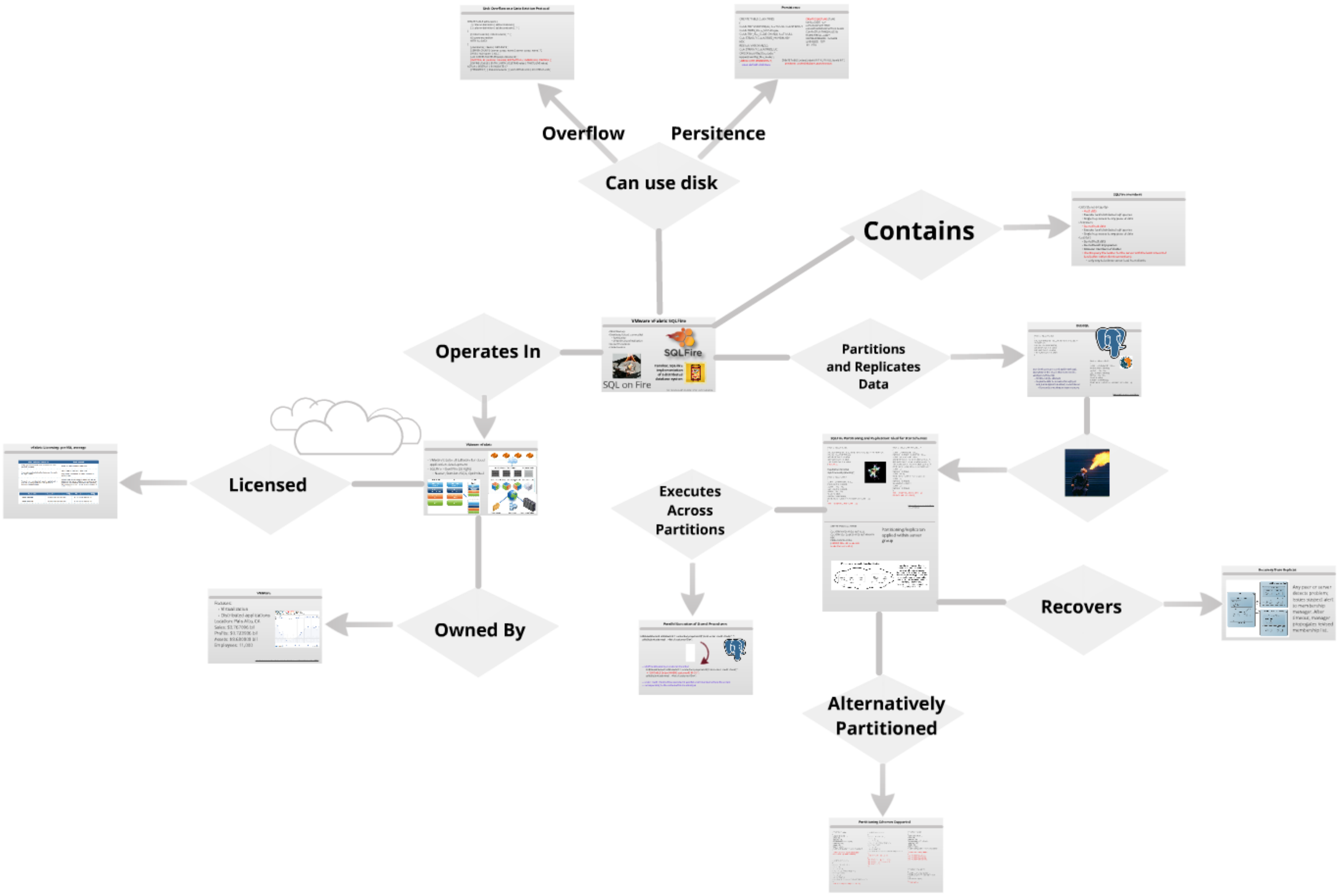


Overflow

Persitence

Can use disk

Contains

Operates In

VMware vFabric SQLFire

SQLFire

SQL on Fire

Partitions and Replicates Data

Licensed

Executes Across Partitions

Recovers

Owned By

Alternatively Partitioned

VMware vFabric SQLFire

- Main Memory
- Distributed (cloud, commodity)
  - Partitioning
  - (A)synchronous Replication
- Stored Procedures
- Closed Source

SQL on Fire

Familiar, SQL-like implementation of a distributed database system

Unless otherwise noted, info and examples are from VMware's documentation.

Operates In

VMware vFabric SQLFire

- Main Memory
- Distributed (cloud, commodity)
  - Partitioning
  - (A)synchronous Replication
- Stored Procedures
- Closed Source

SQLFire

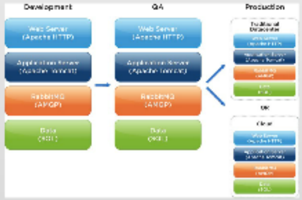
SQL on Fire

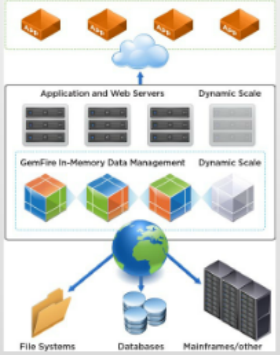
Familiar, SQL-like implementation of a distributed database system

Unless otherwise noted, info and examples are from VMware's documentation.

VMware vFabric

- VMware's suite of software for cloud application development
- SQLFire ≈ GemFire (at right)
  - Newer, Standard SQL, Optimized

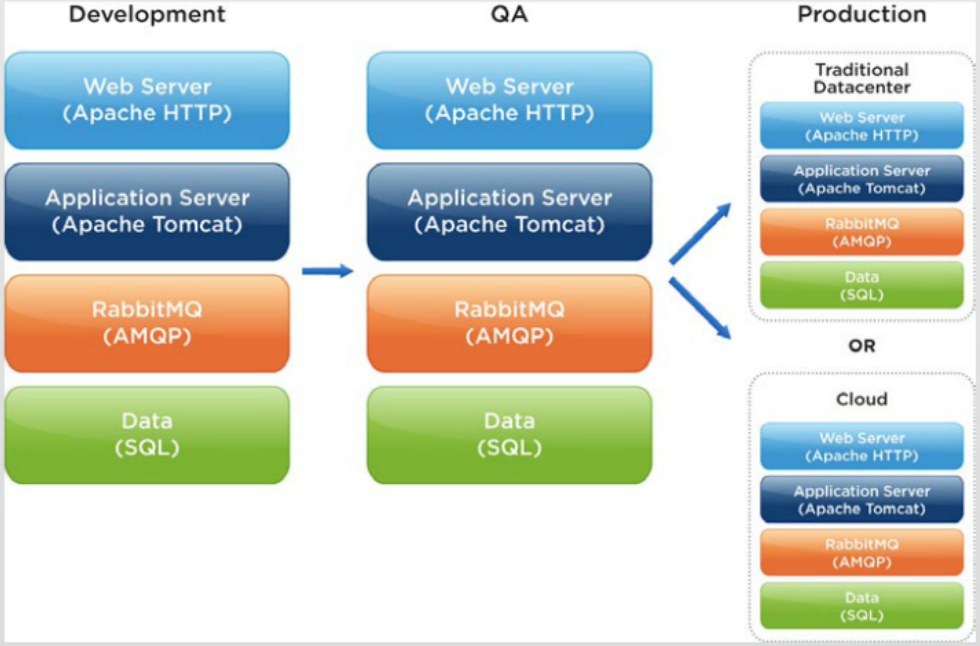




Executes Across

VMware vFabric

- VMware's suite of software for cloud application development
- SQLFire ≈ GemFire (at right)
  - Newer, Standard SQL, Optimized



Development

Web Server (Apache HTTP)

Application Server (Apache Tomcat)

RabbitMQ (AMQP)

Data (SQL)

QA

Web Server (Apache HTTP)

Application Server (Apache Tomcat)

RabbitMQ (AMQP)

Data (SQL)

Production

Traditional Datacenter

Web Server (Apache HTTP)

Application Server (Apache Tomcat)

RabbitMQ (AMQP)

Data (SQL)

OR

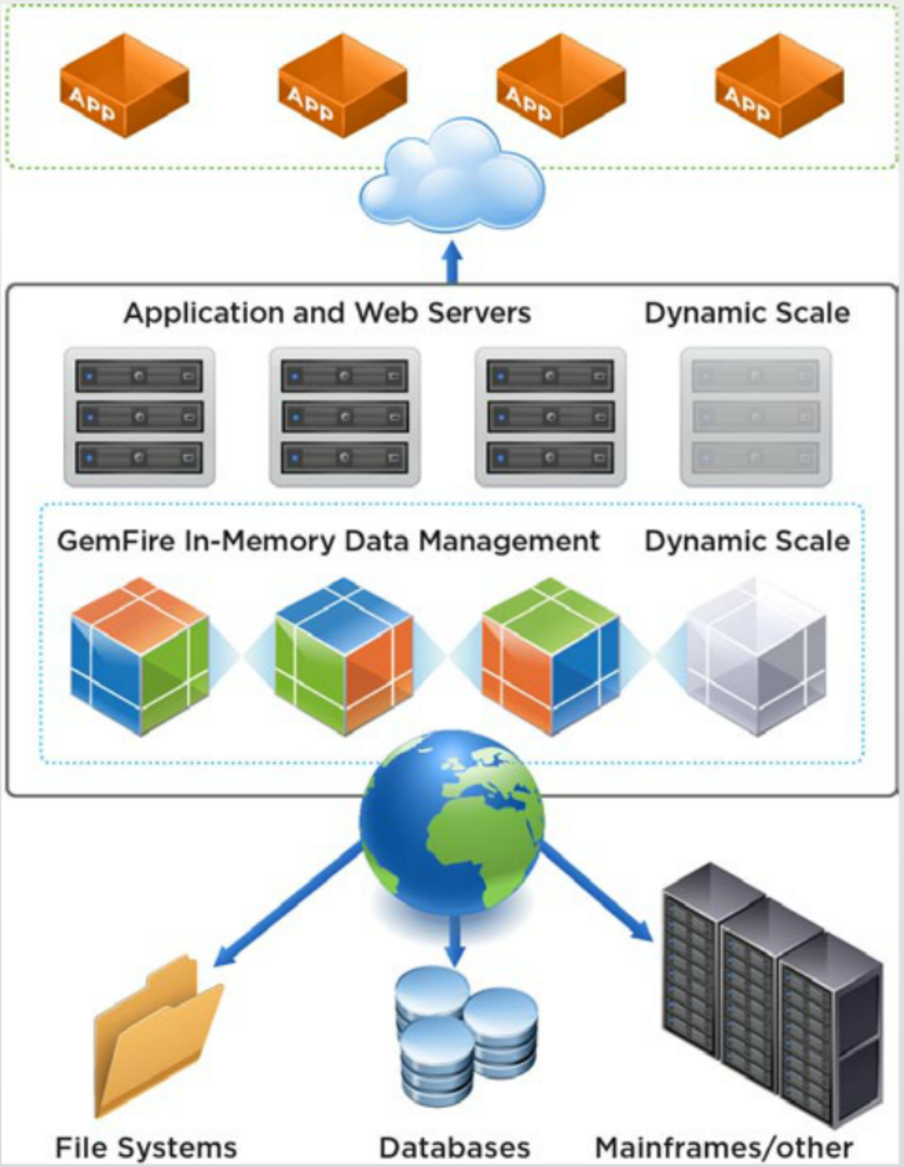
Cloud

Web Server (Apache HTTP)

Application Server (Apache Tomcat)

RabbitMQ (AMQP)

Data (SQL)



Application and Web Servers

Dynamic Scale

GemFire In-Memory Data Management

Dynamic Scale

File Systems

Databases

Mainframes/other

e

nce

age.

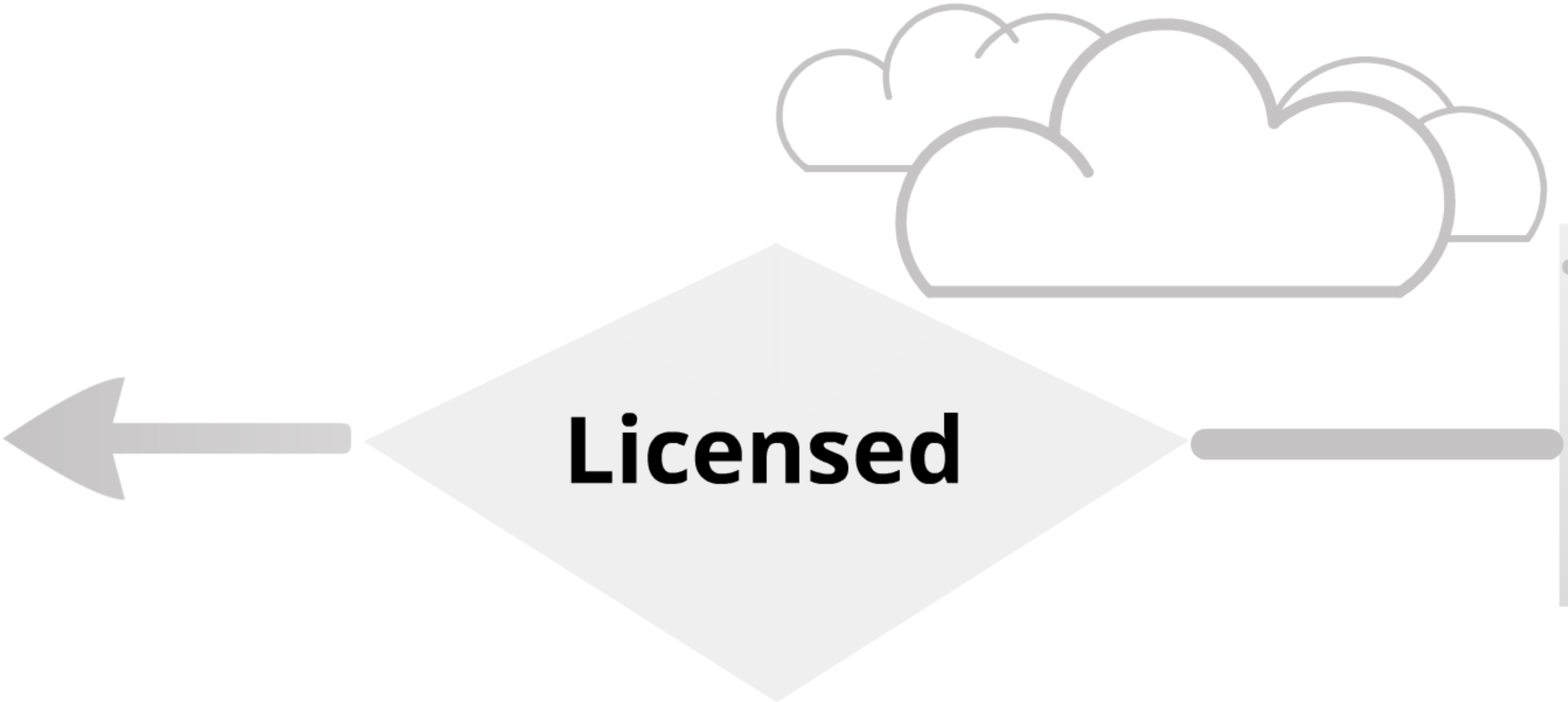
to accommodate

ation tiers: web tier,
nd message servers.
 vFabric Advanced
e-deploy them as 10

Support (USD)

 annual SnS

 annual SnS


Licensed

- VMw
 appl
- SQL
  - N

VMWare

Focuses:

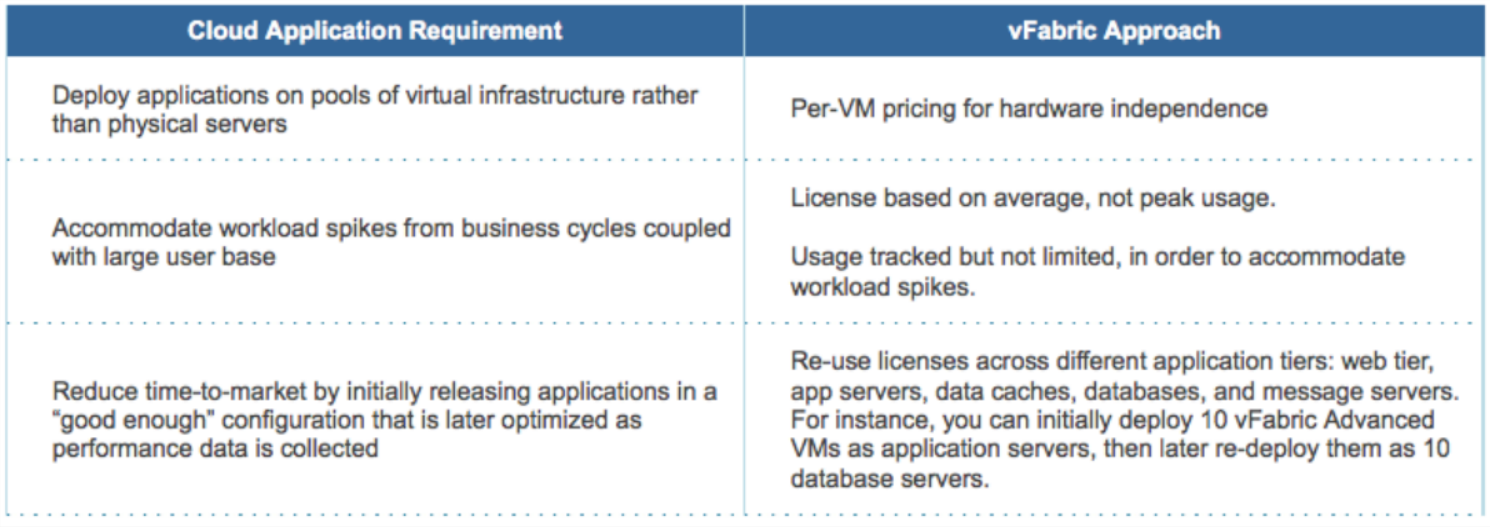
- Virtualization
- Distributed applications

Location: Palo Alto, CA

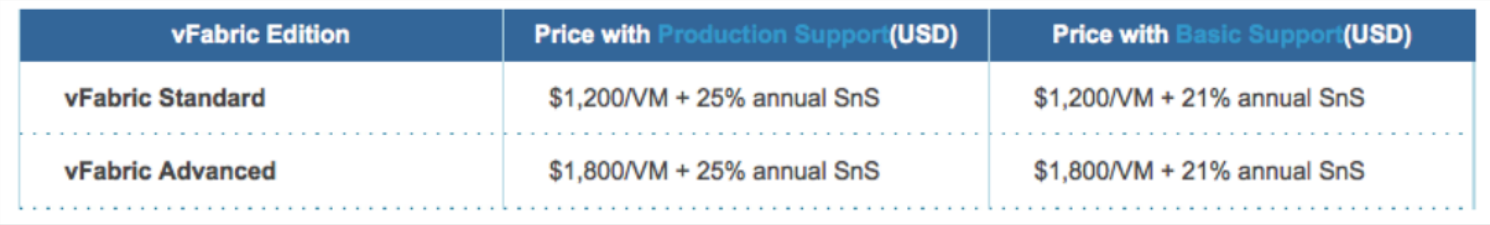
Sales: $3.767096 bil

Profits: $0.723936 bil

vFabric Licensing: per VM, average



Cloud Application Requirement	vFabric Approach
Deploy applications on pools of virtual infrastructure rather than physical servers	Per-VM pricing for hardware independence
Accommodate workload spikes from business cycles coupled with large user base	License based on average, not peak usage. Usage tracked but not limited, in order to accommodate workload spikes.
Reduce time-to-market by initially releasing applications in a "good enough" configuration that is later optimized as performance data is collected	Re-use licenses across different application tiers: web tier, app servers, data caches, databases, and message servers. For instance, you can initially deploy 10 vFabric Advanced VMs as application servers, then later re-deploy them as 10 database servers.



vFabric Edition	Price with Production Support(USD)	Price with Basic Support(USD)
vFabric Standard	$1,200/VM + 25% annual SnS	$1,200/VM + 21% annual SnS
vFabric Advanced	$1,800/VM + 25% annual SnS	$1,800/VM + 21% annual SnS

ensed

Application Server (Apache Tomcat)
Application Server (Apache Tomcat)
RabbitMQ (AMQP)
RabbitMQ (AMQP)
Data (SQL)
Data (SQL)
OR
Cloud
File Systems
Databases
Mainframes/other

Owned By

VMWare

cations
CA
l
l

VMW 107.11 ↓ 1.79 (1.64%)
wikinvest

ttp://finapps.forbes.com/finapps/jsp/finance/compinfo/CIAtAGlance.jsp?tkr=VMW

VMWare

Focuses:

- Virtualization
- Distributed applications

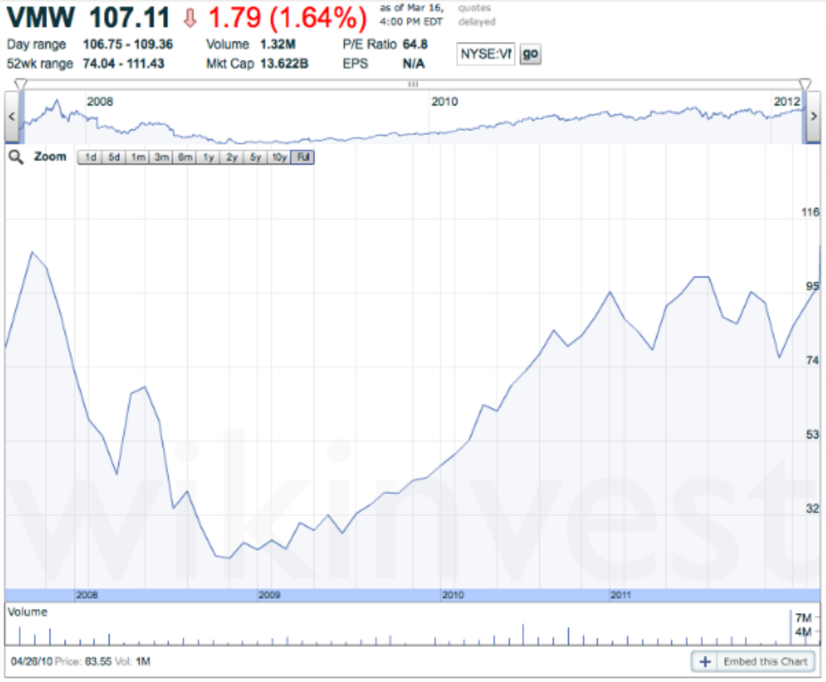
Location: Palo Alto, CA

Sales: $3.767096 bil

Profits: $0.723936 bil

Assets: $8.680808 bil

Employees: 11,000



VMW 107.11 ⇩ 1.79 (1.64%) as of Mar 16, 4:00 PM EDT quotes delayed

Day range 106.75 - 109.36

52wk range 74.04 - 111.43

Volume 1.32M

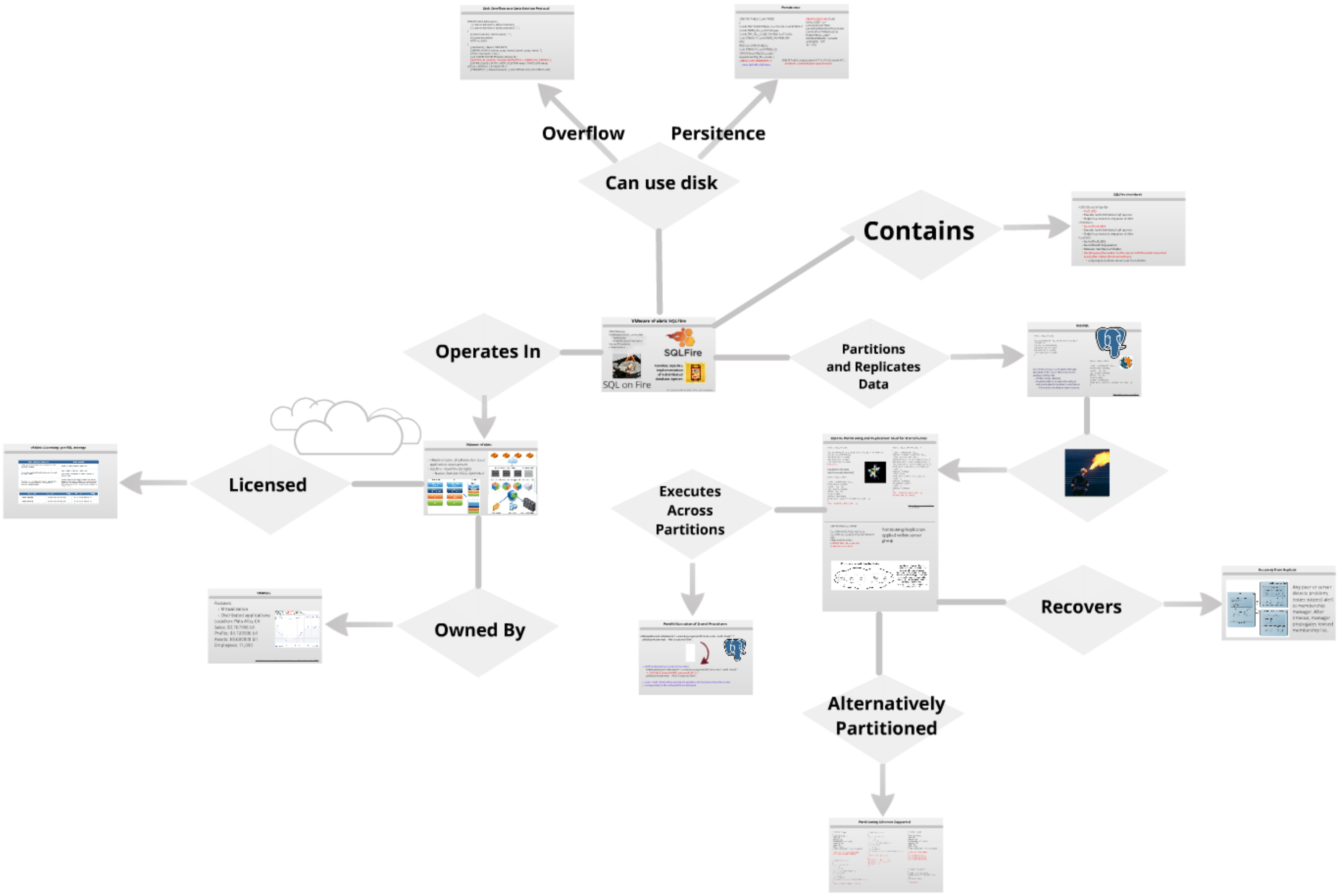
Mkt Cap 13.622B

P/E Ratio 64.8

EPS N/A

04/28/10 Price: 83.55 Vol: 1M

http://finapps.forbes.com/finapps/jsp/finance/compinfo/CIAtAGlance.jsp?tkr=VMW



VMware vFabric SQLFire

SQLFire

SQL on Fire

Can use disk

Overflow

Persitence

Contains

Operates In

Partitions and Replicates Data

Licensed

Owned By

Executes Across Partitions

Recovers

Alternatively Partitioned

Contains

- Data Stores (m
  - Host data
  - Execute loc
  - Single-hop
- Accessors
  - Do not hos
  - Execute loc
  - Single-hop
- Locators
  - Do not hos
  - Do not tou
  - Discover m
  - Clients que load (other
    - Only wa

OLDSQL

```
CREATE TABLE AIRLINES
(
AIRLINE CHAR(2) NOT NULL CONSTRAINT AIRLINES_PK
PRIMARY KEY,
AIRLINE_FULL VARCHAR(24),
```

SQLFire members

- Data Stores (majority)
  - Host data
  - Execute local/distributed sqlf queries
  - Single-hop access to any piece of data
- Accessors
  - Do not host data
  - Execute local/distributed sqlf queries
  - Single-hop access to any piece of data
- Locators
  - Do not host data
  - Do not touch any queries
  - Discover members of cluster
  - Clients query the locator for the server with the least amount of load (other active client connections)
    - Only way to balance server load from clients

Partitions and Replicates Data

OLDSQL

```
CREATE TABLE AIRLINES
(
AIRLINE CHAR(2) NOT NULL CONSTRAINT AIRLINES_PK
PRIMARY KEY,
AIRLINE_FULL VARCHAR(24),
ECONOMY_SEATS INTEGER,
BUSINESS_SEATS INTEGER,
FIRSTCLASS_SEATS INTEGER
)
```

```
CREATE TABLE FLIGHTS
(
FLIGHT_ID CHAR(6) NOT NULL ,
ORIG_AIRPORT CHAR(3),
DEPART_TIME TIME,
DEST_AIRPORT CHAR(3),
ARRIVE_TIME TIME,
MILES INTEGER,
AIRCRAFT VARCHAR(6),
CONSTRAINT FLIGHTS_PK PRIMARY KEY (FLIGHT_ID)
)
```

Our developers are comfortable with SQL. Operating in the cloud, they now need to easily and efficiently:

- Partition large datasets
- Replicate data to increase throughput and guard against (isolated) node failures
  - Remember, working in main memory

http://www.infoq.com/news/2012/01/sqlfire-1-0 easily and efficiently:

- Partition large datasets
- Replicate data to increase throughput and guard against (isolated) node failures
  - Remember, working in main memory

```
ARRIVE_TIME TIME,
MILES INTEGER,
AIRCRAFT VARCHAR(6),
CONSTRAINT FLIGHTS_PK PRIMARY KEY (FLIGHT_ID)
)
```

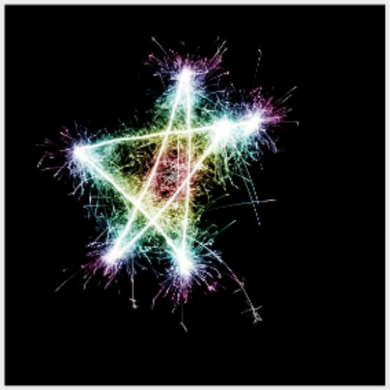
http://www.infoq.com/news/2012/01/sqlfire-1-0

SQLFire Partitioning and Replication: Ideal for Star Schemas

```
CREATE TABLE AIRLINES
(
AIRLINE CHAR(2) NOT NULL CONSTRAINT AIRLINES_PK PRIMARY KEY,
AIRLINE_FULL VARCHAR(24),
ECONOMY_SEATS INTEGER,
BUSINESS_SEATS INTEGER,
FIRSTCLASS_SEATS INTEGER
) REPLICATE;
```

Replication handled synchronously (blocking)

```
CREATE TABLE FLIGHTS
(
FLIGHT_ID CHAR(6) NOT NULL ,
ORIG_AIRPORT CHAR(3),
DEPART_TIME TIME,
DEST_AIRPORT CHAR(3),
ARRIVE_TIME TIME,
MILES INTEGER,
AIRCRAFT VARCHAR(6),
CONSTRAINT FLIGHTS_PK PRIMARY KEY (FLIGHT_ID)
)
PARTITION BY COLUMN (FLIGHT_ID);
```

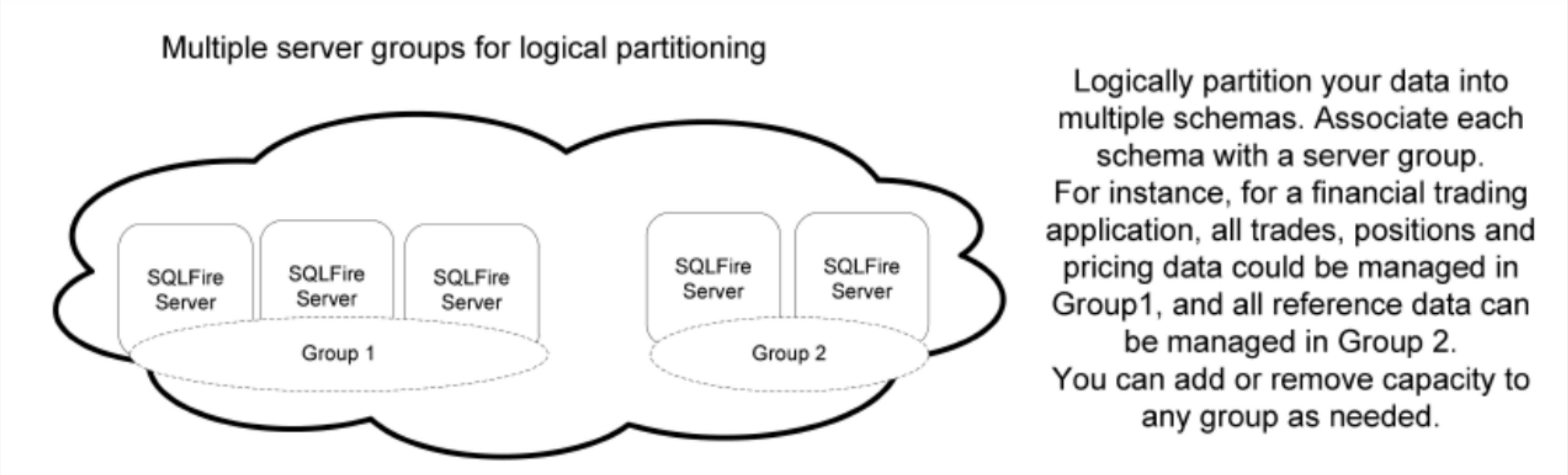


```
CREATE TABLE FLIGHTAVAILABILITY
(
FLIGHT_ID CHAR(6) NOT NULL ,
SEGMENT_NUMBER INTEGER NOT NULL ,
FLIGHT_DATE DATE NOT NULL ,
ECONOMY_SEATS_TAKEN INTEGER DEFAULT 0,
BUSINESS_SEATS_TAKEN INTEGER DEFAULT 0,
FIRSTCLASS_SEATS_TAKEN INTEGER DEFAULT 0,
CONSTRAINT FLIGHTAVAIL_PK PRIMARY KEY (
FLIGHT_ID,
SEGMENT_NUMBER,
FLIGHT_DATE),
CONSTRAINT FLIGHTS_FK2 Foreign Key (
FLIGHT_ID,
SEGMENT_NUMBER)
REFERENCES FLIGHTS (
FLIGHT_ID,
SEGMENT_NUMBER)
)
PARTITION BY COLUMN (FLIGHT_ID)
COLOCATE WITH (FLIGHTS);
```

http://www.infoq.com/news/2012/01/sqlfire-1-0
and documentation

```
CREATE TABLE COUNTRIES
(
COUNTRY VARCHAR(26) NOT NULL,
COUNTRY_ISO_CODE CHAR(2) NOT PRIMARY
KEY,
REGION VARCHAR(26),
) SERVER GROUPS (OrdersDB,
OrdersReplicationGrp)
```

Partitioning/Replicaton applied within server group



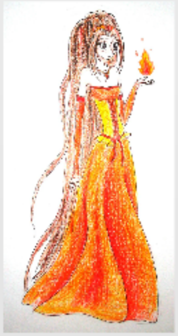
Multiple server groups for logical partitioning

Logically partition your data into multiple schemas. Associate each schema with a server group.
For instance, for a financial trading application, all trades, positions and pricing data could be managed in Group1, and all reference data can be managed in Group 2.
You can add or remove capacity to any group as needed.

Executes Across Partitions

Parallel Execution of Stored Procedures

```
CallableStatement callableStmt = connection.prepareCall("{CALL order_credit_check(?) ");
  callableStmt.setArray(1, <list of customer IDs>);
```



```
// SQLFire data-aware procedure invocation
     CallableStatement callableStmt = connection.prepareCall("{CALL order_credit_check() "
       + "ON TABLE Orders WHERE customerID IN (?)}");
     callableStmt.setArray(1, <list of customer IDs>);

// order_credit_check will be executed in parallel on all members where the orders
// corresponding to the customerIDs are managed
```

ecutes Across rtitions

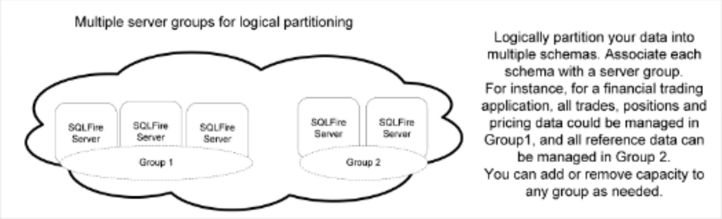
```
ARRIVE_TIME TIME,
MILES INTEGER,
AIRCRAFT VARCHAR(6),
CONSTRAINT FLIGHTS_PK PRIMARY KEY (FLIGHT_ID)
)
PARTITION BY COLUMN (FLIGHT_ID);
```

```
PARTITION BY COLUMN (FLIGHT_ID)
COLOCATE WITH (FLIGHTS);
```

http://www.infoq.com/news/2012/01/sqlfire-1-0
and documentation

```
CREATE TABLE COUNTRIES
(
COUNTRY VARCHAR(26) NOT NULL,
COUNTRY_ISO_CODE CHAR(2) NOT PRIMARY
KEY,
REGION VARCHAR(26),
) SERVER GROUPS (OrdersDB,
OrdersReplicationGrp)
```

Partitioning/Replicaton applied within server group



Multiple server groups for logical partitioning

Logically partition your data into multiple schemas. Associate each schema with a server group.
For instance, for a financial trading application, all trades, positions and pricing data could be managed in Group1, and all reference data can be managed in Group 2.
You can add or remove capacity to any group as needed.

l Execution of Stored Procedures

```
= connection.prepareCall("{CALL order_credit_check(?) ");
f customer IDs>);
```

re invocation

```
Stmt = connection.prepareCall("{CALL order_credit_check() "
RE customerID IN (?)}");
st of customer IDs>);
```

xecuted in parallel on all members where the orders
nerIDs are managed

Alternatively

Alternatively Partitioned

Partitioning Schemes Supported

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY COLUMN ( CustomerName )
SERVER GROUPS ( OrdersDBServers);
```

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY LIST ( Status )
```

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY RANGE ( Priority )
(
```

Partitioning Schemes Supported

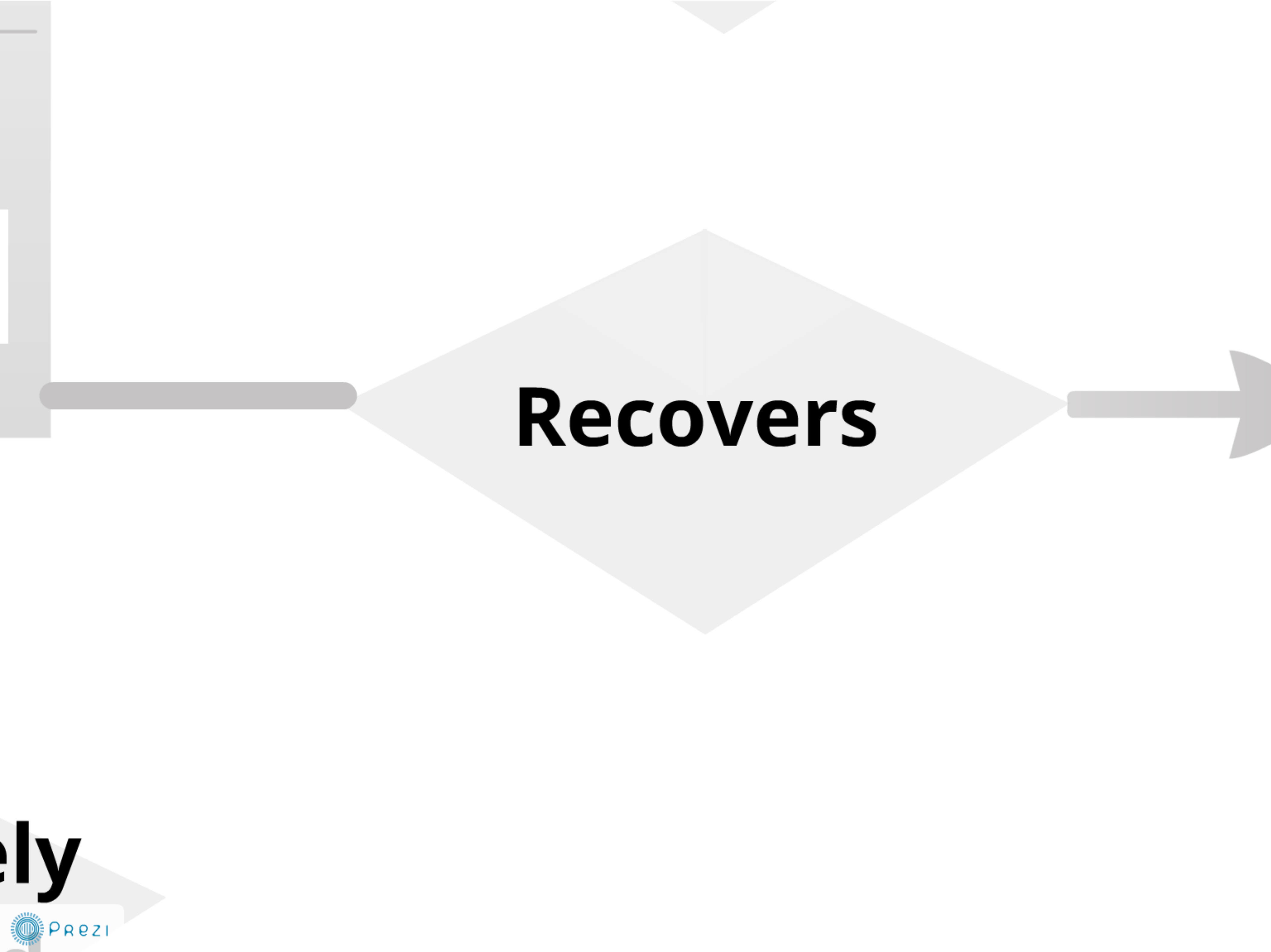
```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY COLUMN ( CustomerName )
SERVER GROUPS ( OrdersDBServers);
```

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY ( MONTH( OrderDate ) );
```

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY LIST ( Status )
(
VALUES ( 'pending', 'returned' ),
VALUES ( 'shipped', 'received' ),
VALUES ( 'hold' )
);
```

```
CREATE TABLE Orders
(
OrderId INT NOT NULL,
ItemId INT,
NumItems INT,
CustomerName VARCHAR(100),
OrderDate DATE,
Priority INT,
Status CHAR(10),
CONSTRAINT Pk_Orders PRIMARY KEY (OrderId)
)
PARTITION BY RANGE ( Priority )
(
VALUES BETWEEN 1 AND 11,
VALUES BETWEEN 11 AND 31,
VALUES BETWEEN 31 AND 50);
```

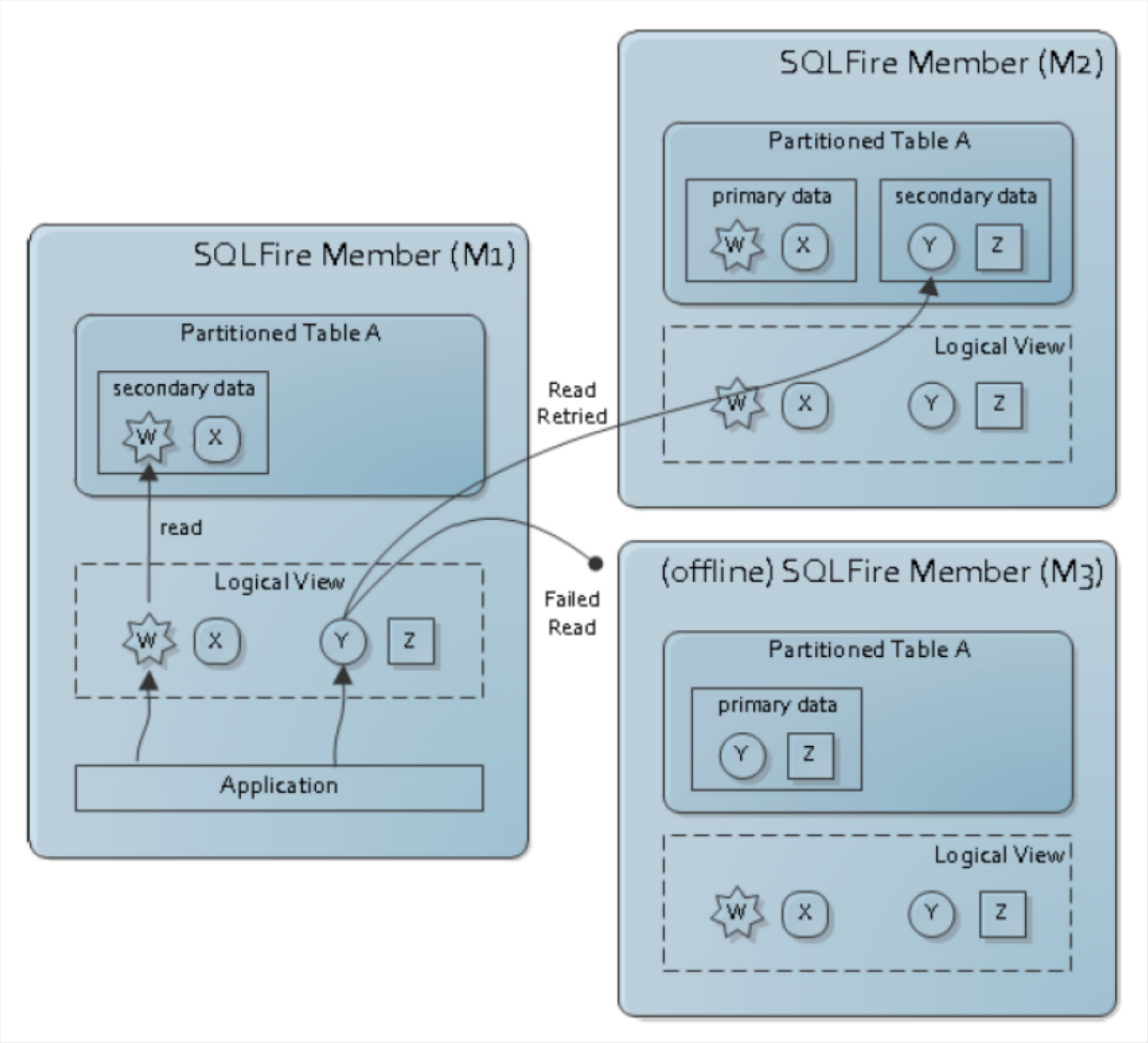
```
CREATE TABLE COUNTRIES
(
COUNTRY VARCHAR(26) NOT NULL,
COUNTRY_ISO_CODE CHAR(2) NOT PRIMARY
KEY,
REGION VARCHAR(26),
)
REDUNDANCY 1
```



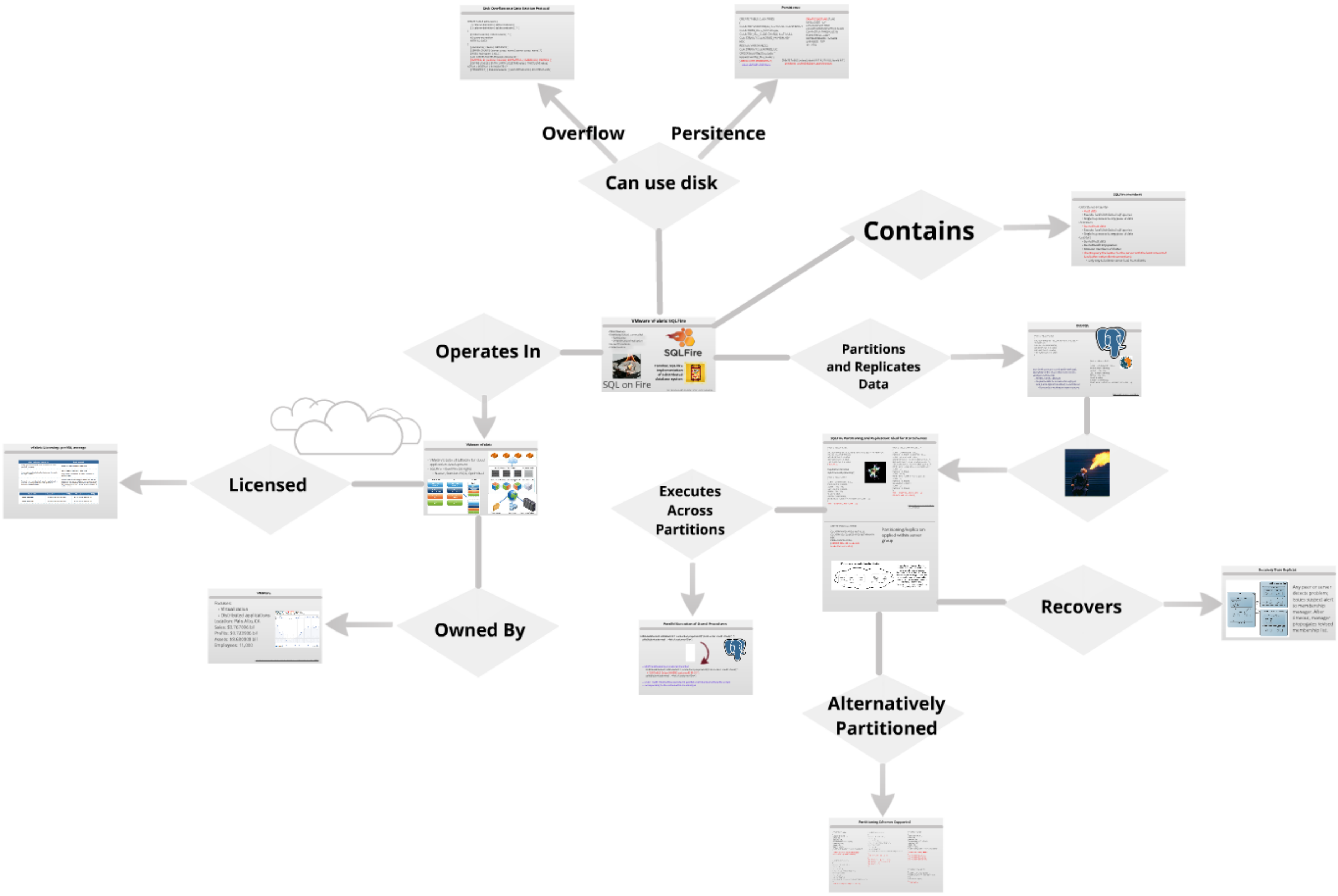
Recovers

ely

Recovery from Replicas



Any peer or server detects problem; issues suspect alert to membership manager. After timeout, manager propogates revised membership list.



Can use disk

Overflow

Persitence

Contains

Operates In

VMware vFabric SQLFire

SQLFire

SQL on Fire

Partitions and Replicates Data

Licensed

Owned By

Executes Across Partitions

Recovers

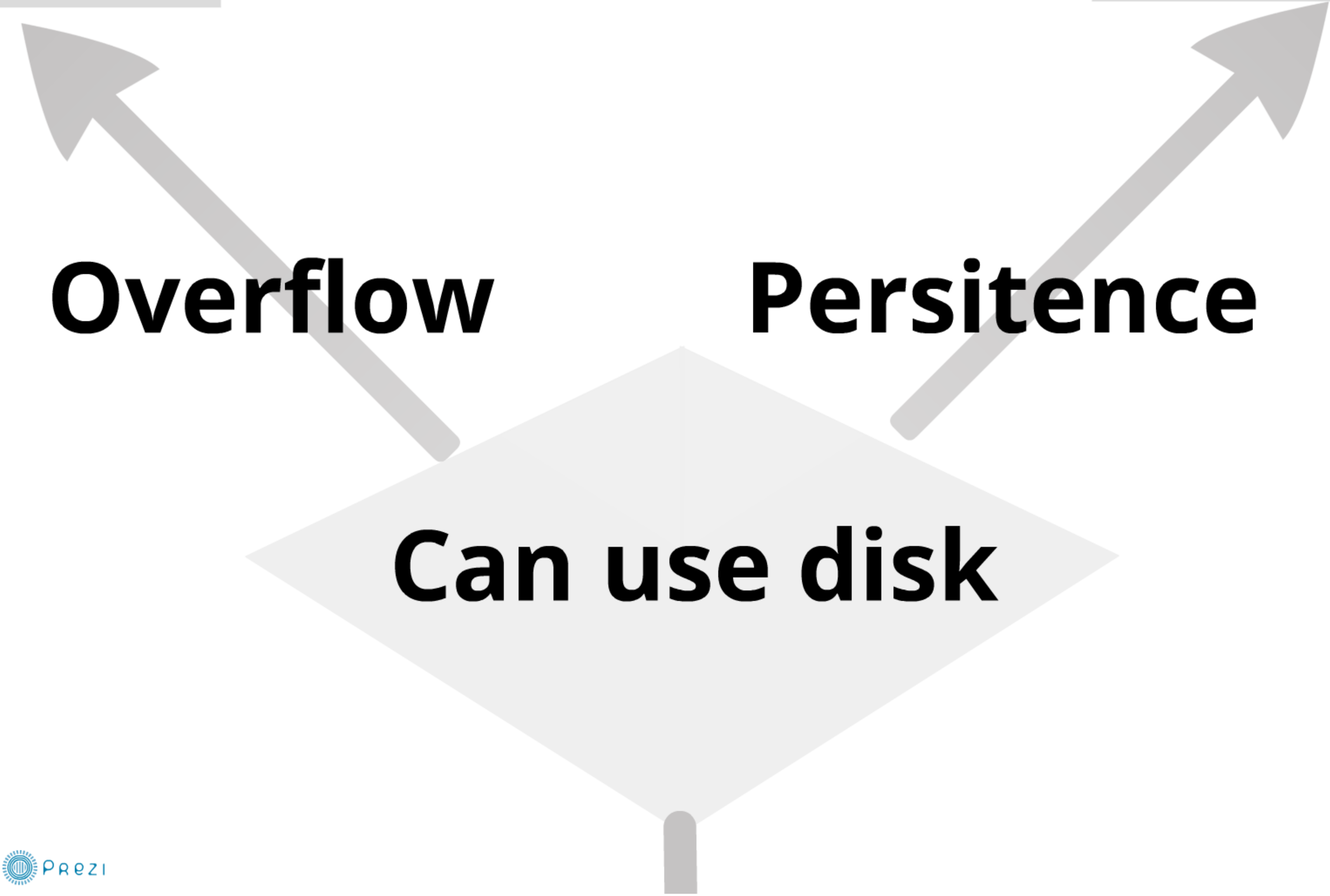
Any peer or server detects problem; issues suspect alert to membership manager. After timeout, manager propagates revised membership list.

Alternatively Partitioned

```
LOW | DESTROY }]
ETOLIVE value}

SYNCHRONOUS ]
```

```
) REPLICATE PERSISTENT;
-- uses default diskstore
```



Overflow

Persitence

Can use disk

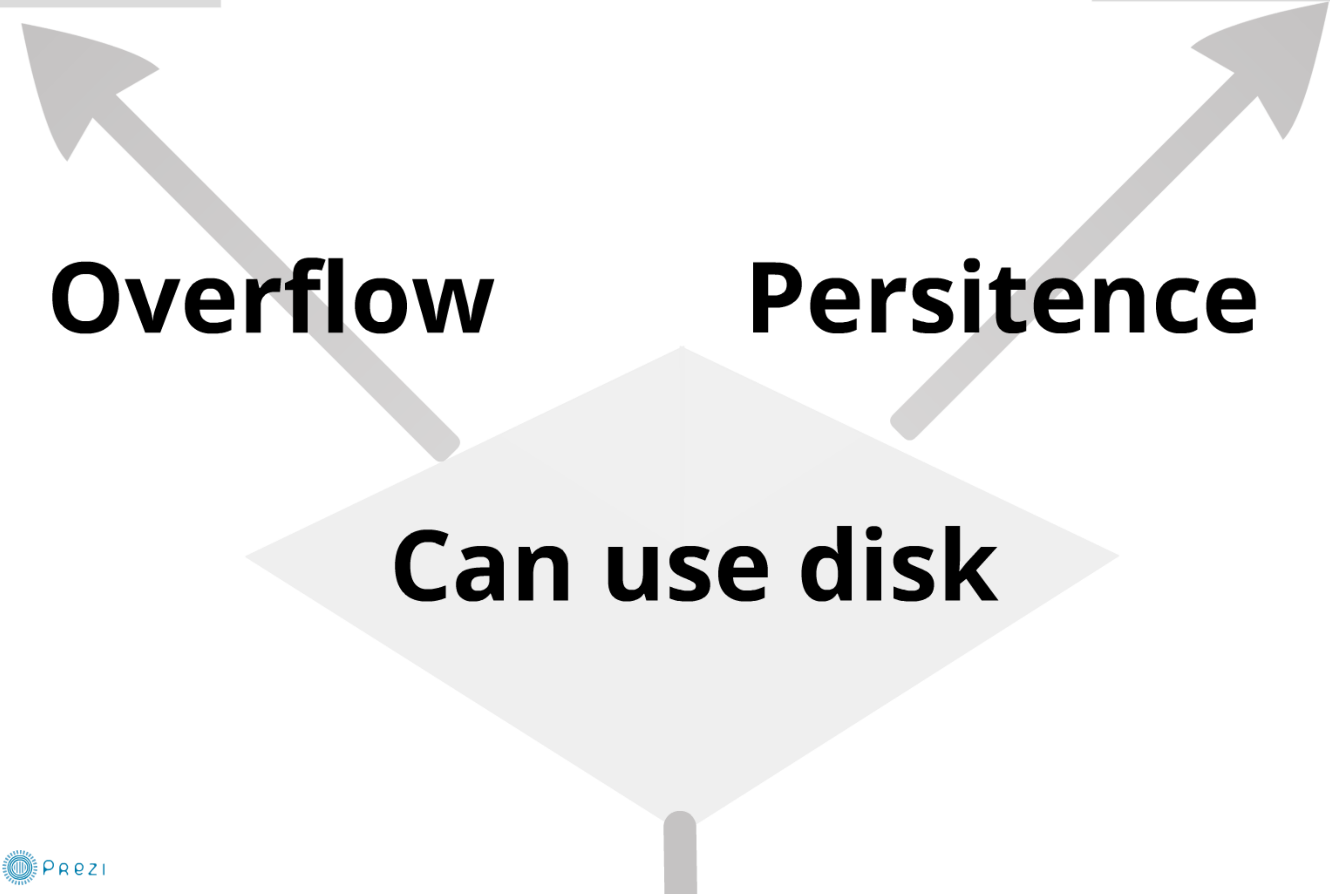
Disk Overflow as a Data Eviction Protocol

```
CREATE TABLE table-name {
     ( { column-definition | table-constraint }
    [, { column-definition | table-constraint } ] * )
|
    [( column-name [, column-name ] * ) ]
    AS query-expression
    WITH NO DATA
}
    [ partitioning_clause | REPLICATE ]
    [ SERVER GROUPS ( server_group_name [, server_group_name ]*)]
    [ HUB ( 'hub-name' | ALL ) ]
    [ ASYNCEVENTLISTENER (async-listener-id) ]
    [ EVICTION BY {eviction_criterion} EVICTACTION { OVERFLOW | DESTROY }]
    [ EXPIRE { TABLE | ENTRY } WITH { IDLETIME value | TIMETOLIVE value}
ACTION { DESTROY | INVALIDATE } ]*
    [ PERSISTENT ] [ 'disk-store-name' ] [ ASYNCHRONOUS | SYNCHRONOUS ]
```

```
LOW | DESTROY }]
ETOLIVE value}

 SYNCHRONOUS ]
```

```
) REPLICATE PERSISTENT;
-- uses default diskstore
```



Overflow

Persitence

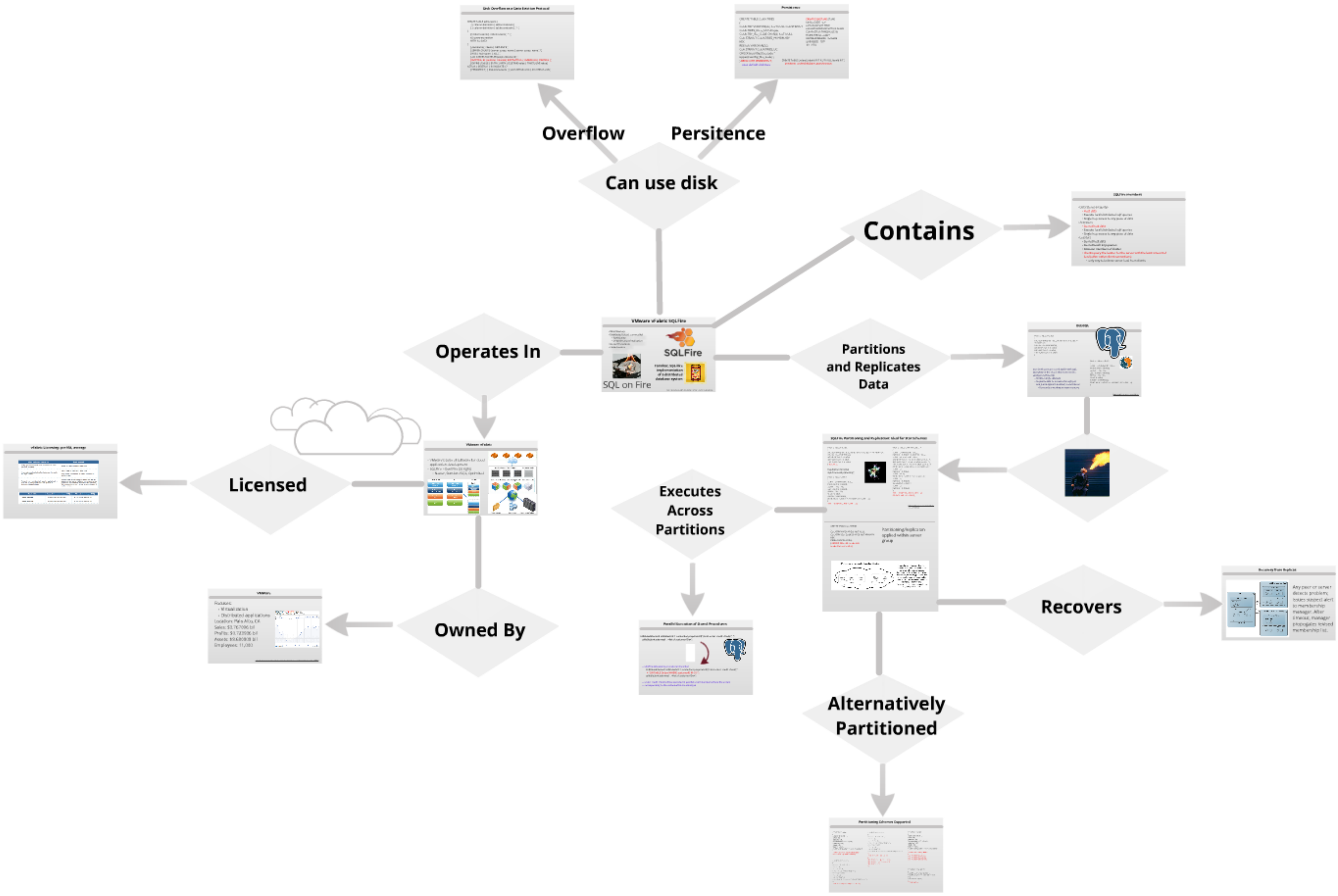
Can use disk

Persistence

```
CREATE TABLE COUNTRIES
(
COUNTRY VARCHAR(26) NOT NULL CONSTRAINT
COUNTRIES_UNQ_NM Unique,
COUNTRY_ISO_CODE CHAR(2) NOT NULL
CONSTRAINT COUNTRIES_PK PRIMARY
KEY,
REGION VARCHAR(26),
CONSTRAINT COUNTRIES_UC
CHECK (country_ISO_code =
upper(country_ISO_code) )
) REPLICATE PERSISTENT;
-- uses default diskstore
```

```
CREATE DISKSTORE STORE1
MAXLOGSIZE 1024
AUTOCOMPACT TRUE
ALLOWFORCECOMPACTION FALSE
COMPACTIONTHRESHOLD 80
TIMEINTERVAL 223344
WRITEBUFFERSIZE 19292393
QUEUESIZE 17374
'dir1'(456)
```

```
CREATE TABLE Orders(OrderId INT NOT NULL,ItemId INT )
    persistent 'OrdersDiskStore' asynchronous
```



Can use disk

Overflow

Persitence

Contains

VMware vFabric SQLFire

SQLFire

SQL on Fire

Operates In

Partitions and Replicates Data

Licensed

Executes Across Partitions

Owned By

Recovers

Alternatively Partitioned

VMware vFabric SQLFire

- Main Memory
- Distributed (cloud, commodity)
    - Partitioning
    - (A)synchronous Replication
- Stored Procedures
- Closed Source

SQLFire

SQL on Fire

Familiar, SQL-like implementation of a distributed database system

Unless otherwise noted, info and examples are from VMware's documentation.