

Appendix 3.A: CREATE TABLE Statements for the University Database Tables

The following are the CREATE TABLE statements for the university database tables (Tables 3-1, 3-3, 3-4, 3-6, and 3-7) using the standard SQL data types and syntax. Data names vary by DBMS. For example, Microsoft Access SQL supports the TEXT data type instead of CHAR and VARCHAR. In Oracle, you should use VARCHAR2 instead of VARCHAR. In addition, certain syntax such as the rules for referenced rows follow the SQL standard, not Oracle syntax.

```
CREATE TABLE Student
(
    StdNo          CHAR(11),
    StdFirstName   VARCHAR(50) CONSTRAINT StdFirstNameRequired NOT NULL,
    StdLastName    VARCHAR(50) CONSTRAINT StdLastNameRequired NOT NULL,
    StdCity        VARCHAR(50) CONSTRAINT StdCityRequired NOT NULL,
    StdState       CHAR(2)      CONSTRAINT StdStateRequired NOT NULL,
    StdZip         CHAR(10)     CONSTRAINT StdZipRequired NOT NULL,
    StdMajor       CHAR(6),
    StdClass       CHAR(2),
    StdGPA         DECIMAL(3,2),
    CONSTRAINT PKStudent PRIMARY KEY (StdNo)
);
```

```
CREATE TABLE Course
(
    CourseNo       CHAR(6),
    CrsDesc        VARCHAR(250) CONSTRAINT CrsDescRequired NOT NULL,
    CrsUnits       INTEGER,
    CONSTRAINT PKCourse PRIMARY KEY (CourseNo),
    CONSTRAINT UniqueCrsDesc UNIQUE (CrsDesc)
);
```

```
CREATE TABLE Faculty
(
    FacNo          CHAR(11),
    FacFirstName   VARCHAR(50) CONSTRAINT FacFirstNameRequired NOT NULL,
    FacLastName    VARCHAR(50) CONSTRAINT FacLastNameRequired NOT NULL,
    FacCity        VARCHAR(50) CONSTRAINT FacCityRequired NOT NULL,
    FacState       CHAR(2)      CONSTRAINT FacStateRequired NOT NULL,
    FacZipCode     CHAR(10)     CONSTRAINT FacZipRequired NOT NULL,
    FacHireDate    DATE,
    FacDept        CHAR(6),
    FacRank        CHAR(4),
    FacSalary      DECIMAL(10,2),
    FacSupervisor  CHAR(11),
    CONSTRAINT PKFaculty PRIMARY KEY (FacNo),
    CONSTRAINT FKFacSupervisor FOREIGN KEY (FacSupervisor) REFERENCES Faculty
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

-- ON UPDATE clause is not Oracle syntax.

```
CREATE TABLE Offering
(
    OfferNo          INTEGER,
    CourseNo         CHAR(6)          CONSTRAINT OffCourseNoRequired NOT NULL,
    OffLocation      VARCHAR(50),
    OffDays          CHAR(6),
    OffTerm          CHAR(6)          CONSTRAINT OffTermRequired NOT NULL,
    OffYear          INTEGER          CONSTRAINT OffYearRequired NOT NULL,
    FacNo            CHAR(11),
    OffTime          DATE,
    CONSTRAINT PKOffering PRIMARY KEY (OfferNo),
    CONSTRAINT FKCourseNo FOREIGN KEY (CourseNo) REFERENCES Course
        ON DELETE RESTRICT
        ON UPDATE RESTRICT,
    CONSTRAINT FKFacNo FOREIGN KEY (FacNo) REFERENCES Faculty
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

-- ON UPDATE clause is not Oracle syntax.

CREATE TABLE Enrollment
(
    OfferNo          INTEGER,
    StdNo            CHAR(11),
    EnrGrade         DECIMAL(3,2),
    CONSTRAINT PKENrollment PRIMARY KEY (OfferNo, StdNo),
    CONSTRAINT FKOfferNo FOREIGN KEY (OfferNo) REFERENCES Offering
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT FKStdNo FOREIGN KEY (StdNo) REFERENCES Student
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- ON UPDATE clause is not Oracle syntax.
```

Appendix 3.B: SQL:2016 Syntax Summary

This appendix provides a convenient summary of the SQL:2016 syntax for the CREATE TABLE statement along with several related statements. For brevity, only the syntax of the most common parts of the statements is described. SQL:2016 is the current version of the SQL standard. The syntax in SQL:2016 for the statements described in this appendix is identical to the syntax in the previous SQL standards, SQL:2011, SQL:2008, SQL:2003, SQL:1999 and SQL-92. For the complete syntax, refer to a SQL:2016 or a SQL-92 reference book such as Groff and Weinberg (2002). The conventions used in the syntax notation are listed before the statement syntax:

- Uppercase words denote reserved words.
- Mixed-case words without hyphens denote names that the user substitutes.
- The asterisk * after a syntax element indicates that a comma-separated list can be used.
- The plus symbol + after a syntax element indicates that a list can be used. No commas appear in the list.
- Names enclosed in angle brackets <> denote definitions defined later in the syntax.
The definitions occur on a new line with the element and colon followed by the syntax.
- Square brackets [] enclose optional elements.
- Curly brackets { } enclose choice elements. One element must be chosen among the elements separated by the vertical bars |.
- The parentheses () denote themselves.
- Double hyphens — denote comments that are not part of the syntax.

CREATE TABLE¹ Syntax

```

CREATE TABLE  TableName
  ( <Column-Definition>*  [ ,  <Table-Constraint>*  ]  )

<Column-Definition>: ColumnName DataType
  [ DEFAULT { DefaultValue | USER | NULL } ]
  [ <Embedded-Column-Constraint>+ ]

<Embedded-Column-Constraint>:
  { [ CONSTRAINT ConstraintName ] NOT NULL      |
    [ CONSTRAINT ConstraintName ] UNIQUE        |
    [ CONSTRAINT ConstraintName ] PRIMARY KEY    |
    [ CONSTRAINT ConstraintName ] FOREIGN KEY
      REFERENCES TableName [ ( ColumnName ) ]
      [ ON DELETE  <Action-Specification> ]
      [ ON UPDATE  <Action-Specification> ] }

<Table-Constraint>: [ CONSTRAINT ConstraintName ]
  { <Primary-Key-Constraint>  |
    <Foreign-Key-Constraint>  |
    <Uniqueness-Constraint>  }

<Primary-Key-Constraint>: PRIMARY KEY ( ColumnName* )

<Foreign-Key-Constraint>: FOREIGN KEY ( ColumnName* )
  REFERENCES TableName [ ( ColumnName* ) ]
  [ ON DELETE  <Action-Specification> ]
  [ ON UPDATE  <Action-Specification> ]

<Action-Specification>: { CASCADE | SET NULL |
  SET DEFAULT | RESTRICT }

<Uniqueness-Constraint>: UNIQUE ( ColumnName* )

```

Other related statements

The ALTER TABLE and DROP TABLE statements support modification of a table definition and deleting a table definition. The ALTER TABLE statement is particularly useful because table definitions often change over time. In both statements, the keyword RESTRICT means that the statement cannot be performed if related tables exist. The keyword CASCADE means that the same action will be performed on related tables.

¹ The CHECK constraint, an important kind of table constraint, is described in Chapter 16.

```
ALTER TABLE  TableName
{ ADD { <Column-Definition> | <Table-Constraint> } |
  ALTER ColumnName { SET DEFAULT DefaultValue |
    DROP DEFAULT } |
  DROP ColumnName { CASCADE | RESTRICT } |
  DROP CONSTRAINT ConstraintName {CASCADE | RESTRICT }
}

DROP TABLE  TableName { CASCADE | RESTRICT }
```

Notes on Oracle Syntax

The CREATE TABLE statement in Oracle 12c SQL conforms closely to the SQL:2016 standard. Here is a list of the most significant syntax differences:

- Oracle SQL does not support the ON UPDATE clause for referential integrity constraints.
- Oracle SQL only supports CASCADE and SET NULL as the action specifications of the ON DELETE clause. If an ON DELETE clause is not specified, the deletion is not allowed (restricted) if related rows exist.
- Oracle does not support the BOOLEAN data type. A typical substitution is to use CHAR(1) with text values such as T/F or Y/N.
- Oracle SQL does not support dropping columns in the ALTER statement.
- Oracle SQL supports the MODIFY keyword in place of the ALTER keyword in the ALTER TABLE statement (use MODIFY ColumnName instead of ALTER ColumnName).
- Oracle SQL supports data type changes using the MODIFY keyword in the ALTER TABLE statement.

Appendix 3.C: Generation of Unique Values for Primary Keys

The SQL:2016 standard provides the GENERATED clause to support the generation of unique values for selected columns, typically primary keys. The GENERATED clause is used in place of a default value as shown in the following syntax specification. Typically a whole number data type such as INTEGER should be used for columns with a GENERATED clause. The START BY and INCREMENT BY keywords can be used to indicate the initial value and the increment value. The ALWAYS keyword indicates that the value is always automatically generated. The BY DEFAULT clause allows a user to specify a value, overriding the automatic value generation.

```
<Column-Definition>: ColumnName DataType
    [ <Default-Specification> ]
    [ <Embedded-Column-Constraint>+ ]

<Default-Specification>:
    { DEFAULT { DefaultValue | USER | NULL } |
      GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY
        START WITH NumericConstant
        [ INCREMENT BY NumericConstant ] }
```

Conformance to the SQL:2016 syntax for the GENERATED clause varies among DBMSs. IBM DB2 conforms closely to the syntax. Microsoft SQL Server uses slightly different syntax and only supports the ALWAYS option unless a SET IDENTITY statement is also used. Microsoft Access provides the AutoNumber data type to generate unique values. Oracle provides an additional default generation option (ON NULL) and other choices for sequence generation besides start and increment. In older Oracle versions (prior to Oracle 12c), sequence objects had to be used in place of the GENERATED clause. Oracle sequences have similar features except that users must maintain the association between a sequence and a column, a burden not necessary with the SQL:2016 standard.

The following examples contrast the SQL:2016 and Oracle approaches for automatic value generation. Note that the primary key constraint is not required for columns with generated values although generated values are mostly used for primary keys. The first Oracle example conforms closely to the SQL:2016 standard although it is supported only in Oracle 12c and beyond. The other example, for Oracle versions before 12c, contains two statements: one for the sequence creation and another for the table creation. Because sequences are not associated with columns, Oracle provides functions that should be used when inserting a row into a table. In contrast, the usage of extra functions is not necessary in SQL:2016.

SQL:2016 GENERATED Clause Example

```
CREATE TABLE Customer
( CustNo INTEGER GENERATED ALWAYS AS IDENTITY
  START WITH 1 INCREMENT BY 1,
.../
CONSTRAINT PKCustomer PRIMARY KEY (CustNo) )
```

Oracle GENERATED Clause Example

```
-- Oracle 12c only
-- User must have CREATE SEQUENCE privilege.
-- Oracle creates a sequence to support identity columns.
CREATE TABLE Customer
( CustNo INTEGER GENERATED ALWAYS AS IDENTITY
  (START WITH 1 INCREMENT BY 1),
.../
CONSTRAINT PKCustomer PRIMARY KEY (CustNo) )
```

Oracle Sequence Example

```
-- Oracle versions prior to Oracle 12c
CREATE SEQUENCE CustNoSeq START WITH 1 INCREMENT BY 1;

CREATE TABLE Customer
( CustNo INTEGER,
.../
CONSTRAINT PKCustomer PRIMARY KEY (CustNo) );
```