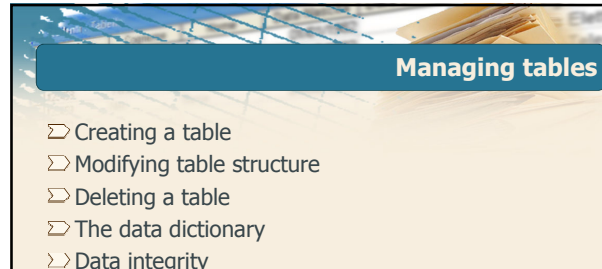


SQL language: basics

Managing tables


DBG



Managing tables

- Creating a table
- Modifying table structure
- Deleting a table
- The data dictionary
- Data integrity

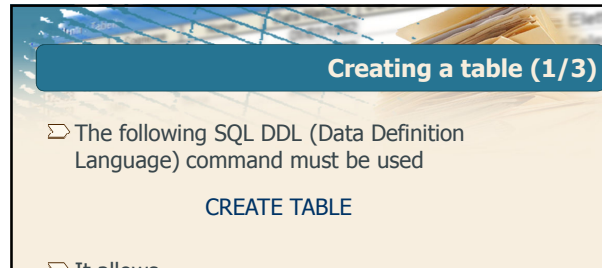
DBG



Managing tables

Creating a table

DBG



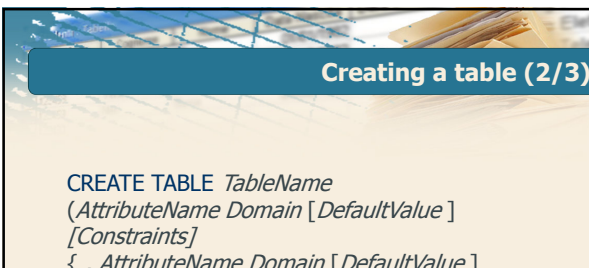
Creating a table (1/3)

- The following SQL DDL (Data Definition Language) command must be used

CREATE TABLE

- It allows
 - defining all attributes (i.e., columns) in the table
 - defining integrity constraints on the table data

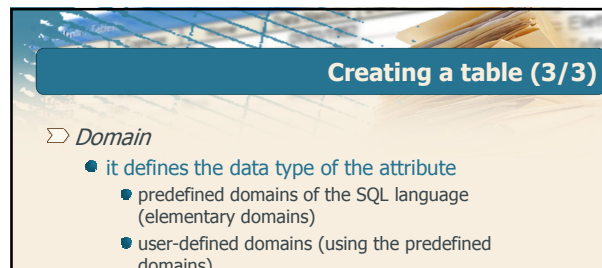
DBG



Creating a table (2/3)

```
CREATE TABLE TableName
(AttributeName Domain [DefaultValue]
[Constraints]
{ , AttributeName Domain [DefaultValue]
[Constraints]}
OtherConstraints
);
```

DBG



Creating a table (3/3)

- *Domain*
 - it defines the data type of the attribute
 - predefined domains of the SQL language (elementary domains)
 - user-defined domains (using the predefined domains)
- *Constraints*
 - it allows specifying integrity constraints for the attribute
- *OtherConstraints*
 - it allows specifying general integrity constraints on the table

DBG

Domain definition (1/2)

- ▷ *Default Value*

- it allows specifying a default value for the attribute

DEFAULT

< *GenericValue* | USER | CURRENT_USER |
SESSION_USER | SYSTEM_USER | NULL >



Domain definition (2/2)

- ▷ *GenericValue*

- a value compatible with the attribute domain

- ▷ *USER

- user identifier

- ▷ NULL

- base default value



Elementary domains (1/6)

- ▷ Character: single characters or strings (possibly variable-length)

CHARACTER [VARYING] [(*Length*)]
[CHARACTER SET *CharacterFamilyName*]

- VARCHAR for short

- ▷ Single bits (booleans) or bit strings

BIT [VARYING] [(*Length*)]



Elementary domains (2/6)

- ▷ Exact numeric domains

NUMERIC [(*Precision*, *Scale*)]

DECIMAL [(*Precision*, *Scale*)]

INTEGER

SMALLINT

- ▷ NUMERIC and DECIMAL are base-ten numbers



Elementary domains (3/6)

NUMERIC [(*Precision*, *Scale*)]

DECIMAL [(*Precision*, *Scale*)]

- ▷ Precision

- total number of digits
 - for the NUMERIC domain, precision represents an exact requirement
 - for the DECIMAL domain, precision is a minimum requirement



Elementary domains (3/6)

NUMERIC [(*Precision*, *Scale*)]

DECIMAL [(*Precision*, *Scale*)]

- ▷ Scale

- number of decimal places

- ▷ Example: for number 123.45

- precision is 5, scale is 2



Elementary domains (4/6)

- ▷ Approximate numeric domains
 - FLOAT [(*n*)]
 - REAL
 - DOUBLE PRECISION
- ▷ *n* specifies precision
 - it is the number of bits used to store the mantissa of a floating point number represented in scientific notation
 - it is a value ranging from 1 to 53
 - the default value is 53



Elementary domains (5/6)

INTERVAL *FirstUnitOfTime*
[TO *LastUnitOfTime*]

- ▷ Units of time are divided into two groups
 - year, month
 - day, hour, minute, second
- ▷ Example: INTERVAL year TO month
 - stores a period of time using the year and month fields
- ▷ Example: INTERVAL day TO second
 - stores a period of time using the day, hour, minute and second field



Elementary domains (6/6)

- ▷ TIMESTAMP [(*Precision*)] [WITH TIME ZONE]
 - it stores the values specifying the year, the month, the day, the hour, the minutes, the seconds and possibly the fraction of second
 - it uses 19 characters, plus the characters needed to represent the precision
 - notation
 - YYYY-MM-DD hh:mm:ss:p



Defining a domain (1/2)

- ▷ CREATE DOMAIN command
 - it defines a new domain that may be used in attribute definitions
- ▷ Syntax


```
CREATE DOMAIN DomainName AS DataType
  [ DefaultValue ] [ Constraint ]
```
- ▷ *DataType* is an elementary domain



Defining a domain (2/2)

- ▷ Example


```
CREATE DOMAIN Grade AS SMALLINT
  DEFAULT NULL
  CHECK (Grade >= 18 and Grade <=30)
```



Definition of the supplier and product DB

- ▷ Creation of the supplier table

S			
Sid	SName	#Employees	City

```
CREATE TABLE S (Sid          CHAR(5),
                 SName       CHAR(20),
                 #Employees  SMALLINT,
                 City         CHAR(15));
```

- ▷ The definition of integrity constraints is missing



Definition of the supplier and product DB

⇒ Creation of the product table

P				
PId	PName	Color	Size	Store

```
CREATE TABLE P (PId      CHAR(6),
                PName    CHAR(20),
                Color    CHAR(6),
                Size     SMALLINT,
                Store    CHAR(15));
```

⇒ The definition of integrity constraints is missing



Definition of the supplier and product DB

⇒ Creation of the supplier-product table

SP		
SId	PId	Qty

```
CREATE TABLE SP (SId   CHAR(5),
                 PId   CHAR(6),
                 Qty   INTEGER);
```

⇒ The definition of integrity constraints is missing



Managing tables

Modifying table structure



The ALTER TABLE command (1/3)

- ⇒ The following "alterations" are possible
- adding a new column
 - defining a new default value for an existing column (attribute)
 - for example, replacing a previous default value
 - deleting an existing column (attribute)
 - defining a new integrity constraint
 - deleting an existing integrity constraint



The ALTER TABLE command (2/3)

```
ALTER TABLE TableName
< ADD COLUMN <Attribute-Definition> |
  ALTER COLUMN AttributeName
    < SET <Default-Value-Definition> | DROP DEFAULT > |
  DROP COLUMN AttributeName
    < CASCADE | RESTRICT > |
  ADD CONSTRAINT [ConstraintName]
    < unique-constraint-definition > |
    < referential-integrity-constraint-definition > |
    < check-constraint-definition > |
  DROP CONSTRAINT [ConstraintName]
    < CASCADE | RESTRICT >
```



The ALTER TABLE command (3/3)

- ⇒ RESTRICT
- the element (column or constraint) is not removed if it appears in the definition of some other element
 - default option
- ⇒ CASCADE
- all elements with a dependency on a deleted element will be removed, until there are no unresolved dependencies (i.e., there are no more elements whose definition references a deleted element)



The ALTER TABLE command: example no.1

➤ Add column #Members to the supplier table

S				
SId	SName	#Employees	City	#Members

```
ALTER TABLE S
ADD COLUMN #Members SMALLINT;
```



The ALTER TABLE command: example no.2

➤ Delete column #Employees from the supplier table

S			
SId	SName	#Employees	City

```
ALTER TABLE S
DROP COLUMN #Employees RESTRICT;
```



The ALTER TABLE command: example no.3

➤ Add a default value of 0 to column Quantity of the supplier-product table

SP		
SId	PId	Qty

```
ALTER TABLE SP
ALTER COLUMN Qty SET DEFAULT 0;
```



Managing tables

Deleting a table



Deleting a table

```
DROP TABLE TableName
[RESTRICT | CASCADE];
```

➤ All of the table rows are deleted along with the table

➤ RESTRICT

- the table is not deleted if it appears in the definition of some table, constraint or view
- default option

➤ CASCADE

- if the table appears in the definition of some view, the latter is also deleted




Deleting a table: example

➤ Delete the supplier table

S			
SId	SName	#Employees	City

```
DROP TABLE S;
```

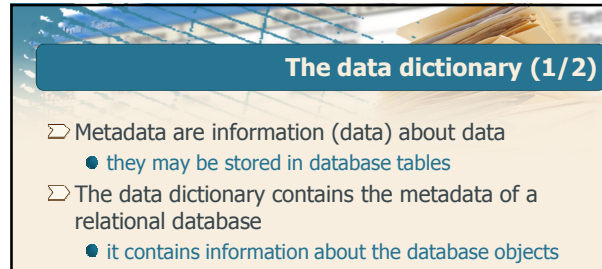




Managing tables

The data dictionary


DBG



The data dictionary (1/2)

- ⊃ Metadata are information (data) about data
 - they may be stored in database tables
- ⊃ The data dictionary contains the metadata of a relational database
 - it contains information about the database objects
 - it is managed directly by the relational DBMS
 - it may be queried by means of SQL commands

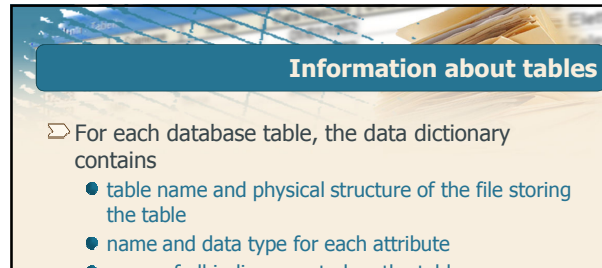
DBG



The data dictionary (2/2)

- ⊃ It contains various pieces of information
 - descriptions of all database structures (tables, indices, views)
 - SQL stored procedures
 - user privileges
 - statistics
 - on the database tables
 - on the database indices
 - on the database views
 - on the evolution of the database

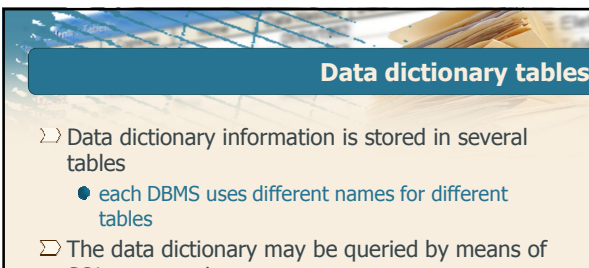
DBG



Information about tables

- ⊃ For each database table, the data dictionary contains
 - table name and physical structure of the file storing the table
 - name and data type for each attribute
 - name of all indices created on the table
 - integrity constraints

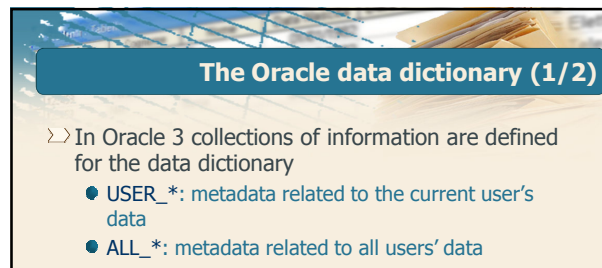
DBG



Data dictionary tables

- ⊃ Data dictionary information is stored in several tables
 - each DBMS uses different names for different tables
- ⊃ The data dictionary may be queried by means of SQL commands

DBG



The Oracle data dictionary (1/2)

- ⊃ In Oracle 3 collections of information are defined for the data dictionary
 - USER_*: metadata related to the current user's data
 - ALL_*: metadata related to all users' data
 - DBA_*: metadata about system tables

DBG

The Oracle data dictionary (2/2)

- ↳ USER_* contains different tables and views, including:
- USER_TABLES contains metadata to the user tables
 - USER_TAB_STATISTICS contains statistics computed on the user tables
 - USER_TAB_COL_STATISTICS contains statistics computed on user table columns



Querying the data dictionary no.1

- ↳ Show the name of user-defined tables and the number of tuples stored in each table

```
SELECT Table_Name, Num_Rows
FROM USER_TABLES;
```

R

Table_Name	Num_Rows
S	5
P	6
SP	12



Querying the data dictionary no.2 (1/2)

- ↳ For each attribute in the supplier-product table, show the attribute name, the number of distinct values and the number of tuples with a NULL value

```
SELECT Column_Name, Num_Distinct, Num_Nulls
FROM USER_TAB_COL_STATISTICS
WHERE Table_Name = 'SP'
ORDER BY Column_Name;
```



Querying the data dictionary no.2 (2/2)

```
SELECT Column_Name, Num_Distinct, Num_Nulls
FROM USER_TAB_COL_STATISTICS
WHERE Table_Name = 'SP'
ORDER BY Column_Name;
```

R

Column_Name	Num_Distinct	Num_Nulls
SId	4	0
PId	6	0
Qty	4	0



Managing tables

Data integrity



Integrity constraints

- ↳ Data in a database are correct if they satisfy a set of correctness rules
- rules are called *integrity constraints*
 - example: Qty >=0
- ↳ Data update operations define a new state for the database, which may not necessarily be correct



Integrity checks

- ⊃ Checking the correctness of a database state may be done
 - by *application procedures*, performing all required checks
 - through the definition of *integrity constraints* on the tables
 - through the definition of *triggers*



Application procedures

- ⊃ Each application includes all required correctness checks
- ⊃ Pros
 - efficient approach
- ⊃ Cons
 - checks may be circumvented by interacting directly with the DBMS
 - a coding error may have significant outcomes on the database
 - the knowledge of correctness rules is typically "hidden" inside applications



Table integrity constraints (1/2)

- ⊃ Integrity constraints are
 - defined in the CREATE or ALTER TABLE statements
 - stored in the system data dictionary
- ⊃ Each time data are updated, the DBMS automatically verifies that the constraints are satisfied



Table integrity constraints (2/2)

- ⊃ Pros
 - *declarative* definition of constraints, whose verification is delegated to the system
 - the data dictionary describes all of the constraints in the system
 - unique centralized check point
 - constraint verification may not be circumvented



Table integrity constraints (2/2)

- ⊃ Pros
 - *declarative* definition of constraints, whose verification is delegated to the system
 - the data dictionary describes all of the constraints in the system
 - unique centralized check point
 - constraint verification may not be circumvented
- ⊃ Cons
 - they may slow down application execution
 - it is not possible to define constraints of an arbitrary type
 - example: constraints on aggregated data



Triggers (1/2)

- ⊃ Triggers are procedures executed automatically when specific data updates are performed
 - defined through the CREATE TRIGGER command
 - stored in the system data dictionary
- ⊃ When a modification event occurs on data under the trigger's control, the procedure is automatically executed



Triggers (2/2)

- ⊃ Pros
 - they allow defining complex constraints
 - normally used in combination with constraint definition on the tables
 - unique centralized check point
 - constraint verification may not be circumvented
- ⊃ Cons
 - complex
 - they may slow down application execution



Fixing violations

- ⊃ If an application tries to execute an operation that causes a constraint violation, the system may
 - block the operation, causing an error in the application execution
 - execute a compensating action so that a new correct state is reached
 - example: when a supplier is deleted, also delete its supplies



Integrity constraints in SQL-92

- ⊃ The SQL-92 standard introduced the possibility to specify integrity constraints in a declarative way, delegating to the system the verification of their consistency
 - table constraints
 - restrictions on the data allowed in table columns
 - referential integrity constraints
 - manage references among different tables
 - based on the concept of foreign key



Table constraints (1/2)

- ⊃ They may be defined on one or more table columns
- ⊃ They are specified in the commands for creating
 - tables
 - domains
- ⊃ Types of constraints
 - primary key
 - admissibility of the NULL value
 - uniqueness
 - general tuple constraints



Table constraints (2/2)

- ⊃ They are verified after each SQL command operating on the table subject to the constraint
 - inserting new data
 - updating values in the columns subject to the constraint
- ⊃ If the constraint is violated, the SQL command causing the violating generates an execution error



Primary key

- ⊃ A primary key is a set of attributes that uniquely identifies rows in a tables
- ⊃ Only one primary key may be specified for a given table
- ⊃ Primary key definition
 - composed of a single attribute

```
AttributeName Domain PRIMARY KEY
```



Primary key: example no. 1

```
CREATE TABLE S (SId      CHAR(5) PRIMARY KEY,
                SName    CHAR(20),
                #Employees SMALLINT,
                City      CHAR(15));
```



Primary key

- ⊃ A primary key is a set of attributes that uniquely identifies rows in a tables
- ⊃ Only one primary key may be specified for a given table
- ⊃ Primary key definition
 - composed of one or more attributes

PRIMARY KEY (*AttributeList*)



Primary key: example no. 2

```
CREATE TABLE SP (SId  CHAR(5),
                 PId  CHAR(6),
                 Qty  INTEGER
                 PRIMARY KEY (SId, PId));
```



Admissibility of the NULL value

- ⊃ The NULL value indicates absence of information
- ⊃ When a value must always be specified for a given attribute

AttributeName Domain NOT NULL

- the NULL value is not allowed



NOT NULL: example

```
CREATE TABLE S (SId      CHAR(5),
                SName    CHAR(20) NOT NULL,
                #Employees SMALLINT,
                City      CHAR(15));
```



Uniqueness

- ⊃ An attribute or a set of attributes may not take the same value in different rows of the table
 - for a single attribute

AttributeName Domain UNIQUE

- for one or more attributes

UNIQUE (*AttributeList*)


- ⊃ Repetition of the NULL value in multiple rows is allowed (it is seen as a different value in each row)



Candidate key


- ⊃ A candidate key is a set of attributes that may serve as a primary key
 - it is unique
 - it might not allow the NULL value
- ⊃ The combination **UNIQUE NOT NULL** allows defining a candidate key that does not allow null values

AttributeName Domain UNIQUE NOT NULL



Uniqueness: example


```
CREATE TABLE P ( PId    CHAR(6),
                 PName  CHAR(20) NOT NULL UNIQUE,
                 Color  CHAR(6),
                 Size   SMALLINT,
                 Store  CHAR(15));
```



General tuple constraints


- ⊃ They allow expressing general conditions on each tuple
 - tuple or domain constraints

AttributeName Domain CHECK (Condition)
 - predicates allowed in the WHERE clause may be specified as a condition
- ⊃ The database is correct if the condition is true



General tuple constraints: example

```
CREATE TABLE S ( SId      CHAR(5) PRIMARY KEY,
                 SName    CHAR(20) NOT NULL,
                 #Employees SMALLINT
                 CHECK (#Employees>0),
                 City     CHAR(15));
```




Referential integrity constraints

- ⊃ They allow managing relationships among tables through the values of the attributes
- ⊃ Example

S			
SId	SName	#Employees	City

SP		
SId	PId	Qty


 - column SId in SP may assume values that are already present in column SId in the S table
 - SId in SP: referencing column (or foreign key)
 - SId in S: referenced column (usually the primary key)



Foreign key definition

- ⊃ A foreign key is defined in the CREATE TABLE statement of the referencing table


```
FOREIGN KEY (ReferencingAttributeList)
REFERENCES
TableName [(ReferencedAttributeList)]
```
- ⊃ If referencing attributes have the same name as the referenced attributes, they may be omitted



Foreign key definition: example

```
CREATE TABLE SP (SId CHAR(5),
                 PId CHAR(6),
                 Qty INTEGER,
                 PRIMARY KEY (SId, PId),
                 FOREIGN KEY (SId)
                   REFERENCES S(SId),
                 FOREIGN KEY (PId)
                   REFERENCES P(PId));
```



Constraint management: example no.1

- ⊃ SP (referencing table)
 - insert (new tuple) -> No
 - update (SId) -> No
 - delete (tuple) -> Ok
- ⊃ S (referenced table)
 - insert (new tuple) -> Ok
 - update (SId) -> cascaded update (cascade)
 - delete (tuple) -> cascaded update (cascade)
 - prevent action (no action)



Constraint management: example no.2 (1/3)

- ⊃ Employees (EId, EName, City, DId)
- ⊃ Departments (DId, DName, City)



Constraint management: example no.2 (2/3)

- ⊃ Employees (referencing table)



Constraint management: example no.2 (2/3)

- ⊃ Employees (referencing table)
 - insert (new tuple) -> No
 - update (DId) -> No
 - delete (tuple) -> Ok



Constraint management: example no.2 (3/3)

- ⊃ Departments (referenced table)
 - insert (new tuple) -> Ok
 - update (DId) -> cascaded update (cascade)
 - delete (tuple) -> cascaded update (cascade)
 - prevent action (no action)
 - set to unknown value (set null)
 - set to default value (set default)



Constraint management policies (1/3)

- ⇒ Integrity constraints are checked after each SQL command that may cause their violation
- ⇒ Insert or update operations on the referencing table that violate the constraints are not allowed



Constraint management policies (2/3)

- ⇒ Update or delete operations on the referenced table have the following outcome on the referencing table:
 - **CASCADE**: the update or delete operation is propagated
 - **SET NULL/DEFAULT**: a null or default value is set in the columns for the tuples whose values are no longer present in the referenced table
 - **NO ACTION**: the offending action is not executed



Constraint management policies (3/3)

- ⇒ In the CREATE TABLE statement of the referencing table

```
FOREIGN KEY (ReferencingAttributeList)
REFERENCES
  TableName [(ReferencedAttributeList)]
[ON UPDATE
  <CASCADE | SET DEFAULT | SET NULL |
  NO ACTION>]
[ON DELETE
  <CASCADE | SET DEFAULT | SET NULL |
  NO ACTION>]
```



Example database (1/4)

- ⇒ supplier and product DB
 - **table P**: it describes available products
 - primary key: PId
 - the product name may not assume null or duplicate values
 - size is always greater than zero
 - **table S**: it describes suppliers
 - primary key: SId
 - the supplier name may not assume null or duplicate values
 - the number of employees is always greater than zero



Example database (1/4)

- ⇒ supplier and product DB
 - **table SP**: it describes supplies, by relating products to the suppliers that provide them
 - primary key: (SId, PId)
 - quantity may not assume the null value and is greater than zero
 - referential integrity constraints



Example database (2/4)

```
CREATE TABLE P ( PId      CHAR(6) PRIMARY KEY,
                  PName    CHAR(20) NOT NULL UNIQUE,
                  Color     CHAR(6),
                  Size       SMALLINT
                    CHECK (Size > 0),
                  Store     CHAR(15));
```



Example database (3/4)

```
CREATE TABLE S (SId      CHAR(5) PRIMARY KEY,  
                SName    CHAR(20) NOT NULL,  
                #Employees SMALLINT  
                CHECK (#Employees>0),  
                City      CHAR(15));
```



Example database (4/4)

```
CREATE TABLE SP (SId      CHAR(5),  
                 PId      CHAR(6),  
                 Qty      INTEGER  
                 CHECK (Qty IS NOT NULL and Qty>0),  
                 PRIMARY KEY (SId, PId),  
                 FOREIGN KEY (SId)  
                   REFERENCES S(SId)  
                 ON DELETE NO ACTION  
                 ON UPDATE CASCADE,  
                 FOREIGN KEY (PId)  
                   REFERENCES P(PId)  
                 ON DELETE NO ACTION  
                 ON UPDATE CASCADE);
```

