицы.
- **MODIFY**– изменение типа данных существующего столбца таблицы.

Ключевые слова и синтаксис может слегка отличаться у разных СУБД.

Не все допускают удаление столбцов.

Примеры изменения описания столбцов таблицы

Выполним изменения описания таблицы клиентов Customers, добавим поле электронного адреса, удалим столбец города:

```
ALTER TABLE Customers
ADD COLUMN Email varchar(255);

ALTER TABLE Customers
DROP COLUMN City;

ALTER TABLE Persons
MODIFY COLUMN DateOfBirth year;
```

Инструкции CREATE VIEW для создания представления

Инструкции CREATE VIEW служит для создания представлений – виртуальных таблиц, основанных на результатах выполнения запросов SELECT:

```
CREATE VIEW название_представления AS
Оператор SELECT
;
```

Под представление не выделяется пространство для хранения,
оно существует лишь во время сеанса приложения с СУБД.
Но после создания его можно использовать в запросах как существующие таблицы.

Пример создания и использования представления

Создадим представление [Brazil Customers] для клиентов из Бразилии, включающее их имена и контакты:

```
CREATE VIEW [Brazil Customers] AS
      SELECT CustomerName, ContactName
      FROM Customers
      WHERE Country = 'Brazil'
;
```

Теперь мы можем обратиться к представлению [Brazil Customers] как к таблице:

```
SELECT *
FROM [Brazil Customers]
;
```

Пример изменения представления

Инструкция CREATE OR REPLACE VIEW изменяет представление. Следующая инструкция SQL добавляет столбец City в представление [Brazil Customers]:

```
CREATE  OR REPLACE VIEW [Brazil Customers] AS
      SELECT CustomerName, ContactName, City
      FROM Customers
      WHERE Country = ’Brazil’
;
```

Инструкции INSERT для добавления строк в таблицу

Инструкция INSERT INTO используется для вставки новых записей в таблицу. Написать инструкцию INSERT INTO можно двумя способами.

1-й способ определяет как имена столбцов, так и значения, которые будут вставлены, при этом можно опускать имена и значения для полей, допускающих NULL:

```
INSERT INTO имя_таблицы (столбец1, столбец2, ...)
   VALUES (значение1, значение2, ...);
```

2-й способ, если вы добавляете значения для всех столбцов таблицы, вам не нужно указывать имена столбцов в SQL запросе. Однако нужно соблюдать порядок значений согласно порядку столбцов в таблице при определении:

```
INSERT INTO имя_таблицы
   VALUES (значение1, значение2, ...);
```

Пример добавления строк в таблицу

Инструкция `INSERT INTO` добавляет новую запись в таблицу Customers по 1-му способу:

```
INSERT INTO Customers
(CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway')
;
```

По 2-му способу эта инструкция выглядит так:

```
INSERT INTO Customers
VALUES
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway')
;
```

Пример добавления строк в таблицу

Инструкция CREATE OR REPLACE VIEW изменяет представление. Следующая инструкция SQL добавляет столбец City в представление [Brazil Customers]:

```
CREATE  OR REPLACE VIEW [Brazil Customers] AS
      SELECT CustomerName, ContactName, City
      FROM Customers
      WHERE Country = 'Brazil'
;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

Инструкции UPDATE для обновления данных в таблице

Инструкция UPDATE используется для изменения существующих записей в таблице. Предложение WHERE указывает, какие записи должны быть обновлены:.

```
UPDATE имя_таблицы
SET столбец1 = значение1, столб2 = value2, ...
WHERE condition;
```

Будьте осторожны при обновлении записей в таблице!
Если вы *опустите предложение WHERE*, то *все записи в таблице будут обновлены*!

Пример обновления данных в таблице

Следующая инструкция SQL обновляет первого клиента (CustomerID = 1) указанием нового контактного лица и нового города:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Выводы по использованию SQL в инженерии данных

- Наряду с широким применением инструкции SELECT важное значение имеют инструкции создания, наполнения и обновления таблиц CREATE, INSERT, UPDATE
- Представления VIEW (знакомые запросы в MS Access) существенно упрощают обработку данных, хотя снижают производительность
- Для воспроизведения всего набора операций по созданию и наполнению БД часто используют сценарий – файл с типом SQL, содержащий все необходимые инструкции.
- Не стоит злоупотреблять возможностью изменять таблицы с помощью ALTER TABLE.

Терпения и удачи всем, кто связан с базами данных

Валерий Иванович Артемьев

МГТУ имени Н.Э. Баумана, кафедра ИУ-5

Банк России
Департамент данных, проектов и процессов

Тел.: +7(495) 753-96-25
e-mail: viart@bmstu.ru

Спасибо за внимание!