Database Management Systems
Course 236363

Tutorial 2:
SQL

Outline
• Preliminaries
  • Relational model definitions
  • Example
• Basic SQL Queries
  • SELECT
  • WHERE
• DB Manipulation
  • DML – DB Data manipulation
  • DDL – DB Definition manipulation

Relational Model - Recap

Field, Property, Attribute
Table, Relation

<table>
<thead>
<tr>
<th>Field, Property, Attribute</th>
<th>Table, Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>purple</td>
</tr>
<tr>
<td>age</td>
<td>pink</td>
</tr>
<tr>
<td>animal</td>
<td>black</td>
</tr>
</tbody>
</table>

Database example

• Library Database

- Customers
  - Cust_Id: Customer ID (unique)
  - Cust_Name: Customer Name
  - Faculty: Faculty Name

- Books
  - Book_Id
  - Book_Name
  - Year
  - Max_Time
  - Pages

- Borrowed
  - Cust_Id
  - Book_Id
  - Order_Date

- Orders
  - Cust_Id
  - Book_Id
  - From_Date
  - To_Date

Database example

• Customers(Cust_Id, Cust_Name, Faculty)
  - Cust_Id: Customer ID (unique)
  - Cust_Name: Customer Name
  - Faculty: Faculty Name

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Cust_Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Moshe Cohen</td>
<td>CS</td>
</tr>
<tr>
<td>23456</td>
<td>Avi Barak</td>
<td>EE</td>
</tr>
<tr>
<td>34567</td>
<td>Avi Barak</td>
<td>MED</td>
</tr>
<tr>
<td>46789</td>
<td>Lior Edri</td>
<td>EE</td>
</tr>
<tr>
<td>58798</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
<tr>
<td>67890</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
</tbody>
</table>
Database example

- Books(\textbf{Book Id}, Book Name, Year, Max_Time, Faculty, Pages)
  - \textbf{Book Id}: Unique book id
  - Book Name: Book title
  - Year: Year of print
  - Max_Time: Maximum borrowing time in days
  - Faculty: Faculty name
  - Pages: Page number

Database example

<table>
<thead>
<tr>
<th>Book Id</th>
<th>Book Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>1</td>
<td>390</td>
<td>EE</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>

Database example

- Ordered(\textbf{Cust Id}, Book Id, Order_Date)
  - \textbf{Cust Id}: Customer ID
  - \textbf{Book Id}: Book ID
  - Order_Date: Date of book order

Database example

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Book_Id</th>
<th>Order_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>1111</td>
<td>14-Oct-2002</td>
</tr>
<tr>
<td>45678</td>
<td>1112</td>
<td>24-Oct-2002</td>
</tr>
<tr>
<td>12345</td>
<td>1113</td>
<td>30-Oct-2002</td>
</tr>
<tr>
<td>45678</td>
<td>2222</td>
<td>12-Oct-2002</td>
</tr>
</tbody>
</table>

Database example

- Borrowed(\textbf{Book Id}, \textbf{Cust Id}, From_Date, To_Date)
  - \textbf{Book Id}: Book ID
  - \textbf{Cust Id}: Customer ID
  - From_Date: Borrowing date
  - To_Date: Return date

Database example

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Cust_Id</th>
<th>From_Date</th>
<th>To_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>5555</td>
<td>56789</td>
<td>13-Oct-2002</td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Preliminaries
  - Relational model definitions
  - Example
- Basic SQL Queries
  - SELECT
  - WHERE
- DB Manipulation
  - DML – DB Data manipulation
  - DDL – DB Definition manipulation

SQL Queries: General Format

//Choose the data
SELECT [ALL | DISTINCT] {table.* | expr [alias], expr [alias], …}
//Data sources
FROM table [alias], table [alias], ...
//Condition on the data
[WHERE condition]
//Aggregations
[GROUP BY expr, expr, … [HAVING condition]]
//Groups arithmetics
[[INTERSECT | EXCEPT | UNION | UNION ALL ] SELECT …]
//Sorting
[ORDER BY expr [ASC | DESC ], expr [ASC | DESC],…];

- The expressions inside [] are optional

SQL Queries: SELECT

Specific columns selection (by name):

SELECT column1, column2, column3,…
FROM table;

All columns selection:

SELECT *
FROM table;

SQL Queries: SELECT

• Example: select the name of every book, and its page number

SELECT Book_Name, Pages
FROM Books;

<table>
<thead>
<tr>
<th>Book_Name</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Systems</td>
<td>348</td>
</tr>
<tr>
<td>Database Systems</td>
<td>348</td>
</tr>
<tr>
<td>Database Systems</td>
<td>424</td>
</tr>
<tr>
<td>Database And Knowledge</td>
<td>390</td>
</tr>
<tr>
<td>Database And Knowledge</td>
<td>390</td>
</tr>
<tr>
<td>Electronic Circuits</td>
<td>180</td>
</tr>
<tr>
<td>Genes</td>
<td>7 580</td>
</tr>
<tr>
<td>Anatomy</td>
<td>450</td>
</tr>
</tbody>
</table>

SQL Queries: Where

• Filtering tuples by Boolean condition.

SELECT column1, column2, column3,…
FROM table
WHERE Boolean.Condition;

• Example: the names of the books which printed after 1990

SELECT Book_Name
FROM Books
WHERE Year > 1990;

<table>
<thead>
<tr>
<th>Book_Name</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Systems</td>
<td>1998</td>
</tr>
<tr>
<td>Database Systems</td>
<td>1998</td>
</tr>
<tr>
<td>Database Systems</td>
<td>2001</td>
</tr>
<tr>
<td>Database And Knowledge</td>
<td>1998</td>
</tr>
<tr>
<td>Database And Knowledge</td>
<td>1998</td>
</tr>
<tr>
<td>Electronic Circuits</td>
<td>1998</td>
</tr>
<tr>
<td>Electronic Circuits</td>
<td>1998</td>
</tr>
<tr>
<td>Genes</td>
<td>1998</td>
</tr>
<tr>
<td>Anatomy</td>
<td>1998</td>
</tr>
</tbody>
</table>
SQL Queries: Where

SELECT Book_Name FROM Books
WHERE Year > 1990

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>1</td>
<td>390</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>390</td>
<td>EE</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>

Outline

• Preliminaries
  • Relational model definitions
  • Example
• Basic SQL Queries
  • SELECT
  • WHERE
• DB Manipulation
  • DML – DB Data manipulation
  • DDL – DB Definition manipulation

UPDATE DATA

UPDATE tablename
SET column-assignment-list
WHERE conditional-expression ;

UPDATE DATA

• Goal: update values in existing tuples
• First option: set fix value in specific columns in every tuple.

• Example: move every book to the general library and limit their borrowing time to 7 days

• UPDATE Books
  SET Max_Time = 7, Faculty = 'GEN';

Books

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1988</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>1</td>
<td>390</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>390</td>
<td>EE</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>
**UPDATE DATA: Example**

**UPDATE Books**

```sql
SET Max_Time = 7, Faculty = 'GEN';
```

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>GEN</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>GEN</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>GEN</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>GEN</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>GEN</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>7</td>
<td>180</td>
<td>GEN</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>GEN</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>GEN</td>
</tr>
</tbody>
</table>

**UPDATE DATA**

- **Second option:** update only part of the tuples by adding a `WHERE` condition.

- **Example:** limit the maximum borrowing time to a week, for all books in CS faculty.

**UPDATE Books**

```sql
SET Max_Time = 7;
WHERE Faculty = 'CS';
```

**UPDATE DATA: Example**

**Books**

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>CS</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>

**UPDATE DATA**

- **Third option:** set an expression

- **Example:** increase borrowing time limit by 1, to all books.

**UPDATE Books**

```sql
SET Max_Time = Max_Time + 1;
```

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>300</td>
<td>CS</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>
UPDATE DATA: Example

**UPDATE Books**

```
UPDATE Books SET Max_Time = Max_Time + 1;
```

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>8</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>15</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>8</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>2</td>
<td>390</td>
<td>CS</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>22</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>8</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>8</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>

**INSERT DATA**

```
INSERT INTO tablename [( column-list )]
VALUES ( constant-list );
```

**INSERT DATA: Example**

**Goal:** adding new tuples

**First option:** add one given tuple.

**Example:** add a client to Customers table

```
INSERT INTO Customers
VALUES (78901, 'Roy Peled', 'EE');
```

**INSERT DATA: Example**

```
INSERT INTO Customers VALUES (78901, 'Roy Peled', 'EE');
```

**INSERT DATA**

- You can initiate only specific fields
- Example:
  ```
  INSERT INTO Customers(Cust_Id,Cust_Name) VALUES (78901, 'Roy Peled');
  ```

- Important!
  - Uninitialized fields (Faculty) will be set as NULL
  - If a column does not allow NULL values, the INSERT action is illegal and will fail.
**INSERT DATA: Example**

**Customers**

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Cust_Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Moshe Cohen</td>
<td>CS</td>
</tr>
<tr>
<td>23456</td>
<td>Avi Barak</td>
<td>EE</td>
</tr>
<tr>
<td>34567</td>
<td>Avi Barak</td>
<td>MED</td>
</tr>
<tr>
<td>45678</td>
<td>Lior Edri</td>
<td>EE</td>
</tr>
<tr>
<td>56789</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
<tr>
<td>67890</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
</tbody>
</table>

**INSERT INTO Customers(Cust_Id, Cust_Name) VALUES (78901, 'Roy Peled');**

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Cust_Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Moshe Cohen</td>
<td>CS</td>
</tr>
<tr>
<td>23456</td>
<td>Avi Barak</td>
<td>EE</td>
</tr>
<tr>
<td>34567</td>
<td>Avi Barak</td>
<td>MED</td>
</tr>
<tr>
<td>45678</td>
<td>Lior Edri</td>
<td>EE</td>
</tr>
<tr>
<td>56789</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
<tr>
<td>67890</td>
<td>Moshe Cohen</td>
<td>EE</td>
</tr>
<tr>
<td>78901</td>
<td>Roy Peled</td>
<td></td>
</tr>
</tbody>
</table>

**INSERT DATA**

- You can add into a table the result given by a query.

- Example: Insert the client ID into the table Readers, for all client who ordered a book.

  ```sql
  INSERT INTO Readers( Id)
  (SELECT Cust_Id
  FROM Ordered);
  ```

  This is legal only if the table Readers exists.

**INSERT DATA: Example**

**Ordered:**

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Book_Name</th>
<th>Order_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Database Systems</td>
<td>14-Oct-2002</td>
</tr>
<tr>
<td>56789</td>
<td>Anatomy</td>
<td>24-Oct-2002</td>
</tr>
<tr>
<td>12345</td>
<td>Database And Knowledge</td>
<td>30-Oct-2002</td>
</tr>
<tr>
<td>56789</td>
<td>Electronic Circuits</td>
<td>12-Oct-2002</td>
</tr>
</tbody>
</table>

**INSERT INTO Readers (SELECT * FROM Readers)**

**Readers:**

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Cust_Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Moshe Cohen</td>
<td>CS</td>
</tr>
<tr>
<td>45678</td>
<td>Lior Edri</td>
<td>EE</td>
</tr>
</tbody>
</table>

**INSERT DATA: Example**

- What will happen if we'll call INSERT into a table with the data from the same table?
- INSERT is executed over a temporary copy of the table
- Example:

  ```sql
  INSERT INTO Readers
  (SELECT * FROM Readers);
  ```

  - This will not cause an infinite loop but will only duplicate the table.
DELETE DATA

\[
\text{DELETE FROM } \text{tablename} \\
\text{WHERE } \text{conditional-expression}
\]

DELETE DATA

- DELETE deletes tuples from the table.
- You need to define which tuple you want to delete.
- Example: delete all orders
  \[
  \text{DELETE FROM } \text{Ordered};
  \]
- The table itself is not deleted, but is contain 0 tuples.

DELETE DATA

- Adding WHERE allows us to choose which tuples will be deleted.
- Example: Delete every order of the customer with ID 12345:
  \[
  \text{DELETE FROM } \text{Ordered} \\
  \text{WHERE Cust_Id = 12345;}
  \]

DELETE DATA: Example

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Book_Name</th>
<th>Order_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Database Systems</td>
<td>14-Oct-2002</td>
</tr>
<tr>
<td>45678</td>
<td>Anatomy</td>
<td>24-Oct-2002</td>
</tr>
<tr>
<td>12345</td>
<td>Database And Knowledge</td>
<td>30-Oct-2002</td>
</tr>
<tr>
<td>45678</td>
<td>Electronic Circuits</td>
<td>12-Oct-2002</td>
</tr>
</tbody>
</table>

Ordered:

<table>
<thead>
<tr>
<th>Cust_Id</th>
<th>Book_Name</th>
<th>Order_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>45678</td>
<td>Anatomy</td>
<td>24-Oct-2002</td>
</tr>
<tr>
<td>45678</td>
<td>Electronic Circuits</td>
<td>12-Oct-2002</td>
</tr>
</tbody>
</table>

Outline

- Preliminaries
  - Relational model definitions
  - Example
- Basic SQL Queries
  - SELECT
  - WHERE
- DB Manipulation
  - DML – DB Data manipulation
- DDL – DB Definition manipulation

Create Table

- Create an empty table:
  \[
  \text{CREATE TABLE table (column_name column_type [(length)]) [NOT NULL]};
  \]
- Create a table from a query results:
  \[
  \text{CREATE TABLE table AS SELECT } ...;
  \]
Create Table: Example

- **CREATE TABLE** EXAMPLE
  (AB VARCHAR(15),
  CD INTEGER,
  EF INTEGER NOT NULL,
  GH DECIMAL(5),
  IJ DECIMAL(7,2),
  KL TEXT default('DB'));

PostgreSQL Types

- **VARCHAR (n)** – String with length <= n
- **TEXT** – String with unlimited length
- **BOOLEAN** – true/false
- **DATE** – date
- **INTEGER** – integer
- **DECIMAL [p,s]** – number with given decimal precision.

Additional PostgreSQL types
http://www.postgresql.org/docs/9.6/static/datatype.html#DATATYPE-TABLE

Primary Key

- **CREATE TABLE** EXAMPLE
  (AB VARCHAR(15),
  CD INTEGER,
  EF DECIMAL(5),
  GH DECIMAL(7,2),
  PRIMARY KEY (AB,CD));

A primary key cannot be NULL

UNIQUE

- **CREATE TABLE** EXAMPLE
  (AB VARCHAR(15),
  CD INTEGER,
  EF DECIMAL(5),
  GH DECIMAL(7,2),
  PRIMARY KEY (AB,CD),
  UNIQUE (AB,CD));

Foreign Key

- Recall Ordered table:
  Ordered(Cust_Id, Book_Id, Order_Date).

- How to impose that every Cust_id is valid?

  CREATE TABLE Ordered
  (Cust_Id INTEGER,
  Book_Id INTEGER,
  Order_Date TIMESTAMP,
  FOREIGN KEY (Cust_Id)
  REFERENCES Customers(Cust_Id));

Foreign Key

- You can also write
  CREATE TABLE Ordered
  (Cust_Id INTEGER,
  Book_Id INTEGER,
  Order_Date TIMESTAMP,
  FOREIGN KEY (Cust_Id) REFERENCES Customers);

- CREATE TABLE Ordered
  (Cust_Id INTEGER REFERENCES Customers,
  Book_Id INTEGER,
  Order_Date TIMESTAMP);
Foreign Key

- Let’s assume that there is an order for customer with id 1002, and now we delete this client. What will happen?

- You can define
  
  ```sql
  CREATE TABLE Ordered
  (Cust_Id INTEGER,
   Book_Id INTEGER,
   Order_Date TIMESTAMP,
   FOREIGN KEY (Cust_Id)
   REFERENCES Customers(Cust_Id)
   ON DELETE CASCADE);
  ```

Check

- **CREATE TABLE EXAMPLE**
  
  ```sql
  (A TEXT,
   B INTEGER,
   C INTEGER,
   CHECK (B > 0),
   CHECK (B < C));
  ```

- **CREATE TABLE EXAMPLE**
  
  ```sql
  (A TEXT,
   B INTEGER CHECK (B > 0),
   C INTEGER,
   CHECK (B < C));
  ```

CREATE TABLE: Example

Example: create a new table named CSBooks which contains all the books from 'CS' faculty.

```sql
CREATE TABLE CSBooks AS
SELECT Book_Id, Book_Name
FROM Books
WHERE Faculty = 'CS';
```

CREATE TABLE: Example

**Books**

<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>424</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>1</td>
<td>390</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1998</td>
<td>7</td>
<td>390</td>
<td>EE</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1985</td>
<td>7</td>
<td>180</td>
<td>EE</td>
</tr>
<tr>
<td>4444</td>
<td>Genes 7</td>
<td>1985</td>
<td>7</td>
<td>580</td>
<td>MED</td>
</tr>
<tr>
<td>5555</td>
<td>Anatomy</td>
<td>1988</td>
<td>7</td>
<td>450</td>
<td>MED</td>
</tr>
</tbody>
</table>

Create Table

- New table schema:
  - Columns types are determined by the CREATE query.
  - You can rename the columns

```sql
CREATE TABLE CSBooks(Id, Name) AS
SELECT Book_Id, Book_Name
FROM Books
WHERE Faculty = 'CS';
```
DROP TABLE

You can delete the table by calling DROP
Example: delete CSBooks table:

```
DROP TABLE CSBooks;
```

VIEWS

- Motivation: aliasing a query result as a table without actually create the table.
- The query that defines the view is saved in the DB and calculated every time it is being called.
- View can be a “window” where you can see a part of the database, on your needs.
- Usage: simplify queries code, calculations, privacy & security.

CREATE VIEW view_name AS SELECT...;

- Example:
  CREATE VIEW CSBooksView AS
  SELECT Book_Id, Book_Name, Max_Time FROM Books WHERE Faculty = 'CS';
- CSBooksView hides every columns in Books, except Book_Id, Book_Name, Max_Time, and holds only the data relevant for CS.
- You cannot use ORDER BY when creating a view.

```
<table>
<thead>
<tr>
<th>Book_Id</th>
<th>Book_Name</th>
<th>Year</th>
<th>Max_Time</th>
<th>Pages</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Database Systems</td>
<td>1998</td>
<td>7</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1112</td>
<td>Database Systems</td>
<td>1998</td>
<td>14</td>
<td>348</td>
<td>CS</td>
</tr>
<tr>
<td>1113</td>
<td>Database Systems</td>
<td>2001</td>
<td>7</td>
<td>404</td>
<td>CS</td>
</tr>
<tr>
<td>2222</td>
<td>Database And Knowledge</td>
<td>1988</td>
<td>1</td>
<td>390</td>
<td>CS</td>
</tr>
<tr>
<td>2223</td>
<td>Database And Knowledge</td>
<td>1988</td>
<td>7</td>
<td>390</td>
<td>EE</td>
</tr>
<tr>
<td>3333</td>
<td>Electronic Circuits</td>
<td>1998</td>
<td>21</td>
<td>580</td>
<td>EE</td>
</tr>
</tbody>
</table>
```

DROP VIEW

- You can delete a view by DROP VIEW
- Example:
  DROP VIEW CSBooksView;
- The underlined table remains intact.

```
SELECT Book_Name
FROM CSBooksView
WHERE Max_Time = 7;
```
Question

- What is difference?

CREATE TABLE CSBooks AS
SELECT Book_Id, Book_Name
FROM Books
WHERE Faculty = 'CS';

CREATE VIEW CSBooksView AS
SELECT Book_Id, Book_Name
FROM Books
WHERE Faculty = 'CS';