Constraints (3)

Exceptions Table Summarized

A table

```
CREATE TABLE Emp (
   empno NUMBER PRIMARY KEY,
   ename VARCHAR2(30),
   sal NUMBER,
   deptno NUMBER
);
```

Violating Tuples (I)

```
INSERT INTO Emp(empno, ename, sal, deptno)
            VALUES(9, 'Jon', 300, 20);

ALTER TABLE Emp ADD CONSTRAINT check_sal CHECK(sal >= 500);

--The constraint cannot be created at all, because there is
--a violating tuple.
```

- To identify those tuples that violate a constraint whose activation failed, one can use the clause

  EXCEPTIONS INTO Exceptions

  Exceptions is a table that we should create and stores information about the violating tuples.

Violating Tuples (II)

```
INSERT INTO Emp(empno, ename, sal, deptno)
            VALUES(9, 'ALEX', 300, 20);

ALTER TABLE Emp ADD CONSTRAINT check_sal CHECK(sal >= 500)
   EXCEPTIONS INTO Exceptions;

--The constraint cannot be created at all, because there is
--a violating tuple.
```

Violating Tuples (cont.)

- First we have to create the **`Exceptions`** table:

```
CREATE TABLE Exceptions(
    row_id ROWID,
    owner VARCHAR2(30),
    table_name VARCHAR2(30),
    constraint VARCHAR2(30)
);
```

Every tuple has a (pseudo) column of type rowid that is used to identify tuples.

row_id here will reference to rowid in the Emp table.

Besides the row_id, the name of the table, the table owner, as well as the name of the violated constraint are stored.

- Then, we can query it:

```
SELECT Emp.*, constraint
FROM Emp, Exceptions
WHERE Emp.rowid = Exceptions.row_id;
```