Data Guard:
Additional Benefits apart from DR

Uwe Hesse
Senior Principal Instructor OU
Oracle Certified Master

http://uhesse.com/about

ORACLE

What is Active Data Guard?

- Option for Oracle Database 11*g* Enterprise Edition
- Used to offload resource-intensive activities from a Production Database to a Standby Database
- Includes 2 features:
  - Real-time query (Main focus of this Presentation)
  - Block Change Tracking on a Physical Standby Database

ORACLE DATABASE 11g

The initial Data Guard Configuration

- Primary & Physical Standby with Data Guard Broker:

```
DGMGRL> show configuration;

Configuration - myconf

  Protection Mode: MaxPerformance
  Databases:
    prima - Primary database
    physt - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

The initial Physical Standby

- Not yet doing Real-Time Query:

```
DGMGRL> show database physt

Database - physt

  Role:              PHYSICAL STANDBY
  Intended State:    APPLY-ON
  Transport Lag:     0 seconds
  Apply Lag:         0 seconds
  Real Time Query:   OFF
  Instance(s):
    physt

Database Status:
SUCCESS
```

Enabling Real-Time Query

```
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE     OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY MOUNTED

SQL> select status from v$managed_standby where process like 'MRP%';

STATUS
------------
APPLYING_LOG

SQL> alter database open;

Database altered.

SQL> select database_role,open_mode from v$database;

DATABASE_ROLE     OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY READ ONLY WITH APPLY
```

Read-Only OPEN without Real-Time Query:

```
DGMGRL> edit database physt set state=apply-off;
Succeeded.
DGMGRL> show database physt;

Database - physt

  Role:              PHYSICAL STANDBY
  Intended State:    APPLY-OFF
  Transport Lag:     0 seconds
  Apply Lag:         0 seconds
  Real Time Query:   OFF
  Instance(s):
    physt

Database Status:
SUCCESS
```

```
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE          OPEN_MODE

---------------- --------------------

PHYSICAL STANDBY READ ONLY
```

Client Connectivity & Active Data Guard

- We need to get our Clients connected appropriately
- That is done via *services*:
  - One for the Primary (*prod*)
  - One for the Standby (*ronly*)
- We take Role Transitions into account with an Event Trigger

Creation of Services for Active Data Guard

- On the Primary:

```
begin
dbms_service.create_service(service_name=>'prod',
                            network_name=>'prod');
dbms_service.start_service(service_name=>'prod');
end;
/

begin
dbms_service.create_service(service_name=>'ronly',
                            network_name=>'ronly');
end;
/
```

- On the Standby:

```
begin
dbms_service.start_service(service_name=>'ronly');
end;
/
```

Creation of the Event Trigger for Service Management

```
create or replace trigger service_management
after startup on database
declare
 vrole varchar(30);
 vopen_mode varchar(30);
begin
 select database_role into vrole from v$database;
 select open_mode into vopen_mode from v$database;
 if vrole = 'PRIMARY' then begin
 dbms_service.start_service ('prod');
 dbms_service.stop_service ('ronly');
 end;
 elsif vrole = 'PHYSICAL STANDBY' then begin
 if vopen_mode like 'READ ONLY%' then
 dbms_service.start_service ('ronly');
 end if;
 dbms_service.stop_service ('prod');
 end;
 end if;
end;
/
```

Local Naming Method for Active Data Guard

- Connect Descriptors in the `tnsnames.ora`:

```
PROD =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = uhesse1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = uhesse2)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = prod)
    )
  )

RONLY =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = uhesse1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = uhesse2)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ronly)
    )
  )
```

Implement TAF for Active Data Guard

- On the Primary (reaches Standby via Redo-Apply):

```
begin
  dbms_service.modify_service
    (service_name=>'prod',
     failover_method => 'BASIC',
     failover_type => 'SELECT',
     failover_retries => 200,
     failover_delay => 1);
end;
/
begin
  dbms_service.modify_service
    (service_name=>'ronly',
     failover_method => 'BASIC',
     failover_type => 'SELECT',
     failover_retries => 200,
     failover_delay => 1);
end;
/
```

Check Session Details with SYS_CONTEXT

- Is my session „indestructible“?

```
set serveroutput on
begin
dbms_output.put_line('USER:               '||sys_context('userenv','session_user'));
dbms_output.put_line('SESSION ID:         '||sys_context('userenv','sid'));
dbms_output.put_line('CURRENT_SCHEMA:     '||sys_context('userenv','current_schema'));
dbms_output.put_line('INSTANCE NAME:      '||sys_context('userenv','instance_name'));
dbms_output.put_line('DATABASE ROLE:      '||sys_context('userenv','database_role'));
dbms_output.put_line('OS USER:            '||sys_context('userenv','os_user'));
dbms_output.put_line('CLIENT IP ADDRESS:  '||sys_context('userenv','ip_address'));
dbms_output.put_line('SERVER HOSTNAME:    '||sys_context('userenv','server_host'));
dbms_output.put_line('CLIENT HOSTNAME:    '||sys_context('userenv','host'));
end;
/
```

- … after a *Role Transistion*? For example after a SWITCHOVER:

```
DGMGRL> switchover to physt;
```

What is Lag in a Data Guard Configuration?

- A Standby Database may lag behind the Primary because of:
  - Insufficient CPU capacity
  - High network latency
  - Limited bandwidth
- Real-Time Query can be configured to return results according to a certain service level.

Checking whether there is a Lag

- For testing purpose, following command creates a Transport Lag artificially:

```
DGMGRL> edit database physt set property logshipping=off;
```

- Now we check for the Lag:

```
SQL > create database link ronly
      connect to system identified by manager using 'ronly';

SQL > select (select current_scn from v$database) as primary_scn,
      (select current_scn from v$database@ronly) as standby_scn
      from dual;

SQL > select to_char(scn_to_timestamp((select current_scn from   v$database)),'hh24:mi:ss')
      as p_time,
      to_char(scn_to_timestamp((select current_scn from v$database@ronly)),'hh24:mi:ss')
      as s_time
      from dual;
```

Difference between Transport Lag and Apply Lag

- *Apply Lag*: Degree to which the data in a Standby Database lags behind the data in the Primary, due to delays in propagating and applying redo to the Standby
- *Transport Lag*: Degree to which the transport of redo to the Standby Database lags behind the generation of redo on the primary database

Setting a Predetermined Service Level for Currency of Standby Queries

- `STANDBY_MAX_DATA_DELAY` session parameter: Specifies a session-specific limit for the amount of time (in seconds) allowed to elapse between when changes are committed on the primary and when those same changes can be queried on the standby database

```
ALTER SESSION
SET STANDBY_MAX_DATA_DELAY = {INTEGER|NONE}
```

- If the limit is exceeded, an error message is returned:

  `ORA-3172 STANDBY_MAX_DATA_DELAY has been exceeded`

- This setting is ignored for the `SYS` user.

Configuring Zero Lag Between the Primary and Standby Databases

- Certain applications have zero tolerance for any lag.
- Enforce by setting `STANDBY_MAX_DATA_DELAY` to `0`.
- Results are guaranteed to be the same as the primary database, else `ORA-3172` error is returned to the query.
- `SYNC` must be specified for redo transport.

Supporting Read-Mostly Applications

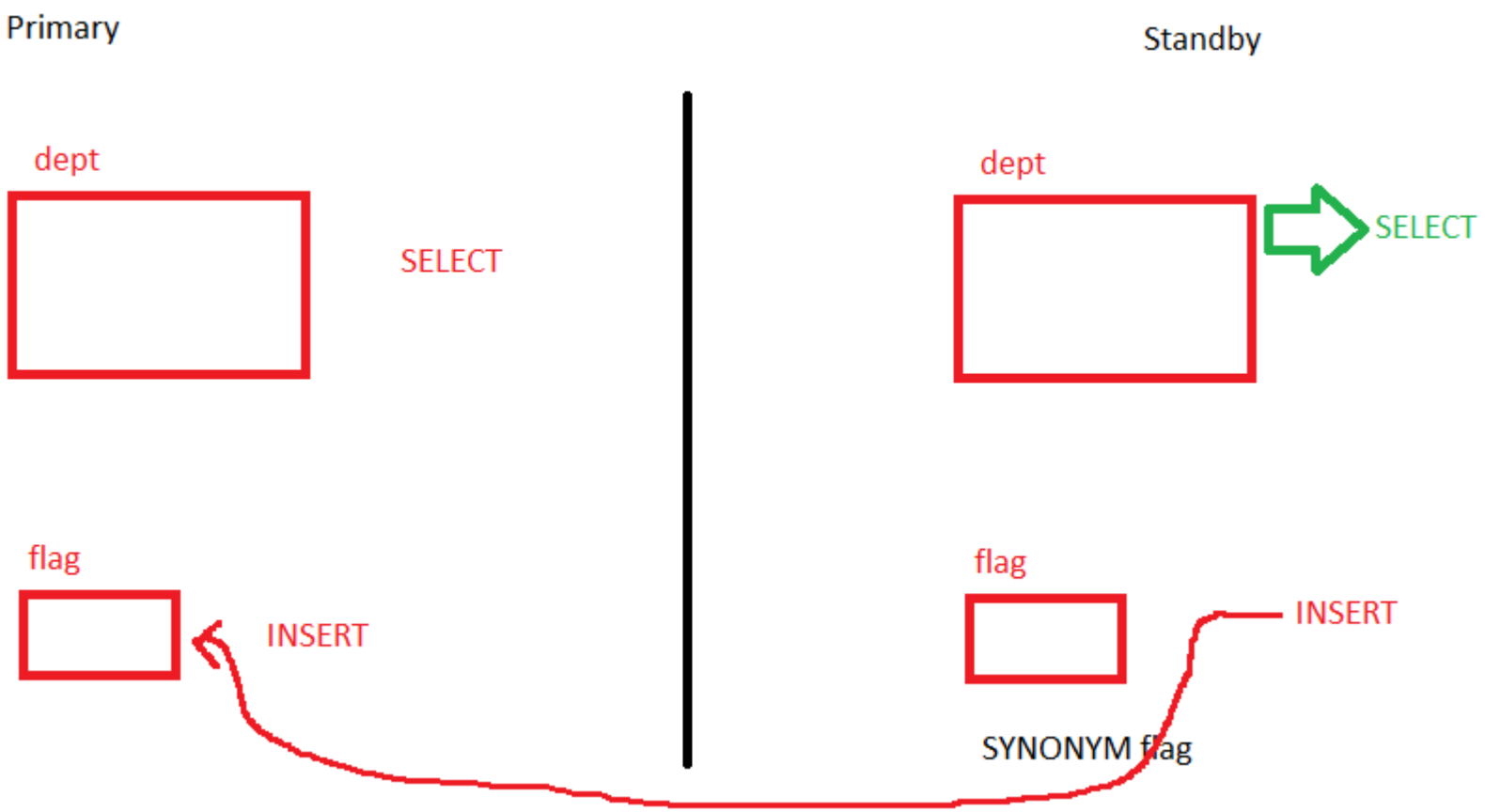
- *Read-mostly applications* are predominantly read-only applications, but require limited read-write database access.
- Active Data Guard supports the read-only portion of read-mostly applications if writes are redirected to the primary database or a local database.
- Writes can be transparently redirected if the application adheres to the following:
  - Modified objects must not be qualified by a schema name.
  - SQL commands must be issued directly from the client, not in stored procedures.

An Example “Read-Mostly Application”:

- The code below returns errors if run on the Standby because of the inserts

```
vari c char(30)
exec select SYS_CONTEXT('USERENV','DATABASE_ROLE') into :c from
    dual;
insert into flag values(:c, sysdate,null);
commit;
select * from dept;
insert into flag values(:c, null, sysdate);
commit;
```

Transparently Redirecting Writes to the Primary Database: The Big Picture



Transparently Redirecting Writes to the Primary Database: The dummy schema

```
grant create session to rmostly identified by rmostly;
grant select on scott.dept to rmostly;
grant all on scott.flag to rmostly;

create synonym rmostly.dept for scott.dept;
create public database link prod
 connect to scott identified by tiger using 'prod';
create synonym rmostly.flag for scott.flag@prod;
```

Transparently Redirecting Writes to the Primary Database: Setting the dummy schema

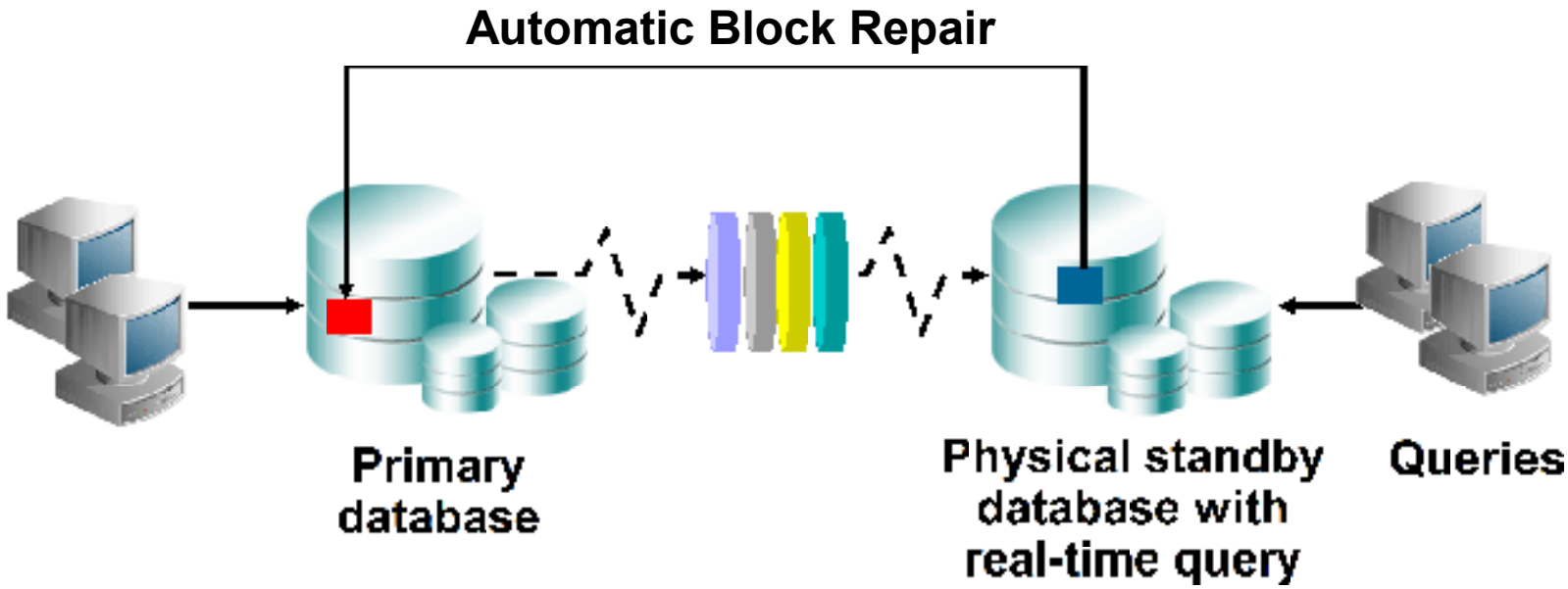
Dummy schema `rmostly` is set on Standby with

Select on `dept` and Insert into `flag@Primary`

```
create or replace trigger switch_schema_trigger
 after logon on scott.schema
 begin
 if (sys_context('userenv','database_role')
 = ('PHYSICAL STANDBY'))
 then
 execute immediate
 'alter session set current_schema = rmostly';
 end if;
end;
/
```

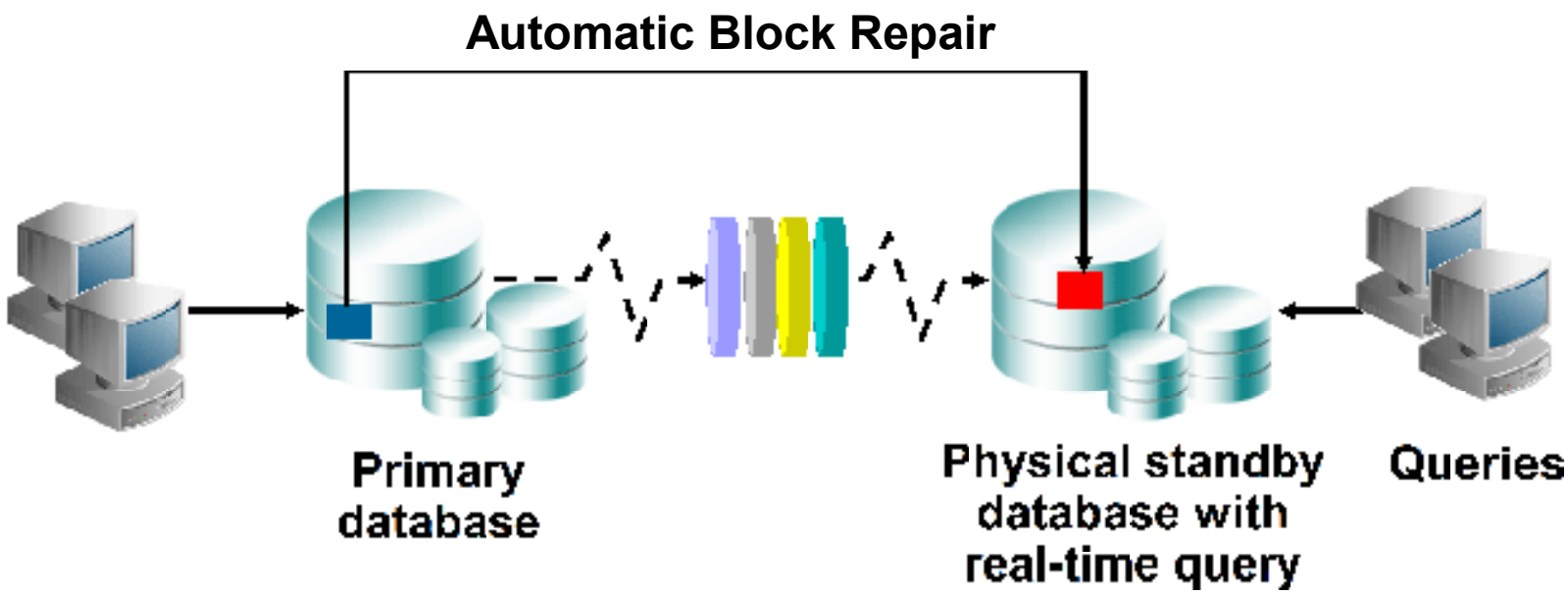
Enhancements to Block Media Recovery

- Corrupted blocks in the primary database are automatically repaired by using blocks from a physical standby database.
- Real-time query and Active Data Guard must be enabled on the physical standby database.



Enhancements to Block Media Recovery

- Corrupted blocks in the physical standby database are automatically repaired by using blocks from the primary database
- Real-time query and Active Data Guard must be enabled on the physical standby database



Thank you for your attention!

Any questions or remarks?

uwe.hesse@oracle.com http://uhesse.com