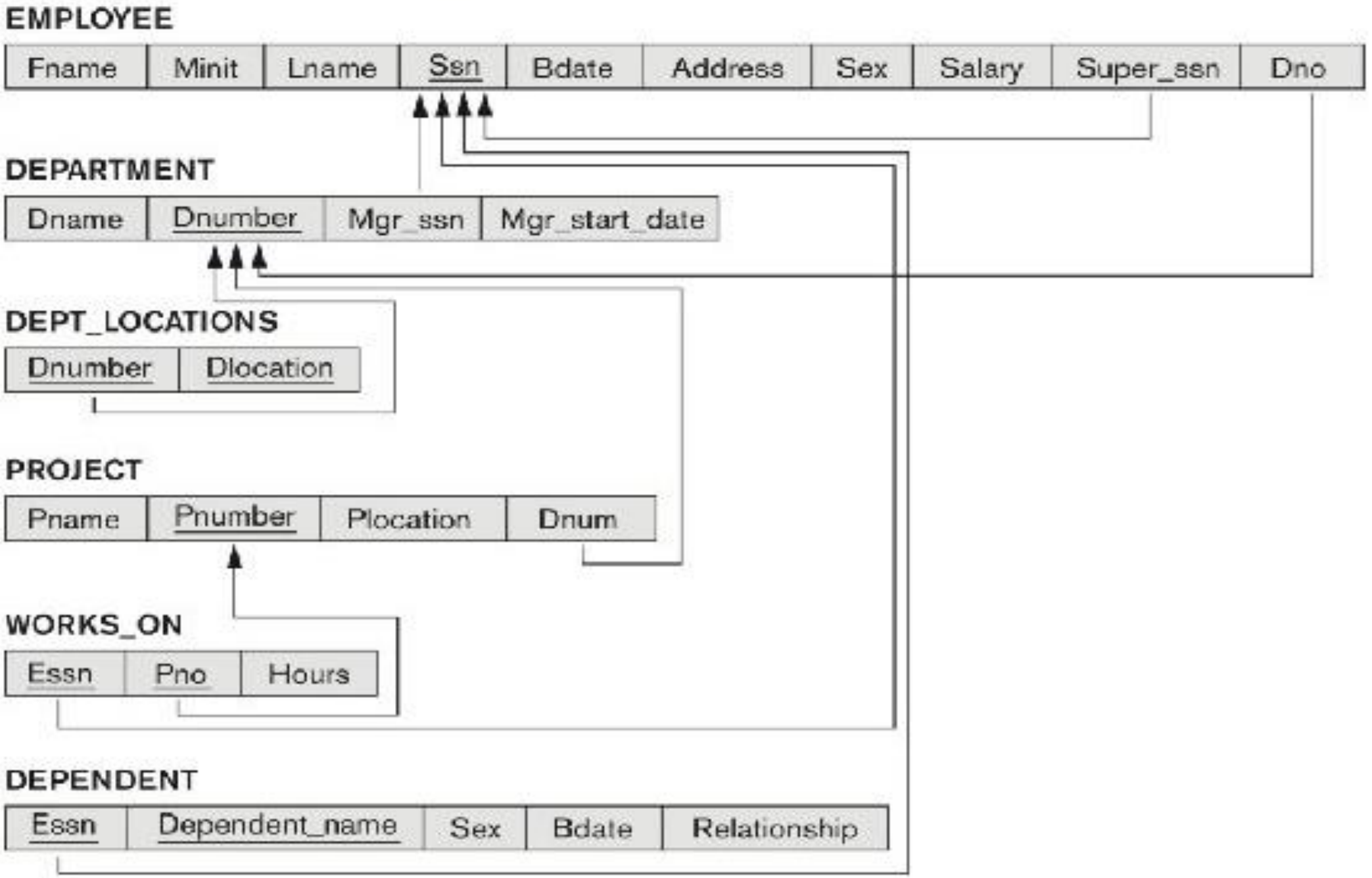
Advanced Database


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

CREATE Schema In SQL

- The following statement creates a schema called COMPANY, owned by the user with authorization identifier ‘Jsmith’.

CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

In general, not all users are authorized to create schemas and schema elements. The privilege to create schemas, tables, and other constructs must be explicitly granted to the relevant user accounts by the system administrator or DBA.

The CREATE TABLE Command in SQL

- The **CREATE TABLE** command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints.
- The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and any attribute constraints, such as NOT NULL.
- The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command .

CREATE TABLE COMPANY.EMPLOYEE ...

rather than

CREATE TABLE EMPLOYEE .

Attribute Data Types and Domains in SQL

- **Numeric** data types include integer numbers of various sizes (INTEGER or INT, and SMALLINT) and floating-point (real).
- **Character-string** data types are either fixed length—CHAR(n) or CHARACTER(n), where n is the number of characters—or varying length— VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n), where n is the maximum number of characters.
- A **Boolean** data type has the traditional values of TRUE or FALSE.
- The **DATE** data type has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD.

```
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),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
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) );
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) );
```

Specifying Constraints in SQL

- Because SQL allows NULLs as attribute values, a constraint **NOT NULL** may be specified if NULL is not permitted for a particular attribute.
- It is also possible to define a default value for an attribute by appending the clause **DEFAULT** to an attribute definition.
- default default value is NULL for attributes that do not have the NOT NULL constraint.
- Another type of constraint can restrict attribute or domain values using the **CHECK** clause following an attribute or domain definition.
- Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);
- The **PRIMARY KEY** clause specifies one or more attributes that make up the primary key of a relation.

Specifying Constraints in SQL

- For example, the primary key of DEPARTMENT can be specified as follows:

Dnumber INT PRIMARY KEY;

- The **UNIQUE** clause can also be specified directly for a secondary key if the secondary key is a single attribute, as in the following example:

Dname VARCHAR(15) UNIQUE;

- Referential integrity is specified via the FOREIGN KEY clause. A referential integrity constraint can be violated when tuples are inserted or deleted, or when a foreign key or primary key attribute value is modified.
- The schema designer can specify an alternative action to be taken by attaching a referential triggered action clause to any foreign key constraint. The options include:
  - **SET NULL**
  - **CASCADE**
  - **SET DEFAULT**.

Specifying Constraints in SQL

- An option must be qualified with either **ON DELETE or ON UPDATE**. Here, the database designer chooses ON DELETE SET NULL and ON UPDATE CASCADE for the foreign key Super_ssn of EMPLOYEE.
- Other table constraints can be specified through additional **CHECK** clauses at the end of a CREATE TABLE statement. These can be called tuple-based constraints because they apply to each tuple individually and are checked whenever a tuple is inserted or modified.
- For the DEPARTMENT table make sure that a manager's start date is later than the department creation date: CHECK (Dept_create_date <= Mgr_start_date);

```
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);
```