Relational Databases

Terminology used in describing databases

ACID

- Atomicity (a database **transaction either succeeds or fails**)
- Consistency (use of rules and constraints so **state is always valid)**
- Isolation (**transactions happen in isolation**)
- Duratibility (**committed transactions are permanent**/stored in non-volatile storage like hard disk)

BASE

- Basically Available (generally available for queries; **no isolation/waiting for other transactions**)
- Soft State (because consistency is *eventual*, **state may show some lag**)
- Eventually Consistent (as system's **state is gradually replicated across all nodes**, it eventually becomes consistent.)

CAP Theorem

(architectural goal is all three, distributed networks reality - choose two)

- Consistency – no updates to any node if nodes can't communicate (nodes stay in sync)
- Availability – system always responds
- Partition Tolerance – replication across nodes

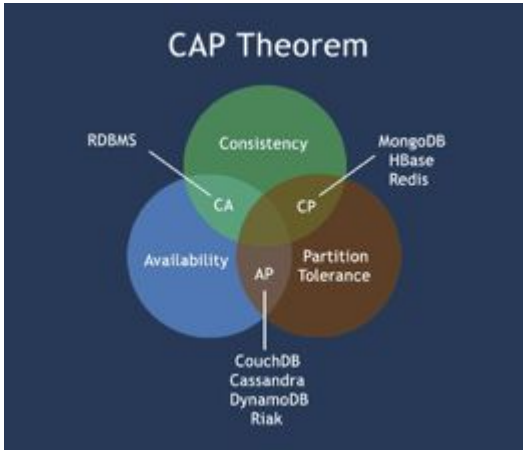

Image: https://practice.geeksforgeeks.org/problems/cap-theorem

Relational Database Management System (RDBMS) / SQL Database / Transactional Database

A relational database contains multiple tables, each of which are **related** to one or multiple other tables in the database.

- Reduce redundancy
- Most commonly used structure in enterprise scenarios
- Build-in data integrity
- Support ACID compliance to guarantee data validity
- Difficult to scale

Rules and Constraints

Rules are imposed on tables to ensure data structure and integrity within and between tables and while performing queries.

- Data type of a column
- Whether or not NULL values are allowed
- Whether or not each value has to be unique

Not NULL

Unique

Varchar Float Int Int Float

State	Murder	Assult	Urban_pop	Rape
Alabama	13.2	236	58	21.0
Alaska	10.0	263	48	44.5
Arizona	8.2	294	80	31.0

Keys

Keys uniquely identify each row. Keys can be a single column or a combination of multiple.

- Should not change over time (time-invariant)
- Follows unique and non-null rules
- Can be multiple possible keys but only one **Primary Key** per table
- Information about keys stored in database metadata
- A **Foreign Key** in a table is a **Primary Key** in another table, used to join data (not necessarily a key in the given table)

Primary Key

State	Murder	Assault	Urban_pop	Rape
Alabama	13.2	236	58	21.0
Alaska	10.0	263	48	44.5
Arizona	8.2	294	80	31.0

Data Normalization

Refers is a way to reduce redundancy. Common practice is to use **Third Normal Form(3NF).** The State_Crimes table on the left below can be normalized by creating a state table and a crime type table. Tables like State and Crime are sometimes called lookup tables.

State_Crimes

State	Crime_type	Value	Urban_pop
Alabama	Murder	13.2	58
Alabama	Assault	236	58
Alabama	Rape	21	58
Alaska	Murder	10	48
Alaska	Assault	263	48
Alaska	Rape	44.5	48
Arizona	Murder	8.2	80
Arizona	Assault	294	80

State_Crimes

State_id	Crime_id	Value
1	1	13.2
1	2	236
1	3	21
2	1	10
2	2	263
2	3	44.5
3	1	8.2
3	2	294

State

id	Name	Urban_pop
1	Alabama	58
2	Alaska	48
3	Arizona	80

Crime

id	Name
1	Murder
2	Assault
3	Rape

https://en.wikipedia.org/wiki/Third_normal_form

Entity Relationship Diagram

A visualization describing the relationships between tables in a database. Maps relationships between Primary Keys and Foreign Keys between tables

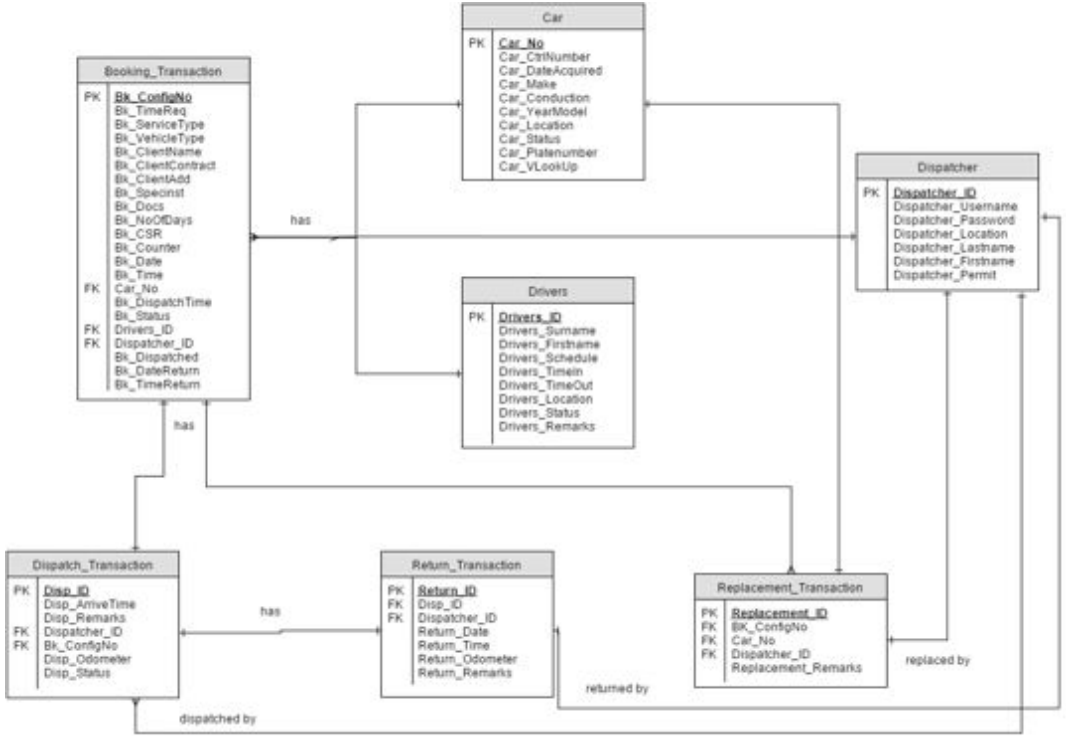

Image: https://sites.google.com/site/dracvbms/entity-relationship-diagram

Referential Integrity

When a **Foreign Key** references a **Primary Key** in another table, the referenced record must exist.

- The **Primary Key** should always exist for a given **Foreign Key**
- The data type of the **Primary Key** and **Foreign Key** must match
- **Foreign Keys** enforce **Referential Integrity**

Referential Integrity violations:

Remove **Primary Key** that is referenced in another table

Table A		
PK	col_1	col_2
1	a	b
2	a	g
3	a	h

Table B		
PK	FK	col_1
A	1	f
B	1	g
C	2	d

Add **Foreign Key** without corresponding **Primary Key**

Table A		
PK	col_1	col_2
1	a	b
2	a	g
3	a	h

Table B		
PK	FK	col_1
A	1	f
B	1	g
C	2	d
D	4	s