The ABAP Dictionary (SE11) allows

- Centralized type definition and management
- Creation of user-defined data elements, structures, and table types
- Definition of tables, indexes, and views
- Definition of services that support program development
  - Table locking and lock releasing
  - Defining input help (F4)
  - Defining field help (F1)

Changes within the ABAP Dictionary are immediately in effect for those programs which use that dictionary elements.
Data Definition

Data Types  Three types can be defined:
- Data Element
- Structure
- Table Types (internal tables)

The Type Group option provided by the Dictionary that is primarily there to support older development. (Type Groups were available before Data Types were supported.)

Type Groups now are primarily used to define non-object-oriented constants.

Domain data is available to support the definition of data types, but domain data is not accessible in an ABAP Program.

There are 24 data specifications in the domain that are used in Data Dictionary Data Types.

Data Objects and the Data Dictionary

- Internal Table
  - Line structure
- Database Table
  - Fields
- Structure
  - Components
- Data Element
- Domain

ABAP Program
Tables in the ABAP Dictionary

Tables consist of columns (fields) and rows (entries).
In the ABAP Dictionary tables have names, and general attributes such as a delivery class and maintenance authorization.
A table has one or more key fields, designated as the primary key.
Table fields are commonly based on data elements which are based on domain data.
Table fields may be typed without reference to a data element. In this case the field is said to be direct typed.
Tables have technical settings which govern how the system should optimize the data storage and access the table.
What is the expected table size?
Should the table be buffered? Should changes to the table be logged?

Transparent tables

When a table is defined in the ABAP Dictionary it is a transparent table. When the transparent table is activated, it is created in the underlying DBMS.
The DBMS-specific table is the same name as the ABAP Dictionary transparent table name.
The field types in the DBMS-specific table are converted to whatever the appropriate data type definition is in the DBMS.
Underlying elements of the table (such as field order, etc.) may be changed as necessary.
The transparent table in the ABAP Dictionary is the database-independent representation of the underlying table, and is used in ABAP programming.
**Open SQL**

Open SQL statements in ABAP program converted to database-specific SQL by the database interface.

It is possible to access database with Native SQL, but the code is then database specific and less transportable.

Many Open SQL statements mirror those of "traditional" SQL, but with subtle differences and refinements.
SAP LUW Concept

SAP LUW (Logical Unit of Work)—set of logical changes to a database that all belong together.

A LUW is done in accordance with the "all or nothing" principle. If entire sequence cannot be done, the operation is rolled back.

Commit operation committed implicitly or via ABAP statement COMMIT WORK. Rollback done via statement ROLLBACK WORK.

Necessity of LUW

SAP ERP is three tiered client-server architecture. On the application server, end user applications are executed via work processes.

Multiple work processes in effect simultaneously. By partitioning database operations into LUWs, database integrity can be maintained and a work process can access and release database resources rapidly.
Database Locking

To avoid simultaneous/concurrent changes to database tables from multiple programs, locks are used.

Example: The command `SELECT SINGLE field FROM dbtable FOR UPDATE` will lock the record selected in anticipation of a pending change.

At the end of the LUW, the lock is released.

To lock a resource for a more extended sequence, the system maintains a global lock table which manages locks set programmatically for various resources.

To set the lock: `ENQUEUE lockobject`.

To release the lock: `DEQUEUE lockobject`.

Open SQL

Open SQL contains only Data Manipulation Language (DML) commands. Data Definition Language (DDL) operations are done using the ABAP Dictionary.

Four basic Open SQL commands:

- `SELECT`
- `UPDATE`
- `INSERT`
- `DELETE`
Database related sy fields

sy-subrc is set after all database operations.

  0 = successful completion.

  Other values indicate different error types.

sy-dbcnt is set to the number of records affected by the last operation.

---

Select Statement

5 clauses:

**SELECT <result>**

  Determines fields to be chosen and whether 1 record or multiple will be retrieved.

  **SELECT SINGLE**—retrieves 1 row: the first match.

**FROM <table>**

**INTO <target>**

  Specifies data object where result will be stored.

  Results loaded based on order specified, unless CORRESPONDING FIELDS OF added to statement.

**WHERE <condition>**

  Specifies which rows are to be chosen.

sy-subrc

  0 = success

  4 = empty resultset (no data found)
MANDT field

Most system tables contain an MANDT field designating a client. Open SQL commands by default are done for the current client only. To override this behavior add the CLIENT SPECIFIED clause to the command.

Allows a client number to be specified, for example in a WHERE clause in a query.

If CLIENT SPECIFIED is used but no client-limiting code is included, data for all clients is returned.

```
DATA iTab TYPE TABLE OF spfli.
DATA str LIKE LINE OF iTab.

SELECT mandt carrid connid cityfrom cityto FROM spfli INTO CORRESPONDING FIELDS OF TABLE iTAB.
LOOP AT iTab INTO str.
WRITE: / str-mandt, str-carrid, str-connid,
    str-cityfrom, str-cityto.
ENDLOOP.
```

```
DATA iTab TYPE TABLE OF spfli.
DATA str LIKE LINE OF iTab.

SELECT mandt carrid connid cityfrom cityto FROM spfli
    CLIENT SPECIFIED INTO CORRESPONDING FIELDS OF TABLE iTAB.
LOOP AT iTab INTO str.
WRITE: / str-mandt, str-carrid, str-connid,
    str-cityfrom, str-cityto.
ENDLOOP.
```
SELECT loop

If INTO clause indicates INTO TABLE, result put into internal table in a single step operation called an array fetch.

If INTO clause does not specify INTO TABLE, data copied iteratively to a data object (or objects).

Creates a select loop where body of the loop can specify actions to be taken on the data object during each iteration.

Loop ends with the ENDSELECT statement.

```
DATA str TYPE SPFLI. "structure matching table row
SELECT CARRID CONNID CITYFROM CITYTO FROM SPFLI
  INTO CORRESPONDING FIELDS OF str.
  WRITE: / str-CARRID, 'flight', str-CONNID,
    str-CITYFROM, 'to', str-CITYTO.
ENDSELECT.
WRITE: / sy-dbcnt, 'rows output'.
```

Select loop into individual data objects

In lieu of a structure as the copy destination, a set of individual data objects may be specified:

```
SELECT fields FROM table INTO (var1, var2) WHERE ...
```

```
DATA: my-client TYPE c LENGTH 3,
    my-carrid TYPE c LENGTH 3,
    my-connid TYPE c LENGTH 4.

SELECT mandt carrid connid FROM spfli CLIENT
  SPECIFIED INTO (my-client, my-carrid, my-connid).
  WRITE: / my-client, my-carrid, my-connid.
ENDSELECT.

WRITE: / sy-dbcnt, 'rows output'.
```
**INSERT**

INSERT INTO dbtable VALUES structure.

INSERT INTO dbtable FROM structure.

INSERT dbtable FROM TABLE internaltable.

sy-subrc

0 = success

4 = insertion not possible as line with same key already present

For FROM TABLE option if even one line of internaltable cannot be inserted due to duplicate key in dbtable, the entire operation is rolled back.

If desire all valid rows to be inserted, add ACCEPTING DUPLICATE KEYS to the statement.

Duplicate rows still skipped, but other rows inserted.

sy-subrc set to 4 to indicate this condition, sy-dbcnt set to number of rows successfully inserted.

---

**UPDATE a single record or multiple records**

UPDATE dbtable FROM str.

Key field values in str are used for row selection.

UPDATE dbtable SET field = value field = value ... WHERE keyfield = value AND keyfield = value.

Only fields specified in SET clause are changed.

If conditions specified in WHERE are not unique keys, multiple records may be changed.

UPDATE dbtable FROM TABLE internaltable.

sy-subrc

0 = success

4 = update not possible (key not found or other problem)

sy-dbcnt set to number of updated rows
MODIFY

If record does not exist, INSERT operation done. If record exists, UPDATE operation done.

MODIFY dbtable FROM structure.

MODIFY dbtable FROM TABLE internaltable.

sy-subrc

0 = success
4 = modify not possible (key not found or other problem)

sy-dbcnt set to number of updated rows

DELETE

DELETE FROM dbtable WHERE key = value AND key = value
DELETE FROM dbtable WHERE condition.
DELETE FROM dbtable FROM TABLE internaltable.

Key fields as found in internaltable deleted from dbtable.

sy-subrc

0 = success
4 = delete not possible (key not found or other problem)

sy-dbcnt set to number of updated rows
Joins

It is possible to join tables in a query.

```
SELECT * FROM table1 INNER JOIN table2 ON table1~field = table2~field WHERE <condition>.
```

Called a link in ABAP. If the join done in a program it is a dynamic link.

Pre-existing links, static links, are defined in the ABAP Dictionary. These are called views.

Additional details on joins are beyond the scope of this class, but are well documented in the ABAP documentation.