Database Queries

Motivating questions

- How is data stored in a database?
- How do we create databases and tables?
- How do we insert data into a database?
- How do we get the data that we need from a single table in a relational database?

SQL and SQL Data Types

Structured Query Language (SQL)

- Data definition: Operations to build tables and views (virtual tables)
- Data manipulation: INSERT, DELETE, UPDATE or retrieve (SELECT) data
- Data integrity: Referential integrity and transactions
  - Enforces primary and foreign keys
- Access control: Security for multiple types of users
- Data sharing: Database accessed by concurrent users

Structured Query Language (SQL)

- SQL is not a complete language like Java, Visual Basic or C++
  - SQL is sub-language of about 30 statement types
- Embedded in another language or tool for database access
- SQL has several inconsistencies; NULLs can be problematic
- Portable across operating systems and somewhat portable among vendors
- Declarative language, not a procedural language

Variations among SQL implementations

- Error codes
- Data types supported (dates/times, currency, string/text variations)
- Whether case matters (upper, lower case)
- System tables (the structure of the database itself)
- Programming interface (no vendor follows the standard)
- Report and query generation tools
- Implementer-defined variations within the standard
- Database initialization, opening and connection

Core MySQL Data Types – Numeric

Numeric Type	Description
INT	A standard integer
BIGINT	A large integer
DECIMAL	A fixed-point number
FLOAT	A single-precision, floating-point number
DOUBLE	A double-precision, floating-point number
BIT	A bit field

Core MySQL Data Types – Strings (Text)

String Type	Description
CHAR	A fixed-length, non-binary string (character)
VARCHAR	A variable-length, non-binary string
NCHAR	Same as above + Unicode Support
NVARCHAR	Same as above + Unicode Support
BINARY	A fixed-length, binary string
VARBINARY	A variable-length, binary string
TINYBLOB	A very small BLOB (binary large object)
BLOB	A small BLOB
TEXT	A small, non-binary string

Core MySQL Data Types – Dates/Times

Date / Time Type	Description
DATE	A date value in 'CCYY-MM-DD' format
TIME	A time value in 'hh:mm:ss' format
DATETIME	Date/Time in 'CCYY-MM-DD hh:mm:ss’ format
TIMESTAMP	Timestamp in 'CCYY-MM-DD hh:mm:ss’ format
YEAR	A year value in CCYY or YY format

Key points from lesson

- SQL (Structured Query Language) contains the commands we use to create, manage and query relational databases
- There are different vendor implementations of SQL, but most have a common set of data types and commands
- We have discussed commonly used data types and will cover the most common queries

Creating Databases and Tables

Example data model

- It is typical to use the same name for columns with the same meaning across different tables
- We give them different names here so that we don't have to say tablename.columnname:
  - Here: OfficeNbr and RepOffice
  - Preferred: Offices.OfficeNbr and SalesReps.OfficeNbr

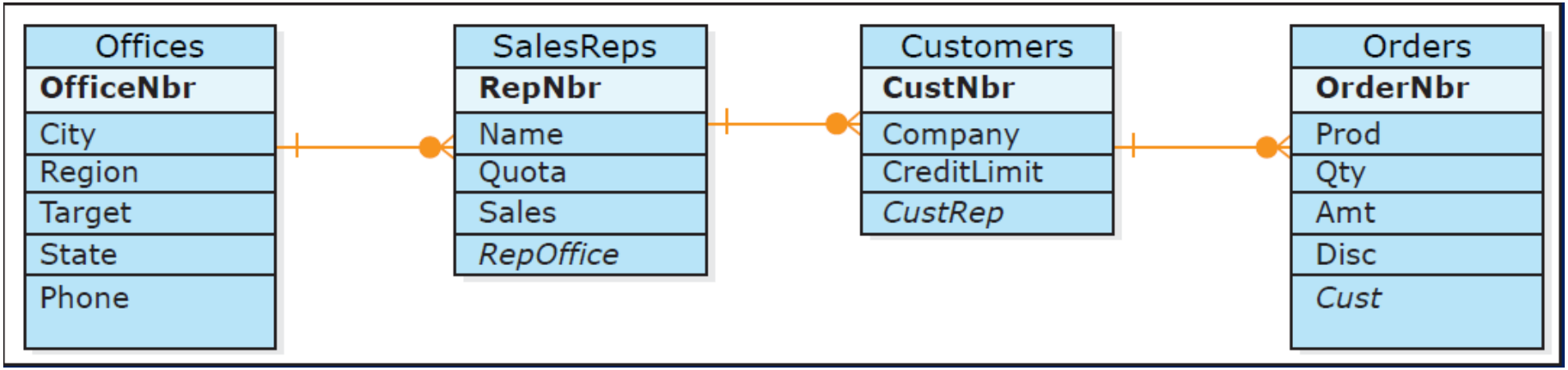


Image by MIT OpenCourseWare.

Create a database for SC4x

- Recall, database is a collection of tables
- First we must CREATE a database:

```
DROP DATABASE IF EXISTS SC4x;
CREATE DATABASE SC4x;
USE SC4x;
```

Create Offices table

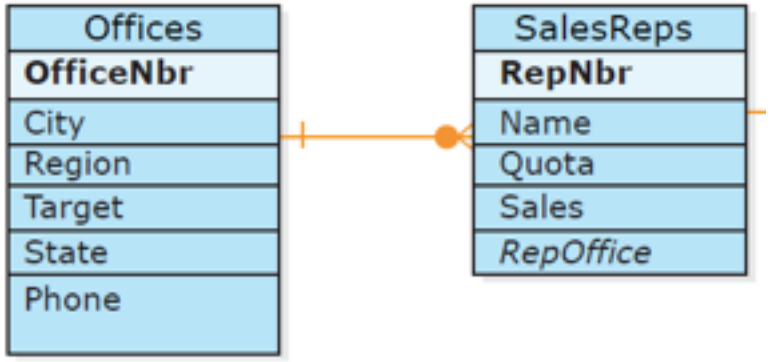
```
CREATE TABLE Offices (
OfficeNbr NCHAR(2) NOT NULL PRIMARY KEY,
City NVARCHAR(20) NOT NULL,
State NCHAR(2) NOT NULL,
Region NCHAR(5) NOT NULL,
Target DECIMAL(10,2) NOT NULL,
Sales DECIMAL(10,2) NOT NULL,
Phone NVARCHAR(15) NOT NULL);
```

Offices

OfficeNbr
City
State
Region
Target
Sales
Phone

Create SalesReps table

```
CREATE TABLE SalesReps (
RepNbr NCHAR(3) NOT NULL PRIMARY KEY,
Name NVARCHAR(20) NOT NULL,
RepOffice NCHAR(2) NOT NULL,
Quota DECIMAL(10,2),  # Allow NULLs
Sales DECIMAL(10,2) NOT NULL,
FOREIGN KEY (RepOffice)
REFERENCES Offices(OfficeNbr));
```



SalesReps

RepNbr
Name
RepOffice
Quota
Sales

Create Customers table

```
CREATE TABLE Customers (
CustNbr NCHAR(3) NOT NULL PRIMARY KEY,
Company NVARCHAR(30) NOT NULL,
CustRep NCHAR(3) NOT NULL,
CreditLimit DECIMAL(10,2) NOT NULL,
FOREIGN KEY (CustRep)
REFERENCES SalesReps(RepNbr));
```

Customers

CustNbr
Company
CustRep
CreditLimit

Create Orders table

```
CREATE TABLE Orders (
OrderNbr INT NOT NULL PRIMARY KEY
AUTO_INCREMENT,
Cust NCHAR(3) NOT NULL,
Prod NVARCHAR(20) NOT NULL,
Qty INT NOT NULL,
Amt DECIMAL(10,2) NOT NULL,
Disc DECIMAL(3,1) NOT NULL,
FOREIGN KEY (Cust)
REFERENCES Customers(CustNbr));
```

Orders

OrderNbr
Cust
Prod
Qty
Amt
Disc

Create Employees table

```
CREATE TABLE Employees(
EmpNbr NCHAR(5) NOT NULL PRIMARY KEY,
Name NVARCHAR(20) NOT NULL,
Title NVARCHAR(20) NOT NULL,
Mgr NCHAR(5));
```

Employees

EmpNbr
Name
Title
Mgr

Create Parts table

```
CREATE TABLE Parts (
PartID NCHAR(4) NOT NULL PRIMARY KEY,
Vendor NCHAR(4) NOT NULL);
```

Parts

PartID
Vendor

Key points from lesson

- Databases are made using the CREATE DATABASE command
- The USE command tells the system which database to use
- New tables are declared using the CREATE TABLE command, we can also set the name and data type of each attribute
- When creating new tables, we can specify primary keys and foreign key relationships and whether or not NULL or empty values are allowed

Inserting Data into a Database

Inserting data into a new database

- The data model acts as a guide to load data into a new database:
  - It may build well, which usually means you found the real business rules
  - It may build with some errors, which usually means you have the real business rules, but the data is sloppy
  - It may build with many errors, which usually means that you were told the business rules people wish to have or think they have, not the ones they actually use

- It's often useful to get some sample data and browse it while building the data model

Insert offices into the Offices table

```
INSERT INTO Offices
VALUES('1', 'Denver', 'CO', 'West',
3000000, 130000, '970.586.3341');

INSERT INTO Offices
VALUES('2', 'New York', 'NY', 'East',
200000, 300000, '212.942.5574');

INSERT INTO Offices
VALUES('57', 'Dallas', 'TX', 'West', 0, 0,
'214.781.5342');
```

Offices

- OfficeNbr
- City
- State
- Region
- Target
- Sales
- Phone

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342

Insert sales representatives into the SalesReps table

```
INSERT INTO SalesReps
VALUES('53', 'Bill Smith', '1', 100000, 0);

INSERT INTO SalesReps
VALUES('89', 'Jen Jones', '2', 50000, 130000);
```

RepNbr	Name	RepOffice	Quota	Sales
53	Bill Smith	1	100000.00	0.00
89	Jen Jones	2	50000.00	130000.00

SalesReps

RepNbr
Name
RepOffice
Quota
Sales

Insert customers into the Customers table

```
INSERT INTO Customers
VALUES('211', 'Connor Co', '89', 50000);

INSERT INTO Customers
VALUES('522', 'Amaratunga Enterprises', '89',
40000);

INSERT INTO Customers
VALUES('890', 'Feni Fabricators', '53', 1000000);
```

CustNbr	Company	CustRep	CreditLimit
211	Connor Co	89	50000.00
522	Amaratunga Enterprises	89	40000.00
890	Feni Fabricators	53	1000000.00

Customers

CustNbr
Company
CustRep
CreditLimit

Insert orders into the Orders table

```
INSERT INTO Orders (Cust, Prod, Qty, Amt, Disc)
VALUES('211', 'Bulldozer', 7, 31000, 0.2);

INSERT INTO Orders (Cust, Prod, Qty, Amt, Disc)
VALUES('522', 'Riveter', 2, 4000, 0.3);

INSERT INTO Orders (Cust, Prod, Qty, Amt, Disc)
VALUES('522', 'Crane', 1, 500000, 0.4);
```

Orders

OrderNbr
Cust
Prod
Qty
Amt
Disc

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000.00	0.20
2	522	Riveter	2	4000.00	0.30
3	522	Crane	1	500000.00	0.40

Final tables

Orders

OrderNbr	Cust	Prod	Qty	Amt	Disc
1	211	Bulldozer	7	31000.00	0.20
2	522	Riveter	2	4000.00	0.30
3	522	Crane	1	500000.00	0.40

Customers

CustNbr	Company	CustRep	CreditLimit
211	Connor Co	89	50000.00
522	Amaratunga Enterprises	89	40000.00
890	Feni Fabricators	53	1000000.00

SalesReps

RepNbr	Name	RepOffice	Quota	Sales
53	Bill Smith	1	100000.00	0.00
89	Jen Jones	2	50000.00	130000.00

Offices

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342

Key points from lesson

- We have now created and inserted initial data into our SC4x office database

- These new tables and records can be viewed using database management tools

SQL SELECT queries

SELECT

- SELECT statements are constructed of clauses to get records from one or more tables or views.
- Clauses must be in order, only SELECT and FROM are required:

```
SELECT attributes/columns
INTO new table
FROM table or view
WHERE specific records or a join is created
GROUP BY grouping conditions (attributes)
HAVING group-property (specific records)
ORDER BY ordering criterion ASC | DESC
```

SELECT attributes FROM SalesReps

- List the sales reps and their current sales and quotas:

```
SELECT Name, Sales, Quota
FROM SalesReps;
```

Name	Sales	Quota
Bill Smith	0.00	100000.00
Jen Jones	130000.00	50000.00

SalesReps

RepNbr
Name
RepOffice
Quota
Sales

SELECT attributes FROM SalesReps

- Find the amount each rep is over/under quota:

```
SELECT Name, Sales, Quota, (Sales - Quota)
FROM SalesReps;
```

Name	Sales	Quota	(Sales – Quota)
Bill Smith	0.00	100000.00	-100000.00
Jen Jones	130000.00	50000.00	80000.00

SalesReps

RepNbr
Name
RepOffice
Quota
Sales

SELECT slackers FROM SalesReps

- Find the slackers:

```
SELECT Name, Sales, Quota, (Sales - Quota)
FROM SalesReps
WHERE Sales < Quota;
```

Name	Sales	Quota	(Sales – Quota)
Bill Smith	0.00	100000.00	-100000.00

SalesReps

RepNbr
Name
RepOffice
Quota
Sales

SELECT all records FROM Offices

- Select all attributes (fields) from Offices:

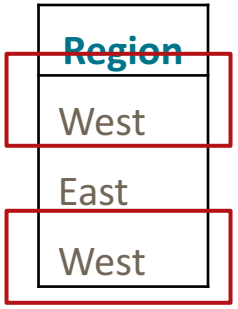
```
SELECT *
FROM Offices;
```

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342

SELECT regions without duplicates

- Select all Region records:

```
SELECT Region
FROM Offices;
```



- Select unique regions from Offices:

```
SELECT DISTINCT Region
FROM Offices;
```

Region
West
East

SELECT unique values and counts

- Find the unique set of office city/state combinations:

```
SELECT DISTINCT City, State
FROM Offices;
```

City	State
Denver	CO
New York	NY
Dallas	TX

- Find the number of sales for a customer:

```
SELECT COUNT(*)
FROM Orders
WHERE Cust = '211';
```

COUNT(*)
1

SELECT sums and averages FROM Orders

- Find the sum of all sales:

```
SELECT SUM(Amt)
FROM Orders;
```

SUM(Amt)
535000.00

- Find the average of all sales:

```
SELECT AVG(Amt)
FROM Orders;
```

AVG(Amt)
178333.33

- Find the average sale for customer 211:

```
SELECT AVG(Amt)
FROM Orders;
WHERE Cust = '211';
```

AVG(Amt)
31000.00

Wildcards in SQL

- The official SQL standard only has 2 wildcards:

% any string of zero or more characters

_ any single character

```
SELECT OfficeNbr, Phone
FROM Offices
WHERE Phone LIKE '21%';
```

OfficeNbr	Phone
2	212.942.5574
57	214.781.5342

```
SELECT OfficeNbr, Phone
FROM Offices
WHERE Phone NOT LIKE '21%';
```

OfficeNbr	Phone
1	970.586.3341

Wildcards in SQL

- Most database implementations offer additional regular expressions wildcards

- MySQL has:

`[list]` match any single character in list, e.g. `[a-f]`
`[^list]` match any single character not in list, e.g. `[^h-m]`

```
SELECT OfficeNbr, Phone
FROM Offices
WHERE Phone REGEXP '21[1-2]';
```

OfficeNbr	Phone
2	212.942.5574

Additional expressions here: http://dev.mysql.com/doc/refman/5.7/en/regexp.html

Key points from lesson

- SELECT returns a set of attributes in a query
- The WHERE clause is used to identify the set of records that meets a specific set of conditions
- SELECT and WHERE are used together to pick a subset of attributes and records from a table
- The DISTINCT keyword, statistical functions can be applied to the attributes selected in a query
- Regular expressions can be used to find records which match complex string patterns

Editing a table

INSERT INTO Offices

- Add an office:

```
INSERT INTO Offices (OfficeNbr, City, State,
Region, Target, Sales, Phone)
VALUES ('55', 'Dallas', 'TX', 'West', 200000,
0, '214.333.2222');
```

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342
55	Dallas	TX	West	200000.00	0.00	214.333.2222

UPDATE Offices

- Change a sales target:

```
UPDATE Offices
SET Target = 300000
WHERE OfficeNbr = '55';
```

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342
55	Dallas	TX	West	300000.00	0.00	214.333.2222

DELETE Offices

- Remove an office from the database:

```
DELETE FROM Offices
WHERE OfficeNbr = '55';
```

- Watch out for referential integrity when deleting records

OfficeNbr	City	State	Region	Target	Sales	Phone
1	Denver	CO	West	3000000.00	130000.00	970.586.3341
2	New York	NY	East	200000.00	300000.00	212.942.5574
57	Dallas	TX	West	0.00	0.00	214.781.5342

Key points from lesson

- INSERT is used to add a new record to a table that contains specific values for a set of attributes in that table
- The UPDATE keyword is used to modify a specific value or set of values for a set of records in a table
- DELETE is used to remove records from a table that meet a specific condition