CSCD 327: Relational Database Systems

Data integrity

Instructor: Dr. Dan Li

Data Integrity

- Data integrity refers to the correctness and completeness of the data in a database.
- The integrity can be lost
  - *INSERT* a new order, but the product doesn't exist.
  - *UPDATE* a salesperson to a nonexistent office.
  - DELETE an office, but didn't reassign salespersons to new offices.
  - Power failure

Integrity Constraints

- Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.
  - Required data – NOT NULL
  - Validity checking – CHECK(…)
  - Entity integrity – PRIMARY KEY, UNIQUE
  - Referential integrity – FOREIGN KEY… ON DELETE/UPDATE
  - Advanced constraint – ASSERTION
  - Business rules – TRIGGER

Required Data

- When a column must contain a non-null value, you can add an integrity constraint NOT NULL in the CREATE TABLE statement.

E.g., Create Table OFFICES

```
    (OFFICE    INTEGER NOT NULL,
     …)
```

- Every INSERT statement that adds one or more new rows to the table must specify a non-NULL data value for the column. An attempt to insert a row containing a NULL value (either explicitly or implicitly) results in an error.
- Every UPDATE statement that updates the column must assign it a non-NULL data value. Again, an attempt to update the column to a NULL value results in an error.

Validity Checking

- A column check constraint specifies a search condition similar to a WHERE clause.

```
e.g., Create Table OFFICES
      (OFFICE    INTEGER            NOT NULL
                 CHECK (OFFICE BETWEEN 11 and 22),
       …)
```

- CHECK constraint can be added to an existing table as part of an ALTER TABLE statement.
- CHECK can also be added to a **CREATE DOMAIN** statement.

```
e.g., CREATE DOMAIN VALID_EMPLOYEE_ID INTEGER
         CHECK (VALUE BETWEEN 101 AND 199);

      CREATE TABLE SALESREPS
         (EMPL_NUM VALID_EMPLOYEE_ID,
        …)
```

Entity Integrity

- Entity integrity constraint requires primary keys to have unique values.
- Use PRIMARY KEY to enforce entity integrity constraint.

e.g., Create Table OFFICES

```
      (OFFICE    INTEGER NOT NULL,
        …
        PRIMARY KEY (OFFICE))
```

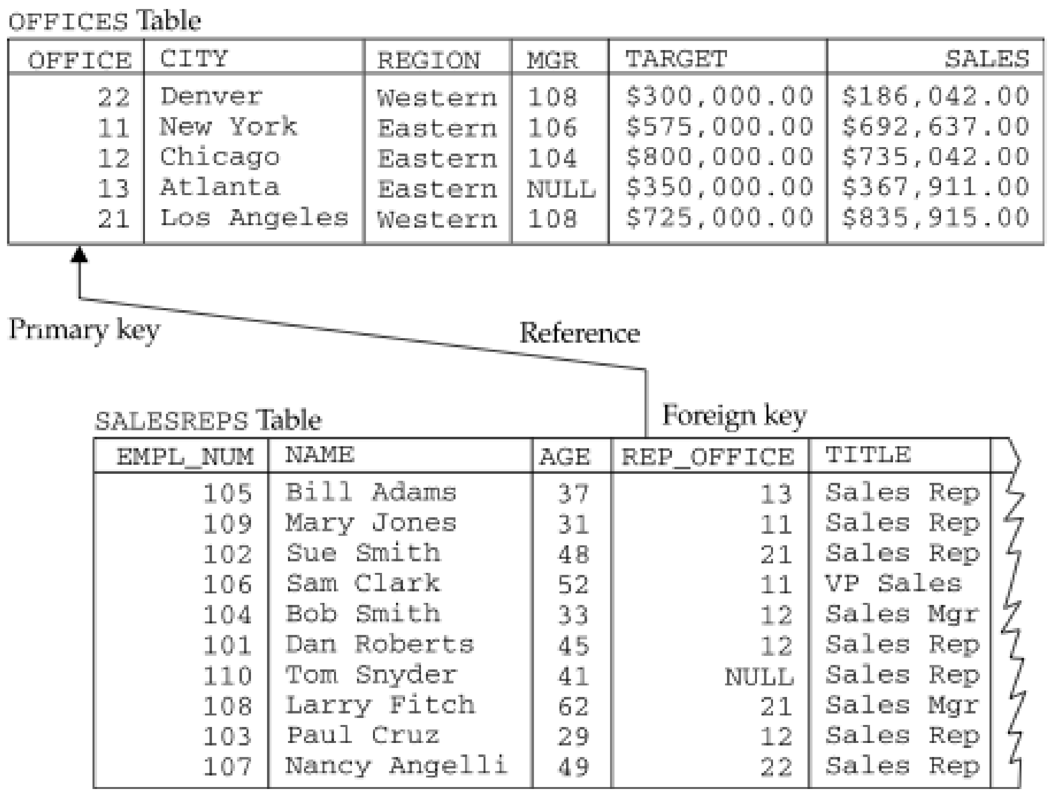
- It is sometimes appropriate to require a column that is not the primary key of a table to contain a unique value in every row. Use UNIQUE to enforce the constraint.

e.g., Create Table OFFICES

```
      (OFFICE    INTEGER NOT NULL,
       CITY      VARCHAR(15)     NOT NULL UNIQUE,
        …)
```

Referential Integrity

- Parent/Child relationship



OFFICES Table

OFFICE	CITY	REGION	MGR	TARGET	SALES
22	Denver	Western	108	$300,000.00	$186,042.00
11	New York	Eastern	106	$575,000.00	$692,637.00
12	Chicago	Eastern	104	$800,000.00	$735,042.00
13	Atlanta	Eastern	NULL	$350,000.00	$367,911.00
21	Los Angeles	Western	108	$725,000.00	$835,915.00

SALESREPS Table

EMPL_NUM	NAME	AGE	REP_OFFICE	TITLE
105	Bill Adams	37	13	Sales Rep
109	Mary Jones	31	11	Sales Rep
102	Sue Smith	48	21	Sales Rep
106	Sam Clark	52	11	VP Sales
104	Bob Smith	33	12	Sales Mgr
101	Dan Roberts	45	12	Sales Rep
110	Tom Snyder	41	NULL	Sales Rep
108	Larry Fitch	62	21	Sales Mgr
103	Paul Cruz	29	12	Sales Rep
107	Nancy Angelli	49	22	Sales Rep

```
INSERT INTO SALESREPS (EMPL_NUM, NAME, REP_OFFICE, AGE,
                       HIRE_DATE, SALES)
     VALUES (115, 'George Smith', 31, 37, '2008-04-01', 0.00);
```

Referential Integrity (Cont.)

- **Problem 1**: Inserting a new child row
- **Solution 1**: Referential integrity constraint will automatically check the values of the foreign key columns before the INSERT statement is permitted. If they don't match a primary key value, the INSERT statement is rejected with an error message.
- **Problem 2**: Updating the foreign key in a child row
- **Solution 2**: Referential integrity constraint will automatically check the updated foreign key value. If there is no matching primary key value, the UPDATE statement is rejected with an error message.

```
CREATE TABLE SALESREPS
(….
   REP_OFFICE INTEGER,
   FOREIGN KEY WORKSIN (REP_OFFICE)
   REFERENCES OFFICES)
```

Referential Integrity (Cont.)

- **Problem 3**: Deleting a parent row
- **Solution 3**.1: Prevent the office from being deleted until the salespeople are reassigned. **ON DELETE RESTRICT**
- **Solution 3.2**: Automatically delete the salespeople from the SALESREPS table as well. **ON DELETE CASCADE**
- **Solution 3.3**: Set the REP_OFFICE column for the salespeople to NULL, indicating that their office assignment is unknown. **ON DELETE SET NULL**
- **Solution 3.4**: Set the REP_OFFICE column for the salespeople to some default value, such as the office number for the headquarters office in New York, indicating that the salespeople are automatically reassigned to that office. **ON DELETE SET DEFAULT**

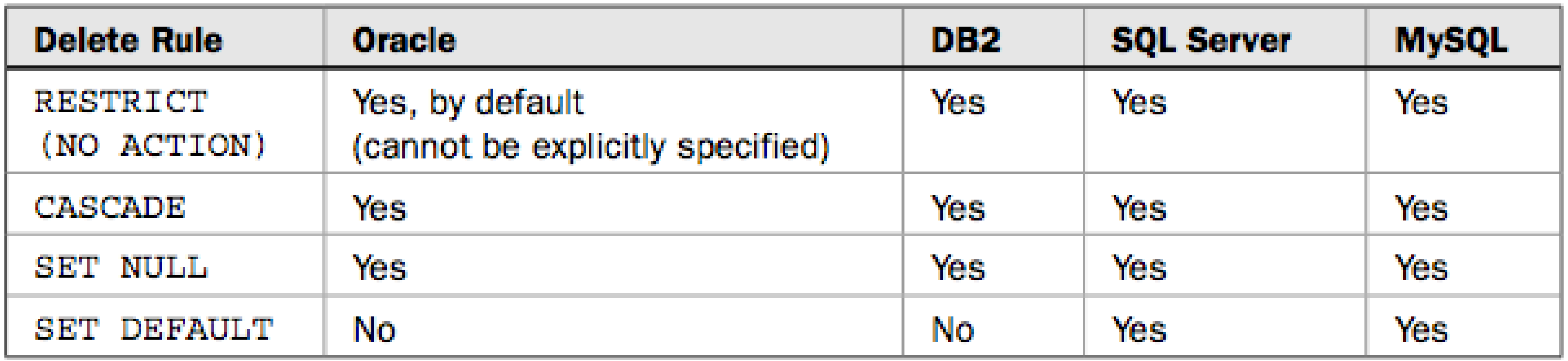
Referential Integrity (Cont.)

- **Problem 4**: Updating the primary key in a parent row
- **Solution 4.1**: Prevent the office number from being changed until the salespeople are reassigned. . **ON UPDATE RESTRICT**
- **Solution 4.2**: Automatically update the office number for the two salespeople in the SALESREPS table, so that their rows are still linked to the Los Angeles row in the OFFICES table, via its new office number. **ON UPDATE CASCADE**
- **Solution 4.3**: Set the REP_OFFICE column for the two salespeople to NULL, indicating that their office assignment is unknown.

ON UPDATE SET NULL

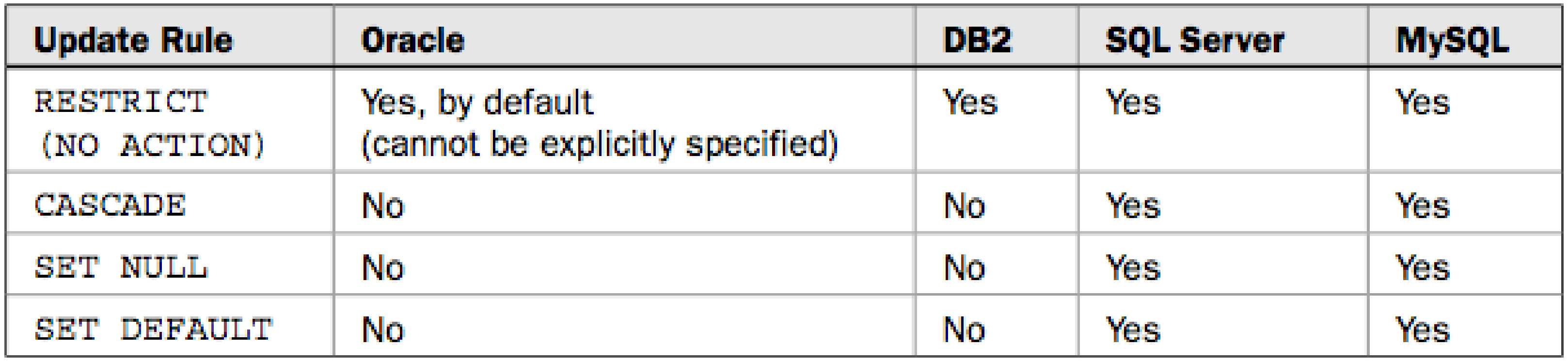
- **Solution 4.4**: Set the REP_OFFICE column for the two salespeople to some default value, such as the office number for the headquarters office in New York, indicating that the salespeople are automatically reassigned to that office. **ON UPDATE SET DEFAULT**

Referential Integrity (Cont.)



Delete Rule	Oracle	DB2	SQL Server	MySQL
RESTRICT (NO ACTION)	Yes, by default (cannot be explicitly specified)	Yes	Yes	Yes
CASCADE	Yes	Yes	Yes	Yes
SET NULL	Yes	Yes	Yes	Yes
SET DEFAULT	No	No	Yes	Yes

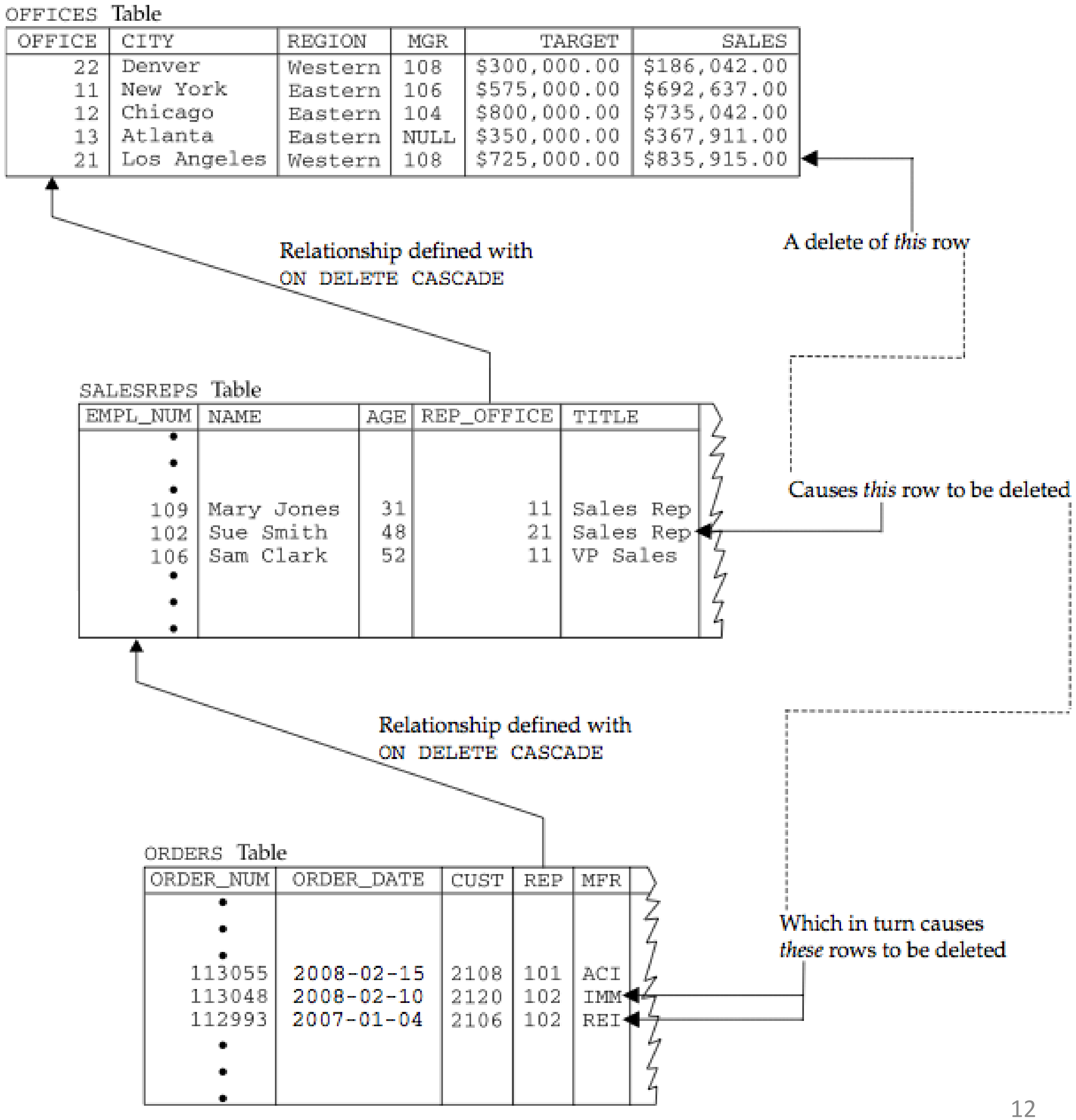
Table 11-1 Delete Rule Support in Popular DBMSs



Update Rule	Oracle	DB2	SQL Server	MySQL
RESTRICT (NO ACTION)	Yes, by default (cannot be explicitly specified)	Yes	Yes	Yes
CASCADE	No	No	Yes	Yes
SET NULL	No	No	Yes	Yes
SET DEFAULT	No	No	Yes	Yes

Table 11-2 Update Rule Support in Popular DBMSs

Use Referential Constraint with Care!!



Relationship defined with ON DELETE CASCADE
A delete of *this* row
Causes *this* row to be deleted
Relationship defined with ON DELETE CASCADE
Which in turn causes *these* rows to be deleted

OFFICES Table

OFFICE	CITY	REGION	MGR	TARGET	SALES
22	Denver	Western	108	$300,000.00	$186,042.00
11	New York	Eastern	106	$575,000.00	$692,637.00
12	Chicago	Eastern	104	$800,000.00	$735,042.00
13	Atlanta	Eastern	NULL	$350,000.00	$367,911.00
21	Los Angeles	Western	108	$725,000.00	$835,915.00

SALESREPS Table

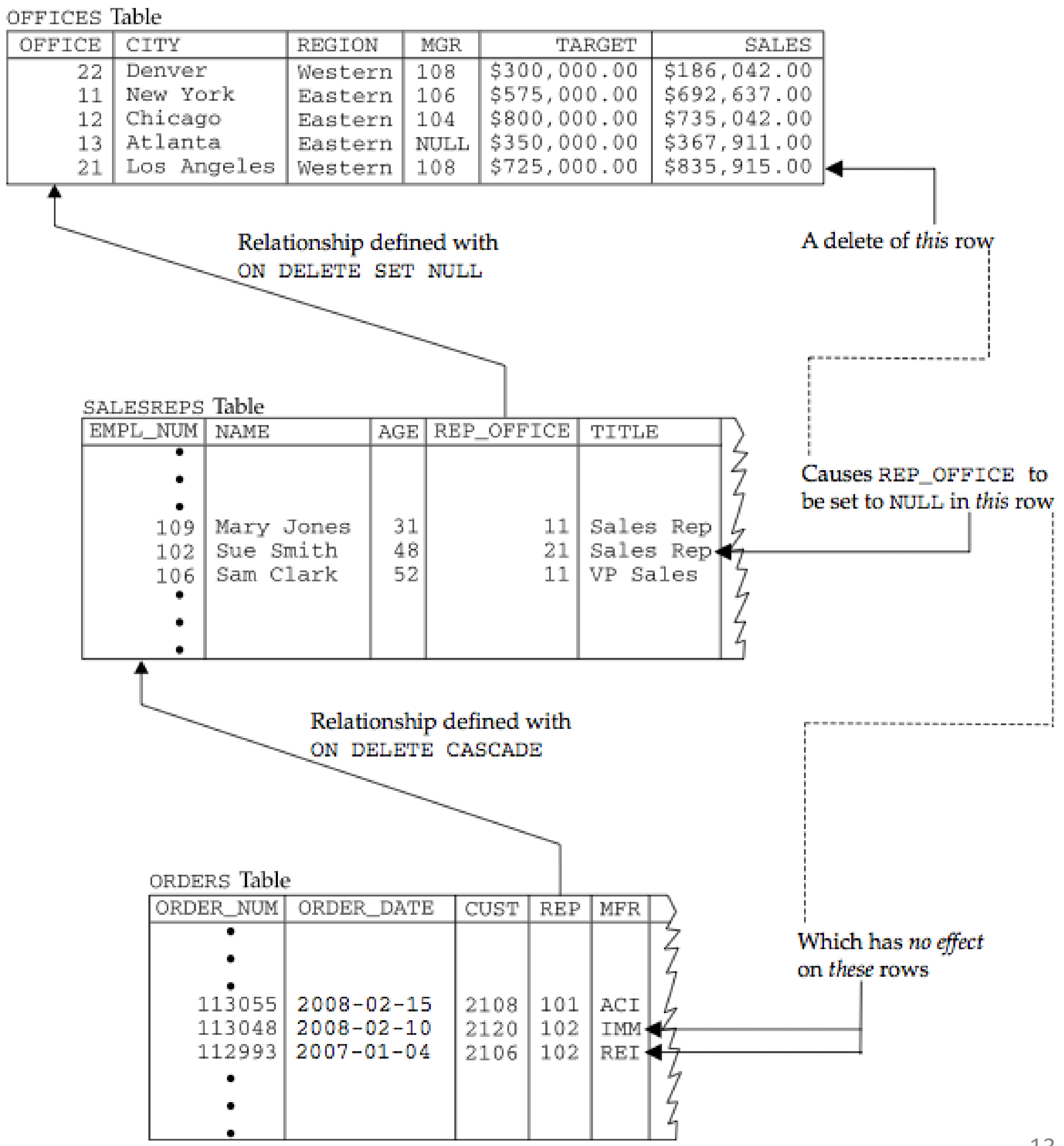
EMPL_NUM	NAME	AGE	REP_OFFICE	TITLE
•				
•				
•				
109	Mary Jones	31	11	Sales Rep
102	Sue Smith	48	21	Sales Rep
106	Sam Clark	52	11	VP Sales
•				
•				
•				

ORDERS Table

ORDER_NUM	ORDER_DATE	CUST	REP	MFR
•				
•				
•				
113055	2008-02-15	2108	101	ACI
113048	2008-02-10	2120	102	IMM
112993	2007-01-04	2106	102	REI
•				
•				
•				

FIGURE 11-3 Two levels of CASCADE rules

Use Referential Constraint with Care!!



OFFICES Table

OFFICE	CITY	REGION	MGR	TARGET	SALES
22	Denver	Western	108	$300,000.00	$186,042.00
11	New York	Eastern	106	$575,000.00	$692,637.00
12	Chicago	Eastern	104	$800,000.00	$735,042.00
13	Atlanta	Eastern	NULL	$350,000.00	$367,911.00
21	Los Angeles	Western	108	$725,000.00	$835,915.00

SALESREPS Table

EMPL_NUM	NAME	AGE	REP_OFFICE	TITLE
•				
•				
•				
109	Mary Jones	31	11	Sales Rep
102	Sue Smith	48	21	Sales Rep
106	Sam Clark	52	11	VP Sales
•				
•				
•				

ORDERS Table

ORDER_NUM	ORDER_DATE	CUST	REP	MFR
•				
•				
•				
113055	2008-02-15	2108	101	ACI
113048	2008-02-10	2120	102	IMM
112993	2007-01-04	2106	102	REI
•				
•				
•				

Relationship defined with ON DELETE SET NULL
A delete of *this* row
Causes REP_OFFICE to be set to NULL in *this* row
Relationship defined with ON DELETE CASCADE
Which has *no effect* on *these* rows

FIGURE 11-4 A combination of delete rules

Advanced Constraint

- How to specify constraints across multiple tables?

  CREATE ASSERTION

e.g. Ensure that an office's target does not exceed the sum of the quotas for its salespeople.

```
CREATE ASSERTION target_valid
CHECK ((OFFICES.TARGET <= SUM(SALESREPS.QUOTA)) AND
        (SALESREPS.REP_OFFICE = OFFICES.OFFICE));
```

- Very few current SQL implementations support assertions ☹
- In theory, assertions could cause a large amount of database processing overhead as they are checked for each statement that might modify the database.
- In practice, the DBMS will analyze the assertion and determine which tables and columns it involves.

TRIGGER

- If A happens, B should happen accordingly.
- Evens that may trigger an action include INSERT, DELETE, and UPDATE.

```
CREATE TRIGGER NEWORDER
    ON ORDERS
   FOR INSERT
    AS UPDATE SALESREPS
          SET SALES = SALES + INSERTED.AMOUNT
         FROM SALESREPS, INSERTED
        WHERE SALESREPS.EMPL_NUM = INSERTED.REP
       UPDATE PRODUCTS
          SET QTY_ON_HAND = QTY_ON_HAND - INSERTED.QTY
         FROM PRODUCTS, INSERTED
        WHERE PRODUCTS.MFR_ID = INSERTED.MFR
          AND PRODUCTS.PRODUCT_ID = INSERTED.PRODUCT;
```

- Advantages and disadvantages