Chapter 4

Intermediate SQL: Integrity Constraints
Integrity Constraints

- Ensure *Data Consistency*

- *Domain Constraints*
  - enforce valid attribute values from domain sets
  - domain set may exclude null values

- Referential Integrity (especially Foreign-Key Integrity)
  - a constraint involving two relations
  - used for specifying a relationship among tuples in two relations, the referencing relation and the referenced relation
  - (foreign-Key Constraint) tuples in the *referencing relation*, \( r_2 \), have attributes \( FK \) (called foreign key) that reference the primary key \( PK \) of the *referenced relation* \( r_1 \)
  - \( FK \) of \( r_2 \) is a foreign key of \( r_1 \) with primary key \( PK \) if
    \[
    \forall t_2 \in r_2, \exists t_1 \in r_1 \text{ such that } t_1[PK] = t_2[FK]
    \]
Integrity Constraints

- **Updates on foreign keys:** Let $FK$ of $r_2$ be a foreign key that references the primary key $PK$ of $r_1$. To enforce the referential integrity:

  - **Insert:** add $t_2$ to $r_2$. Then $t_2[FK] \in \pi_{PK}(r_1)$; otherwise, reject

  - **Delete:** remove $t_1$ from $r_1$. If $t_1[PK] \in \pi_{FK}(r_2)$, then
    - Action 1. Deny the deletion request (an inappropriate option), or
    - Action 2. **Delete** all tuples $t$ in $r_2$ such that $t[FK] = t_1[PK]$

- **Modify:**
  - **Referencing relation** - if $t_2[FK]$ in $r_2$ is modified to $t_2'[FK]$, then it must be the case that $t_2'[FK] \in \pi_{PK}(r_1)$; otherwise, reject
  - **Referenced relation** - if $t_1[PK]$ in $r_1$ is modified to $t_1'[PK]$, then use the deletion policy for modification on $t_1[PK]$

- **Referential integrity in SQL** - indicate primary key/candidate key/foreign key during the creation of a table
Example.

Create table customer
(\textit{customer-name} \texttt{char}(20) \texttt{not null},
\textit{customer-street} \texttt{char}(30),
\textit{customer-city} \texttt{char}(30),
\texttt{primary key} (\textit{customer-name}))

Create table branch
(\textit{branch-name} \texttt{char}(15) \texttt{not null},
\textit{branch-city} \texttt{char}(30),
\textit{assets} \texttt{integer},
\texttt{primary key} (\textit{branch-name}),
\texttt{check} (\textit{assets} > 0))

Create table account
(\textit{account-number} \texttt{char}(10) \texttt{not null},
\textit{branch-name} \texttt{char}(15),
\textit{balance} \texttt{integer},
\texttt{primary key} (\textit{account-number}),
\texttt{foreign key} (\textit{branch-name}) \texttt{references} branch,
\texttt{check} (\textit{balance} \geq 0))

Create table depositor
(\textit{customer-name} \texttt{char}(20) \texttt{not null},
\textit{account-number} \texttt{char}(10) \texttt{not null},
\texttt{primary key} (\textit{customer-name}, \textit{account-number}),
\texttt{foreign key} (\textit{customer-name}) \texttt{references} customer,
\texttt{foreign key} (\textit{account-number}) \texttt{references} account)
Given the following relations:

<table>
<thead>
<tr>
<th>Student</th>
<th>Course#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young</td>
<td>CS 451</td>
</tr>
<tr>
<td>Adams</td>
<td>EE 321</td>
</tr>
<tr>
<td>Lee</td>
<td>CS 451</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE 321</td>
<td>Circuit Design</td>
</tr>
<tr>
<td>Eng 316</td>
<td>Tech Wrtg</td>
</tr>
<tr>
<td>CS 451</td>
<td>DBMS</td>
</tr>
</tbody>
</table>

where Course# in Course is the primary key for Course and Course# in Is.taking is a foreign key.

Discuss what actions should be taken for the following database update requests so that violations of referential integrity will not occur.

(a) [4 pts] Modify CS 451 in the Course relation to CS 353.

(b) [4 pts] Insert the tuple < Smith, Math 411 > into the Is.taking relation.

(c) [4 pts] Delete the tuple < EE 321, Circuit Design > from the Course relation.
SQL: CREATE TABLE Statements

- Basic syntax of the CREATE TABLE statement:

  CREATE TABLE <table name> (<table element list>)
  
  <table element list> ::= <table element> | <table element>, <table element list>
  
  <table element> ::= <column definition> | <table constraint definition>

- The table definition (relational properties in Oracle) includes:
  - *data* in the column definitions, and
  - *rules* for handling data in the table constraint definition

- A table is more like an *object* (with its data and methods) than just a simple passive file.
Relational Properties (Oracle 8i)
SQL: Column Definitions in CREATE TABLE

- Column definitions:

  \[
  \text{<column definition>} ::= \text{<column name>} \text{ <data type>}
  \[
  \text{[<default clause>]}
  \[
  \text{[<column constraint …>]}\]
  \]

- Each column name must have a \text{data type} (e.g. numeric, char strings, \textit{bit}), and \textit{optional} default values and constraints.

- Each column definition can be modified by using a ALTER TABLE statement.

- The default clause specify the \textit{default value} to be inserted into the column if an explicit value does not exist.
SQL: Column Definitions in CREATE TABLE

- Default Clause:

  `<default clause> ::= DEFAULT <default option>`

  `<default option> ::= <literal> | <system value> | NULL`

- A literal value is a string of alphabetic or numeric characters.

- A system value can be current timestamp/date/current user ID.

- Example.

  CREATE TABLE Account
  (Acc_num INTEGER DEFAULT 1,
   Acc_type CHAR(1) DEFAULT 'A',
   Acc_descr CHAR(20) DEFAULT 'New Account');
Column Definitions in CREATE TABLE: Default Clause

- If a DEFAULT clause is not provided, the column definition acts as if DEFAULT NULL has been declared.

- If NULL is the default value for a column, then the NOT-NULL constraint cannot be part of the column definition.

- Most commonly used default values include
  - ‘0’ in numeric columns
  - “Unknown” for a missing string value
  - “System timestamp” to mark a transaction, etc.

- NULL cannot be the default value for a PRIMARY KEY column, even though DEFAULT NULL PRIMARY KEY is allowed.

- Built-in functions can be specified as the default values.
Column Constraints in CREATE TABLE: Column Constraints

- Syntax:

  \[
  \text{<column constraint>} ::= \text{[CONSTRAINT constraint\_name]} \mid \text{NOT NULL} \mid \text{<UNIQUE/PK specification>} \mid \text{<REFERENCES specification>} \mid \text{<CHECK constraint definition>}
  \]

- Constraint rules, as defined in a table creation statement, enforce data-integrity constraints on columns.

- All the (new/modified) rows in the table are validated against them.
Column Constraints (Oracle 8i)
A constraint **forbids** the use of NULLs in a column; **NOT NULL** constraint requires that a column receives a value during *insert* or *update* operations.

If there is a (i) **NOT NULL** constraint and (ii) no default value is specified on a column, then a value **must** be entered into the column.

**NULL** is a special value in SQL that belongs to all data types.

The interpretation of a **NULL**: *missing, unknown, inapplicable, and miscellaneous*, and NULLs cause a lot of irregular features in SQL, e.g. the strange 3-valued logic.

**Example.**

```sql
CREATE TABLE items
(Item_num INTEGER,
Menu_code CHAR(3) NOT NULL,
Descrip CHAR(20));
```
Column Constraints (Oracle 8i)
Column Constraints in CREATE TABLE: Unique & Primary Key

- Syntax:

  \[<\text{unique specification}> := \text{UNIQUE} | \text{PRIMARY KEY}\]

- UNIQUE constraint states that no duplicate values are allowed in the column

- Differences between UNIQUE and PRIMARY KEY:
  1) There can be only one PRIMARY KEY but many UNIQUE columns
  2) PRIMARY KEY is automatically declared to have a NOT NULL constraint, but a UNIQUE column can have NULL value

Example.

CREATE TABLE Account
  (Acct_num INTEGER PRIMARY KEY,
   Acct_code INTEGER UNIQUE);
Column Constraints in CREATE TABLE: Unique & Primary Key

- When a PRIMARY KEY constraint is created, the DBS automatically creates an *internal index* on the column(s).

- Multiple-column of the form `<unique specification>`, which means that the combination of a number of columns is unique, is allowed, e.g.,

  PRIMARY KEY(Account_Number, Customer_Name)

**Example.**

```sql
CREATE TABLE depositor
  (Acct_num char(10),
   Customer_name char(15),
   PRIMARY KEY(Acct_num, Customer_name));
```

A *Table-definition constraint*, not a *Column-definition constraint*
Column Constraints (Oracle 8i)
Column Constraints in CREATE TABLE: REFERENCE

- Syntax:

  \[
  \text{<reference specification> ::= REFERENCES } \text{<referenced table name>}
  \]
  
  \[
  [\langle \text{<reference column>} \rangle]
  \]

  - If no \langle \text{<reference column>} \rangle is specified, then the PRIMARY KEY column of the referenced table is assumed to be the target.

- Specify referential integrity (foreign key integrity)

- Example.

  CREATE TABLE account
  (Account# char(10),
   Branch_Name char(15) REFERENCES branch,
   Balance number(4, 2),
   PRIMARY KEY(Account#));

  OR

  CREATE TABLE account
  (Account# char(10),
   Branch_Name char(15),
   Balance number(4, 2),
   PRIMARY KEY(Account#),
   FOREIGN KEY (Branch_Name) REFERENCES branch);

  - Several columns can reference the same target column.

  - A PRIMARY KEY value cannot be deleted from the referenced table if it is referenced in a referencing table (without cascading).
Column Constraints in CREATE TABLE: REFERENCE

- REFERENCE Clauses with ON DELETE CASCADE

- Use the ON DELETE CASCADE option to specify the deletion from the referencing table when the corresponding referenced rows are deleted from the referenced table.

- Reduces the quantity of SQL statements that are required for performing the delete operations.

- **Example.**

  ```sql
  CREATE TABLE depositor
  (Acct_num CHAR(10) REFERENCES account
    ON DELETE CASCADE,
  Customer_name CHAR(15),
  Bname CHAR(15) REFERENCES branch (Branch_name)
    ON DELETE SET NULL,
  PRIMARY KEY(Acct_num, Customer_name);
  ```
Column Constraints in CREATE TABLE: REFERENCE

- Default values for the Referenced Column:
  - If the referenced table is different from the referencing table, then by default, the referenced column is the primary-key column of the referenced table.
  - If the referenced and referencing tables are the same, then the referenced column must be specified.

- Referential Relationships within a Table
  - A referential relationship between two columns of the same table can be established, e.g.,

```
CREATE TABLE Employee
  (Emp_num INTEGER,
   Mgr_num INTEGER REFERENCES Employee (Emp_num),
   PRIMARY KEY(Emp_num));
```
The column referenced by a FOREIGN KEY can be either a PRIMARY KEY or a column with a UNIQUE constraint.

In SQL-92, the following constraints are equivalent:

- PRIMARY KEY $\equiv$ CHECK (UNIQUE (SELECT <key columns> FROM <table>)) AND (<key columns> IS NOT NULL)
- UNIQUE $\equiv$ CHECK(UNIQUE (SELECT <key column> FROM <table>))
- NOT NULL $\equiv$ CHECK(<column> IS NOT NULL)
Column Constraints (Oracle 8i)
Syntax:

<check constraint constraint> ::= CHECK <search condition>

Tests the rows of a table against a *logical expression*, which is called *search condition*

- *Rejects* rows whose search condition returns FALSE.
- *Accepts* rows when the search condition returns TRUE (or in other implementations, UNKNOWN, a “benefit-of-the doubt" feature).

- *Range* checking, e.g., CHECK(rating BETWEEN 1 AND 10)

- Checking on values in *enumerated sets*, e.g., CHECK(Color IN ('Red’, ‘Blue’, ‘Orange’, ‘Green’, ‘Yellow’))

- Subqueries cannot be specified in a CHECK clause
Column Constraints in CREATE TABLE: Multiple-Column

- Usage: associate one or more columns in a table with a CHECK constraint.

  - Example.

  ```
  CREATE TABLE Multi-Ref
  (Acct_Id CHAR(10),
  Loan_Amt INTEGER,
  Credit INTEGER,
  CHECK (Credit > 0 AND Loan_Amt < 9999),
  CHECK (Credit > Loan_Amt));
  ```

- (In some DBMSs) When the constraint is created on a number of columns

  - the columns *cannot* be assigned any default values, and
  - referential relationships *cannot* be established between them
Column Constraints (Oracle 8i)
Optional in a check constraint clause, but is a good idea to use it.

Constraint names appear in error messages when corresponding constraint violations occur.

Examples.

```sql
CREATE TABLE account
(Account# char(10),
Branch_Name char(15),
Balance number(4, 2),
CONSTRAINT Min_Balance
CHECK (balance >= 0));
```

```sql
CREATE TABLE depositor
(Account# char(10),
Customer_Name char(15),
CONSTRAINT Acct_Verification
CHECK To_NUMBER(Account#) BETWEEN 1 AND 9999));
```

When a constraint is created, the DBS adds a row for that constraint to the system catalog table.

CHECK() clause allows complex expressions to be specified that verify relationships among rows/tables/constants.
SQL: ALTER TABLE

- Use the ALTER TABLE statement to *update* (add, delete, modify) the *definition* of (columns in) a table.

```
ALTER TABLE <table name> <ALTER TABLE Action>
```

**<ALTER TABLE Action> ::=**
- ADD (<Column Definition>) |
- DROP COLUMN <Column Name> [CASCADE CONSTRAINTS] |
- DROP CONSTRAINT
- MODIFY (<Column Definition>) |
- MODIFY CONSTRAINT <Constraint Name><Constraint_State> | …

- Add Column
  - Extends the existing table by putting a new, unique column on it.
  - Example.

```
ALTER TABLE Account
ADD Customer_ID CHAR(5) DEFAULT '00000' NOT NULL;
```
ADD Table-Constraint Option (Oracle 8i)
ALTER TABLE: Add Table Constraints

- Specify a new constraint in a table
- Some DBMS requires the usage of ALTER TABLE statements to add *referential integrity constraints* to a table, rather than allowing the declaration be made at schema creation time.

- **Example.**

  ```sql
  ALTER TABLE Account
  ADD FOREIGN KEY(Branch_name) REFERENCES Branch
  ADD PRIMARY KEY(Account_number)
  ADD CONSTRAINT Pos_Bal CHECK (Balance > 0);
  ```
Drop-Column Option (Oracle 8i)
ALTER TABLE: Drop Column

- Removes an existing column from a table.

- CASCADE CONSTRAINTS drops all *referential integrity* constraints defined on the dropped columns that are either *primary* or *unique keys*.

- Without the CASCADE CONSTRAINTS clause, if any constraint is referenced by
  - columns from other tables, or
  - remaining columns in the same table,
  the DROP COLUMN statement *aborts* and an *error* is returned.

- **Example.**

  ```sql
  ALTER TABLE Account
  DROP COLUMN Customer_ID CASCADE CONSTRAINTS;
  ```
Drop-Constraint Clause (Oracle 8i)
ALTER TABLE: Drop Constraint

- Drop an existing constraint in a table:
  - Default values, (NOT-)NULLs, Primary Key, Unique, etc.
  - If the modified column is referenced by other tables, then those referential constraints could not be dropped, unless the CASCADE clause is specified.
    - To restore the constraints to the referencing tables again, e.g., foreign key, use the Alter Table statement.

- Example.

  ```sql
  ALTER TABLE Branch
  DROP PRIMARY KEY CASCADE;
  ```
Column Constraints (Oracle 8i)

Column_constraints:

```
CONSTRANIT  constraint
  NOT      NULL
  UNIQUE   
  PRIMARY  KEY
  REFERENCES
    schema
    table
    column
  condition

CHECK   
constraint_state

FOREIGN   KEY
         column
         REFERENCES
            schema

ON      DELETE
          CASCADE
          SET     NULL

ON      DELETE
          CASCADE
          SET     NULL
```

Foreign_Key Clause:
Cascading Actions in SQL

- If there is a chain of foreign-key dependencies across multiple relations, with **on delete cascade** specified for each dependency, a **deletion** or **update** at one end of the chain can propagate across the entire chain.

- If a **cascading update to delete** causes a **constraint violation** that cannot be handled by a further cascading operation, the system aborts the transaction.
  - As a result, all the changes caused by the transaction and its cascading actions are undone.

- Referential integrity is only checked at the end of a transaction
  - Intermediate steps are allowed to violate referential integrity provided later steps remove the violation
  - Otherwise, it would be impossible to create some database states, e.g., insert two tuples whose foreign keys point to each other (e.g., *spouse* attribute of relation *married-person*)
An example on specifying cascading actions in SQL:

```
CREATE TABLE account
    ...
    FOREIGN KEY(branch-name) REFERENCES branch
    ON DELETE CASCADE
    ON UPDATE CASCADE
    ...
```

Due to the ON DELETE CASCADE clauses, if a deletion of a tuple in `branch` results in referential-integrity constraint violation, the delete “cascades” to the `account` relation and will delete the tuple that refers to the branch.

Cascading updates are similar.
Cascading Actions in SQL

- Alternative to cascading:
  - on delete set null
  - on delete set default

- Null values in foreign key attributes complicate SQL referential integrity semantics, and are best prevented using not null
Modify-Column Option (Oracle 8i)

Column Definition:

Column_Constraint:
ALTER TABLE: Modify Column

- Changes an existing column and its definition
- Exactly what can be modified vary from DBMS to DBMS
- Some of the updates that can be made in Oracle 8i:
  - modify a data type to a compatible data type, e.g., INTEGER to REAL
  - allow NULL by changing NOT NULL (or DEFAULT) to NULL
  - add NOT NULL constraint clause

- Example.

```
ALTER TABLE Account
Modify (Balance number(5,2) DEFAULT 0,
      Branch_name NOT NULL);
```
ALTER TABLE: NULL Values

- One can modify an existing column that did not permit NULLs to permit NULLs.
  
  - **Example.**
    
    ```sql
    ALTER TABLE Account
    MODIFY Customer_ID CHAR(6) NULL;
    ```

- One can change a column from “allows NULL values” to “disallow NULL values,” provided that the column contains no NULL values.
  
  - **Example.** See example on previous slides.

- **Alternative:** one can permit a non-NULL column to allow NULLs by using the DROP CONSTRAINT clause to drop the NOT-NULL constraint on the column.
Modify Constraint State Option (Oracle 8i)
ALTER TABLE: Modify Constraint State

- Modify Constraint *constraint_name* *Constraint State*

- Modify an existing constraint named *constraint* in a table to a new constraint state.

  - Possible *constraint states*:
    - Enable/Disable
    - Validate/No Validate
    - Initially Immediate/Deferred

  - **Example**.

    ALTER TABLE Depositor
    MODIFY CONSTRAINT Acct_verification ENABLE;
SQL: Char Functions

- **Substr(String, m, [, n])**

  - Extract a substring of *String*, beginning at position *m*.
    - The extracted substring is of length *n*.
    - If *n* is omitted, then the default is the substring from position *m* to the end of *String*.

  - **Example.**

    Let *String* = ‘ABCDE’. Then
    
    \[
    \text{substr(String, 2, 3)} = \text{’BCD’}, \text{ and } \\
    \text{substr(String, 4)} = \text{’DE’}
    \]