DevOps Crash Course

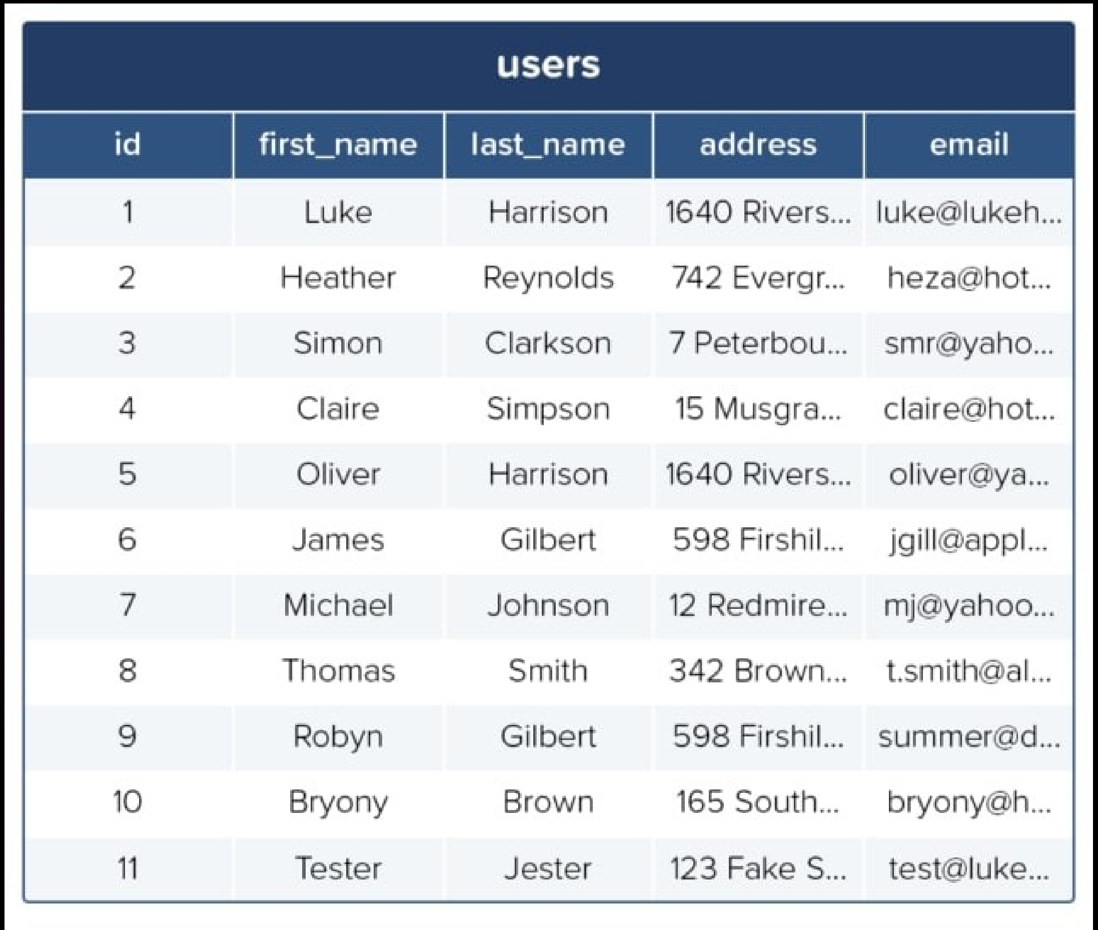
by Oleksii Yakivchik

softserve

Relational databases

A relational database is a digital database based on the relational model of data, as proposed by E. F. Codd in 1970.A software system used to maintain relational databases is a relational database management system (RDBMS). Many relational database systems have an option of using the SQL (Structured Query Language) for querying and maintaining the database.

Relational dtabases



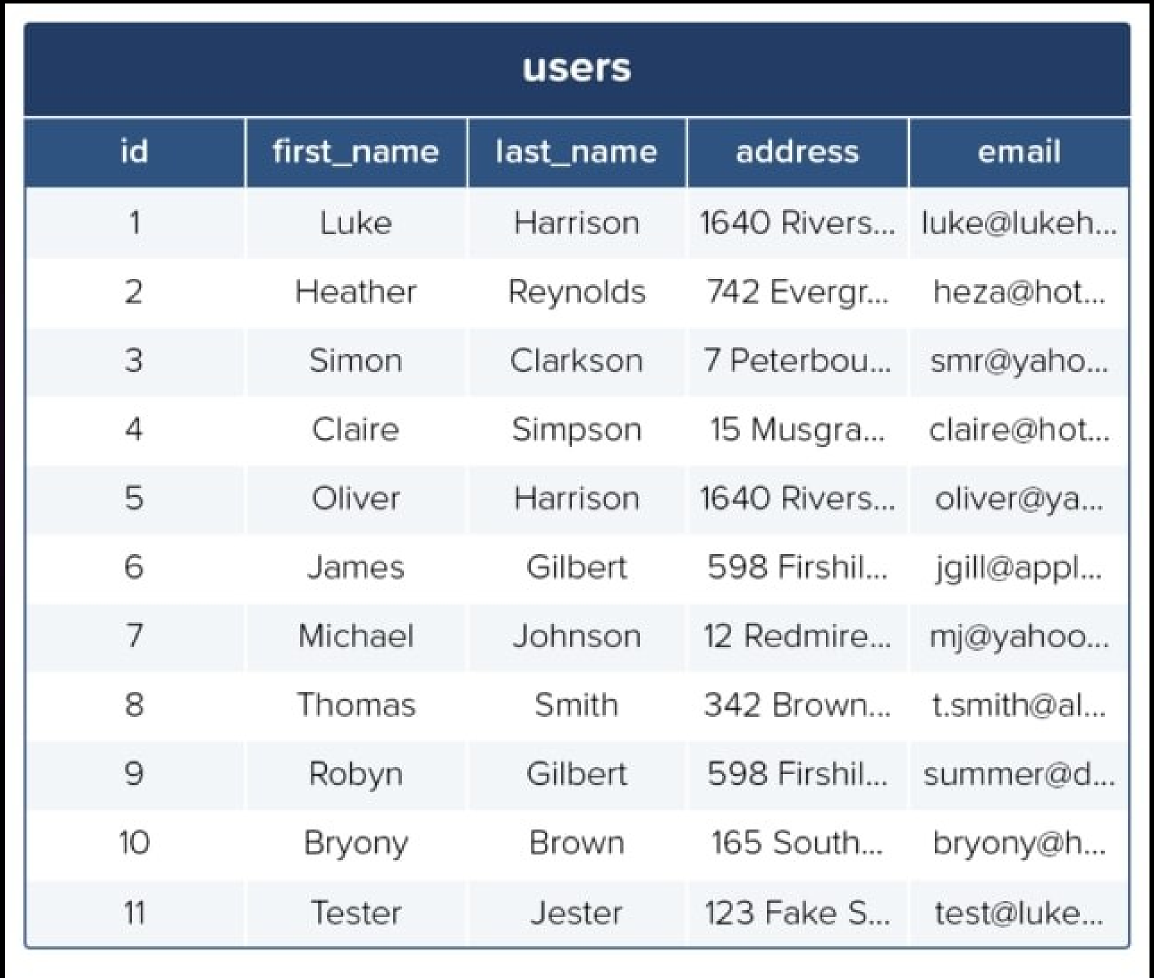
users				
id	first_name	last_name	address	email
1	Luke	Harrison	1640 Rivers...	luke@lukeh...
2	Heather	Reynolds	742 Evergr...	heza@hot...
3	Simon	Clarkson	7 Peterbou...	smr@yaho...
4	Claire	Simpson	15 Musgra...	claire@hot...
5	Oliver	Harrison	1640 Rivers...	oliver@ya...
6	James	Gilbert	598 Firshil...	jgill@appl...
7	Michael	Johnson	12 Redmire...	mj@yahoo...
8	Thomas	Smith	342 Brown...	t.smith@al...
9	Robyn	Gilbert	598 Firshil...	summer@d...
10	Bryony	Brown	165 South...	bryony@h...
11	Tester	Jester	123 Fake S...	test@luke...

Relational databases

This model organizes data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row. Rows are also called records or tuples. Columns are also called attributes. Generally, each table/relation represents one "entity type" (such as customer or product). The rows represent instances of that type of entity (such as "Lee" or "chair") and the columns representing values attributed to that instance (such as address or price).

For example, each row of a class table corresponds to a class, and a class corresponds to multiple students, so the relationship between the class table and the student table is "one to many"

Relational databases



users

id	first_name	last_name	address	email
1	Luke	Harrison	1640 Rivers...	luke@lukeh...
2	Heather	Reynolds	742 Evergr...	heza@hot...
3	Simon	Clarkson	7 Peterbou...	smr@yaho...
4	Claire	Simpson	15 Musgra...	claire@hot...
5	Oliver	Harrison	1640 Rivers...	oliver@ya...
6	James	Gilbert	598 Firshil...	jgill@appl...
7	Michael	Johnson	12 Redmire...	mj@yahoo...
8	Thomas	Smith	342 Brown...	t.smith@al...
9	Robyn	Gilbert	598 Firshil...	summer@d...
10	Bryony	Brown	165 South...	bryony@h...
11	Tester	Jester	123 Fake S...	test@luke...

Relational databases

Each row in a table has its own unique key. Rows in a table can be linked to rows in other tables by adding a column for the unique key of the linked row (such columns are known as foreign keys). Codd showed that data relationships of arbitrary complexity can be represented by a simple set of concepts.

Relational databases

Part of this processing involves consistently being able to select or modify one and only one row in a table. Therefore, most physical implementations have a unique primary key (PK) for each row in a table. When a new row is written to the table, a new unique value for the primary key is generated; this is the key that the system uses primarily for accessing the table.

Relational databases

System performance is optimized for PKs. Other, more natural keys may also be identified and defined as alternate keys (AK). Often several columns are needed to form an AK (this is one reason why a single integer column is usually made the PK). Both PKs and AKs have the ability to uniquely identify a row within a table.

Relational databases

The primary keys within a database are used to define the relationships among the tables. When a PK migrates to another table, it becomes a foreign key in the other table. When each cell can contain only one value and the PK migrates into a regular entity table, this design pattern can represent either a one-to-one or one-to-many relationship.

Relational databases

Most relational database designs resolve many-to-many relationships by creating an additional table that contains the PKs from both of the other entity tables – the relationship becomes an entity; the resolution table is then named appropriately and the two FKs are combined to form a PK.

Relational databases

The migration of PKs to other tables is the second major reason why system-assigned integers are used normally as PKs; there is usually neither efficiency nor clarity in migrating a bunch of other types of columns.

SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

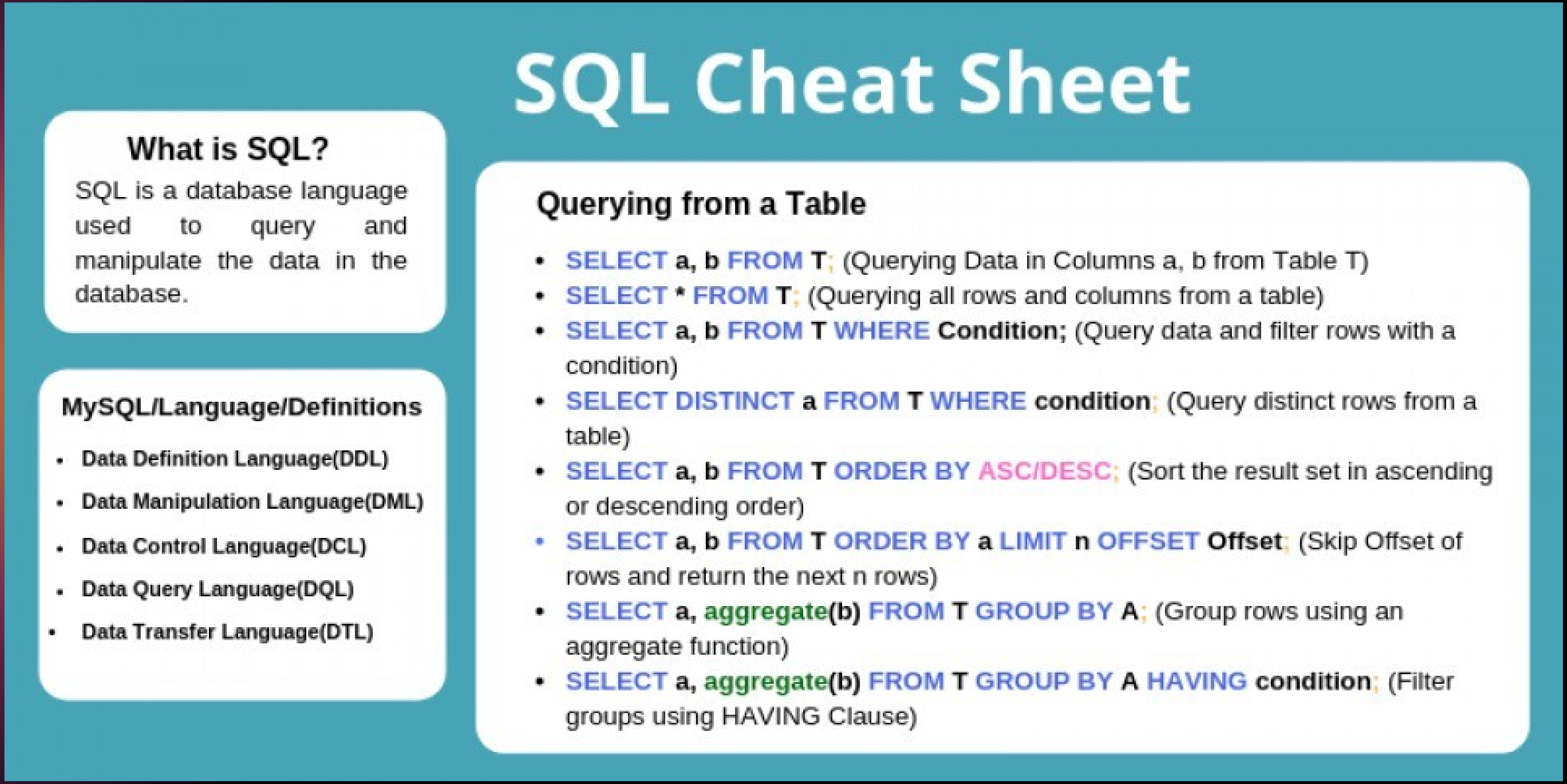
SQL

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database

SQL

- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL



SQL Cheat Sheet

What is SQL?

SQL is a database language used to query and manipulate the data in the database.

MySQL/Language/Definitions

- Data Definition Language(DDL)
- Data Manipulation Language(DML)
- Data Control Language(DCL)
- Data Query Language(DQL)
- Data Transfer Language(DTL)

Querying from a Table

- SELECT a, b FROM T; (Querying Data in Columns a, b from Table T)
- SELECT * FROM T; (Querying all rows and columns from a table)
- SELECT a, b FROM T WHERE Condition; (Query data and filter rows with a condition)
- SELECT DISTINCT a FROM T WHERE condition; (Query distinct rows from a table)
- SELECT a, b FROM T ORDER BY ASC/DESC; (Sort the result set in ascending or descending order)
- SELECT a, b FROM T ORDER BY a LIMIT n OFFSET Offset; (Skip Offset of rows and return the next n rows)
- SELECT a, aggregate(b) FROM T GROUP BY A; (Group rows using an aggregate function)
- SELECT a, aggregate(b) FROM T GROUP BY A HAVING condition; (Filter groups using HAVING Clause)

SQLAlchemy

SQLAlchemy is a popular SQL toolkit and Object Relational Mapper. It is written in Python and gives full power and flexibility of SQL to an application developer. It is an open source and cross-platform software released under MIT license. SQLAlchemy is famous for its object-relational mapper (ORM), using which classes can be mapped to the database, thereby allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

SQLAlchemy

Installation:

```
pip install sqlalchemy
```

SQLAlchemy is designed to operate with a DBAPI implementation built for a particular database. It uses dialect system to communicate with various types of DBAPI implementations and databases. All dialects require that an appropriate DBAPI driver is installed.

The following are the dialects included: MySQL, PostgreSQL, SQLite, etc

SQLAlchemy Core

Importing and connection to database:

```
from sqlalchemy import create_engine, MetaData, Table,
Column, Integer, String, DateTime

engine = create_engine('sqlite:///access.db', echo = True)
```

SQLAlchemy Core

Creating table:

```
meta = MetaData()

access_logs = Table(
   'access_logs', meta,
   Column('id', Integer, primary_key = True),
   Column('hostname', String),
   Column('ip_address', String),
   Column('date_time', DateTime),
   Column('message', String),
)
meta.create_all(engine)
```

SQLAlchemy Core

- Connect to engine

```
conn = engine.connect()
```

- Insert some value

```
ins = access_logs.insert().values(hostname = line.group(3),
ip_address = line.group(6), date_time = datetime_obj, message =
line.group(5))
result = conn.execute(ins)
```

SQLAlchemy Core

- Bulk insert:

```
logs_entries = [] #list with dicts
result = conn.execute(access_logs.insert(None),
logs_entries)
```

- Connection close

```
conn.close()
```

SQLAlchemy ORM

- Declare mapping

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base

engine = create_engine('sqlite:///access_logs.db', echo = True)
```

SQLAlchemy ORM

```
Base = declarative_base()

class LogEntry(Base):
   __tablename__ = 'access_logs'
   id = Column(Integer, primary_key=True)
   hostname = Column(String)
   ip_address = Column(String)
   date_time = Column(DateTime)
   message = Column(String)

Base.metadata.create_all(engine)
```

SQLAlchemy ORM

- Create session:

```
from sqlalchemy.orm import sessionmaker
Session = sessionmaker(bind = engine)
session = Session()
```

- Add objects:

```
log = LogEntry(hostname=line.group(3), ip_address=line.group(6),
date_time=datetime_obj, message = line.group(5))
session.add(log)
session.commit()
```

SQLAlchemy ORM

- Bulk save

```
logs_entries = [] #List of objects
session.bulk_save_objects(logs_entries)
session.commit()
```

- Close session:

```
session.close()
```