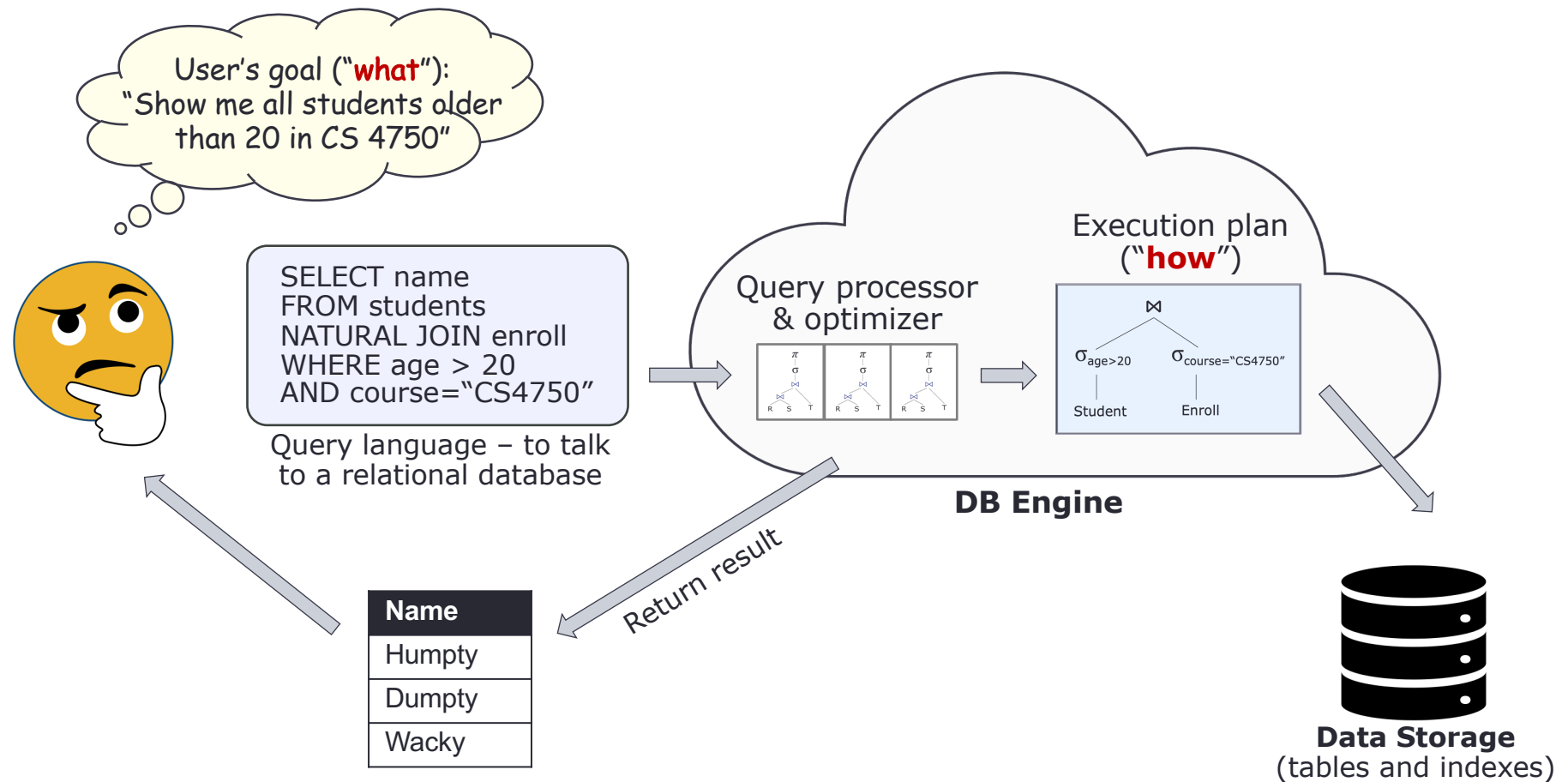


SQL – Basics

CS 4750 Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.3]

SQL: Structured Query Language



SQL: "declarative" – You describe *what you want*
DB engine describes *how to efficiently retrieve it*

SQL Command Overview

DDL (Data Definition Language) – Defining structures

Purpose: create & modify DB objects (schema)

CREATE
(create tables, views, indexes, databases)

ALTER
(change existing object)

DROP
(remove object entirely)

TRUNCATE
(empty table, keep structure)

DML (Data Manipulation Language) – Working with instance

Purpose: insert, update, delete, retrieve data

SELECT
(retrieve data)

INSERT
(add new rows)

UPDATE
(modify existing rows)

DELETE
(remove rows)

DCL (Data Control Language) – Handling permission & security

Purpose: control who can do what on which part of DB

GRANT
(give privileges)

REVOKE
(remove privileges)

TCL (Transaction Control Language) – Managing transactions

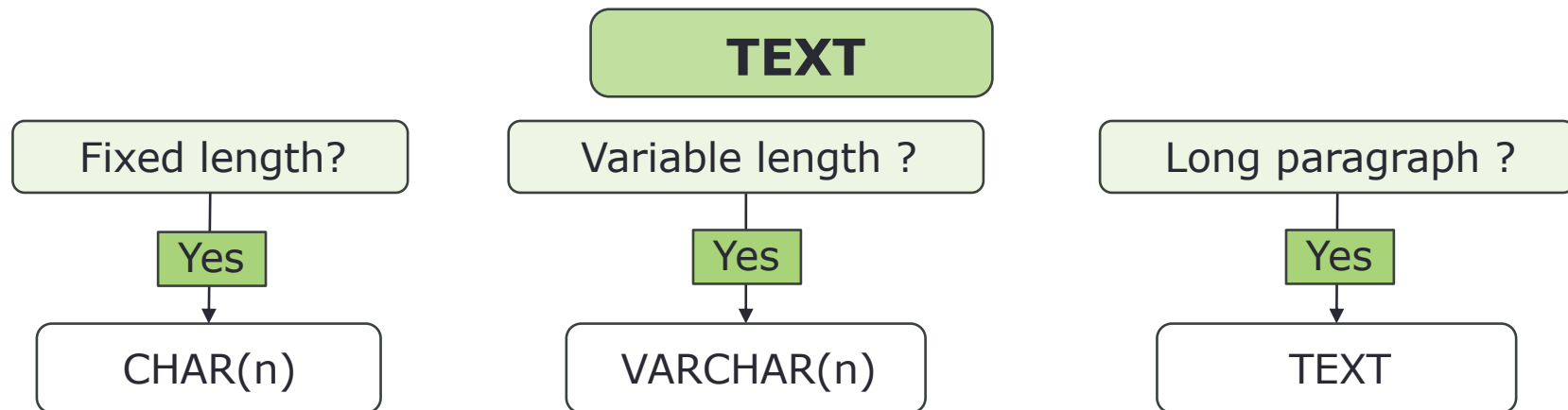
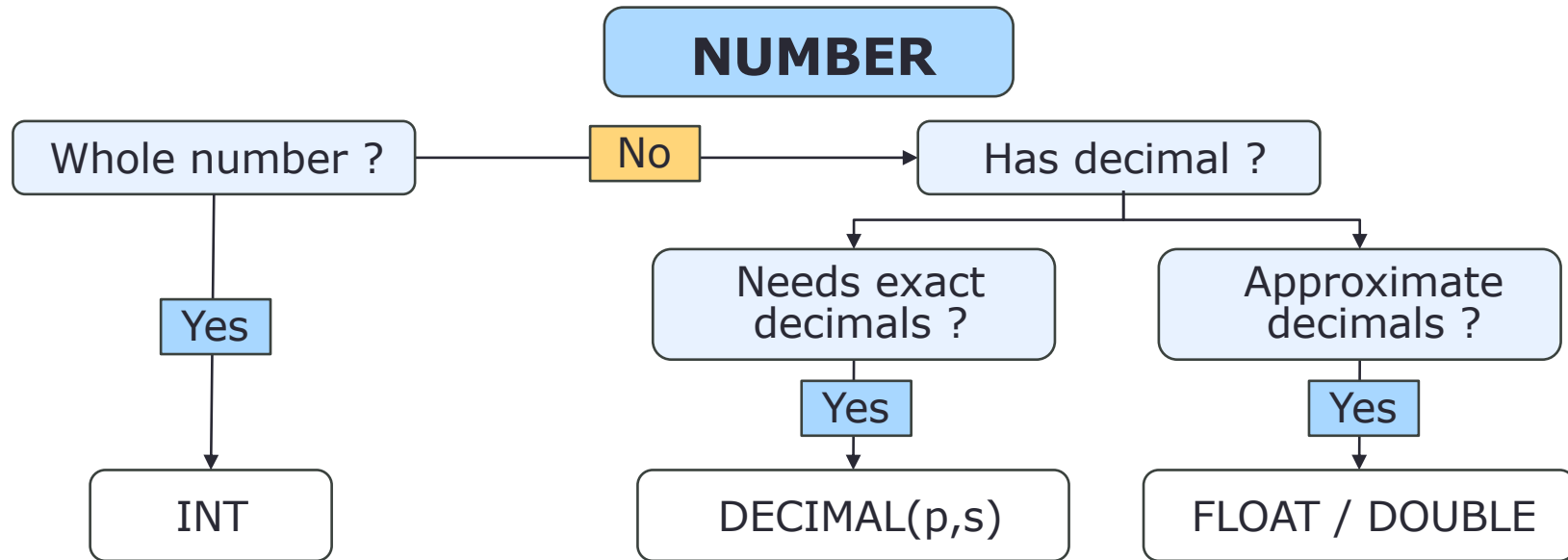
Purpose: control logical units of work

BEGIN TRANSACTION
(set a transaction's boundary)

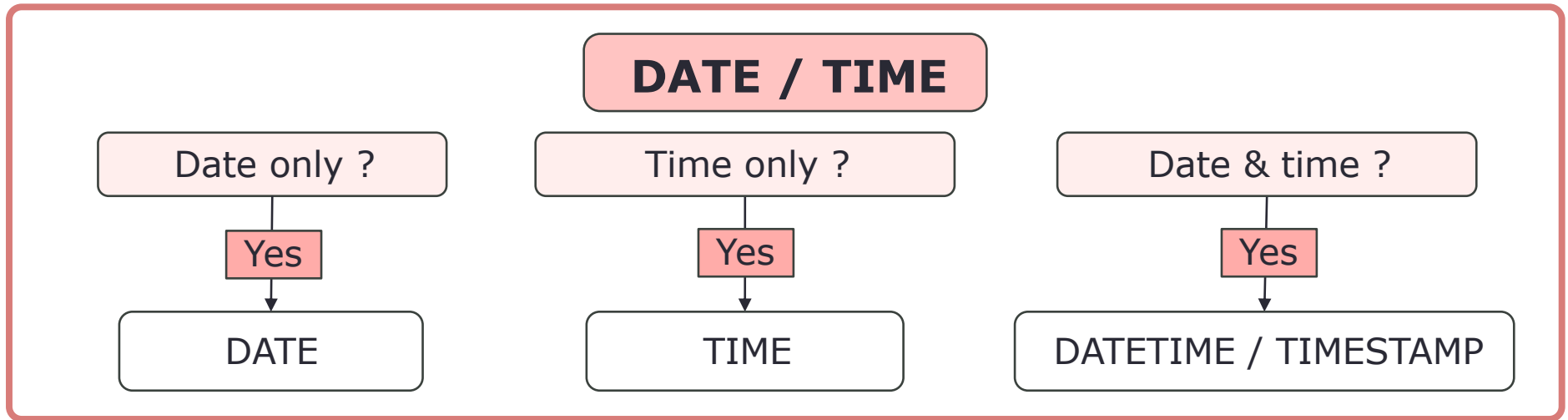
COMMIT
(make changes permanent)

ROLLBACK
(undo changes since last commit)

Standard Data Types



Data Types in SQL (2)



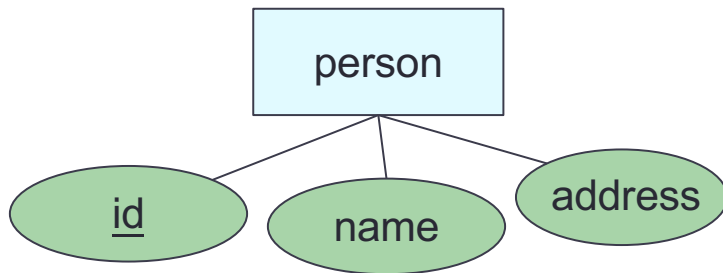
Other data types

- **BOOLEAN**
- **BLOB** (Binary Large Object) – varying-length binary string; for file, image, video, large object

Examples:

Converting ER diagrams into SQL tables (DDL – effects schema)

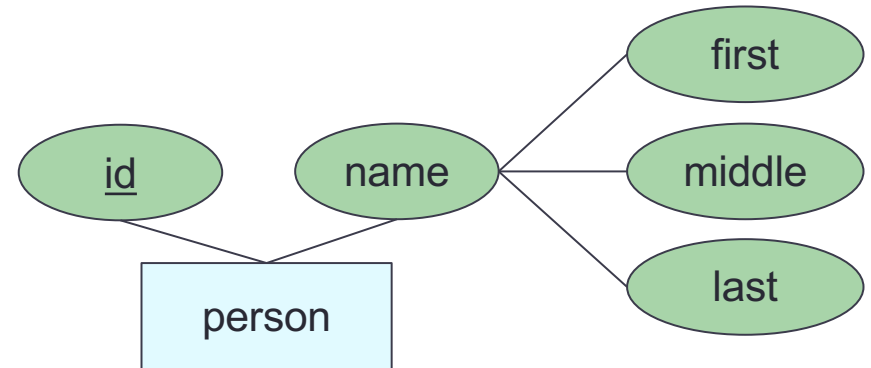
CREATE TABLE (Entity Set)



```
person(id, name, address)
```



```
CREATE TABLE person (  
    id INT,  
    name VARCHAR(40),  
    address VARCHAR(255),  
    PRIMARY KEY (id) );
```

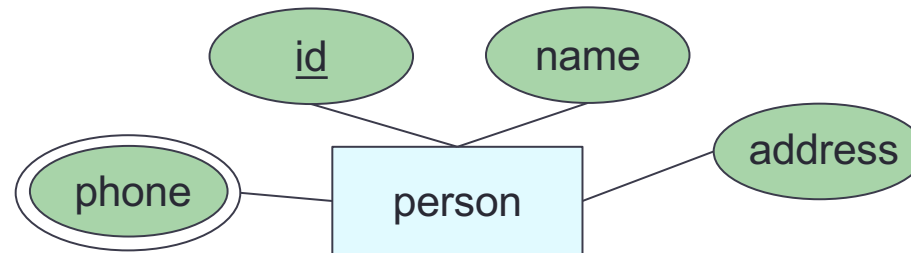


```
person(id, first_name,  
        middle_name, last_name)
```



```
CREATE TABLE person (  
    id INT,  
    first_name VARCHAR(20),  
    middle_name VARCHAR(3),  
    last_name VARCHAR(20),  
    PRIMARY KEY (id) );
```

CREATE TABLE (Entity Set)



```
person(id, name, address)
person_phone(id, phone)
```



```
CREATE TABLE person ( id INT,
                        name VARCHAR(40),
                        address VARCHAR(255),
                        PRIMARY KEY (id) );
```

```
CREATE TABLE person_phone ( id INT REFERENCES person(id),
                              phone VARCHAR(10),
                              PRIMARY KEY (id, phone) );
```

CREATE TABLE (One-to-One)

Option 1: create tables for entity sets and a relationship set



```
CREATE TABLE product(pid VARCHAR(10), ...,  
    PRIMARY KEY (pid) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

```
CREATE TABLE makes(  
    [ cname VARCHAR(100) UNIQUE REFERENCES company(cname),  
      pid VARCHAR(100) UNIQUE REFERENCES product(pid),  
      ... );
```

```
PRIMARY KEY (cname, pid)
```

Do we need all these tables?

CREATE TABLE (One-to-One)

Option 2: create tables for entity sets, store a primary key of one entity in another entity, no table for relationship set



```
CREATE TABLE product(pid VARCHAR(10), ...,  
    PRIMARY KEY (pid) );  
  
CREATE TABLE company(cname VARCHAR(100), ...,  
    pid VARCHAR(10),  
    PRIMARY KEY (cname),  
    FOREIGN KEY (pid) REFERENCES product(pid) );
```

Make use of the one-to-one fact. Fewer tables. Empty possible

CREATE TABLE (Many-to-Many)



```
CREATE TABLE product(pid VARCHAR(10), ...,  
    PRIMARY KEY (pid) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

```
CREATE TABLE makes(  
    cname VARCHAR(100) REFERENCES company(cname),  
    pid VARCHAR(100) REFERENCES product(pid), ...,  
    PRIMARY KEY (cname, pid) );
```

CREATE TABLE (One-to-Many / Many-to-One)

Option 1: create tables for entity sets and a relationship set



```
CREATE TABLE product(pid VARCHAR(10), ...,  
    PRIMARY KEY (pid) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

```
CREATE TABLE makes(  
    [ cname VARCHAR(100) REFERENCES company(cname),  
    pid VARCHAR(100) REFERENCES product(pid), ...,  
    PRIMARY KEY (pid) );
```

```
FOREIGN KEY (cname) REFERENCES company(cname)
```

Do we need all these tables?

CREATE TABLE (One-to-Many / Many-to-One)

Option 2: create tables for entity sets, store a primary key of the “one” side in the “many” side, no table for relationship set



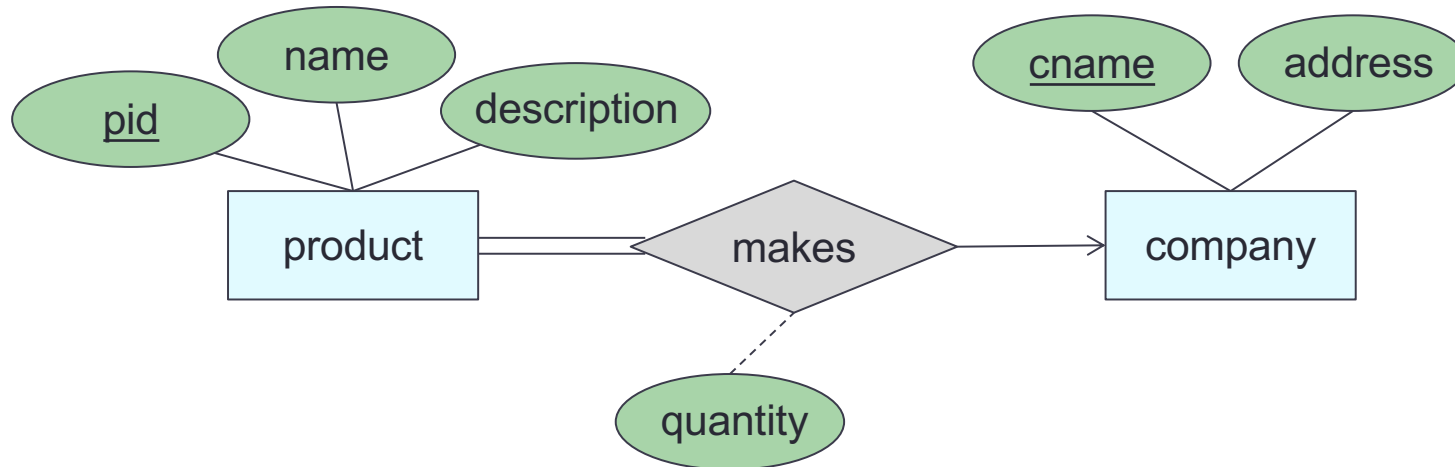
```
CREATE TABLE product(pid VARCHAR(10), ...,  
    cname VARCHAR(100),  
    PRIMARY KEY (pid),  
    FOREIGN KEY (cname) REFERENCES company(cname) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

Fewer tables. Empty possible

CREATE TABLE (Total Participation)

Option 1: use a FOREIGN KEY constraint

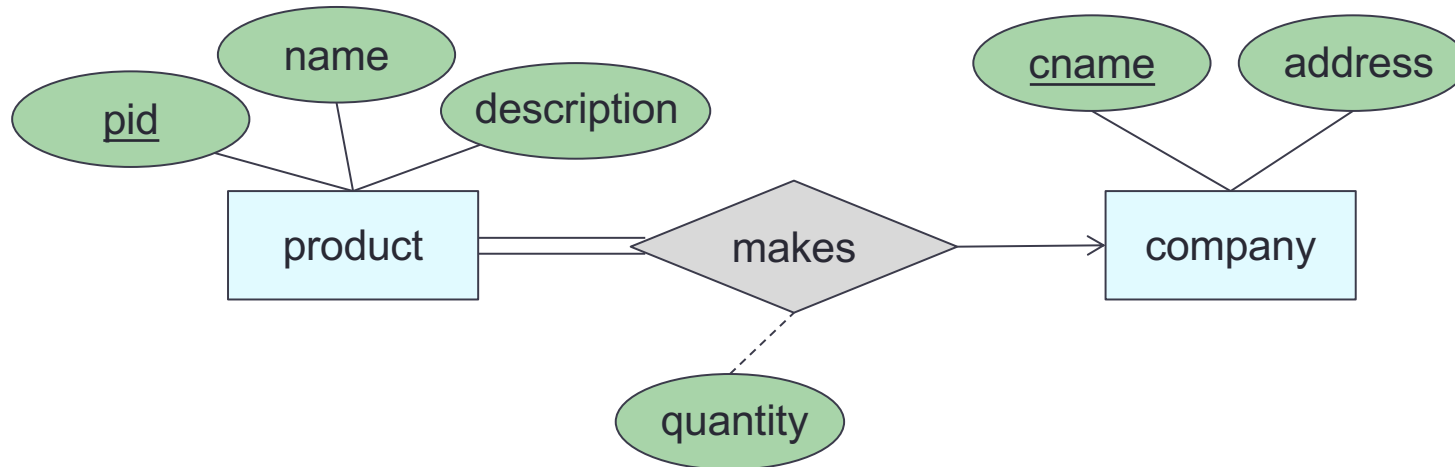


```
CREATE TABLE product(pid VARCHAR(10), ...,  
    quantity INT,  
    cname VARCHAR(100)  
    PRIMARY KEY (pid),  
    FOREIGN KEY (cname) REFERENCES company(cname) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

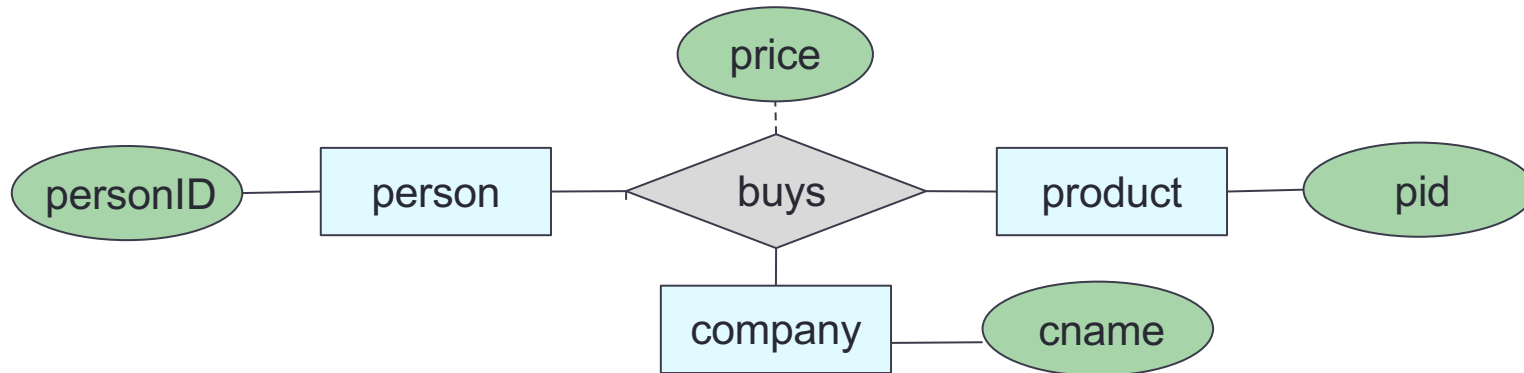
CREATE TABLE (Total Participation)

Option 2: use a NOT NULL constraint



```
CREATE TABLE product(pid VARCHAR(10), ...,  
    quantity INT,  
    PRIMARY KEY (pid),  
    cname VARCHAR(100) NOT NULL REFERENCES company(cname));  
  
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname));
```

CREATE TABLE (Multi-Way Relations)



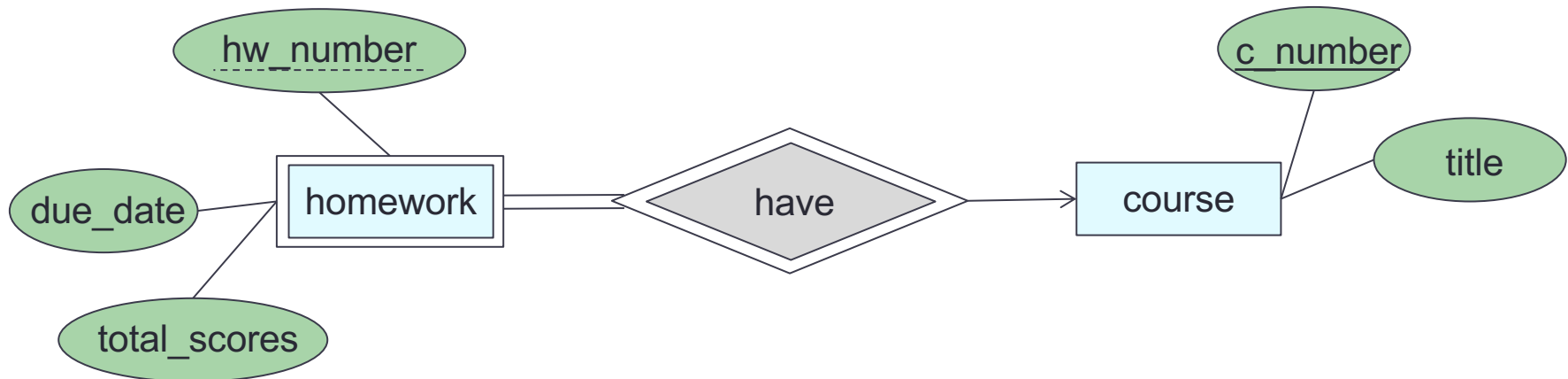
```
CREATE TABLE product(pid VARCHAR(10), ...,  
    PRIMARY KEY (pid) );
```

```
CREATE TABLE company(cname VARCHAR(100), ...,  
    PRIMARY KEY (cname) );
```

```
CREATE TABLE person VARCHAR(10), ...,  
    PRIMARY KEY (personid) );
```

```
CREATE TABLE buys(  
    cname VARCHAR(100) REFERENCES company(cname),  
    pid VARCHAR(100) REFERENCES product(pid),  
    personID VARCHAR(10) REFERENCE person(personID), ...,  
    price FLOAT,  
    PRIMARY KEY (cname, pid, personID,) );
```

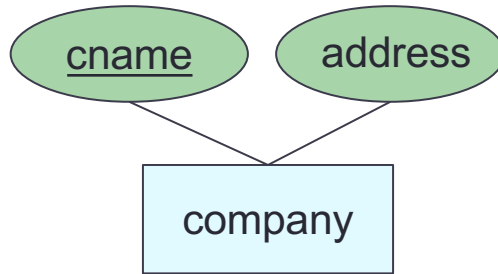
CREATE TABLE (Weak Entity Set)



```
CREATE TABLE course(c_number VARCHAR(10), ...,  
    PRIMARY KEY (c_number) );
```

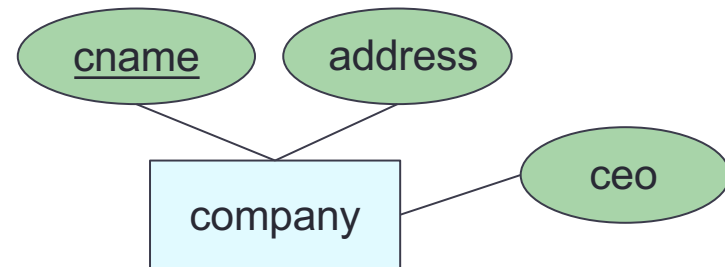
```
CREATE TABLE homework(hw_number VARCHAR(10), ...,  
    c_number VARCHAR(10) REFERENCES course(c_number),  
    PRIMARY KEY (c_number, hw_number) );
```

ALTER TABLE ... ADD/DROP

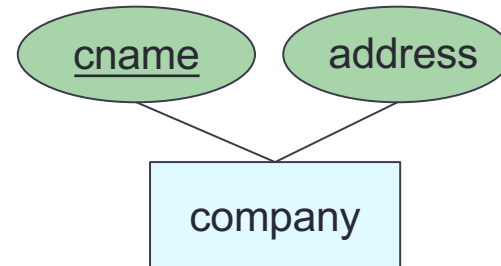


```
CREATE TABLE company (  
    cname VARCHAR(40),  
    address VARCHAR(255),  
    PRIMARY KEY (cname));
```

```
ALTER TABLE company  
ADD ceo VARCHAR(40);
```



```
ALTER TABLE company  
DROP ceo;
```



DROP vs. TRUNCATE vs. DELETE

DROP (DDL)

```
DROP TABLE  
table_name;
```

Destroy everything:

- Removes table definition & data
- Deletes indexes, constraints, triggers
- Frees storage space completely
- Cannot reverse; no Rollback

TRUNCATE (DDL)

```
TRUNCATE TABLE  
table_name;
```

Empty quickly:

- Removes all rows; keep schema
- Faster than DELETE
- Resets AUTO_INCREMENT
- Cannot use WHERE clause
- Cannot reverse; implicit COMMIT

DELETE (DML)

```
DELETE FROM  
table_name  
[WHERE ...];
```

Remove specific rows:

- Removes rows
- Slower; logs each deletion
- Can use WHERE clause for filtering
- Does not reset AUTO_INCREMENT
- Rollback possible (within transaction)

These cannot be undone !!

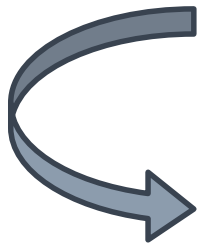
Examples:
Converting ER diagrams
into SQL tables
(DML – effects instances)

INSERT INTO ... VALUE

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
INSERT INTO Student_lecture  
VALUES ("9999", "85 Engineer's Way",  
        "Database Systems", "Humpty");
```



Student_lecture

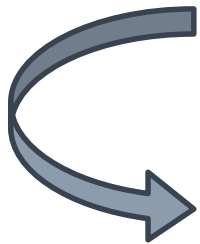
S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie
9999	85 Engineer's Way	Database Systems	Humpty

UPDATE ... SET ... WHERE

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
UPDATE Student_lecture  
SET Teaching_assistant = "Humpty"  
WHERE S_id = "5678";
```



Student_lecture

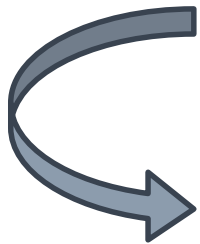
S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Humpty

DELETE FROM ... WHERE

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
DELETE FROM Student_lecture  
WHERE Teaching_assistant = "Humpty";
```



Student_lecture

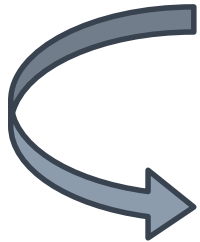
S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

SELECT * FROM ...

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT * FROM Student_lecture;
```



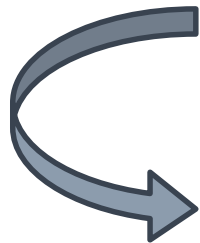
S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

SELECT Specific Attribute

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id, Course FROM Student_lecture;
```



S_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Web Programming Lang.
5678	Software Analysis

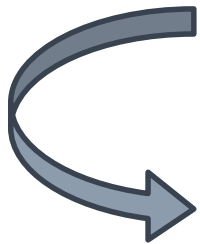
"Projection"

SELECT with WHERE Clause (1)

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT * FROM Student_lecture  
WHERE Teaching_assistant = "Minnie" ;
```



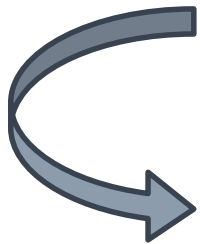
S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

SELECT with WHERE Clause (2)

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id, Course FROM Student_lecture  
WHERE Teaching_assistant = "Minnie" ;
```



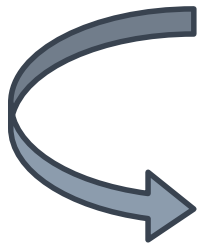
S_id	Course
1234	Database Systems
5678	Software Analysis

SELECT with WHERE Clause (3)

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id, Course FROM Student_lecture
WHERE Teaching_assistant = "Minnie" OR
      Teaching_assistant = "Mickey" ;
```



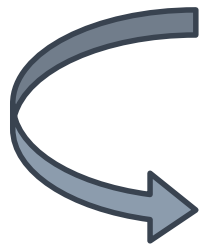
S_id	Course
1234	Database Systems
1234	Web Programming Lang.
5678	Software Analysis

SELECT DISTINCT

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT DISTINCT Teaching_assistant  
FROM Student_lecture ;
```



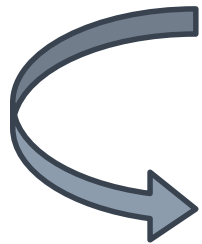
Teaching_assistant
Minnie
Humpty
Dumpty
Mickey

SELECT with Pattern Matching

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT Teaching_assistant FROM Student_lecture  
WHERE Course LIKE 'Data%';
```



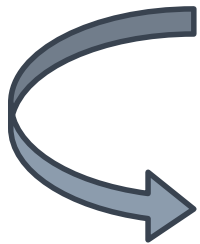
Teaching_assistant
Minnie
Humpty

ORDER BY (Ascending)

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id, Course FROM Student_lecture  
WHERE Teaching_assistant <> "Dumpty"  
ORDER BY S_id;
```



S_id	Course
1234	Database Systems
1234	Web Programming Lang.
2345	Database Systems
5678	Software Analysis

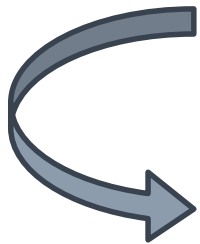
Ascending, by default
Or include ASC

ORDER BY (Descending)

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id, Course FROM Student_lecture  
WHERE Teaching_assistant <> "Dumpty"  
ORDER BY S_id DESC;
```



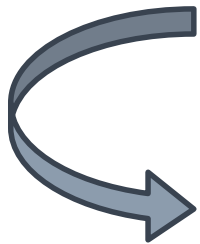
S_id	Course
5678	Software Analysis
2345	Database Systems
1234	Web Programming Lang.
1234	Database Systems

SELECT with Aliases

Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id as ID, Course as "Course Name"  
FROM Student_lecture  
WHERE Teaching_assistant <> "Dumpty" ;
```



ID	Course Name
1234	Database Systems
1234	Web Programming Lang.
2345	Database Systems
5678	Software Analysis

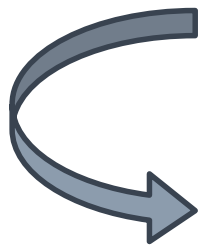
UNION

- Combine two tables
- Each tables must have the **same number of columns**
- The columns must have similar / **compatible data types**
- The columns in both tables must be in the **same order**
- Tuples of the second table comes after the first

S_id	Address	Course	Teaching_assistant	Student_lecture1
1234	57 Hockanum Blvd	Database Systems	Minnie	
2345	1400 E. Bellows	Database Systems	Humpty	

S_id	Address	Course	Teaching_assistant	Student_lecture2
3456	900 S. Detroit	Cloud Computing	Dumpty	

```
(SELECT S_id, Course FROM Student_lecture1) UNION  
(SELECT S_id, Course FROM Student_lecture2)
```



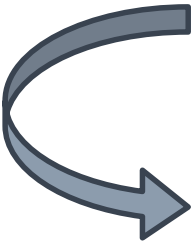
S_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing

INTERSECT

- Combine two tables
- Each table must have the **same number of columns**
- The columns must have similar / **compatible data types**
- The columns in both tables must be in the **same order**
- Tuples of the second table comes after the first

S_id	Address	Course	Teaching_assistant	Student_lecture1
1234	57 Hockanum Blvd	Database Systems	Minnie	
2345	1400 E. Bellows	Database Systems	Humpty	

S_id	Address	Course	Teaching_assistant	Student_lecture2
3456	900 S. Detroit	Cloud Computing	Dumpty	
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey	

 (SELECT S_id FROM Student_lecture1) INTERSECT
(SELECT S_id FROM Student_lecture2)

S_id
1234

Some DBMS (e.g., old versions of MySQL) don't support INTERSECT operator.
Good news!! Our CS server does 😊

Queries Involving Multiple Relations

- Combine the tables based on common columns

TAHiring

computingID	name	year	hours_worked
ht1y	Humpty	4	20
dt2y	Dumpty	3	20
md3y	Mickey	4	15
mn4e	Minnie	4	16
dh5h	Duhhuh	3	10

Payrate

year	hourly_rate
4	12
3	10

```
SELECT computingID, name, TAHiring.year, hourly_rate, hours_worked,  
FROM TAHiring, Payrate  
WHERE TAHiring.year = Payrate.year
```

computingID	name	year	hourly_rate	hours_worked
ht1y	Humpty	4	12	20
dt2y	Dumpty	3	10	20
md3y	Mickey	4	12	15
mn4e	Minnie	4	12	16
dh5h	Duhhuh	3	10	10

DBMS --
"Natural
join"

Wrap-Up

- Keep the number of tables small – reduce the number of tables when you can (in many-to-one and one-to-many)
- Simple, commonly used SQL syntax

What's next?

- More SQL