

# DBS211

# Introduction to Database Systems

## WEEK- 4

## **Data Definition Language (DDL)**

# Learnt so far...

- ▶ File Systems
- ▶ DBMS
- ▶ Relational Model
- ▶ Keys
- ▶ Relationships
- ▶ Referential Integrity
- ▶ Table Types
- ▶ DML- SELECT, INSERT, UPDATE  
DELETE

# Agenda

Data Types

Constraints

How to create a table

How to modify a table afterwards

# DDL

## Data Definition Language

- ▶ Defining or changing the structural objects in the database
  - ▶ Create
  - ▶ Alter
  - ▶ Drop

# Design Specifications for a Table

- ▶ First write a Database Structure chart.
- ▶ We can also call this a **Data Dictionary** of your table.

Column	Type	Length	PK	FK reference	Req'd ?	Unique ?	Validation

- Column = column name
- Type = data type
- length = data length (only some types)
- PK = is Primary Key
- FK = is foreign key and to what
- Req'd = required (NOT NULL)
- Unique = unique values
- Validation = rules (could be range, value set, etc.)

# Basic Data Types

Programming Type	Database Types	Storage (bytes)	Ranges
Integers	int smallint tinyint mediumint bigint	4 2 1 3 8	-2147483648 to 2147483647 -32768 to 32767 -128 to 128 -8388608 to 8388607 $-2^{63}$ to $2^{63} - 1$
Decimal, Float, Real	decimal(precision, scale) numeric(precision, scale)		Precision – number of sig. dig. Scale – number of decimal places
String	char(length) or character(length) varchar(length) or varcharter(length) <i>more another day....</i>	1 byte per length	Number of characters
Dates and Times	date, datetime, time		

•p - precision, the total number of digits, s - scale, number of digits right of the decimal place, n - size, numeric value

# Constraints

Constraints are rules that are applied to the database at the table level

There are 7 base constraints

- ▶ **Primary Key**      Unique Identifier
- ▶ **Foreign Key**      Relationship with another PK
- ▶ **Required**              Nulls not allowed
- ▶ **Unique**              Unique Values
- ▶ **Validation (Check)**      Data ranges, limits or permitted values
- ▶ **Default Value**          If null, the use this value
- ▶ **Index** (unique or non-unique)      pre-sorting data for easy search

# Constraints

## Primary Keys

- ▶ Chosen identifying field
- ▶ Composite Keys
- ▶ Surrogate Keys

## Foreign key

- ▶ **FK Reference:**
  - ▶ A parent-child specification to the PK of another table
  - ▶ Enforcing “Referential Integrity”

## Required

- ▶ May not be left NULL, enforcing data integrity
- ▶ PKs, by default are required



# More Constraints

- ▶ **Unique?:**
  - ▶ Means only that the value can only appear once in this column
- ▶ **Validation:**
  - ▶ Specify the range of values or the specific values that are allowed for this column
- ▶ **Default Value:**
  - ▶ If a null is being inserted into a field, the default value will replace the null.
- ▶ **Index:**
  - ▶ A way the table pre-sorts the data to make searching easier
  - ▶ Some DBMS result in the duplication of data (storage space increased)

# Table Definition

- ▶ Format for defining the table  
PLAYERS:

TABLE: players					
FieldName	Type	Size	Required	PK/FK	Other
playerid	integer		✓	✓	
firstname	string	20	✓		
lastname	string	20	✓		
teamid	integer				

# Creating Tables in SQL

## CREATE TABLE

► Used to create a table

## Syntax

```
CREATE TABLE tablename(  
    field1 datatype fieldsize,  
    field2 datatype fieldsize,  
    :  
    fieldN datatype fieldsize,  
    UNIQUE (...),  
    CHECK (...),  
    PRIMARY KEY (fieldname(s)),  
    FOREIGN KEY (fieldname) REFERENCES tablename (PKfieldname))
```

# Creating Table PLAYERS

```
DROP TABLE players;
```

```
-- run this command first
```

```
if you still have the players table in your  
database from last week.
```

```
CREATE TABLE players
```

```
( playerId INT PRIMARY KEY,  
firstName VARCHAR(20) NOT NULL,  
lastName VARCHAR(20) NOT NULL,  
teamID INT );
```

# Default Value

- ▶ You can specify a default value for a column.
- ▶ A default value is the value to be inserted into a column if no other value is specified.
- ▶ If not explicitly specified, the default value of a column is NULL.

# Primary Key Constraint

- ▶ The primary key ensures the value of the PK column is specified for every row.
- ▶ A row can be accessed rapidly by using the primary
- ▶ The primary key guarantees the uniqueness of the PK column

# More tables (Continued)

TABLE: teams					
FieldName	Type	Size	Required	PK/FK	Other
teamid	integer		✓	✓	
teamname	string	15	✓		
maxPlayers	int		✓		default 0, range from 0 to 25
shirtcolor	string	20			
homeField	string	15			

# Teams table

```
CREATE TABLE teams
```

```
( teamID INT PRIMARY KEY,
```

```
teamName VARCHAR(15) NOT NULL,
```

```
maxPlayers INT DEFAULT 0,
```

```
shirtColor VARCHAR(10),
```

```
homeField VARCHAR(15),
```

```
CONSTRAINT
```

```
maxPlayer_chk CHECK (maxPlayers BETWEEN 0 AND 25) );
```



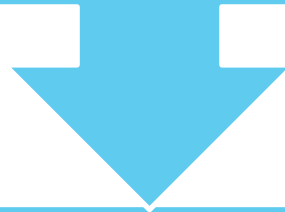
# Not Null Constraint

- ▶ NOT NULL constraint
  - ▶ guarantees that the user must specify a value.
- ▶ NOT NULL DEFAULT
  - ▶ The user must specify a value to be inserted if the value for that column is not provided.

```
CREATE TABLE table_name (  
    ...  
    column_name data_type NOT NULL  
    ...  
);
```

# Unique Constraint

Rows are now not allowed to be inserted or updated if the value of a unique column occurs more than once.



Columns with unique values:

Social Insurance Number	Driver's License Number	Ontario Health Card Number
----------------------------	----------------------------	-------------------------------

# Unique Constraint (Continued)

- ▶ Apply the unique constraints to columns

```
CREATE TABLE table_name (  
    ...  
    column_name data_type UNIQUE  
    ...  
);
```

OR

```
CREATE TABLE table_name (  
    ...  
    column_name data_type,  
    ...,  
    CONSTRAINT unique_constraint_name UNIQUE(column_name1,...)  
);
```

# Check Constraint



The check constraint enforces domain integrity. The database guarantees the inserted or updated values are valid with respect to a condition.



A column value is not allowed to be inserted or updated if it violated the check condition. Here are some common validations:



(Salary  $\geq$  2000 and Age  $\leq$  12000)



(age BETWEEN 18 and 40)



(Grade IN('A','B','C','D','F'))

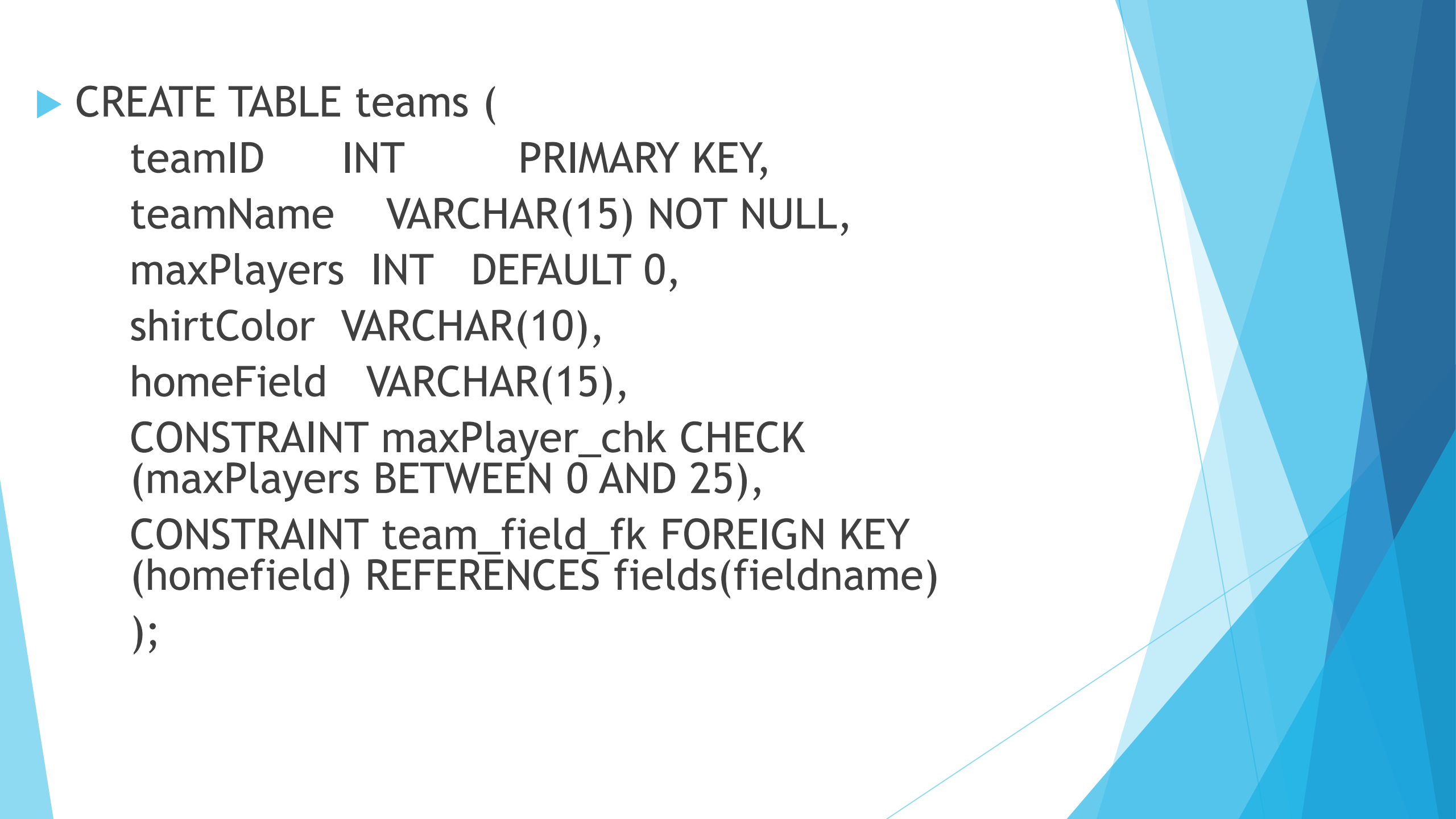
# Check Constraint (Continued)

- ▶ Creating a check constraint on a column

```
CREATE TABLE table_name (  
    ...  
    column_name data_type CHECK (expression),  
    ...  
);
```

OR

```
CREATE TABLE table_name (  
    ...,  
    column_name data_type,  
    ...,  
    CONSTRAINT check_constraint_name CHECK (expresssion)  
);
```

A decorative background on the right side of the slide consisting of overlapping, semi-transparent blue triangles and polygons in various shades of blue, creating a modern, abstract geometric pattern.

▶ CREATE TABLE teams (  
    teamID     INT       PRIMARY KEY,  
    teamName   VARCHAR(15) NOT NULL,  
    maxPlayers INT   DEFAULT 0,  
    shirtColor VARCHAR(10),  
    homeField  VARCHAR(15),  
    CONSTRAINT maxPlayer\_chk CHECK  
    (maxPlayers BETWEEN 0 AND 25),  
    CONSTRAINT team\_field\_fk FOREIGN KEY  
    (homefield) REFERENCES fields(fieldname)  
);

# Foreign Key Constraint

- ▶ The Foreign Key enforces relational integrity between the two tables.
- ▶ The Foreign Key is used to create 1:M or 1:1 relationships between two tables. It is used to get information from another table using the primary key of that table.
- ▶ The table with the foreign key column is the child table.
- ▶ The table that is referred by the foreign key column in the CHILD table is the PARENT table.

# Changing a Table Definition

- ▶ Data Definition Language
  - ▶ Is used to create/update table definition.



# Alter a Table

- ▶ **ALTER TABLE**

- ▶ Used to update a database definition

- ▶ **Syntax**

**ALTER TABLE** `table_name` `action`;

- ▶ **ALTER TABLE** is used to modify a table specification, such as:

- ▶ Add column/columns
  - ▶ Modify a column/attribute
  - ▶ Drop a column
  - ▶ Add a constraint
  - ▶ Drop a constraint
  - ▶ Rename table

# Add a Column

## ► To add a column

```
ALTER TABLE table_name  
ADD column_name type constraint;
```

## ► To add Multiple Columns

```
ALTER TABLE table_name  
ADD (  
    column_name type constraint,  
    column_name type constraint,  
    ...  
);
```

# EXAMPLES

▶ ALTER TABLE players  
ADD date\_of\_birth DATE;

ALTER TABLE players  
ADD CONSTRAINT player\_teams\_fk FOREIGN KEY  
(teamID) REFERENCES teams(teamID);

# Modify a Column

- ▶ Modify a Column

```
ALTER TABLE table_name  
MODIFY column_name type constraint;
```

- ▶ Modify multiple columns

```
ALTER TABLE table_name  
MODIFY ( column_1 type constraint,  
          column_2 type constraint,  
          ... );
```

# Rename a Column/Table

- ▶ To rename a column

```
ALTER TABLE table_name  
RENAME COLUMN old_name TO new_name;
```

Rename a table:

```
ALTER TABLE table_name  
RENAME TO new_table_name;
```

# Remove a Column

- ▶ To remove a column

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

- ▶ To remove multiple column

```
ALTER TABLE table_name  
DROP (  
    column_name_1,  
    column_name_2  
);
```

# Add Constraints

- ▶ To add a primary key constraint to an existing table

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name  
PRIMARY KEY (column1, column2, ... column_n);
```

- ▶ Add a unique constraint

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name  
UNIQUE (column1, column2, ... column_n);
```

# Add Constraints (Continued)

- ▶ Add a check constraint

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name  
CHECK (column_name condition);
```

- ▶ Add a foreign key

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name  
FOREIGN KEY (column1, column2, ... column_n)  
REFERENCES parent_table (column1, column2, ... column_n);
```



# Remove a Constraint

- ▶ To remove a constraint

```
ALTER TABLE table_name
```

```
DROP CONSTRAINT constraint_name;
```

# Remove a Table

- ▶ To remove a database object

```
DROP <Object_type> <object_name>;
```

- ▶ To drop a table

```
DROP table table_name;
```

# Import Data

- ▶ It is possible to insert data into a table from another table.
- ▶ The data from a table can be copied to another table as a backup.
- ▶ Caution: constraints are not carried from original table to new table; they would need to be added individually to the new table via Alter Table statement

# How to copy data into a table

- ▶ **Syntax**

```
INSERT INTO target_table (column1, column2, ...)
SELECT column1,
       column2,
       ...
FROM source_table
[WHERE conditions];
```

- ▶ To copy rows into a table INSERT and SELECT statements, the value of every required (NOT NULL) column must be provided.
- ▶ If a column value is not required, the column do not have to be included in the insert statement.

# How to copy data into a table

- ▶ Three step process:
  - ▶ Create a table with the same definition as an existing table
  - ▶ Modify the new empty table definition to add appropriate constraints
  - ▶ Copy the data into the new table from the old table.

# Create Table and Copy Data

- ▶ To create a table by copying all columns from another table with data:

```
CREATE TABLE new_table AS  
(SELECT * FROM old_table);
```

- ▶ The above statement creates a new table exactly the same as the old one and copies all data from the old table to the new one.

# Create Table and Copy Data

- ▶ You can create a table from another one just by including some of the columns in your select statement:

```
CREATE TABLE new_table AS  
(SELECT column_1,  
        column_2,  
        ...  
        column_n  
FROM old_table);
```

# Create Table and Copy Data

- ▶ You can create a table by copying columns from multiple tables:

```
CREATE TABLE new_table AS  
(SELECT column_1, column2, ...  
column_n  
FROM old_table_1, old_table_2, ...  
old_table_n  
[WHERE conditions]);
```



# ▶ SUMMARY

- ▶ DDL commands
  - ▶ CREATE
  - ▶ ALTER
    - ▶ ADDING COLUMNS
    - ▶ ADDING CONSTRAINTS
  - ▶ DROP
  - ▶ COPY OR INSERT DATA FROM EXISTING TABLES