
ORACLE®

Oracle Database 12c Release 2 CoreTech Seminar

12.2.0.1
Sharding

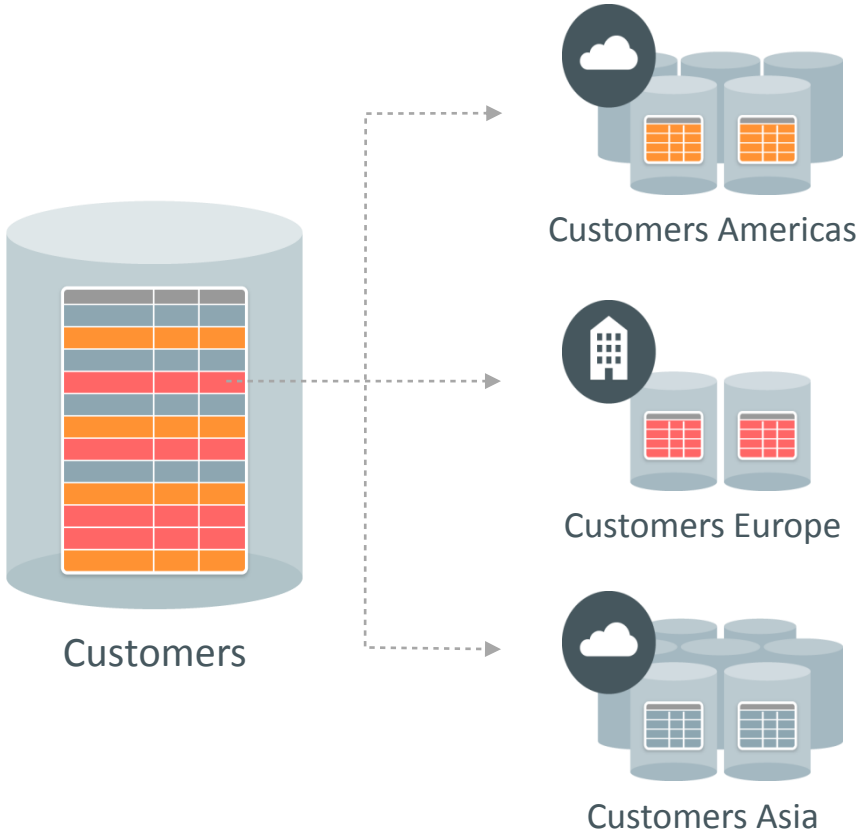
日本オラクル株式会社
クラウド・テクノロジー事業統括
Cloud/Big Data/DISプロダクト本部
後藤　陽介
2016/10

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

New in 12.2 on Oracle Cloud

極めて高い拡張性と信頼性を求めるOLTPアプリのためのデータベース･シャーディング



1つの巨大なデータベースを多数の小さなデータベース(シャード)に分割

- 99%のアプリケーションの要件に対し、アプリケーションの透過性を維持という点でもRACとData Guardが適合する
- グローバル規模のOLTPアプリケーションでは巨大なデータベースをより小さなデータベース・ファームに分割するほうが適する
- ワークロードがファームの特定のシャードに自動的に到達できるようなアプリケーションの設計が必要
- 最大1000シャードに分割された表へのSQL

本セッションの目的

- 機能の位置付けを理解する
- 動作を理解する
- 制限事項・注意事項を理解する

Agenda

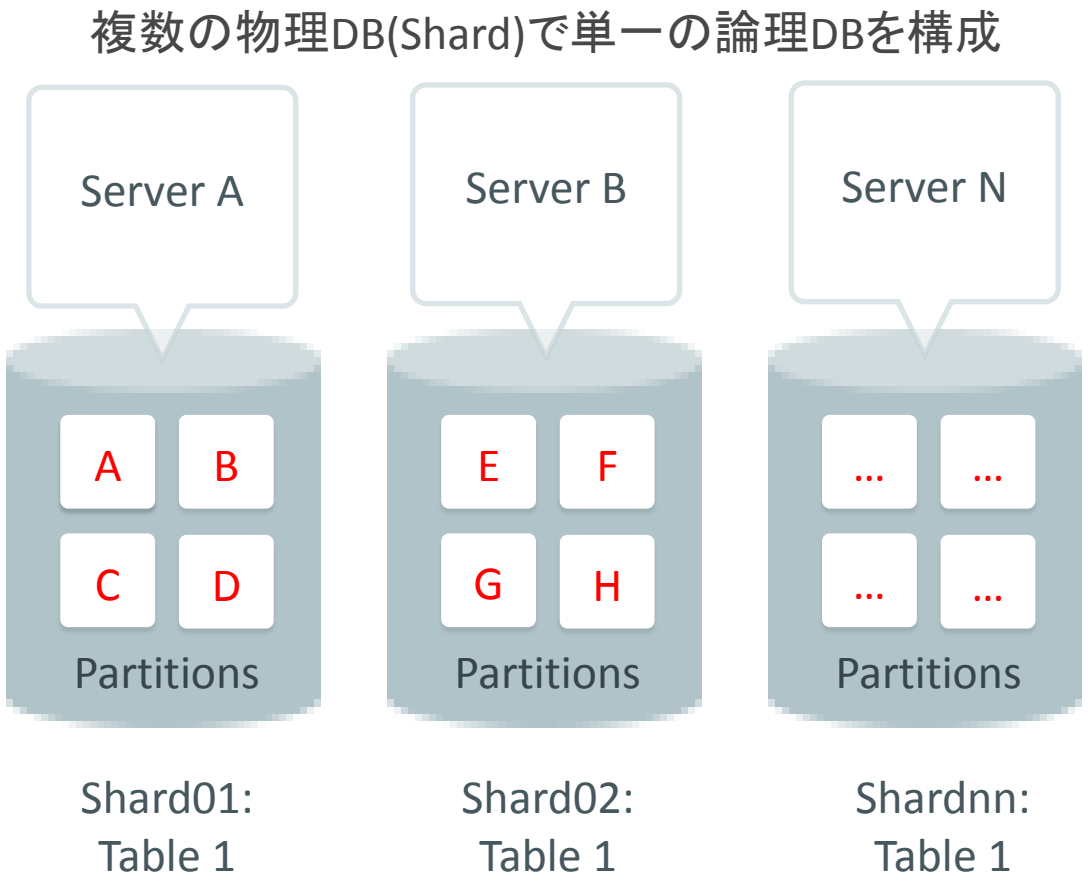
1. 機能概要
2. 環境構築
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

Agenda

1. 機能概要
2. 環境構築
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

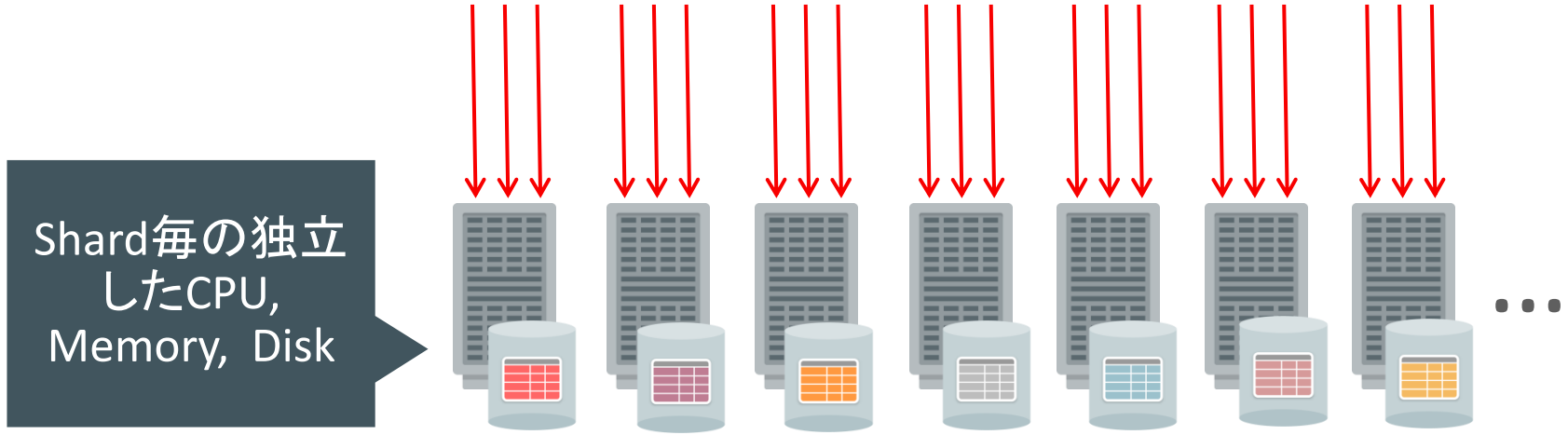
Database Sharding

- 複数のDBサーバーで論理DBを構成
  - 水平分割(Horizontal Partitioning)
  - パーティション・キーを指定(顧客IDなど)
- DBサーバー間で共有H/Wを持たない
  - 共有ディスク不要
  - クラスタウェア不要
  - インターコネクト不要
  - **Cloud環境にも適した構成**
- 単一の物理DBを”Shard”と呼ぶ



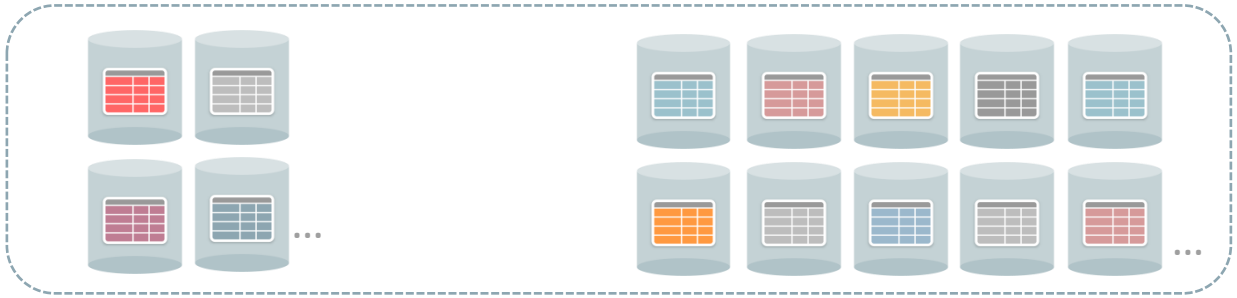
Sharding のメリット – スケーラビリティ

- ワークロードを**均等に分散**することで、リニアなスケーラビリティが得られる
  - 数10、数100レベル
  - 安価なH/Wで構成可能



Cloud環境でのSharding

Private, Public or Hybrid

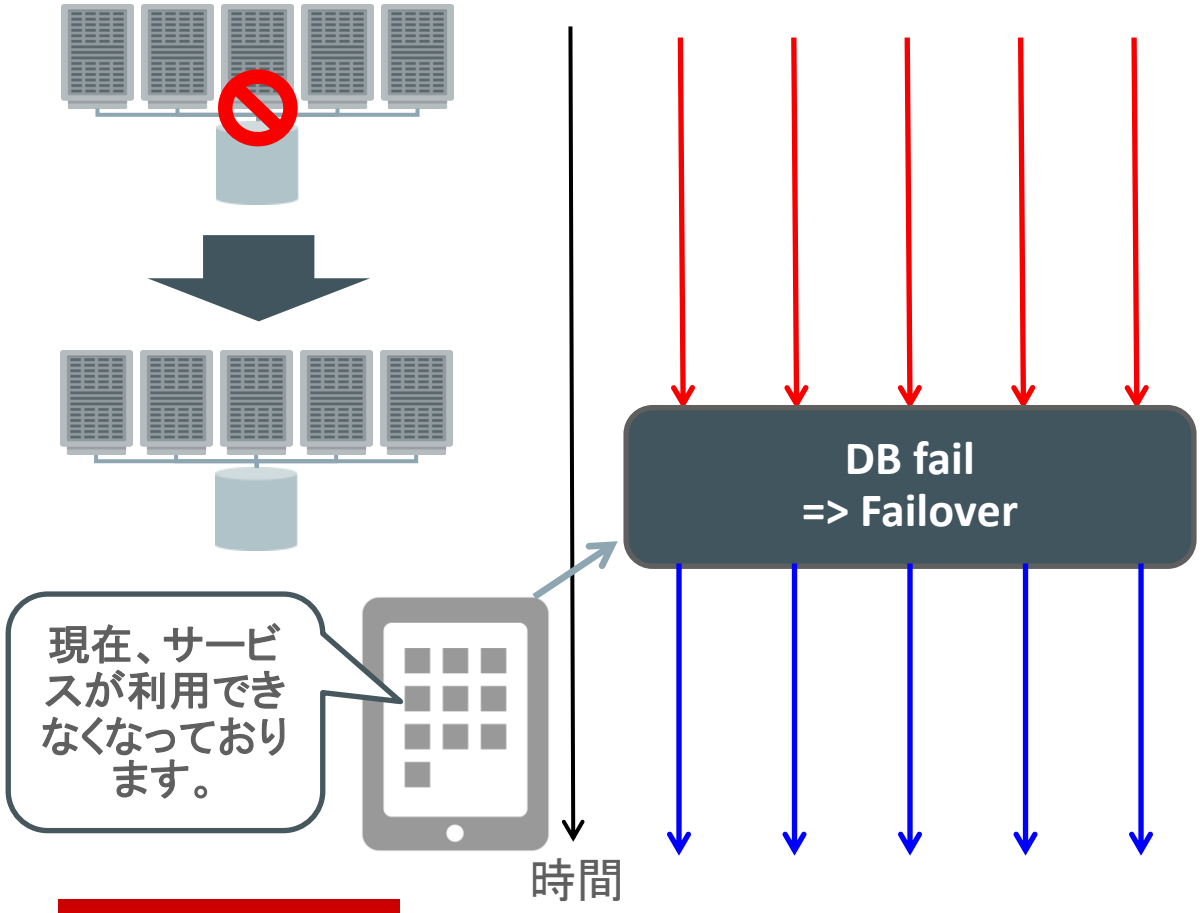


Scale out & Scale Back

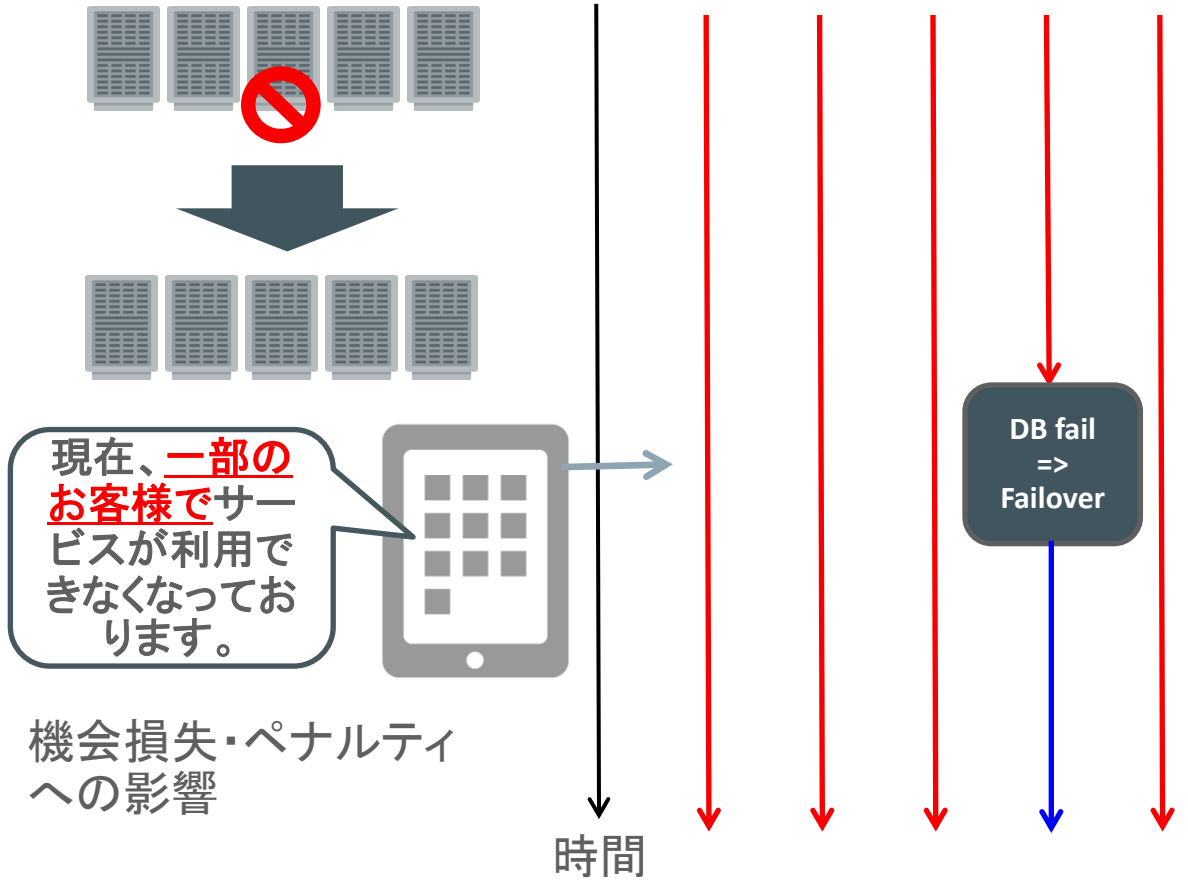
- **期間限定のキャパシティ拡張に対応 – より多くのトランザクション、顧客、データ**
- **新製品・サービスのローンチなど、新規顧客のためのキャパシティ拡張に対応**

Sharding のメリット – 障害影響範囲の極小化

Cluster + Standby DBの業務継続イメージ



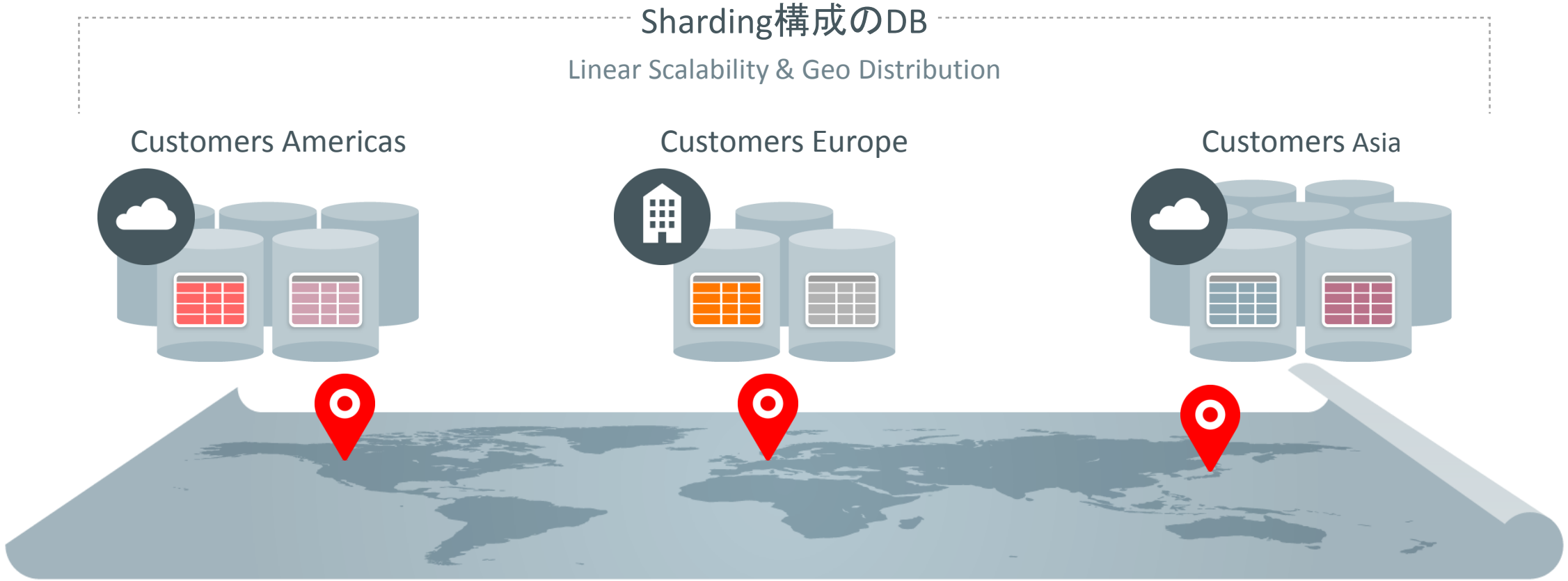
Sharding + Standby DB の業務継続イメージ



機会損失・ペナルティへの影響

Shardingのメリット - 地理分散

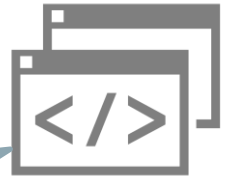
分散配置, Hybrid Cloud構成で論理DBを構築

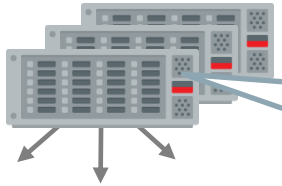


Shardingを構成する上での課題

アプリケーション

•全でのShardでテーブル作成・変更をする必要がある
•集計バッチはどう作れば良いだろうか？





ルーティング

•顧客ID「XXXX」のレコードはどのShardにある？
•新規顧客のレコードはどのShardに入れる？

DB構築・管理

•構築が大変そうだ・・・
•後からShardを追加した後の領域管理はどうするか・・・

Oracle Database 12c Release 2

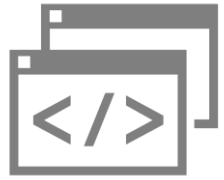
フレームワークにより”単一のDB” としてSharding を使用可能

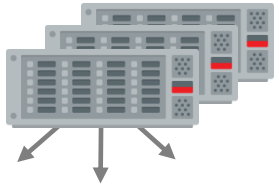
アプリケーション

• スキーマオブジェクトの一括管理
•複数Shardへの一括検索

ルーティング

• 接続時にキー(Sharding Key)を指定すれば、正しいShardに自動ルーティング





Sharding 構成・管理フレームワーク

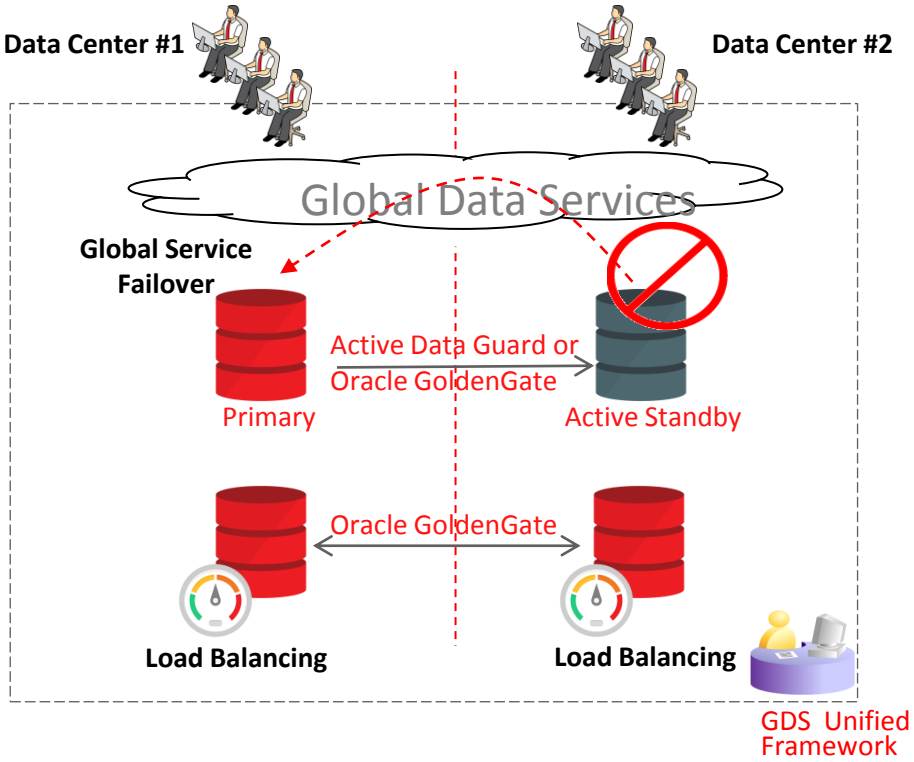
DB構築・管理

• Shard の自動構成
• Shardの追加・削除時のデータの自動再配置

FAQ

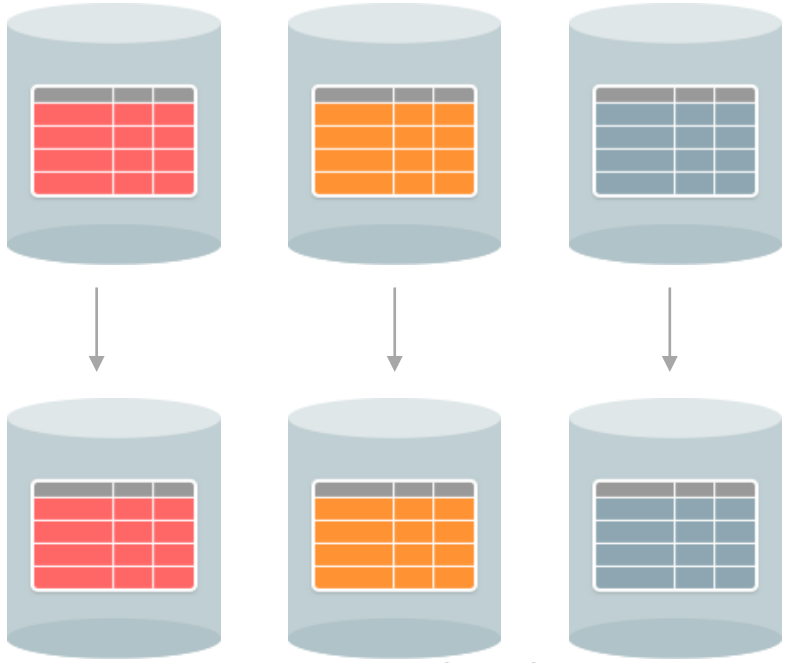
- Q1 : Sharding構成・管理フレームワークとは何ですか？
- A1 : Global Data Serviceを使用しています

- **Global Data Service(GDS)とは**
  - Oracle Database 12.1からの機能
  - レプリカ(Active Data Guard/GoldenGate)など複数DBをまたがった接続ルーティング、負荷分散の仕組みを提供
  - Oracle Database 12.2 よりデータを分散配置したDB(Sharding)の構成・管理フレームワークとしても動作

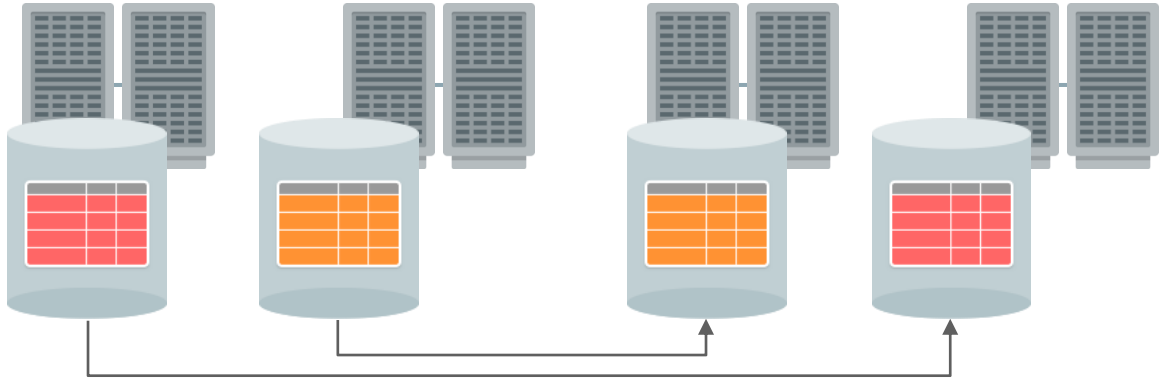


FAQ

- Q2 : 可用性・耐障害性は？
- A2 : Shard単位でData Guard, RAC を構成します。今後GoldenGateに対応予定です。Shard単位でバックアップ管理を行います。



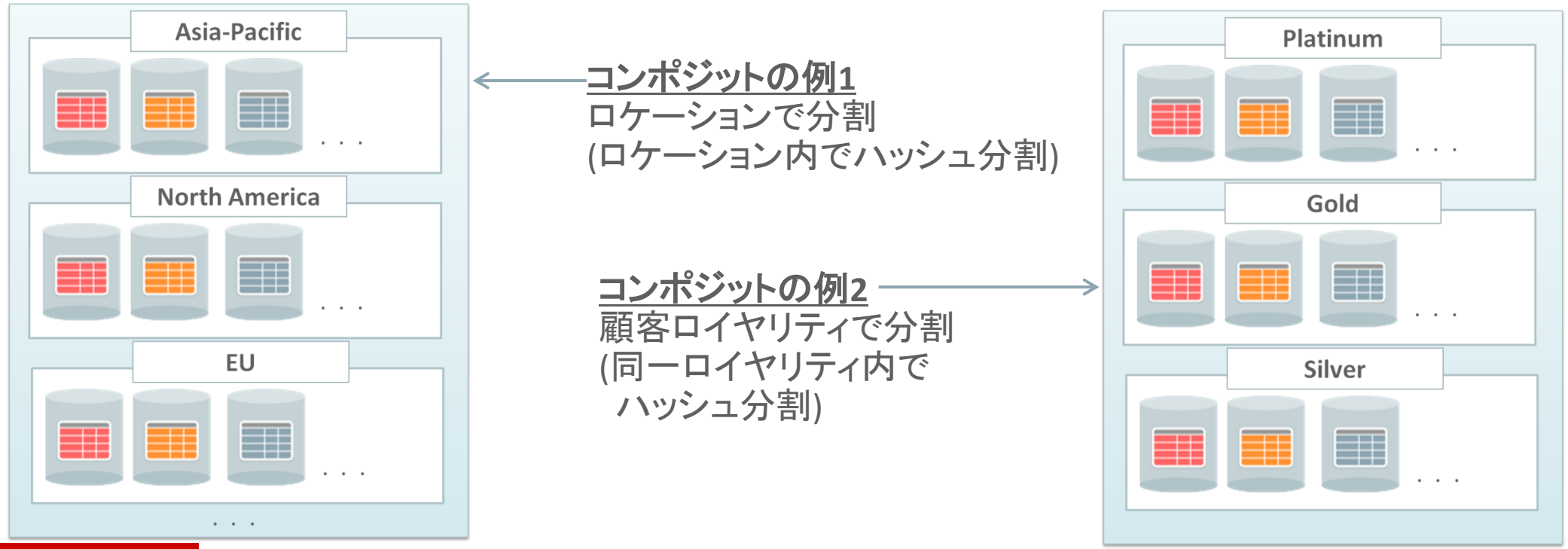
Active Data Guard with Fast-Start Failover



Sharding & RAC

FAQ

- Q3 : データはどのように分割されるか？
- A3 : 初期リリースでは、ハッシュとコンポジット(レンジーハッシュ、リストーハッシュ)に対応します



FAQ

- Q4 : どのようなアプリケーションが適していますか？
- A4 : 多数のアクセスユーザーを持つOLTPアプリケーションに適しています

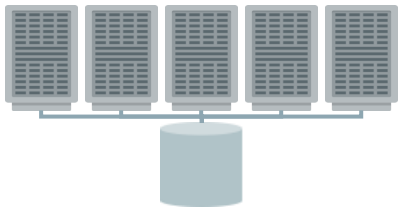
アプリケーションの例

- Large billing systems
- Airline ticketing systems
- Online financial services
- Media companies
- Online information services
- Social media companies

FAQ

- Q5 : RAC と Sharding の使い分けは？
- A5 : 要件・処理特性が適するものについてShardingの使用を検討します。RACとShardingの組み合わせも可能です

RAC
•アプリケーションの透過性に優れており(全てのサーバーから全てのデータにアクセス可能)汎用的
•高い拡張性・可用性



Sharding
•データが分散されるので、アプリケーションの考慮が必要
•極めて高い拡張性、障害範囲の極小化

Agenda

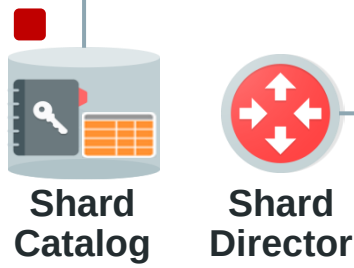
1. 機能概要
2. **環境構築**
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

Oracle Sharding の構成イメージ

業務ロジック・
DBへの接続リクエスト

Application

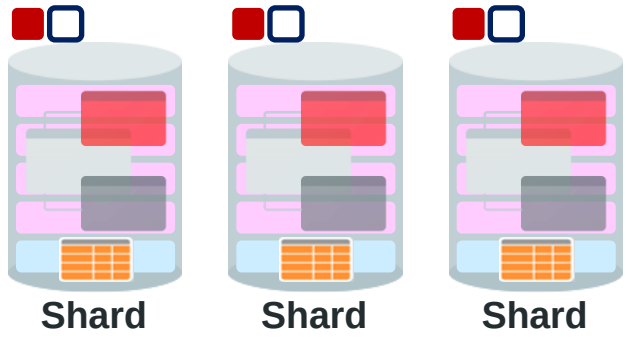
Shardingの構成・管理
フレームワーク



Shard Catalog
Shardingの構成情報や構成・管理タスクを保持するリポジトリDB

Shard Director
各Shardに対する構成・管理処理の実行
アプリケーション接続ルーティングを行う
インスタンス(OSプロセス群+共有メモリ)

アプリケーションの
データを分散配置する
DB群(Shard)



Listener
Scheduler Agent

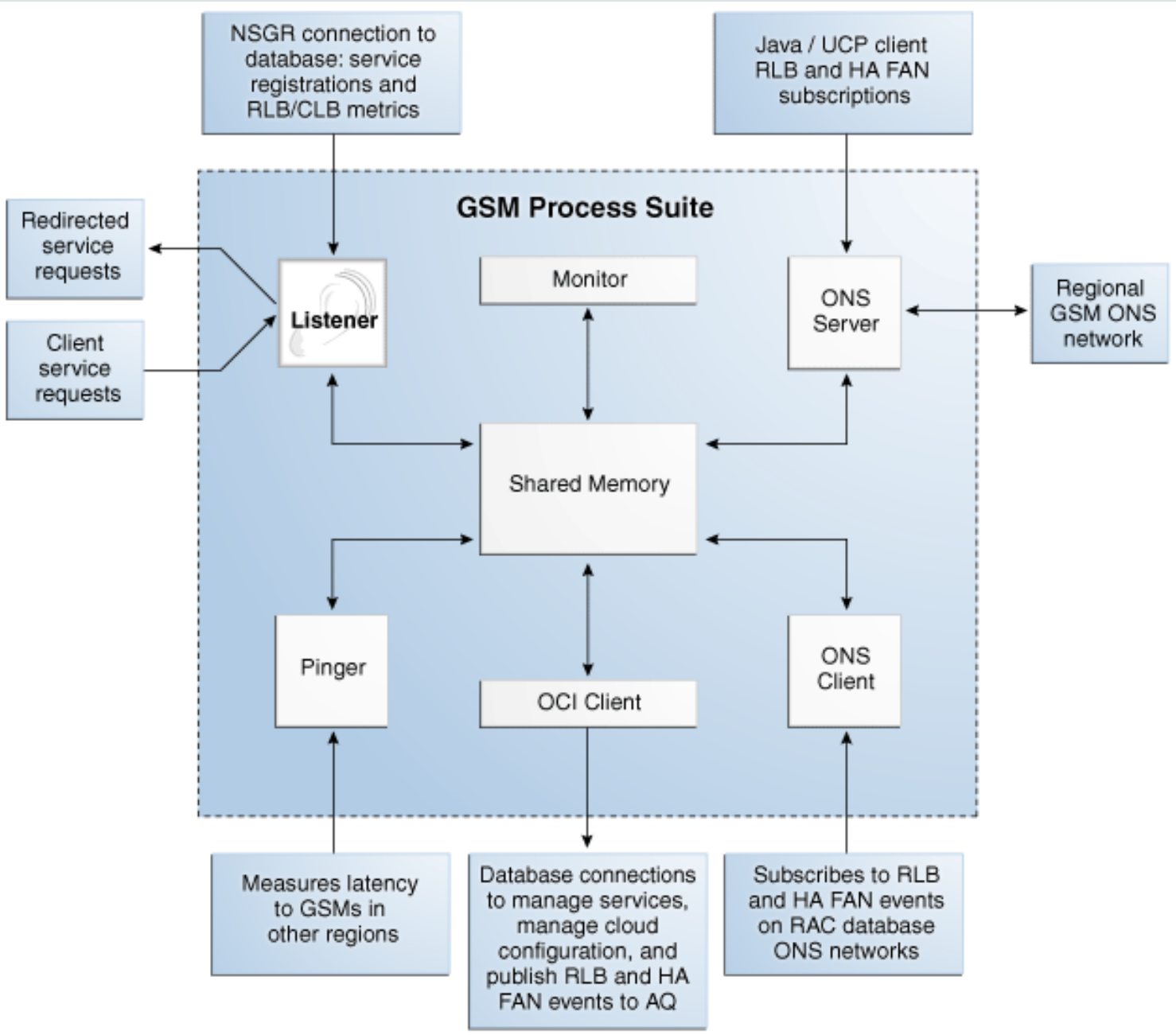
Shard Catalog – GSMADMIN_INTERNALスキーマ

```
SQL>  select object_type, count(*) num_of_objects from all_objects
      where owner = 'GSMADMIN_INTERNAL'  group by object_type order by num_of_objects desc;

OBJECT_TYPE             NUM_OF_OBJECTS
----------------------- --------------
TYPE                                45
INDEX                               41
TABLE                               41
JOB                                 19
SEQUENCE                            16
PACKAGE                              8
VIEW                                 8
PACKAGE BODY                         8
TRIGGER                              5
LIBRARY                              4
TYPE BODY                            3
SYNONYM                              2
RULE SET                             2
PROCEDURE                            2
EVALUATION CONTEXT                   1
CREDENTIAL                           1
```

- Oracle Databaseのデフォルトスキーマ
- Shardingを構成・管理するテーブル群やパッケージを保持

Shard Director

- 複数のOSプロセスと共有メモリーから構成される
- GSM はShard Directorの別名



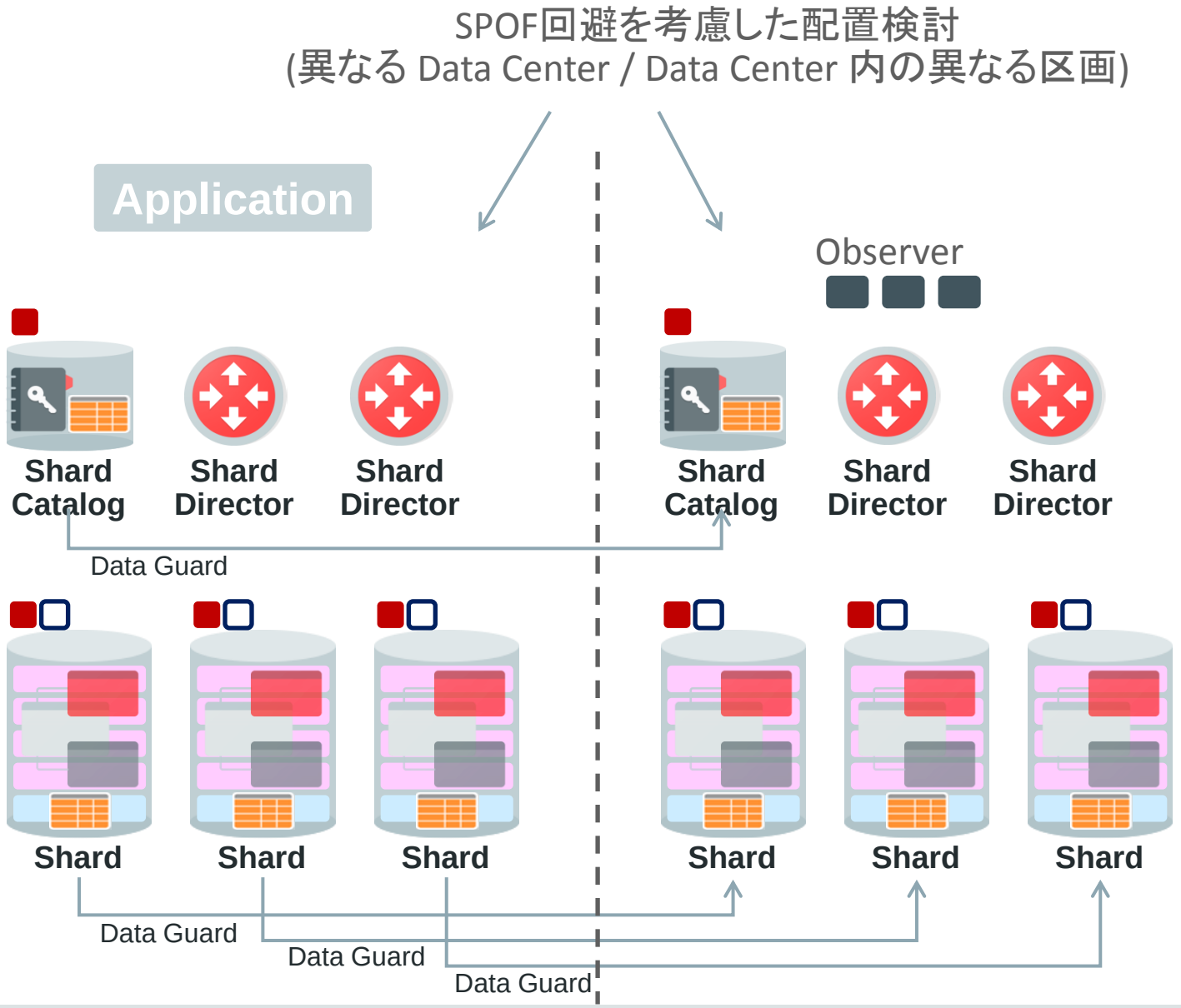
Shard Director

- OSから確認できるプロセス群

```
$ ps -ef | grep SHARDDIRECTOR | grep -v grep
oracle   26381     1  0 Sep29 ?       00:00:20 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/gsmmon SHARDDIRECTOR1
-inherit
oracle   26383     1  0 Sep29 ?       00:00:30 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/gsmoci
ifile=/u01/app/oracle/product/12.2.0/gsmhome_1/network/admin/gsm.ora SHARDDIRECTOR1 SNLSM:4ab30005 -inherit
oracle   26393     1  0 Sep29 ?       00:01:40 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/tnslsnr
ifile=/u01/app/oracle/product/12.2.0/gsmhome_1/network/admin/gsm.ora SHARDDIRECTOR1 SNLSM:4ab30005 -inherit
-mode gsm
oracle   26395     1  0 Sep29 ?       00:00:29 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/gsmping
ifile=/u01/app/oracle/product/12.2.0/gsmhome_1/network/admin/gsm.ora SHARDDIRECTOR1 SNLSM:4ab30005 -inherit
oracle   26397     1  0 Sep29 ?       00:00:30 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/gsmopxy
ifile=/u01/app/oracle/product/12.2.0/gsmhome_1/network/admin/gsm.ora SHARDDIRECTOR1 SNLSM:4ab30005 -inherit
oracle   26399     1  0 Sep29 ?       00:00:01 /u01/app/oracle/product/12.2.0/gsmhome_1/bin/gsmonsc
ifile=/u01/app/oracle/product/12.2.0/gsmhome_1/network/admin/gsm.ora SHARDDIRECTOR1 SNLSM:4ab30005 -inherit
```

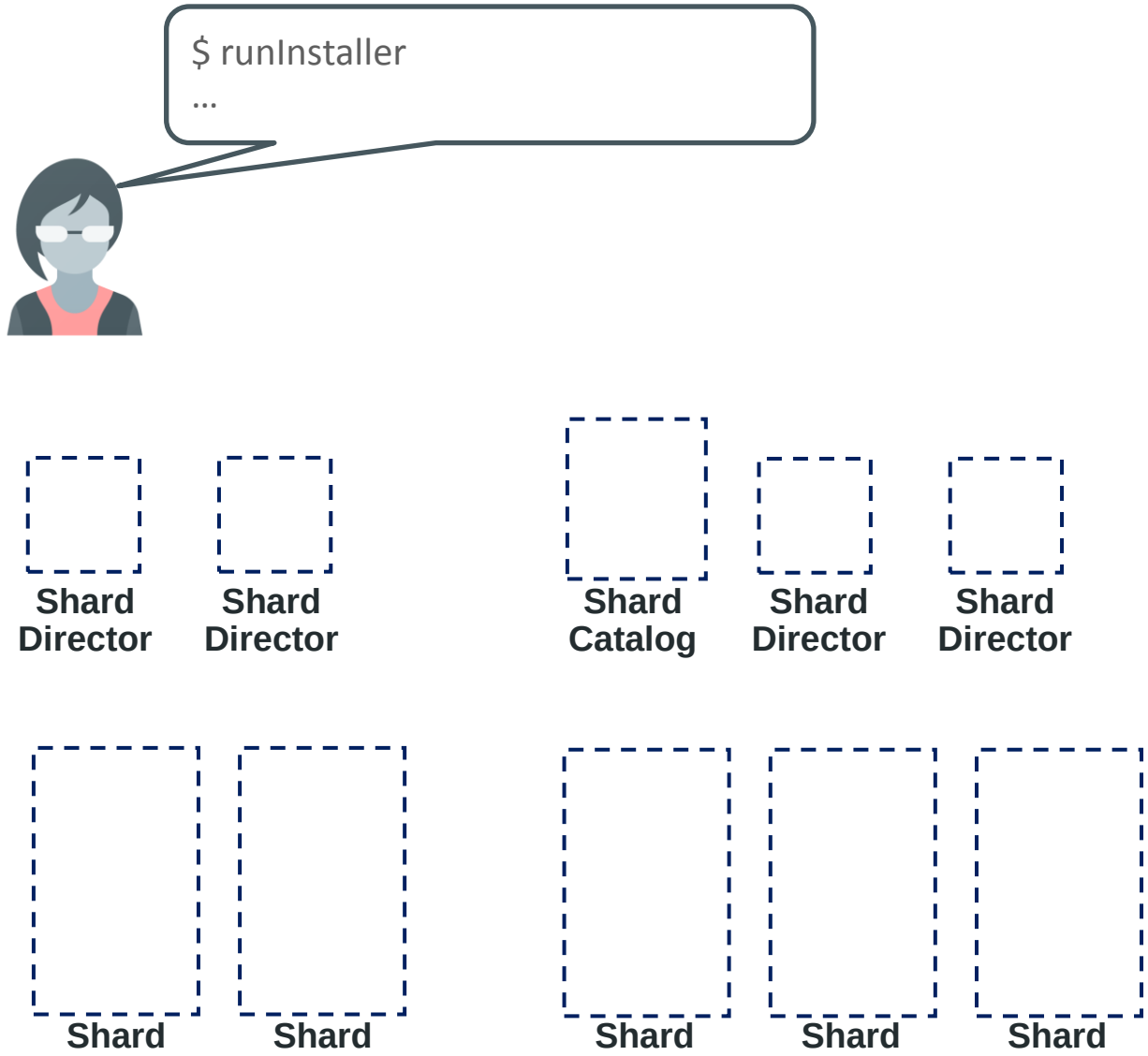
現実的な構成イメージ

- 可用性を高めるために
  各層を冗長化する
  - Shard CatalogをData Guard構成にする
  - Shard Directorを複数起動する
  - 各ShardでData Guardを構成する
  - 本構成はSystem-Managed Sharding(ハッシュ分割)を想定



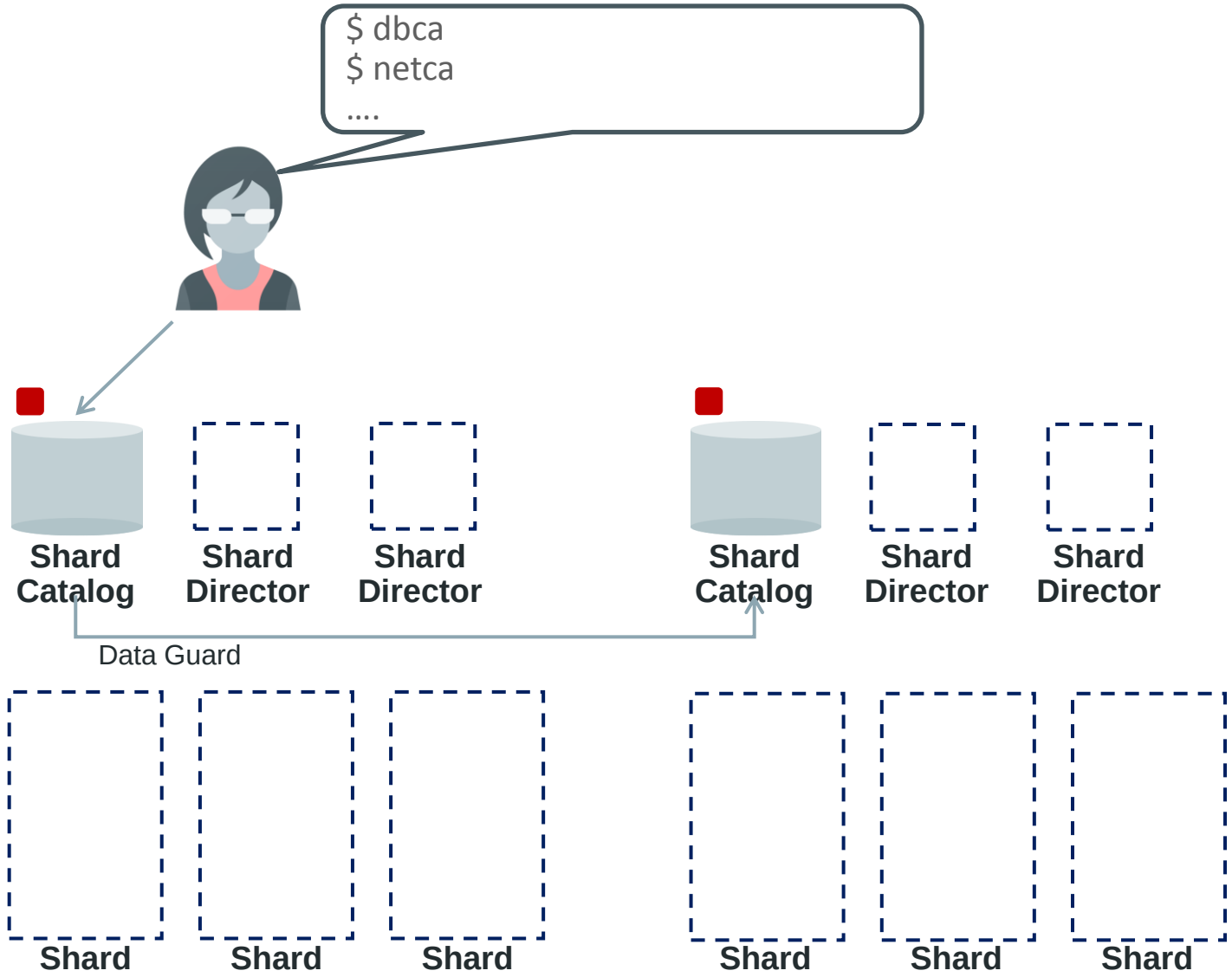
導入の流れ

- コンポーネント毎にサーバーを用意
- ソフトウェアをインストール
  - Shard Catalog : Oracle Database
  - Shard : Oracle Database
  - Shard Director : Global Service Manager (データベースのメディアに含まれています)



導入の流れ

- Shard カタログとして使用するDBとリスナーを手動作成(DBCA / NETCA)
  - Data Guard化のタイミングは任意

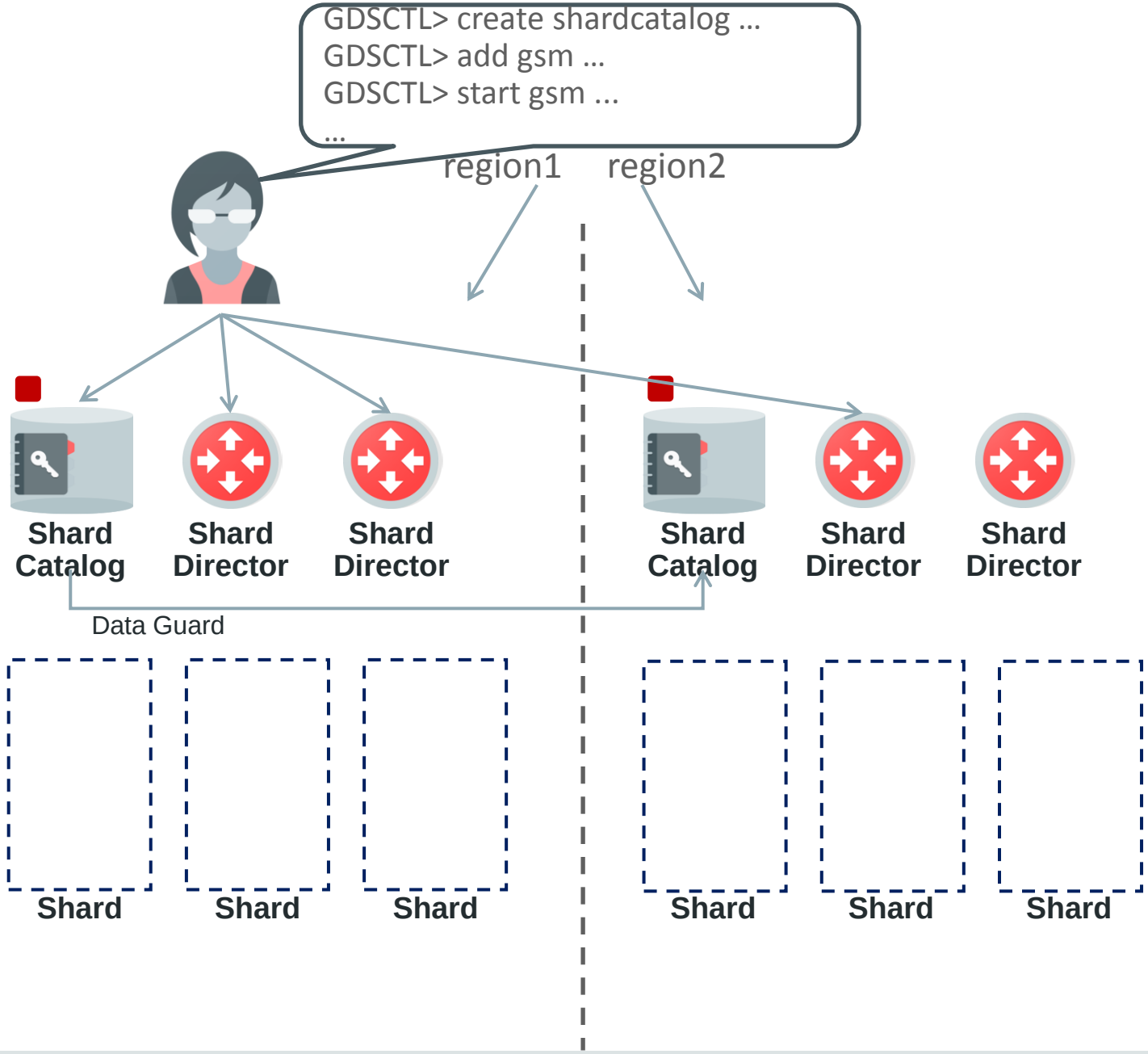


Sharding構成・管理コマンドラインツール

- GDSCTL
  - Global Service Manager の Oracle Homeより起動
  - GDSCTLで実行した構成内容はShard Catalogに格納される
  - マニュアル「Global Data Services Concepts and Administration Guide」にコマンドリファレンスを記載

導入の流れ

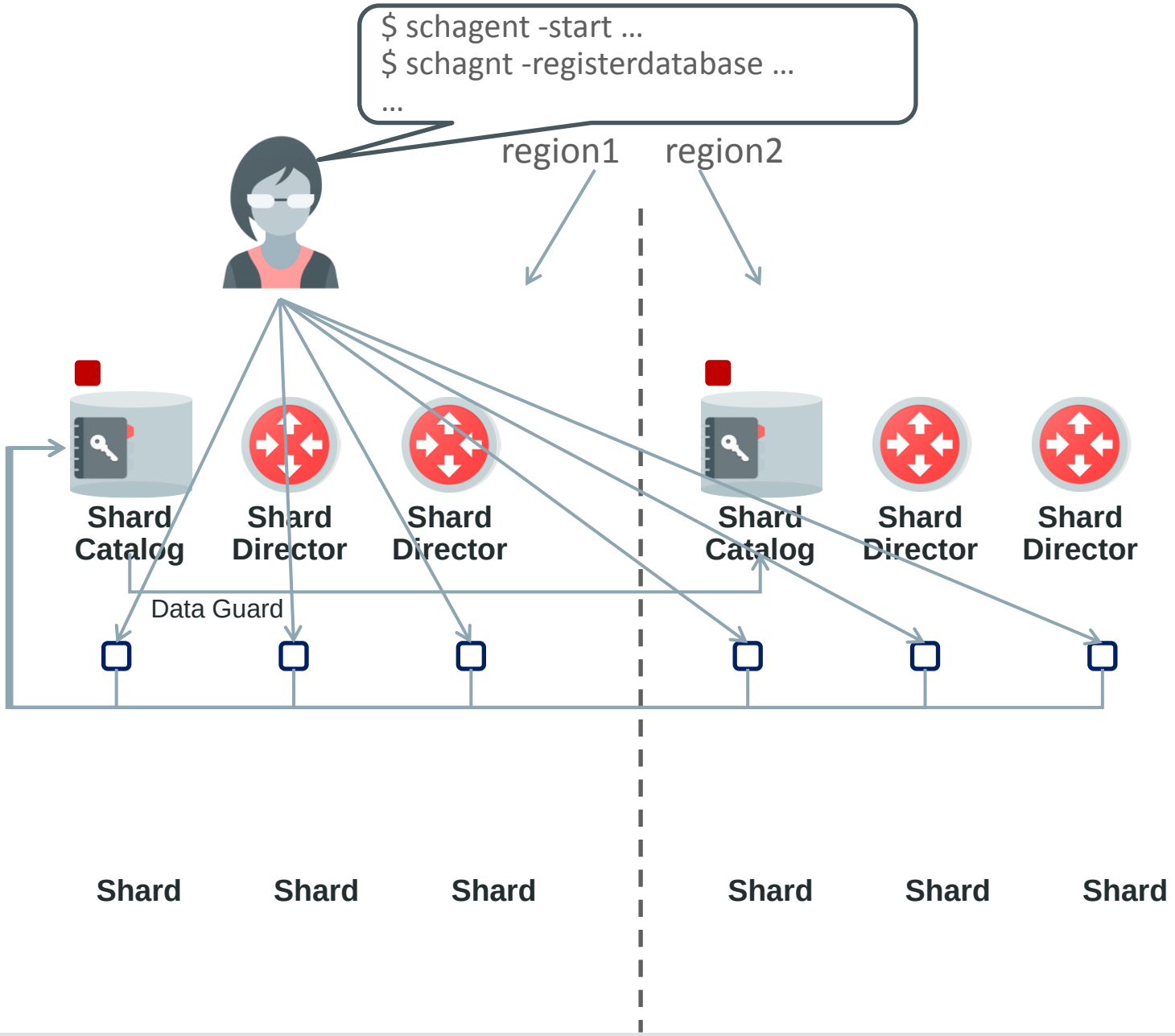
- Sharding構成の定義
  - Shard Catalogとして使うDBの指定
  - Shardingメソッド(System-Managed (ハッシュ) or Composite)を指定
  - コンポーネントの配置ロケーション (今回の例では region1 / region2)
  - etc
- Shard Directorの構成と起動
  - Shard Catalogとregion に紐づける



導入の流れ

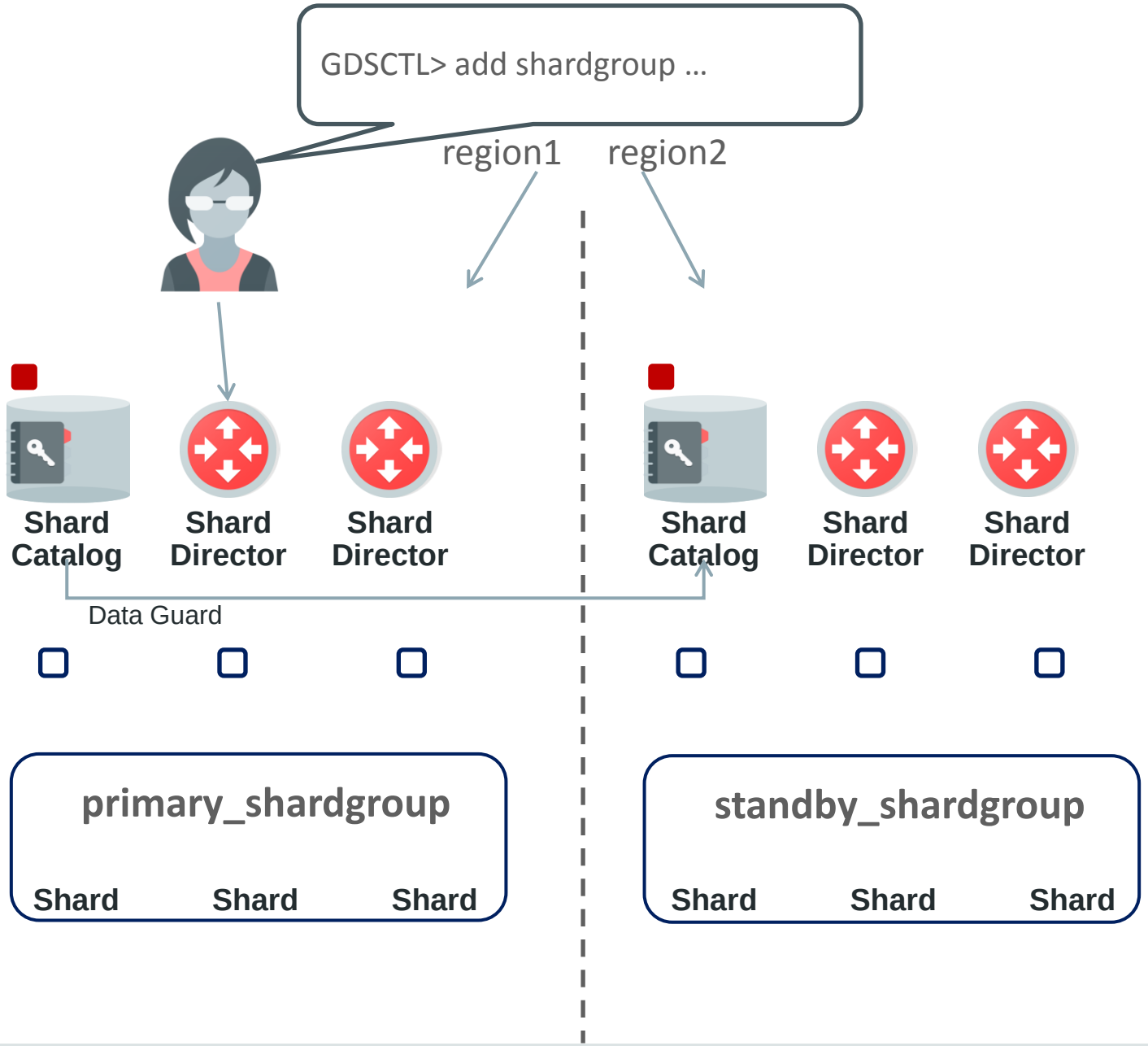
- Scheduler Agent の登録
  - 各ShardのOracle Homeより、Scheduler Agentを起動
  - Scheduler Agentの情報をShard Catalogに登録

従来のリリースから存在する、
Oracle Schedulerのリモートジョブ実行の仕組みをSharding構成時に使用
(NETCA, DBCAの実行など)



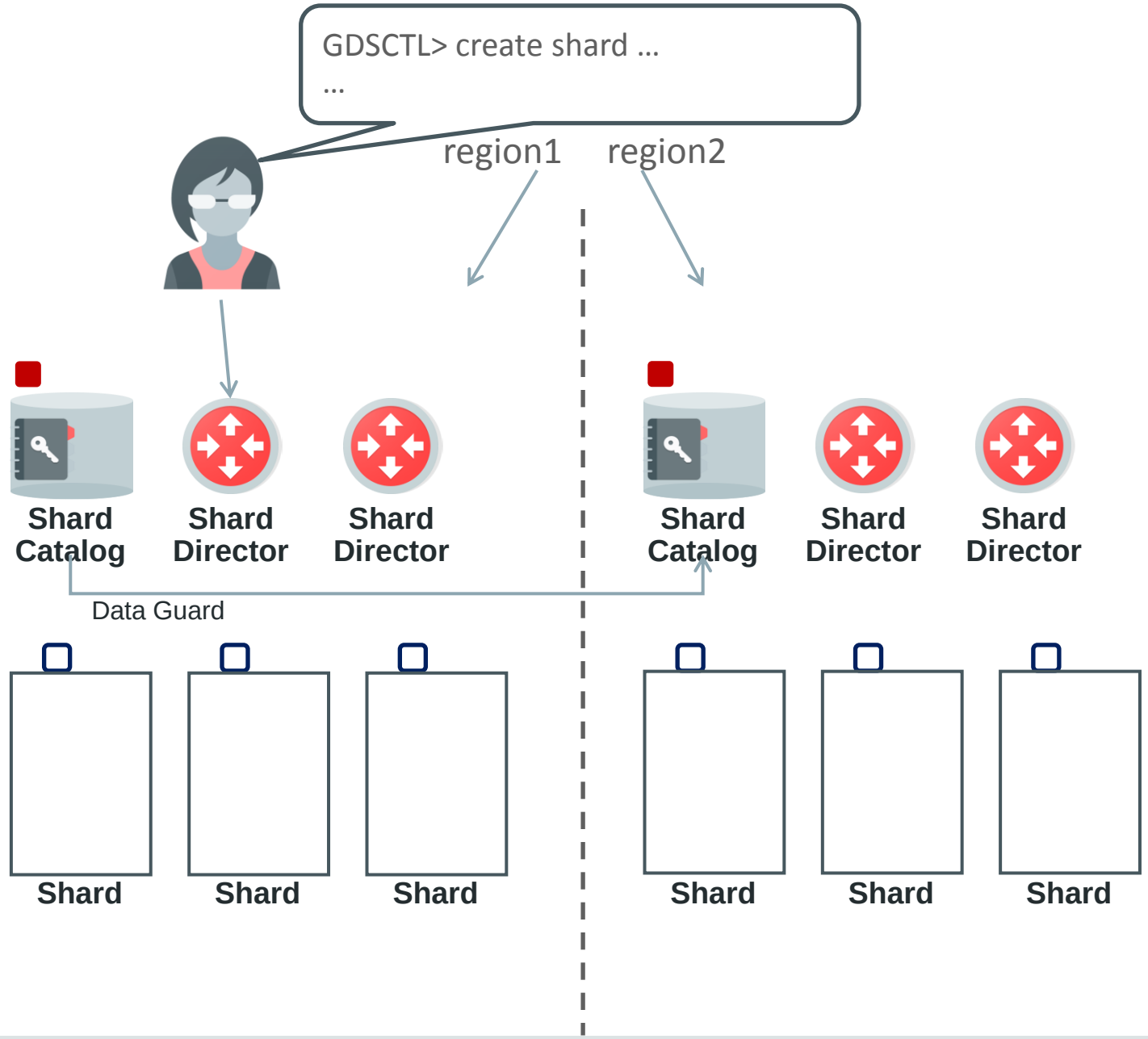
導入の流れ

- Shard Group の作成
  - Data GuardのプライマリになるshardとスタンバイになるShardのグループ名をそれぞれ指定
  - Shard Groupをregionに紐づける
  - Shard Groupの複数指定(複数スタンバイ構成)も可能



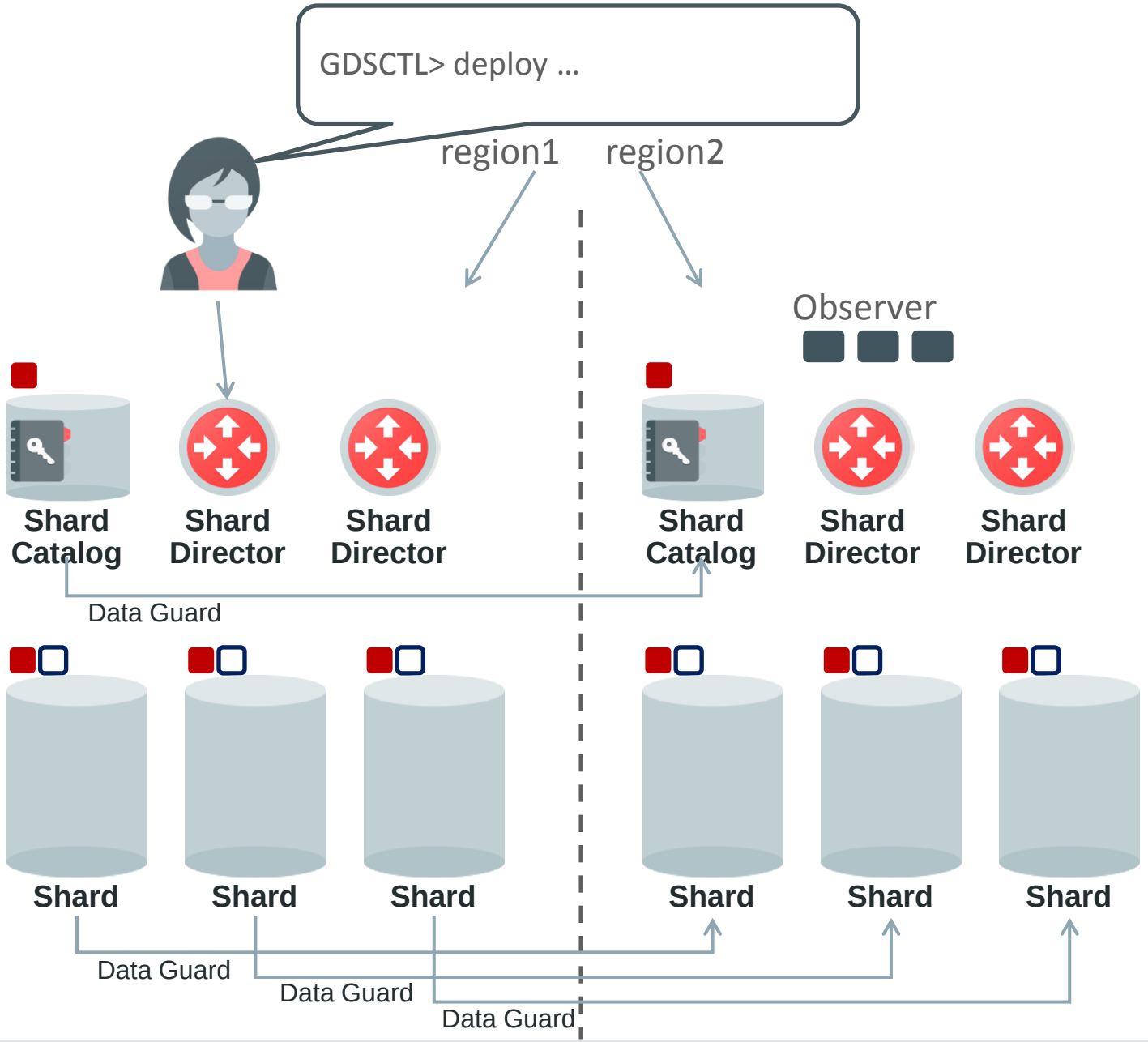
導入の流れ

- Shardの定義
  - Oracle Databaseをインストールした各サーバーをShardとして定義する
  - ShardがどのShard Groupに属するかを指定
- デフォルトのDB構成
  - “SH1”から昇順にdb_unique_nameが振られる
  - Oracle Home内のデフォルトのDBCAテンプレートが使用される
- 自作のDBCAテンプレートを使用することも可能



導入の流れ

- Sharding構成の実装(デプロイ)
- Shard Catalogからリモートスケジューラジョブとして以下を実行
  - NETCAでListener作成
  - DBCAでDB作成
  - RMANでスタンバイ・データベースの作成
  - Data Guard Broker / Fast-start Failoverの設定(Observerはスタンバイ側regionのShard Directorホストに配置)
- データベース・ロールに紐づくサービスを追加



GDSCTLコマンド

主要コンポーネント

create shardcatalog **-database** <shardcat host>:1521:shardcat **-user** sdb_admin/passwd_sdb_admin
add gsm **–gsm** sharddirector1 **-listener** 1571 **-pwd** passwd_gsmcatuser **-catalog** <shardcat host>:1521:shardcat
..
add credential -credential oracle_cred -osaccount oracle -ospassword < >
add shardgroup **-shardgroup** shgrp1 -deploy_as primary **-region** Availability_Domain1
create shard **-shardgroup** shgrp1 **-destination** <host1> **-credential** oracle_cred
…

可用性を考慮した複製コンポーネント

add gsm **–gsm** sharddirector2 **-listener** 1571 **-pwd** passwd_gsmcatuser **-catalog** <shardcat host>:1521:shardcat
..
add shardgroup **-shardgroup** shgrp2 -deploy_as active_standby **-region** Availability_Domain2
create shard **-shardgroup** shgrp2 **-destination** <host11> **-credential** oracle_cred
…

deploy
add service **-service** oltp_rw_srvc **-role** primary

Sharding 構成の注意点

- RAC構成のShardを作成する場合
  - GDSCTL の “create shard” および “deploy”による構築に未対応
  - 手動でRAC DBを作成し、Sharding構成に手動追加(add shard)

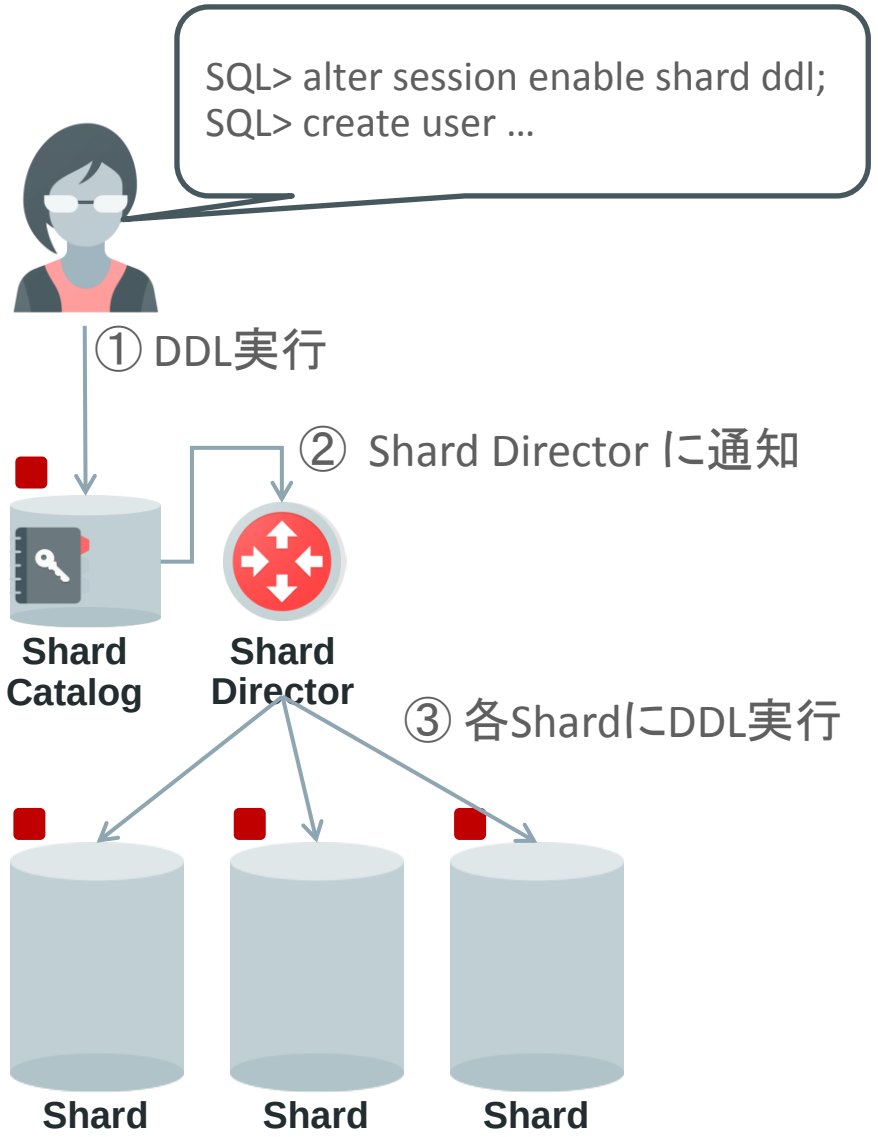
Agenda

1. 機能概要
2. 環境構築
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

Sharding 構成に対するDDL

- Shard CatalogからDDLを実行する
  - alter session enable shard ddl コマンドを実行しSharding管理コマンドであることを宣言
  - 以降のDDLはShard Directorを経由して各Shardで実行される

- 初期構築で実行する主なDDL
  - アプリケーション用Oracleユーザー作成
  - 表領域
  - ユーザーオブジェクト(テーブルなど)



Shard Catalog – GSMADMIN_INTERNALスキーマ

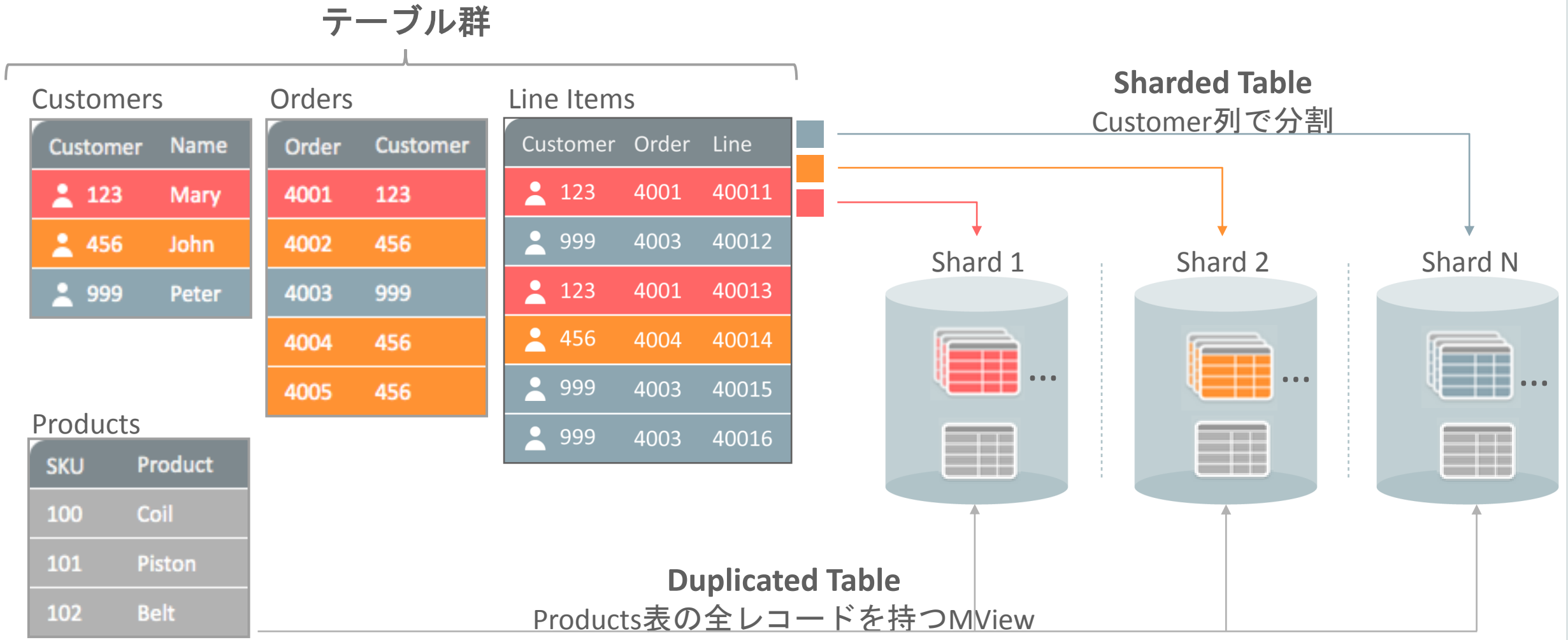
```
SQL>  select object_type, count(*) num_of_objects from all_objects
      where owner = 'GSMADMIN_INTERNAL'  group by object_type order by num_of_objects desc;

OBJECT_TYPE             NUM_OF_OBJECTS
----------------------- --------------
TYPE                                45
INDEX                               41
TABLE                               41
JOB                                 19
SEQUENCE                            16
PACKAGE                              8
VIEW                                 8
PACKAGE BODY                         8
TRIGGER                              5
LIBRARY                              4
TYPE BODY                            3
SYNONYM                              2
RULE SET                             2
PROCEDURE                            2
EVALUATION CONTEXT                   1
CREDENTIAL                           1
```

- Oracle Databaseのデフォルトスキーマ
- Shardingを構成・管理するテーブル群やパッケージを保持

Sharding構成に作成するテーブル

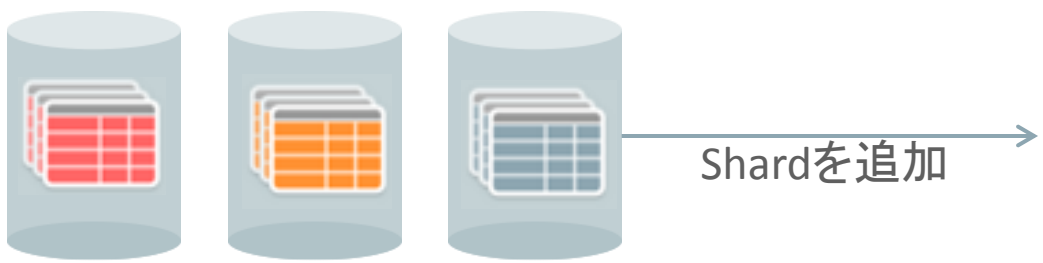
- **Sharded Table**
  - Shardに分散配置するテーブル
    - 明細データなどを想定
  - 全てのSharded Table は共通の分割キー列(Sharding Key)を持つ必要がある(顧客ID列 など)
- **Duplicated Table**
  - 全てのShardで同じデータを持つテーブル
    - マスタ表などを想定
  - Shard Catalogにテーブル実体を保持し、Shardには実体に対するマテリアライズド・ビューを作成する(デフォルトのリフレッシュ間隔は60秒)
- 明細とマスタのJOINが単一のShard内で完結

Shardingスキーマの例



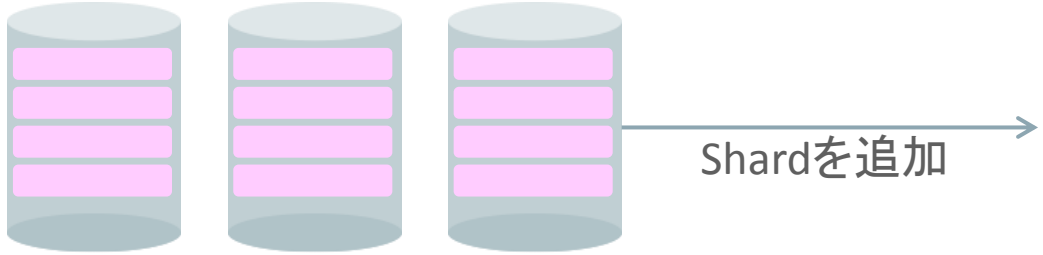
Shardの領域管理

- Shardingに求められる拡張性

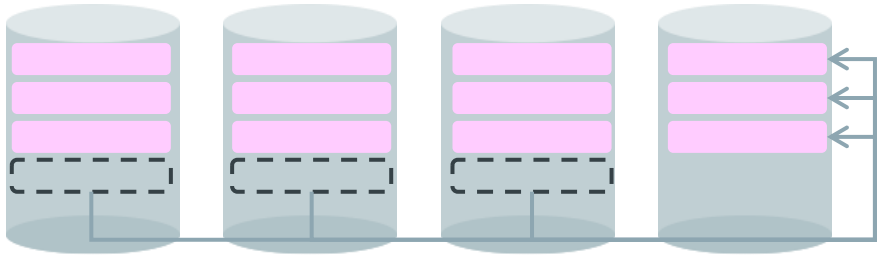


・スケールさせるために、データを均等に再分散したい
・効率的な方法方は？

- “Chunk”という領域管理の考え方



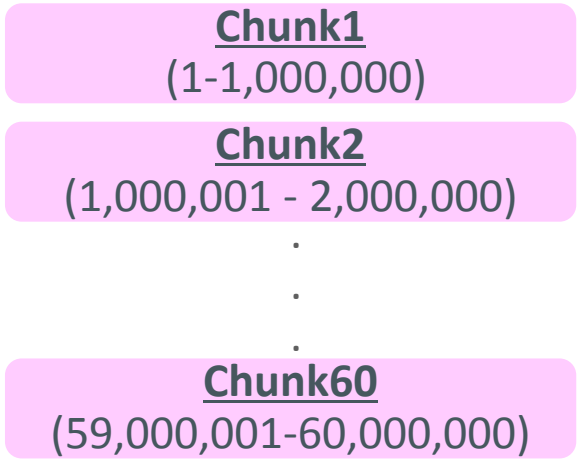
初期構成時にShardを複数の
領域(**Chunk**)に分割して管理



最小限のChunkの移動によって
再分散を完了

Chunk

- Chunkにはハッシュ値の範囲が均等に割り当てられる
  - イメージ

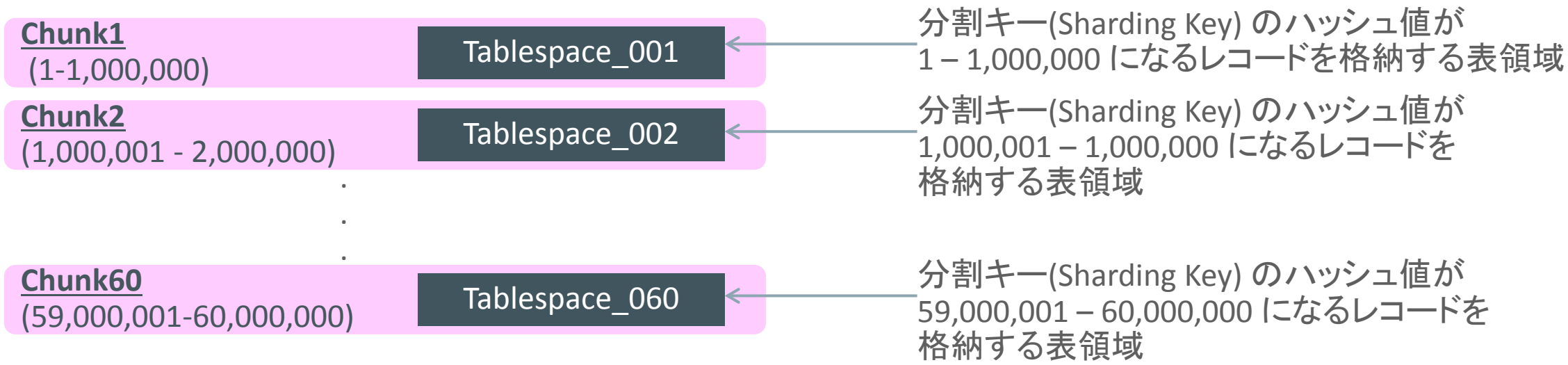


- Chunkの数
  - デフォルト: 1Shardにつき120
  - Sharding構成時に任意の数に変更可能

※10,000,000単位のハッシュ値の範囲はイメージであり、実際の値とは異なります

Chunk と 表領域

- Chunkの実体は 分散配置するテーブル(Sharded Table)用の表領域



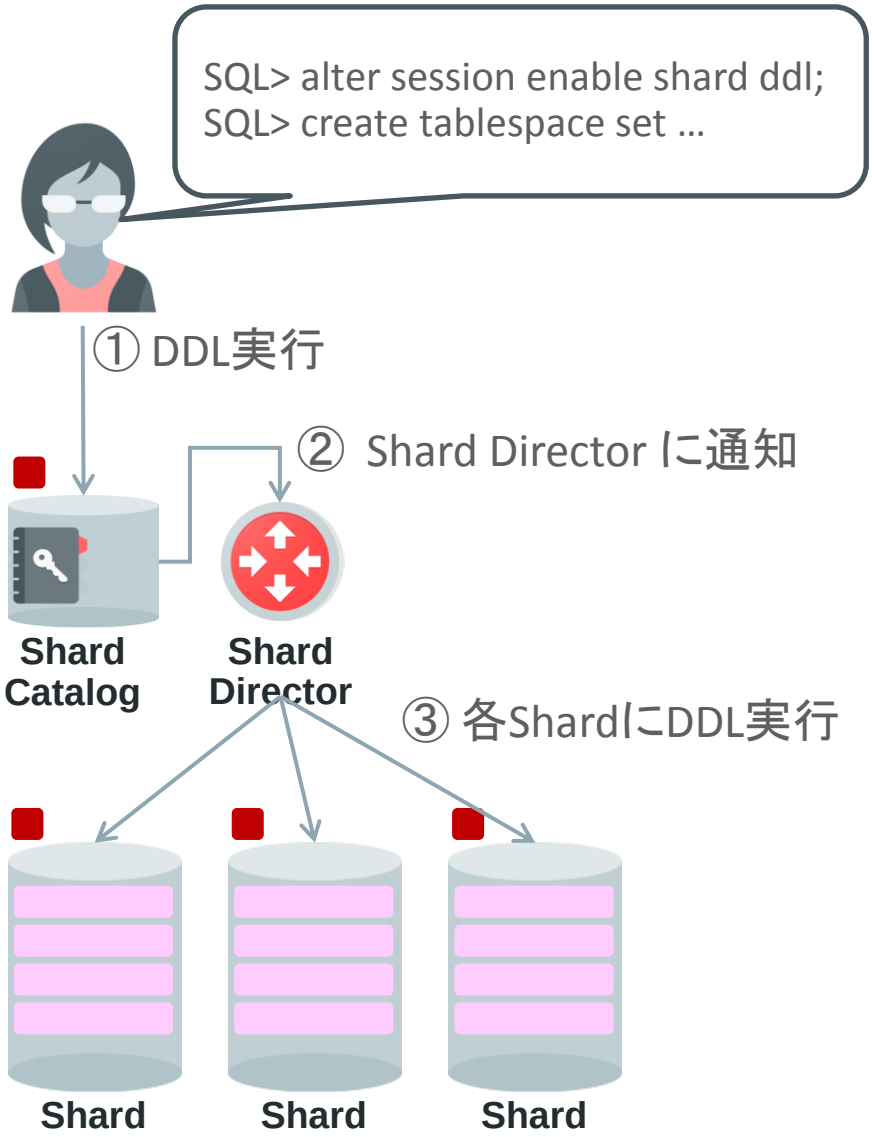
Tablespace Set

- Chunkの実体として作成される表領域群を”Tablespace Set”と呼ぶ
- DDL文

CREATE TABLESPACE SET TSP_SET_1

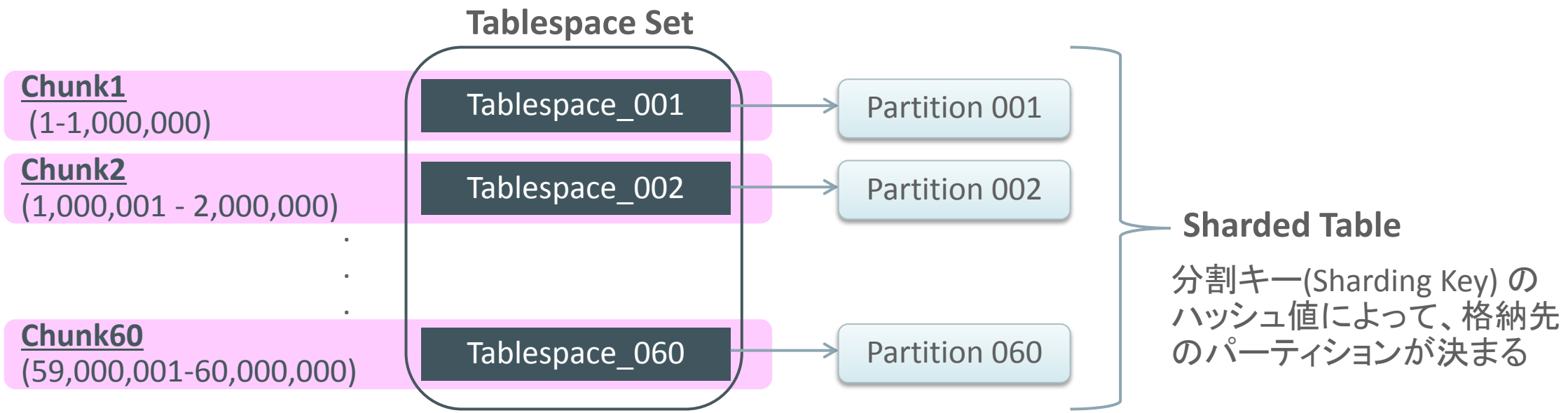
```
using template (
datafile size 100m
autoextend on
next 10M
maxsize unlimited
extent management local
segment space
management auto);
```

- 従来のCreate Tablespaceの構文
- db_create_file_destでの設定場所にファイル作成

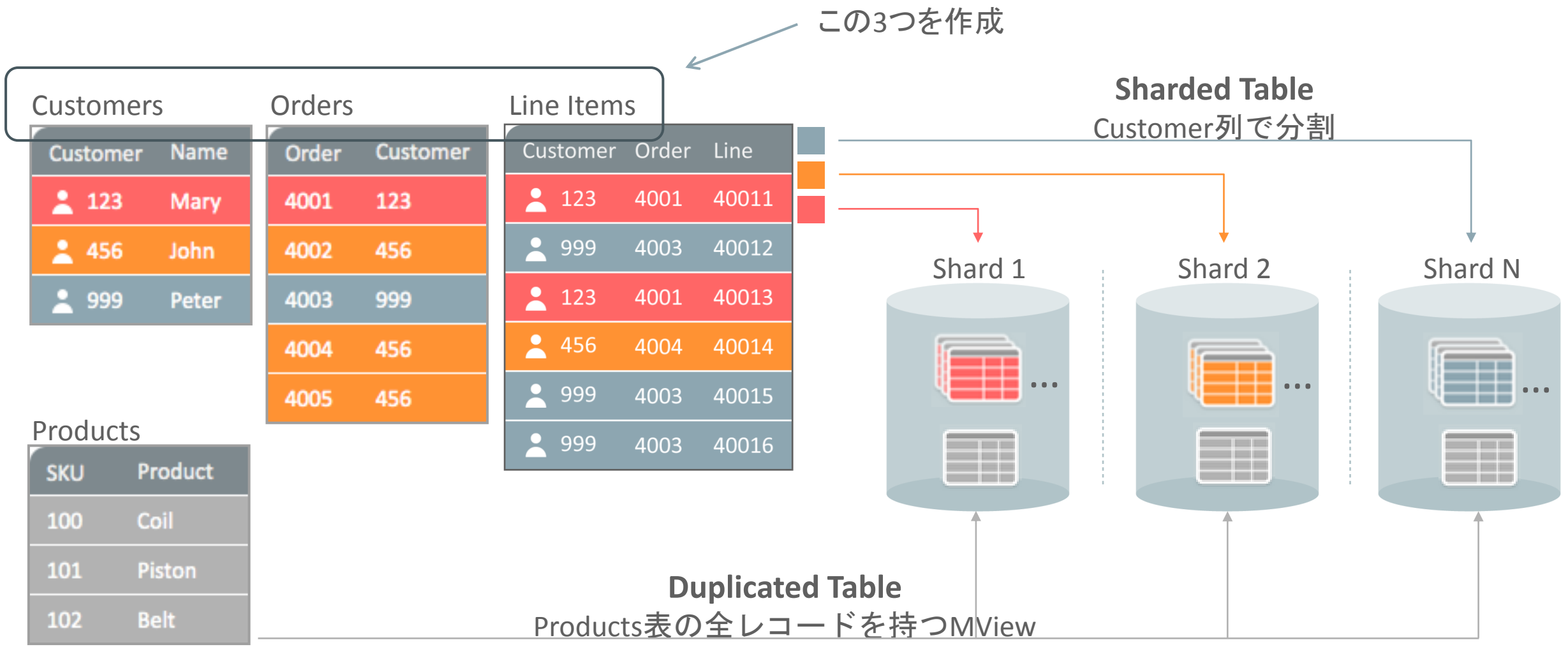


Chunk と Tablespace Set と Sharded Table

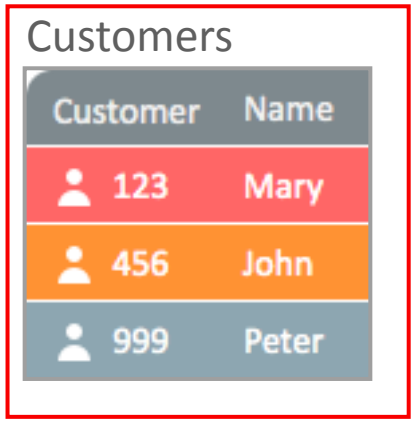
- Chunkに関連付けられる表領域はSharded Tableのパーティションとして使用される
- Sharded Table = Shard をまたがるパーティション表



Shardingスキーマの例(再掲)



Sharded Tableの作成




```
CREATE SHARDED TABLE Customers
( Customer NUMBER NOT NULL,
  Name VARCHAR2(60),
    …
  CONSTRAINT pk_customers
PRIMARY KEY(Customer)
) PARTITION BY CONSISTENT HASH (Customer)
PARTITIONS AUTO
TABLESPACE SET TSP_SET_1;
```

→ Sharded Table の作成

列定義

主キー制約の指定

→ 最初に作成した Sharded Tableの
パーティション・キーがSharding Keyになる

指定したTablespace Set に対して自動分散される

Sharded Tableの作成

Customers

Customer	Name
123	Mary
456	John
999	Peter

Orders

Order	Customer
4001	123
4002	456
4003	999
4004	456
4005	456

```
CREATE SHARDED TABLE Orders (          ⟶ Sharded Table の作成
  Order   NUMBER,
  Customer    NUMBER,                  ⎫ 列定義
   …                                   ⎭
  CONSTRAINT pk_orders  PRIMARY KEY (Customer, Order),   ⟶ 主キー制約の指定
  CONSTRAINT fk_orders_parent
    FOREIGN KEY (Customer) REFERENCES  Customers(Customer)   ⎫ Sharding Key に対する
                                                             ⎭ 外部キー制約の指定
) PARTITION BY REFERENCE (fk_orders_parent) ;   ⟶ 外部キー制約で
                                                  リファレンス・パーティションを定義
```

Sharded Tableの作成

Customers

Customer	Name
123	Mary
456	John
999	Peter

Orders

Order	Customer
4001	123
4002	456
4003	999
4004	456
4005	456

Line Items

Customer	Order	Line
123	4001	40011
999	4003	40012
123	4001	40013
456	4004	40014
999	4003	40015
999	4003	40016

```
CREATE SHARDED TABLE LineItems (        ⟶ Sharded Table の作成
  Line    NUMBER,
  Order NUMBER,
  Customer    NUMBER,
   …                                    ⟵ 列定義
  CONSTRAINT pk_lineitems  PRIMARY KEY (Customer, Order, Line),   ⟶ 主キー制約の指定
  CONSTRAINT fk_lineitems_parent
    FOREIGN KEY (Customer) REFERENCES  Customers(Customer)        ⟵ Sharding Key に対する外部キー制約の指定
) PARTITION BY REFERENCE (fk_lineitems_parent) ;   ⟶ 外部キー制約でリファレンス・パーティションを定義
```

Sharded Table の作成

- Sharding Key列への外部キーを持たない表も作成可能
- 明示的にSharding Key列を持つ親表を指定

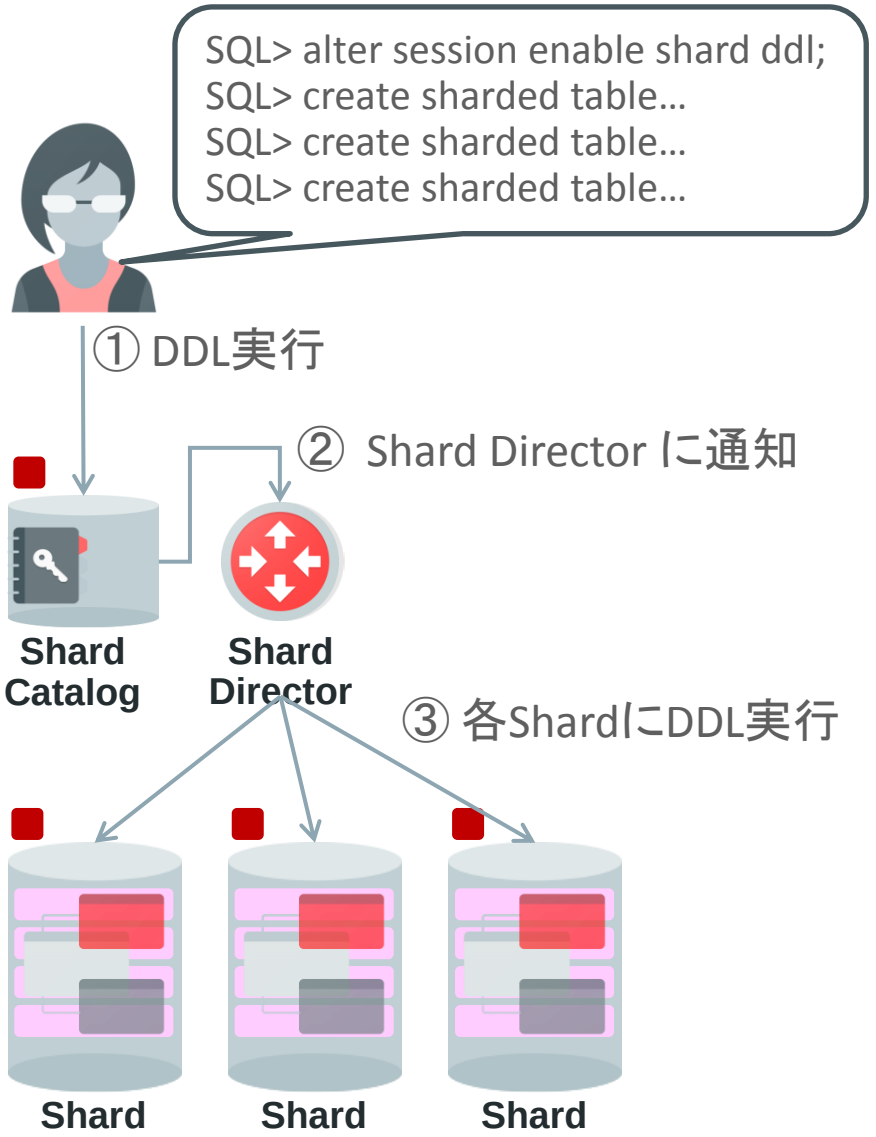
```
CREATE SHARDED TABLE Test (
Customer NUMBER,
Val char(10)
)
PARENT Customers
PARTITION BY CONSISTENT HASH (Customer)
TABLESPACE SET TSP_SET_1;
```

→ 親表の指定

} パーティションの設定

Sharded Tableの作成

- Shard Catalog に対してDDLを実行することで適切に作成される



Shardingスキーマの例(再掲)

Customers

Customer	Name
123	Mary
456	John
999	Peter

Orders

Order	Customer
4001	123
4002	456
4003	999
4004	456
4005	456

Line Items

Customer	Order	Line
123	4001	40011
999	4003	40012
123	4001	40013
456	4004	40014
999	4003	40015
999	4003	40016

Sharded Table
Customer列で分割

Shard 1 | Shard 2 | Shard N

Products

SKU	Product
100	Coil
101	Piston
102	Belt

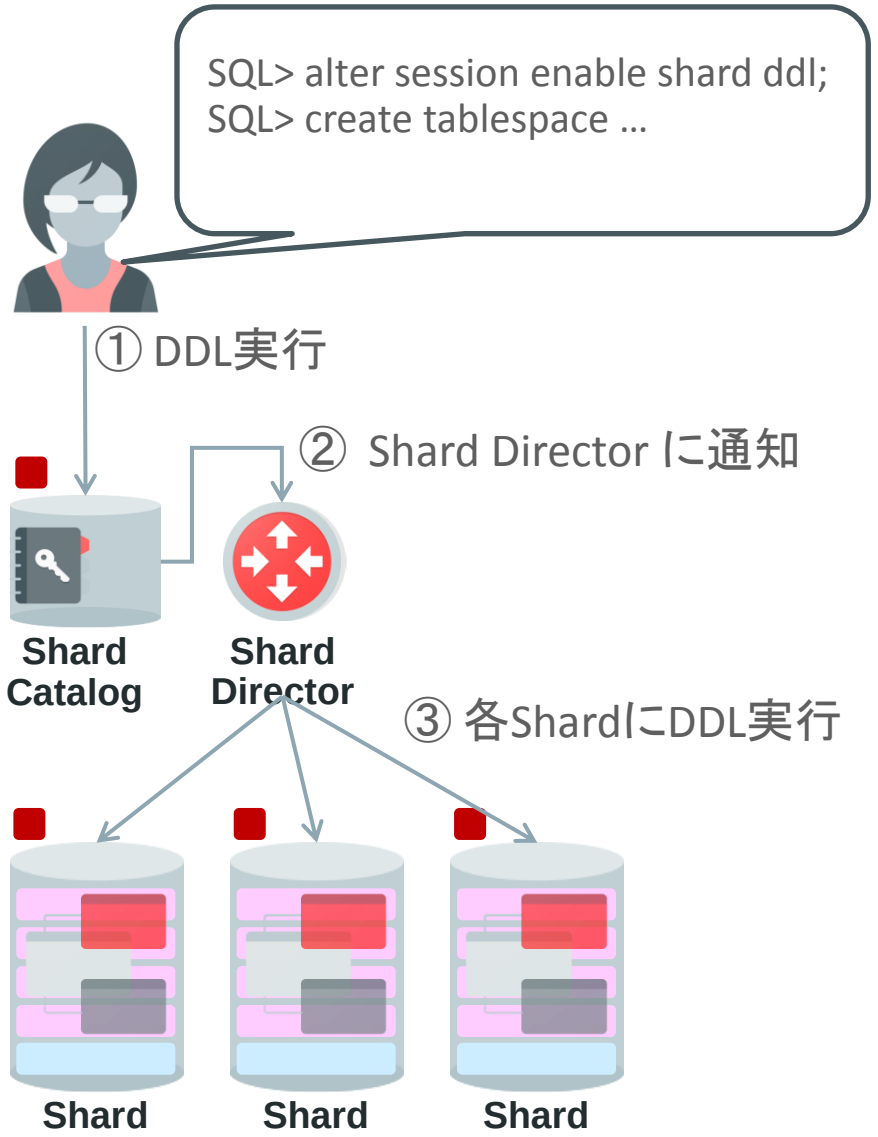
Duplicated Tableを作成

Duplicated Table
Products表の全レコードを持つMView

表領域の作成

- Duplicated Table は分散配置は行わないため、従来の表領域を作成

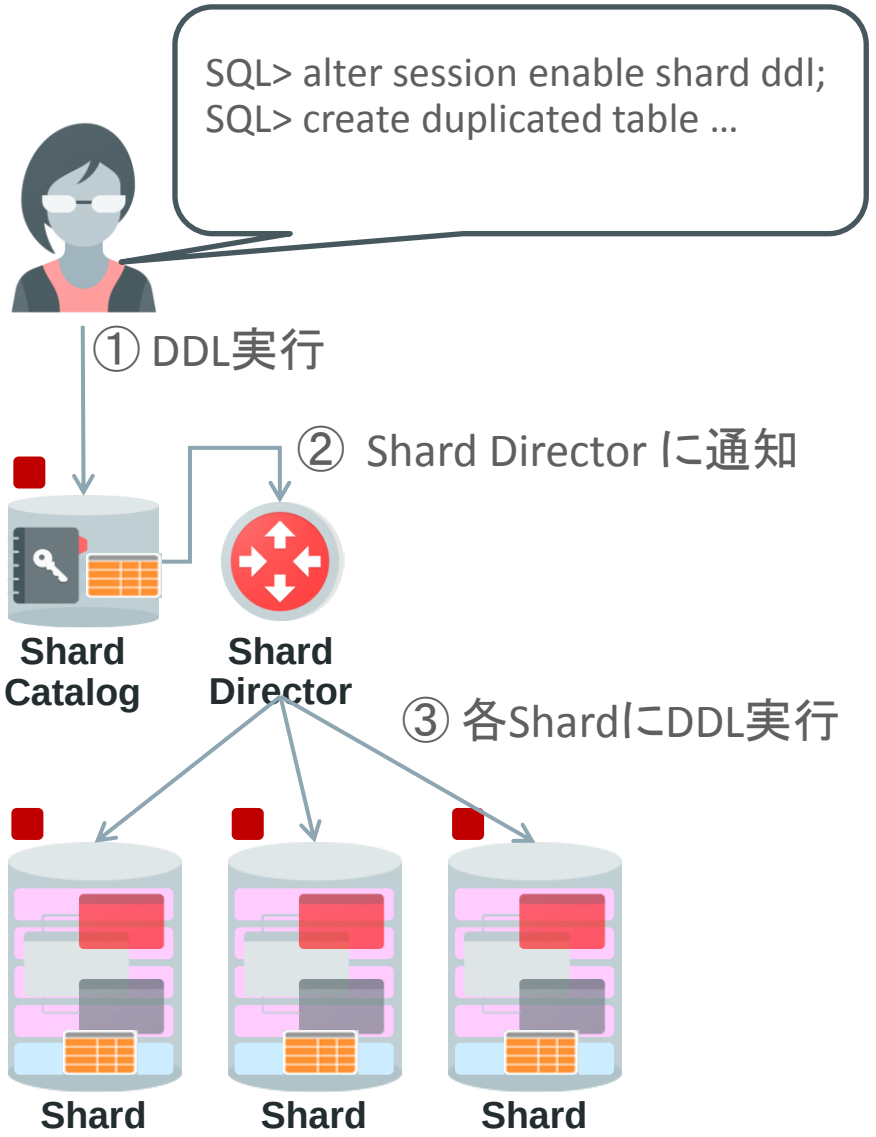
```
CREATE TABLESPACE products_tsp
datafile size 100m
autoextend on
next 10M
maxsize unlimited
extent management local
uniform size 1m;
```



Duplicated Tableの作成

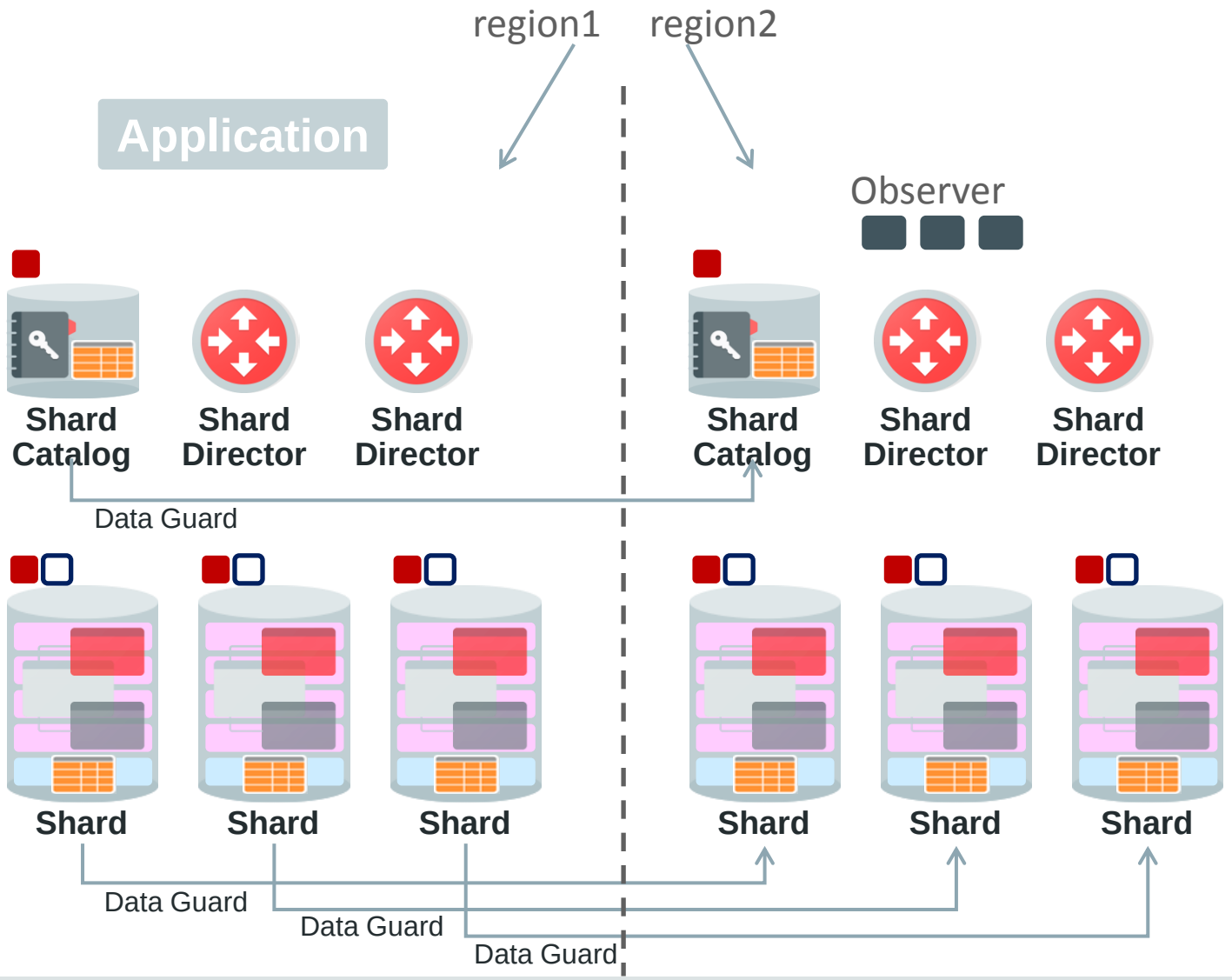
- Duplicated Tableの実体はShard Catalog上に作成される
- 各ShardにはShard Catalogに対するマテリアライズド・ビューが作成される

```
CREATE DUPLICATED TABLE Products
 (
   ProductId  INTEGER GENERATED BY DEFAULT AS
IDENTITY PRIMARY KEY,
   Name       VARCHAR2(128),
   DescrUri   VARCHAR2(128),
   LastPrice  NUMBER(19,4)
 ) TABLESPACE products_tsp;
```



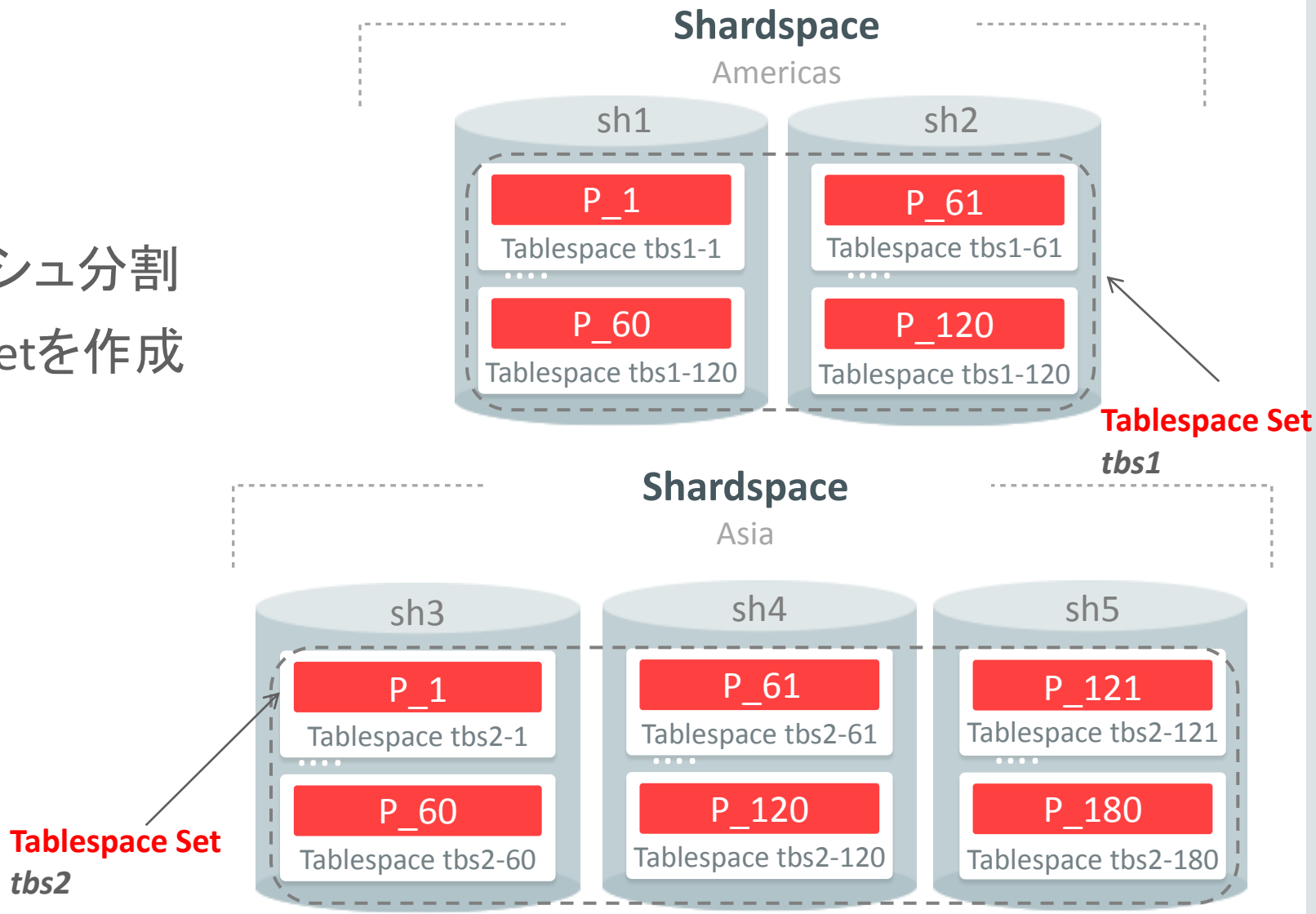
構成イメージ

- Shardingに必要なコンポーネントを構成
- 表領域・テーブルを作成



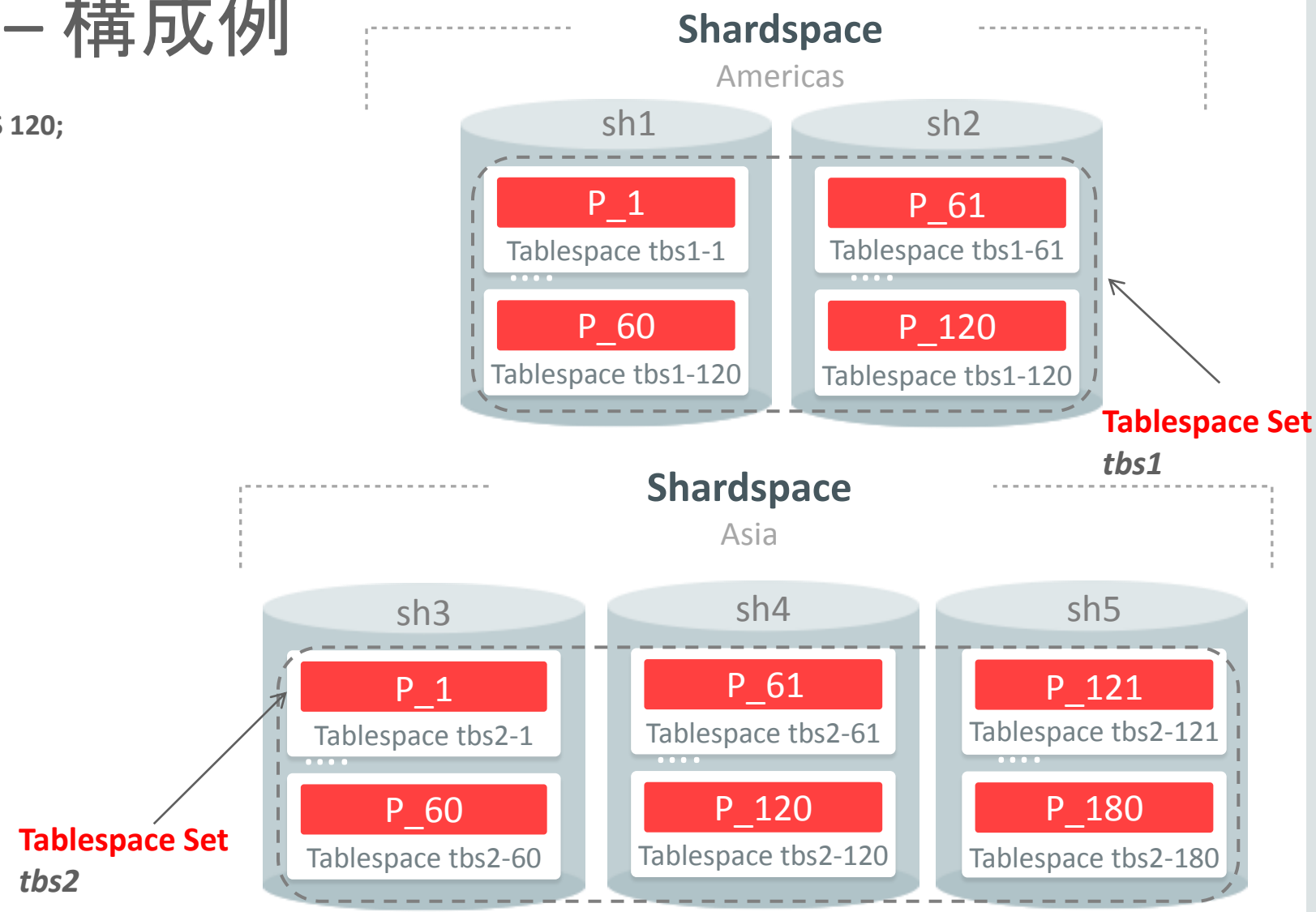
Composite Sharding

- レンジ/リストでの分割単位で”Shardspace”を定義
- Shardspace内のShardでハッシュ分割
- Shardspace毎にTablespace Setを作成
- Sharded Tableには2つのSharding Keyが必要



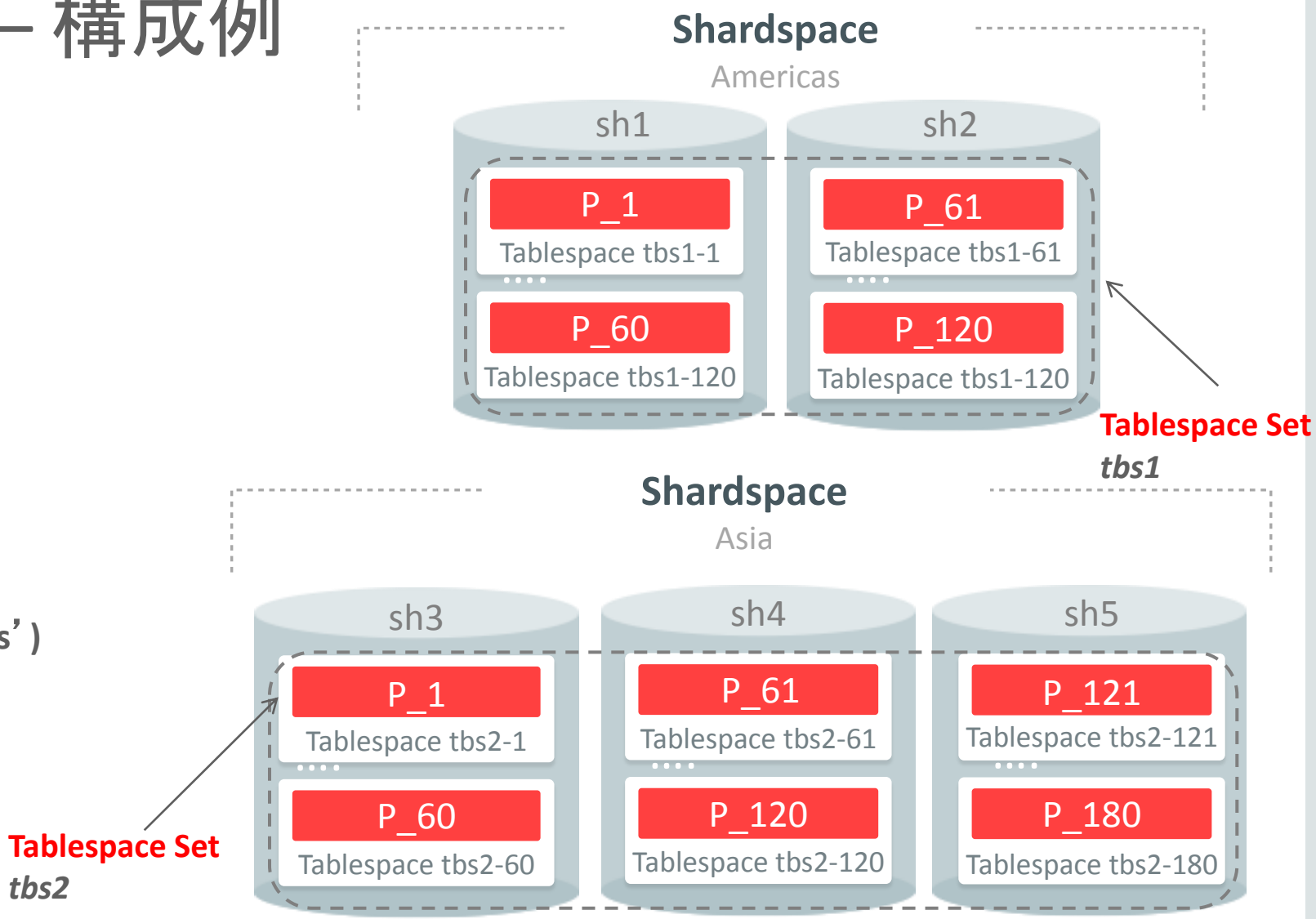
Composite Sharding – 構成例

```
ADD SHARDSPACE –SHARDSPACE Americas –CHUNKS 120;

CREATE SHARD –SHARDSPACE Americas
  -DESTINATION <host1> -CREDENTIAL oracle_cred;
CREATE SHARD –SHARDSPACE Americas
  -DESTINATION <host2> -CREDENTIAL oracle_cred;
DEPLOY;
CREATE TABLESPACE SET tbs1 IN
      SHARDSPACE Americas;

ADD SHARDSPACE –SHARDSPACE Asia –CHUNKS 180;

CREATE SHARD –SHARDSPACE Asia
  -DESTINATION <host3> -CREDENTIAL oracle_cred;
CREATE SHARD –SHARDSPACE Asia
  -DESTINATION <host4> -CREDENTIAL oracle_cred;
CREATE SHARD –SHARDSPACE Asia
  -DESTINATION <host5> -CREDENTIAL oracle_cred;
DEPLOY;
CREATE TABLESPACE SET tbs2 IN
      SHARDSPACE Asia;
```



Composite Sharding – 構成例

```
CREATE SHARDED TABLE customers
( CustId VARCHAR2(60) NOT NULL,
  ..
  Geo       VARCHAR2(8),
  CONSTRAINT cust_pk
    PRIMARY KEY(CustId)
)
PARTITIONSET BY LIST (Geo)
PARTITION BY CONSISTENT HASH (CustId)
PARTITIONS AUTO
(PARTITIONSET Americas VALUES ( 'Americas' )
     TABLESPACE SET tbs1,
 PARTITIONSET Asia VALUES ( 'Asia' )
     TABLESPACE SET tbs2) ;
```

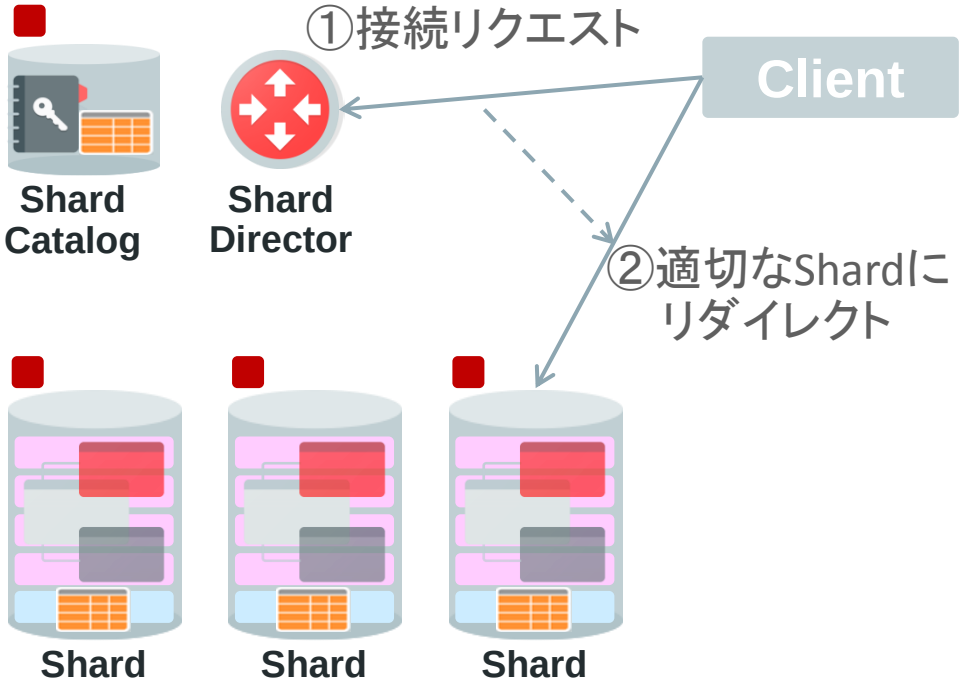


Agenda

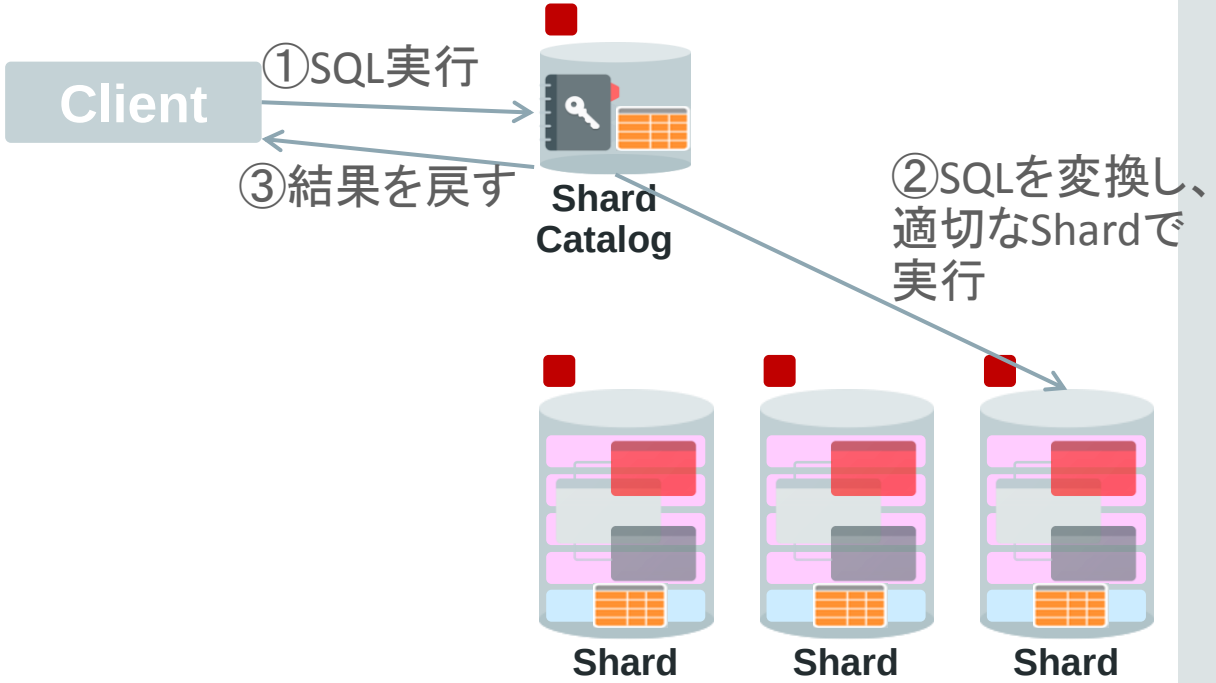
1. 機能概要
2. 環境構築
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

アプリケーション(Oracle Client)からShardingへの接続

- Direct Routing
  - Sharding Key を指定してShard Directorからのリダイレクトで適切なShardに直接接続する

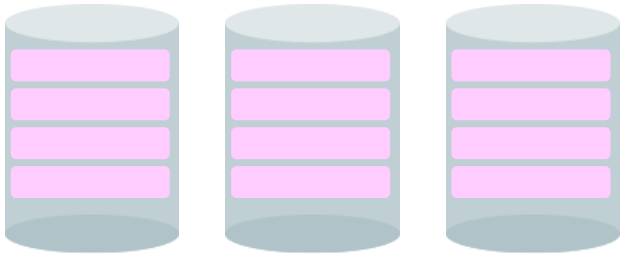


- Proxy Routing
  - ClientからShard CatalogにSQL実行
  - Shard Catalogは、SQLから適切なShardを判断し実行、結果をClientに戻す



Direct Routing

- Shardingのデータ分散のおさらい



Shard内は複数のChunkに領域分割されている

- Sharding keyが分かれば、データ格納先のChunkが分かる
- Chunkが分かればデータ格納先のShardが分かる

Chunk1 (1-1,000,000) → Tablespace_001 → Partition 001

Chunk2 (1,000,001 - 2,000,000) → Tablespace_002 → Partition 002

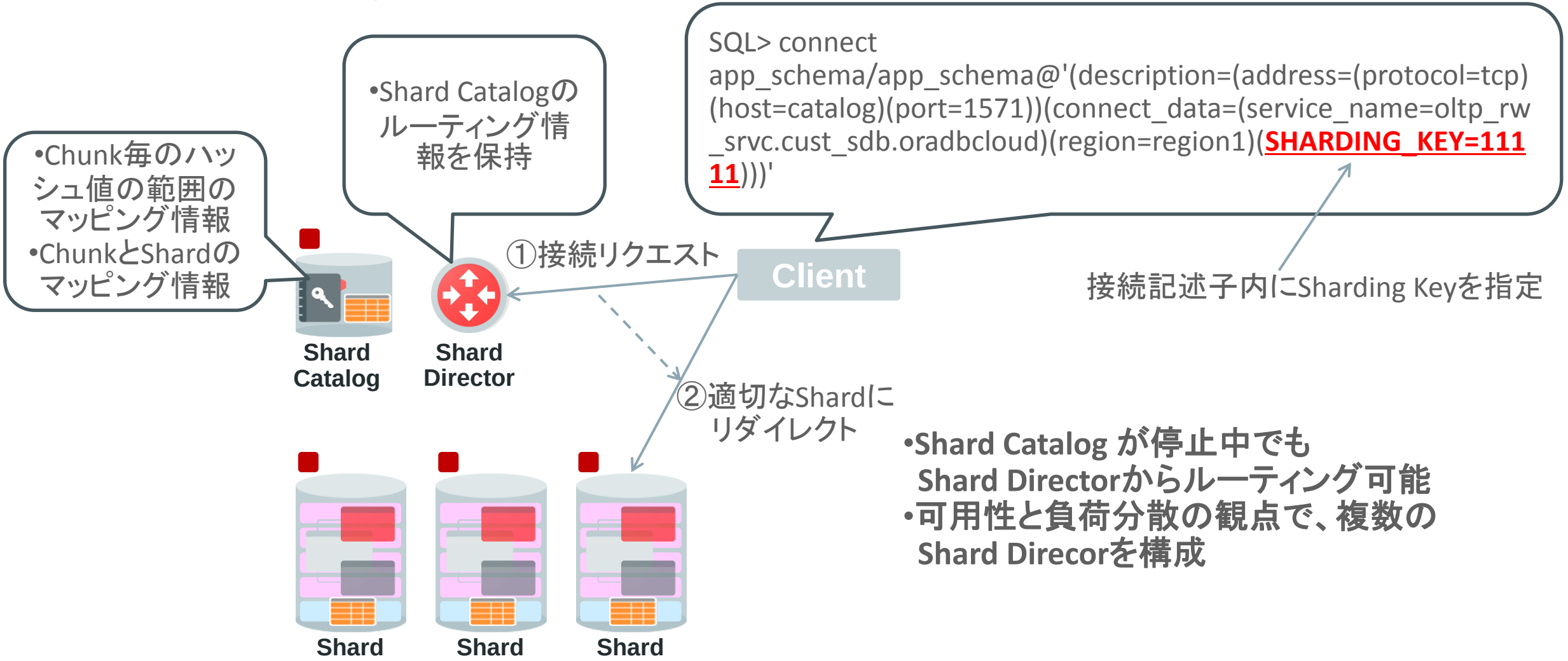
.
.
.

Chunk60 (59,000,001-60,000,000) → Tablespace_060 → Partition 060

Sharded Table

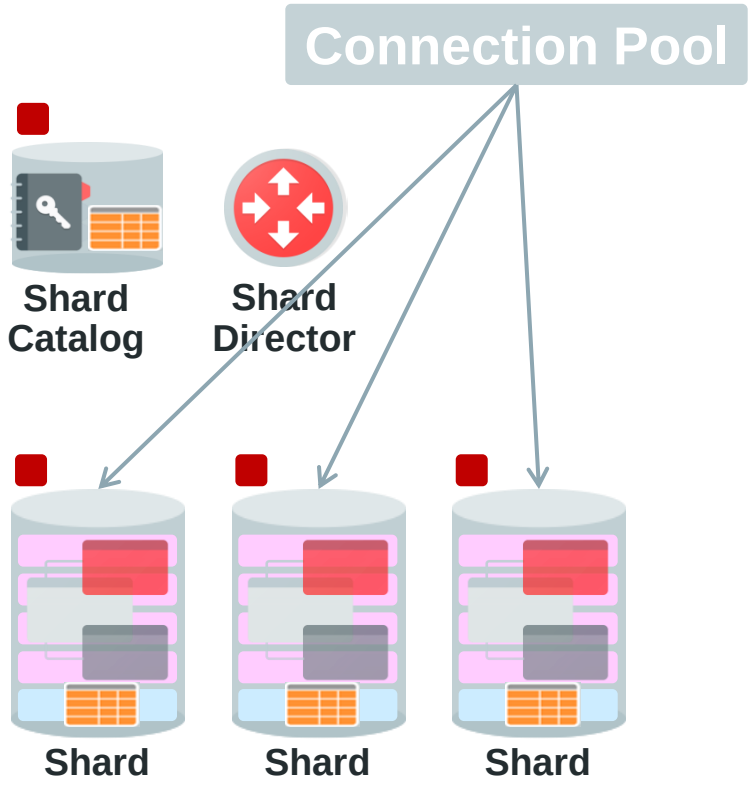
分割キー(Sharding Key) のハッシュ値によって、格納先のパーティションが決まる

Direct Routing



Direct Routingのコネクションプール対応

- Oracle Database 12.2 の JDBC, UCP, OCI, ODP.NET でShardingに対応



Direct Routingのコネクションプール対応(JDBC/UCP)

- Sharding対応のAPIを提供
  - Sharding Keyの生成
  - Sharding Keyを指定した接続の取得

```
OracleDataSource datasource =     new OracleDataSource();
                              OR
PoolDataSource datasource =  PoolDataSourceFactory.getPoolDataSource();
```

```
OracleShardingKey keyMaryEmail =
  datasource.createShardingKeyBuilder()
    .subkey("mary.smith@xyz.com", OracleType.VARCHAR2)
    .build();

OracleShardingKey superKeyGoldClass =
  datasource.createShardingKeyBuilder()
    .subkey("GOLD", OracleType.VARCHAR2)
    .build();
```

Composit Sharding の場合に使用する上位のSharding Key

Direct Routingのコネクションプール対応(JDBC/UCP)

- **Oracle JDBC driver**

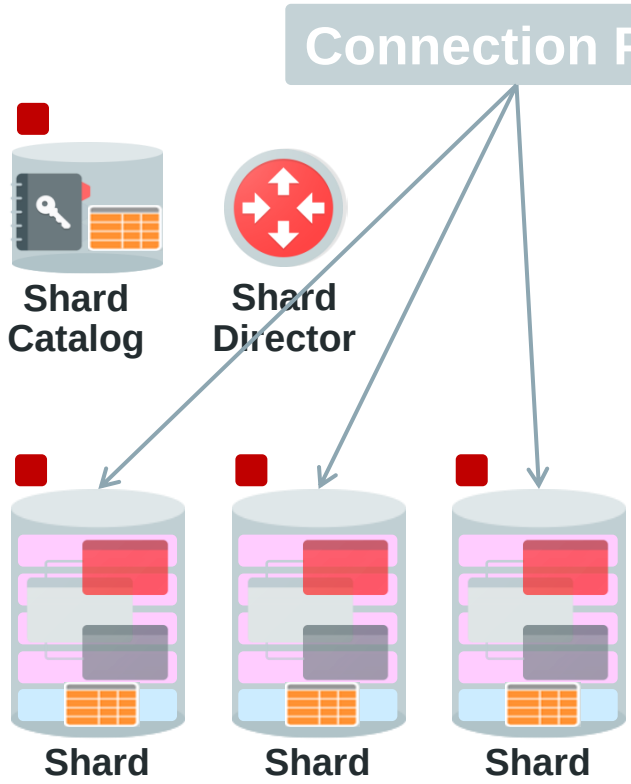
```
OracleDataSource ods =
    new OracleDataSource();

// コネクションプロパティーの設定
ods.setURL(DB_URL);
ods.setUser("hr");
ods.setPassword("*****");
// Get an Oracle JDBC connection for a shard
Connection conn =        ods.createConnectionBuilder()
    .shardingKey(shardingKey)
    .build();
```

- **Oracle Universal Connection Pool (UCP)**

```
PoolDataSource pds =
    PoolDataSourceFactory.getPoolDataSource();
// コネクションプールプロパティーの設定
    pds.setURL(DB_URL);
    pds.setUser("hr");
    pds.setPassword("*****");
    pds.setInitialPoolSize(10);
    pds.setMinPoolSize(20);
    pds.setMaxPoolSize(30);
// Get an UCP connection for a shard
Connection conn =
    pds.createConnectionBuilder()
    .shardingKey(shardingKey)
    .build();
```

Direct Routingのコネクションプール対応(JDBC/UCP)

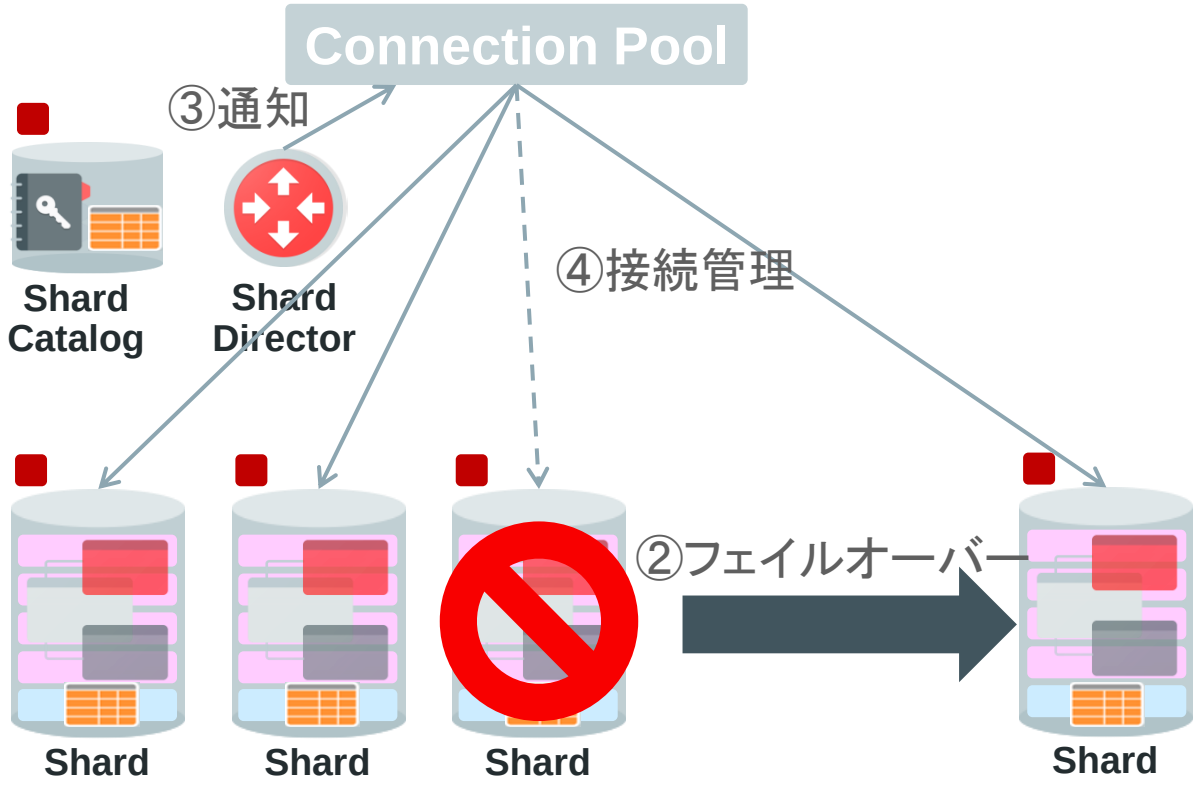
- 接続初期化時に、Chunk/Shardのマッピング情報を取得し、コネクションプール単体でのルーティングを可能にする



•Shard Catalog とShard Directorが停止中でもルーティング可能

Direct Routingのコネクションプール対応(JDBC/UCP)

- Shard Directorと連携し、障害時切り替え・構成変更に追従

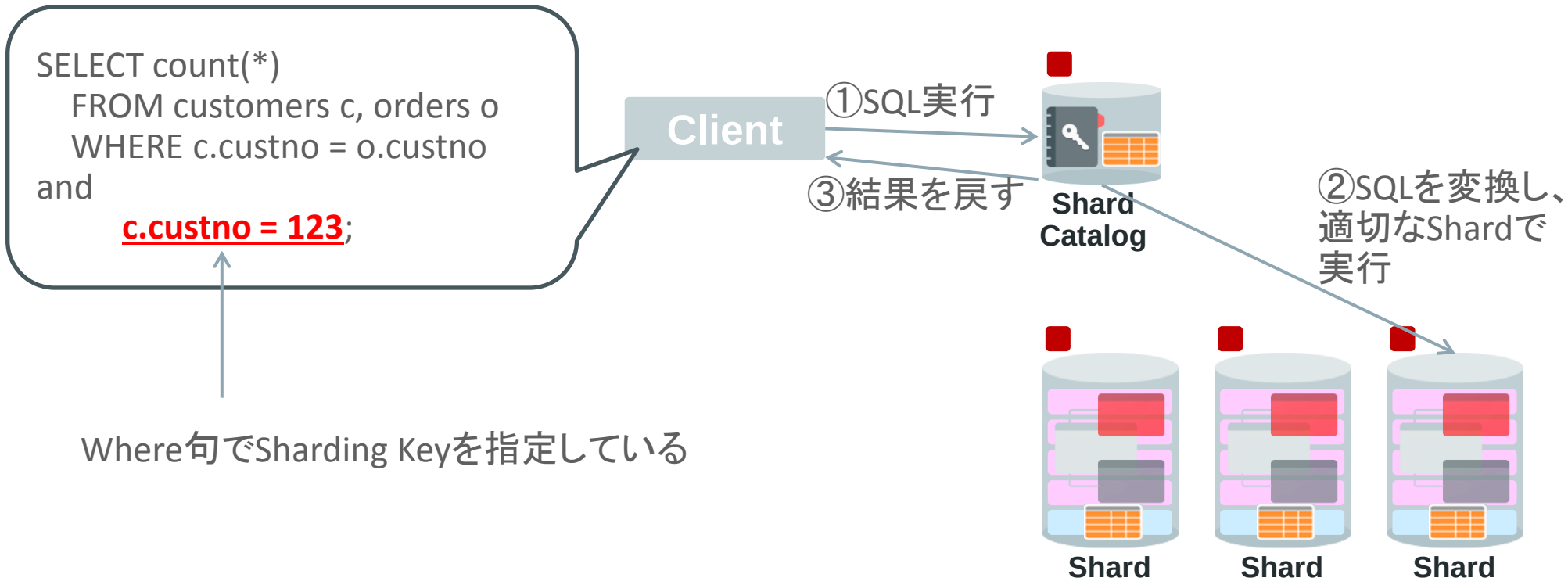


Direct Routingの注意点

- 接続時に使用したSharding Keyを使用したSQLを実行するようにアプリケーションを記述する
- トランザクションはShard内で完結する必要がある

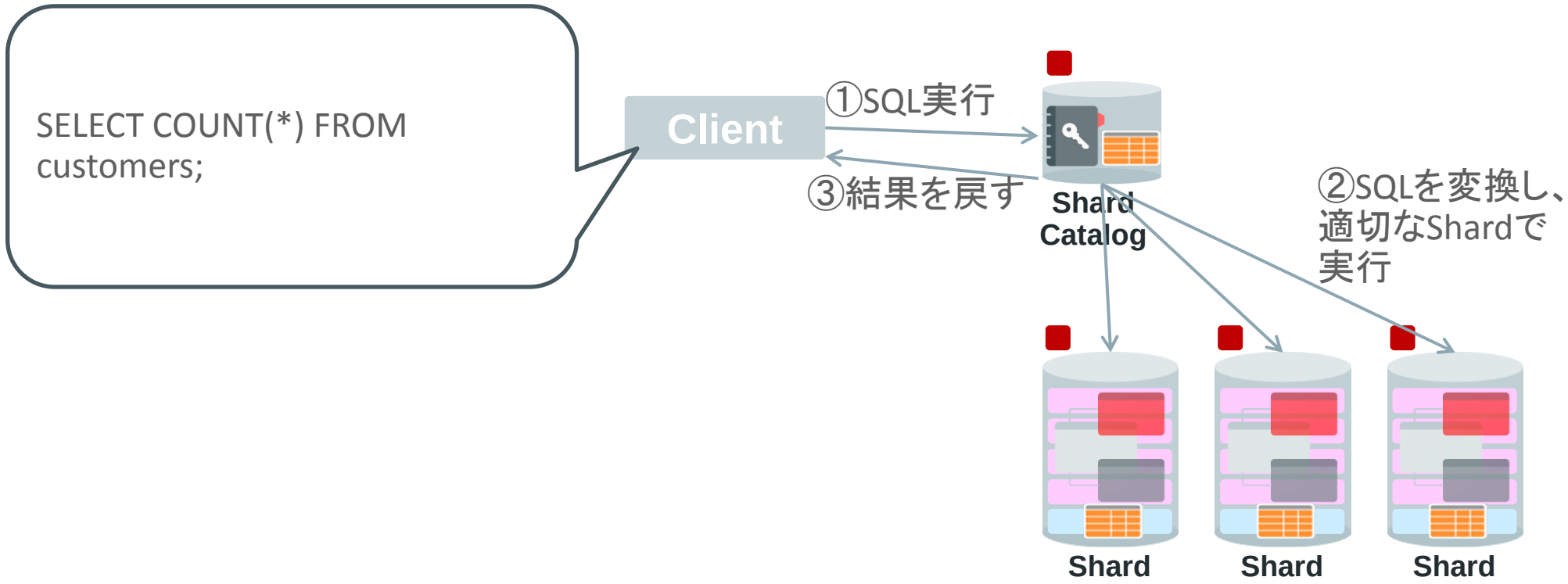
Proxy Routing

- 単一のShardで処理されるケース
  - SELECT, INSERT, UPDATE, DELETEに対応



Proxy Routing

- 複数のShardで処理されるケース(Multi-Shard Query)
  - SELECTのみ対応



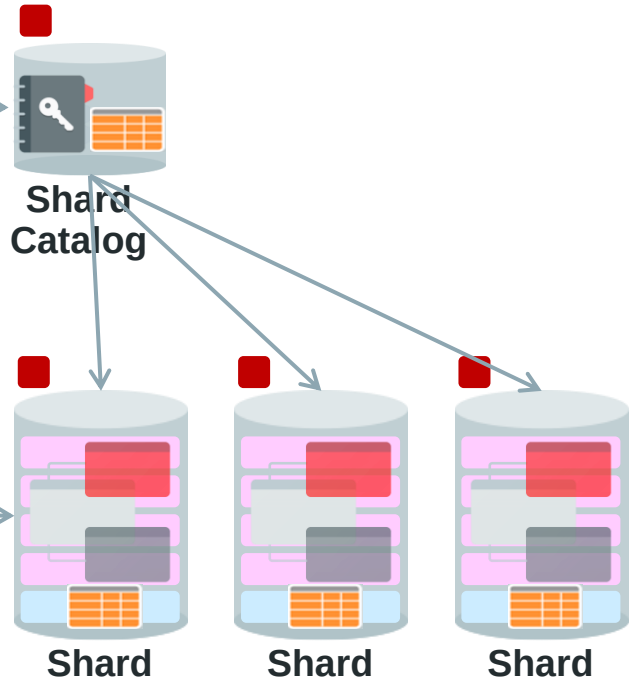
Proxy Routing

- 複数のShardで処理されるケース(Multi-Shard Query)
  - Shard Catalog内の動作

Oracle Shardingの処理方式

パターン1

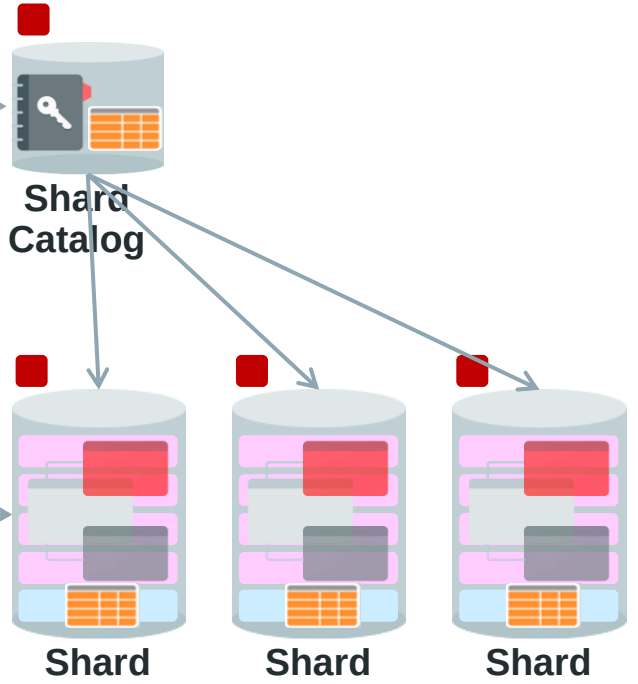
取得したデータを
Shard Catalog上
で集計



各Shardから
テーブルの
全データを
取得する

パターン2

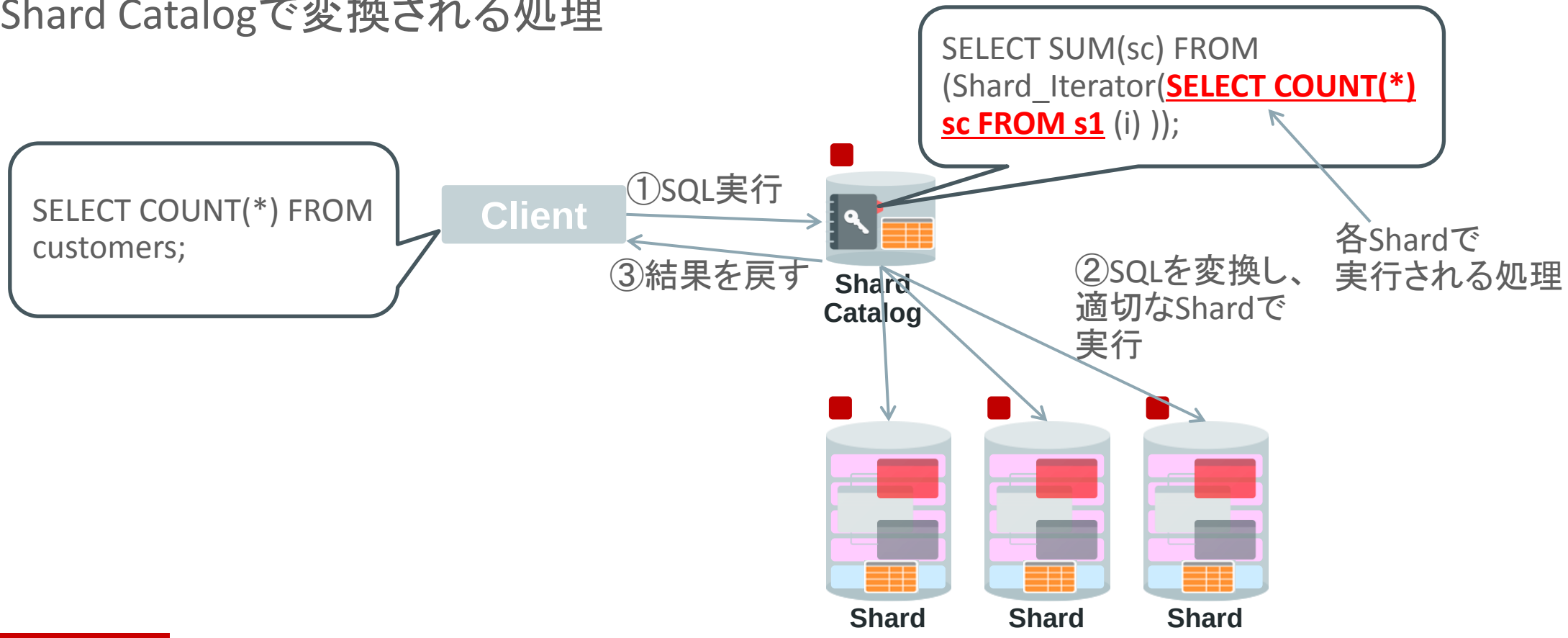
取得したデータを
Shard Catalog上
で集計



Shard内で**できる**
集計をした上で
取得する

Proxy Routing

- 複数のShardで処理されるケース(Multi-Shard Query)
  - Shard Catalogで変換される処理



Proxy Routing 注意点

- Shard CatalogがSPOFなるため、Data Guard化/RAC化を推奨
- Shard Catalogがパフォーマンス・ボトルネックになる可能性
  - ワークロードの大半はDirect Routingで行い、
    バッチ・レポーティング等の一部の処理をProxy Routingにする
- Multi-Shard Queryの制限事項
  - Composite Shardingでは非対応
  - Sharding key 以外を使用したSemi-join (EXISTS) 非対応
  - Anti-join (NOT EXISTS) 未対応
  - etc.

  ※ マニュアルをご参照ください

Agenda

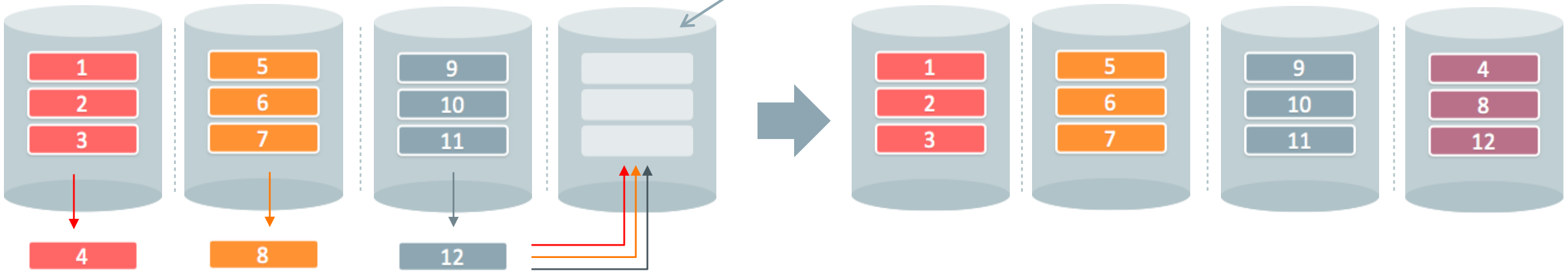
1. 機能概要
2. 環境構築
3. スキーマ作成
4. ルーティング
5. ライフサイクル管理

ライフサイクル管理

- Shardの追加・削除が可能
- Chunkの移動、分割が可能
- Shard Catalog, 各Shardでバックアップを実施
- opatchauto による一括でのパッチ適用が可能
  - パッチレベルの異なるSharding構成も可能

データ再配置の自動化

シングル・インスタンスの場合
ソフトウェアインストールのみ or DB作成済みの状態から追加可能
RACの場合
作成済みのRAC DBのみ追加可能



- Shardの追加/削除、データやワークロードの偏りに対して、自動/手動でデータを再配置
- RMAN増分バックアップとトランスポータブル表領域(TTS)のテクノロジーを応用

Chunk Split

- Chunkを2つに分割する
  - ハッシュ値のレンジの中間で分割
- Chunkを分割するケース
  - Chunkの肥大化
  - 特定のChunkがhot spotになっている
  - Chunkの数を増やしたい場合
    - Sharding構成後にChunkの追加はできないため、splitで対応

Shardingが関連する主なマニュアル

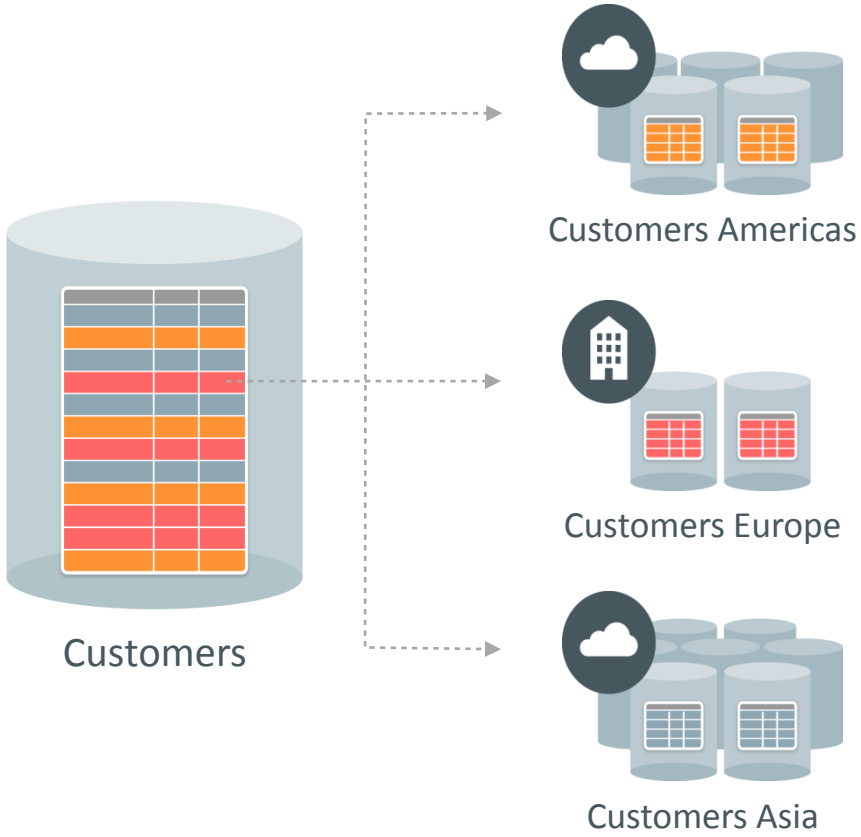
- 管理者ガイド
  - 第VII部 シャード・データベースの管理
  - http://docs.oracle.com/cd/E82638_01/ADMIN/shard_part.htm#GUID-DE8AF949-9F68-4C0C-9B5A-45401EF3C3D1
- Database Global Data Services Concepts and Administration Guide
  - http://docs.oracle.com/cd/E82638_01/GSMUG/toc.htm
- Data Guard概要および管理
  - http://docs.oracle.com/cd/E82638_01/SBYDB/toc.htm
- Universal Connection Pool Developer's Guide
  - 10 Shared Pool for Sharded Databases
  - http://docs.oracle.com/cd/E82638_01/JJUCP/ucp-database-sharding-support.htm#JJUCP-GUID-12685D3A-F083-433A-90DF-C5533009B841
- Universal Connection Pool Java API Reference (Javadoc)
  - http://docs.oracle.com/cd/E82638_01/JJUAR/toc.htm

Shardingが関連する主なマニュアル –cont.

- Database JDBC Developer's Guide
  - 24 JDBC Support for Database Sharding
  - http://docs.oracle.com/cd/E82638_01/JJDBC/database-sharding.htm#JJDBC-GUID-1D7795CA-79DC-452B-9FCC-0EF430F87461
- JDBC Java API Reference (Javadoc)
  - http://docs.oracle.com/cd/E82638_01/JAJDB/toc.htm
- Call Interface Programmer's Guide
  - http://docs.oracle.com/cd/E82638_01/LNOCI/toc.htm
- 新機能ガイド
  - シャーディング
  - http://docs.oracle.com/cd/E82638_01/NEWFT/GUID-6213395A-CFDD-4D42-8970-F8FA062CC4FA.htm#GUID-DF6C0B73-6A6B-485E-B820-72737DA03B2D

New in 12.2 on Oracle Cloud

極めて高い拡張性と信頼性を求めるOLTPアプリのためのデータベース･シャーディング



1つの巨大なデータベースを多数の小さなデータベース(シャード)に分割

- 99%のアプリケーションの要件に対し、アプリケーションの透過性を維持という点でもRACとData Guardが適合する
- グローバル規模のOLTPアプリケーションでは巨大なデータベースをより小さなデータベース・ファームに分割するほうが適する
- ワークロードがファームの特定のシャードに自動的に到達できるようなアプリケーションの設計が必要
- 最大1000シャードに分割された表へのSQL

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®