Introduction to

Databases with SQL

Scaling

Scalability

Ability to increase or decrease capacity to meet demand

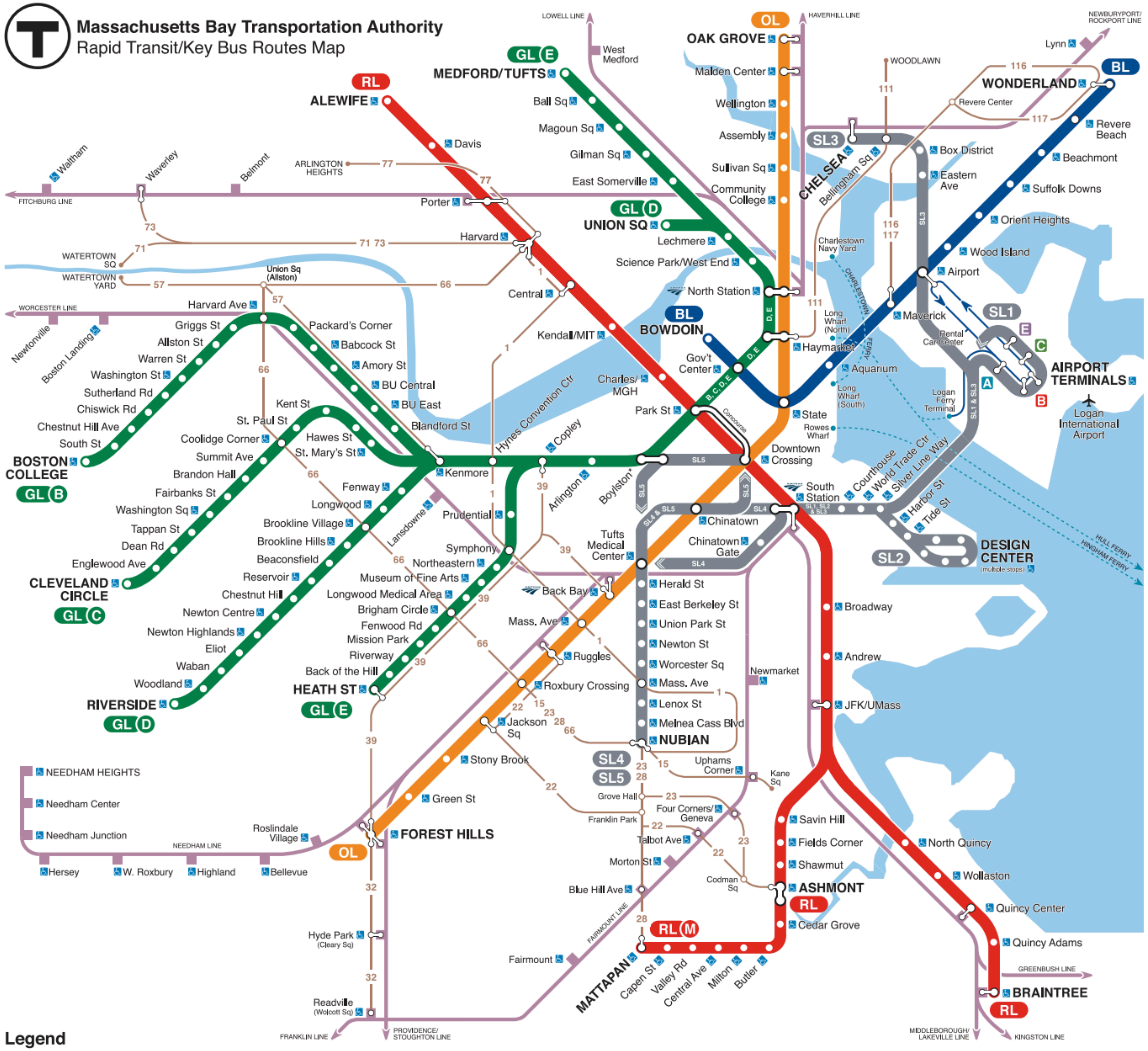
MySQL

PostgreSQL

...

Massachusetts Bay Transportation Authority

Rapid Transit/Key Bus Routes Map



Legend

RL RED LINE
M MATTAPAN LINE
OL ORANGE LINE
BL BLUE LINE

SL SILVER LINE and branches SL1 SL3 SL2 SL4 SL5

GL GREEN LINE and branches
B Terminates at Gov't Ctr D Terminates at Union Sq
C Terminates at Gov't Ctr E Terminates at Medford/Tufts

COMMUTER RAIL

– 000 – KEY BUS ROUTE
Frequent service

FERRY

*Boylston: Accessible for Silver Line only

Accessible station
All MBTA and Massport bus and ferry services are accessible

Rapid Transit transfer station

Commuter Rail transfer station

Free Logan Airport shuttle bus

Amtrak service
Back Bay, North & South Stations

@MBTA
/TheMBTA
@TheMBTA
/MBTAGM
@TheMBTA

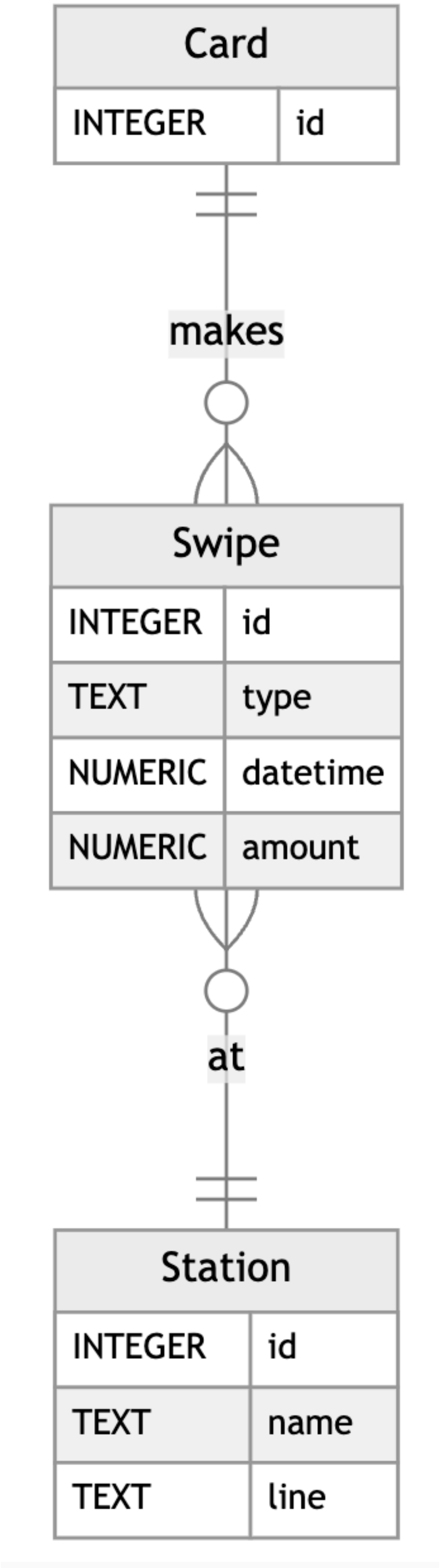
Customer Communications & Travel Info
617-222-3200, 800-392-6100,
TTY 617-222-5146, www.mbta.com

MBTA Transit Police: 911
TTY 617-222-1200

Elevator/Escalator availability: 617-222-2828

Not to scale

CharlieCard
U
T
Massachusetts Bay Transportation Authority
Exp: 02/62



Card	
INTEGER	id

Swipe	
INTEGER	id
TEXT	type
NUMERIC	datetime
NUMERIC	amount

Station	
INTEGER	id
TEXT	name
TEXT	line

Types

```
CREATE TABLE "cards" (
    "id" INTEGER,
    PRIMARY KEY("id")
);
```

Integers

Data Type
TINYINT
SMALLINT
MEDIUMINT
INT
BIGINT

Data Type	Size (in Bytes)
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT	4
BIGINT	8

Data Type	Size (in Bytes)	Minimum Value (Signed)	Maximum Value (Signed)
TINYINT	1	-128	127
SMALLINT	2	-32,768	32,767
MEDIUMINT	3	-8,388,608	8,388,607
INT	4	-2,147,483,648	2,147,483,647
BIGINT	8	-2^{63}	$2^{63} - 1$

Data Type	Size (in Bytes)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
TINYINT	1	0	255
SMALLINT	2	0	65,535
MEDIUMINT	3	0	16,777,215
INT	4	0	4,294,967,295
BIGINT	8	0	$2^{64} - 1$

```
CREATE TABLE "cards" (
    "id" INTEGER,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE `cards` (
    `id` INTEGER,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE `cards` (
    `id` INT,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE `cards` (
    `id` INT AUTO_INCREMENT,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE "stations" (
    "id" INTEGER,
    "name" TEXT NOT NULL UNIQUE,
    "line" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

Strings

CHAR

VARCHAR

CHAR(M)

VARCHAR(M)

TEXT

TINYTEXT

TEXT

MEDIUMTEXT

LONGTEXT

BLOB

ENUM

SET

```
CREATE TABLE "stations" (
    "id" INTEGER,
    "name" TEXT NOT NULL UNIQUE,
    "line" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE `stations` (
    `id` INTEGER,
    `name` TEXT NOT NULL UNIQUE,
    `line` TEXT NOT NULL,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE `stations` (
    `id` INT AUTO_INCREMENT,
    `name` TEXT NOT NULL UNIQUE,
    `line` TEXT NOT NULL,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE `stations` (
    `id` INT AUTO_INCREMENT,
    `name` VARCHAR(32) NOT NULL …,
    `line` TEXT NOT NULL,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE `stations` (
    `id` INT AUTO_INCREMENT,
    `name` VARCHAR(32) NOT NULL …,
    `line` ENUM('blue', …) NOT NULL,
    PRIMARY KEY(`id`)
);
```

```
CREATE TABLE "swipes" (
    …,
    "type" TEXT NOT NULL CHECK("type" IN ('enter', …)),
    "datetime" NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

Dates

Times

DATE

TIME

DATETIME

TIMESTAMP

YEAR

```
DATE

TIME(fsp)

DATETIME(fsp)

TIMESTAMP(fsp)

YEAR
```

Real Numbers

Data Type	Size (in Bytes)
FLOAT	4
DOUBLE PRECISION	8

Floating-Point Imprecision

Fixed Precision

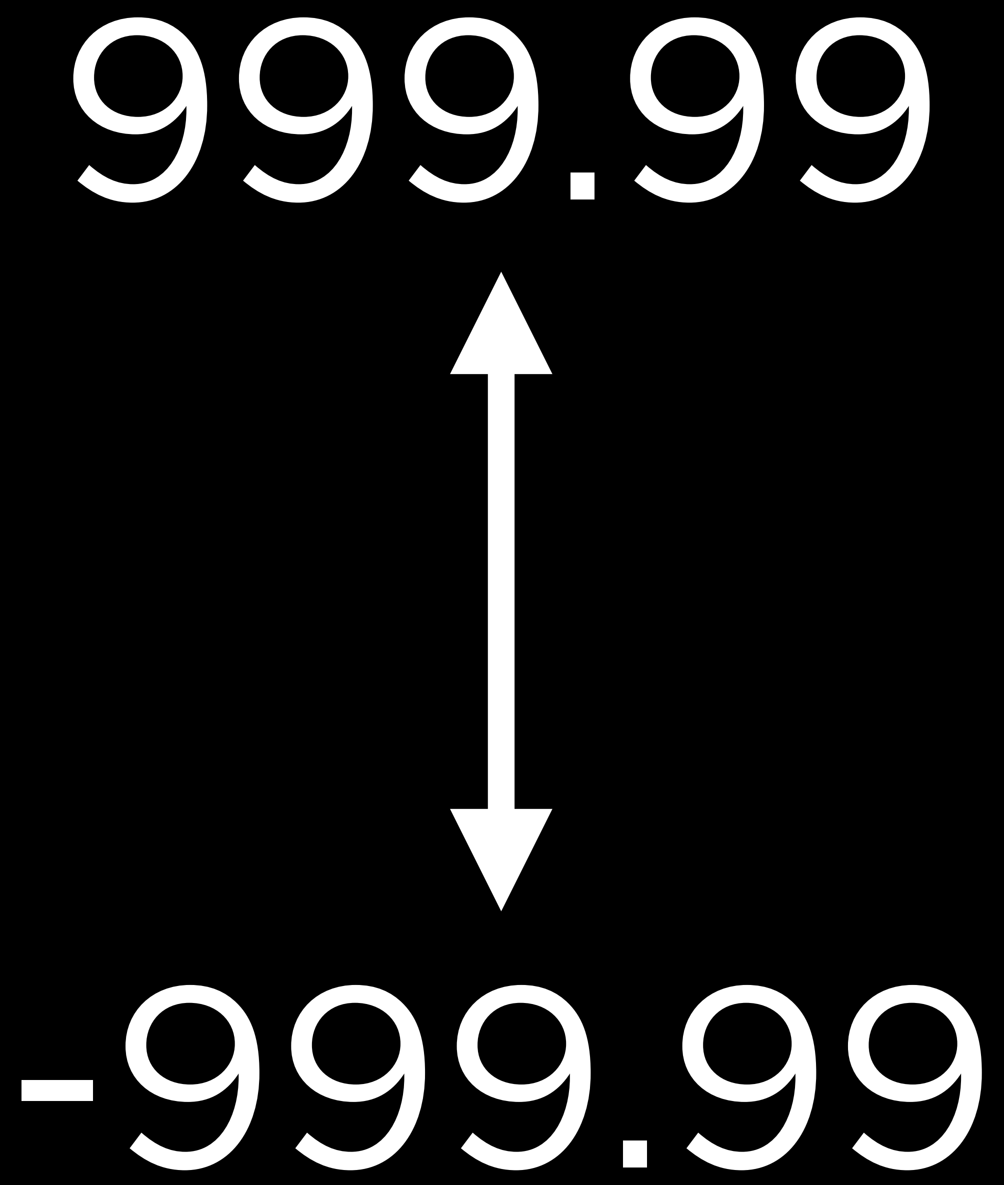
DECIMAL

```
DECIMAL(M,D)
```

DECIMAL(5,2)

999.99
↕
-999.99

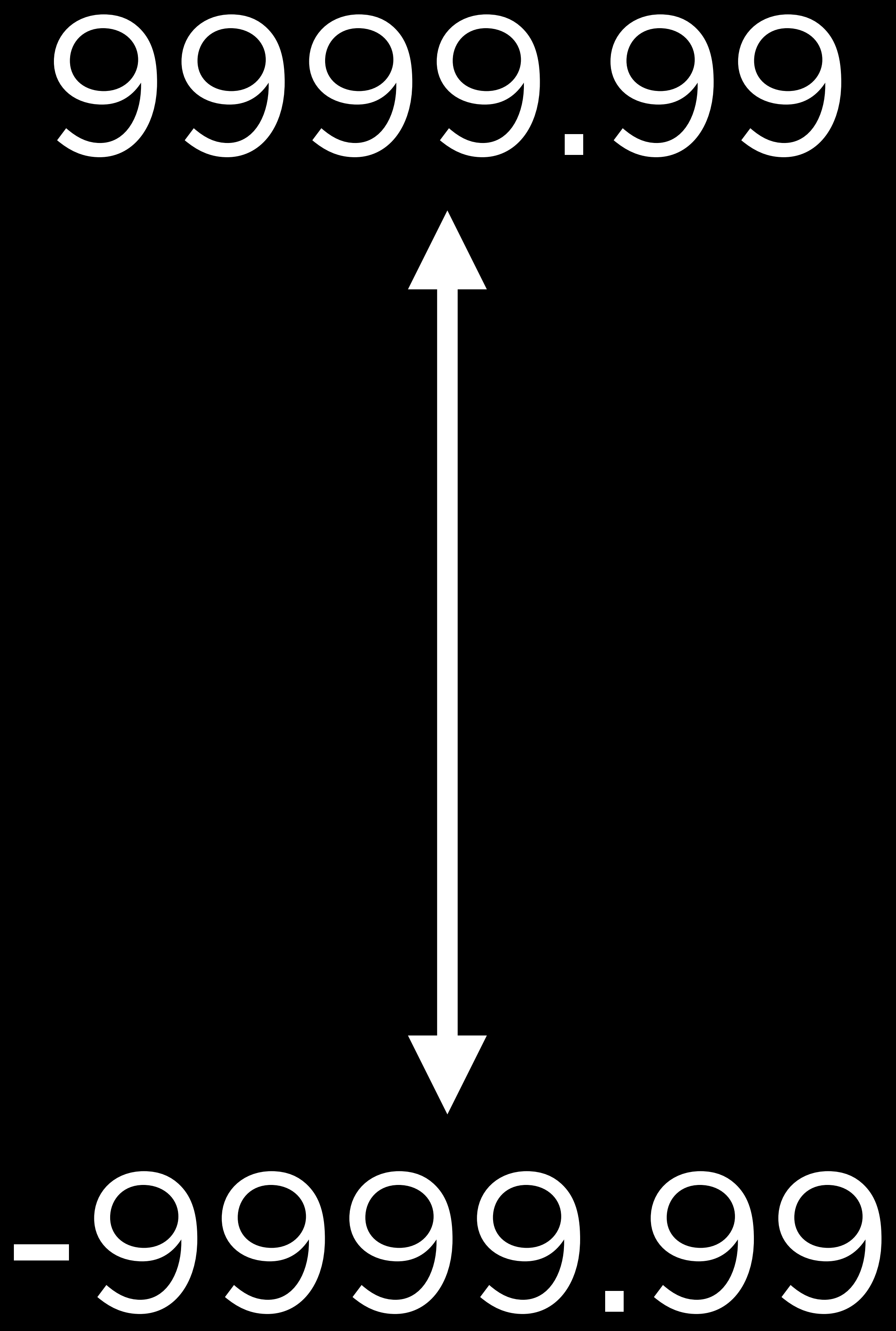
DECIMAL(5,2)



999.99

-999.99

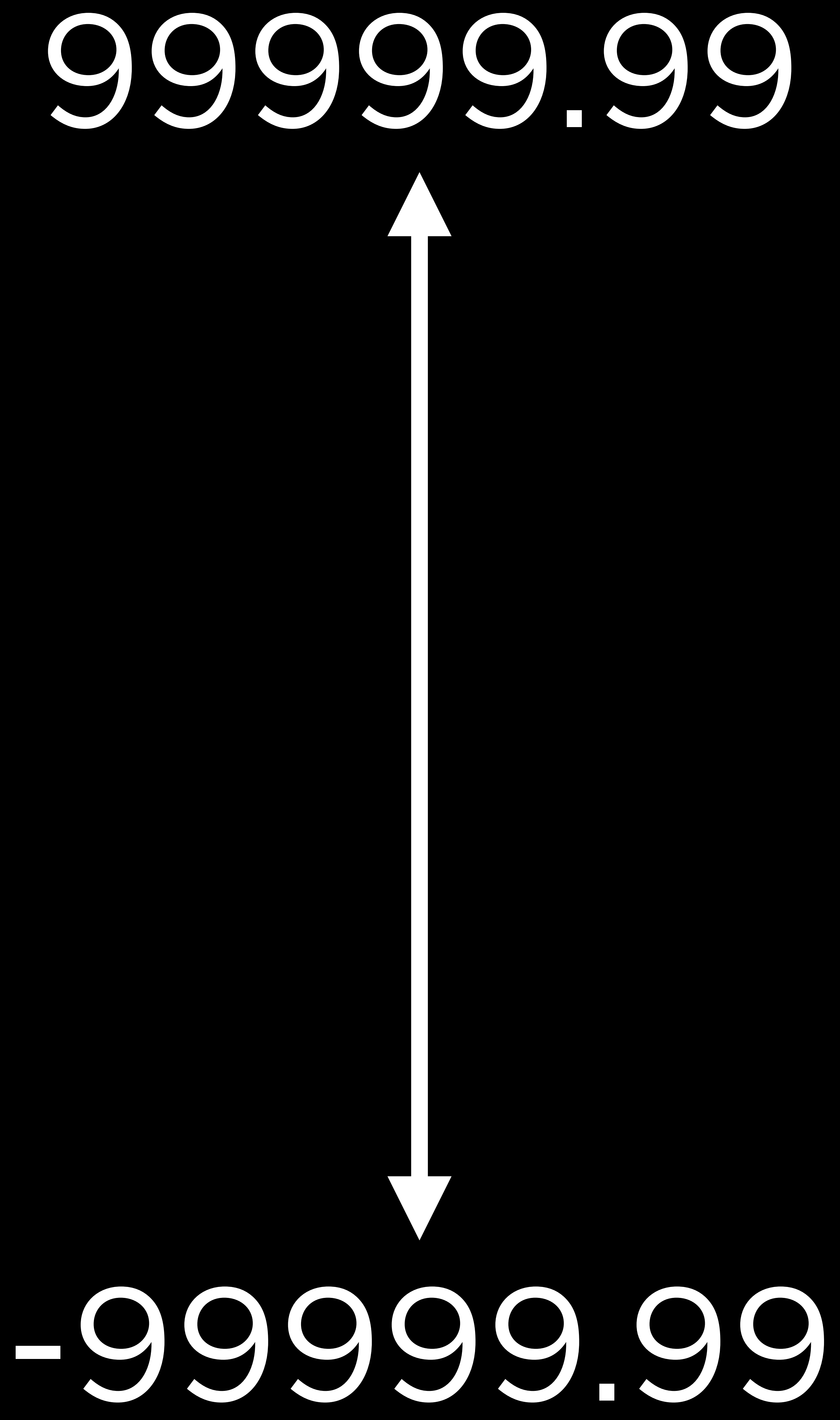
DECIMAL(6,2)



9999.99

-9999.99

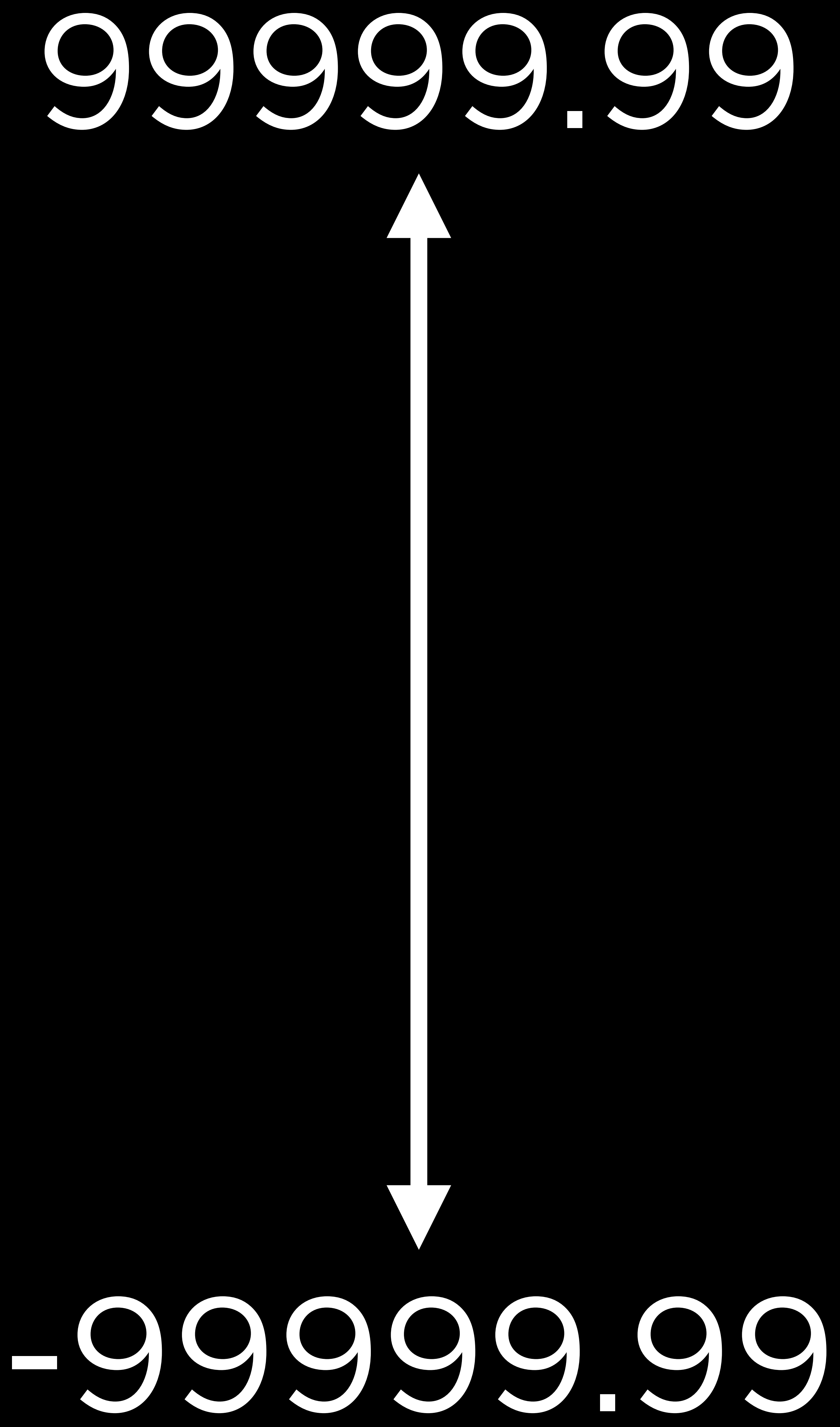
DECIMAL(7,2)



99999.99

-99999.99

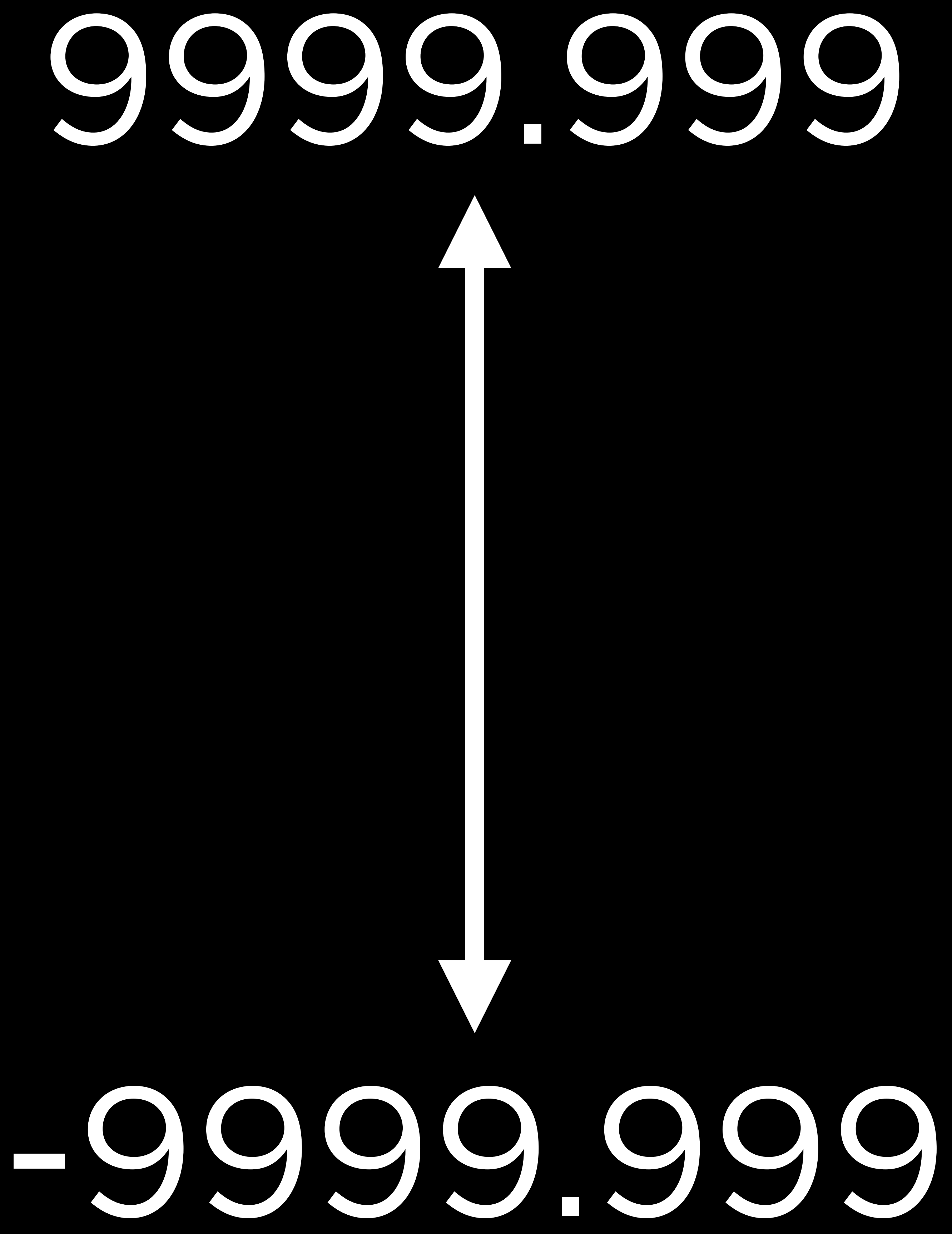
DECIMAL(7,2)



99999.99

-99999.99

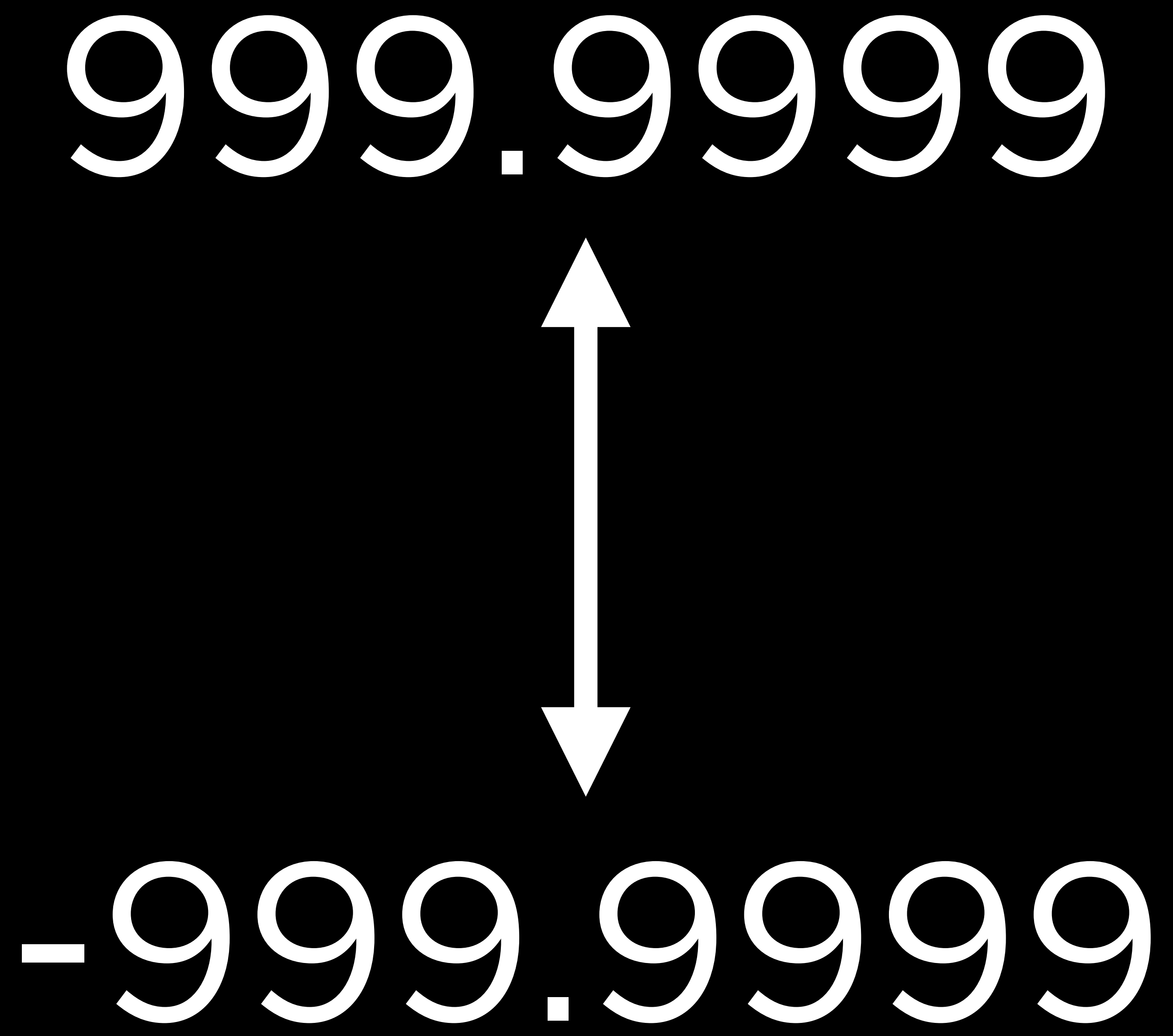
$\text{DECIMAL}(7,\underline{3})$



9999.999

-9999.999

DECIMAL(7,4)



999.9999

-999.9999

```
CREATE TABLE "swipes" (
    …,
    "type" TEXT NOT NULL CHECK("type" IN ('enter', …)),
    "datetime" NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

```
CREATE TABLE `swipes` (
    …,
    `type` TEXT NOT NULL CHECK(`type` IN ('enter', …)),
    `datetime` NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `amount` NUMERIC NOT NULL CHECK(`amount` != 0),
    …
);
```

```
CREATE TABLE `swipes` (
    …,
    `type` ENUM('enter', …) NOT NULL,
    `datetime` NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `amount` NUMERIC NOT NULL CHECK(`amount` != 0),
    …
);
```

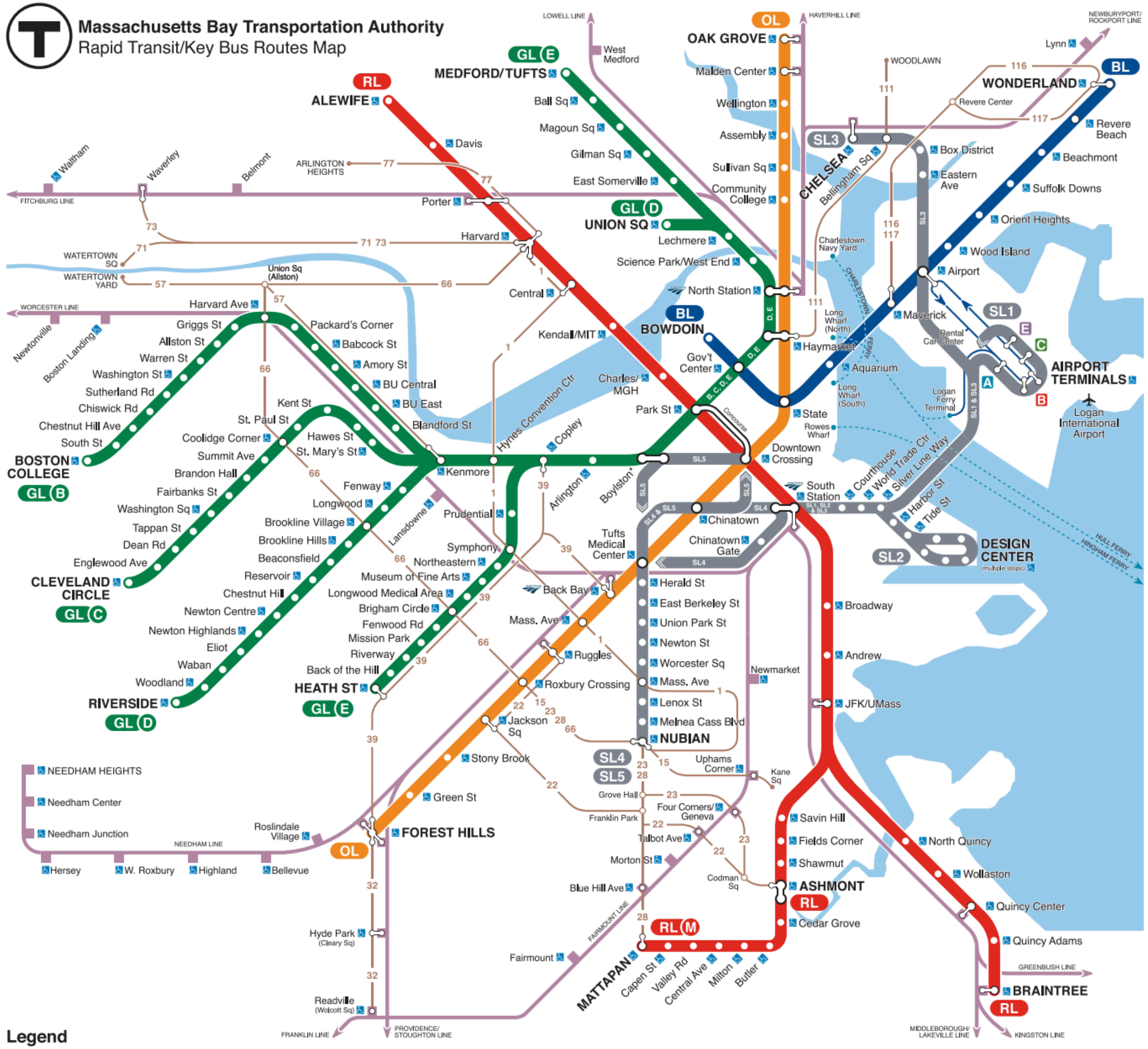
```
CREATE TABLE `swipes` (
    …,
    `type` ENUM('enter', …) NOT NULL,
    `datetime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `amount` NUMERIC NOT NULL CHECK(`amount` != 0),
    …
);
```

```
CREATE TABLE `swipes` (
    …,
    `type` ENUM('enter', …) NOT NULL,
    `datetime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `amount` DECIMAL(5,2) NOT NULL CHECK(`amount` != 0),
    …
);
```

Altering Tables

Massachusetts Bay Transportation Authority

Rapid Transit/Key Bus Routes Map



Legend

RL RED LINE

Ⓜ MATTAPAN LINE

OL ORANGE LINE

BL BLUE LINE

SL SILVER LINE and branches SL1 SL3 SL2 SL4 SL5

GL GREEN LINE and branches

Ⓑ Terminates at Gov't Ctr Ⓓ Terminates at Union Sq

Ⓒ Terminates at Gov't Ctr Ⓔ Terminates at Medford/Tufts

COMMUTER RAIL

– 000 – KEY BUS ROUTE
Frequent service

FERRY

*Boylston: Accessible for Silver Line only

Accessible station
All MBTA and Massport bus and ferry services are accessible

Rapid Transit transfer station

Commuter Rail transfer station

Free Logan Airport shuttle bus

Amtrak service
Back Bay, North & South Stations

@MBTA

/TheMBTA

@TheMBTA

/MBTAGM

@TheMBTA

Customer Communications & Travel Info
617-222-3200, 800-392-6100,
TTY 617-222-5146, www.mbta.com

MBTA Transit Police: 911
TTY 617-222-1200

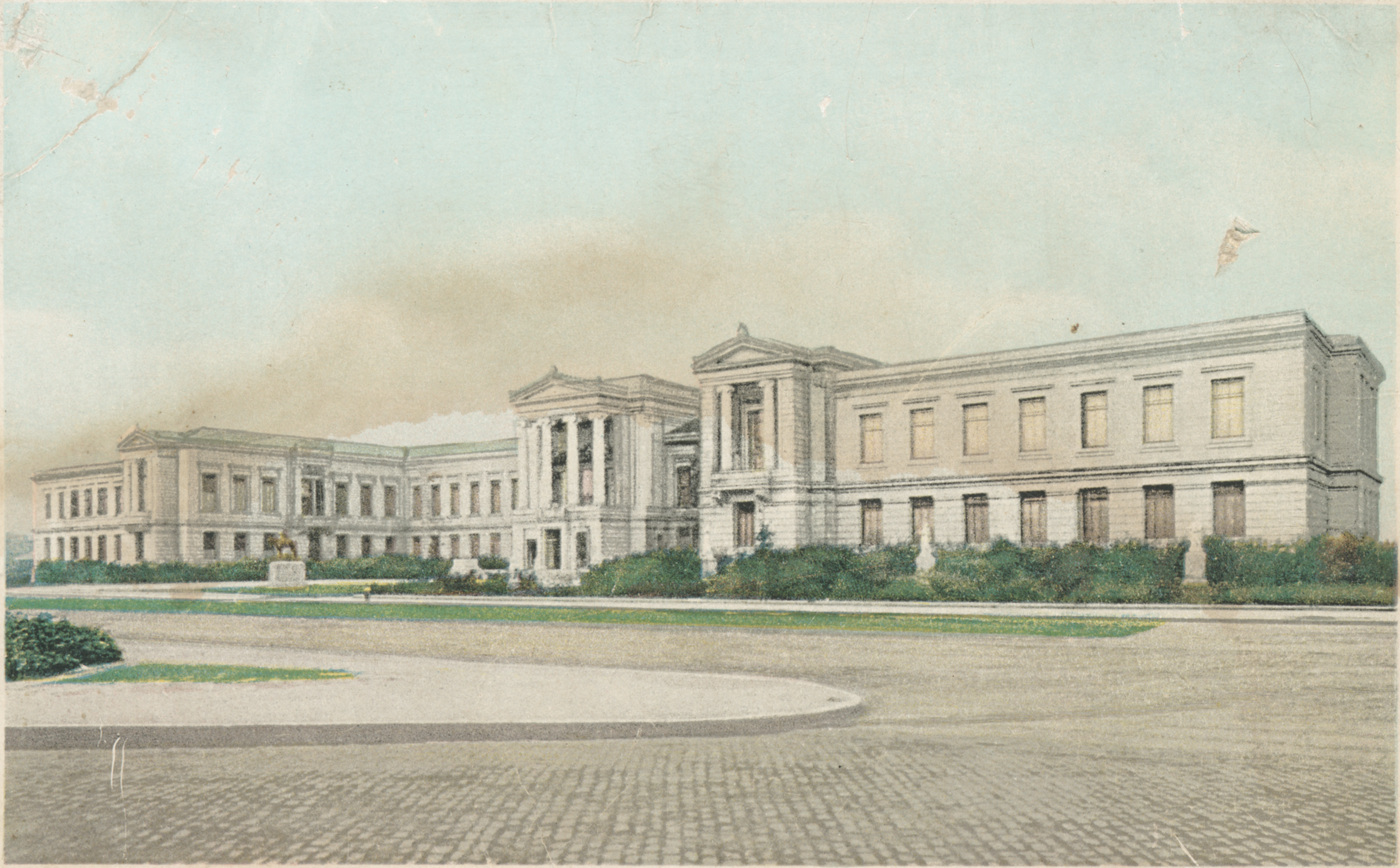
Elevator/Escalator availability: 617-222-2828

Not to scale

ALTER TABLE …

```
ALTER TABLE …
MODIFY …;
```

Stored Procedures



71541 MUSEUM OF FINE ARTS, BOSTON, MASS.

COPR. DETROIT PUBLISHING CO.

collections

id	title	deleted
1	Farmers working ...	0
2	Imaginative land...	0
3	Profusion of ...	0
4	Peonies and ...	0

collections

id	title	deleted
1	Farmers working ...	1
2	Imaginative land...	0
3	Profusion of ...	0
4	Peonies and ...	0

```
SELECT * FROM "collections"
WHERE "deleted" = 0;
```

collections

id	title	deleted
1	Farmers working ...	0
2	Imaginative land...	0
3	Profusion of ...	0
4	Peonies and ...	0

current_collections

id	title
1	Farmers working ...
2	Imaginative land...
3	Profusion of ...
4	Peonies and ...

collections

id	title	deleted
1	Farmers working ...	0
2	Imaginative land...	0
3	Profusion of ...	0
4	Peonies and ...	0

current_collections

id	title
1	Farmers working ...
2	Imaginative land...
3	Profusion of ...
4	Peonies and ...

collections

id	title	deleted
1	Farmers working …	1
2	Imaginative land…	0
3	Profusion of …	0
4	Peonies and …	0

current_collections

id	title
2	Imaginative land…
3	Profusion of …
4	Peonies and …

```
CREATE PROCEDURE name
CREATE PROCEDURE name

BEGIN
```

```
CREATE PROCEDURE name
BEGIN
…
```

```
CREATE PROCEDURE name
BEGIN
…
END;
```

```
CREATE PROCEDURE name(parameters)

BEGIN

…

END;
```

IF, ELSEIF, ELSE

LOOP

REPEAT

WHILE

…

PostgreSQL

```
CREATE TABLE "cards" (
    "id" INTEGER,
    PRIMARY KEY("id")
);
```

Integers

Data Type	Size (in Bytes)	Minimum Value (Signed)	Maximum Value (Signed)
`SMALLINT`	2	-32,768	32,767
`INT`	4	-2,147,483,648	2,147,483,647
`BIGINT`	8	-2^{63}	$2^{63} - 1$

Data Type	Size	Minimum Value	Maximum Value
	(in Bytes)	(Unsigned)	(Unsigned)
SMALLINT	2	0	65,535
INT	4	0	4,294,967,295
BIGINT	8	0	$2^{64} - 1$

SMALLSERIAL

SERIAL

BIGSERIAL

```
CREATE TABLE "cards" (
    "id" INTEGER,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "cards" (
    "id" SERIAL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "stations" (
    "id" INTEGER,
    "name" TEXT NOT NULL UNIQUE,
    "line" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "stations" (
    "id" SERIAL,
    "name" TEXT NOT NULL UNIQUE,
    "line" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "stations" (
    "id" SERIAL,
    "name" VARCHAR(32) NOT NULL …,
    "line" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "stations" (
    "id" SERIAL,
    "name" VARCHAR(32) NOT NULL …,
    "line" VARCHAR(32) NOT NULL,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "swipes" (
    …,
    "type" TEXT NOT NULL CHECK("type" IN ('enter', …)),
    "datetime" NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

ENUM

```
CREATE TYPE "swipe_type"
AS ENUM ('enter', 'exit', 'deposit');
```

Dates

Times

TIMESTAMP

DATE

TIME

INTERVAL

```
TIMESTAMP(p)

DATE

TIME(p)

INTERVAL(p)
```

Real Numbers

```
MONEY

NUMERIC(precision,scale)
```

```
CREATE TABLE "swipes" (
    …,
    "type" TEXT NOT NULL CHECK("type" IN ('enter', …)),
    "datetime" NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

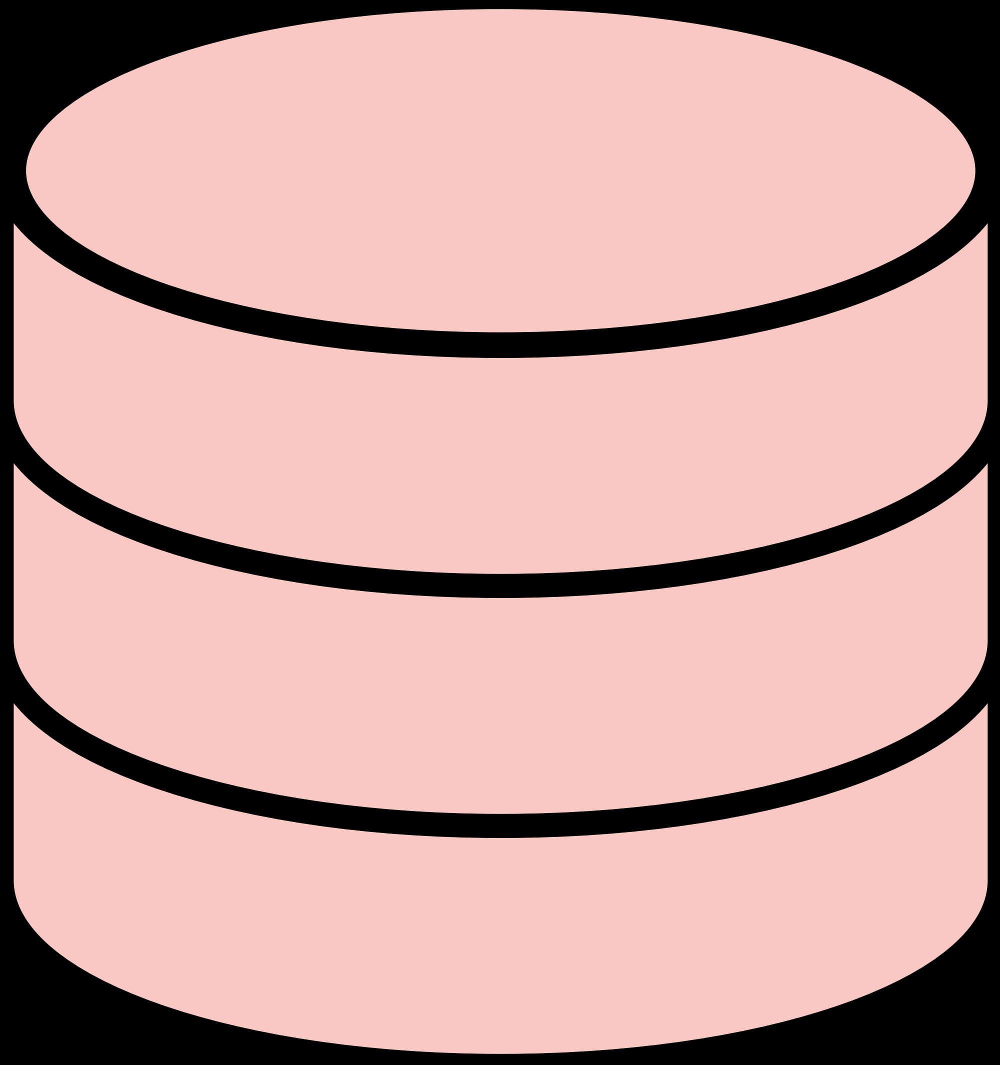
```
CREATE TYPE "swipe_type" AS ENUM('enter, …);

CREATE TABLE "swipes" (
    …,
    "type" "swipe_type" NOT NULL,
    "datetime" NUMERIC NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

```
CREATE TYPE "swipe_type" AS ENUM('enter, …);

CREATE TABLE "swipes" (
    …,
    "type" "swipe_type" NOT NULL,
    "datetime" TIMESTAMP NOT NULL DEFAULT now(),
    "amount" NUMERIC NOT NULL CHECK("amount" != 0),
    …
);
```

```
CREATE TYPE "swipe_type" AS ENUM('enter, …);

CREATE TABLE "swipes" (
    …,
    "type" "swipe_type" NOT NULL,
    "datetime" TIMESTAMP NOT NULL DEFAULT now(),
    "amount" NUMERIC(5,2) NOT NULL CHECK("amount" != 0),
    …
);
```

100 reads/minute

50 writes/minute



1,000 reads/minute

500 writes/minute

10,000 reads/minute

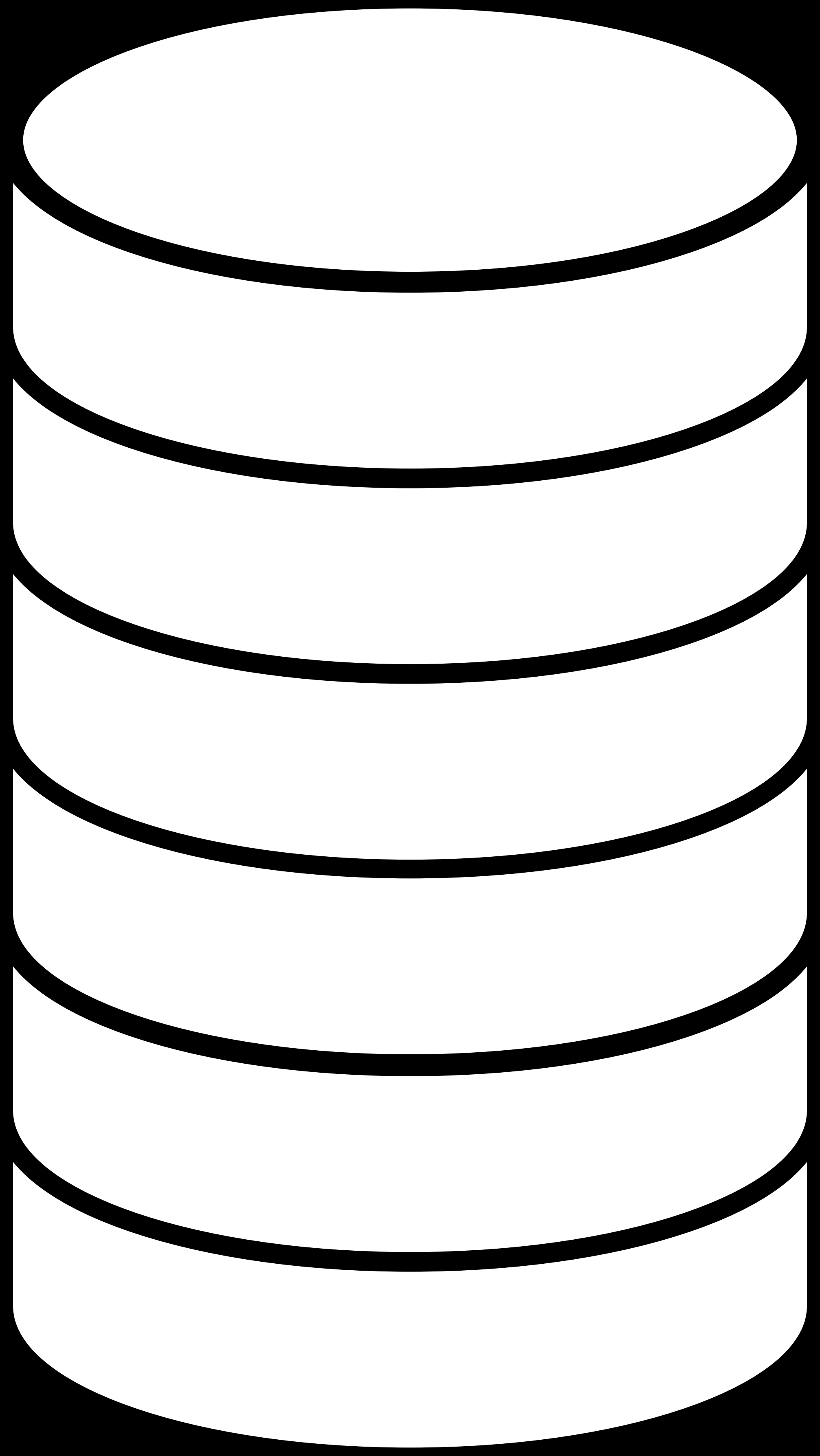
5,000 writes/minute

100,000 reads/minute

50,000 writes/minute

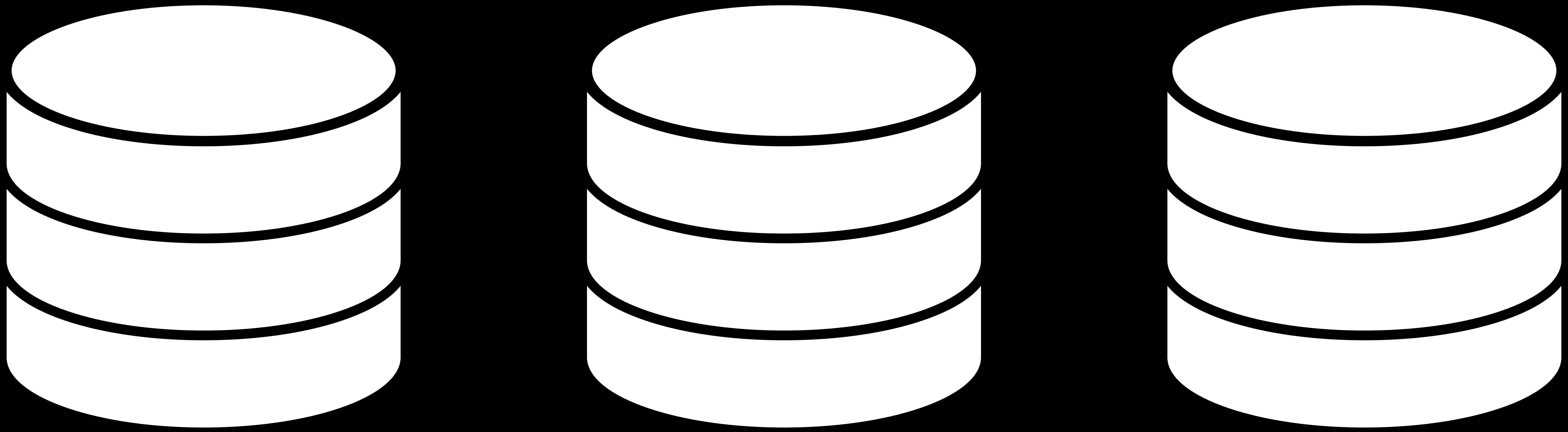
Vertical Scaling

Increasing capacity by increasing a server's computing power



Horizontal Scaling

Increasing capacity by distributing load across multiple servers



Replication

Keeping copies of a database on multiple servers

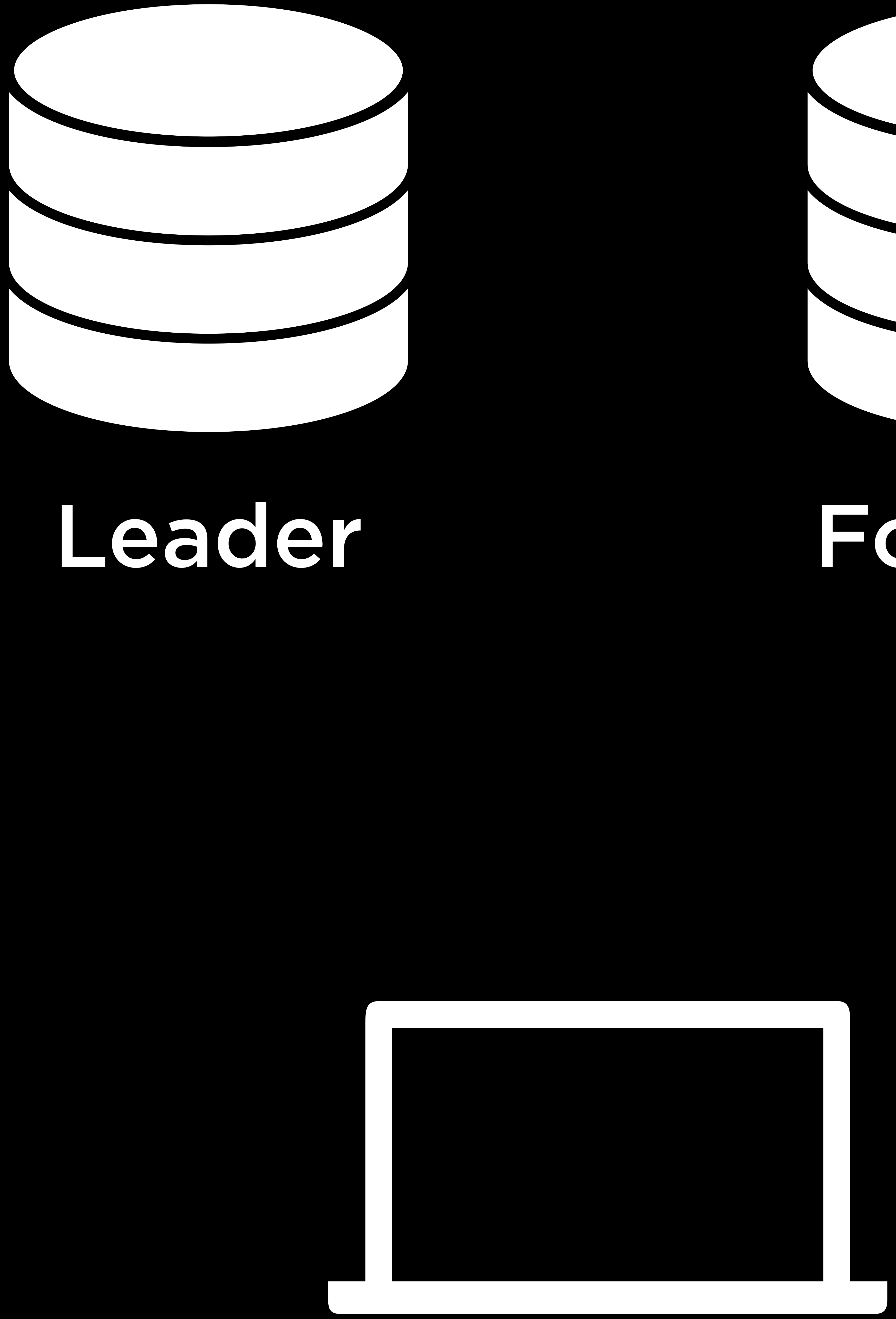
Single-Leader

Multi-Leader

Leaderless

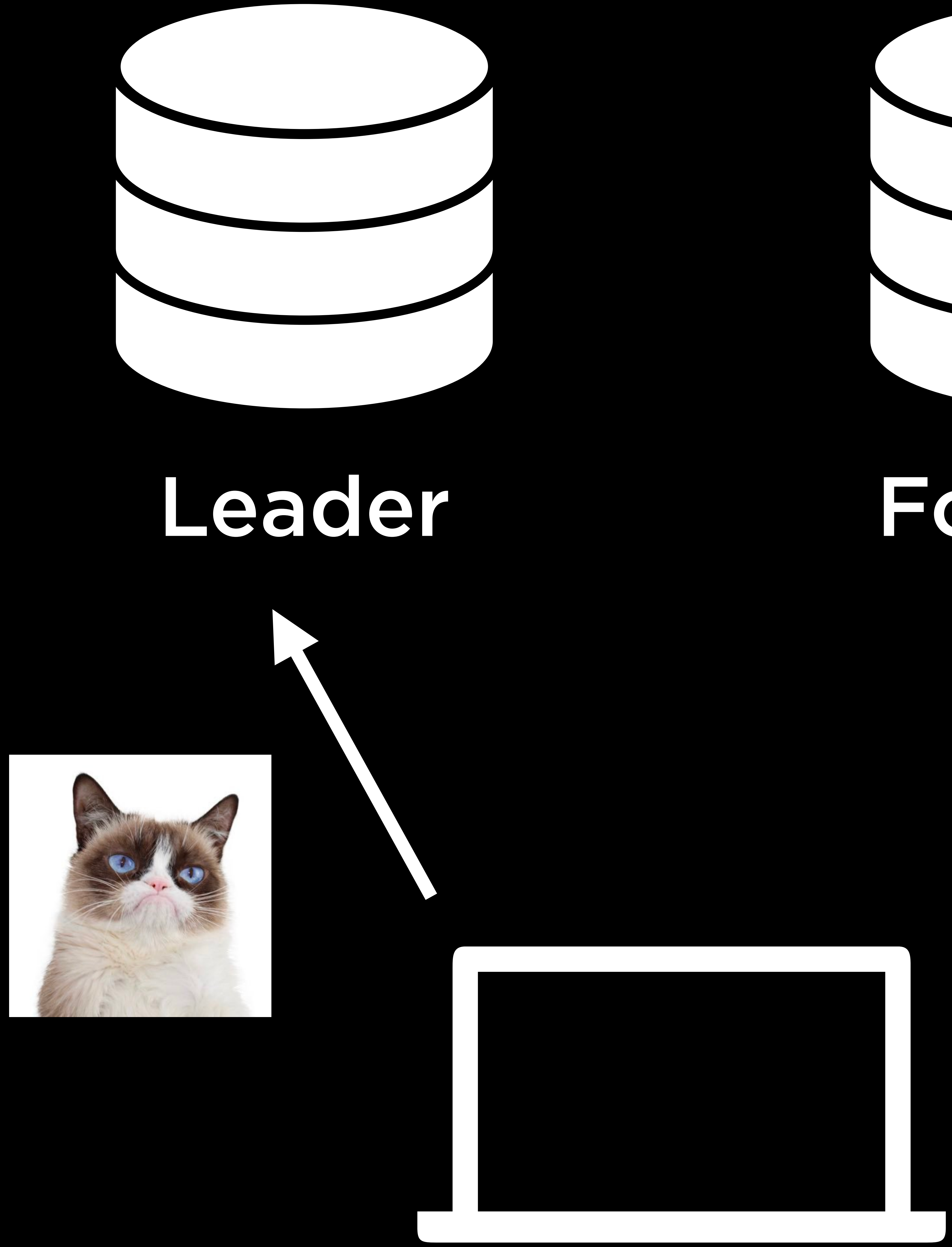
...

Single-Leader



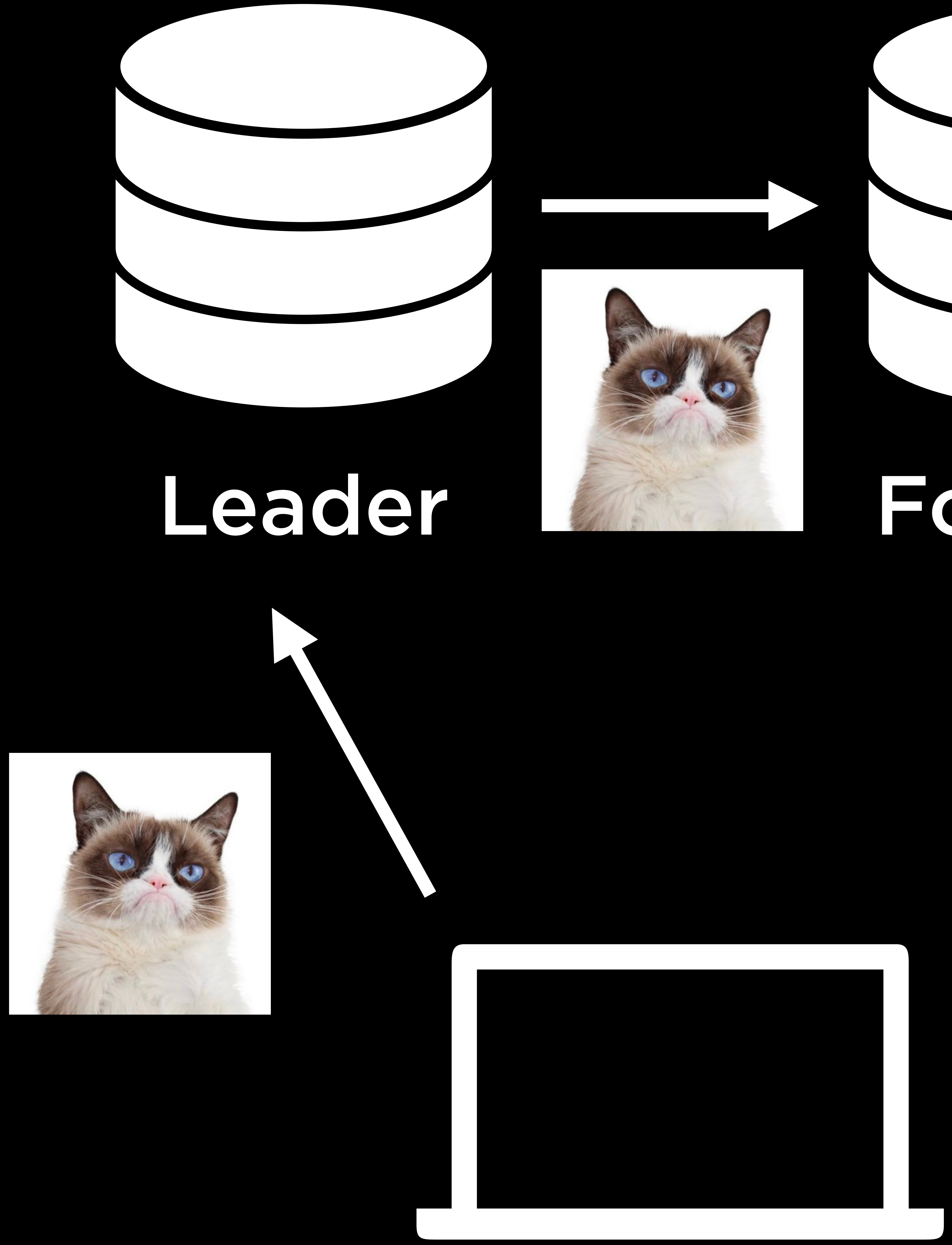
Leader

Follower



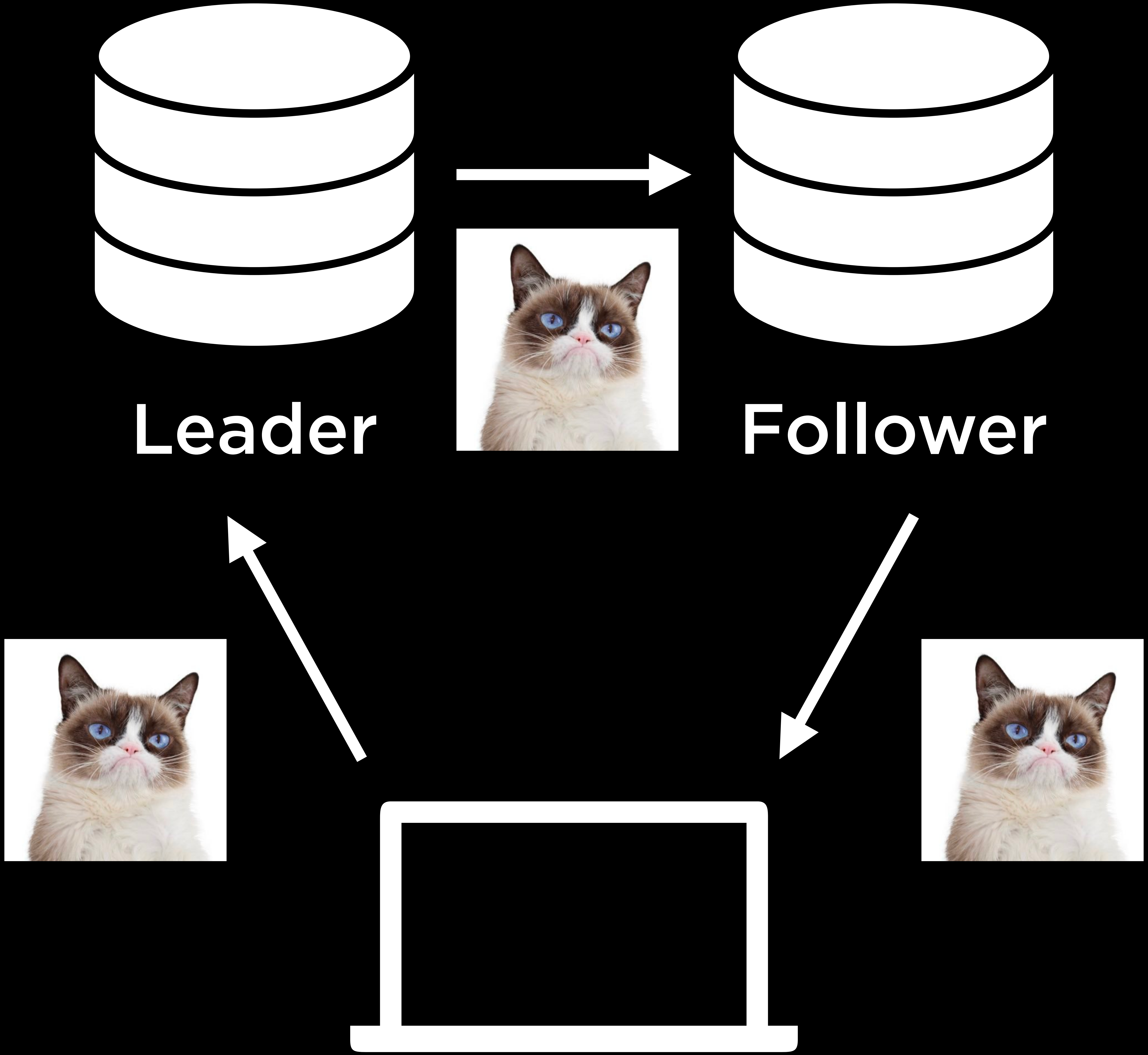
Leader

Follower



Leader

Follower



Leader

Follower

Read Replica

A copy of a database from which data may only be read

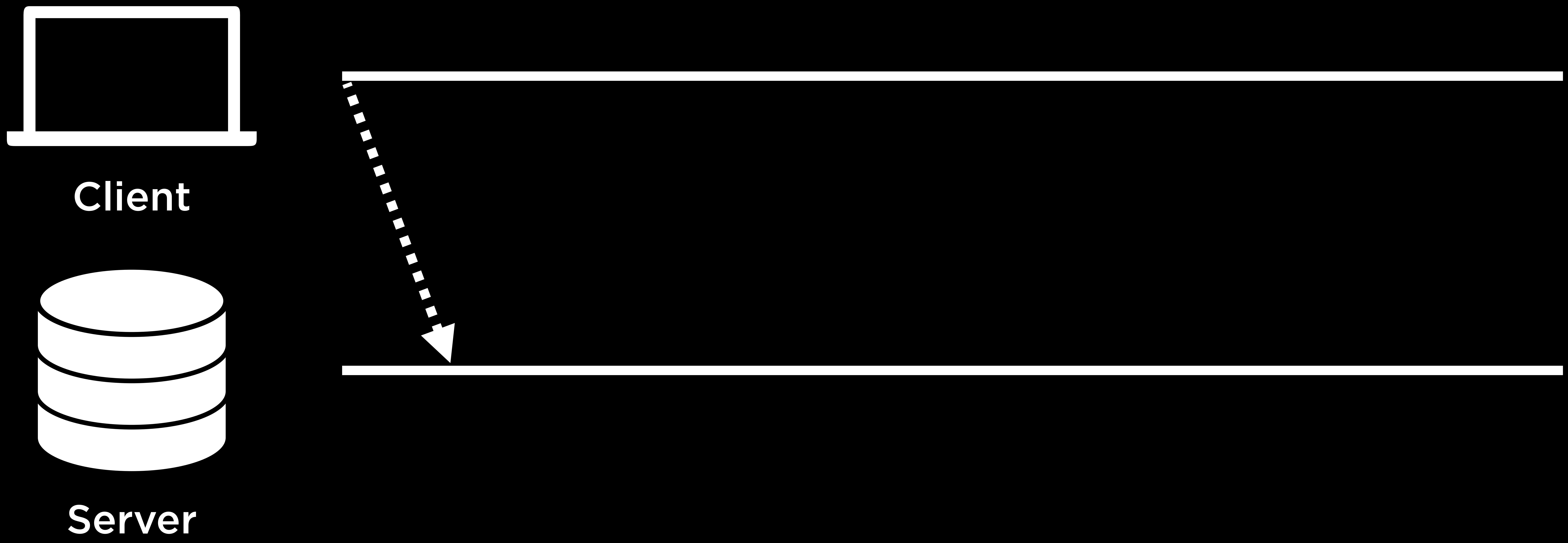
Synchronous

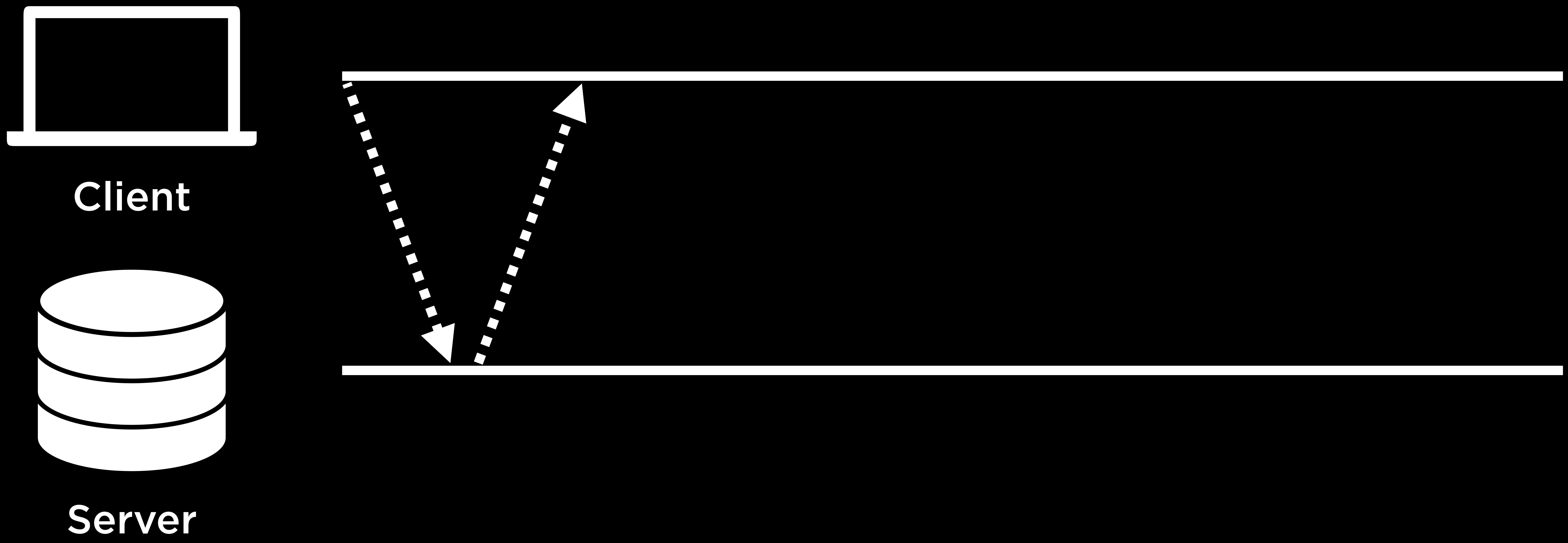
Asynchronous

Synchronous

Client

Server

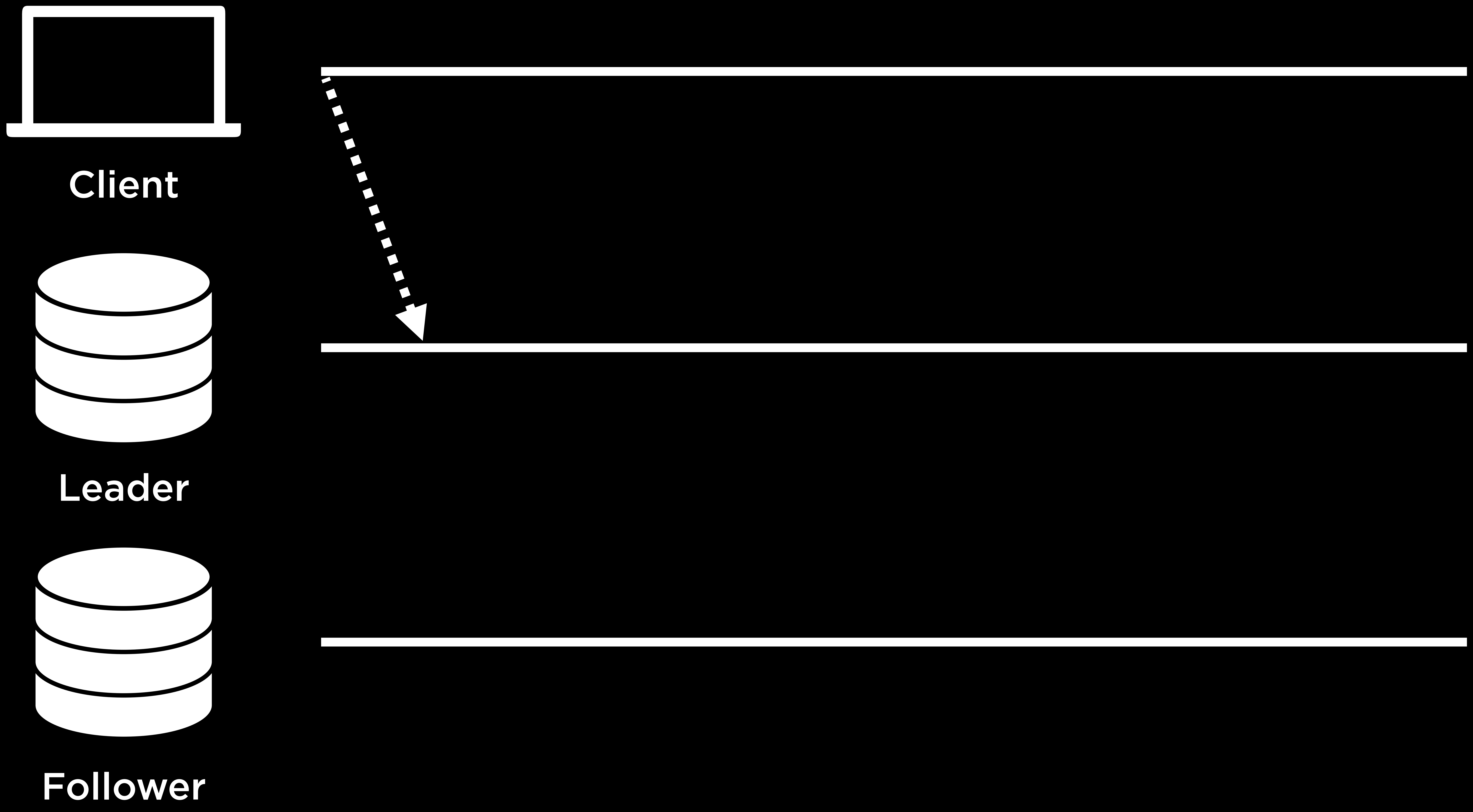

Client
Server

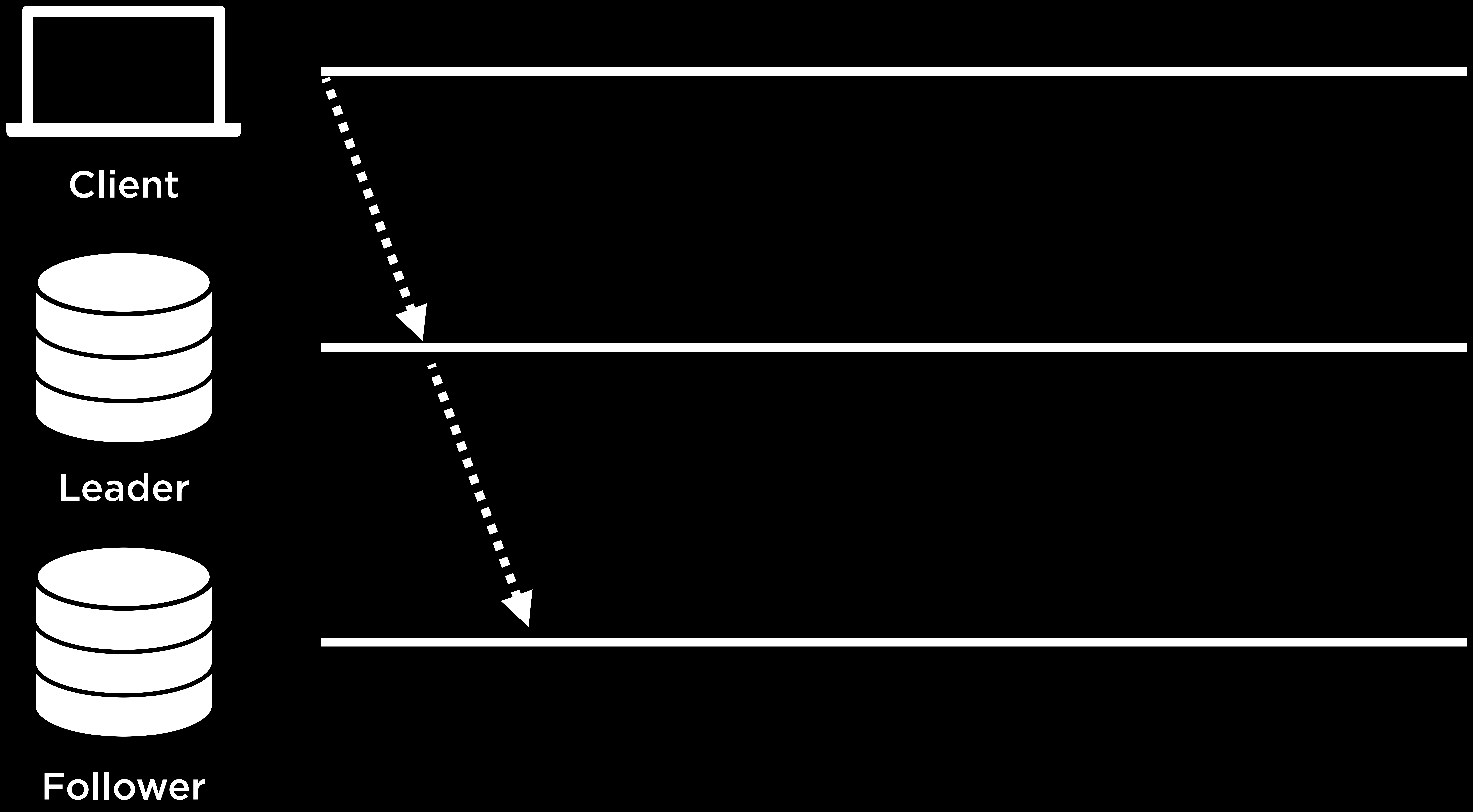

Client
Server

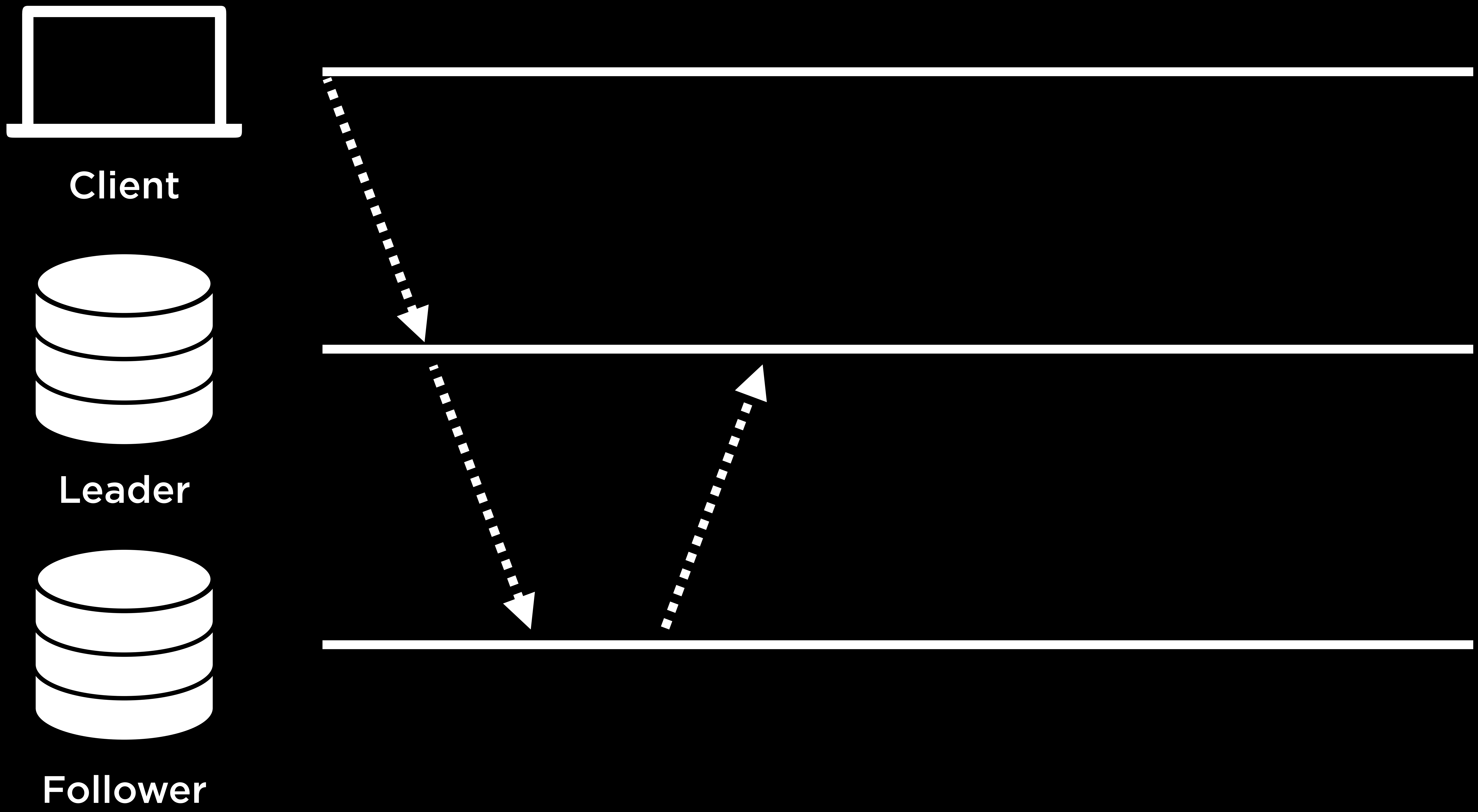
Client

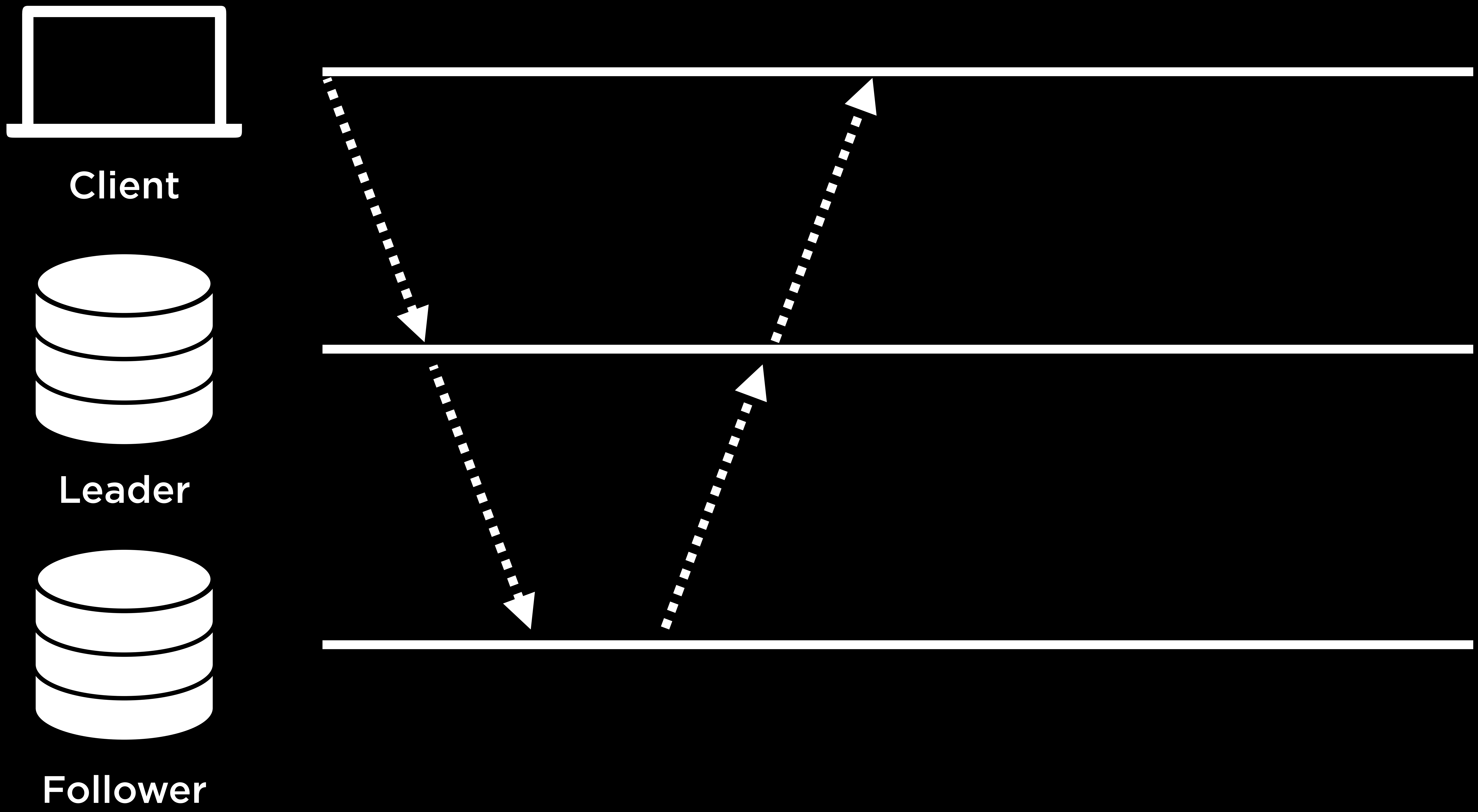
Leader

Follower

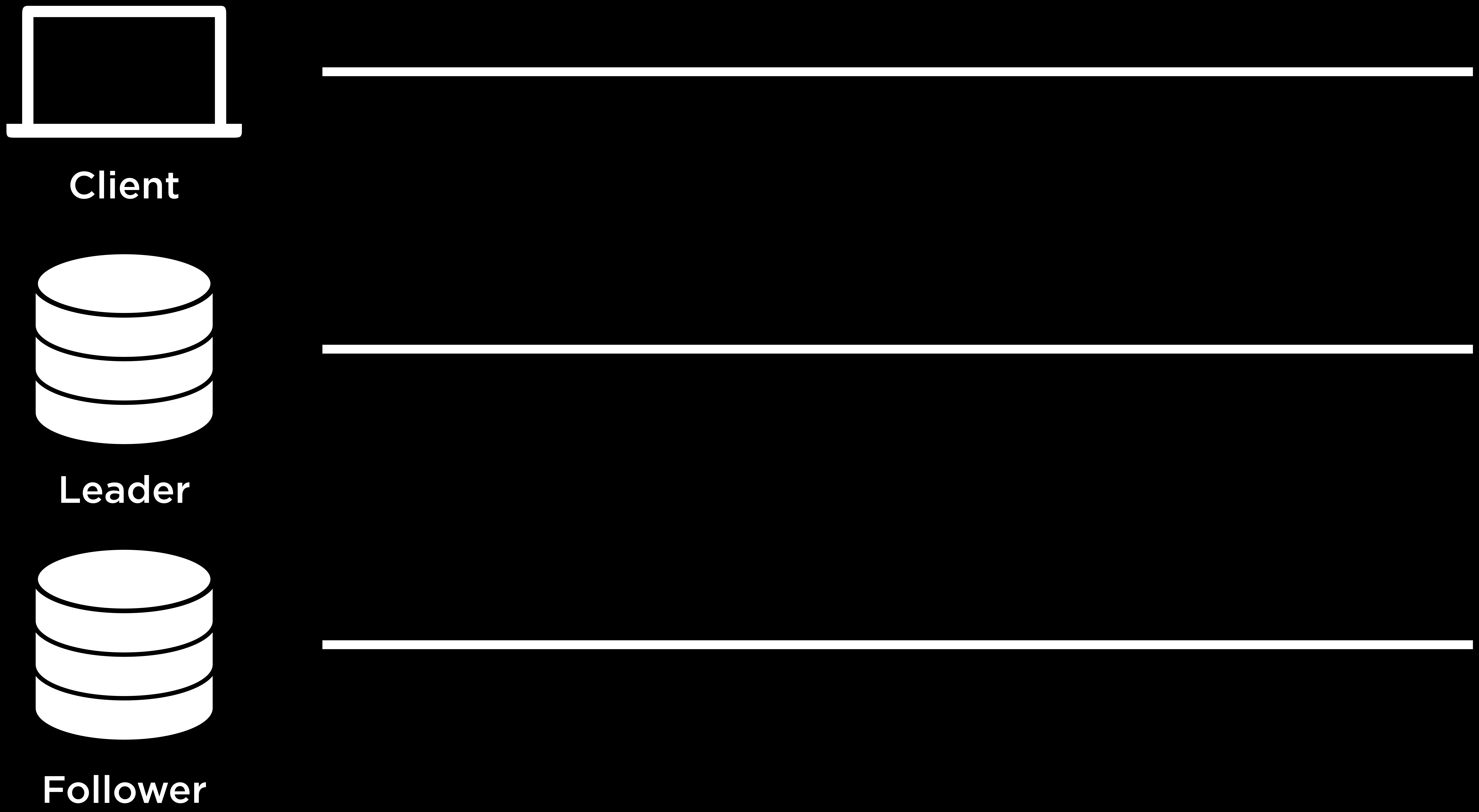

Client
Leader
Follower


Client
Leader
Follower


Client
Leader
Follower


Client
Leader
Follower

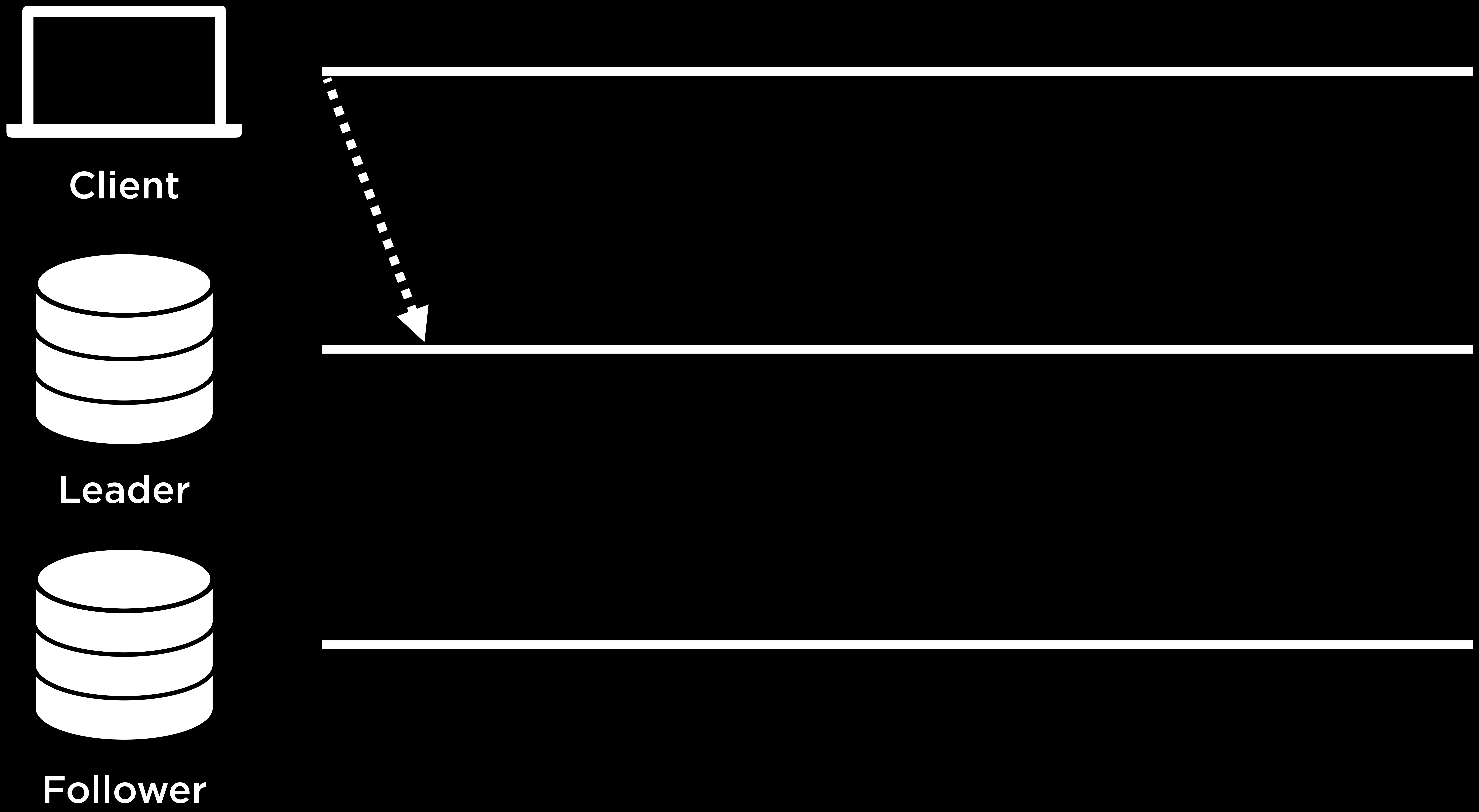
Asynchronous



Client

Leader

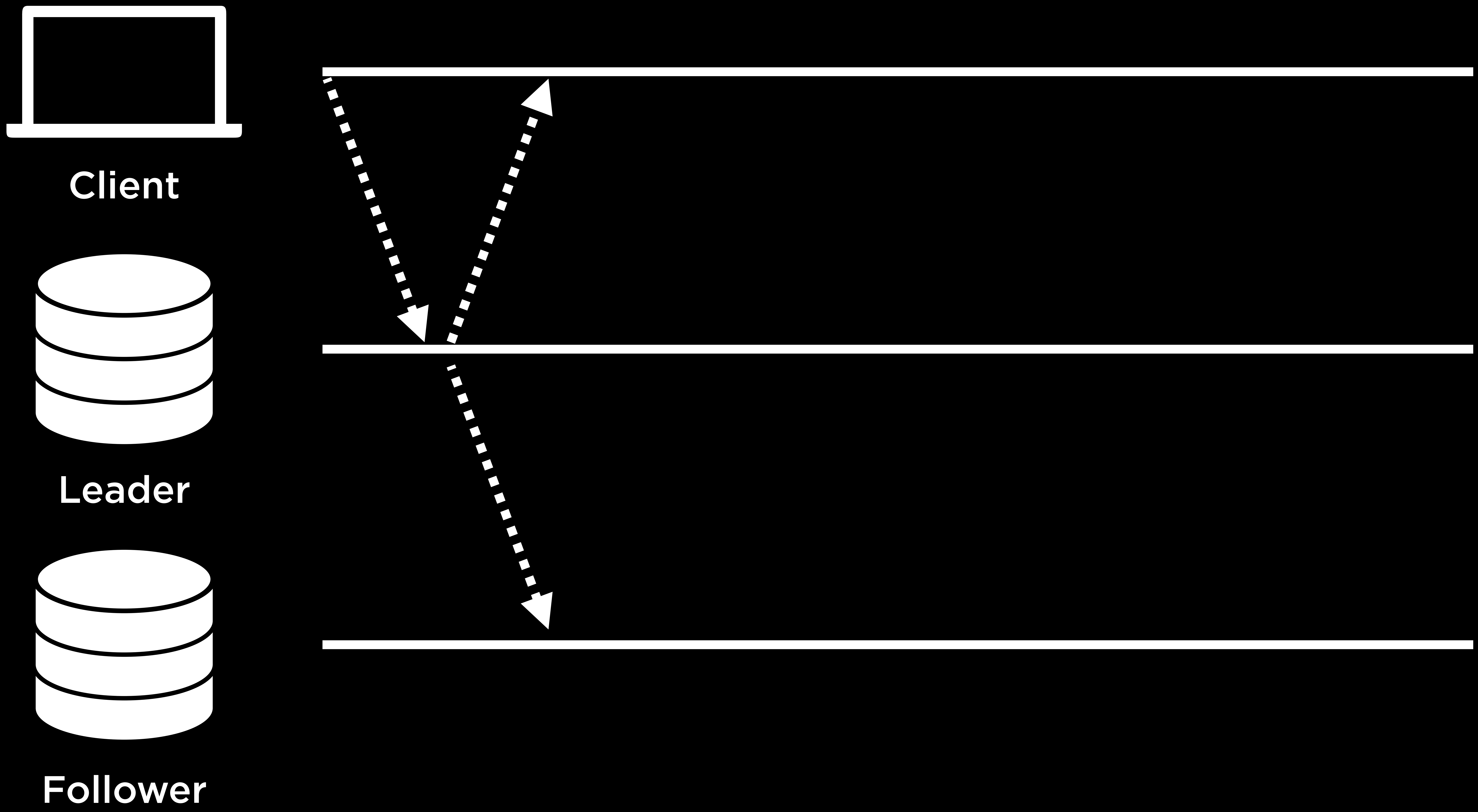
Follower

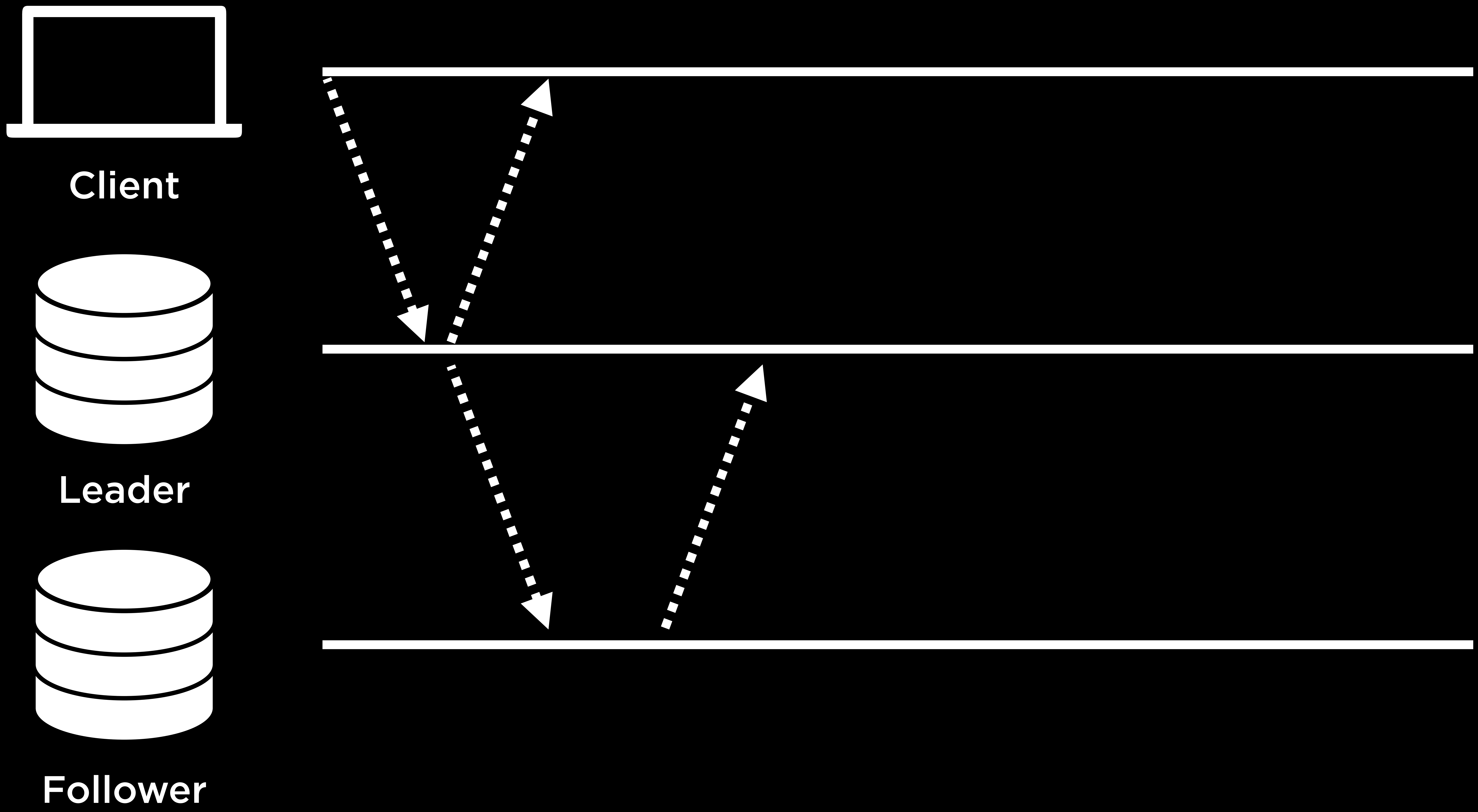


Client

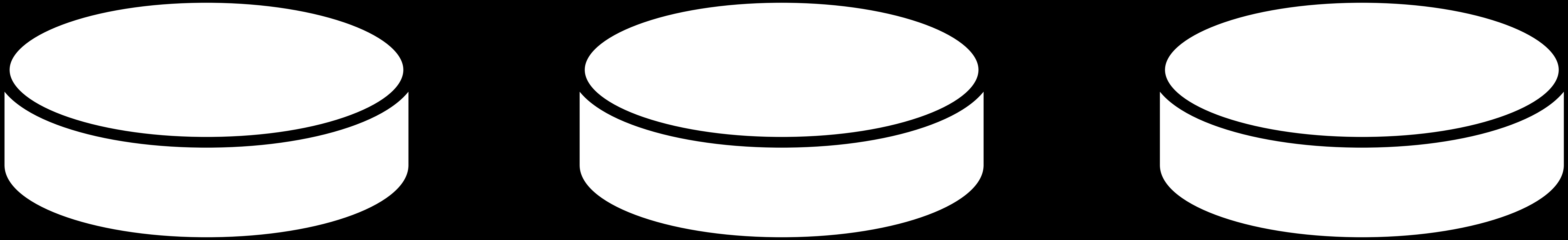
Leader

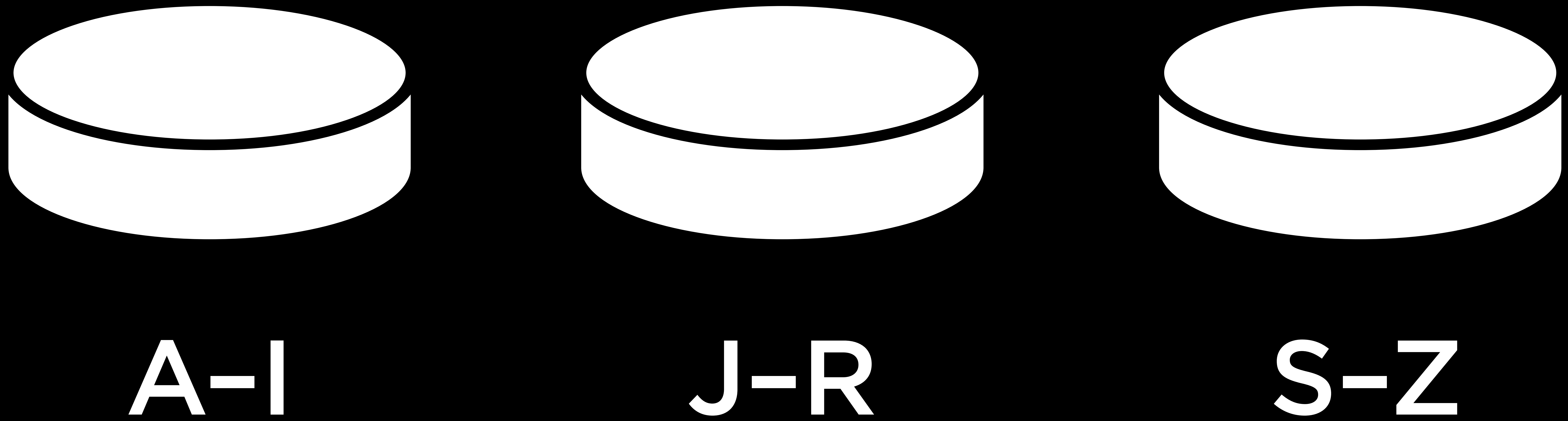
Follower


Client
Leader
Follower


Client
Leader
Follower

Sharding

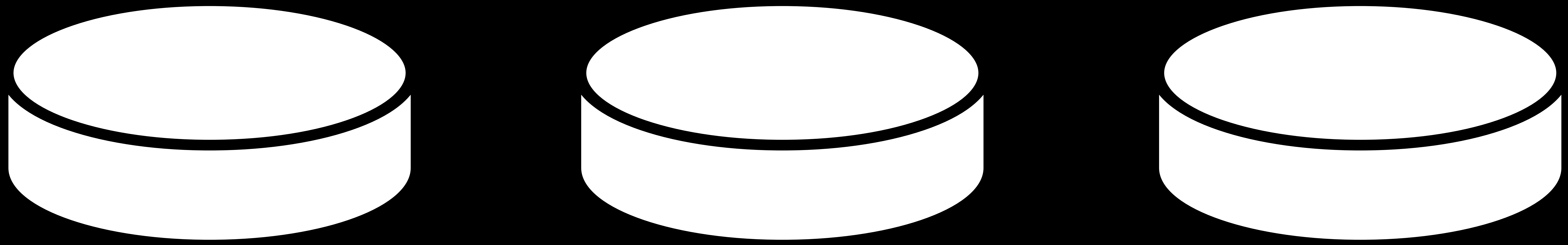




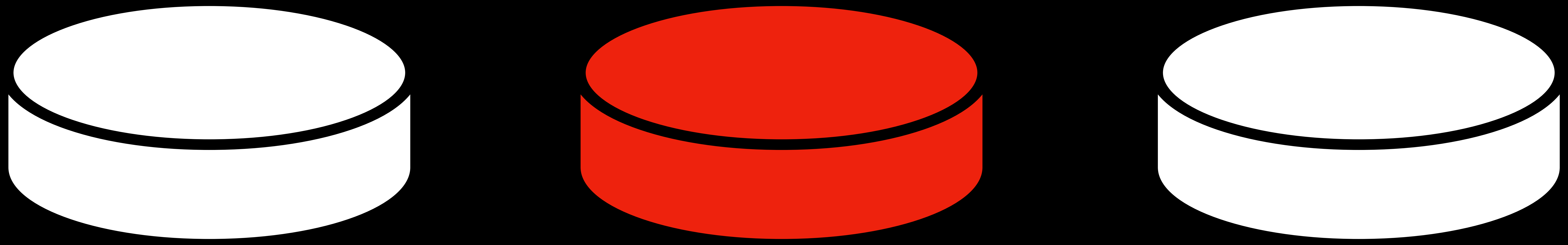
A–I

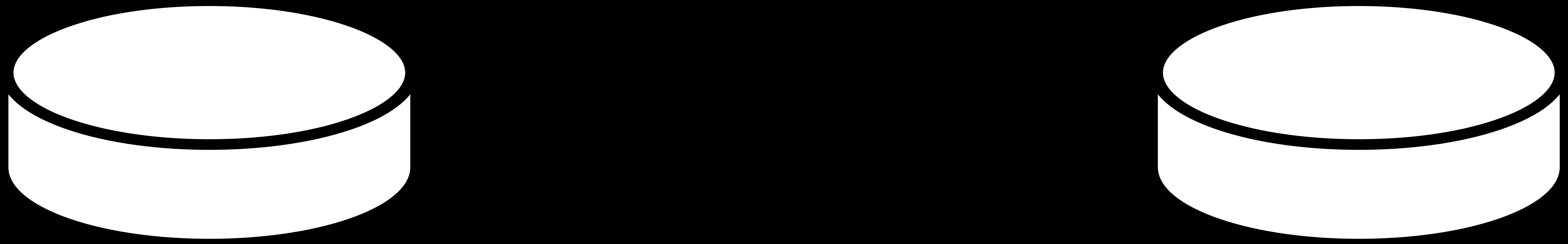
J–R

S–Z



1–1000 1001–2000 2001–3000





Access Control

```
CREATE USER …;
```

```
GRANT privilege, … TO user;
REVOKE privilege, … FROM user;
```

ALL

CREATE

INSERT

SELECT

UPDATE

DELETE

…

SQL Injection Attacks

```
SELECT `id` FROM `users`
WHERE `username` = '…'
AND `password` = '…';
```

```
SELECT `id` FROM `users`
WHERE `username` = 'Carter'
AND `password` = '…';
```

```
SELECT `id` FROM `users`
WHERE `username` = 'Carter'
AND `password` = 'password';
```

```
SELECT `id` FROM `users`
WHERE `username` = 'Carter'
AND `password` = '…';
```

```
SELECT `id` FROM `users`
WHERE `username` = 'Carter'
AND `password` = 'password'
OR '1' = '1';
```

```
SELECT `balance` FROM `accounts`
WHERE `id` = …;
```

```
SELECT `balance` FROM `accounts`
WHERE `id` = 123456789;
```

```
SELECT `balance` FROM `accounts`
WHERE `id` = …;
```

```
SELECT `balance` FROM `accounts`
WHERE `id` = 123456789 UNION
SELECT `id` FROM `accounts`;
```

Prepared Statements

```
PREPARE name FROM statement;
```

```
SELECT `balance` FROM `accounts`
WHERE `id` = ?;
```

```
EXECUTE name USING value0, …;
```

Querying

Relating

Designing

Writing

Viewing

Optimizing

Scaling

Introduction to

Databases with SQL

Scaling