Chapter 7

SQL – Data Definition

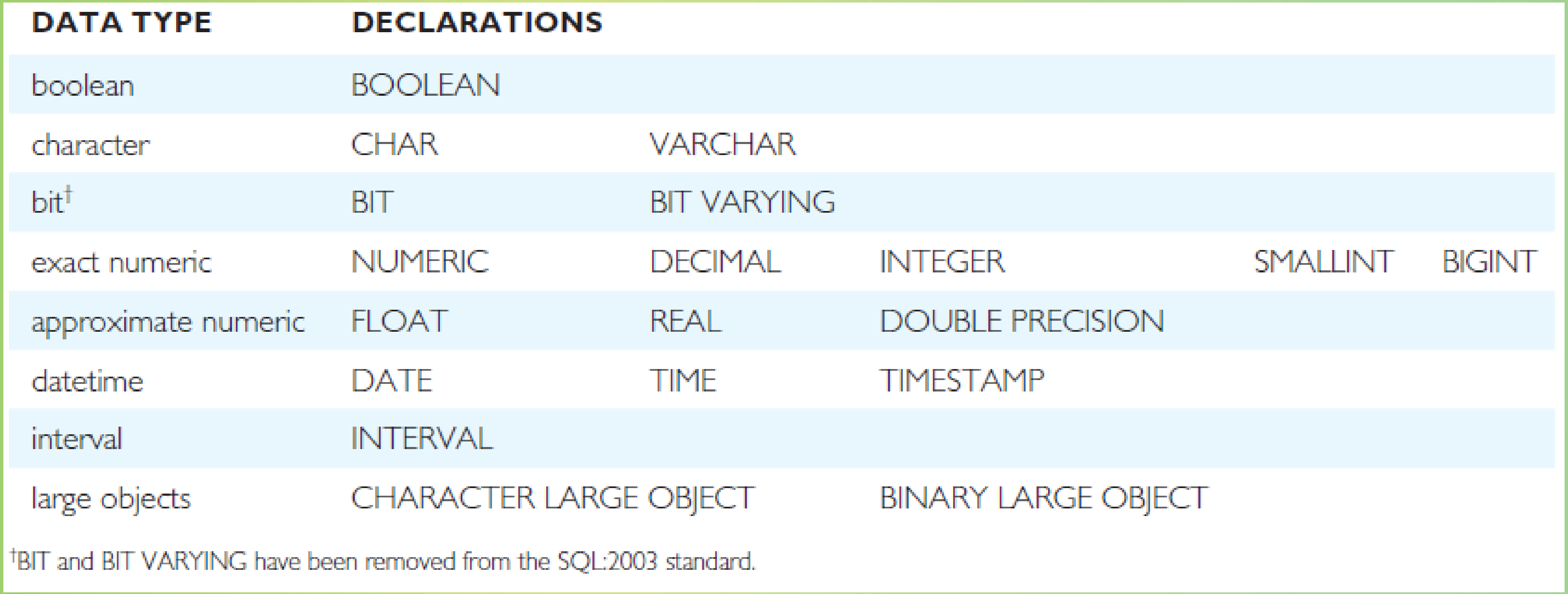
Chapter 7 - Objectives

- Data types supported by SQL standard.
- Purpose of integrity enhancement feature of SQL.
- How to define integrity constraints using SQL.
- How to use the integrity enhancement feature in the CREATE and ALTER TABLE statements.

Chapter 7 - Objectives

- Purpose of views.
- How to create and delete views using SQL.
- How the DBMS performs operations on views.
- Under what conditions views are updatable.
- Advantages and disadvantages of views.
- How the ISO transaction model works.
- How to use the GRANT and REVOKE statements as a level of security.

ISO SQL Data Types



DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit[†]	BIT	BIT VARYING			
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION		
datetime	DATE	TIME	TIMESTAMP		
interval	INTERVAL				
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT		

[†]BIT and BIT VARYING have been removed from the SQL:2003 standard.

Integrity Enhancement Feature

- Five types of integrity constraints:
    1. required data
    2. domain constraints
    3. entity integrity
    4. referential integrity
    5. general constraints.

Integrity Enhancement Feature

<u>Required Data</u>

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

Integrity Enhancement Feature

Domain Constraints

(a) CHECK

- **Format:**

  CHECK (searchCondition)

- **Example:**

  gender CHAR NOT NULL CHECK (gender IN (‘M’, ‘F’))

Integrity Enhancement Feature

(b) CREATE DOMAIN

```
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
```

For example:

```
CREATE DOMAIN genderType AS CHAR
        CHECK (VALUE IN ('M', 'F'));
gender genderType NOT NULL
```

Integrity Enhancement Feature

- *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]
```

IEF - Entity Integrity

- Primary key of a table must contain a unique, non-null value for each row.
- ISO standard supports PRIMARY 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)
```

IEF - 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
```

IEF - Referential Integrity

- 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

IEF - Referential Integrity

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.

SET NULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.

SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.

NO ACTION: Reject delete from parent. Default.

IEF - Referential Integrity

```
FOREIGN KEY (staffNo) REFERENCES Staff
    ON DELETE SET NULL

FOREIGN KEY (ownerNo) REFERENCES Owner
    ON UPDATE CASCADE
```

IEF - General Constraints

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

```
CREATE ASSERTION AssertionName
CHECK (searchCondition)
```

IEF - General Constraints

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

Data Definition

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.
- Main SQL DDL statements are:

CREATE SCHEMA	DROP SCHEMA
CREATE/ALTER DOMAIN	DROP DOMAIN
CREATE/ALTER TABLE	DROP TABLE
CREATE VIEW	DROP VIEW

- Many DBMSs also provide:

CREATE INDEX	DROP INDEX

Data Definition

- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of set of schemas.
- Schema is named collection of related database objects.
- Objects in a schema can be tables, views, domains, assertions, collations, translations, and character sets. All have same owner.

CREATE SCHEMA

- **CREATE SCHEMA [Name | AUTHORIZATION CreatorId]**
- **If the creator of a schema SqlTests is Smith, the SQL statement is:**

CREATE SCHEMA SqlTests **AUTHORIZATION** Smith;

- **A schema can be destroyed using DROP SCHEMA statement:**

DROP SCHEMA Name [RESTRICT | CASCADE]

- **With RESTRICT (default), schema must be empty or operation fails.**
- **With CASCADE, operation cascades to drop all objects associated with schema in order defined above. If any of these operations fail, DROP SCHEMA fails.**

CREATE TABLE

CREATE TABLE TableName

({columName dataType [**NOT NULL**] **[UNIQUE]**

[DEFAULT defaultOption] **[CHECK** (searchCondition)] [, . . .]}

[PRIMARY KEY (listOfColumns),]

{[UNIQUE (listOfColumns)] [, . . .]}

{[FOREIGN KEY (listOfForeignKeyColumns)

REFERENCES ParentTableName [(listOfCandidateKeyColumns)]

[MATCH {PARTIAL | **FULL**} **]**

[ON UPDATE referentialAction]

[ON DELETE referentialAction]] [, . . .]}

{[CONSTRAINT ConstaintName **CHECK** (searchCondition)] [, . . .]});

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL.
- FOREIGN KEY clause specifies FK along with the referential action.

Example 7.1 - CREATE TABLE

Create the PropertyForRent table using the available features of the CREATE TABLE statement.

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT;
        CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PRent AS DECIMAL(6,2)
        CHECK(VALUE BETWEEN 0 AND 9999.99);
```

Example 7.1 - CREATE TABLE

```
CREATE TABLE PropertyForRent (
    propertyNo PNumber     NOT NULL, ….
    rooms      PRooms      NOT NULL DEFAULT 4,
    rent       PRent       NOT NULL,   DEFAULT 600,
    ownerNo    OwnerNumber        NOT NULL,
    staffNo    StaffNumber
                 Constraint StaffNotHandlingTooMuch ….
    branchNo   BranchNumber       NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff
               ON DELETE SET NULL ON UPDATE CASCADE ….);
```

ALTER TABLE

- Add a new column to a table.
- Drop a column from a table.
- Add a new table constraint.
- Drop a table constraint.
- Set a default for a column.
- Drop a default for a column.

ALTER TABLE

ALTER TABLE TableName

[ADD [**COLUMN**] columnName dataType **[NOT NULL**] **[UNIQUE]**

[DEFAULT defaultOption**]** **[CHECK** (searchCondition)]]

[DROP [**COLUMN**] columnName **[RESTRICT** | **CASCADE**]]

[ADD [CONSTRAINT [ConstraintName]] tableConstraintDefinition]

[DROP CONSTRAINT ConstraintName **[RESTRICT** | **CASCADE**]]

[ALTER [**COLUMN**] **SET DEFAULT** defaultOption**]**

[ALTER [**COLUMN**] **DROP DEFAULT]**

Example 7.2(a) - ALTER TABLE

Change Staff table by removing default of 'Assistant' for position column and setting default for gender column to female ('F').

```
ALTER TABLE Staff
ALTER position DROP DEFAULT;
ALTER TABLE Staff
ALTER gender SET DEFAULT 'F';
```

Example 7.2(b) - ALTER TABLE

Remove constraint from PropertyForRent that staff are not allowed to handle more than 100 properties at a time. Add new column to Client table.

```
ALTER TABLE PropertyForRent
    DROP CONSTRAINT StaffNotHandlingTooMuch;
ALTER TABLE Client
    ADD prefNoRooms PRooms;
```

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

e.g. DROP TABLE PropertyForRent;

- **Removes named table and all rows within it.**
- **With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.**
- **With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).**

Views

View

Dynamic result of one or more relational operations operating on base relations to produce another relation.

- Virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.

Views

- Contents of a view are defined as a query on one or more base relations.
- With view resolution, any operations on view are automatically translated into operations on relations from which it is derived.
- With view materialization, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.

SQL - CREATE VIEW

```
CREATE VIEW ViewName [ (newColumnName [,...]) ]
    AS subselect
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

- Can assign a name to each column in view.
- If list of column names is specified, it must have same number of items as number of columns produced by *subselect*.
- If omitted, each column takes name of corresponding column in *subselect*.

SQL - CREATE VIEW

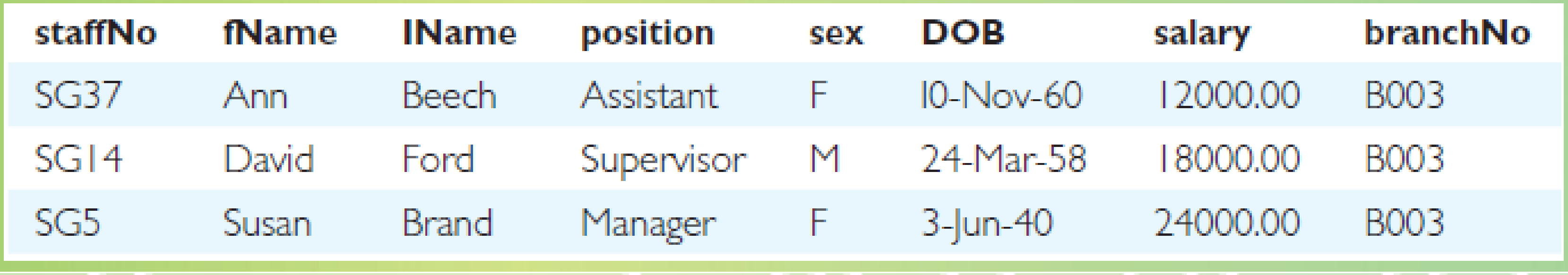
- List must be specified if there is any ambiguity in a column name.
- The *subselect* is known as the <u>defining query</u>.
- WITH CHECK OPTION ensures that if a row fails to satisfy WHERE clause of defining query, it is not added to underlying base table.
- Need SELECT privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns.

Example 7.3 - Create Horizontal View

Create view so that the manager at branch B003 can only see details for staff who work in his or her office.

```
CREATE VIEW Manager3Staff
AS      SELECT *
        FROM Staff
        WHERE branchNo = 'B003';
```

```
SELECT * FROM Manager3Staff;
```

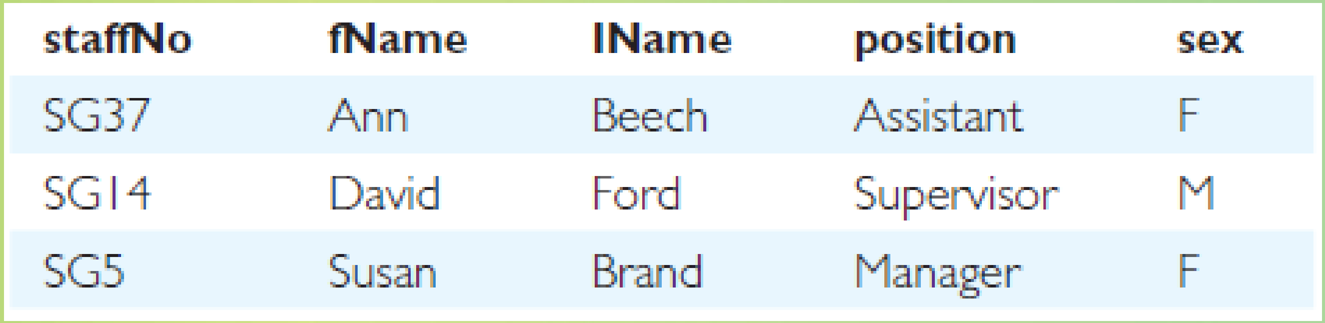


staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

Example 7.4 - Create Vertical View

Create a view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff3
AS SELECT staffNo, fName, lName, position, gender
   FROM Staff
   WHERE branchNo = ‘B003’;
```



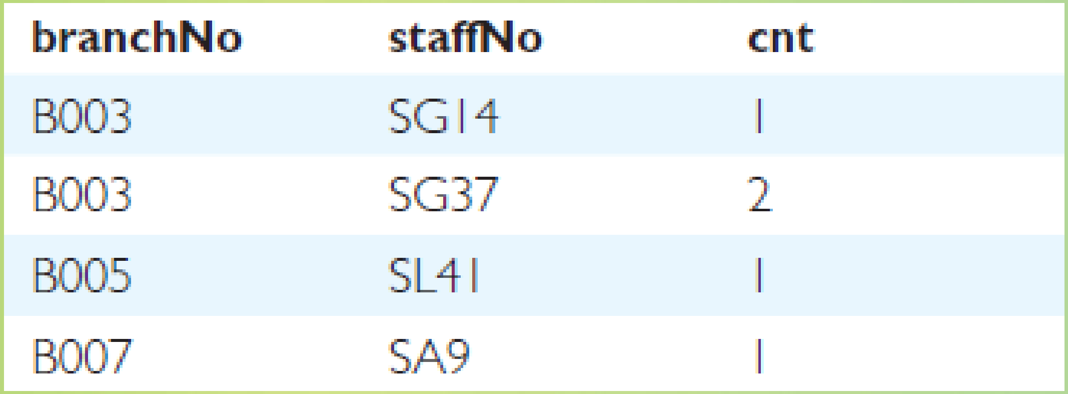
staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

Example 7.5 - Grouped and Joined Views

Create a view of staff who manage properties for rent, including branch number they work at, staff number, and number of properties they manage.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
 AS SELECT s.branchNo, s.staffNo, COUNT(*)
      FROM Staff s, PropertyForRent p
      WHERE s.staffNo = p.staffNo
      GROUP BY s.branchNo, s.staffNo;
```

Example 7.3 - Grouped and Joined Views



branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

SQL - DROP VIEW

```
DROP VIEW ViewName [RESTRICT | CASCADE]
```

- Causes definition of view to be deleted from database.
- For example:

```
DROP VIEW Manager3Staff;
```

SQL - DROP VIEW

- **With CASCADE, all related dependent objects are deleted; i.e. any views defined on view being dropped.**
- **With RESTRICT (default), if any other objects depend for their existence on continued existence of view being dropped, command is rejected.**

View Resolution

Count number of properties managed by each member at branch B003.

```
SELECT staffNo, cnt
FROM StaffPropCnt
WHERE branchNo = ‘B003’
ORDER BY staffNo;
```

View Resolution

(a) View column names in SELECT list are translated into their corresponding column names in the defining query:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

(b) View names in FROM are replaced with corresponding FROM lists of defining query:

```
FROM Staff s, PropertyForRent p
```

View Resolution

(c) WHERE from user query is combined with WHERE of defining query using AND:

```
WHERE s.staffNo = p.staffNo AND branchNo = ‘B003’
```

(d) GROUP BY and HAVING clauses copied from defining query:

```
GROUP BY s.branchNo, s.staffNo
```

(e) ORDER BY copied from query with view column name translated into defining query column name

```
ORDER BY s.staffNo
```

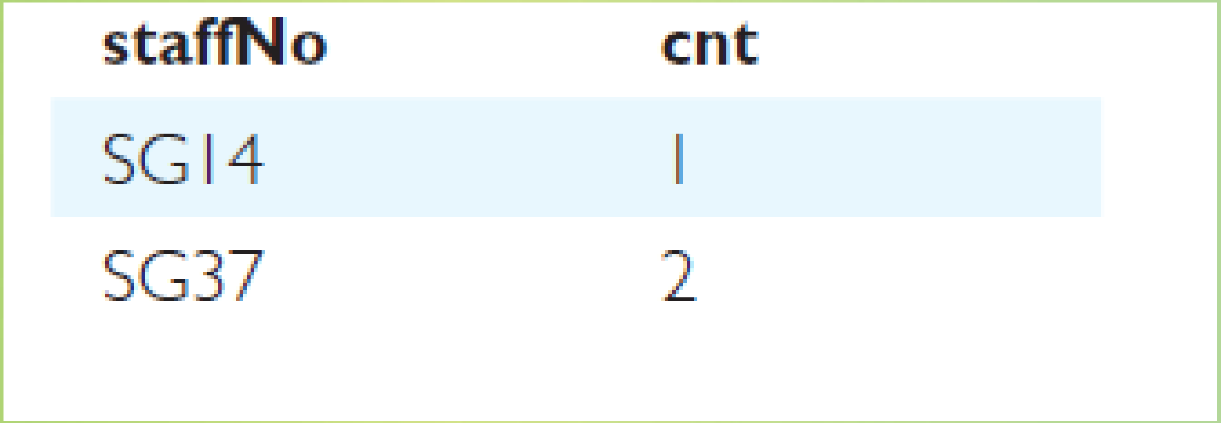
View Resolution

(f) Final merged query is now executed to produce the result:

```
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo AND
            branchNo = ‘B003’
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

View Resolution

This gives the result table shown below



staffNo	cnt
SG14	1
SG37	2

Restrictions on Views

SQL imposes several restrictions on creation and use of views.

(a) If column in view is based on an aggregate function:

- Column may appear only in SELECT and ORDER BY clauses of queries that access view.
- Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.

Restrictions on Views

- For example, following query would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

- Similarly, following query would also fail:

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

Restrictions on Views

(b) Grouped view may never be joined with a base table or a view.

- **For example, StaffPropCnt view is a grouped view, so any attempt to join this view with another table or view fails.**

View Updatability

- All updates to base table reflected in all views that encompass base table.
- Similarly, may expect that if view is updated then base table(s) will reflect change.

View Updatability

- However, consider again view StaffPropCnt.
- If we tried to insert record showing that at branch B003, SG5 manages 2 properties:

```
INSERT INTO StaffPropCnt
VALUES (‘B003’, ‘SG5’, 2);
```

- Have to insert 2 records into PropertyForRent showing which properties SG5 manages. However, do not know which properties they are; i.e. do not know primary keys!

View Updatability

- If we change definition of view and replace count with actual property numbers as follows:

```
CREATE VIEW StaffPropList (branchNo,
                    staffNo, propertyNo)
AS SELECT s.branchNo, s.staffNo, p.propertyNo
    FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo;
```

View Updatability

- Now try to insert the record:

```
INSERT INTO StaffPropList
VALUES ('B003', 'SG5', 'PG19');
```

- Still problem, because in PropertyForRent all columns except postcode/staffNo are not allowed nulls.
- However, have no way of giving remaining non-null columns values.

View Updatability

- **ISO specifies that a view is updatable if and only if:**

  - DISTINCT is not specified.

  - Every element in SELECT list of defining query is a column name and no column appears more than once.

  - FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.

  - No nested SELECT referencing outer table.

  - No GROUP BY or HAVING clause.

  - Also, every row added through view must not violate integrity constraints of base table.

Updatable View

For a view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source table.