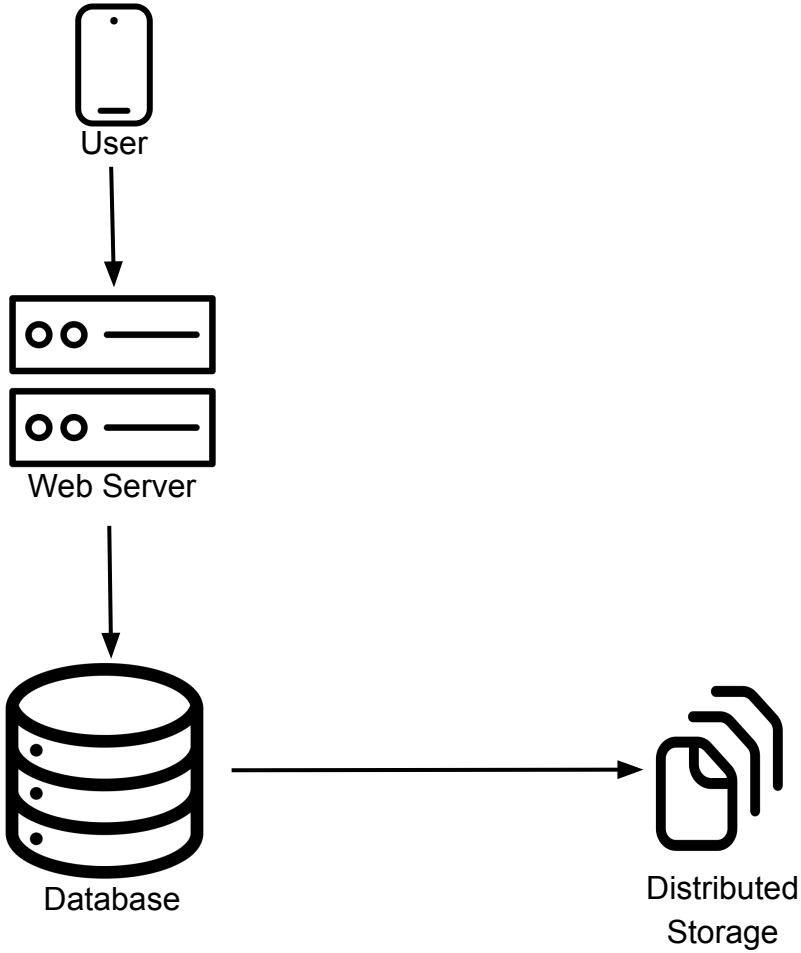
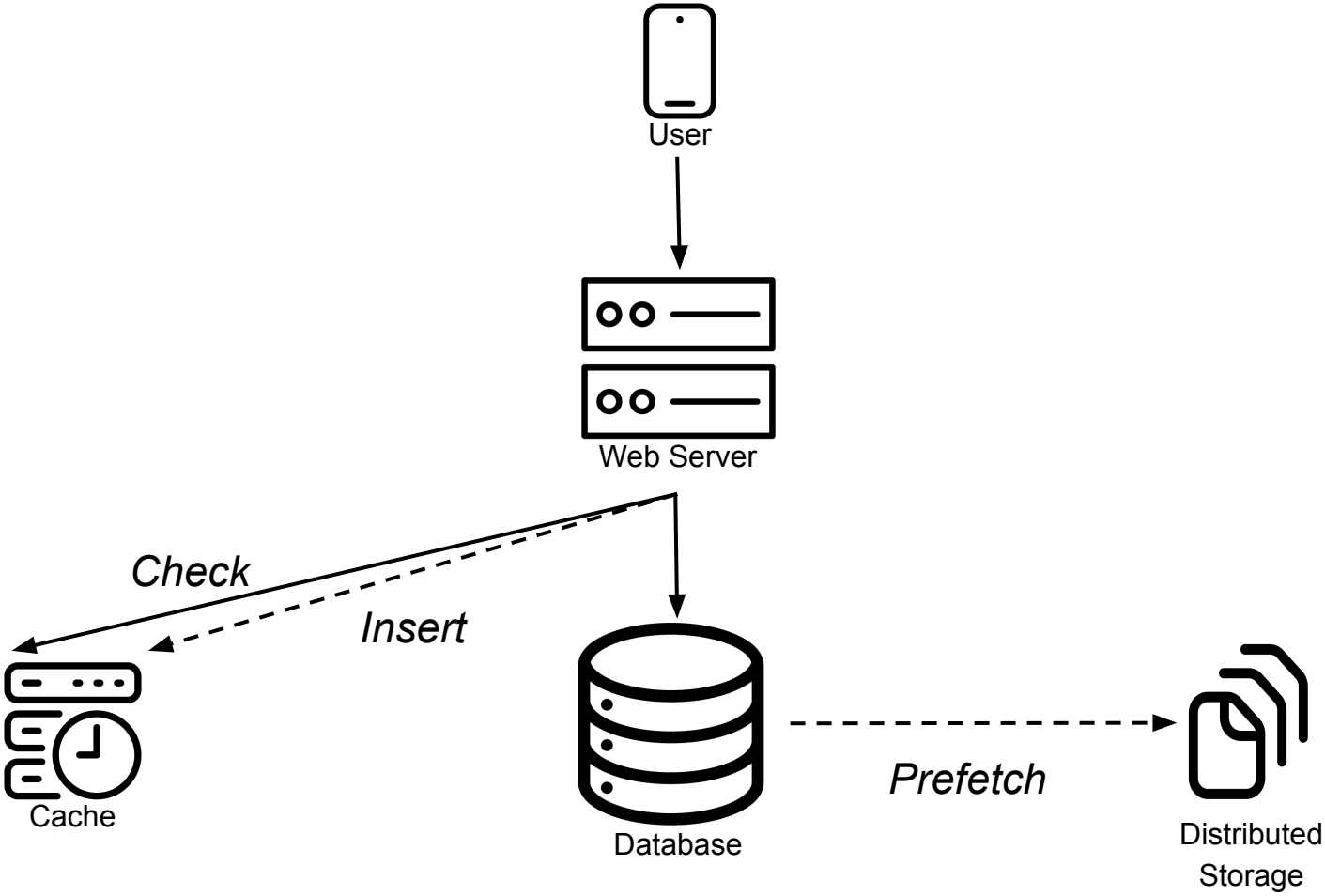
CASH RULES EVERYTHING AROUND ME

Michael Abebe (Salesforce)

CACHE
RULES
EVERYTHING
AROUND
ME

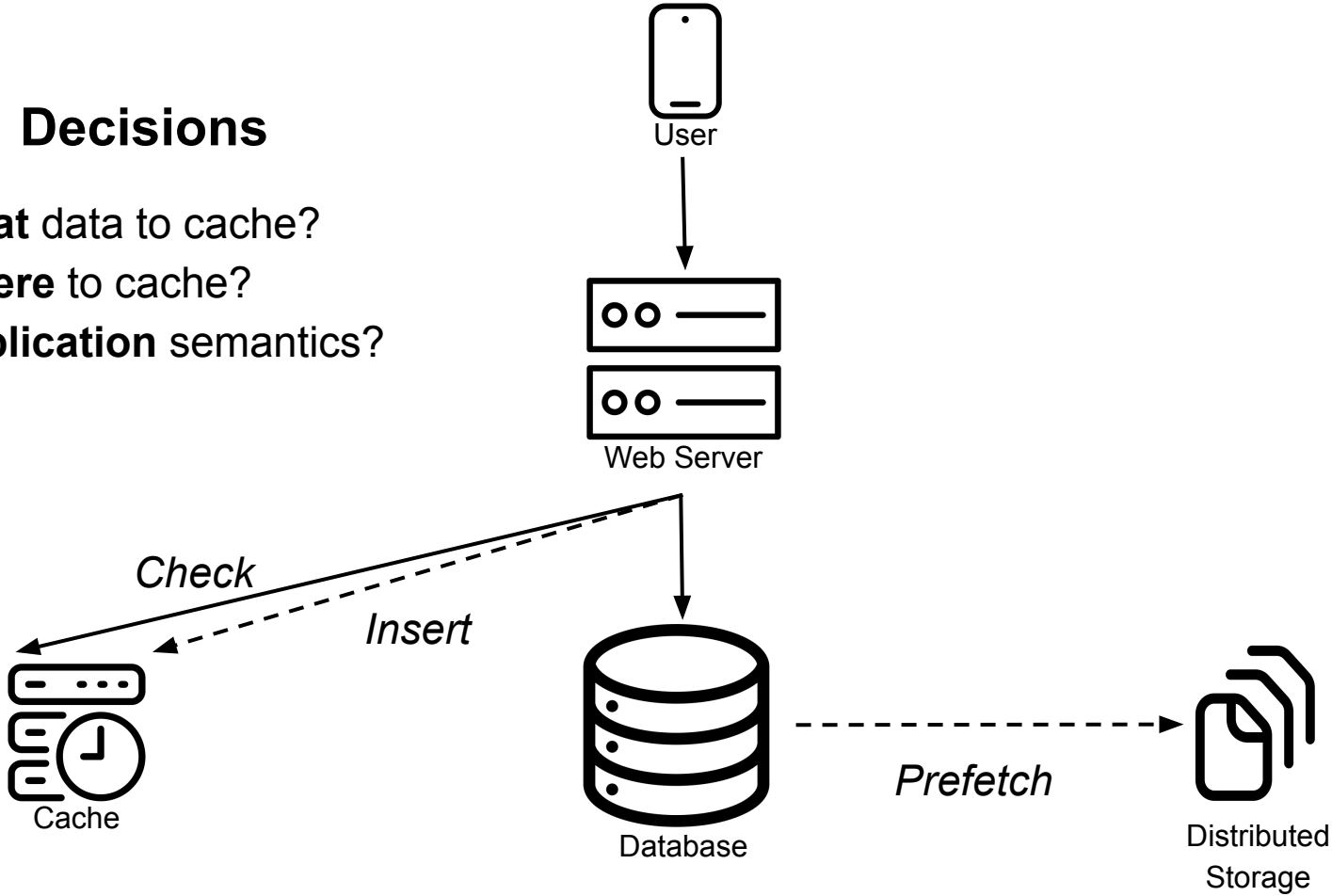
Michael Abebe (Salesforce)


User
Web Server
Database
Distributed
Storage


User
Web Server
Check
Insert
Cache
Database
Prefetch
Distributed
Storage

Decisions

- **What** data to cache?
- **Where** to cache?
- **Application** semantics?



CACHES
REPLICATE
EVERYTHING
AROUND
ME

Michael Abebe (Salesforce)

Caching is a form of *replication*

Caching Decisions

- **What** data to cache?
- **Where** to cache?
- **Application** semantics?

Caching is a form of *replication*

Caching Decisions *become* replication decisions

- **What** data to ~~cache~~ replicate?
- **Where** to ~~cache~~ replicate?
- **Application** semantics?

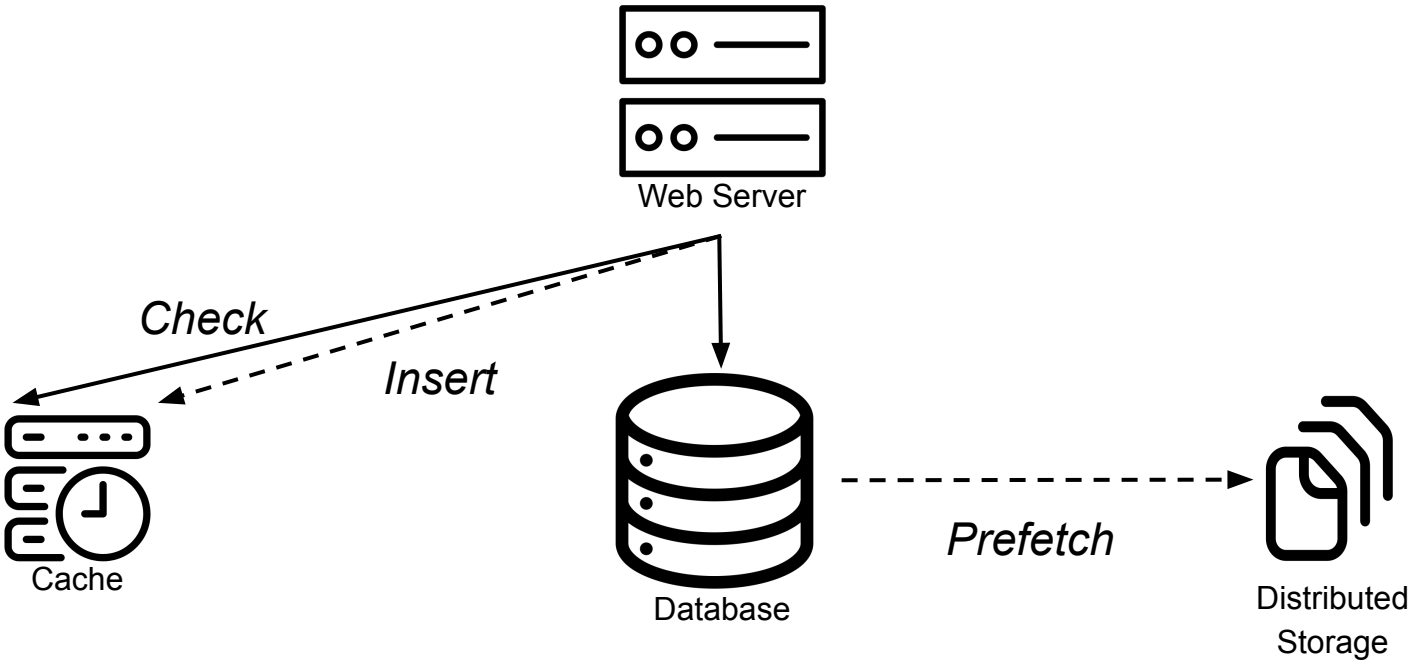
Caching is a form of *adaptive replication*

Caching Decisions *become* adaptive replication decisions

- **What** data to ~~cache~~ replicate?
- **Where** to ~~cache~~ replicate?
- **Application** semantics?

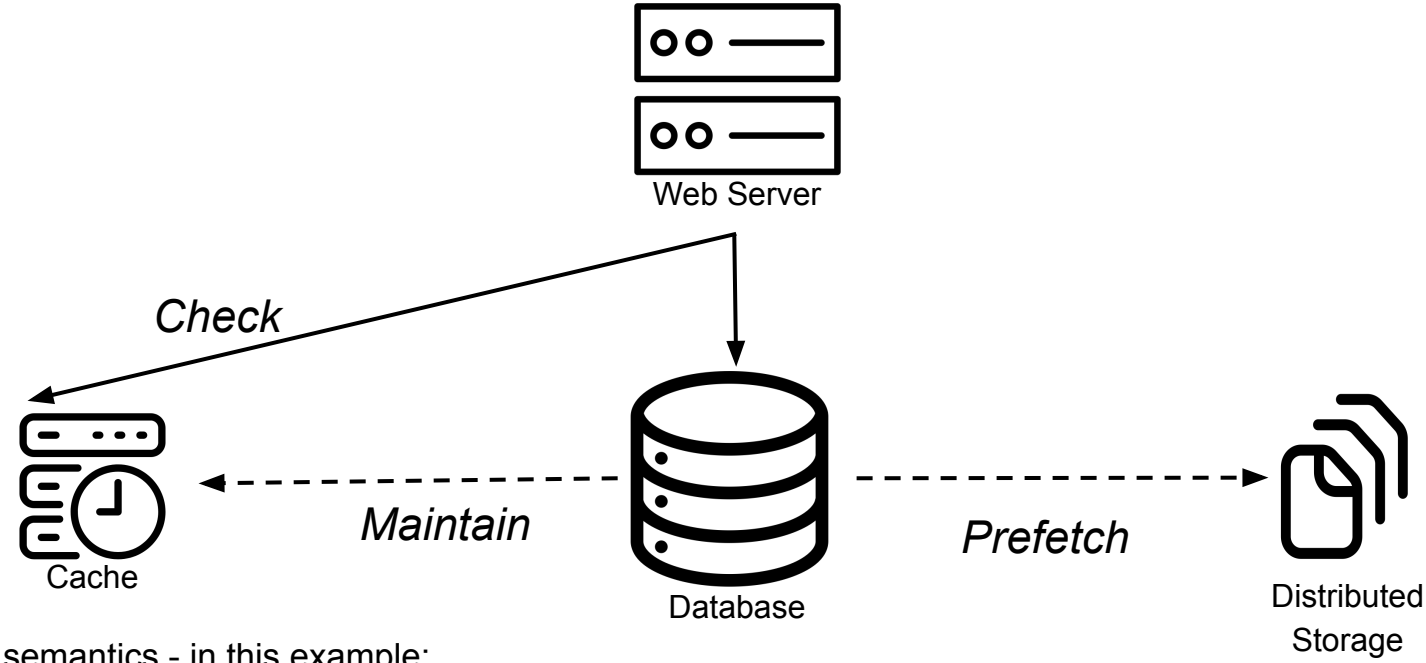
Can the database manage these caches?

- Database can make more **informed decisions** (query/data statistics) that benefit **execution strategies**
- Databases have **defined semantics** (Isolation Levels, Consistency Protocols, etc.)


Web Server
Check
Insert
Cache
Database
Prefetch
Distributed
Storage

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → **Q1('Michael', '1234'),** Snapshot**: 17 → {42}**

SELECT ***MAX***(*ORDER_ID*) **FROM**
ORDERS **WHERE** *O_CUST_ID* = ? → **Q2(42),** Snapshot**: 17 → {9823}**



Application semantics - in this example:
snapshot isolation

Cache Maintenance

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → **Q1('Michael', '1234'),** Snapshot**: 17 → {42}**

SELECT *MAX*(*ORDER_ID***) FROM**
ORDERS **WHERE** *O_CUST_ID* = ? → **Q2(42),** Snapshot**: 17 → {9823}**

INSERT INTO *ORDERS* **(10000, 42, …)** ***Creates*** *new snapshot (23)*

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → ***Cache miss (snapshot advanced)*** → **Q1('Michael', '1234'),** Snapshot**: 23 → {42}**

SELECT *MAX*(*ORDER_ID***) FROM**
ORDERS **WHERE** *O_CUST_ID* = ? → ***Cache miss (snapshot advanced)*** → **Q2(42),** Snapshot**: 23 → {10000}**

Application semantics - in this example:
snapshot isolation

Predictive Cache Maintenance

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → **Q1('Michael', '1234'),** Snapshot**: 17 → {42}**

SELECT *MAX*(*ORDER_ID*) FROM *ORDERS* **WHERE** *O_CUST_ID* = ? → **Q2(42),** Snapshot**: 17 → {9823}**

INSERT INTO *ORDERS* **(10000, 42, …)** ***Creates*** *new snapshot (23)*

Cache miss
(snapshot advanced)

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → **Q1('Michael', '1234'),** Snapshot**: 23 → {42}**

Predictively execute

SELECT *MAX*(*ORDER_ID*) FROM *ORDERS* **WHERE** *O_CUST_ID* = ? — ***Cache hit*** → **Q2(42),** Snapshot**: 23 → {10000}**

Application semantics - in this example:
snapshot isolation

Predictive Cache Maintenance

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ?

→ **Q1('Michael', '1234'),** Snapshot**: 17** → **{42}**

SELECT *MAX*(*ORDER_ID***) FROM**
ORDERS **WHERE** *O_CUST_ID* = ?

→ **Q2(42),** Snapshot**: 17** → **{9823}**

INSERT INTO *ORDERS* **(10000, 42, …)**

Creates *new snapshot (23)*

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ?

Predictively execute

SELECT *MAX*(*ORDER_ID***) FROM**
ORDERS **WHERE** *O_CUST_ID* = ?

Predictive execution based on learned query correlations and query parameters

Application semantics - in this example:
snapshot isolation

Predictive Cache Maintenance

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? → **Q1('Michael', '1234'),** Snapshot**: 17 → {42}**

SELECT ***MAX****(ORDER_ID)* **FROM** *ORDERS* **WHERE** *O_CUST_ID* = ? → **Q2(42),** Snapshot**: 17 → {9823}**

INSERT INTO *ORDERS* **(10000, 42, …)** ***Creates*** *new snapshot (23)* ***and predictively maintain cache***

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ? —***Cache hit***→ **Q1('Michael', '1234'),** Snapshot**: 23 → {42}**

SELECT ***MAX****(ORDER_ID)* **FROM** *ORDERS* **WHERE** *O_CUST_ID* = ? —***Cache hit***→ **Q2(42),** Snapshot**: 23 → {10000}**

Application semantics - in this example:
snapshot isolation

Predictive Cache Maintenance

SELECT *C_ID* **FROM** *CUSTOMER*
WHERE *C_NAME* = ? **AND** *C_PASS* = ?

→ **Q1('Michael', '1234'),** Snapshot**: 17** → **{42}**

SELECT *MAX*(*ORDER_ID***) FROM** *ORDERS* **WHERE** *O_CUST_ID* = ?

→ **Q2(42),** Snapshot**: 17** → **{9823}**

INSERT INTO *ORDERS* **(10000, 42, …)**

Creates *new snapshot (23)* ***and predictively maintain cache***

SELECT *C_ID*, ***MAX***(*ORDER_ID*) **FROM** *CUSTOMER*, *ORDERS*
WHERE *O_CUST_ID* = C_ID **AND** *C_NAME* = ? **AND** *C_PASS* = ?

→ **Q12('Michael', '1234'),** Snapshot**: 17** → **{42, 10000}**

Query rewriting and query containment challenge

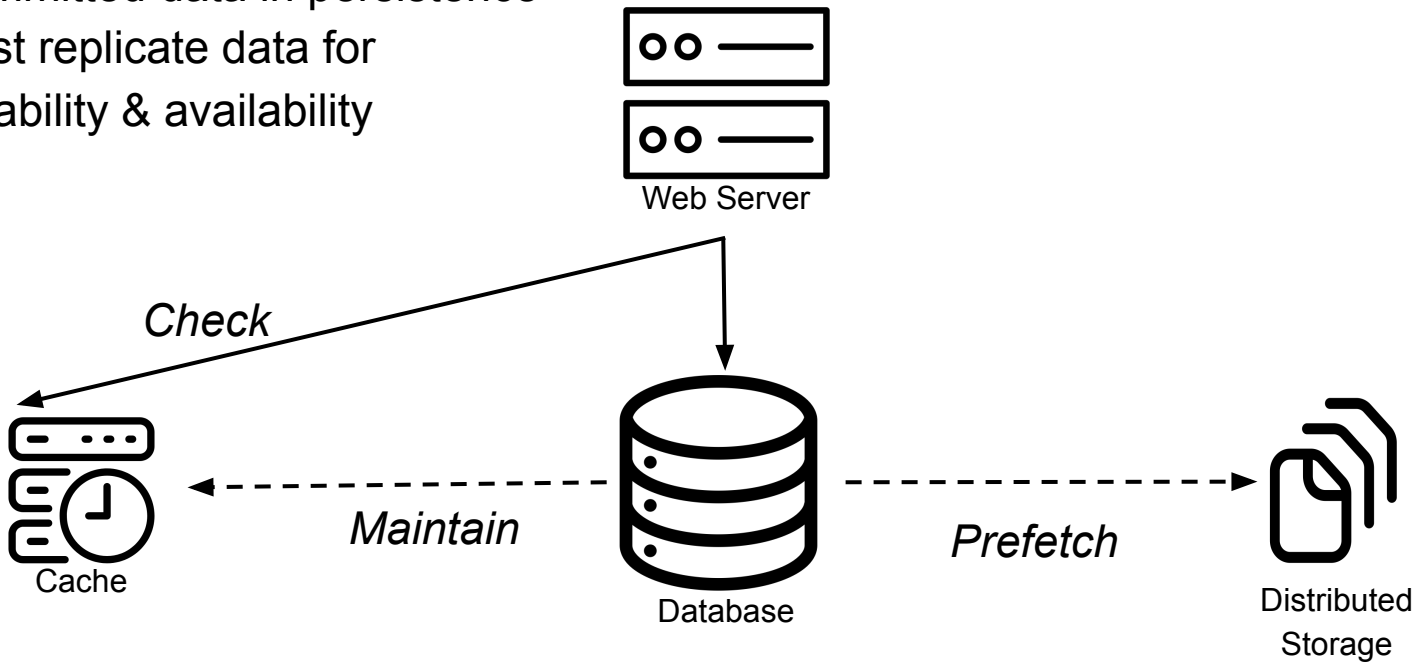
Database has formalized semantics!

Result cache is a form of a replicated materialized view!

Distributed Storage

Distributed Storage stores

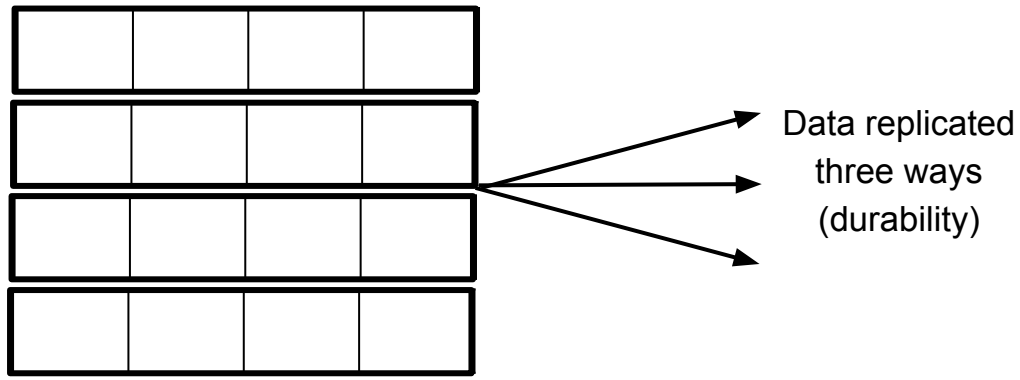
- Committed data in persistence
- Must replicate data for durability & availability

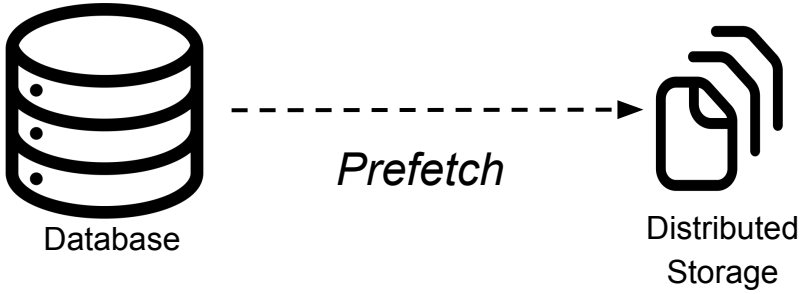


Alternative Data Layouts & Caches

Distributed Storage stores

- Committed data in persistence
- Must replicate data for durability & availability





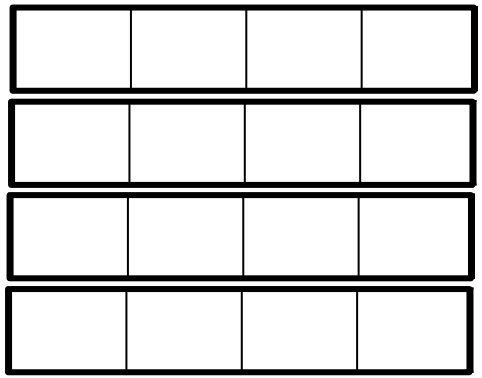
Alternative Data Layouts & Caches

Distributed Storage stores

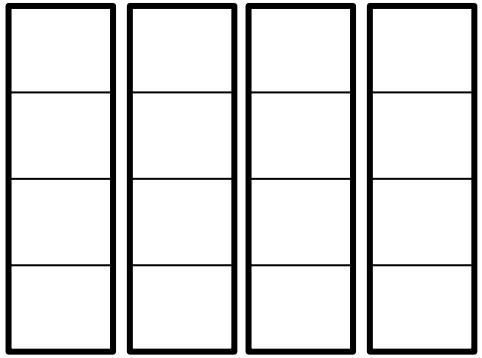
- Committed data in persistence
- Must replicate data for durability & availability

Database can

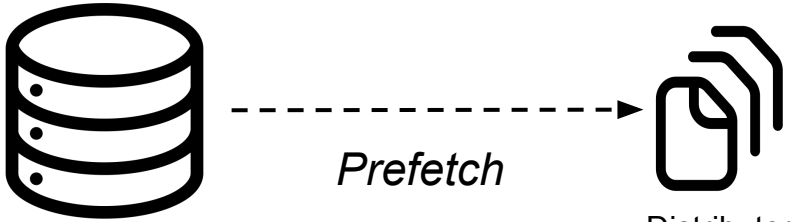
- Store the same data in alternative layouts
- Layout dependent access driven by queries



Row Layout (OLTP)



Column Layout (OLAP)



Database

Distributed Storage

Data Tiering & Caches

Distributed Storage stores

- Committed data in persistence
- Must replicate data for durability & availability

Storage Tiers

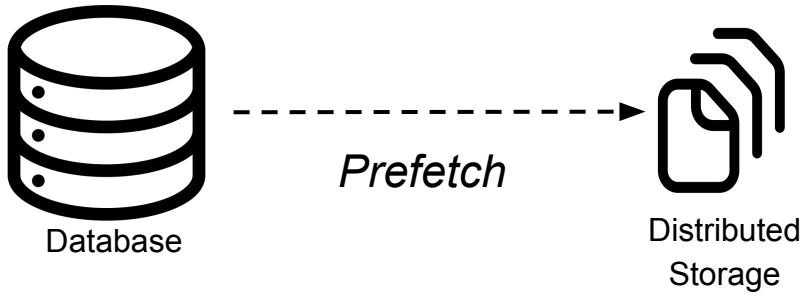
- Memory, NVM, SDD, HDD, Tape
- …

Database knows workload/semantics

- Append-only, unlikely to be read (e.g. audit trail)
- Read-write (e.g. product orders)
- Optional and recreatable (eg. search index)

Database can

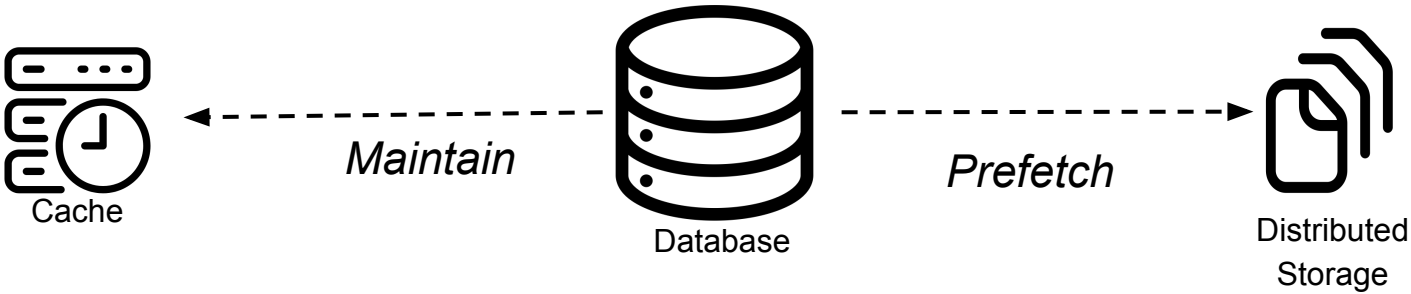
- **Select** storage tier and degree of replication



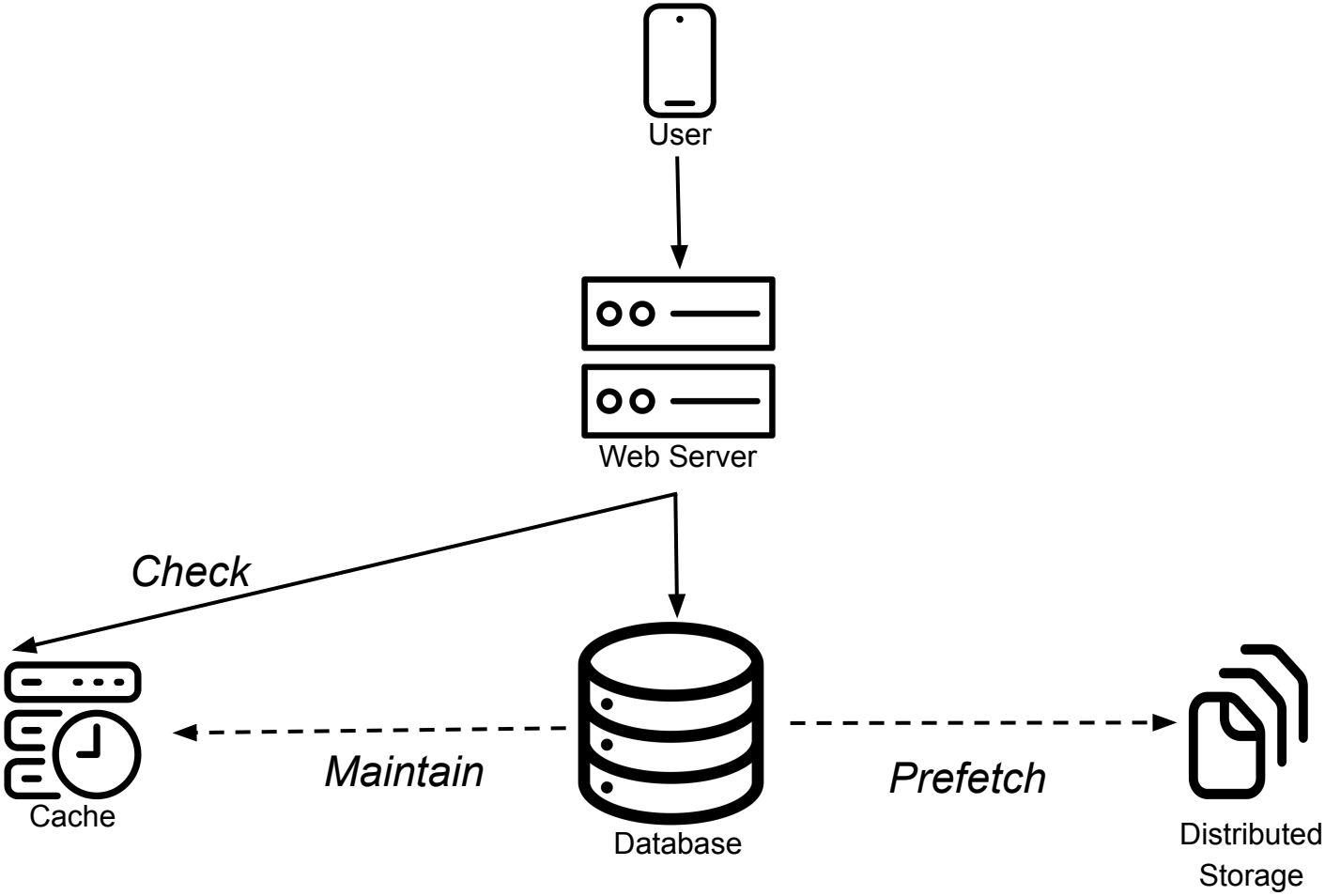
Making Caching/Adaptive Replication Decisions

What to Cache and Where?

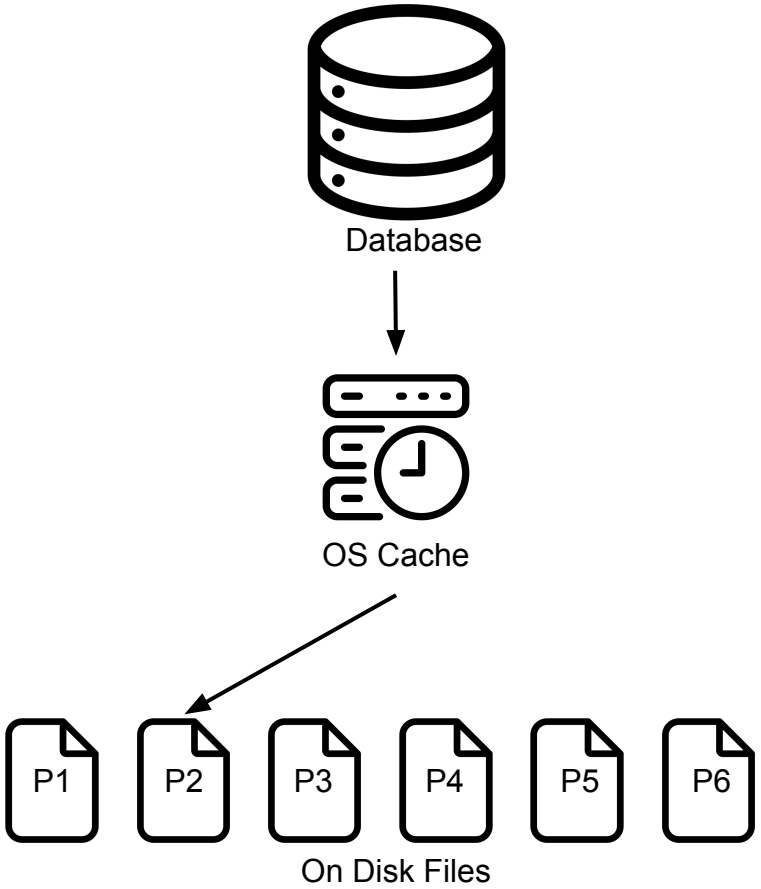
- Constraint driven by resource limitations and semantics
  - Finite memory, network bandwidth, CPU, isolation,etc.
- Optimize for performance metrics
  - (Tail) Latency, Throughput
- Online (learned) decisions driven by workload observations



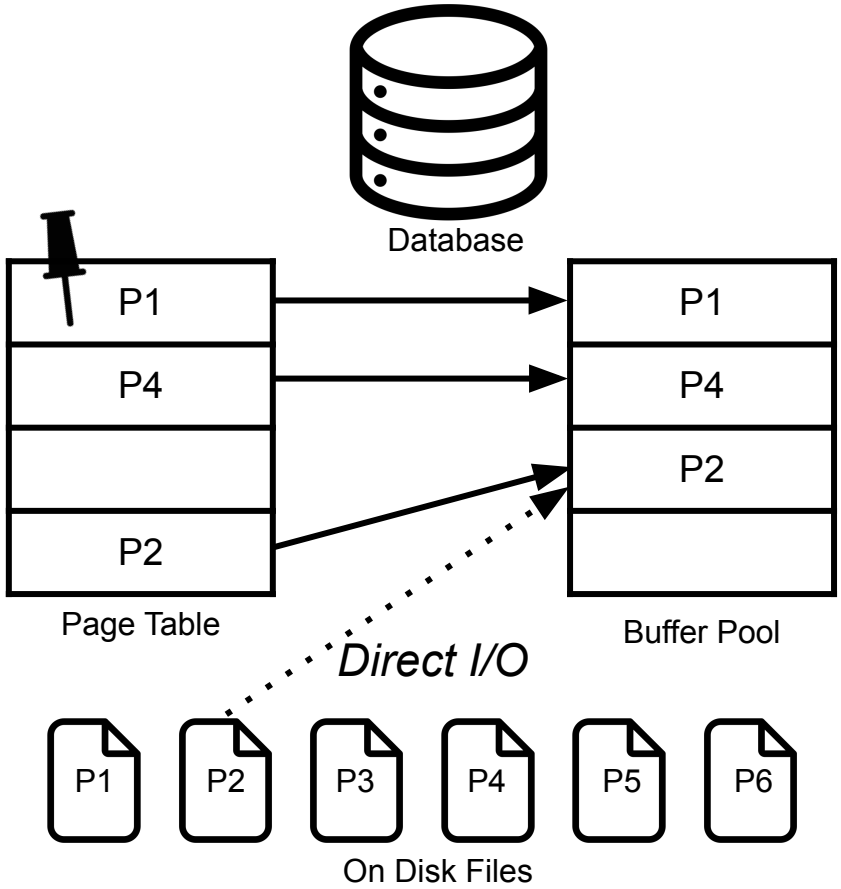
Separation of Concerns?



Separation of Concerns?



Separation of Concerns?



Databases have always broken abstractions!

- Performance
- Resource management
- Domain knowledge of access patterns

Caches Replicate Everything Around Me

Caches are a form of adaptive database replication, let the database manage them!

- **Database** adaptively decide *what data* to cache as replicas and *where*
  - Constrained by resources
  - (Learned) optimizations based on workload observations
- Databases can make more **informed execution decisions** (query/data statistics)
- Leverage database knowledge of semantics and existing protocols
- Trade off in separation of concerns