Database Foundations

6-3
Data Definition Language (DDL)
Roadmap

Introduction to Oracle Application Express

Structured Query Language (SQL)

Data Definition Language (DDL)

Data Manipulation Language (DML)

Transaction Control Language (TCL)

You are here

Retrieving Data Using SELECT

Restricting Data Using WHERE

Sorting Data Using ORDER BY

Joining Tables Using JOIN

Data Definition Language (DDL)
Objectives

This lesson covers the following objectives:

• Identify the steps needed to create database tables
• Describe the purpose of the data definition language (DDL)
• List the DDL operations needed to build and maintain a database's tables
## Database Objects

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Is the basic unit of storage; consists of rows</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates numeric values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives an alternative name to an object</td>
</tr>
</tbody>
</table>
Naming Rules for Tables and Columns

Table names and column names must:

• Begin with a letter
• Be 1–30 characters long
• Contain only A–Z, a–z, 0–9, _, $, and #
• Not duplicate the name of another object owned by the same user
• Not be an Oracle server–reserved word
CREATE TABLE Statement

• To issue a `CREATE TABLE` statement, you must have:
  – The `CREATE TABLE` privilege
  – A storage area

```
CREATE TABLE [schema.]table
  (column datatype [DEFAULT expr][, ...]);
```

• Specify in the statement:
  – Table name
  – Column name, column data type, column size
  – Integrity constraints (optional)
  – Default values (optional)
Creating Tables

• Create the table:

```sql
CREATE TABLE dept
(deptno NUMBER(2),
dname VARCHAR2(14),
loc VARCHAR2(13),
create_date DATE DEFAULT SYSDATE);
```

• Confirm table creation:

```sql
DESCRIBE dept
```
## Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2(size)</td>
<td>Variable-length character data (A maximum size must be specified; minimum size is 1.) Maximum size: 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY</td>
</tr>
<tr>
<td>CHAR(size)</td>
<td>Fixed-length character data of length (size) bytes. (Default and minimum size is 1; maximum size is 2,000)</td>
</tr>
<tr>
<td>NUMBER(p, s)</td>
<td>Variable-length numeric data. Precision is p, and scale is s. (Precision is the total number of decimal digits, and scale is the number of digits to the right of the decimal point; precision can range from 1 to 38, and scale can range from -84 to 127.)</td>
</tr>
<tr>
<td>DATE</td>
<td>Date and time values to the nearest second between January 1, 4712 B.C, and December 31, 9999 A.D.</td>
</tr>
<tr>
<td>LONG</td>
<td>Variable-length character data (up to 2 GB)</td>
</tr>
</tbody>
</table>
## Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOB</td>
<td>A character large object (CLOB) containing single-byte or multibyte characters. Maximum size is ((4 \text{ GB} - 1) \times \text{DB BLOCK SIZE}); stores national character set data.</td>
</tr>
<tr>
<td>NCLOB</td>
<td>A CLOB containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is ((4 \text{ GB} - 1) \times \text{database block size}); stores national character set data.</td>
</tr>
<tr>
<td>RAW (Size)</td>
<td>Raw binary data of length (\text{size}) bytes. You must specify (\text{size}) for a RAW value. Maximum size: 32767 bytes if (\text{MAX_SQL_STRING_SIZE = EXTENDED}) 4000 bytes if (\text{MAX_SQL_STRING_SIZE = LEGACY})</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>Raw binary data of variable length up to 2 GB.</td>
</tr>
<tr>
<td>BLOB</td>
<td>A binary large object. Maximum size is ((4 \text{ GB} - 1) \times \text{DB BLOCK_SIZE}) initialization parameter (8 (\text{TB to 128 TB})).</td>
</tr>
<tr>
<td>BFILE</td>
<td>Binary data stored in an external file (up to 4 GB).</td>
</tr>
<tr>
<td>ROWID</td>
<td>Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn</td>
</tr>
</tbody>
</table>
Example: Creating a Table with Different Data Types

```
CREATE TABLE print_media
(product_id NUMBER(6),
 media_id NUMBER(6),
 media_desc VARCHAR2(100),
 media_composite BLOB,
 media_sourcetext CLOB,
 media_finaltext CLOB,
 media_photo BLOB,
 media_graphic BFILE);
```
## Date Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>Enables storage of time as a date with fractional seconds. It stores the year, month, day, hour, minute, the second value of the DATE data types, and the fractional seconds value. There are several variations of this data type, such as WITH TIMEZONE and WITH LOCALTIMEZONE.</td>
</tr>
<tr>
<td>INTERVAL YEAR TO MONTH</td>
<td>Enables storage of time as an interval of years and months. Used to represent the difference between two datetime values in which the only significant portions are the year and month.</td>
</tr>
<tr>
<td>INTERVAL DAY TO SECOND</td>
<td>Enables storage of time as an interval of days, hours, minutes, and seconds; used to represent the precise difference between two datetime values.</td>
</tr>
<tr>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>Variant of TIMESTAMP that includes a time zone region name or time zone offset in its value.</td>
</tr>
</tbody>
</table>
Examples: Date Data Types

• Example of TIMESTAMP data type:

  CREATE TABLE table_ts(c_id NUMBER, c_ts TIMESTAMP);
  INSERT INTO table_ts VALUES(1, '01-JAN-2003 2:00:00');

• Example of a table with TIMESTAMP, INTERVAL YEAR TO MONTH and INTERVAL DAY TO SECOND columns:

  CREATE TABLE time_table
  (start_time TIMESTAMP,
   duration_1 INTERVAL DAY (6) TO SECOND (5),
   duration_2 INTERVAL YEAR TO MONTH);
DEFAULT Option

- Specify a default value for a column during CREATE TABLE.

... hire_date DATE DEFAULT SYSDATE, ...

- Literal values, expressions, or SQL functions are legal values.

- Another column's name or a pseudocolumn are illegal values.

- The default data type must match the column data type.

CREATE TABLE hire_dates
(id          NUMBER(8),
 hire_date DATE DEFAULT SYSDATE);
table HIRE_DATES created.
Case Scenario: Creating Tables

How about creating the tables for the simplified library database?
Case Scenario: Creating Tables

CREATE TABLE BOOK_TRANSACTION
(
TRANSACTION_ID VARCHAR2(6),
TRANSACTION_DATE DATE DEFAULT SYSDATE ,
TRANSACTION_TYPE VARCHAR2 VARCHAR2(10),
BOOK_ID VARCHAR2(6),
MEMBER_ID VARCHAR2(6)
)
;

CREATE TABLE AUTHOR
(
AUTHOR_ID VARCHAR2(6),
AUTHOR_NAME VARCHAR2(60)
)
;

CREATE TABLE MEMBERS
(
MEMBER_ID VARCHAR2(6),
FIRST_NAME VARCHAR2(50),
LAST_NAME VARCHAR2(50),
STREET_ADDRESS VARCHAR2(50),
CITY VARCHAR2(20),
STATE VARCHAR2(2),
ZIP VARCHAR2(10)
)
;

CREATE TABLE BOOKS
(
BOOK_ID VARCHAR2(6),
TITLE VARCHAR2(255),
PUBLISHER_ID VARCHAR2(6),
AUTHOR_ID VARCHAR2(6)
)
;

CREATE TABLE PUBLISHER
(
PUBLISHER_ID VARCHAR2(6),
PUBLISHER_NAME VARCHAR2(100)
) ;
Case Scenario: Creating Tables

CREATE TABLE AUTHOR
(
    AUTHOR_ID VARCHAR2(6),
    AUTHOR_NAME VARCHAR2(50)
);

CREATE TABLE MEMBERS
(
    MEMBER_ID VARCHAR2(6),
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    STREET_ADDRESS VARCHAR2(50),
    CITY VARCHAR2(20),
    STATE VARCHAR2(2),
    ZIP VARCHAR2(10)
);

CREATE TABLE PUBLISHER
(
    PUBLISHER_ID VARCHAR2(6),
    PUBLISHER_NAME VARCHAR2(100) NOT NULL
);

CREATE TABLE BOOKS
(
    BOOK_ID VARCHAR2(6),
    TITLE VARCHAR2(255) NOT NULL,
    PUBLISHER_ID VARCHAR2(6),
    AUTHOR_ID VARCHAR2(6)
);

CREATE TABLE BOOK TRANSACTION

Successful creation of tables
Including Constraints

• Constraints enforce rules at the table level.
• Constraints ensure the consistency and integrity of the database.
• The following constraint types are valid:
  – NOT NULL
  – UNIQUE
  – PRIMARY KEY
  – FOREIGN KEY
  – CHECK
## Data Integrity Constraints

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT NULL</td>
<td>The column cannot contain a null value.</td>
</tr>
<tr>
<td>UNIQUE</td>
<td>The values for a column or a combination of columns must be unique for all rows in the table.</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>The column (or a combination of columns) must contain the unique AND IS NOT NULL value for all rows.</td>
</tr>
<tr>
<td>FOREIGN KEY</td>
<td>The column (or a combination of columns) must establish and enforce a reference to a column or a combination of columns in another (or the same) table.</td>
</tr>
<tr>
<td>CHECK</td>
<td>A condition must be true.</td>
</tr>
</tbody>
</table>
Constraint Guidelines

• Name a constraint (otherwise, the Oracle server generates a name in the SYS_Cn format).

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_C0014370</td>
<td>Primary Key</td>
</tr>
</tbody>
</table>

• Create a constraint at either of the following times:
  – At the same time as the creation of the table
  – After the creation of the table

• Define a constraint at the column or table level.

• View a constraint in the data dictionary.
Defining Constraints

• CREATE TABLE with CONSTRAINTS syntax:

```
CREATE TABLE schema.table
(column datatype [DEFAULT expr]
[column_constraint],
...
[table_constraint][,...]);
```

• Column-level constraint syntax:

```
column [CONSTRAINT constraint_name] constraint_type,
```

• Table-level constraint syntax:

```
column,...
[CONSTRAINT constraint_name] constraint_type
(column, ...),
```
Examples: Defining Constraints

• Column-level constraint:

```sql
CREATE TABLE employees(
    employee_id  NUMBER(6)
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,
    first_name   VARCHAR2(20),
    ...);
```

• Table-level constraint:

```sql
CREATE TABLE employees(
    employee_id  NUMBER(6),
    first_name   VARCHAR2(20),
    ...
    job_id       VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
    PRIMARY KEY (EMPLOYEE_ID));
```
NOT NULL Constraint

Ensures that null values are not permitted for the column:

Note: NOT NULL constraints can be created ONLY at the column level.
UNIQUE KEY Constraint

EMPLOYEES

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>SKING</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>NKOCHMAR</td>
</tr>
<tr>
<td>102</td>
<td>De Haan</td>
<td>LDEHAAN</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>AHUNOLD</td>
</tr>
<tr>
<td>104</td>
<td>Ernst</td>
<td>BERNST</td>
</tr>
</tbody>
</table>

UNIQUE KEY constraint

INSERT INTO

| 208 SMITH | JSMITH |
| 209 SMITH | JSMITH |

Allowed

Not allowed: already exists
UNIQUE Constraint

Defined at either the table level or the column level:

```sql
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
CONSTRAINT emp_email_uk UNIQUE(email));
```
PRIMARY KEY Constraint

DEPARTMENTS

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Administration</td>
<td>200</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Marketing</td>
<td>201</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>Purchasing</td>
<td>114</td>
<td>1700</td>
</tr>
<tr>
<td>40</td>
<td>Human Resources</td>
<td>203</td>
<td>2400</td>
</tr>
<tr>
<td>50</td>
<td>Shipping</td>
<td>121</td>
<td>1500</td>
</tr>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>70</td>
<td>Public Relations</td>
<td>204</td>
<td>2700</td>
</tr>
<tr>
<td>80</td>
<td>Sales</td>
<td>145</td>
<td>2500</td>
</tr>
</tbody>
</table>

Not allowed (null value)

INSERT INTO

<table>
<thead>
<tr>
<th>(null)</th>
<th>Public Accounting</th>
<th>124</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 Finance</td>
<td>124</td>
<td>1500</td>
<td></td>
</tr>
</tbody>
</table>

Not allowed (50 already exists)
### FOREIGN KEY Constraint

#### DEPARTMENTS

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Administration</td>
<td>200</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Marketing</td>
<td>201</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>Purchasing</td>
<td>114</td>
<td>1700</td>
</tr>
<tr>
<td>40</td>
<td>Human Resources</td>
<td>203</td>
<td>2400</td>
</tr>
<tr>
<td>50</td>
<td>Shipping</td>
<td>121</td>
<td>1500</td>
</tr>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
</tbody>
</table>

#### EMPLOYEES

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>90</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>90</td>
</tr>
<tr>
<td>102</td>
<td>De Haan</td>
<td>90</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>60</td>
</tr>
<tr>
<td>104</td>
<td>Ernst</td>
<td>60</td>
</tr>
</tbody>
</table>

#### INSERT INTO

- **Not allowed** (9 does not exist)
- **Allowed**
FOREIGN KEY Constraint

Defined at either the table level or the column level:

```sql
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
    REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```
FOREIGN KEY Constraint: Keywords

• FOREIGN KEY: Defines the column in the child table at the table-constraint level

• REFERENCES: Identifies the table and column in the parent table

• ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted

• ON DELETE SET NULL: Converts dependent foreign key values to null
CHECK Constraint

- It defines a condition that each row must satisfy.
- It cannot reference columns from other tables.

```sql
..., salary NUMBER(8,2)
CONSTRAINT emp_salary_min
  CHECK (salary > 0),...
```
CREATE TABLE teach_emp (  
empno   NUMBER(5) PRIMARY KEY,  
ename   VARCHAR2(15) NOT NULL,  
job     VARCHAR2(10),  
mgr     NUMBER(5),  
hiredate DATE DEFAULT (sysdate),  
photo   BLOB,  
sal     NUMBER(7,2),  
deptno  NUMBER(3) NOT NULL  
CONSTRAINT admin_dept_fkey REFERENCES departments(department_id));
Case Scenario: Creating Tables

How about adding the constraints to the simplified library database tables?
Case Scenario: Adding Constraints

```
CREATE TABLE AUTHOR
(
    AUTHOR_ID VARCHAR2(6),
    AUTHOR_NAME VARCHAR2(60),
    CONSTRAINT author_author_id_pk PRIMARY KEY (AUTHOR_ID)
);

CREATE TABLE MEMBERS
(
    MEMBER_ID VARCHAR2(6),
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    STREET_ADDRESS VARCHAR2(50),
    CITY VARCHAR2(20),
    STATE VARCHAR2(2),
    ZIP VARCHAR2(10),
    CONSTRAINT members_member_id_pk PRIMARY KEY (MEMBER_ID)
);
```
Case Scenario: Adding Constraints

```
CREATE TABLE PUBLISHER
(
  PUBLISHER_ID   VARCHAR2(6),
  PUBLISHER_NAME VARCHAR2(100) NOT NULL,
  CONSTRAINT publisher_id_pk PRIMARY KEY (PUBLISHER_ID)
);

CREATE TABLE BOOKS
(
  BOOK_ID       VARCHAR2(6),
  TITLE         VARCHAR2(255) NOT NULL,
  PUBLISHER_ID  VARCHAR2(6),
  AUTHOR_ID     VARCHAR2(6),
  CONSTRAINT book_book_id_pk PRIMARY KEY (BOOK_ID),
  CONSTRAINT bk_auth_fk FOREIGN KEY (author_id) REFERENCES author(author_id),
  CONSTRAINT bk_publ_fk FOREIGN KEY (publisher_id) REFERENCES publisher(publisher_id)
);

CREATE TABLE BOOK_TRANSACTION
(
  TRANSACTION_ID VARCHAR2(6),
  TRANSACTION_DATE DATE DEFAULT SYSDATE NOT NULL,
  TRANSACTION_TYPE VARCHAR2(10),
  BOOK_ID        VARCHAR2(6),
  MEMBER_ID      VARCHAR2(6),
  CONSTRAINT booktrans_id_pk PRIMARY KEY (TRANSACTION_ID),
  CONSTRAINT bk_trns_fk FOREIGN KEY (book_id) REFERENCES books(book_id),
  CONSTRAINT bk_mem_fk FOREIGN KEY (member_id) REFERENCES members(member_id)
);
```
ALTER TABLE Statement

Use the `ALTER TABLE` statement to:

- Add a column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change a table to read-only status
ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns:

```
ALTER TABLE table
ADD     (column data type [DEFAULT expr]
[, column data type]...);
```

```
ALTER TABLE table
MODIFY    (column data type [DEFAULT expr]
[, column data type]...);
```

```
ALTER TABLE table
DROP (column [, column] ...);
```
Adding a Column

• You use the ADD clause to add columns:

```sql
ALTER TABLE dept80
ADD     (job_id VARCHAR2(9));
```

• The new column becomes the last column:

```
<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>17-JUN-03</td>
<td>-</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>21-SEP-05</td>
<td>-</td>
</tr>
<tr>
<td>102</td>
<td>De Haan</td>
<td>13-JAN-01</td>
<td>-</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>03-JAN-06</td>
<td>-</td>
</tr>
</tbody>
</table>
```
Modifying a Column

• You can change a column's data type, size, and default value:

```
ALTER TABLE dept80
MODIFY (last_name VARCHAR2(30));
```

• A changed default value affects only subsequent insertions in the table.
Dropping a Column

Use the `DROP COLUMN` clause to drop columns that you no longer need:

```sql
ALTER TABLE dept80
DROP (job_id);
```

Table altered.
SET UNUSED Option

- You use the `SET UNUSED` option to mark one or more columns as unused.
- You use the `DROP UNUSED COLUMNS` option to remove the columns that are marked as unused.
SET UNUSED Option

You can specify the **ONLINE** keyword to indicate that DML operations on the table will be allowed while marking the column or columns **UNUSED**.

```sql
ALTER TABLE <table_name>
SET UNUSED(<column_name> [, <column_name>]);

OR

ALTER TABLE <table_name>
SET UNUSED COLUMN <column_name> [, <column_name>];

ALTER TABLE <table_name>
DROP UNUSED COLUMNS;
```
Case Scenario: Altering Tables

Faculty

Sean, I was reviewing the AUTHORS table and realized that
- The author's email address field is missing.
- The author’s name column length needs to be increased.
Can you make these changes?

Student

Sure, I can do it. Because the modification is adding a new column and is increasing the column length, this should not be an issue.
Case Scenario: Altering Tables

```
ALTER TABLE AUTHOR
ADD EMAIL VARCHAR2(255);
ALTER TABLE AUTHOR
MODIFY AUTHOR_NAME VARCHAR2(100);
```

Looks like the table is modified:

```
table altered
table altered
```
Read-Only Tables

You can use the ALTER TABLE syntax to:

• Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
• Put the table back into read/write mode

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```
Dropping a Table

• Moves a table to the recycle bin
• Removes the table and its data if the PURGE clause is specified
• Invalidates dependent objects and removes object privileges on the table

DROP TABLE dept80;

Table dropped.
Using Oracle SQL Data Modeler to Generate DDL
Summary

In this lesson, you should have learned how to:

• Identify the steps needed to create database tables
• Describe the purpose of the DDL
• List the DDL operations needed to build and maintain a database's tables