

# Reminder: insert statements

```
INSERT INTO Movies (title,year,length,rating,studio)
VALUES('Walk the Line', 2005, 136, 'PG', 'Fox');
```

```
INSERT INTO Movies
VALUES('Pretty Woman', 1990, 119, 'R', 'Disney');
```

```
INSERT INTO Movies
VALUES('The Princess and the Frog', 2009, 97, null, 'Disney');
```

# Update statements

```
UPDATE Movies
```

```
SET rating = 'R'
```

```
WHERE title='Star wars'
```

# Data integrity: basic constraints

# Constraints

- Data types are a way to limit the kind of data that can be stored in a table.
- For many applications, however, the constraint they provide is too coarse.

For example, a column containing a **product price** should probably only accept **positive values**. But there is no standard data type that accepts only positive numbers.

# Two types of constraints

- **Referential constraints:**
  - PRIMARY KEY,
  - FOREIGN KEY
- **Value constraints:**
  - NULL,
  - UNIQUE,
  - CHECK - certain set of values

Referential integrity  
constraints

# Keys in SQL

- Keys play an important role in SQL, because specifying the values of key attributes is a way of referring to a unique tuple in a relation.
- Since updates (entered by users of the database) could potentially violate the uniqueness of a key, DBMSs offer to check this.

# Primary key

```
CREATE TABLE Movies (  
    title CHAR(40) PRIMARY KEY,  
    year INT,  
    length INT,  
    type CHAR(2)  
);
```

Defined at column level

```
CREATE TABLE Movies (  
    title CHAR(40),  
    year INT,  
    length INT,  
    type CHAR(2),  
    PRIMARY KEY (title, year)  
);
```

Defined at table level



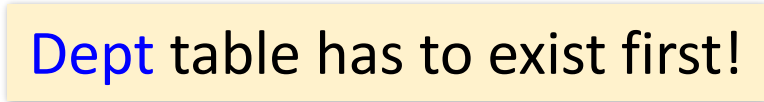
# Foreign keys

- In relational model tables are related to each other through common column
- A column (or a set of columns) in one table is a *primary key* of this table, if its value uniquely identifies each tuple (row). Such table is called a **parent table**
- A column in another table that references a primary key column in the parent table is known as a *foreign key*. This table is called a **child table**

# Foreign key constraints

**Example: Each employee in table Emp must work in a department that is contained in table Dept.**

```
CREATE TABLE Emp (  
    empno INT PRIMARY KEY,  
    ... ,  
    deptno INT REFERENCES Dept (deptno)  
);
```



Dept table has to exist first!

# Foreign keys: movies

**Remark.** If you don't specify primary keys or unique constraints in the parent tables, you can't specify foreign keys in the child tables.

```
CREATE TABLE MovieStars (  
    name VARCHAR2(20) PRIMARY KEY,  
    address VARCHAR2(30),  
    gender VARCHAR2(1),  
    birthdate VARCHAR2(20)  
);
```

```
CREATE TABLE Movies (  
    title VARCHAR2(40),  
    year INT,  
    length INT,  
    type VARCHAR2(2),  
    PRIMARY KEY (title, year)  
);
```

```
CREATE TABLE StarsIn (  
    title VARCHAR2(40),  
    year INT,  
    starName VARCHAR2(20),  
    FOREIGN KEY (title, year) REFERENCES Movies (title, year),  
    FOREIGN KEY (starName) REFERENCES MovieStars (name)  
);
```

Value constraints: check

# CHECK constraints

allow users to restrict the domain of possible attribute values for columns to admissible ones

```
[CONSTRAINT <name>] CHECK (<condition>)
```

# Column-level CHECK constraints: examples

- The name of an employee must consist of upper case letters only;
- The minimum salary of an employee is 500;
- Department numbers must range between 10 and 100:

```
CREATE TABLE Emp (  
    empno NUMBER,  
    ename VARCHAR2(30) CONSTRAINT check_name  
        CHECK( ename = UPPER(ename) ),  
    sal NUMBER CONSTRAINT check_sal  
        CHECK( sal >= 500 ),  
    deptno NUMBER CONSTRAINT check_deptno  
        CHECK( deptno BETWEEN 10 AND 100 )  
);
```

# Enforcing CHECK constraints

- DBMS automatically checks the specified conditions each time a database modification is performed on this relation. E.g., the insertion

```
INSERT INTO emp  
VALUES (7999, 'SCOTT', 450, 10) ;
```

causes a constraint violation and it is rejected.