THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Week 3: SQL

Database Systems & Team Projects

Gabe

ed from Prof. Bhagi Narahari; Silberschatz, Korth, and Sudarshan; Wood, and Ramakrishnan, Gerhke, and Lawrence

Last time…

Relational Model & Flask

SQL DDL

SQL DML

this time…

SQL: Structured Query Language

The standard language for relational data

- Invented by folks at IBM, esp. Don Chamberlin
- Actually not a great language…
- Beat a more elegant competing standard, QUEL, from Berkeley

Separated into a DML & DDL

Data Definition (**DDL**) – to define schema/tables

- Define Schema
- Define Constraints

Data Manipulation (**DML**) component based on relational algebra & calculus

SQL Hello World

Create a table

```
CREATE TABLE Instructor (
  id INT,
  name VARCHAR(50),
  department VARCHAR(5),
  office DECIMAL(4),
  PRIMARY KEY (id)
);
```

SQL Hello World

Create a table

```
CREATE TABLE Instructor (
  id INT,
  name VARCHAR(50),
  department VARCHAR(5),
  office DECIMAL(4),
  PRIMARY KEY (id)
);
```

Insert a row

```
INSERT INTO Instructor (id, name, department, office) VALUES (2,
'Roberto Finlaison', 'CIVIL', 185);
```

SQL Hello World

Create a table

```
CREATE TABLE Instructor (
  id INT,
  name VARCHAR(50),
  department VARCHAR(5),
  office DECIMAL(4),
  PRIMARY KEY (id)
);
```

Insert a row

```
INSERT INTO Instructor (id, name, department, office) VALUES (2,
'Roberto Finlaison', 'CIVIL', 185);
```

Query a table

```
SELECT * FROM Instructor;
id          name             department  office
----------  ---------------  ----------  ----------
1           Modesty Tournay  MENG        173
2           Roberto Finlais  CIVIL       185
```

SQL Basic Rules…read up on SQL syntax

There is a set of **reserved words** that cannot be used as names for database objects. (e.g. SELECT, FROM, WHERE)

SQL is **case-insensitive**.

- Only exception is string constants. 'FRED' not the same as 'fred'.

SQL is free-format and white-space is ignored.

The semi-colon is often used as a statement terminator, although that is not always required.

Date and time constants have defined format:

- Dates: 'YYYY-MM-DD' e.g. '1975-05-17'
- Times: ‘hh:mm:ss[.f] ' e.g. '15:00:00'
- Timestamp: ‘YYYY-MM-DD hh:mm:ss[.f] ' e.g. ‘1975-05-17 15:00:00'

Two single quotes '' are used to represent a single quote character in a character constant. e.g. 'Master''s'.

SQL DDL

SQL data definition language (DDL) allows users to:

- add, modify, and drop tables
- define and enforce integrity constraints
- enforce security restrictions
- Create views

Common commands:

- CREATE TABLE
- ALTER TABLE
- DROP TABLE

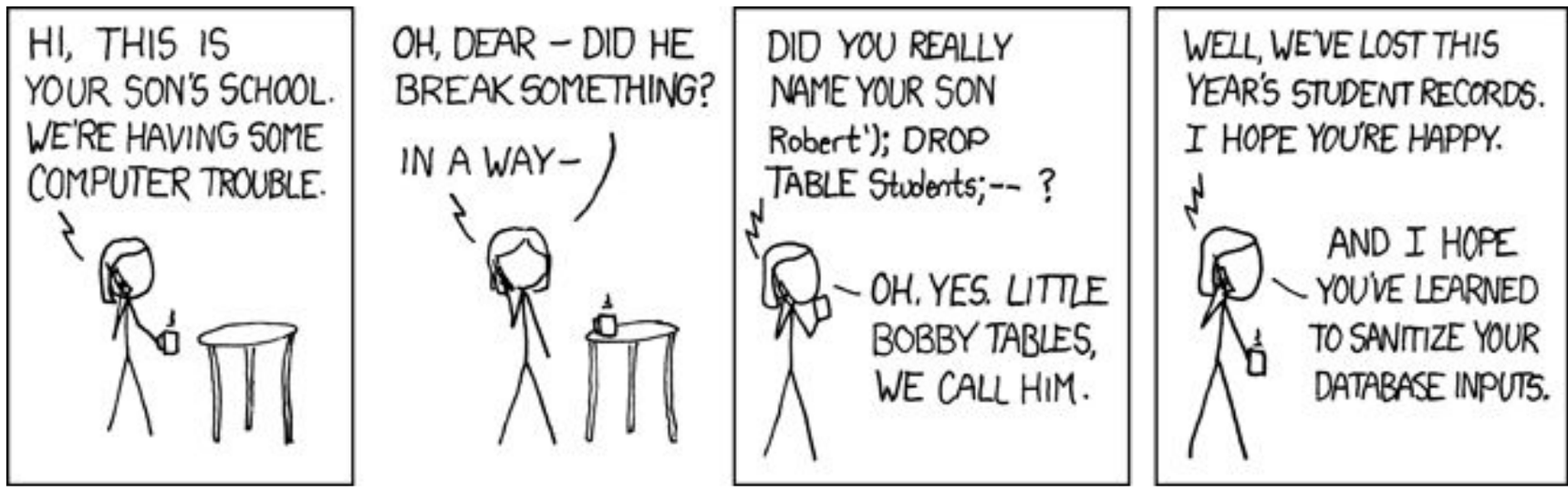
SQL DDL

SQL data definition language (DDL) allows users to:

- add, modify, and drop tables
- define and enforce integrity constraints
- enforce security restrictions
- Create views

Common commands:

- CREATE TABLE
- ALTER TABLE
- **DROP TABLE**



SQL Identifiers and Data types

Identifiers are used to identify objects in the database such as tables, views, and columns.

- The identifier is the name of the database object.
- Rules for SQL identifiers…
- Note: Quoted or **delimited identifiers** enclosed in double quotes allow support for spaces and other characters. E.g. "select“

Data types: each attribute has associated domain of values – i.e., each column has data type

- The DBMS can perform implicit data type conversion when necessary
- Can also do explicit conversion using CAST and CONVERT

SQL also supports user defined data types

- CREATE DOMAIN
- Similar to typedef in C ?

SQL Data Types…similar to prog lang

BOOLEAN	TRUE or FALSE
CHAR	Fixed length string (padded with blanks) e.g. CHAR(10)
VARCHAR	Variable length string e.g. VARCHAR(50)
BIT	Bit string e.g. BIT(4) can store '0101'
NUMERIC or DECIMAL	Exact numeric data type e.g. NUMERIC(7,2) has a precision (max. digits) of 7 and scale of 2 (# of decimals) e.g. 12345.67
INTEGER	Integer data only
SMALLINT	Smaller space than INTEGER
FLOAT or REAL	Approximate numeric data types. Precision dependent on implementation.
DOUBLE PRECISION	
DATE	Stores YEAR, MONTH, DAY
TIME	Stores HOUR, MINUTE, SECOND
TIMESTAMP	Stores date and time data.
INTERVAL	Time interval.
CHARACTER LARGE OBJECT	Stores a character array (e.g. for a document)
BINARY LARGE OBJECT	Stores a binary array (e.g. for a picture, movie)

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

SQL CREATE TABLE

The **CREATE TABLE** command is used to create a table in the database. A table consists of a table name, a set of fields with their names (ids) and data types, and specified constraints.

The general form is:

```
CREATE TABLE tableName (
    attr1Name attr1Type  [attr1_constraints],
    attr2Name attr2Type  [attr2_constraints],
   ...
    attrMName attrMType  [attrM_constraints],
    [primary and foreign key constraints]
);
```

SQL CREATE TABLE Example

The **CREATE TABLE** command for the **Emp** relation:

```
CREATE TABLE emp (
    eno       CHAR(5),
    ename     VARCHAR(30) NOT NULL,
    bdate     DATE,
    title     CHAR(2),
    salary    DECIMAL(9,2),
    supereno  CHAR(5),
    dno   CHAR(5),
);
```

SQL Constraints - Entity Integrity

Entity Integrity constraint: The primary key of a table must contain a unique, non-null value for each row. The primary key is specified using the **PRIMARY KEY** clause.

- e.g. **PRIMARY KEY** (eno) (for Emp relation)
- e.g. **PRIMARY KEY** (eno, pno) (for WorksOn relation)
- It is also possible to use **PRIMARY KEY** right after defining the attribute in the **CREATE TABLE** statement.

There can only be one primary key per relation, other candidate keys can be specified using **UNIQUE**:

- e.g. **UNIQUE (ename)**

Another Example…'mini-banner'

Create **Students** table

- Info on students
- Every student has unique Student ID (**sid**)

```
CREATE TABLE Students (
 sid: CHAR(20),
 name: CHAR(20),
 login: CHAR(10),
 age: INTEGER,
 gpa: REAL,
 PRIMARY KEY (sid)
);
```

Enrollment table holds information about courses that students take

- **What uniqueness property do we want?**

```
CREATE TABLE Enrollment (
 sid: CHAR(20),
 cid: CHAR(20),
 grade: CHAR(2)
);
```

Enrollment Keys

Both of these schemas guarantee that:

- "For a given student and course, there is a single grade"

Do they both meet the logic we actually expect?

```
CREATE TABLE Enrolled1 (
  sid CHAR(20)
  cid  CHAR(20),
  grade CHAR(2),
  PRIMARY KEY  (sid,cid)
)
```

```
CREATE TABLE Enrolled2(
  sid CHAR(20)
  cid  CHAR(20),
  grade CHAR(2),
  PRIMARY KEY  (sid),
  UNIQUE (cid, grade)
)
```

Enrollment Keys

Enrolled1: “For each unique student and course combination, there can only be one grade.”

vs.

```
CREATE TABLE Enrolled1 (
  sid CHAR(20)
  cid  CHAR(20),
  grade CHAR(2),
  PRIMARY KEY  (sid,cid)
)
```

Enrolled 2: “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Enrolled2(
  sid CHAR(20)
  cid  CHAR(20),
  grade CHAR(2),
  PRIMARY KEY  (sid),
  UNIQUE (cid, grade)
)
```

Another example

What if instead of having a grade, we want to record a student's "rank" in the class?

- Best student, 2nd best, 3rd best, etc…

Do these meet the logic we want?

```
CREATE TABLE Enrolled3 (
  sid CHAR(20)
  cid  CHAR(20),
  rank INT,
  PRIMARY KEY (sid,cid),
  UNIQUE (rank)
)
```

```
CREATE TABLE Enrolled4 (
  sid CHAR(20)
  cid  CHAR(20),
  rank INT,
  PRIMARY KEY (sid,cid,rank)
)
```

Another example

What if instead of having a grade, we want to record a student's "rank" in the class?

- Best student, 2nd best, 3rd best, etc…

Prevents students in different classes from having the same rank!

```
CREATE TABLE Enrolled3 (
  sid CHAR(20)
  cid  CHAR(20),
  rank INT,
  PRIMARY KEY (sid,cid),
  UNIQUE (rank)
)
```

```
CREATE TABLE Enrolled4 (
  sid CHAR(20)
  cid  CHAR(20),
  rank INT,
  PRIMARY KEY (sid,cid,rank)
)
```

Allows different students to have the same rank!

Another example

What if instead of having a grade, we want to record a student's "rank" in the class?

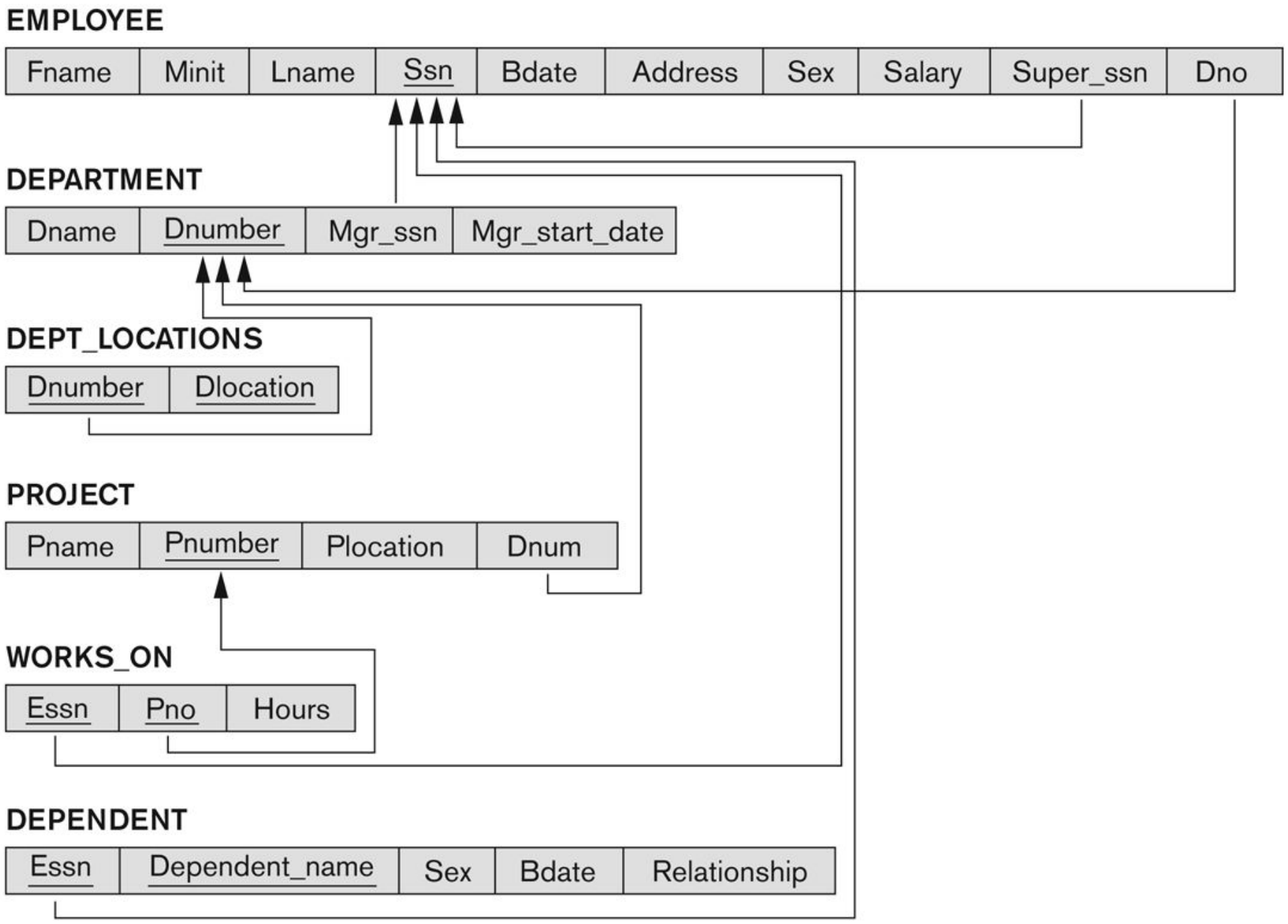
- Each student/course pair should get a rank
- Rank only needs to be unique within a class

```
CREATE TABLE Enrolled5 (
  sid CHAR(20)
  cid  CHAR(20),
  rank INT,
  PRIMARY KEY  (sid,cid)
  UNIQUE (cid, rank)
)
```

Need to be careful about Integrity Constraints!

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



SQL Constraints - Referential Integrity

Referential integrity constraint - Defines a foreign key that references the primary key of another table.

- If a foreign key contains a value that is not `NULL`, that value must be present in some tuple in the relation containing the referenced primary key.

Example: `Workson` contains two foreign keys:

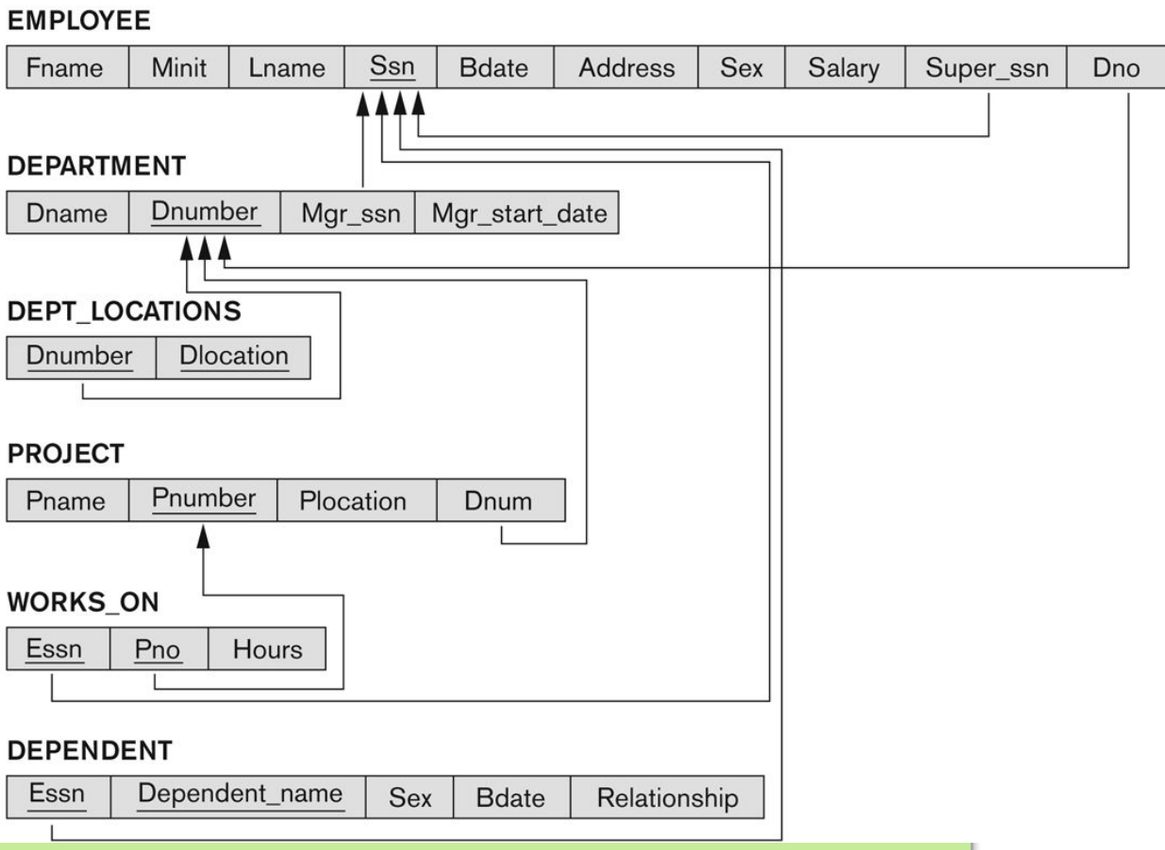
- `workson.essn` references `emp.ssn`
- `workson.pno` references `proj.pnumber`

Specify foreign keys using `FOREIGN KEY` syntax:

```
FOREIGN KEY (essn) REFERENCES emp(ssn)
```

SQL Referential Integrity Example

The **CREATE TABLE** command for the **workson** relation:




```
CREATE TABLE workson (
    essn            CHAR(5),
    pno             CHAR(5),
    resp            VARCHAR(20),
    hours           SMALLINT,
    PRIMARY KEY (essn,pno),
    FOREIGN KEY (essn) REFERENCES employee(ssn),
    FOREIGN KEY (pno) REFERENCES project(pnumber)
);
```

SQL Referential Integrity and Updates

When you try to `INSERT` or `UPDATE` a **row in a relation containing a foreign key** (e.g. `workson`) that operation is rejected if it violates referential integrity.

When you `UPDATE` or `DELETE` a **row in the referenced primary key relation** (e.g. `emp` or `proj`), you have the option on what happens to the values in the foreign key relation (`workson`):

- `CASCADE` - Delete (update) values in foreign key relation when primary key relation has rows deleted (updated).
- `SET NULL` - Set foreign key fields to `NULL` when corresponding primary key relation row is deleted.
- `SET DEFAULT` - Set foreign key values to their default value (if defined).
- `NO ACTION` - Reject the request on the parent table.

SQL Referential Integrity Updates

```
CREATE TABLE workson (
   eno  CHAR(5),
   pno      CHAR(5),
   respVARCHAR(20),
   hours  SMALLINT,
   PRIMARY KEY (eno,pno),
   FOREIGN KEY (eno) REFERENCES emp(eno)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE,
 FOREIGN KEY (pno) REFERENCES proj(pnumber)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE
);
```

What properties does this provide?

workson table

eno	pno	resp	hours
123	2	Manager	23
456	4	Admin	15
789	6	Programmer	35

emp table

eno	name	…
123	T. Wood	…
456	L. Chaufournier	…
789	C. Meadows	…

SQL Referential Integrity Updates

```
CREATE TABLE workson (
   eno  CHAR(5),
   pno      CHAR(5),
   respVARCHAR(20),
   hours  SMALLINT,
   PRIMARY KEY (eno,pno),
   FOREIGN KEY (eno) REFERENCES emp(eno)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE,
 FOREIGN KEY (pno) REFERENCES proj(pnumber)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE
);
```

You don't want to delete an employee who is still working on a project… must delete from **WorksOn** first

workson table

eno	pno	resp	hours
123	2	Manager	23
456	4	Admin	15
789	6	Programmer	35

emp table

eno	name	…
123	T. Wood	…
456	L. Chaufournier	…
789	C. Meadows	…

More Referential Integrity Updates

The **CREATE TABLE** command for the **Emp** relation:

```
CREATE TABLE emp (
    eno       CHAR(5),
   name     VARCHAR(30) NOT NULL,
   bdate      DATE,
   title      CHAR(2),
   salary DECIMAL(9,2),
   supereno CHAR(5),
   dno     CHAR(5),
   PRIMARY KEY (eno),
   FOREIGN KEY (dno) REFERENCES dept(dno)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

What properties does this provide?

emp table

eno	name	…	dno
123	T. Wood	…	AB12
456	L. Chaufournier	…	EF56
789	C. Meadows	…	EF56

dept table

dnumber	dname	…
AB12	Cloud Systems	…
CD34	Marketing	…
EF56	Web Devel	…

More Referential Integrity Updates

The **CREATE TABLE** command for the **Emp** relation:

```
CREATE TABLE emp (
    eno       CHAR(5),
   name     VARCHAR(30) NOT NULL,
   bdate      DATE,
   title      CHAR(2),
   salary DECIMAL(9,2),
   supereno CHAR(5),
   dno     CHAR(5),
   PRIMARY KEY (eno),
   FOREIGN KEY (dno) REFERENCES dept(dno)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

If a department is deleted, do not fire the employee

emp table

eno	name	…	dno
123	T. Wood	…	AB12
456	L. Chaufournier	…	EF56
789	C. Meadows	…	EF56

dept table

dnumber	dname	…
AB12	Cloud Systems	…
CD34	Marketing	…
EF56	Web Devel	…

Database Updates

Database updates such as inserting rows, deleting rows, and updating rows are performed using their own statements.

INSERT

UPDATE

DELETE

Database Updates

Insert is performed using the INSERT command:

```
INSERT INTO tableName [(column list)]
VALUES (data value list)
```

Examples:

```
INSERT INTO emp VALUES ('E9','S. Smith',DATE '1975-03-05',
                        'SA',60000,'E8','D1');
```

```
INSERT INTO proj (pno, pname) VALUES ('P6','Programming');
```

Note: If column list is omitted, values must be specified in order they were created in the table. If any columns are omitted from the list, they are set to NULL.

DDL Summary

SQL contains a data definition language that allows you to **CREATE**, **ALTER**, and **DROP** database objects such as tables, triggers, indexes, schemas, and views.

Constraints are used to preserve the integrity of the database:

- **CHECK** can be used to validate attribute values (only available in recent MySQL versions)
- Entity Integrity constraint - The primary key of a table must contain a unique, non-null value for each row.
- Referential integrity constraint - Defines a foreign key that references a unique key of another table.

INSERT, **DELETE**, and **UPDATE** commands modify the data stored within the database (thus they are part of DML)