

SQL Data Definition Language (DDL)

Outline

- SQL DDL
 - Data Definition
 - Data Types
 - Constraints



Structured Query Language (SQL)

- SQL language: Considered one of the major reasons for the commercial success of relational databases
- The origin of SQL is relational predicate calculus called tuple calculus which was proposed initially as the language SQUARE.
- SQL was designed and implemented at IBM to interface with System R
- SQL actually comes from the word “SEQUEL” which was the original term used in the paper: “SEQUEL TO SQUARE” by Chamberlin and Boyce. IBM could not copyright that term, so they abbreviated to SQL and copyrighted the term SQL.
- Now popularly known as “Structured Query language”.
- SQL is an informal or practical rendering of the relational data model with syntax

SQL Standards

- SQL Standard – ANSI, ISO
- SQL has gone through many standards: starting with SQL-86 or SQL 1.A. SQL-92 is referred to as SQL-2.
- Later standards (from SQL-1999) are divided into **core** specification and specialized **extensions**. The extensions are implemented for different applications – such as data mining, data warehousing, multimedia etc.
- SQL-2006 added XML features; 2008 added Object-oriented features; 2011 added temporal data and windows; 2016 added polymorphic table functions, JSON, high precision decimal data type DECFLOAT, row pattern matching;
- SQL-3 is the current standard which started with SQL-1999. It is not fully implemented in any RDBMS.
- **ISO/IEC JTC 1/SC 32** Data management and interchange Committee in-charge of the SQL standards ISO/IEC 9075-1
 - <https://www.iso.org/committee/45342.html>
- SQLJ, Pro*C, etc

Useful References

- Basics Book: ORACLE 10g PROGRAMMING, A Primer
 - <http://tinman.cs.gsu.edu/~raj/books/oracle10-primer.html>
- <https://livesql.oracle.com>
- <http://sqlzoo.net/>
- <http://asktom.oracle.com>
- <https://docs.oracle.com>
 - <https://docs.oracle.com/en/database/oracle/oracle-database/index.html>
- We will be using Oracle RDBMS in this course



DATA DEFINITION AND DATA TYPES

Data Definition in SQL

- Terminology:
 - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- **CREATE** statement
 - Main SQL command for data definition
- The language has features for: Data definition, Data Manipulation, Transaction control (Transact-SQL, Ch 20), Indexing (Ch 17), Security specification (Grant and Revoke- see Ch 30), Active databases (Ch 26), Multi-media (Ch 26), Distributed databases (Ch 23) etc.



Schema and Catalog Concepts in SQL

■ SQL schema

- Identified by a schema **name** and has an **owner id**
- Includes **descriptors** for each **element**
 - **Tables, constraints, views**, domains, and other constructs
 - *Oracle supports*: tables, indexes, views, partitions, sequences, dimensions, synonyms, procedures, etc.

■ RDBMS SQL Catalog

- Named collection of schemas in an SQL environment
- SQL also has the concept of a cluster of catalogs.

Schema and Catalog Concepts in SQL (cont'd.)

- `CREATE SCHEMA` statement

```
CREATE SCHEMA DBCourse AUTHORIZATION 'radaikka';
```

- Usually DBA creates the schema

- create cluster, procedure, sequence, session, synonym, table, trigger, view

- Each statement in SQL ends with a **semicolon “;”**

- SQL Statements are **NOT** case sensitive

The CREATE TABLE Statement in SQL

- Create a new relation (table)
 - Provide name
 - Specify attributes, their types, and initial constraints
- Can optionally specify schema:

```
CREATE TABLE DBCourse.Employee ...
```

or

```
CREATE TABLE Employee...
```

The CREATE TABLE Statement in SQL

- **Base tables (base relations)**

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

- **Virtual relations (views)**

- Created through the `CREATE VIEW` statement. Do not correspond to any physical file.



Running Example: COMPANY Relational Database Schema

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

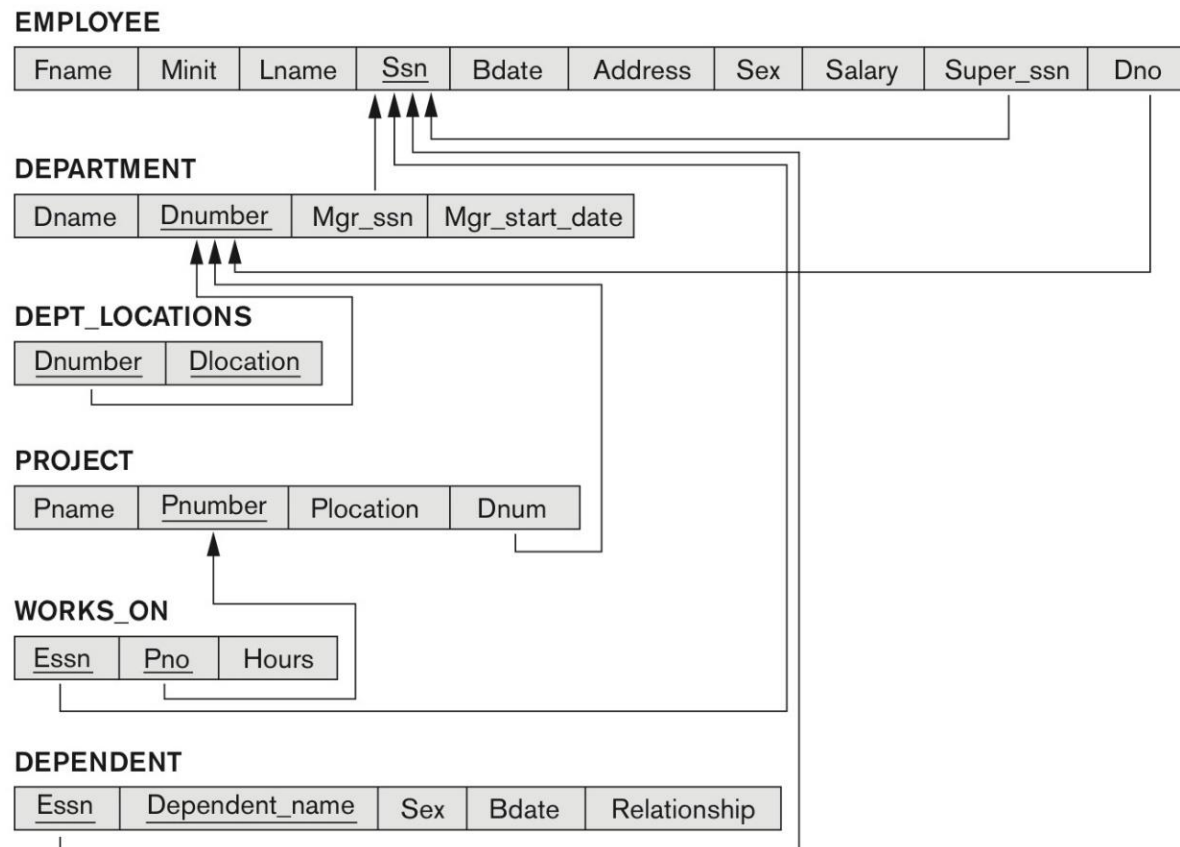


Figure 5.6 One Possible Database State for the COMPANY Relational Database Schema (1 of 2)

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Figure 5.6 One Possible Database State for the COMPANY Relational Database Schema (2 of 2)

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

<u>Pname</u>	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse



Common ORACLE Data types

(Not all ANSI data types are supported)

■ Characters

- `CHAR (size)`: default 1; if stored value is < size, then pad with blank space
- `VARCHAR2 (size [BYTE | CHAR] )`; old version -> `VARCHAR(size)`
 - `n = 1 to [4000|32767] bytes; n = 1 to [4000|32767] characters;`
 - 4000 if `MAX_STRING_SIZE = STANDARD`
 - 32767 if `MAX_STRING_SIZE = EXTENDED`
- `NCHAR (size), NVARCHAR2 (size)`: **UNICODE version**
- **Large Objects (LOB):**
 - `CLOB, NLOB` : Character; Max size is $(4 \text{ GB} - 1) * (\text{db block size})$
 - `BLOB` : Binary ; Max size is $(4 \text{ GB} - 1) * (\text{db block size})$
 - `BFILE` : Binary ; ; Max size is 4 GB
- **LONG**: up to 2 gigabytes-1 (one per table; only for backward compatibility; use LOBs)

Common ORACLE Data types

(Not all ANSI data types are supported)

■ Numeric

- precision (p) up to 39 or 40 decimal digits; scale (s) range from -84 to 127
 - E.g., 123.22; p = 3 and s = 2
- `NUMBER ( p , s ) ; DECIMAL ;`
 - 123.89 -> NUMBER -> 123.89
 - 123.89 -> NUMBER(3) -> 124
 - 123.89 -> NUMBER(4,2) -> exceeds precision
 - 123.89 -> NUMBER(5,2) -> 123.89
 - 123.89 -> NUMBER(6,1) -> 123.9
 - 123.89 -> NUMBER(6,-2) -> 100
 - .000127 -> NUMBER(4,5) -> .00013
- For INTEGER you can use NUMBER (p)
- `BINARY_FLOAT, BINARY_DOUBLE; FLOAT` is also available;

Common ORACLE Data types

(Not all ANSI data types are supported)

- The datetime data types are `DATE`, `TIMESTAMP` (has fractional seconds), `TIMESTAMP WITH TIME ZONE`, and `TIMESTAMP WITH LOCAL TIME ZONE`.
 - Jan 1, 4712 BCE through December 31, 9999 CE (Common Era, or 'AD')
- The interval data types are `INTERVAL YEAR TO MONTH` and `INTERVAL DAY TO SECOND`. Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp
 - `INTERVAL YEAR TO MONTH` stores a period of time using the `YEAR` and `MONTH` datetime fields.
 - This data type is useful for representing the difference between two datetime values when only the year and month values are significant.
- `DATE`, `TIME`, `Timestamp`, `INTERVAL` data types can be **cast** or converted to string formats for comparison.
- `Boolean` in Oracle: `flag char(1) check (flag in ( 'Y', 'N' ))`
- More... Check docs.oracle.com

The CREATE Domain Command in SQL

- DOMAIN

- User defined; Used as data type for attributes;
- Makes it easier to change the data type for a domain that is used by numerous attributes
- Improves schema readability

```
CREATE DOMAIN SSN_Type VARCHAR2 (10) ;
```

- Not directly supported by Oracle, etc
 - User Defined Data types in Oracle – are created as objects and cannot be used as Primary Key attribute

- CREATE OR REPLACE TYPE ...

```
CREATE TYPE ssn_type as OBJECT (ssn_col  
char (11) ) ;
```



Example: Company Database

Create account in livesql.oracle.com

```
CREATE TABLE Employee
(
    fname          VARCHAR2 (15) ,
    minit          VARCHAR2 (1) ,
    lname          VARCHAR2 (15) ,
    ssn            CHAR (9) ,
    bdate          DATE ,
    address        VARCHAR2 (50) ,
    sex            CHAR ,
    salary         DECIMAL (10 , 2) ,
    superssn       CHAR (9) ,
    dno            INTEGER
) ;
```

Example: Company Database

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN,  
BDATE, ADDRESS, SEX, SALARY, DNO, SUPERSSN)  
VALUES ('Evan', 'E', 'Wallis', '222222200',  
TO_DATE('1958-01-16', 'yyyy-mm-dd'), '134 Pelham,  
Milwaukee, WI', 'M', 92000.0, 7, '');  
COMMIT;
```

```
---retrieve all rows  
select * from employee;
```

```
---remove the table from DB  
drop table employee;
```



SPECIFYING CONSTRAINTS IN SQL

Specifying Constraints in SQL

- Constraints:
 - Entity constraints – NOT NULL value
 - Domain constraints – data type on attributes
 - Key integrity constraints – No Duplicate keys
 - Referential integrity constraints – The “foreign key “ must have a value that is already present as a primary key, or may be null.
 - Constraints on each tuple within a relation



Entity & Domain constraints and defaults

- Entity Constraint: Specifying that NULL is not permitted for a particular attribute

```
ssn VARCHAR2(15) NOT NULL
```

- Specifying Default value

```
Alias VARCHAR2(11) DEFAULT ('none')
```

- Restricting values permitted for a particular attribute using **CHECK** clause

```
dno INTEGER CHECK (dno > 0 AND dno < 21);
```

```
id INTEGER CHECK (id BETWEEN 1 AND 9999);
```

```
name VARCHAR2(20) CHECK (name IN ('CS', 'INFO'));
```

Example: Entity & Domain constraints and defaults

```
CREATE TABLE employee
(
    fname      VARCHAR2(15) NOT NULL,
    minit      VARCHAR2(1),
    lname      VARCHAR2(15) NOT NULL,
    ssn        CHAR(9) NOT NULL,
    bdate      DATE,
    address    VARCHAR2(50),
    sex        CHAR,
    salary      DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
    superssn   CHAR(9),
    dno        INTEGER
);
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT,  
LNAME, SSN, BDATE, ADDRESS, SEX,  
SALARY, DNO, SUPERSSN) VALUES  
(  
'Evan', 'E', 'Wallis', '222222200',  
TO_DATE('1958-01-16', 'yyyy-mm-dd'),  
'134 Pelham, Milwaukee, WI',  
'M', 92000.0, 7, '');
```

Try now by deleting certain values to check the constraints specified in the previous slide



Key Constraints

- **PRIMARY KEY** clause

- attributes that make up the primary key of a relation

```
ssn CHAR(11) UNIQUE NOT NULL;  
student_id CHAR(11) PRIMARY KEY;
```

- **UNIQUE** clause: Specifies other candidate keys

```
dname VARCHAR2(15) UNIQUE NOT NULL;  
  
or  
  
UNIQUE (Dname);
```

Example: Key Constraints (simple)

```
CREATE TABLE employee
(
    fname      VARCHAR2(15) NOT NULL,
    minit      VARCHAR2(1),
    lname      VARCHAR2(15) NOT NULL,
    ssn        CHAR(9) PRIMARY KEY,
    bdate      DATE,
    address    VARCHAR2(50),
    sex        CHAR,
    salary     DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
    superssn   CHAR(9),
    dno        INTEGER
);
```

Example: Key Constraints (separate)

```
CREATE TABLE employee
(
    fname      VARCHAR2(15) NOT NULL,
    minit      VARCHAR2(1),
    lname      VARCHAR2(15) NOT NULL,
    ssn        CHAR(9),
    bdate      DATE,
    address    VARCHAR2(50),
    sex        CHAR,
    salary      DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
    superssn   CHAR(9),
    dno        INTEGER,
    PRIMARY KEY (ssn)
);
```

Example: Key Constraints (composite key)

```
CREATE TABLE employee
(
    fname      VARCHAR2(15) NOT NULL,
    minit      VARCHAR2(1),
    lname      VARCHAR2(15) NOT NULL,
    ssn        CHAR(9),
    bdate      DATE,
    address    VARCHAR2(50),
    sex        CHAR,
    salary      DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
    superssn   CHAR(9),
    dno        INTEGER,
    PRIMARY KEY (fname, lname, bdate) ---to demo composite key
);
```

Example: Entity + Key constraints (candidate keys)

```
CREATE TABLE department
(
  dname          VARCHAR2(25) NOT NULL,
  dnumber        INTEGER,
  mgrssn         CHAR(9) NOT NULL,
  mgrstartdate  DATE,
  PRIMARY KEY   (Dnumber),
  UNIQUE (dname)
);
DROP TABLE department;
CREATE TABLE department
(
  dname          VARCHAR2(25) UNIQUE NOT NULL,
  dnumber        INTEGER,
  mgrssn         CHAR(9) NOT NULL,
  mgrstartdate  DATE,
  PRIMARY KEY   (Dnumber)
);
```



Named Constraints

- Keyword **CONSTRAINT**
 - **Name** a constraint.
 - When a constraint is violated the name is displayed.
 - Useful for debugging.
 - Useful for altering the constraint.

Example: Company Database

```
CREATE TABLE employee
(
  fname      VARCHAR2(15) NOT NULL,
  minit      VARCHAR2(1),
  lname      VARCHAR2(15) NOT NULL,
  ssn        CHAR(9),
  bdate      DATE,
  address    VARCHAR2(50),
  sex        CHAR,
  salary     DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
  superssn   CHAR(9),
  dno        INTEGER,
  CONSTRAINT employee_pk PRIMARY KEY (ssn)
);
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT,  
LNAME, SSN, BDATE, ADDRESS, SEX,  
SALARY, DNO, SUPERSSN) VALUES  
( 'Evan', 'E', 'Wallis', '222222200',  
TO_DATE('1958-01-16', 'yyyy-mm-dd'),  
'134 Pelham, Milwaukee, WI',  
'M', 92000.0, 7, '');
```

Run the above twice – Primary Key Violation



Specifying Constraints on Tuples Using CHECK

- CHECK clauses at the end of a CREATE TABLE statement
 - Apply to **each** tuple individually

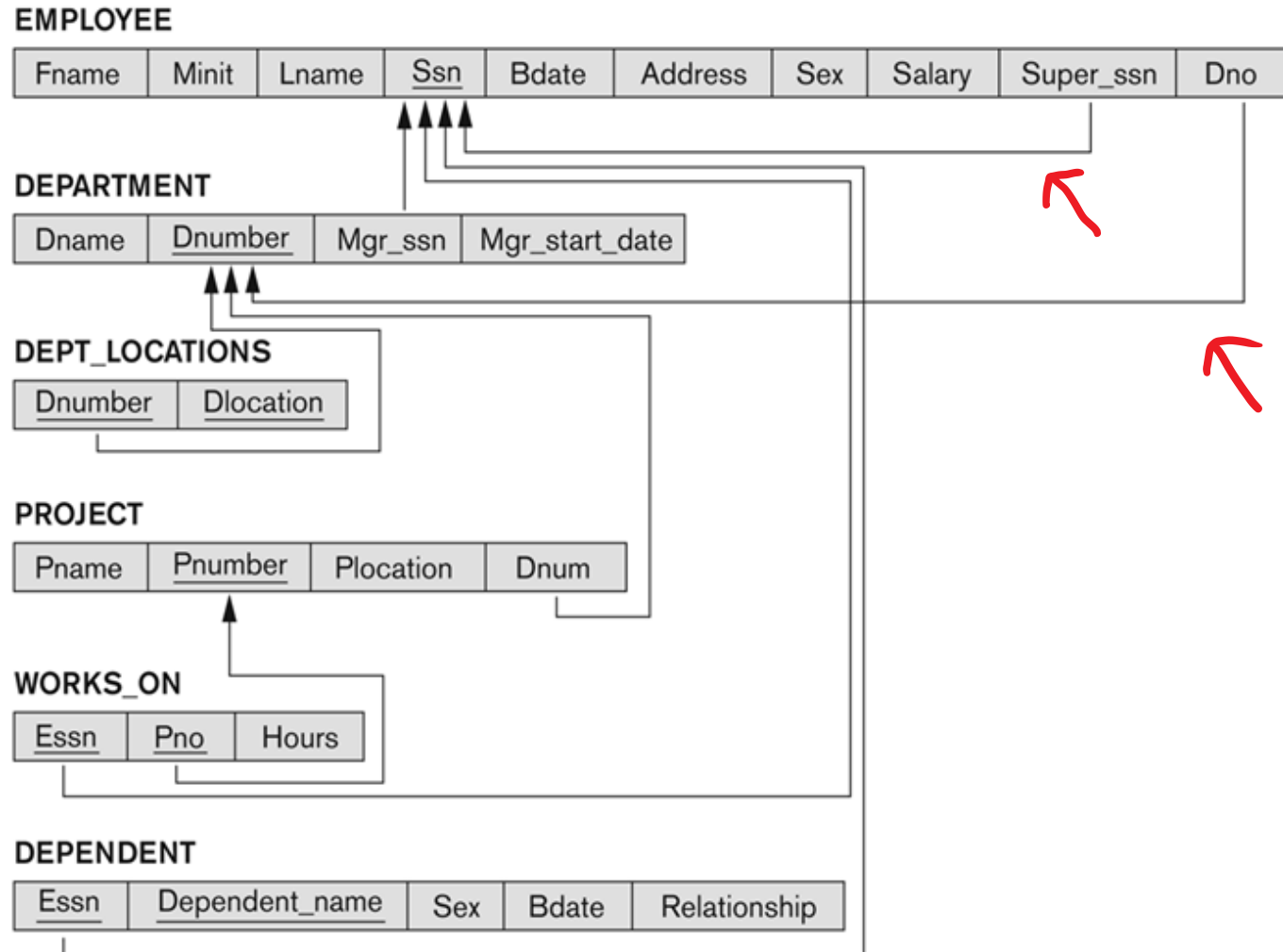
```
CHECK (Dept_create_date <= Mgr_start_date);
```

Example: Company Database

```
CREATE TABLE department
(
    dname          VARCHAR2(25) NOT NULL,
    dnumber        INTEGER,
    mgrssn         CHAR(9) NOT NULL,
    dcreatedate    DATE,
    mgrstartdate   DATE,
    PRIMARY KEY (Dnumber),
    UNIQUE (dname),
    CHECK (dcreatedate <= mgrstartdate)
);
```



Referential Integrity Constraints: Example: COMPANY database



Referential IC Example: Company Database

```
CREATE TABLE employee
(
  fname      VARCHAR2(15) NOT NULL,
  minit      VARCHAR2(1),
  lname      VARCHAR2(15) NOT NULL,
  ssn        CHAR(9),
  bdate      DATE,
  address    VARCHAR2(50),
  sex        CHAR,
  salary      DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
  superssn   CHAR(9),
  dno        INTEGER,
  PRIMARY KEY (ssn),
  FOREIGN KEY (superssn) REFERENCES employee(ssn),
  FOREIGN KEY (dno)      REFERENCES department(dnumber)
);
```

```
CREATE TABLE department
(
  dname      VARCHAR2(25) NOT NULL,
  dnumber    INTEGER,
  mgrssn     CHAR(9) NOT NULL,
  mgrstartdate DATE,
  PRIMARY KEY (dnumber),
  UNIQUE (dname)
FOREIGN KEY (mgrssn)
REFERENCES employee(ssn)
);
```

Referential Integrity Constraints

- What happens when a department is deleted? Should all employees also be deleted?
- **FOREIGN KEY** clause
 - *Default operation: reject the updates on constraint violation*
 - Attach **referential triggered action** clause
 - Options include SET NULL, CASCADE, and SET DEFAULT
 - Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE
 - CASCADE option suitable for “relationship” relations
 - **ORACLE does NOT support all the options!**
 - ON DELETE NO ACTION or ON UPDATE NO ACTION -> Prevent Delete or Update of Parent Key
 - ON DELETE CASCADE -> Delete Child Rows When Parent Key Deleted
 - ON DELETE SET NULL -> Set Foreign Keys to Null When Parent Key Deleted
 - Update Cascade (using initially deferred constraints)

Example: Company Database

```
DROP TABLE employee;
CREATE TABLE employee
(
  fname      VARCHAR2(15) NOT NULL,
  minit      VARCHAR2(1),
  lname      VARCHAR2(15) NOT NULL,
  ssn        CHAR(9),
  bdate      DATE,
  address    VARCHAR2(50),
  sex        CHAR,
  salary     DECIMAL(10,2) DEFAULT (0.0) CHECK (salary >= 0.0),
  superssn   CHAR(9),
  dno        INTEGER,
  PRIMARY KEY (ssn),
  FOREIGN KEY (superssn) REFERENCES employee(ssn) ON DELETE SET NULL,
  FOREIGN KEY (dno) REFERENCES department(dnumber) ON DELETE CASCADE
);
```

```
CREATE TABLE department
(
  dname      VARCHAR2(25) NOT NULL,
  dnumber    INTEGER,
  mgrssn     CHAR(9) NOT NULL,
  mgrstartdate DATE,
  PRIMARY KEY (dnumber),
  UNIQUE (dname),
FOREIGN KEY (mgrssn)
REFERENCES employee(ssn)
);
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX,  
SALARY, DNO, SUPERSSN) VALUES ('Evan', 'E', 'Wallis', '222222200',  
TO_DATE('1958-01-16', 'yyyy-mm-dd'), '134 Pelham, Milwaukee, WI',  
'M', 92000.0, 7, '122222200');
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX,  
SALARY, DNO, SUPERSSN) VALUES ('Bob', 'E', 'Wallis', '122222200',  
TO_DATE('1958-01-16', 'yyyy-mm-dd'), '134 Pelham, Milwaukee, WI',  
'M', 92000.0, 5, '');
```

```
INSERT INTO DEPARTMENT (DNAME, DNUMBER, MGRSSN ) VALUES  
('Research',5, '123');
```

```
INSERT INTO DEPARTMENT (DNAME, DNUMBER, MGRSSN ) VALUES  
('Purchase',7, '123');
```

```
DELETE FROM Department where dnumber = 5;
```



Create COMPANY database

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Emp → Dept (dno)
Dept → Emp (mgrssn)

CYCLE!

Need to handle in the create statement as you cannot create Emp without Dept or Dept without Emp!

∴ and Dr. Raman

Table Creation

- Tables can be created via interactive command prompt or GUI
 - E.g., Oracle LiveSQL, Oracle SQL*PLUS, Oracle SQL developer, MySQL Work Bench
- Separate CREATE statements
- .sql file with all statements
- Table creation order is important
- Sometimes you cannot create tables with constraints in the CREATE TABLE statement due to cycles!

Avoiding Cycles

- Create Department **without** FK relationship
- Create Table Employee **with** the FK referencing Department
 - *Still there are problems inserting data! We will fix that later*
- Alter the Department table and add constraints using
ALTER TABLE department ADD
(CONSTRAINT DEPARTMENT_EMPLOYEE_FK FOREIGN KEY (MGRSSN)
REFERENCES employee (ssn));

See **create_company.sql**

Create COMPANY database

create_company.sql

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

3

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

1

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

2

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

5

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

6

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

4

This is just one order for creating tables.



Schema description

■ desc_company.sql

```
--- In all platform except livesql
DESCRIBE Employee;
DESC Department;
DESC Dept_Locations;
DESC Project;
DESC WORKS_ON;
DESC DEPENDENT;
--- In all platforms
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'EMPLOYEE';
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'DEPARTMENT';
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'DEPT_LOCATIONS';
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'PROJECT';
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'WORKS_ON';
SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME = 'DEPENDENT';
```

■ This does not display the constraints



DROP Tables (remove **table structure** and **data** from the db)

- **drop_company.sql**
- Reverse the creation order
- `DROP TABLE DEPENDENT;`
- `DROP TABLE WORKS_ON;`
- `DROP TABLE PROJECT;`
- `DROP TABLE DEPT_LOCATIONS;`
- `DROP TABLE Employee cascade constraints; ---due to cycle`
- `DROP TABLE Department;`



Insert

Run create_company.sql

Try these

--- Cannot insert department without Employee

```
INSERT INTO DEPARTMENT (DNAME, DNUMBER, MGRSSN, MDGRSTARTDATE)
VALUES ('Research', 7.0, '222222200', to_date('1978-05-22', 'yyyy-
mm-dd'));
```

--- Cannot insert employee without department

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX,
SALARY, SUPERSSN, DNO)
VALUES ('Evan', 'E', 'Wallis', '222222200', TO_DATE('1958-01-16',
'yyyy-mm-dd'), '134 Pelham, Milwaukee, WI', 'M', 92000.0, '', 7);
```

Defer FK - PK constraint checking

- Instruct the database to check the **FK – PK values** only after inserting all the values to both the tables and committing the database

In the CREATE TABLE STATEMENT (use DEFERRABLE)

- CONSTRAINT employee_employee_fk FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ON DELETE SET NULL **DEFERRABLE INITIALLY DEFERRED**
 - **Check constraints only at commit**
- CONSTRAINT employee_department_fk FOREIGN KEY (Dno) REFERENCES Department(Dnumber) ON DELETE SET NULL **DEFERRABLE INITIALLY IMMEDIATE**
 - **Immediate constraint checking unless required to defer explicitly by 'set' statements**
- CONSTRAINT department_employee_fk FOREIGN KEY (mgrssn) REFERENCES employee (ssn) **DEFERRABLE**
 - **Similar to INITIALLY IMMEDIATE**

Insert (check constraints while inserting)

Run **drop_company.sql**

Run **create_company_deferred.sql**

```
SET CONSTRAINTS ALL IMMEDIATE;
```

```
INSERT INTO DEPARTMENT (DNAME, DNUMBER, MGRSSN, MDGRSTARTDATE)
VALUES ('Research', 7.0, '222222200', to_date('1978-05-22', 'yyyy-
mm-dd'));
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX,
SALARY, SUPERSSN, DNO)
VALUES ('Evan', 'E', 'Wallis', '222222200', TO_DATE('1958-01-16',
'yyyy-mm-dd'), '134 Pelham, Milwaukee, WI', 'M', 92000.0, '', 7);
COMMIT;
```

```
select * from employee;
```

```
select * from department;
```

Insert (check constraints while commit)

Run **drop_company.sql**

Run **create_company_deferred.sql**

```
SET CONSTRAINTS ALL DEFERRED;
```

```
INSERT INTO DEPARTMENT (DNAME, DNUMBER, MGRSSN, MDGRSTARTDATE)
VALUES ('Research', 7.0, '222222200', to_date('1978-05-22', 'yyyy-
mm-dd'));
```

```
INSERT INTO EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX,
SALARY, SUPERSSN, DNO)
VALUES ('Evan', 'E', 'Wallis', '222222200', TO_DATE('1958-01-16',
'yyyy-mm-dd'), '134 Pelham, Milwaukee, WI', 'M', 92000.0, '', 7);
---save and check is the db consistent and valid
COMMIT;
```

```
select * from employee;
select * from department;
```



Summary of steps

Run `drop_company.sql`

Run `create_company_deferred.sql`

Run `insert_department.sql`

Run `insert_employee.sql`

```
select * from department;
```

```
select * from employee;
```

Run `drop_company.sql`



Summary

- Data Definition Language
- Data Types
- Constraints
- CREATE, ALTER
- Avoid Cycles – Deferred
- Later
 - Basics of Index, Trigger, View, etc.

Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.