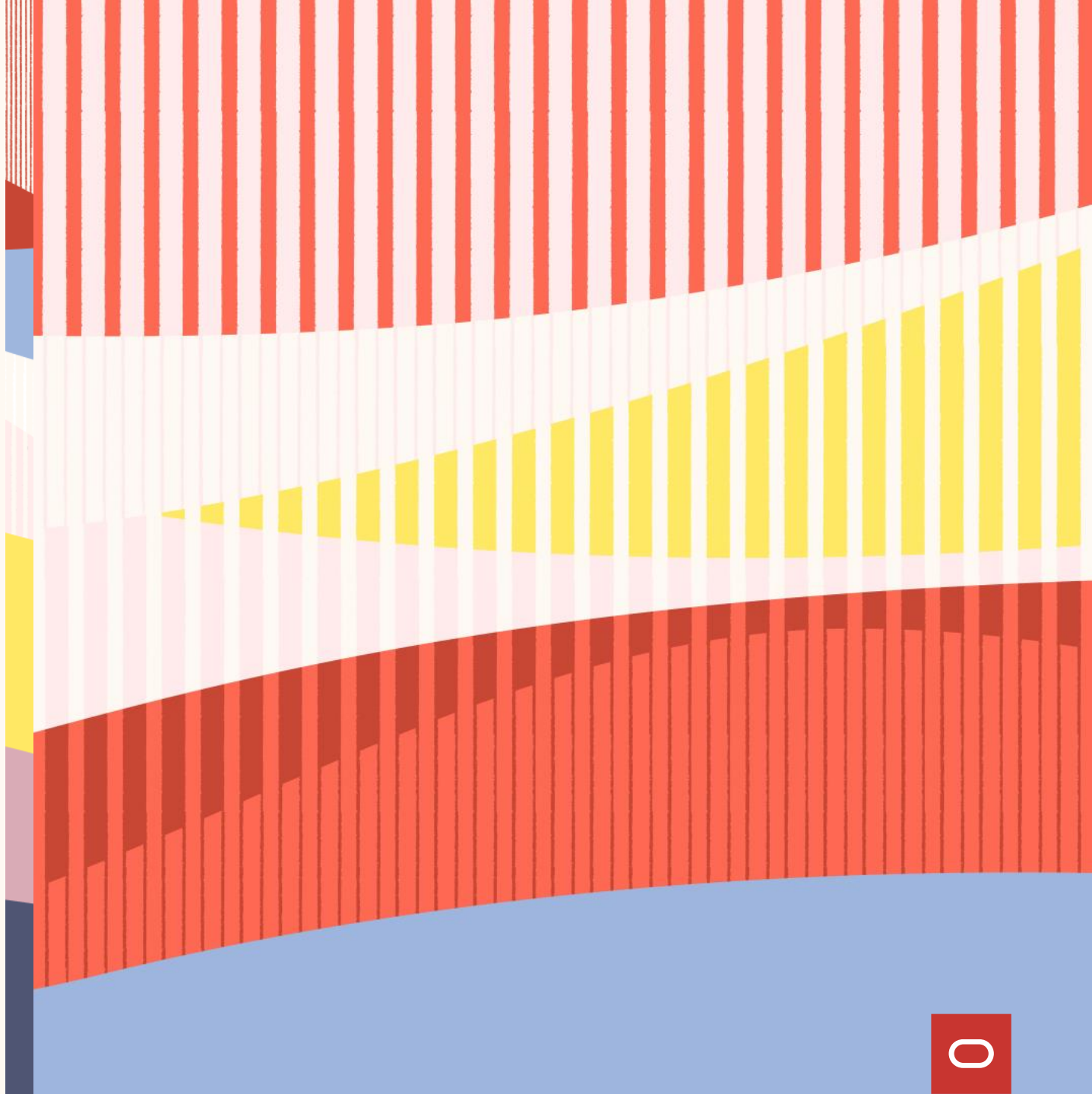
Data Integrity: The Final Frontier

A sneak preview

Toon Koppelaars

Software Architect, Oracle



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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

TL;DR: all of the following may never exist

Integrity Constraints

Not the ones we all know

- Check constraints
- Primary key constraints (and unique constraints)
- Foreign key constraints

Today: sneak preview of SQL assertions

Demo

- tab/dbs
- as1
- dml1
- as2
- dml2
- as3
- dml3
- as

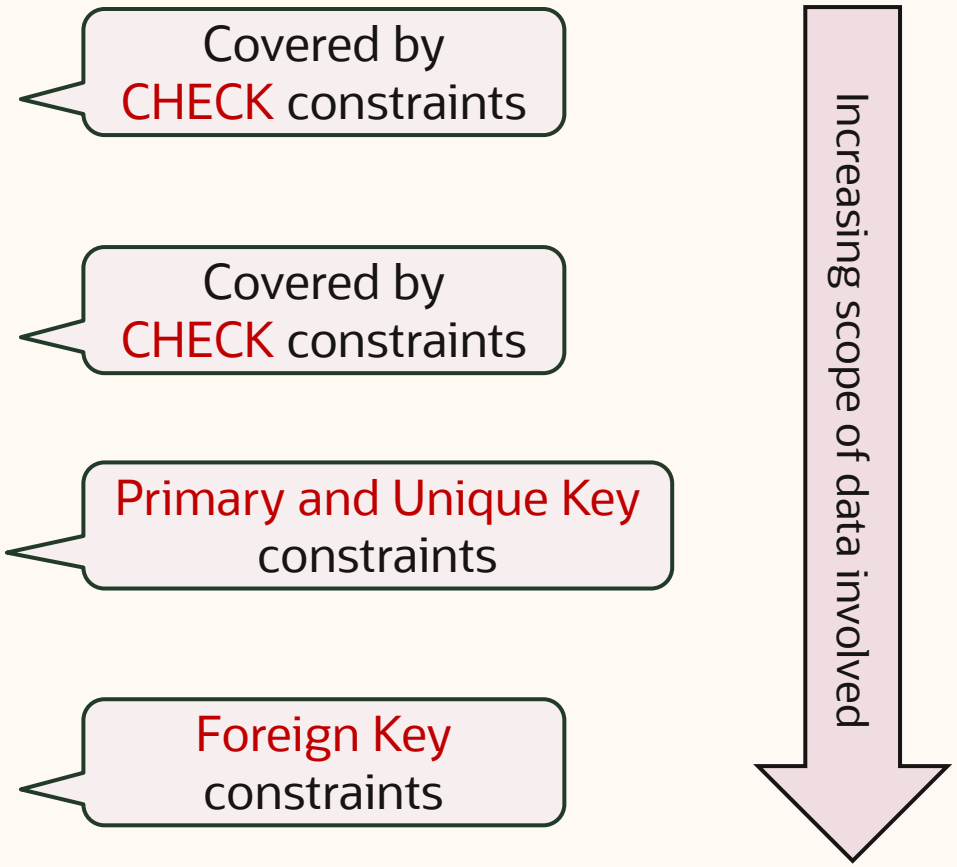
Agenda

- Integrity constraints classification
- SQL assertions
- Execution model: IMMEDIATE vs DEFERRED
- What about performance?
- What about concurrency?

Integrity Constraints

Classification driven by scope-of-data involved

- Column (attribute) constraints
  - values we allow in a column
- Row constraints
  - column value combinations we allow in a row
- Table constraints
  - row combinations we allow in single table
- Database constraints
  - row combinations we allow in different tables



SQL Assertions

Closing the gap

- Column (attribute) constraints
  - values we allow in a column
- Row constraints
  - column value combinations we allow in a row
- Table constraints
  - row combinations
- Database constraints
  - row combinations we allow in different tables

Often your DB design has more constraints in these two classes

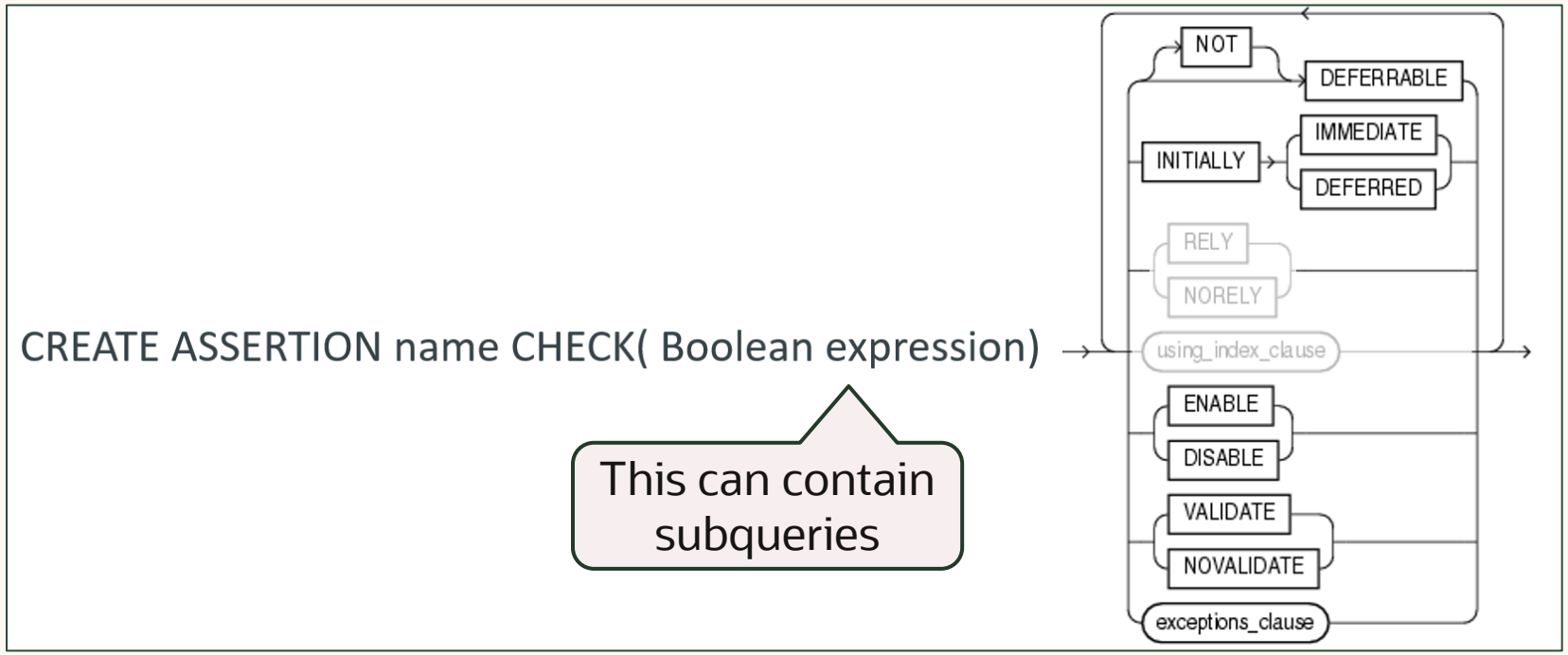
SQL assertions cover the rest

SQL assertions cover the rest

SQL Assertions

What are they?

- Full syntax:



SQL Assertions

What are they?

- Boolean expressions that involve subqueries

```
(not exists
  (select 'A controller in a DEV dept'
     from DEPT d
         ,EMP e
    where d.DEPTNO = e.DEPTNO
      and d.TYPE = 'DEV'
      and e.JOB = 'CONTROLLER'))
```

Subquery

Boolean expression

- DBMS tasked to enforce truth of Boolean expression while transactions change data
- Generic data integrity constraints
- Have been in SQL standard since SQL-92

Some Examples

SQL Assertions

```
create assertion salary_restriction check
(not exists
   (select 'you earning more than your manager'
      from EMP e, EMP m
     where e.MGR = m.EMPNO
       and e.SALARY > m.SALARY))
```

```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

Two levels deep

Some Examples

SQL Assertions

```
create assertion at_most_one_president check
(not exists
  (select 'two presidents'
     from EMP e1, EMP e2
    where e1.JOB = 'PRESIDENT' and e2.JOB = 'PRESIDENT'
      and e1.ROWID != e2.ROWID))
```

```
create assertion president_must_be_there check
(exists
  (select 'a president'
     from EMP e
    where e.JOB = 'PRESIDENT'))
```

Top-level EXISTS

Here's Another One...

```
create assertion Dont_Do_This check
(not exists
  (select ''
     from EMP e1, EMP e2
    where e1.EMPNO = e2.EMPNO
      and e1.ROWID != e2.ROWID))
```

Better to declare as
PRIMARY KEY(EMPNO)

Here's Another One...

```
create assertion Also_Dont_Do_This check
(not exists
   (select ''
      from EMP e
     where not exists
             (select ''
                from DEPT d
               where d.DEPTNO = e.DEPTNO)))
```

Better to declare as
FOREIGN KEY(DEPTNO) references DEPT(DEPTNO)

Here's Another One...

```
create assertion Definitely_Dont_Do_This check
(not exists
   (select ''
      from EMP e
     where JOB not in ('CLERK','TRAINER','MANAGER','PRESIDENT')))
```

Better to declare as
CHECK JOB in (....)

They are generic data integrity constraints
But please do keep using the ones already there

How Complex Can They be?

- Some restrictions in 1st release

Scope of Syntax Supported in First Release

SQL Assertions

- Top-level NOT EXISTS (subquery), or top-level EXISTS (subquery)
- Subquery accesses base schema tables, and can be equijoin of two or more tables
  - Optionally with filter predicates, logically AND-ed, or OR-d
- Filter predicates can be:
  - Regular column-filters, combined with
  - Mix of nested NOT EXISTS and EXISTS → can go up to three levels deep
- Must be deterministic, so cannot reference:
  - SYSDATE, USER, USER_ENV, SYS_CONTEXT, PL/SQL functions, …

- Not yet supported:
  - Group-by
  - Use of aggregate functions
  - Outer-join
  - Set operators union/minus/intersect

} Future release

New System Privileges

Compared With Constraints

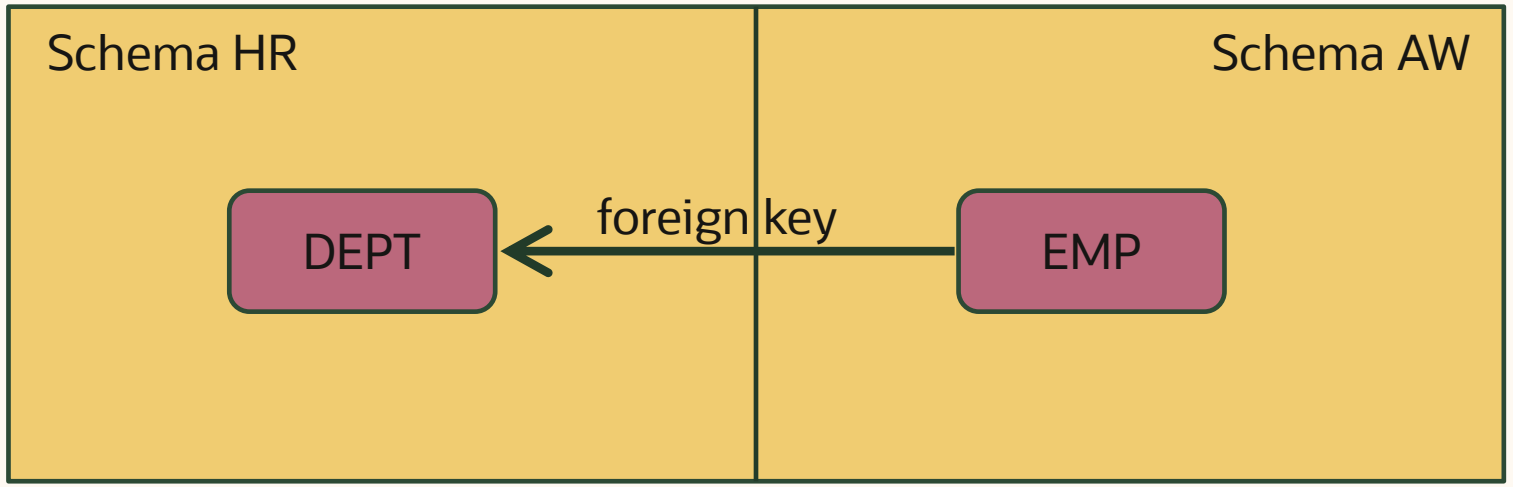
Part of DB_DEVELOPER role

- Schema-level objects, new object-type ASSERTION
- New system privileges: CREATE ASSERTION, CREATE/ALTER/DROP ANY ASSERTION
- Tracked in own data dictionary views user/all/dba_assertions

```
SQL> desc DBA_ASSERTIONS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 OWNER                                               VARCHAR2(128)
 ASSERTION_NAME                                      VARCHAR2(128)
 OBJECT_ID                                 NOT NULL  NUMBER
 STATUS                                              VARCHAR2(8)
 DEFERRABLE                                          VARCHAR2(14)
 DEFERRED                                            VARCHAR2(9)
 VALIDATED                                           VARCHAR2(13)
 INVALID                                             VARCHAR2(7)
 DEFINITION_SQL                                      CLOB

SQL>
```

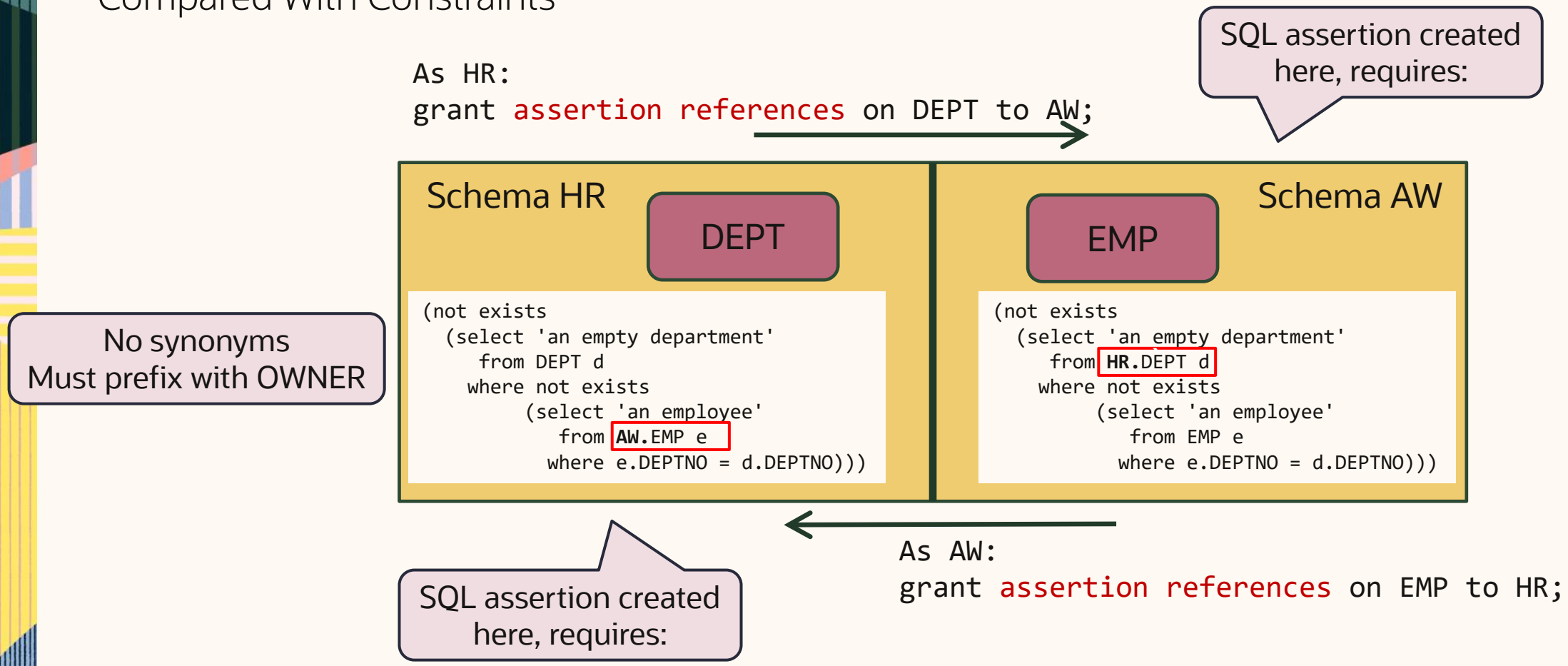
Remember REFERFENCES Object Privilige?




```
As HR:
grant references on DEPT to AW;
```

New Object Privilege for Multi-Schema Support

Compared With Constraints



In All Examples

- I've used NOT twelve times so far...

Negations, Negations, Negations

Difficult for our brain

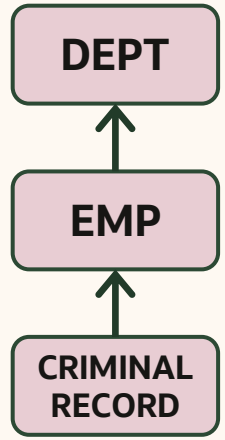
```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

```
create assertion at_least_one_non_criminal check
(not exists
  (select 'dept without EMP without CR'
     from DEPT d
    where not exists
           (select 'EMP without CR'
              from EMP e
             where e.DEPTNO = d.DEPTNO
               and not exists
                    (select 'a CR'
                       from CRIMINAL_RECORD cr
                      where cr.EMPNO = e.EMPNO))))
```

Every department must have an employee that's not criminal

There cannot be department that doesn't have employee that's not a criminal

Three levels deep



New ALL-SATISFY Syntax to Prevent Nested Negations

SQL Assertions

```
create assertion manager_without_clerk check
(ALL
   (select e1.DEPTNO
      from EMP e1
     where e1.JOB = 'MANAGER') m
 SATISFY
   (exists
     (select 'clerk in same department'
        from EMP e2
       where e2.DEPTNO = m.DEPTNO and e2.JOB = 'CLERK')))
```

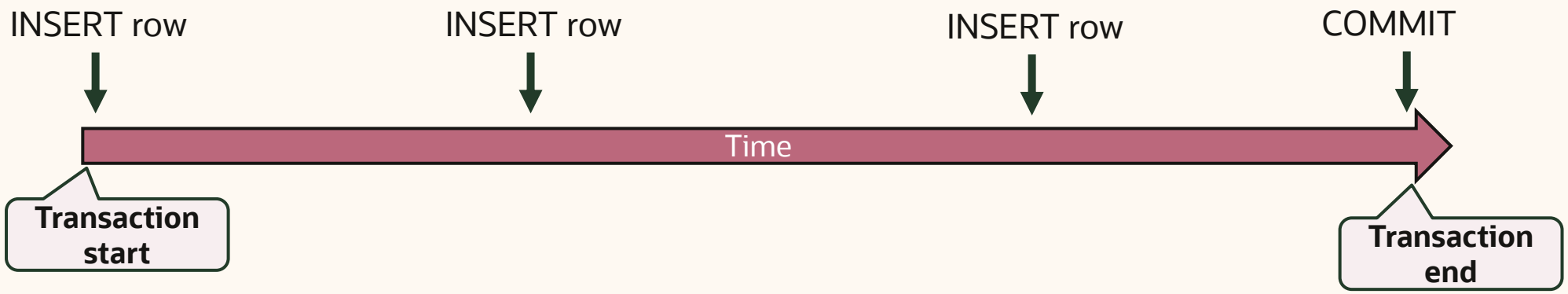
```
create assertion at_least_one_non_criminal check
(ALL
  (select d.DEPTNO
     from DEPT d) d
 SATISFY
  (exists
    (select 'EMP without CR'
       from EMP e
      where e.DEPTNO = d.DEPTNO
        and not exists
              (select 'a CR'
                 from CRIMINAL_RECORD cr
                where cr.EMPNO = e.EMPNO))))
```

Two Execution Models: Immediate and Deferred

Same as with current constraints

Two Execution Models: Immediate and Deferred

Same as with current constraints

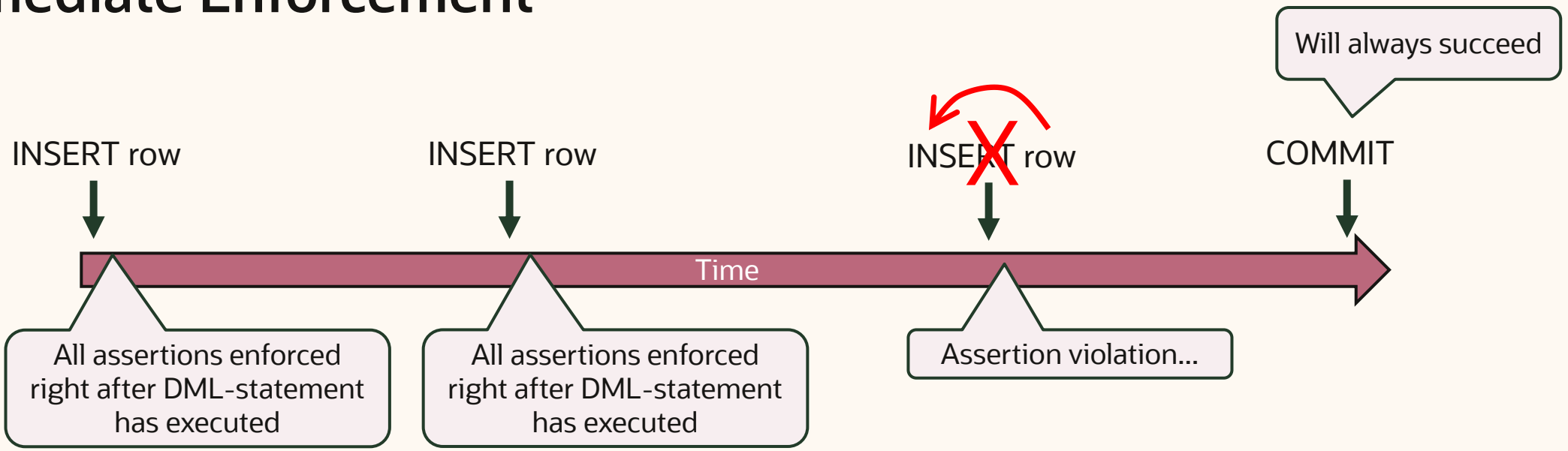


1. Immediate enforcement → assertions are enforced for each DML-statement

2. Deferred enforcement → assertions are enforced at end-of-TX

DML: Data Manipulation Language
→ insert, update, or delete

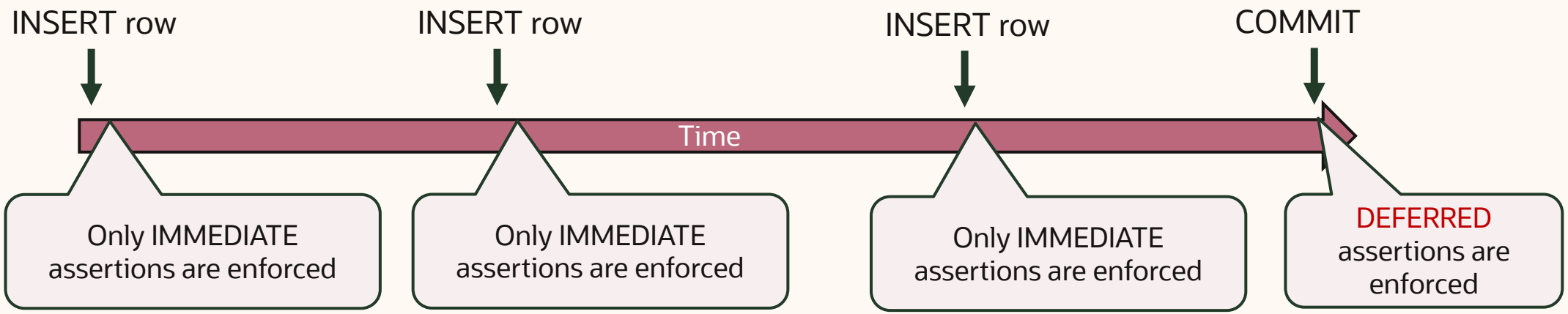
Immediate Enforcement



- The default behavior
- When DML-statement violates assertion → it receives error, and its changes are undone → statement-level rollback

DML: Data Manipulation Language
→ insert, update, or delete

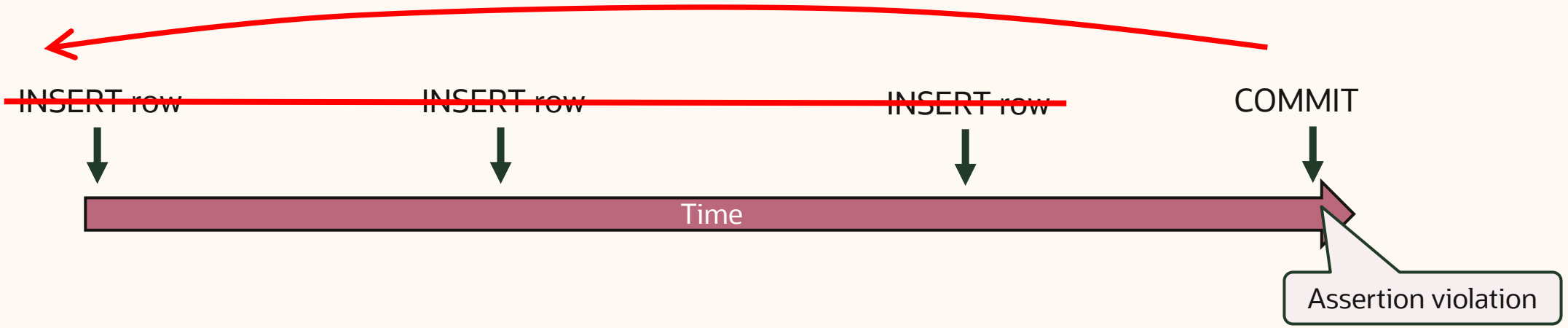
Deferred Enforcement



- IMMEDIATE / DEFERRED execution is property of assertion
- DEFERRED assertions are enforced at COMMIT time

DML: Data Manipulation Language
→ insert, update, or delete

Deferred Enforcement



- When (at least 1) DEFERRED assertion is still in violation at COMMIT → COMMIT receives error, and whole TX is undone

DML: Data Manipulation Language
→ insert, update, or delete

Some Assertions Require INITIALLY DEFERRED

```
create assertion no_empty_departments check
(not exists
   (select 'empty dept'
      from DEPT d
     where not exists
           (select 'an employee'
              from EMP e
             where e.DEPTNO = d.DEPTNO)))
```

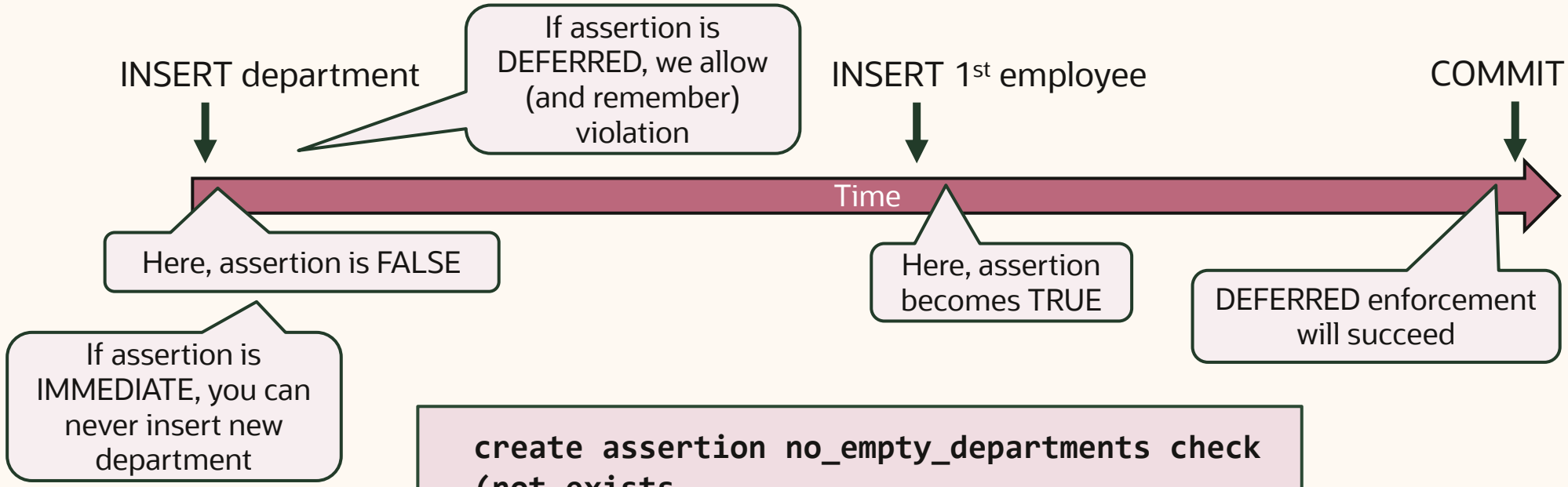
or equivalent,

```
create assertion no_empty_departments check
(ALL(select d.DEPTNO
       from DEPT d) d
 SATISFY(exists
          (select 'an employee'
             from EMP e
            where e.DEPTNO = d.DEPTNO)))
```

Every department must have at least one employee

No Empty Departments

This one cannot be IMMEDIATE




```
create assertion no_empty_departments check
(not exists
   (select 'empty dept'
      from DEPT d
     where not exists
           (select 'an employee'
              from EMP e
             where e.DEPTNO = d.DEPTNO)))
 deferrable initially deferred
```

Must add this

Demo: Deferred Execution

- as4
- dml4
- dml5
- as4b
- dml4 + commit;
- dml4 + dml5 + commit

Agenda

- Integrity constraints classification
- SQL assertions
- Execution model: IMMEDIATE vs DEFERRED
- What about performance?
- What about concurrency?

What About Performance?

- Do we validate every SQL assertion on each DML-statement?

  → No, only when DML could've potentially violated SQL assertion

- Do we validate full Boolean expression of SQL assertion?

  → No, we do incremental re-validation whenever possible

Re-validate Only When Required

Which types of DML-statements could violate?

```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

"Every Manager must have Clerk (in same department)"

1. INSERT of a Manager
2. DELETE of a Clerk

3. JOB-update to Manager
4. JOB-update away from Clerk

5. DEPTNO-update of Manager or Clerk

Re-validate Only When Required

Which types of DML-statements could violate?

```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

1. INSERT of a Manager
2. DELETE of a Clerk

3. JOB-update to Manager
4. JOB-update away from Clerk

5. DEPTNO-update of Manager or Clerk

At end of DML-execution, we check if your DML statement did any of these five

If no, we're done
If yes, we need to ensure SQL assertion is still TRUE

We do that by running modified version of query you gave us

We do Incremental Revalidation

```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

At compile-time we determine if assertion is eligible for incremental revalidation, and if so at what level
In this case: yes, at DEPTNO-level

1. INSERT of a Manager in department 10 → Only revalidate department 10 is OK
2. DELETE of a Clerk from department 10 → Only revalidate department 10 is OK

3. JOB-update to Manager
4. JOB-update away from Clerk

5. DEPTNO-update of Manager or Clerk

We do Incremental Revalidation

```
create assertion manager_without_clerk check
(not exists
   (select 'manager without clerk'
      from EMP e1
     where e1.JOB = 'MANAGER'
       and not exists
             (select 'clerk in same department'
                from EMP e2
               where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

1. INSERT of a Manager in department 10 → Only revalidate department 10 is OK

```
select 'manager without clerk'
  from (select * from inserted_rows where JOB = 'MANAGER') e1
 where e1.JOB = 'MANAGER'
   and not exists
         (select 'clerk in same department'
            from EMP e2
           where e2.DEPTNO = e1.DEPTNO and e2.JOB = 'CLERK')))
```

In this case:
table EMP (e1) is replaced by object that returns managers that were just inserted

This query is equivalent to the one you would've coded yourself

View USER_ASSERTION_DEPENDENCIES

Shows INCREMENTAL possible or not

```
select assertion_name,referenced_name table_name,validation_type,validation_event
from user_assertion_dependencies
order by 1,2
```

```
ASSERTION_NAME            TABLE_NAME      VALIDATI VALIDATION_EVENT
------------------------- --------------- -------- ----------------------
AT_MOST_ONE_PRESIDENT     EMP             FAST     ROWS ADDED OR UPDATED
AT_MOST_ONE_PRESIDENT     EMP             FAST     ROWS ADDED OR UPDATED
MANAGER_WITHOUT_CLERK     EMP             FAST     ROWS ADDED OR UPDATED
MANAGER_WITHOUT_CLERK     EMP             FAST     ROWS DELETED OR UPDATED
NO_CONTROLLERS_IN_DEV     DEPT            FAST     ROWS ADDED OR UPDATED
NO_CONTROLLERS_IN_DEV     EMP             FAST     ROWS ADDED OR UPDATED
PRESIDENT_MUST_BE_THERE   EMP             COMPLETE ROWS DELETED OR UPDATED
SALARY_RESTRICTION        EMP             FAST     ROWS ADDED OR UPDATED
SALARY_RESTRICTION        EMP             FAST     ROWS ADDED OR UPDATED
```

Agenda

- Integrity constraints classification
- SQL assertions
- Execution model: IMMEDIATE vs DEFERRED
- What about performance?
- What about concurrency?

What About Concurrency?

```
(not exists
  (select 'A controller in a DEV dept'
     from DEPT d
         ,EMP e
    where d.DEPTNO = e.DEPTNO
      and d.TYPE = 'DEV'
      and e.JOB = 'CONTROLLER'))
```

- Assume D42 currently empty and of type FIN:

Time	Transaction 1	Transaction 2
1	INSERT EMP 1st CONTROLLER in D42	
2		UPDATE DEPT D42 → DEV

We need to "lock value D42", communicating to other TX's, don't mess with D42

- Like foreign key induced serial execution order:

Time	Transaction 1	Transaction 2
1	INSERT 1st EMP in D42	
2		DELETE DEPT D42 (**blocks**)

- We never lock rows or tables: all done via new enqueue type (AN)
- New wait-event → "enq: AN - SQL Assertion DDL/DML"

Demo: Concurrency

- as6
- dbs (s1)
- dml6 (session 1)
- dbs (s2)
- dml7 (session 2, blocks)
- waiters
- locks
- s1 commit
- s2 error

Data Integrity the Final Frontier: SQL Assertions

In conclusion:

- SQL assertions allow whole new class of constraints
  - Just declare
  - And let DBMS enforce

- Should be powerful feature when combined with Apex

Questions?

toon.koppelaars@oracle.com