Database systems

Wojciech Barczynski

Whoami

Wojciech Barczyński:

- System Engineer
- Lead @ Codility

Quick check

Do you have experience with:

- Programing languages?
- Databases?
- NoSQL?
- Did you ever use SQL?

Aim of the course

1. Relation data model
2. Learn how to query dataases with SQL
3. Design good database
4. Different types of databases

Intuition

- What is a database?
- Let's imagine we need to build one...

Around us

- Youtube
- Self-driving cars

ATM

Learn How to Make ATM Deposits Into Your Bank Account
1 Choose the on-screen option for deposits.
2 Choose the account you want to deposit to.
3 Enter the amount of your deposit if necessary.
$500
4 Insert the envelope, checks, or cash into the ATM.
5 Get a receipt, preferably with images of each check you deposited on the receipt.
the balance

Operations

- Read Balance
- Update Balance
- Give Money

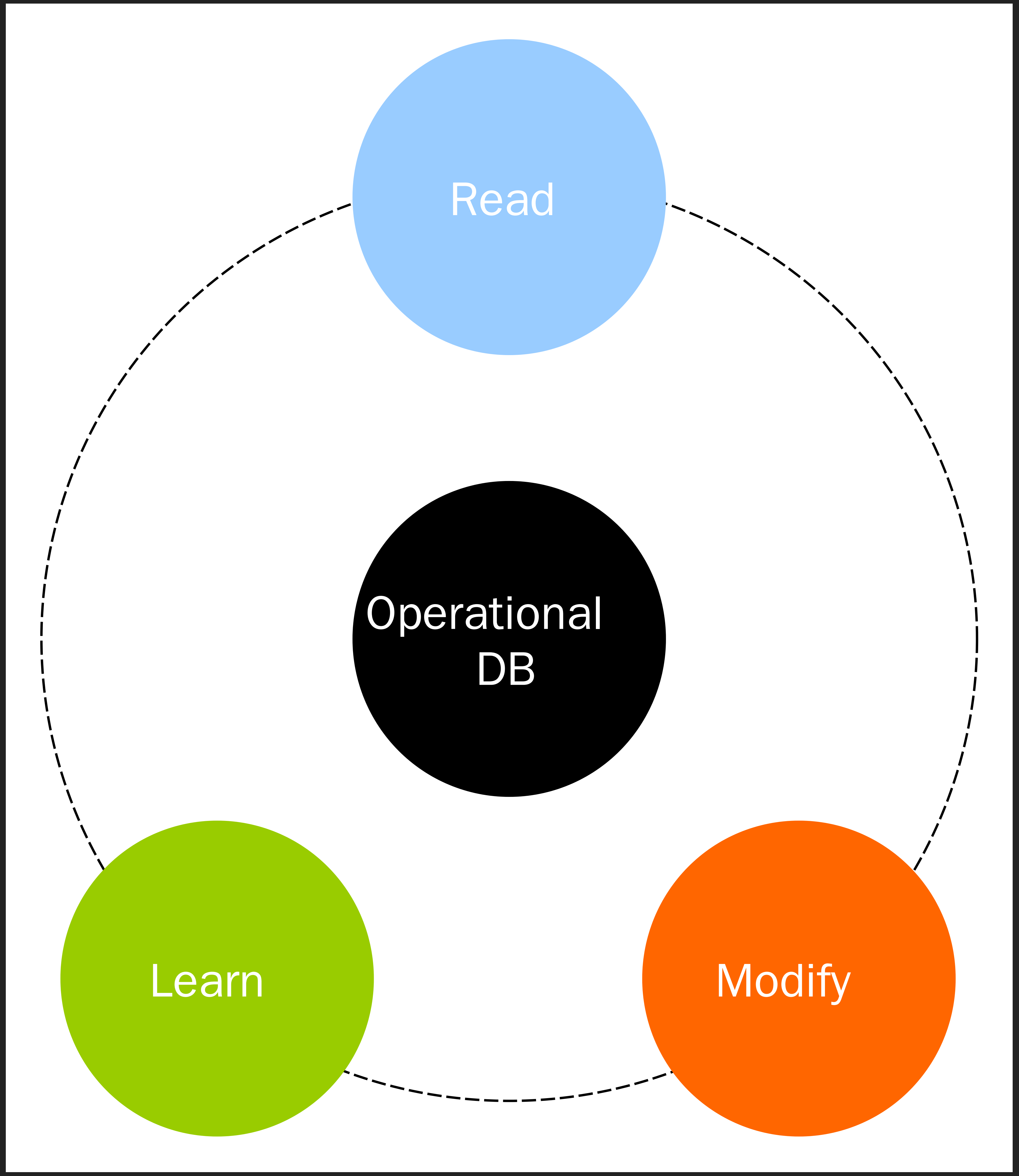
ATM

Learn How to Make ATM Deposits Into Your Bank Account
1 Choose the on-screen option for deposits.
2 Choose the account you want to deposit to.
3 Enter the amount of your deposit if necessary.
$500
4 Insert the envelope, checks, or cash into the ATM.
5 Get a receipt, preferably with images of each check you deposited on the receipt.
the balance

Operations

- Read Balance
- Give Money
- Update Balance

Standard DB



Supporting

- Scale
- Speed
- Stability
- Reliality

Specialisation

- store data (lots of reads)
- optimize historical data (e.g., logs)
- run batch workflows (e.g., training)

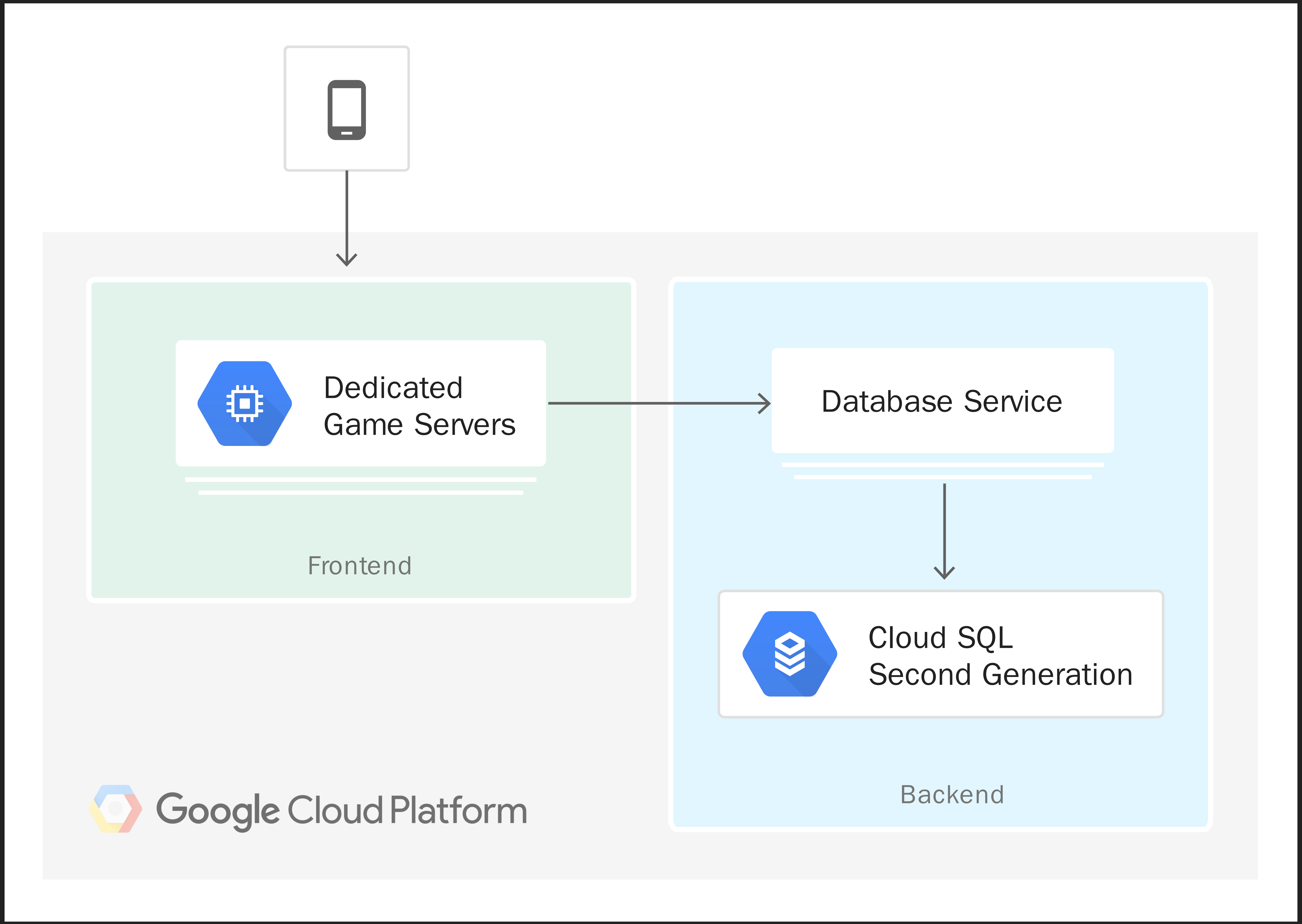
Specialisation

We have over 100+ databases on the market:

- MySQL, Postgres, SAP, Oracle, Sqlite, Mongo,...
- Cloud: GCP, AWS, Bigquery, Coackroach,

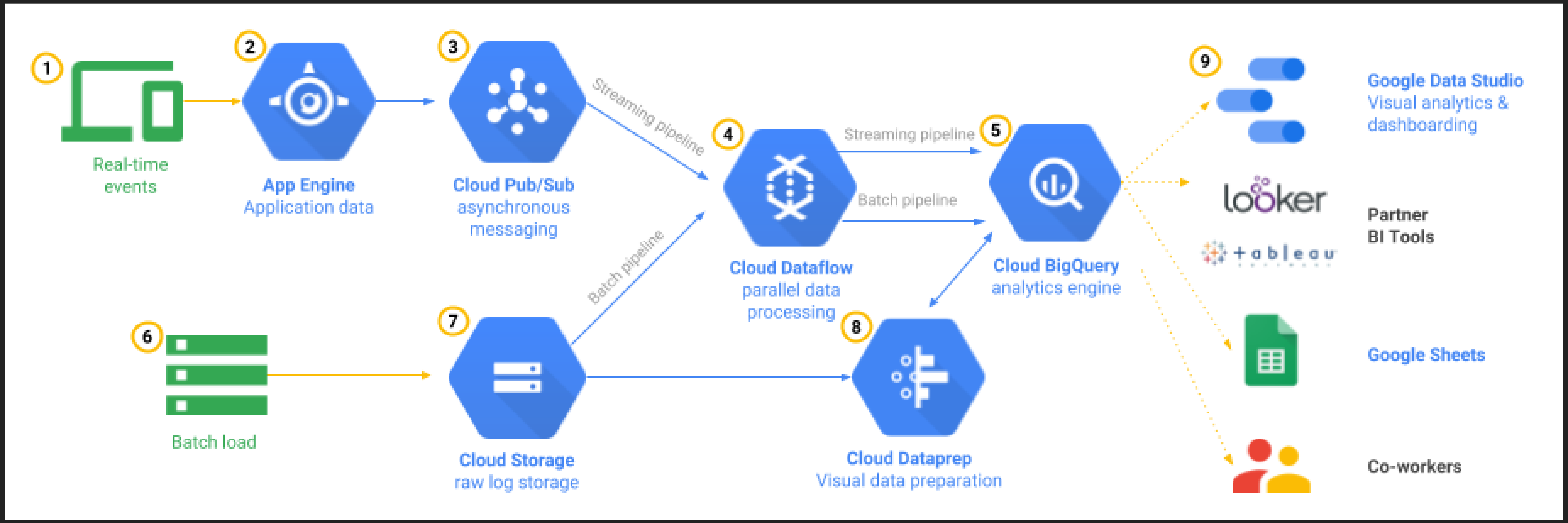
App example

Game app


Dedicated
Game Servers
Database Service
Frontend
Cloud SQL
Second Generation
Google Cloud Platform
Backend

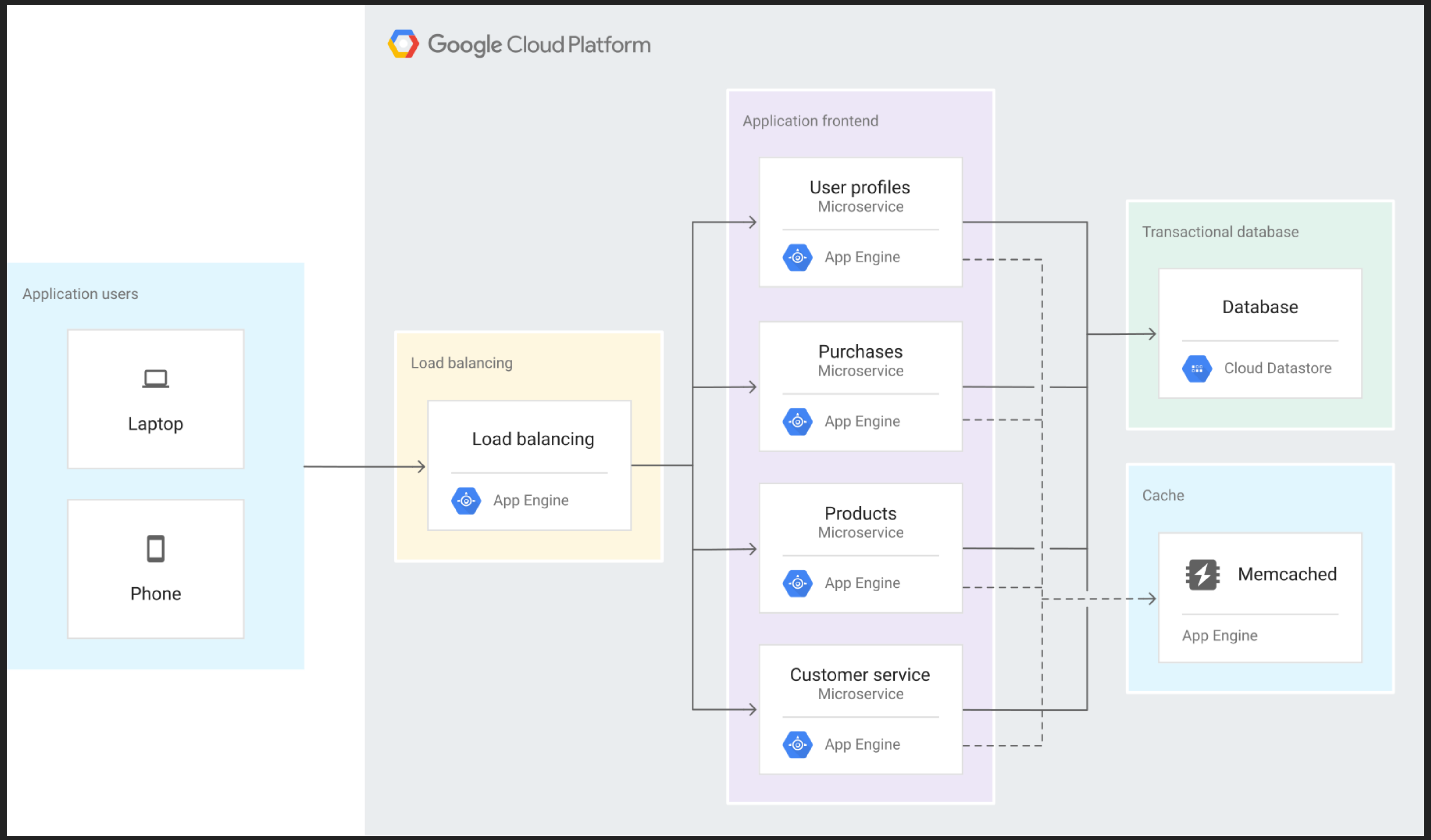
src

Game app


1
Real-time events
2
App Engine
Application data
3
Cloud Pub/Sub
asynchronous messaging
Streaming pipeline
Batch pipeline
4
Cloud Dataflow
parallel data processing
Streaming pipeline
Batch pipeline
5
Cloud BigQuery
analytics engine
6
Batch load
7
Cloud Storage
raw log storage
8
Cloud Dataprep
Visual data preparation
9
Google Data Studio
Visual analytics & dashboarding
looker
tableau
Partner BI Tools
Google Sheets
Co-workers

src

Game app


Google Cloud Platform
Application users
Laptop
Phone
Load balancing
Load balancing
App Engine
Application frontend
User profiles
Microservice
App Engine
Purchases
Microservice
App Engine
Products
Microservice
App Engine
Customer service
Microservice
App Engine
Transactional database
Database
Cloud Datastore
Cache
Memcached
App Engine

src

Databases

Two Big ideas

- Declarative interfaces
- Transactions

Declarative interfaces

- Apps specify what they want, not how to do it
- Example: “store a table with 2 integer columns”, but not how to encode it on disk
- Example: “count records where column1 = 5”

Transactions

- Encapsulate multiple app actions into one atomic request (fails or succeeds as a whole)
- Concurrency models for multiple users
- Clear interactions with failure recovery

Declarative interfaces

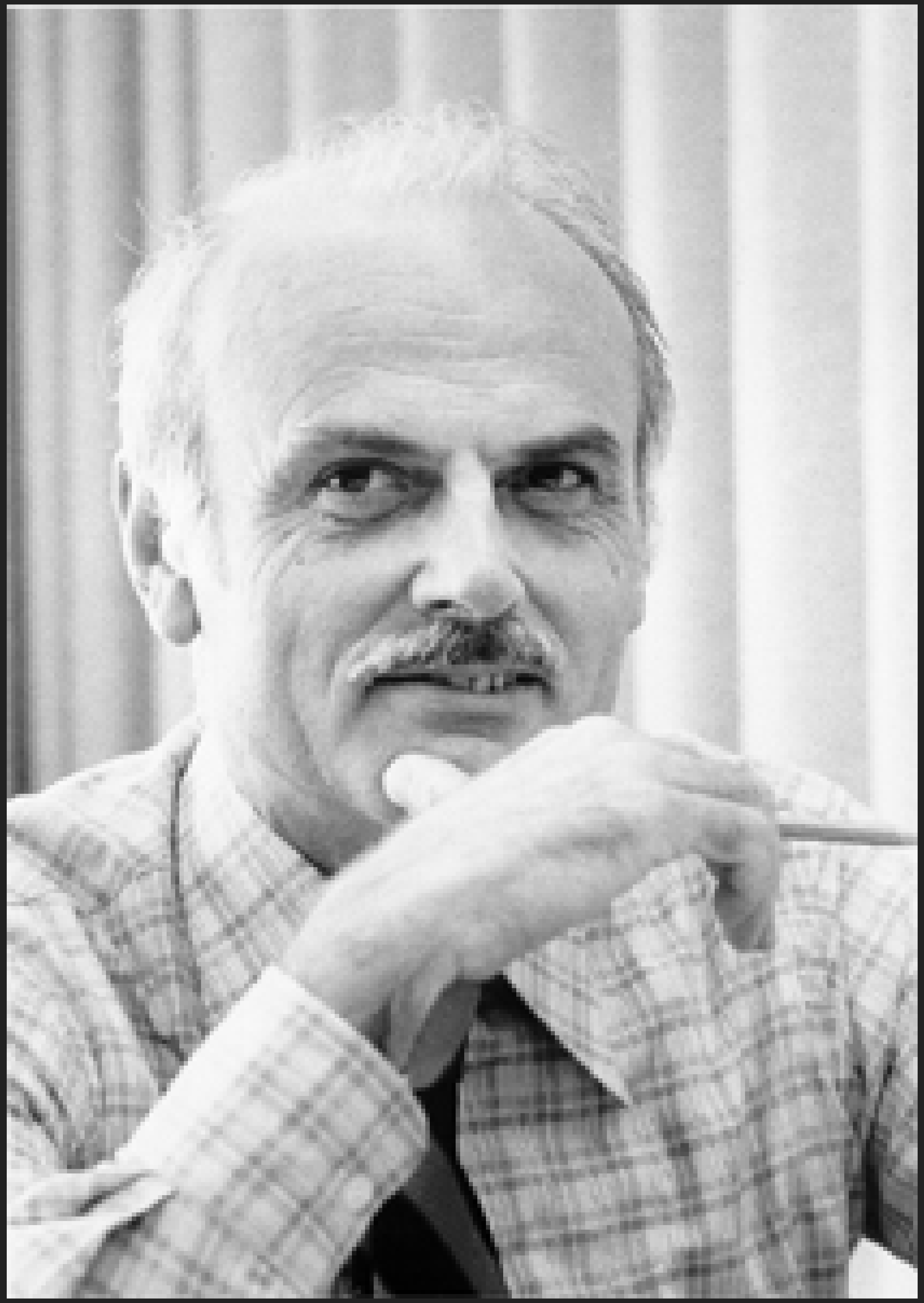
SQL:

- Abstract “table” data model, many physical implementations
- Specify queries in a restricted language that the database can optimize

Trasaction

- SQL:
  - Commands to start, abort or end transactions
  - based on multiple SQL statements
- Apache Spark: multi-part output of a job appear atomically when all particitns are done

Edgar F Codd



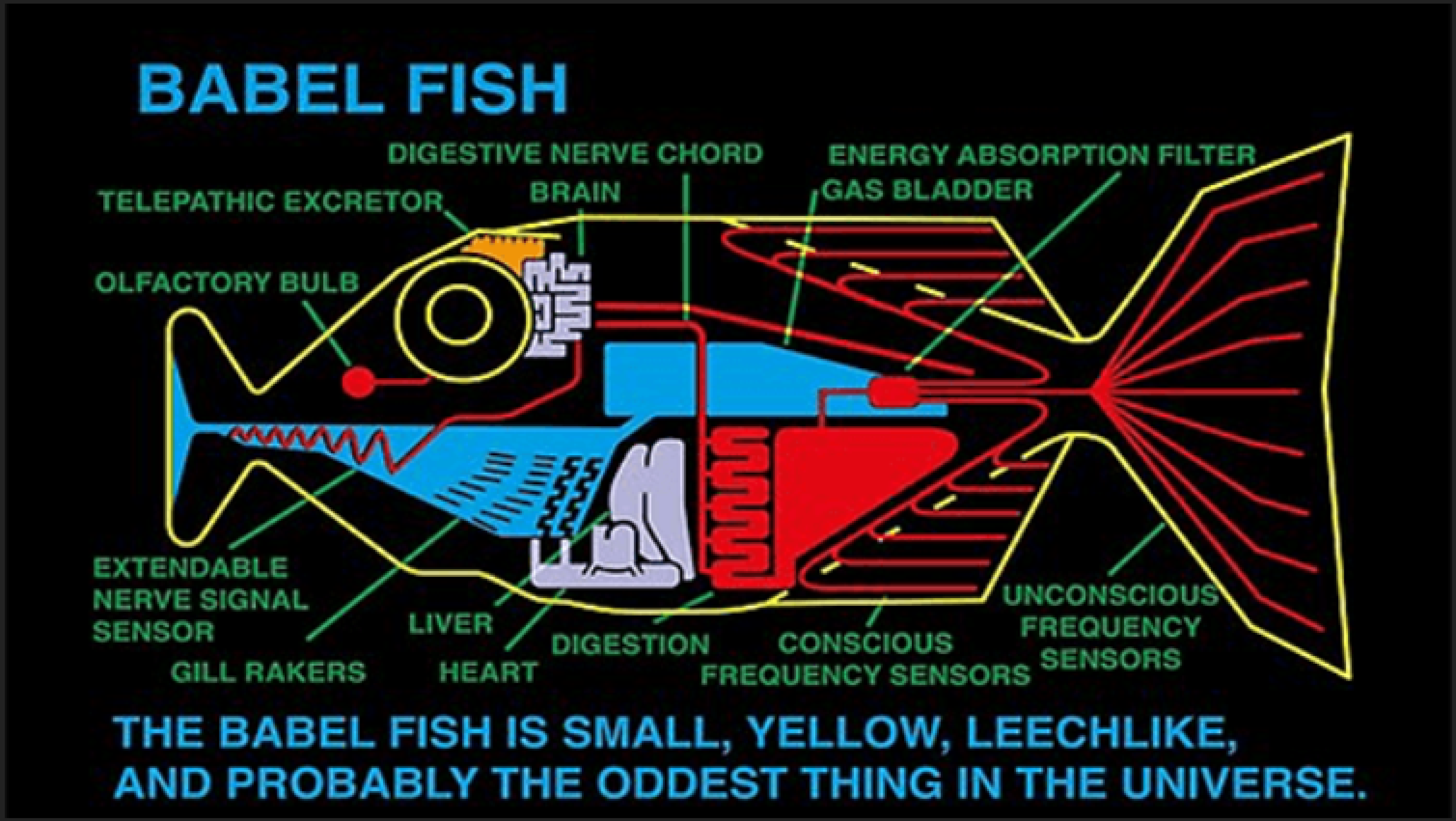
- Proposed the relational DB model,
  with declarative queries & storage (1970)
- Relation = table with unique key identifying each row

History

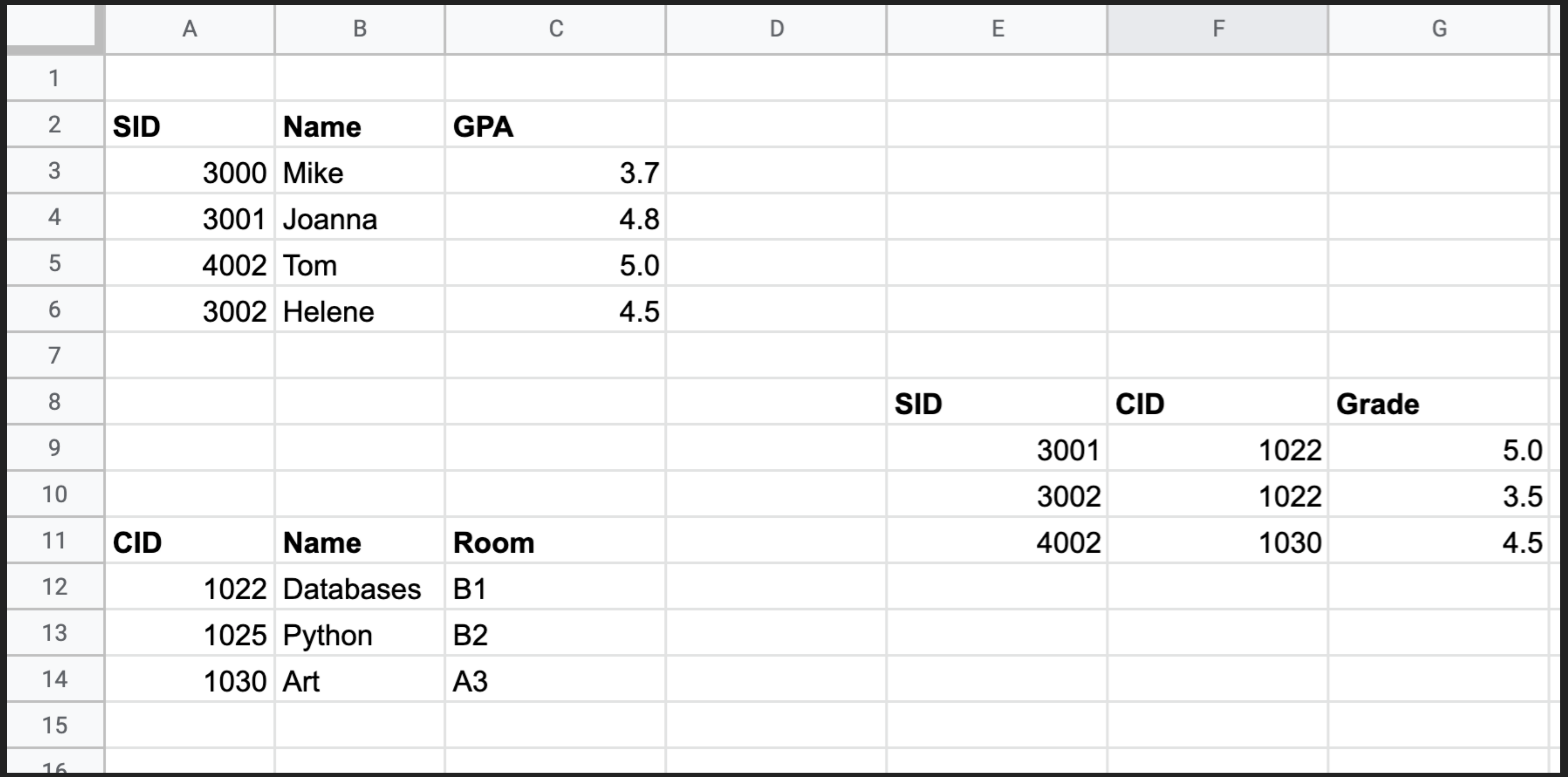
Timeline:

- data storage
- navigational databases (1964)
- IBM System R (1974)
- Ingress (1974) - led to PostgreSQL
- Oracle database (1979)

Intro SQL


BABEL FISH
DIGESTIVE NERVE CHORD
ENERGY ABSORPTION FILTER
TELEPATHIC EXCRETOR
BRAIN
GAS BLADDER
OLFACTORY BULB
EXTENDABLE NERVE SIGNAL SENSOR
GILL RAKERS
LIVER
HEART
DIGESTION
CONSCIOUS FREQUENCY SENSORS
UNCONSCIOUS FREQUENCY SENSORS
THE BABEL FISH IS SMALL, YELLOW, LEECHLIKE, AND PROBABLY THE ODDEST THING IN THE UNIVERSE.

Intro SQL



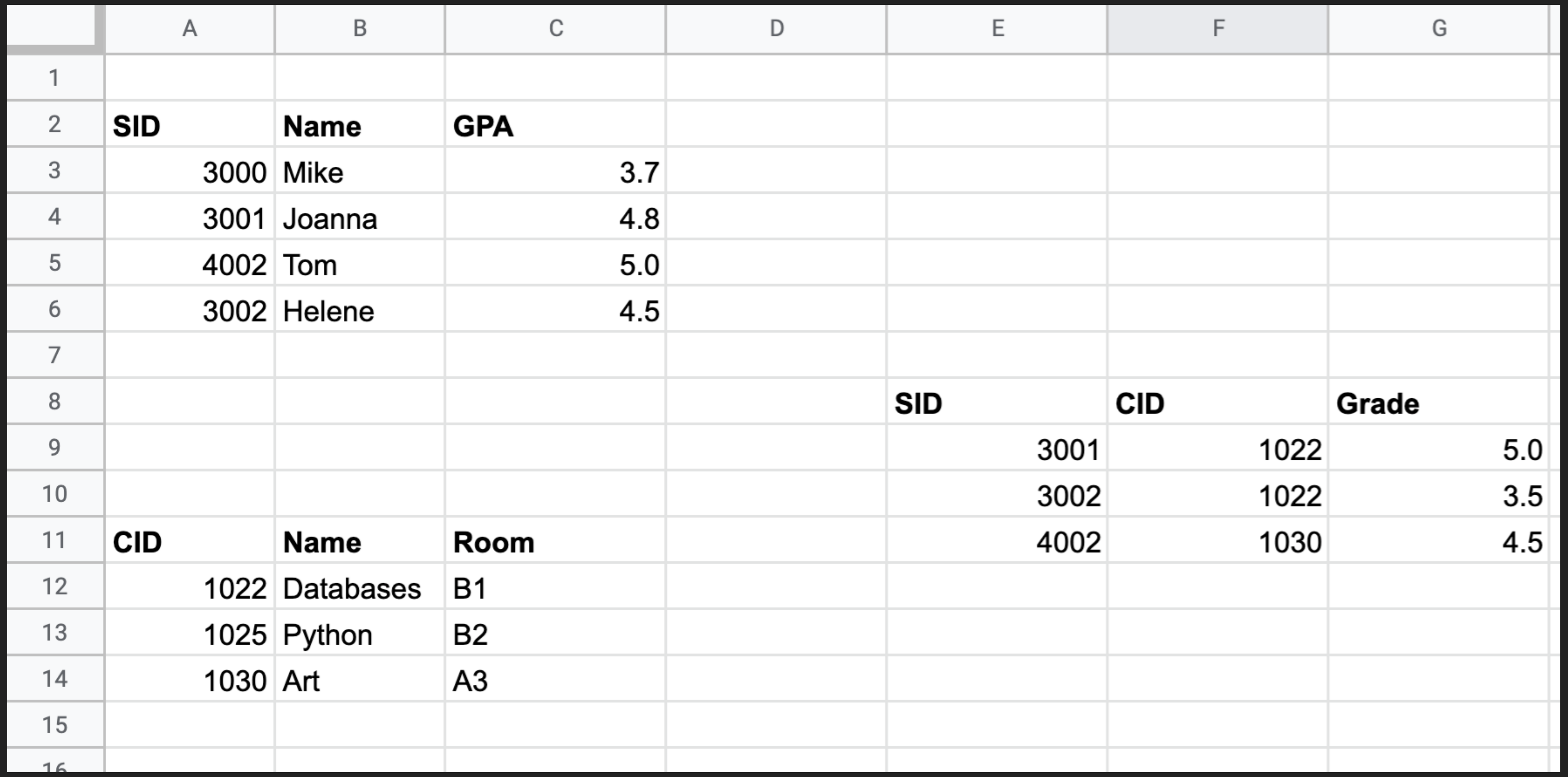
	A	B	C	D	E	F	G
1							
2	**SID**	**Name**	**GPA**				
3	3000	Mike	3.7				
4	3001	Joanna	4.8				
5	4002	Tom	5.0				
6	3002	Helene	4.5				
7							
8					**SID**	**CID**	**Grade**
9					3001	1022	5.0
10					3002	1022	3.5
11	**CID**	**Name**	**Room**		4002	1030	4.5
12	1022	Databases	B1				
13	1025	Python	B2				
14	1030	Art	A3				
15							

Tables

- Student(sid:str, name:str, gpa:float)
- Course(cid:str, name:str, room:str)
- Enrolled(sid:str, cid:str, grade:float)
  sid → Student
  cid → Course

spreadsheet

Intro SQL



	A	B	C	D	E	F	G
1							
2	**SID**	**Name**	**GPA**				
3	3000	Mike	3.7				
4	3001	Joanna	4.8				
5	4002	Tom	5.0				
6	3002	Helene	4.5				
7							
8					**SID**	**CID**	**Grade**
9					3001	1022	5.0
10					3002	1022	3.5
11	**CID**	**Name**	**Room**		4002	1030	4.5
12	1022	Databases	B1				
13	1025	Python	B2				
14	1030	Art	A3				
15							

Queries

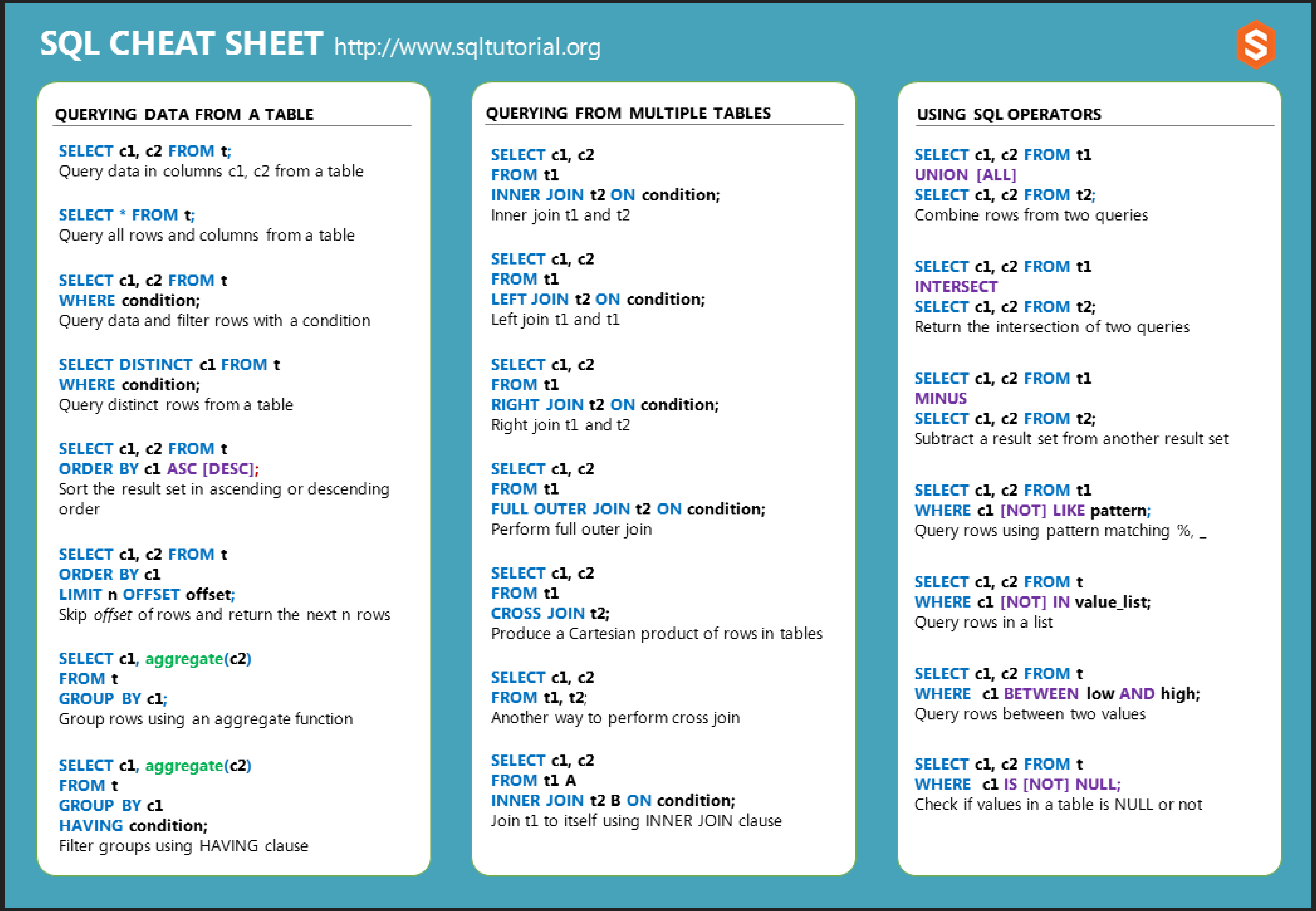
- GPA of Student Tom?
- AVG student GPA?
- Helene's classes?

spreadsheet

Example

- w3schools.com/sql
- trysql

SQL



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

http://www.sqltutorial.org/sql-cheat-sheet/

Intro SQL

- slides

Questions?

Wojciech Barczyński

References

- Data Management and Data Systems
- http://db-class.org/
- Principles of Data-Intensive Systems

Backup slides

Prepare your env

- Open `Terminal`

```
sudo apt-get install python3-venv python3-pip -y
sudo snap install atom --classic

mkdir -p workspace
cd workspace

# zauważ kropka
atom .
```

Prepare your env - Atom

Install the following Atom plugins (Edit->Preferences):

- `platformio-ide-terminal`

Terminal in Atom

- Packages -> platformio-ide-terminal-> Toggle

```
python3 -m venv venv
source venv/bin/activate
```