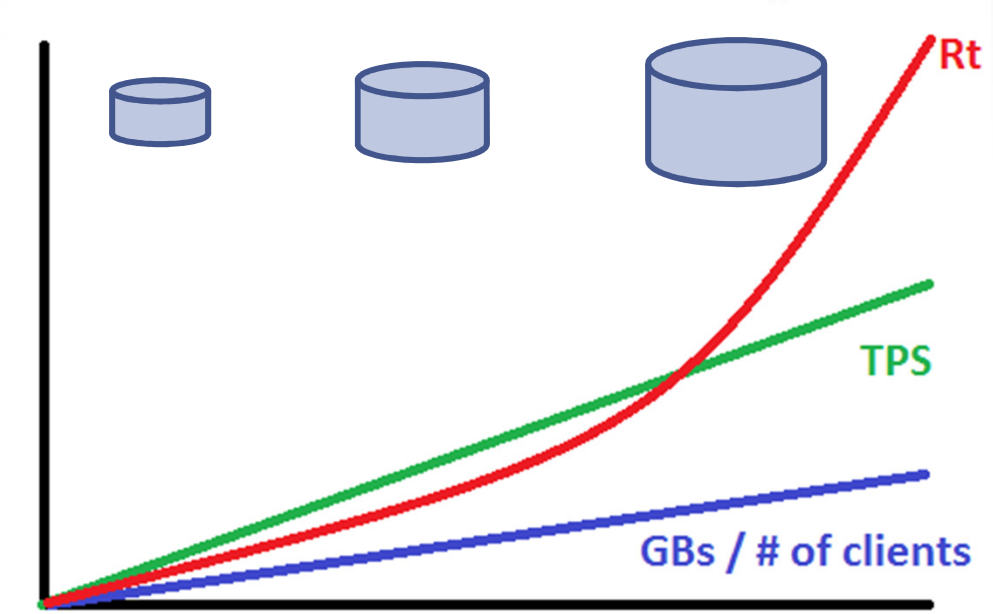
# Data Sharding in OLTP Systems

# Who is this guy with heavy accent?

- Director of Development at Actsoft.
  - I'm responsible for DB Design and Development of OLTP system handling ~2500 TPS during peak time
- 9+ years of experience with Microsoft SQL Server
- MCITP
  - SQL Server Database Developer 2005, 2008
  - SQL Server Database Administrator 2008
- MCPD
  - Enterprise Application Developer
- Blog: http://aboutsqlserver.com
  - Session will be available for download
- Email: dmitri@aboutsqlserver.com

# Let's define the problem




- **What options do we have when system is about to outgrow the hardware?**
  - **Upgrade**
    - It's a shame that CEO and CFO do not share our passion of buying new hardware
  - **Optimize, optimize, optimize!**
    - Biggest bang for the buck but even optimization has it's own limit
  - **Reduce the server workload**

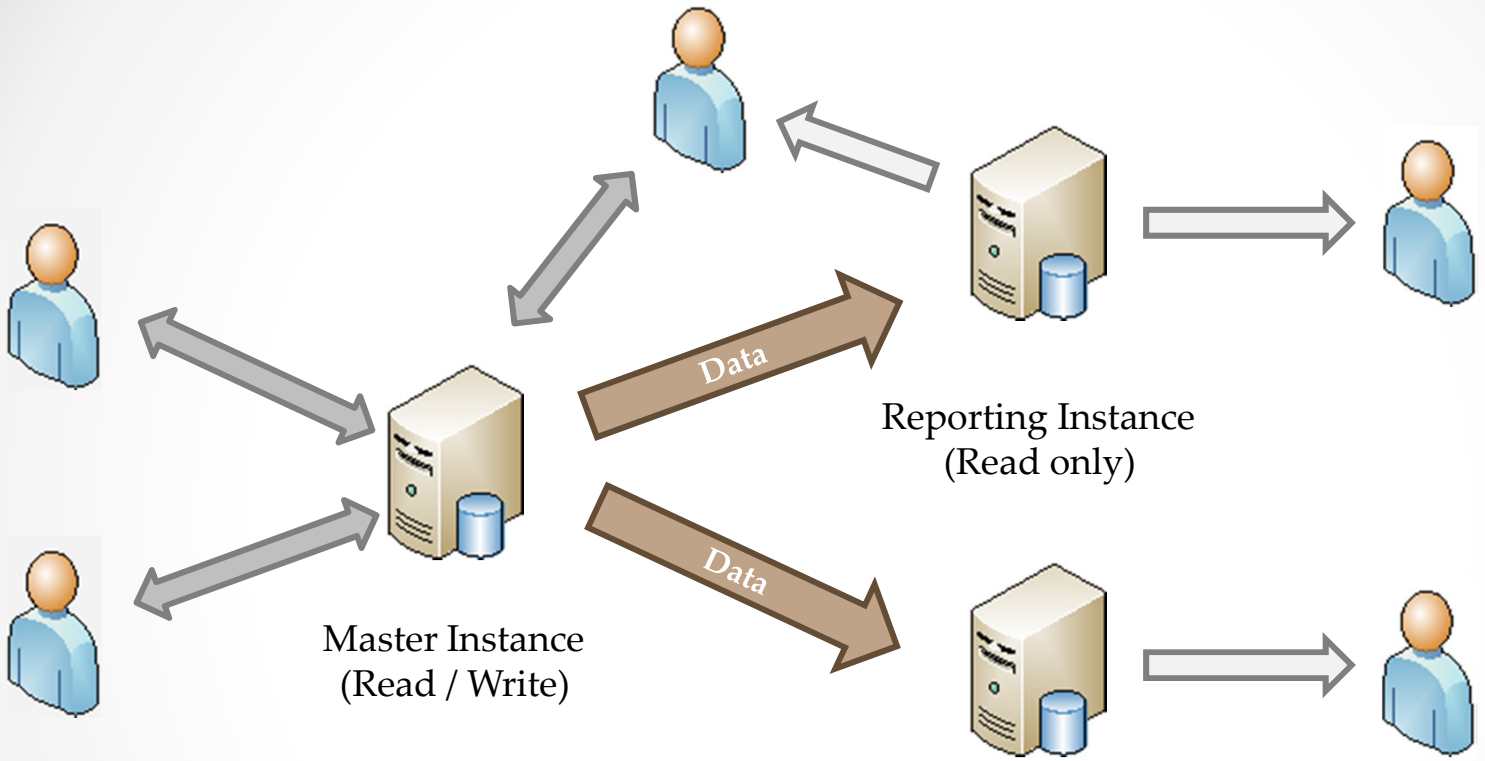
# Agenda

- **We're on BI track but this session does not have anything to do with BI**
- Discuss various techniques and architectural approaches that reduces the server workload
  - Canonical technique - Building separate reporting server(s)
  - Data Sharding by the book (Share Nothing)
  - Sharding of operational and archive data with vertical partitioning
- *This is boring PowerPoint based session*

# Building Reporting Server(s)

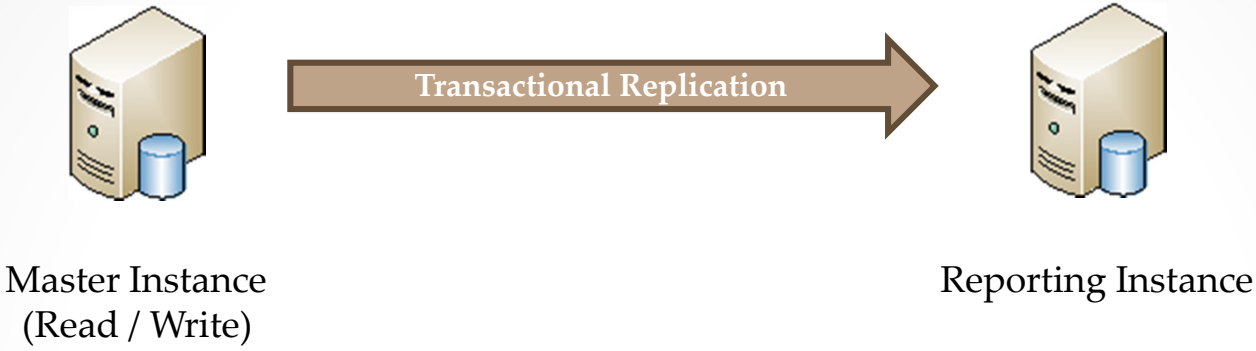
What's available out of the box?

# Reporting Servers




- Operational activity done against Master Instance
- Reporting activity done against Reporting instances
- In some cases data in Reporting instances could be transformed
- Can be implemented as part of High Availability Technologies
  - Extra Licensing cost

# Using Replication

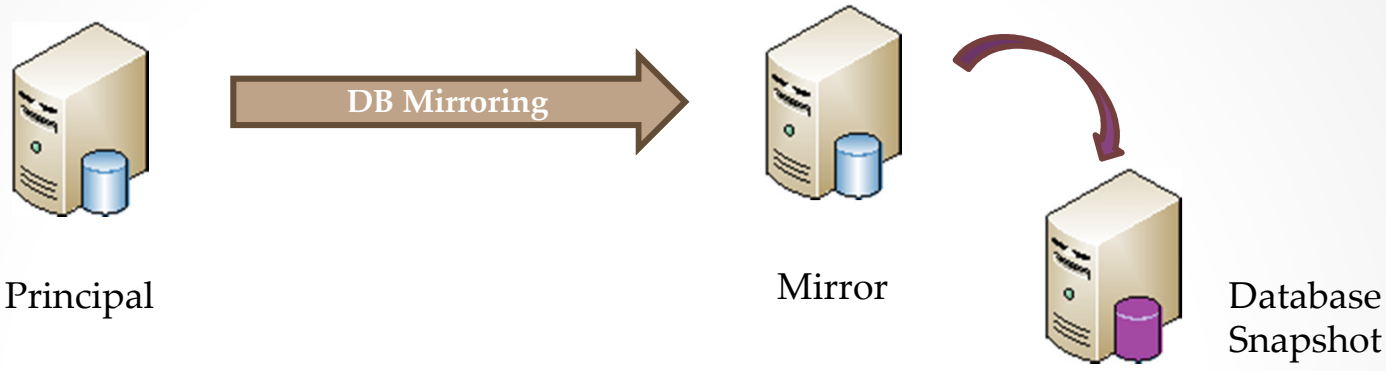



- Almost Real Time (latency is in seconds)
- Can be set up on subset of tables/data
- Can be bi-directional
- Introduces additional overhead (maintenance)

# Using Log Shipping

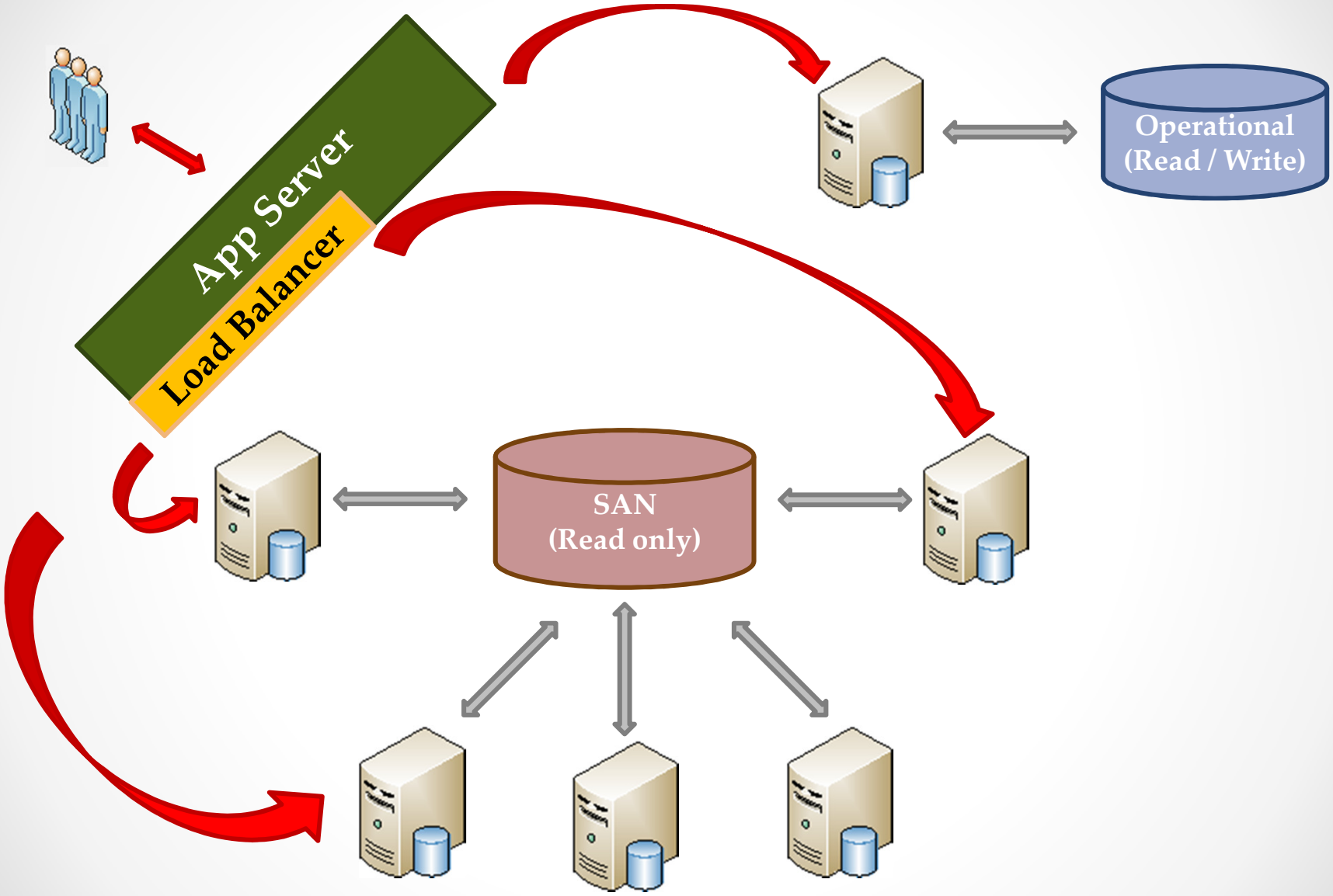

- Latency depends on configuration
- Entire database is replicated
- Read only access only
- Simpler setup and maintenance than with replication

# Using DB Mirroring




- Read-only access to database snapshot from the Mirror (Enterprise Edition only)
- Latency depends on Snapshot recreation time
- Development challenges due Snapshot maintenance
  - Failover support
  - Client connections during snapshot recreation

# Scalable Shared Database



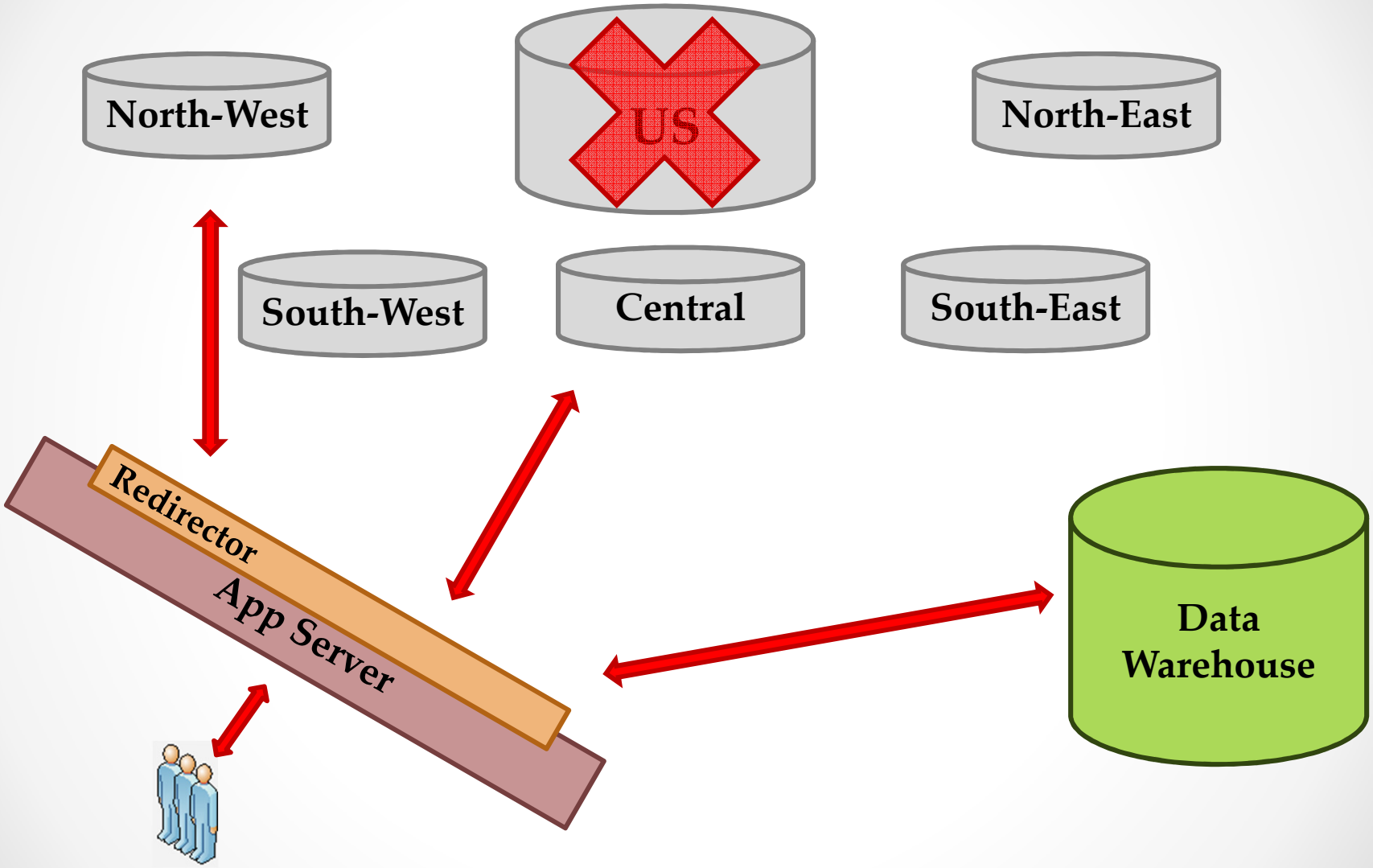
# Conclusions

- All methods but replication have latency updating reporting instances
  - If latency is not acceptable application server needs to query/merge data from the multiple sources
- Some methods introduce interruption of the service for reporting activity
- With Log Shipping and Mirroring reporting database is exact replica of production
- With other methods data in reporting instances can be transformed

# Data Sharding by the Book

Share Nothing Architecture

# Architecture at a glance



# Data Sharding in general..

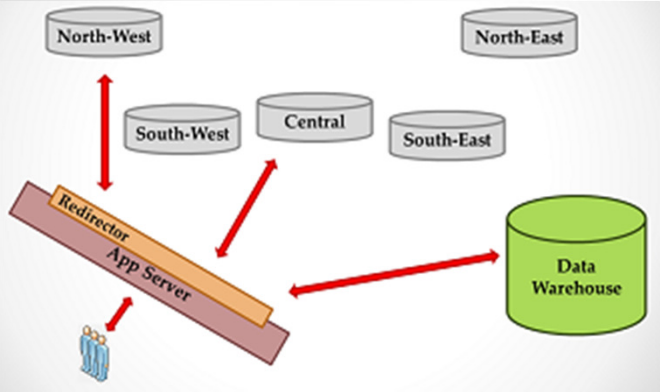
**Pros**

- Lower infrastructure cost
  - Scale-out is typically cheaper than Scale-up
- System is partially available if some shards are down
- Almost unlimited scalability when properly architected
- Better "*cloudability*"
  - Azure – restricted DB size
  - AWS – I/O system limitations

**Cons**

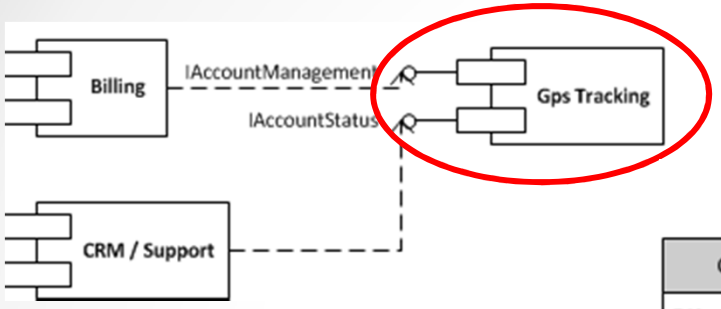
- Higher operational cost
  - Need to support multiple servers
- DR/HA solutions need to be implemented on the shard level
- Higher development cost

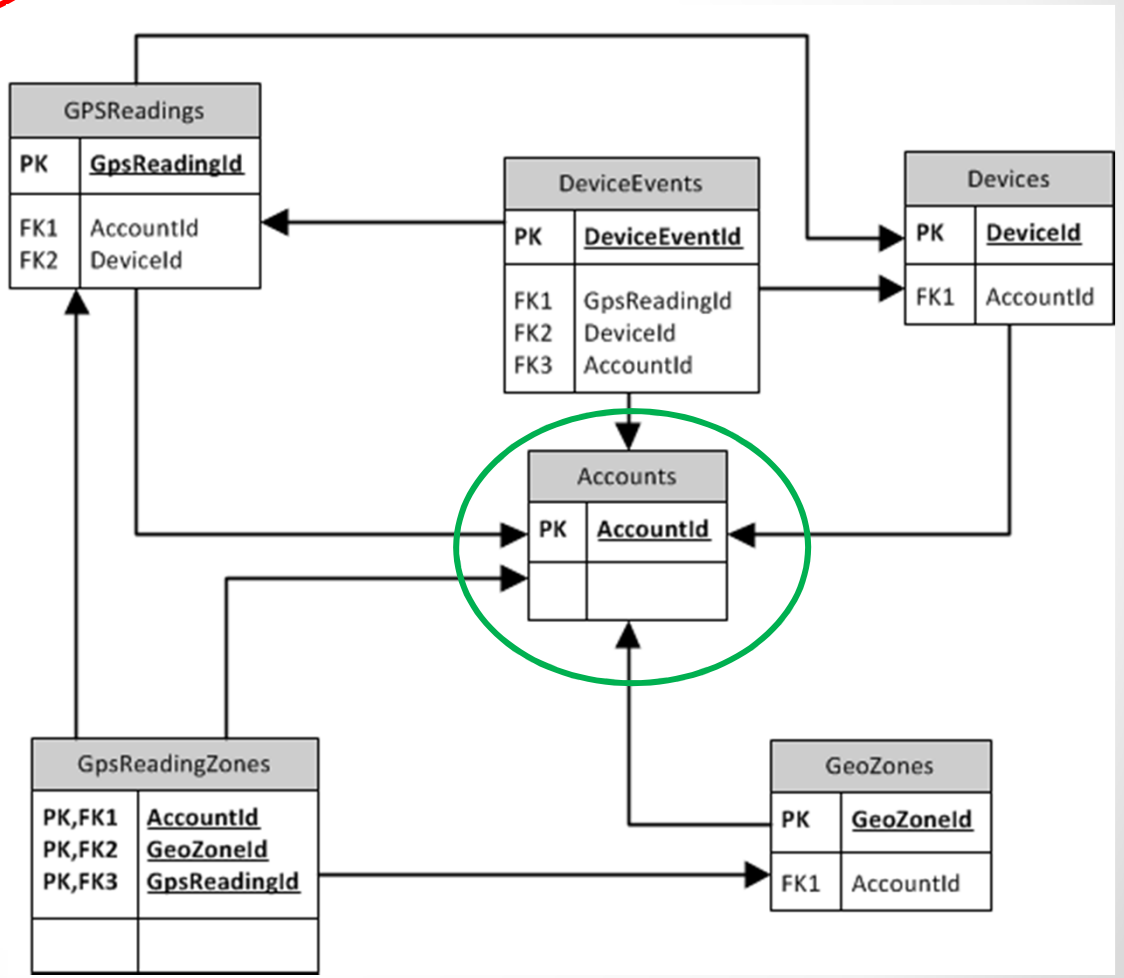
# “Share Nothing” in particular..




- Schema is identical for each Shard
- Good when:
  - System has very small dependencies between the shards
  - Business logic can be implemented within the shard
  - Cross-shard unions and joins are minimal
    - There is the Data Warehouse for reporting/analysis purposes
  - There are no issues with legacy code

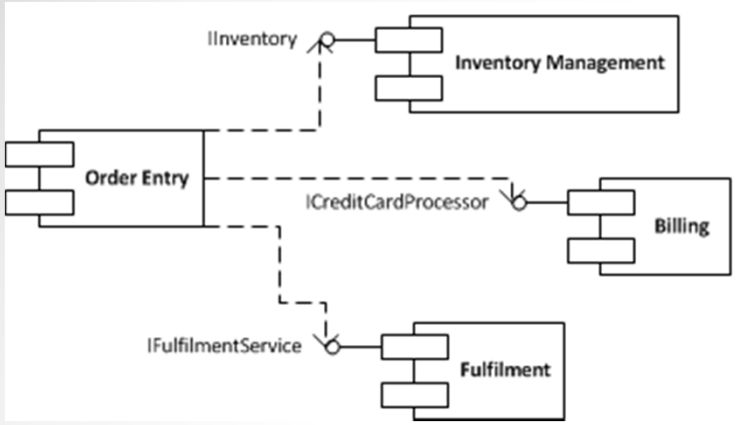
# Example 1 – GPS Tracking

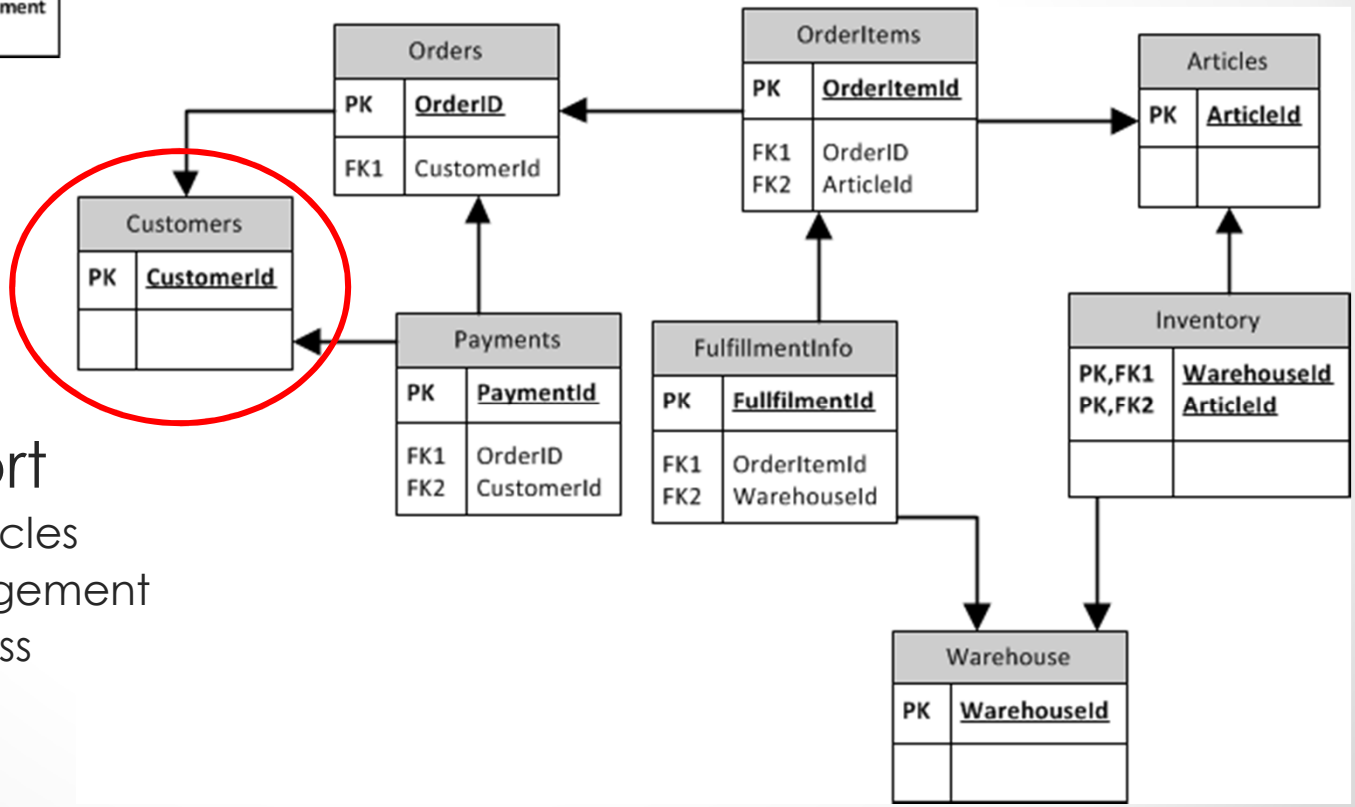



- Everything depend on Account
- We can shard the system by sets of accounts

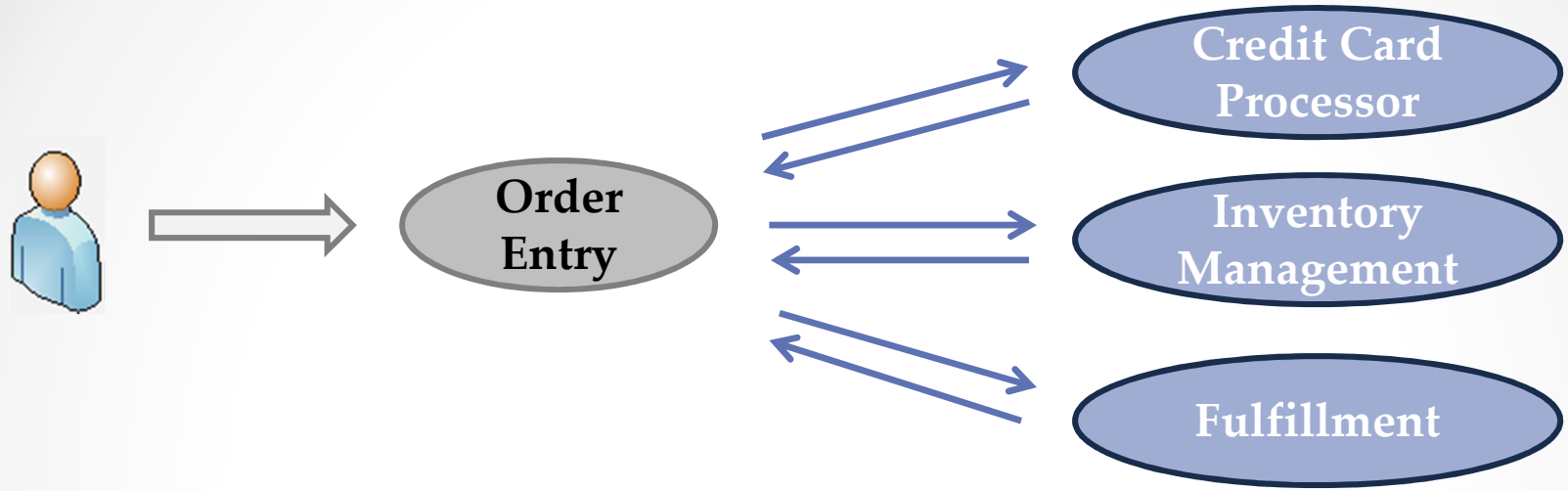
# Example 2 – Online retailer

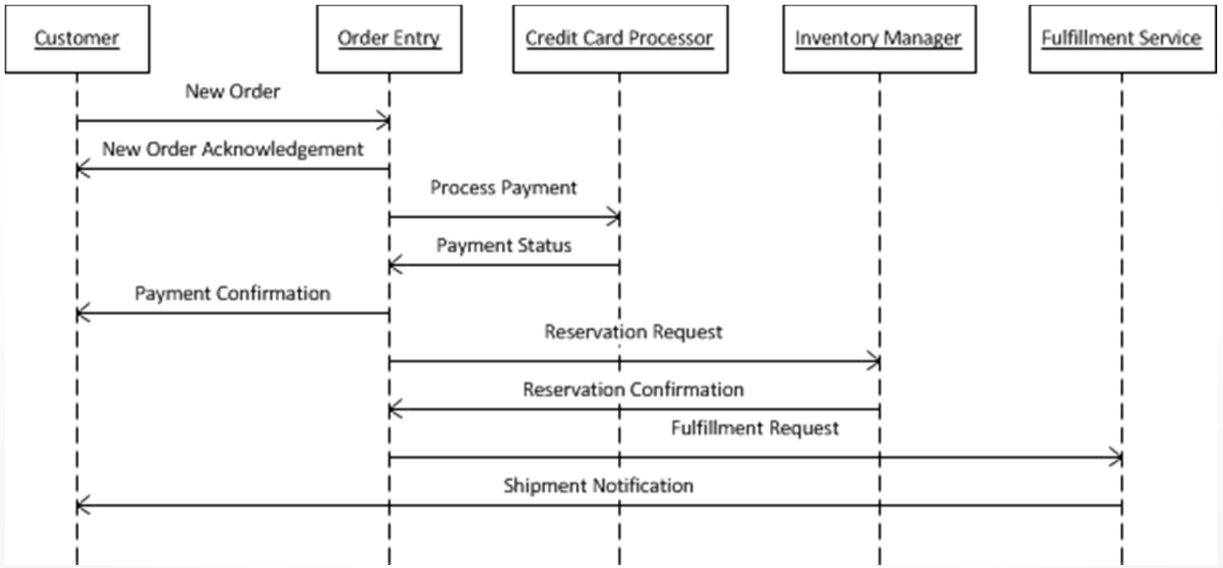



- How to support
  - Shared list of Articles
  - Inventory management
  - Fulfillment process
  - Customer Billing

# Online Retailer - SOA



# Shard Criteria

- Beware of uneven data distribution
  - Shard by region – what if East Cost does 90% of the sales?
  - Shard by ID range (1..100,000; 100,001..200,000) – what if one of the customers produces 100th times more data than others?
- One possible way:
  - SHARD # = ID  mod  TOTAL_#_OF_SHARDS

# Conclusion

- Share-Nothing approach is good when
  - **Shards are independent from each other**
  - There are very small number of catalog entities
  - Business logic can be either implemented within the shard or system designed as SOA
  - **Legacy code can be supported**
  - Operational cost of supporting multiple servers is acceptable

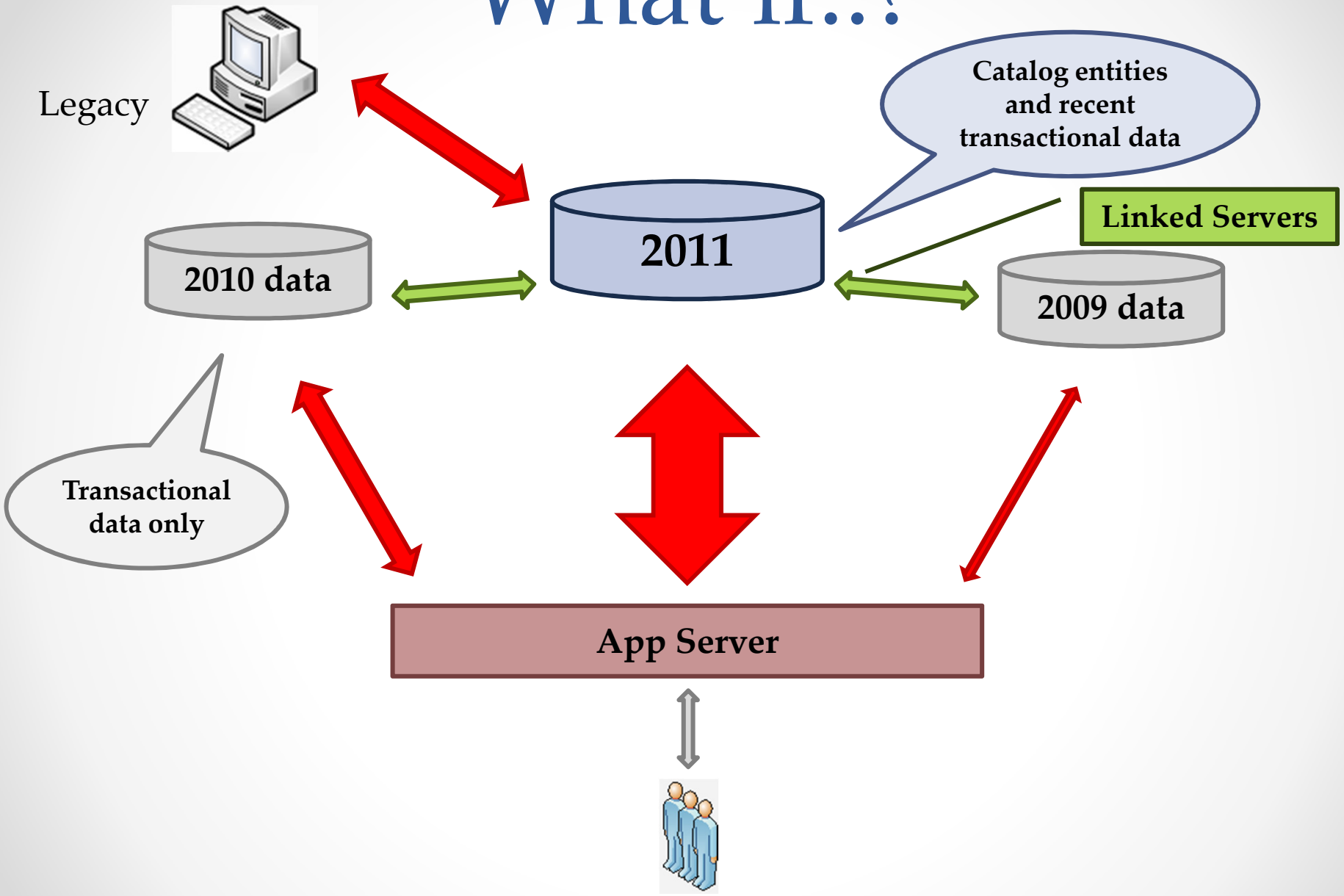
# Vertical Partitioning by Operational Periods

When "Share Nothing" Approach is not the best choice

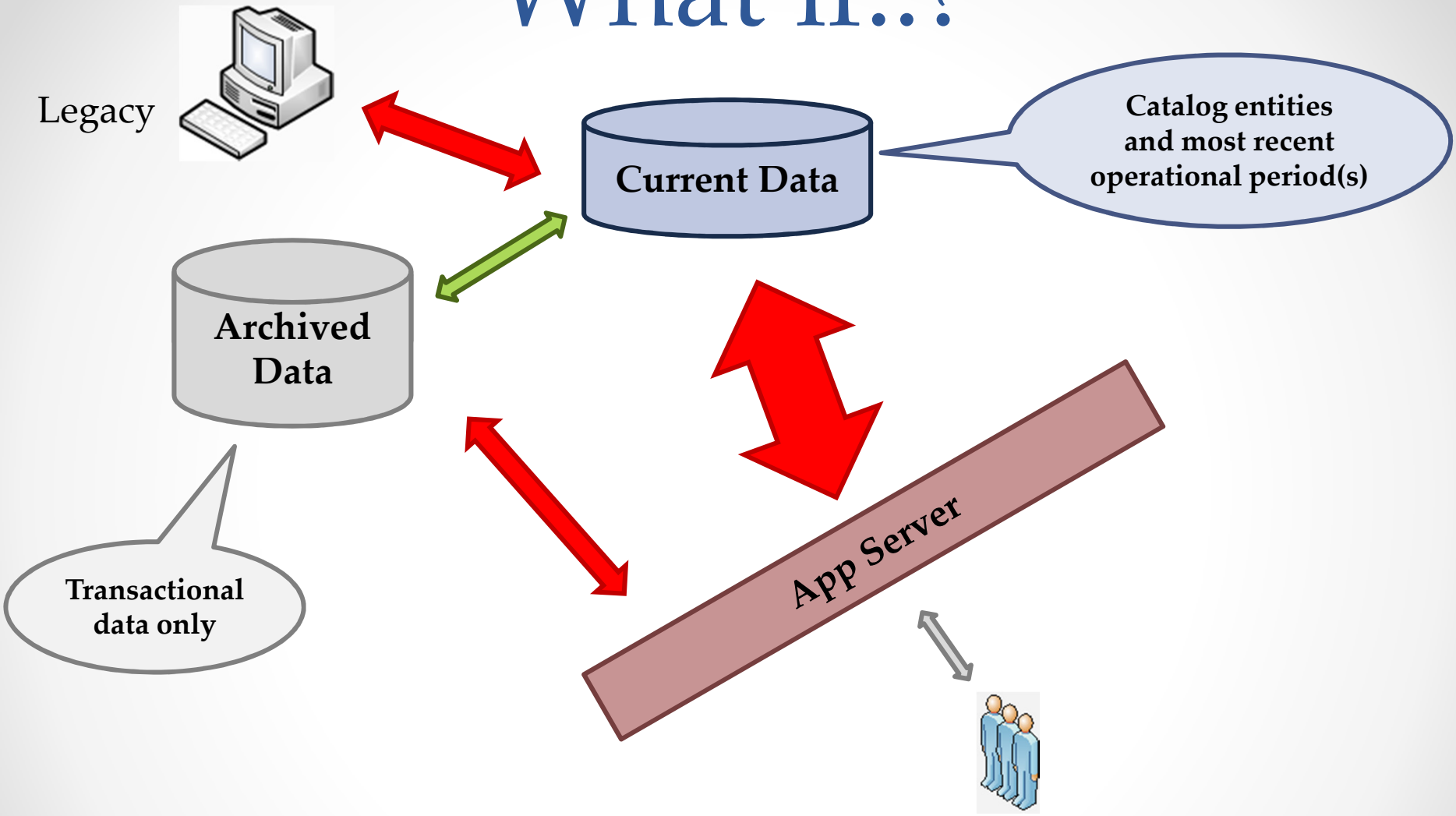
# Do we need old data?

- In most OLTP systems users rarely work with old data
  - Online Retailer
    - How often customers look at the old closed orders?
    - How often business needs to access raw data that does not belong to current or previous tax period?
  - GPS Tracking
    - Operational activity – data from today
    - Analysis of the old data (Breadcrumb trails) limited to the 2 last payroll/billing periods

# What if..?


Legacy
Catalog entities and recent transactional data
2011
Linked Servers
2010 data
2009 data
Transactional data only
App Server

# What if..?


Legacy
Current Data
Catalog entities and most recent operational period(s)
Archived Data
Transactional data only
App Server

# Comparing the methods

## Vertical Partitioning

- Data dependencies are much less important
- Scalability limited by the server hosted current operational data
- "*Cloudability*" is limited by operational server
- Requires operational server to be available all the time
- Legacy is easier to support
- Lower development cost

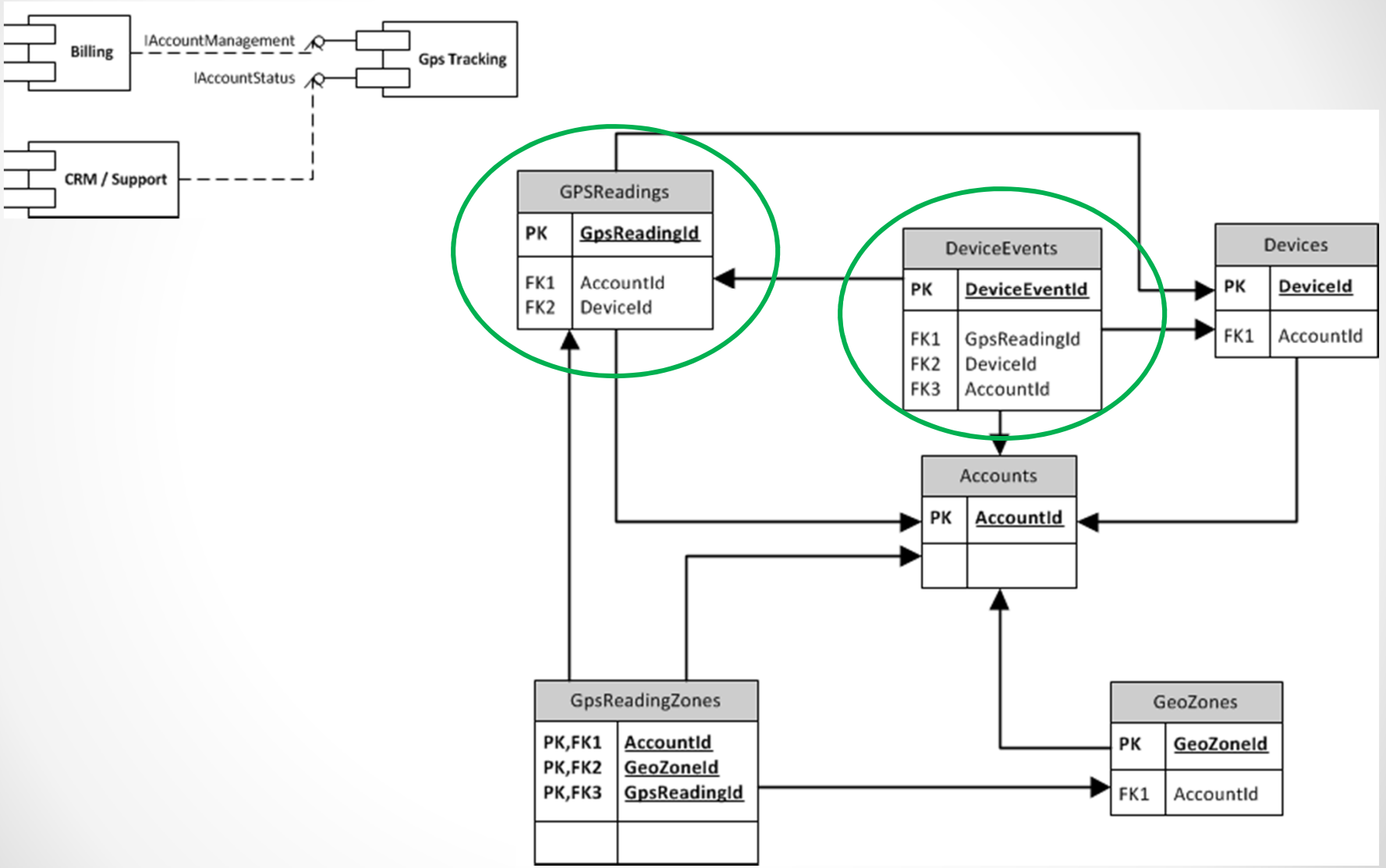
## Share Nothing

- Either
  - Shards are self-contained
  - System implemented as SOA
- Almost unlimited scalability
- Clouds-friendly
- System is partially available if some shards are down
- Legacy is the huge issue
- Higher development cost

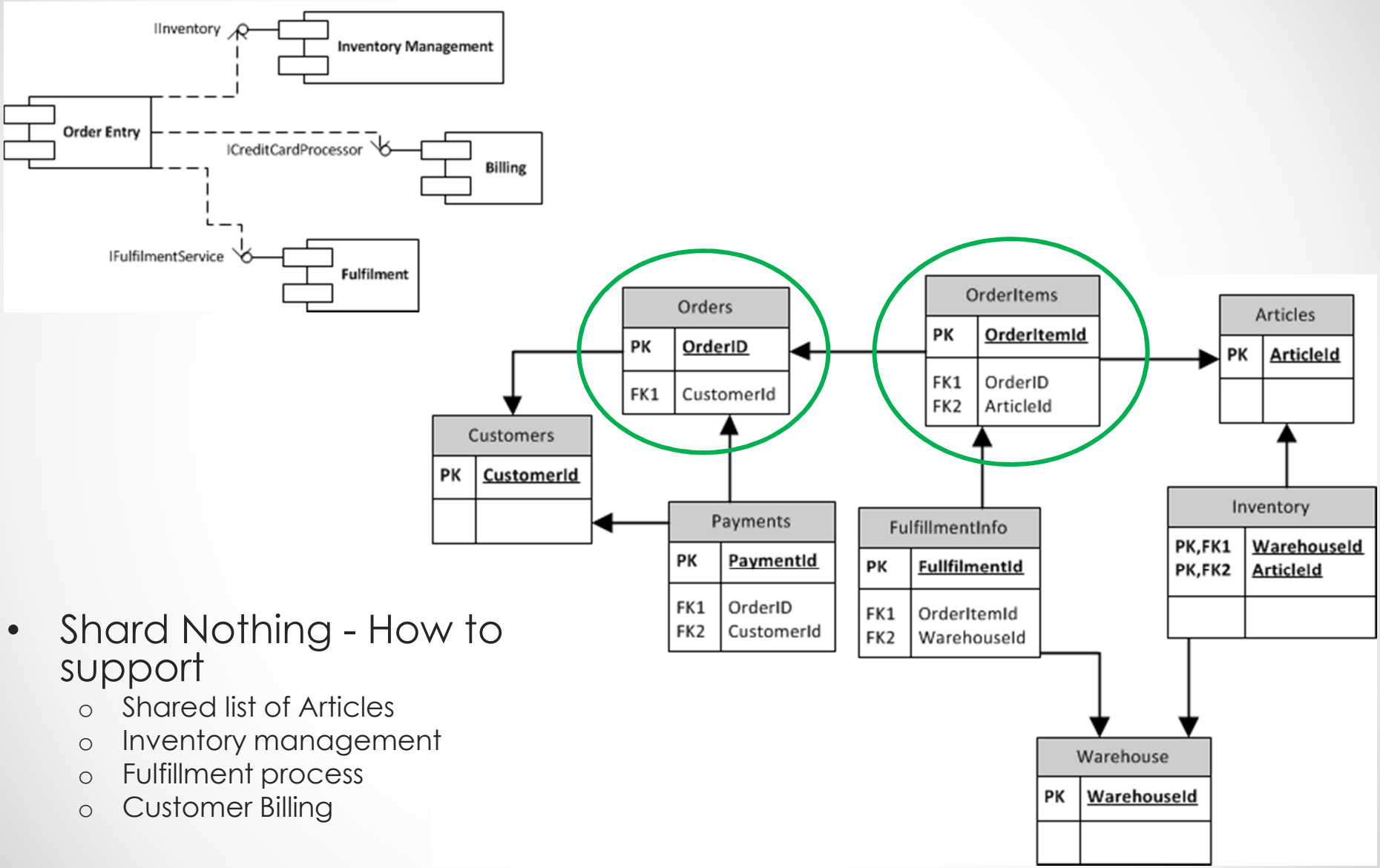
# What to archive

- Transactional data only
  - Keep catalog entities in the main database
- **Keep related data together – goal is to minimize cross-server joins**
- Use same archiving schedule for different entities if possible

# Example 1 – GPS Tracking

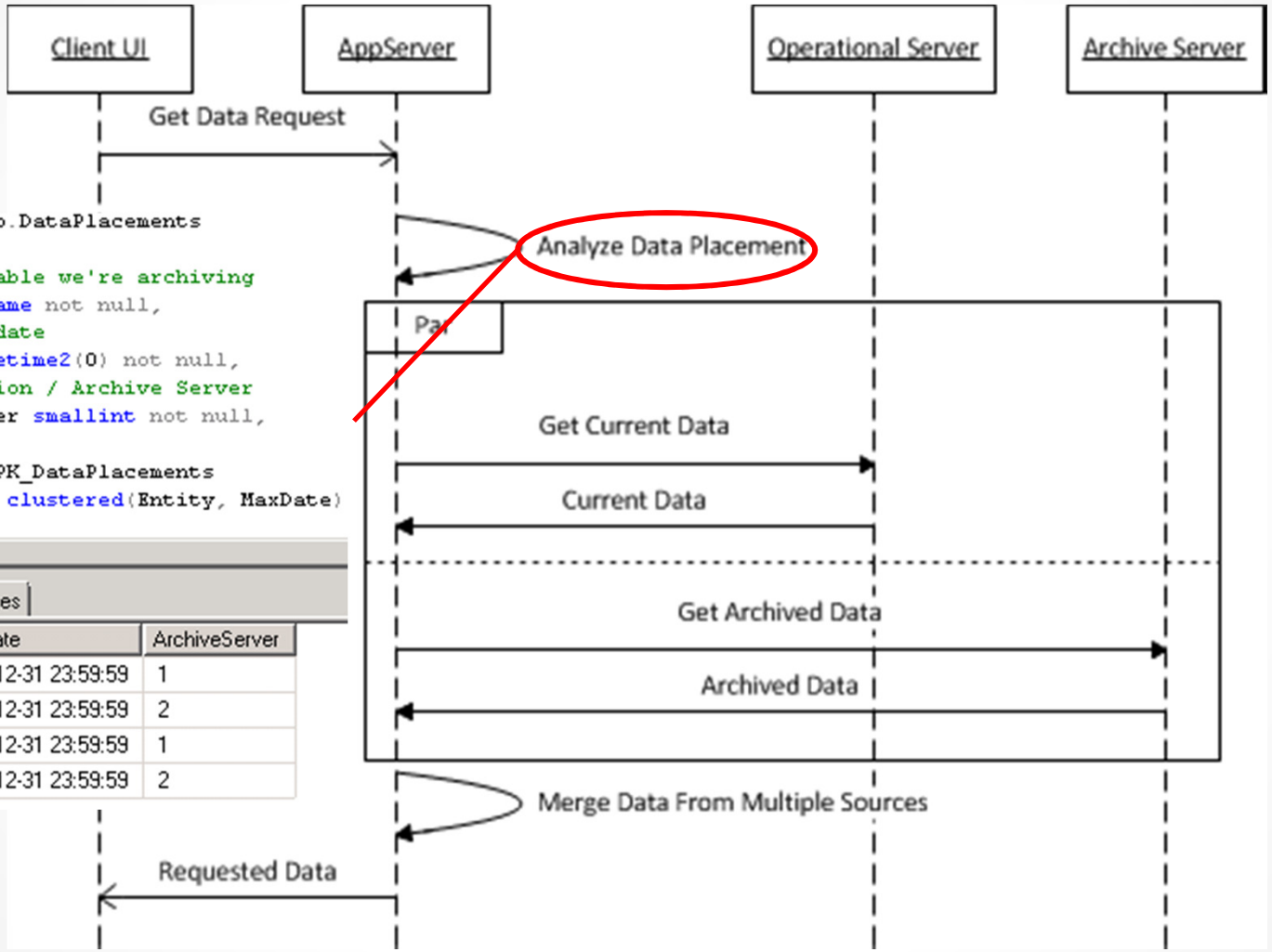


# Example 2 – Online retailer




- Shard Nothing - How to support
  - Shared list of Articles
  - Inventory management
  - Fulfillment process
  - Customer Billing

# Getting Data..




```
create table dbo.DataPlacements
(
    -- Entity/Table we're archiving
    Entity sysname not null,
    -- Archive date
    MaxDate datetime2(0) not null,
    -- Destination / Archive Server
    ArchiveServer smallint not null,

    constraint PK_DataPlacements
    primary key clustered(Entity, MaxDate)
)
```

Results | Messages

| | Entity | MaxDate | ArchiveServer |
|---|---|---|---|
| 1 | AuditTrail | 2009-12-31 23:59:59 | 1 |
| 2 | AuditTrail | 2010-12-31 23:59:59 | 2 |
| 3 | Orders | 2009-12-31 23:59:59 | 1 |
| 4 | Orders | 2010-12-31 23:59:59 | 2 |

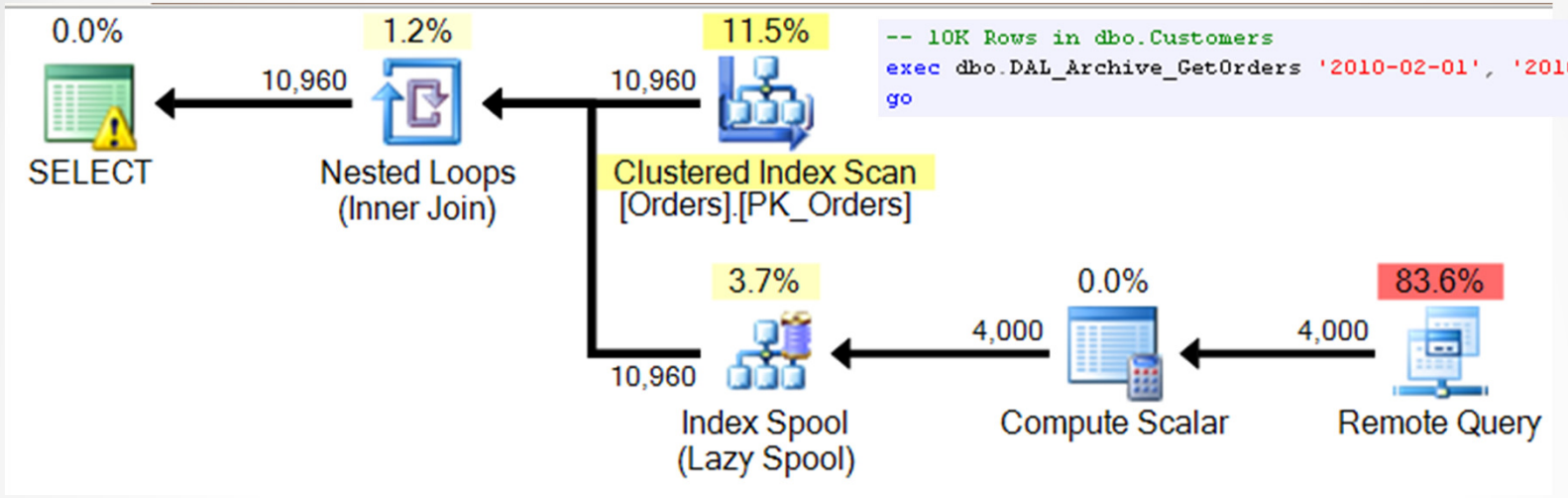
# App Server Considerations

- System must have dedicated Data Access Layer
- 2-tiers data access is preferable
  - App server code
  - Stored Procedures that return unified result sets rather than lookup/merge on app server side

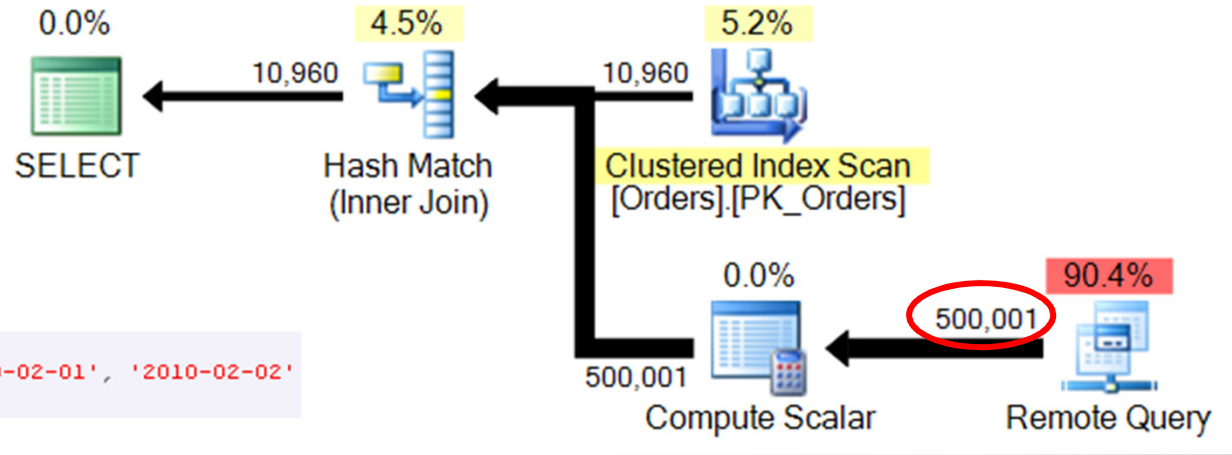
```
create proc dbo.DAL_Prod_GetOrders
(
    @StartDate datetime2(0),
    @StopDate datetime2(0)
)
as
    select
        o.OrderId, o.OrderNum, o.OrderDate,
        o.CustomerId, c.CustomerName, o.Amount
    from
        dbo.Orders o join dbo.Customers c on
            o.CustomerId = c.CustomerId
    where
        o.OrderDate between @StartDate and @StopDate
```

```
create proc dbo.DAL_Archive_GetOrders
(
    @StartDate datetime2(0),
    @StopDate datetime2(0)
)
as
    select
        o.OrderId, o.OrderNum, o.OrderDate,
        o.CustomerId, c.CustomerName, o.Amount
    from
        dbo.Orders o join PRODUCTION.StoreDb.dbo.Customers c on
            o.CustomerId = c.CustomerId
    where
        o.OrderDate between @StartDate and @StopDate
```

# Cross Server Joins




```
-- 10K Rows in dbo.Customers
exec dbo.DAL_Archive_GetOrders '2010-02-01', '2010-02-02'
go
```




```
-- 500K Rows in dbo.Customers
exec dbo.DAL_Archive_GetOrders '2010-02-01', '2010-02-02'
go
```

# Cross Server Joins

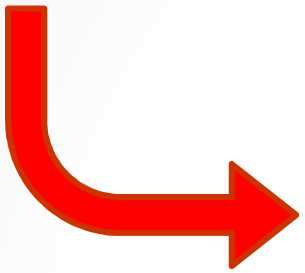
```
create index IDX_Orders_OrderDate on dbo.Orders(OrderDate)
include(CustomerId)
go

create proc dbo.DAL_Archive_GetOrders
(
    @StartDate datetime2(0),
    @StopDate datetime2(0)
)
as
begin
    declare
        @Customers varchar(max)

    create table #Customers(
        CustomerId int not null primary key,
        CustomerName nvarchar(64)
    )

    -- Generate XML: <Data><R @C="1"/>..</Data>
    ;with AnXML(XmlVal)
    as
    (
        select CustomerId as '@C'
        from dbo.Orders
        where OrderDate between @StartDate and @StopDate
        for XML PATH('R'), ROOT('Data')
    )
    select @Customers = CONVERT(varchar(max),XMLVal)
    from AnXML

    insert into #Customers(CustomerId, CustomerName)
        exec PRODUCTION.StoreDb.dbo.DAL_Shard_GetCustomerList @Customers

    select o.OrderId, o.OrderNum, o.OrderDate,
        o.CustomerId, c.CustomerName, o.Amount
    from dbo.Orders o join #Customers c on o.CustomerId = c.CustomerId
    where o.OrderDate between @StartDate and @StopDate
end
```

- Introduces a lot of overhead. Consider pros and cons before implementation
- Good when you know that Customer List is small
- Best when SP needs to filter by Customers too

```
create proc dbo.DAL_Archive_GetOrdersWithCustomerFilters
(
    @StartDate datetime2(0),
    @StopDate datetime2(0),
    @CustomerList dbo.tvpCustomerIDList READONLY
)
```

# Beware of XML

```
alter table dbo.Customers add Attributes xml
go
```

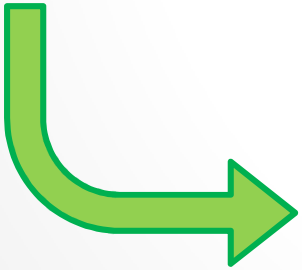


```
select
    o.OrderId, o.OrderNum, o.OrderDate,
    o.CustomerId, c.CustomerName, o.Amount
from
    dbo.Orders o join PRODUCTION.StoreDB.dbo.Customers c on
        o.CustomerId = c.CustomerId
where
    o.OrderDate between @StartDate and @StopDate
```

Messages

```
Msg 9514, Level 16, State 1, Line 6
Xml data type is not supported in distributed queries.
Remote object 'PRODUCTION.StoreDB.dbo.Customers' has xml column(s).
```

```
create view dbo.vCustomers(CustomerId, CustomerName)
as
    select CustomerId, CustomerName
    from dbo.Customers
```



```
select
    o.OrderId, o.OrderNum, o.OrderDate,
    o.CustomerId, c.CustomerName, o.Amount
from
    dbo.Orders o join PRODUCTION.StoreDB.dbo.vCustomers c on
        o.CustomerId = c.CustomerId
where
    o.OrderDate between @StartDate and @StopDate
```

# Archive Server Availability

- Define the strategy how to handle cases when archive instance is down
  - Read-only reporting – run against operational database and present warning message to the user
  - Read-write activity – either rollback or implement asynchronous approach
    - Queue table with SQL Jobs
    - Service Broker
- **Beware of Stored Procedures recompilation when linked servers are referenced**

# Moving data, storing data

- Custom code, SSIS, Replication, … - whatever works
- Operational server – sliding window pattern
  - Partition data if possible
    - Move data on partition-by-partition basis
    - Consider to move data from the temporary table after partition switch
- **Optimize data schema on Archiving server for reporting rather than for operational activity**
- Data compression
  - Operational server – use ROW level compression
  - Archive servers – use PAGE level compression
- Use multiple filegroups with Archive server. Think about piecemeal restore.

# Conclusion

- Vertical partitioning is good solution when
  - There are a lot of dependencies  in the system
  - There are a lot of legacy software that needs to be supported
- Implement 2 tier Data Access Layer with client code and stored procedures
- Beware cross-server joins
- Optimize data schema on Archiving server for reporting purposes
- Implement error handling strategy for the cases when Archive server is down

# Q & A

- Thank you for attending!
- Session will be available for download
  - http://aboutsqlserver.com/presentations
  - http://sqlsaturday.com

- Email: dmitri@aboutsqlserver.com