Authorization and Ownership

•Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.
•Used to determine which objects user may reference and what operations may be performed on those objects.
•Each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
•Owner is only person who may know about it.

Privileges
•Actions user permitted to carry out on given base table or view:

SELECT	Retrieve data from a table.
INSERT	Insert new rows into a table.
UPDATE	Modify rows of data in a table.
DELETE	Delete rows of data from a table.
REFERENCES	Reference columns of named table in integrity constraints.
USAGE	Use domains, collations, character sets, and translations.

•Can restrict INSERT/UPDATE/REFERENCES to named columns.
•Owner of table must grant other users the necessary privileges using GRANT statement.
•To create view, user must have SELECT privilege on all tables that make up view and REFERENCES privilege on the named columns.

GRANT
```
GRANT  {PrivilegeList | ALL PRIVILEGES}
ON  ObjectName
TO  {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```
•*PrivilegeList* consists of one or more of above privileges separated by commas.
•ALL PRIVILEGES grants all privileges to a user.

•PUBLIC allows access to be granted to all present and future authorized users.
•*ObjectName* can be a base table, view, domain, character set, collation or translation.
•WITH GRANT OPTION allows privileges to be passed on.

Give Manager full privileges to Staff table.
```
GRANT ALL PRIVILEGES
ON Staff
TO Manager WITH GRANT OPTION;
```

Give users Personnel and Director SELECT and UPDATE on column salary of Staff.
```
GRANT SELECT, UPDATE (salary)
ON Staff
TO Personnel, Director;
```

Give all users SELECT on Branch table.
```
GRANT SELECT
ON Branch
TO PUBLIC;
```

REVOKE

•REVOKE takes away privileges granted with GRANT.

```
  REVOKE [GRANT OPTION FOR]
  {PrivilegeList | ALL PRIVILEGES}
  ON ObjectName
  FROM {AuthorizationIdList | PUBLIC}
    [RESTRICT | CASCADE]
```

•ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges.
•GRANT OPTION FOR allows privileges passed on via WITH GRANT OPTION of GRANT to be revoked separately from the privileges themselves.
•REVOKE fails if it results in an abandoned object, such as a view, unless the CASCADE keyword has been specified.
•Privileges granted to this user by other users are not affected.

Revoke privilege SELECT on Branch table from all users.

```
  REVOKE SELECT
  ON Branch
  FROM PUBLIC;
```

Revoke all privileges given to Director on Staff table.

```
  REVOKE ALL PRIVILEGES
  ON Staff
  FROM Director;
```

Creating a Trigger

•Triggers are programs that are triggered by an event, typically INSERT, UPDATE, or DELETE.
•They can be used to enforce business rules that referential integrity and constraints alone cannot enforce.
•The basic syntax for creating a trigger is:

```
CREATE TRIGGER <trigger_name> ON <table_name>
[FOR, AFTER, INSTEAD OF] [INSERT, UPDATE, DELETE]
AS
{SQL Code}
```

Advanced SQL

•SQL is a powerful language and there is much more that can be done with it.
•Subqueries allow a user to embed whole independent SELECT statements in the SELECT clause or as a criterion in the WHERE clause.
•Unions allow a user to blend the results of a two-result set into a single tabular output.
•You can use SQL to find and remove duplicates.
•Indexes help a database administrator speed up query results and optimize the database.

Integrity Enhancement Feature

•Consider five types of integrity constraints:

•required data

```
  position  VARCHAR(10)  NOT NULL
```

•domain constraints

(a) CHECK

```
  sex  CHAR  NOT NULL
```

CHECK (sex IN (‘M’, ‘F’))
(b) CREATE DOMAIN
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
For example:
CREATE DOMAIN SexType AS CHAR
CHECK (VALUE IN (‘M’, ‘F’));
sex SexType NOT NULL
•*searchCondition* can involve a table lookup:
CREATE DOMAIN BranchNo AS CHAR(4)
CHECK (VALUE IN (SELECT branchNo
FROM Branch));
•Domains can be removed using DROP DOMAIN:
DROP DOMAIN DomainName
[RESTRICT | CASCADE]

•entity integrity
•Primary key of a table must contain a unique, non-null value for each row.
•ISO standard supports FOREIGN KEY clause in CREATE and ALTER TABLE statements:
•
PRIMARY KEY(staffNo)
PRIMARY KEY(clientNo, propertyNo)
•Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:
UNIQUE(telNo)

•referential integrity
•FK is column or set of columns that links each row in child table containing foreign FK to row of parent table containing matching PK.
•Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.
•ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE:
FOREIGN KEY(branchNo) REFERENCES Branch
•Any INSERT/UPDATE attempting to create FK value in child table without matching CK value in parent is rejected.
•Action taken attempting to update/delete a CK value in parent table with matching rows in child is dependent on <u>referential action</u> specified using ON UPDATE and ON DELETE subclauses:
•CASCADE - SET NULL
•SET DEFAULT - NO ACTION

<u>CASCADE</u>: Delete row from parent and delete matching rows in child, and so on in cascading manner.
<u>SET NULL</u>: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.
<u>SET DEFAULT</u>: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.
<u>NO ACTION</u>: Reject delete from parent. Default.
FOREIGN KEY (staffNo) REFERENCES Staff ON DELETE SET NULL
FOREIGN KEY (ownerNo) REFERENCES Owner ON UPDATE CASCADE

•general constraints.
•Could use CHECK/UNIQUE in CREATE and ALTER TABLE.
•Similar to the CHECK clause, also have:

```
  CREATE ASSERTION AssertionName
  CHECK (searchCondition)
CREATE ASSERTION StaffNotHandlingTooMuch
CHECK (NOT EXISTS    (SELECT staffNo
  FROM PropertyForRent
  GROUP BY staffNo
  HAVING COUNT(*) > 100))
```