

7

SQL - Data Definition

CSF2600700 - BASIS DATA





Acknowledgements

This slide is a modification to supplementary slide of “Database System”, 7th edition, Elmasri/Navathe, 2015: **Chapter 6: Basic SQL** used in “Basis Data” course in academic years 2018/2019 in the Faculty of Computer Science, Universitas Indonesia.

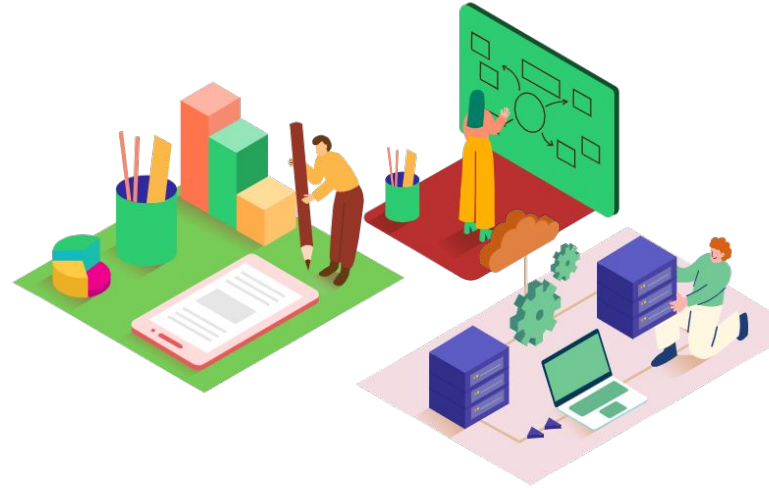
Additional materials from:
<http://www.postgresql.org/docs/current/static/index.html>

Outline

1. SQL Data Definition and Data Types

2. Specifying Constraints in SQL

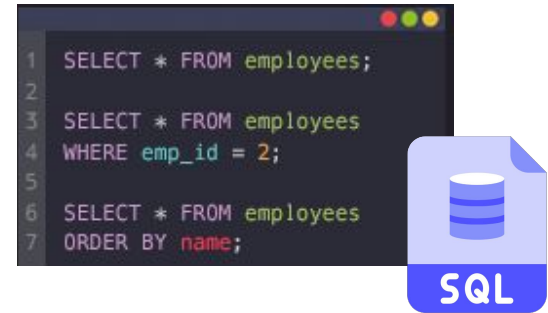
3. Modification Data (INSERT, DELETE, and UPDATE Statements)



Basic SQL

Structured Query Language /'si:kwəl/

- Considered one of the major reasons for the commercial success of relational databases
- **Standardized** – many new feature over time
- **Interactive** via GUI or prompt, or embedded in programs
- **Declarative**, based on relational algebra



SQL DDL and DML

Data Definition Language

Defining the Relational Schema – Relations, Attributes, Domain, The Meta-Data

CREATE table

ALTER table

DROP table

etc.

Data Manipulation Language

Defining the Queries Against the Schema

SELECT

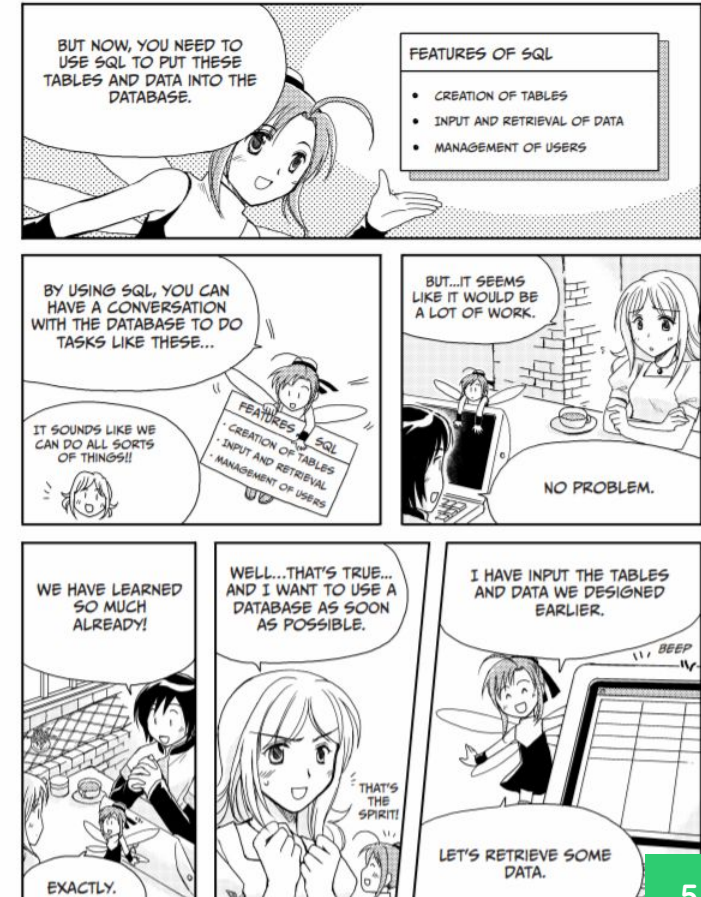
INSERT

UPDATE

DELETE

More commands

Indexes, Constraints, Views, Triggers, Transactions



Domains in SQL

Domain

- It is possible to specify data type directly
- Makes it easier to change the data type for a domain that is used by numerous attributes
- Improves schema readability

Example:

SSN_TYPE in place of CHAR(9), for attribute Ssn, Super_ssn, Mgr_ssn, etc.

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

Schema and Catalog Concept in SQL

SQL Schema

- Group together tables and other constructs that belong to the same database
- Identified by a **schema name**
- Includes an **authorization identifier** and **descriptors** for each element

Schema elements:

Tables, constraints, views, domains, and other constructs

Each statement in SQL ends with a **semicolon** (;)

Create/Drop a Schema

Creating a Schema

Enters new schema into the current database

```
CREATE SCHEMA COMPANY AUTHORIZATION Jsmith;
```

- Create a schema which can be authorized by user Jsmith
- Tables can now be created and added to schema

Dropping a Schema

Remove a schema

```
DROP SCHEMA COMPANY RESTRICT;
```

- Drop operation **fails if schema is not empty**

```
DROP SCHEMA COMPANY CASCADE;
```

- Drop operation **removes everything** in the schema

Data Types in SQL

Each DBMS may have their own DBMS specific data types

Is this good or bad?

Basic Data Types:

Numeric Data Types

- Integer numbers: `INTEGER`, `INT`, and `SMALLINT`
- Arbitrary Precision Numbers: `NUMERIC`
- Floating-point (real) numbers: `FLOAT` or `REAL`, and `DOUBLE PRECISION`

Character-string Data Types

- Fixed length: `CHAR (n)`, `CHARACTER (n)`
- Variable-length with limit: `VARCHAR (n)`, `CHAR VARYING (n)`, `CHARACTER VARYING (n)`
- Variable unlimited length: `TEXT`

Data Types in SQL (contd.)

Bit-string Data Types

- Fixed length: `BIT (n)`
- Varying length: `BIT VARYING (n)`

Boolean Data Types

- Values of `TRUE` or `FALSE` or `NULL`

Date Data Types

- Ten positions
- Components are `YEAR`, `MONTH`, and `DAY` in the form `YYYY-MM-DD`

Data Types in SQL (contd.)

Timestamp Data Types

- Includes the `DATE` and `TIME` fields
 - Plus a minimum of six positions for decimal fractions of seconds
 - Optional `WITH TIME ZONE` qualifier
- Example: '2008-09-27 9:12:47.648302'

Interval Data Types

- Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp

Relation in SQL

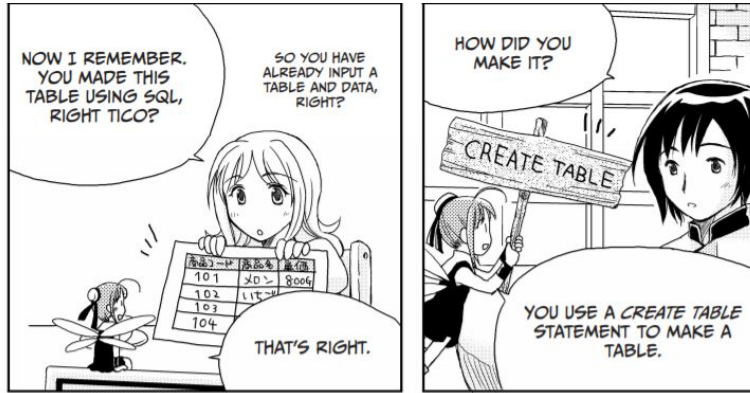
Base relations

- Relation and its tuples are actually created and stored as a file by the DBMS

Virtual relations

- Created through the `CREATE VIEW` statement (Chapter 5)
- May or may not correspond to an actual physical file

Create a TABLE



- Specify a new relation
- Provide **name, attributes, and initial constraints (data types)**
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE EMPLOYEE (Fname VARCHAR(15) NOT NULL, Minit CHAR, ...);
```

- We may specify primary key and foreign keys (more on next slides..)

CREATE TABLE EMPLOYEE

(Fname	VARCHAR(15)	NOT NULL,
Minit	CHAR,	
Lname	VARCHAR(15)	NOT NULL,
Ssn	CHAR(9)	NOT NULL,
Bdate	DATE,	
Address	VARCHAR(30),	
Sex	CHAR,	
Salary	DECIMAL(10,2),	
Super_ssn	CHAR(9),	
Dno	INT	NOT NULL,

PRIMARY KEY (Ssn),

CREATE TABLE DEPARTMENT

(Dname	VARCHAR(15)	NOT NULL,
Dnumber	INT	NOT NULL,
Mgr_ssn	CHAR(9)	NOT NULL,
Mgr_start_date	DATE,	

PRIMARY KEY (Dnumber),

UNIQUE (Dname),

FOREIGN KEY (Mgr_ssn) **REFERENCES** EMPLOYEE(Ssn));

Figure 6.1

SQL CREATE
TABLE data
definition statements
for defining the
COMPANY schema
from Figure 5.7.

```

CREATE TABLE DEPT_LOCATIONS
( Dnumber          INT          NOT NULL,
  Dlocation        VARCHAR(15) NOT NULL,
  PRIMARY KEY (Dnumber, Dlocation),
  FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE PROJECT
( Pname          VARCHAR(15) NOT NULL,
  Pnumber        INT          NOT NULL,
  Plocation      VARCHAR(15),
  Dnum           INT          NOT NULL,
  PRIMARY KEY (Pnumber),
  UNIQUE (Pname),
  FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE WORKS_ON
( Essn          CHAR(9)      NOT NULL,
  Pno           INT          NOT NULL,
  Hours         DECIMAL(3,1) NOT NULL,
  PRIMARY KEY (Essn, Pno),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
  FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );

CREATE TABLE DEPENDENT
( Essn          CHAR(9)      NOT NULL,
  Dependent_name VARCHAR(15) NOT NULL,
  Sex           CHAR,
  Bdate         DATE,
  Relationship   VARCHAR(8),
  PRIMARY KEY (Essn, Dependent_name),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );

```

Figure 6.1
SQL CREATE
TABLE data
definition statements
for defining the
COMPANY schema
from Figure 5.7.

Create a TABLE: Foreign Key

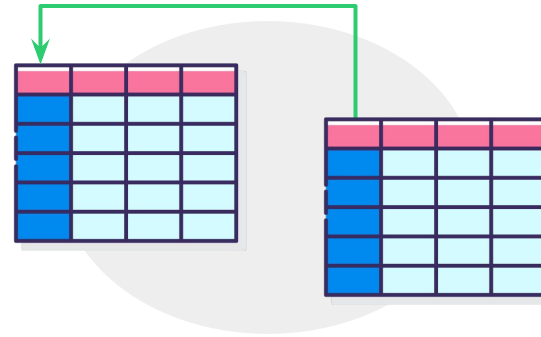
Some foreign keys may cause errors

- **Circular references**
- **Foreign key which refers to the table itself**

Example: Super_ssn refers to the Ssn

- They refer to a table that has not been created

Added later using `ALTER TABLE` statement



ALTER TABLE

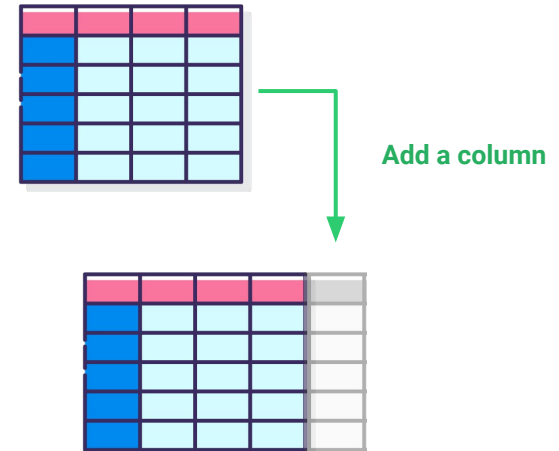
Used to **add an attribute** to one of the base relations
The new attribute will have NULLs in all the tuples of the relation right after the command is executed

The NOT NULL constraint is not allowed for such an attribute

Example:

```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);
```

The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

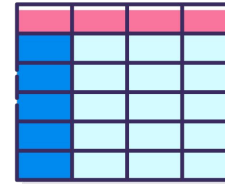


DROP TABLE

Used to **remove** a relation (base table) and its definition
The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

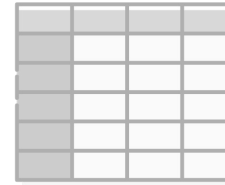
Example:

```
DROP TABLE DEPENDENT;
```





Remove a table

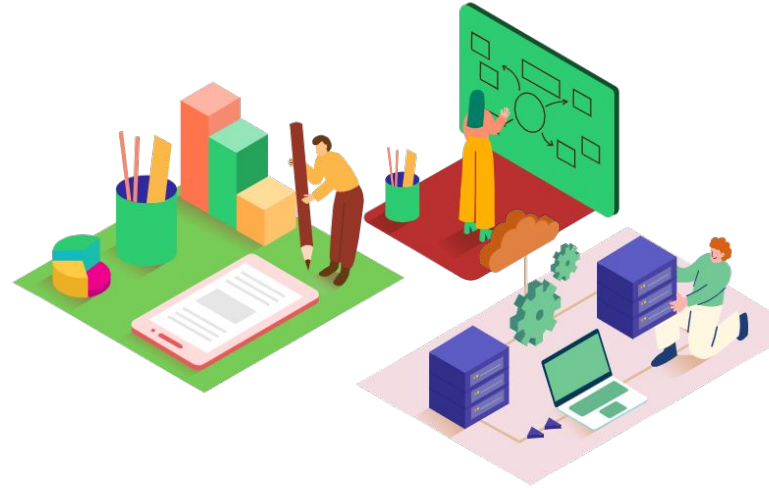


Outline

1. SQL Data Definition and Data Types

2. Specifying Constraints in SQL

3. Modification Data (INSERT, DELETE, and
UPDATE Statements)



Constraints in SQL

Data types are used to limit the data to be stored in a table

There are still many constraints to be handle

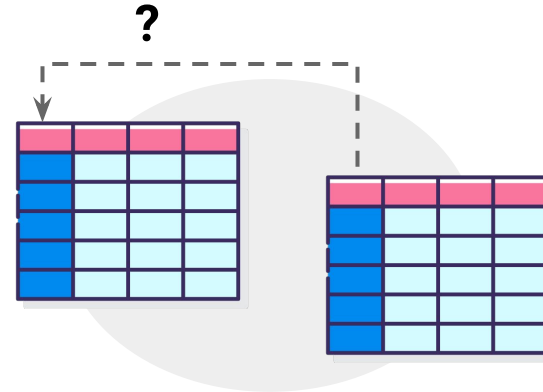
Key constraints

Referential Integrity Constraints

- Other constraints, ex: product price should only accept positive values

SQL allows us to define constraints through DDL

- An error is raised when a constraint is violated



CHECK Constraints

- Most generic constraint type
- Specify that the value in a certain column must satisfy a particular condition

Example:

```
CREATE TABLE PRODUCTS (  
  Product_no INT,  
  Name VARCHAR(10),  
  Price INT CHECK (price > 0) );
```

Constraint may be specified (optional):

```
... Price INT CONSTRAINT POS_PRICE CHECK (price > 0));
```

- Clarifies error messages and allow changes to the constraint
- Can be dropped or replaced with another constraint

NOT NULL Constraints

- Constraint NOT NULL may be specified for a particular attribute
- Implicitly specified for primary keys

Example:

```
CREATE TABLE PRODUCTS (  
  Product_no INT NOT NULL,  
  Name VARCHAR(10) NOT NULL,  
  Price INT);
```

UNIQUE Constraints

Ensure that the data in a column is unique with respect to all the rows in the table

Example:

```
CREATE TABLE PRODUCTS (  
  Product_no INT UNIQUE,  
  Name VARCHAR(10) NOT NULL,  
  Price INT);
```

Primary Key

Combination between unique constraint and NOT NULL constraint

Example:

```
CREATE TABLE PRODUCTS (
  Product_no INT PRIMARY KEY,
  Name VARCHAR(10) NOT NULL,
  Price INT);
```

Another way:

```
CREATE TABLE EXAMPLE (
  A INT,
  B INT,
  C INT,
  PRIMARY KEY (A, C));
```

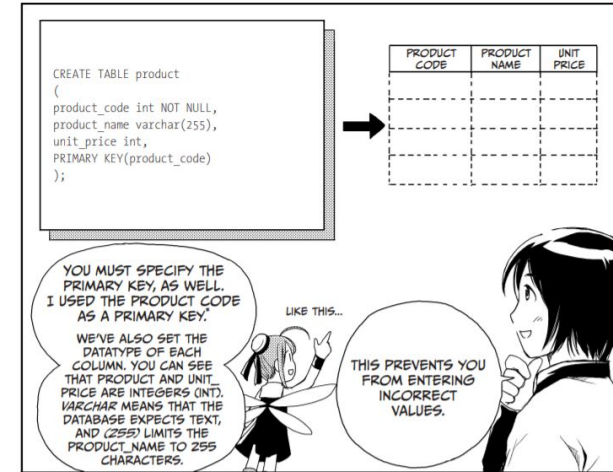


Illustration: Takahashi & Azuma (2016)

Foreign Key

Referential integrity: values in a table must match the values appearing on the other table

Example:

```
CREATE TABLE T1 (  
  A INT PRIMARY KEY,  
  B INT,  
  C INT,  
  FOREIGN KEY (B, C) REFERENCES OTHER_TABLE (C1, C2)  
);
```

Default Values

An attribute can be assigned with default value The value will be created when a new row inserted and no value is specified for that attribute

If no default value is declared explicitly, the default value is NULL

```
CREATE TABLE PRODUCTS (  
  Product_no INT,  
  Name VARCHAR(10) NOT NULL,  
  Price INT DEFAULT 9.99);
```

```

CREATE TABLE EMPLOYEE
(
    ...,
    Dno          INT          NOT NULL      DEFAULT 1,
    CONSTRAINT EMPPK
    PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE SET NULL      ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE SET DEFAULT   ON UPDATE CASCADE);

CREATE TABLE DEPARTMENT
(
    ...,
    Mgr_ssn CHAR(9)          NOT NULL      DEFAULT '888665555',
    ...,
    CONSTRAINT DEPTPK
    PRIMARY KEY (Dnumber),
    CONSTRAINT DEPTSK
    UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
    FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE SET DEFAULT   ON UPDATE CASCADE);

CREATE TABLE DEPT_LOCATIONS
(
    ...,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
        ON DELETE CASCADE      ON UPDATE CASCADE);

```

Figure 6.2

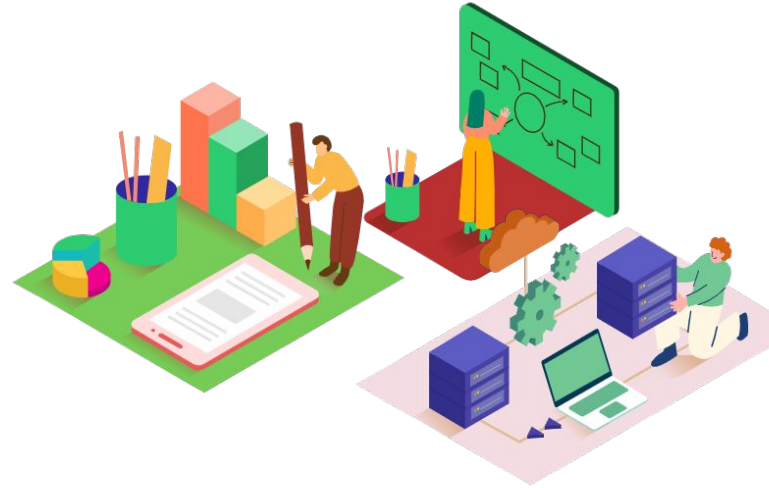
Example illustrating how default attribute values and referential integrity triggered actions are specified in SQL.

Outline

1. SQL Data Definition and Data Types

2. Specifying Constraints in SQL

3. Modification Data (`INSERT`, `DELETE`, and
`UPDATE` Statements)



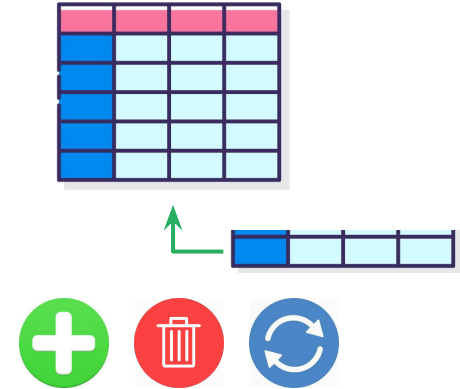
Modifying Database - DML

Three commands to modify the database

INSERT

DELETE

UPDATE



The INSERT Command

In its simplest form, it is used to add one or more tuples to a relation (table)

Attribute values should be listed in the **same order** as the attributes were specified in the CREATE TABLE command

```
INSERT INTO EMPLOYEE
VALUES ('Richard', 'K', 'Marini', '6532988653',
'1962-12-30', '98 Oak Forest, Katy, TX', 'M',
'37000', '653298653', 4);
```

.....U1

Must be listed in same order

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	-----	-------	---------	-----	--------	-----------	-----

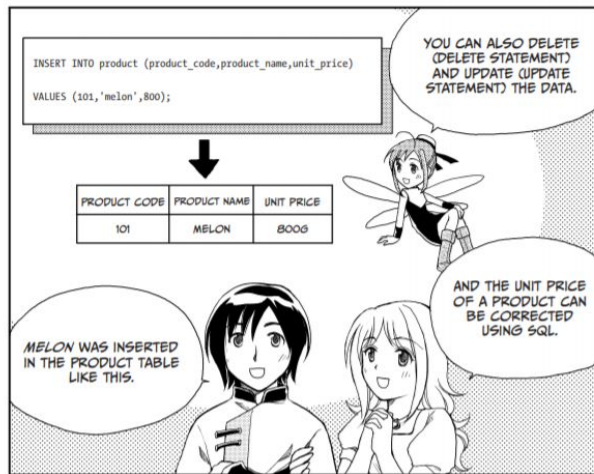
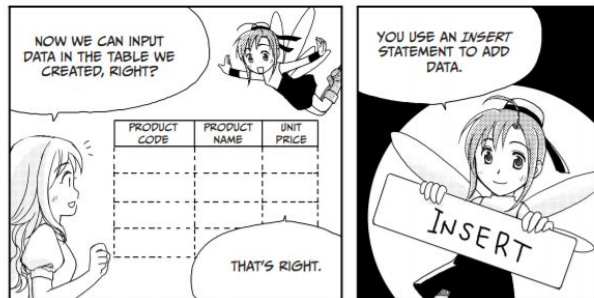


Illustration: Takahashi & Azuma (2016)

The INSERT Command (Contd.)

An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple. Attributes with NULL values can be left out

Example:

Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
VALUES ('Richard', 'Marini', '653298653');
```

.....U1A

Important Note:

Only the constraints **specified in the DDL** commands are automatically enforced by the DBMS when updates are applied to the database

The INSERT Command (Contd.)

Another variation of INSERT allows insertion of multiple tuples resulting from a query into a relation

```
CREATE TABLE DEPTS_INFO  
(DEPT_NAME VARCHAR(10),  
NO_OF_EMPS INTEGER,  
TOTAL_SAL INTEGER);
```

.....U3A

```
INSERT INTO DEPTS_INFO (DEPT_NAME,  
NO_OF_EMPS, TOTAL_SAL)  
SELECT DNAME, COUNT (*), SUM (SALARY)  
FROM DEPARTMENT, EMPLOYEE  
WHERE DNUMBER = DNO  
GROUP BY DNAME;
```

.....U3B

The INSERT Command (Contd.)



The DEPTS_INFO table **may not be up-to-date** if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations after issuing U3B. We have to create a view (see later) to keep such a table up to date.

The DELETE Command

Removes tuples from a relation.

Includes a WHERE clause to select the tuples to be deleted

```
DELETE FROM EMPLOYEE WHERE LName = 'Brown';
```

.....U4A

```
DELETE FROM EMPLOYEE WHERE Ssn = '123456789';
```

.....U4B

```
DELETE FROM EMPLOYEE WHERE Dno = 5;
```

.....U4C

```
DELETE FROM EMPLOYEE;
```

.....U4D

The UPDATE Command

Modify attribute values of one or more selected tuples

Additional SET clause in the UPDATE command

→ **Specifies attributes to be modified and new values**

```
UPDATE PROJECT  
SET Plocation = 'Bellaire', Dno, = 5  
WHERE Pnumber = 10;
```

.....U5

UPDATE

Example:

Give all employees in the 'Research' department a 10% raise in salary.

```
UPDATE EMPLOYEE
SET Salary = Salary * 1.1
WHERE Dno IN (SELECT DNUMBER
              FROM DEPARTMENT
              WHERE DNAME='Research');
```

In this request, the modified SALARY value depends on the original SALARY value in each tuple.

The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Q&A

