Database Systems 2

Lecture 1

Introduction

Data vs Information

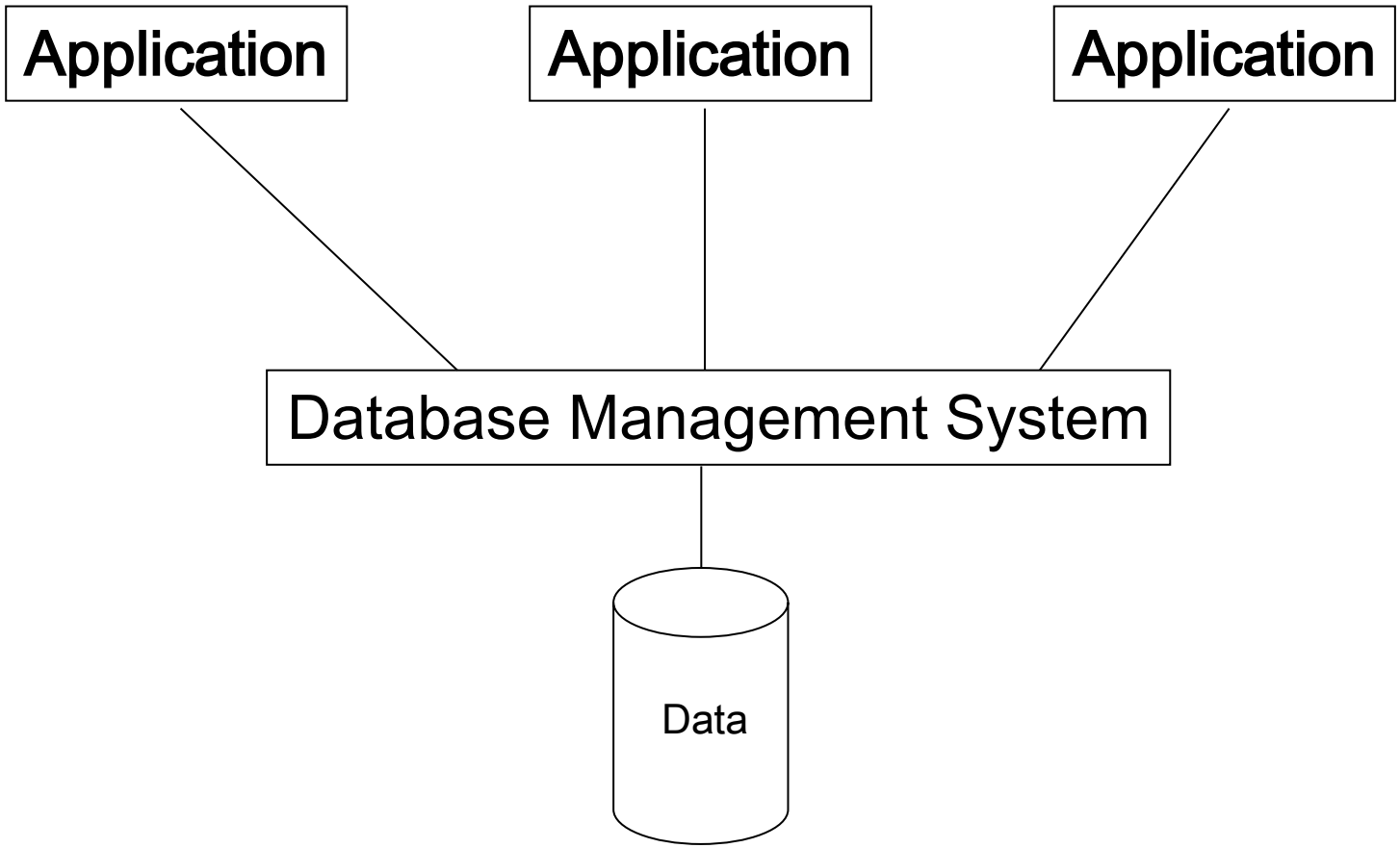
- Data is:
  - Raw facts and figures

- Information is:
  - Data given a context
  - Data given a meaning
  - Data that has been processed

Processing Data into Information

- Classifying
  - Categorise it
- Selecting
  - Find it
- Sorting
  - Rearrange it
- Summarising
  - eg Find Average
- Calculating
  - eg work out VAT.

Database Approach



Database Approach

- Centralised store of data
  - Single repository of data
    - Shared corporate resource
  - Independent of individual applications
    - No one application dictates use, format, etc
  - Self-describing
    - Contains a description of itself – Meta data
      - Data dictionary
  - Program / application independent

Relational Database Model

Many competing models including:

- Hierarchical
- Network

In 1970 Edgar Codd, an IBM employee, defined the theoretical basis for the relational model.

Codd, E. F. (1970). "A relational model of data for large shared data banks". Communications of the ACM 13 (6): 377

Data is stored in tables, which are linked by means of primary and foreign keys.

Primary Key

Definition

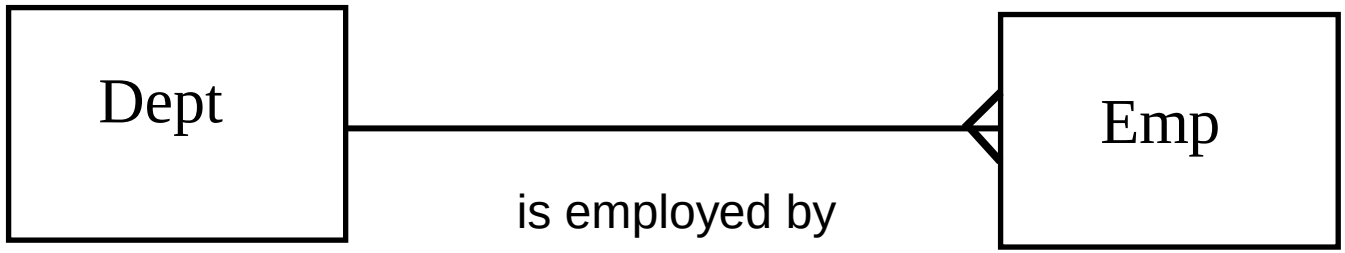
"An attribute (or set of attributes) with the property that, at any time, no two rows of the relation contain the same value in that (set of) attributes."

Primary Key

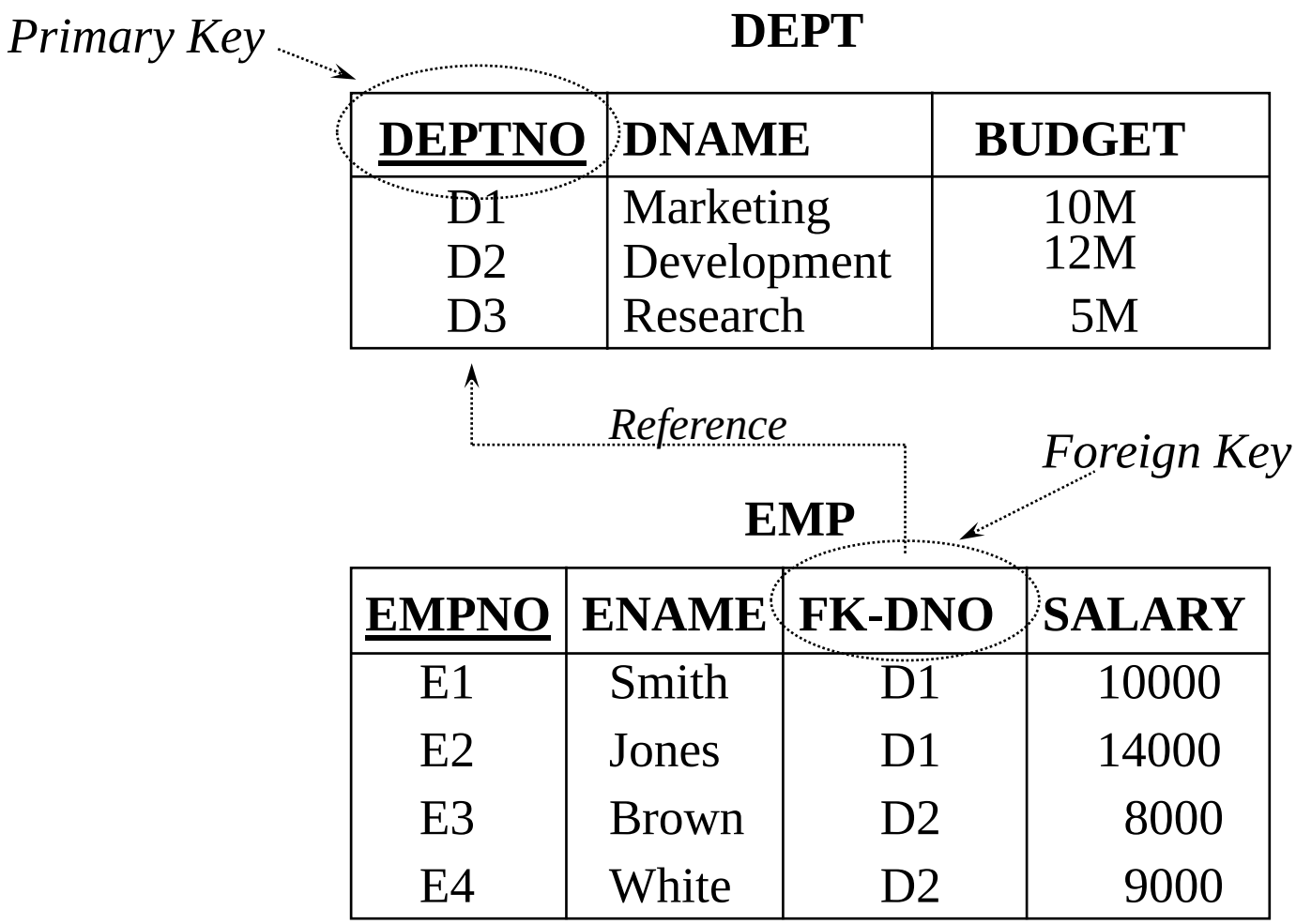
DEPT

DEPTNO	**DNAME**	**BUDGET**
D1	Marketing	10M
D2	Development	12M
D3	Research	10M

Showing Relationships



Foreign Keys



Why not the other way round?

Foreign Key

DEPT

DEPTNO	DNAME	BUDGET	FK-EMPNO
D1	Marketing	10M	E1, E4
D2	Development	12M	E2, E3
D3	Research	5M	E5, E6, E7

EMP

EMPNO	ENAME	SALARY
E1	Smith	10000
E2	Jones	14000
E3	Brown	8000
E4	White	9000

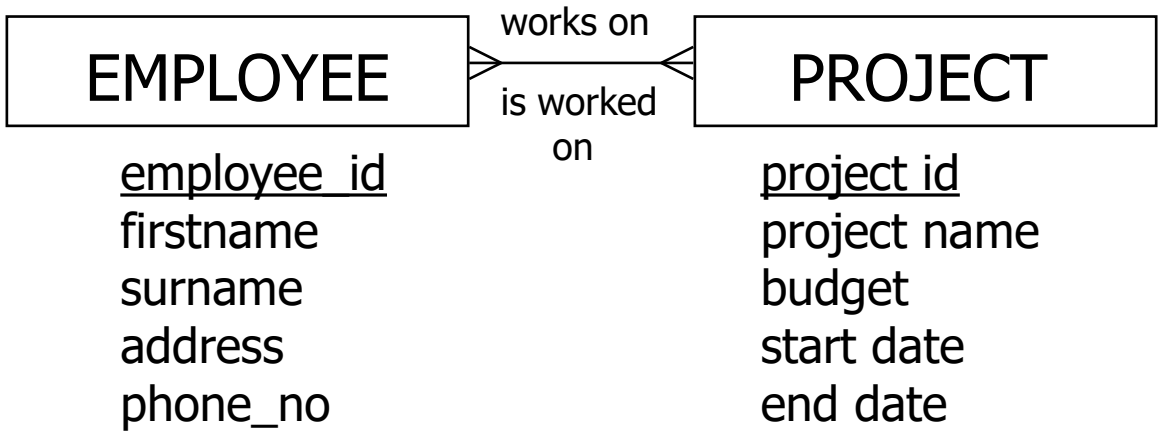
Multiple values in one column shows the database has not been designed properly.

Relationships

- 'An association among entities'
- A *named* significant association
- A binary association that links *two* entities
- Drawn as the line on the diagram
- Cardinality
  - one to one
  - one to many
  - many to many

Many-to-Many Relationship

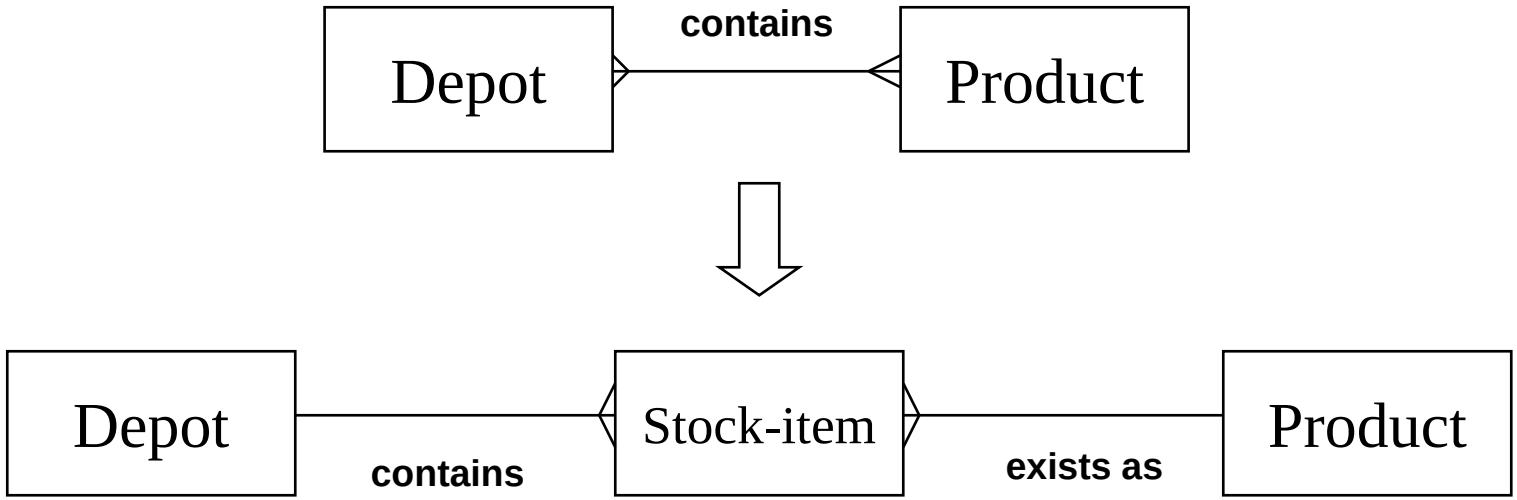
- One EMPLOYEE works on one or more PROJECTs
- One PROJECT can be worked on by one or more EMPLOYEEs



Implementing M:N Relationships

- Resolve all M:N relationship types to two 1:M relationship types

A depot contains many stock-items, a stock-item is one product
A product exists as many stock-items, each stock-item is in one depot

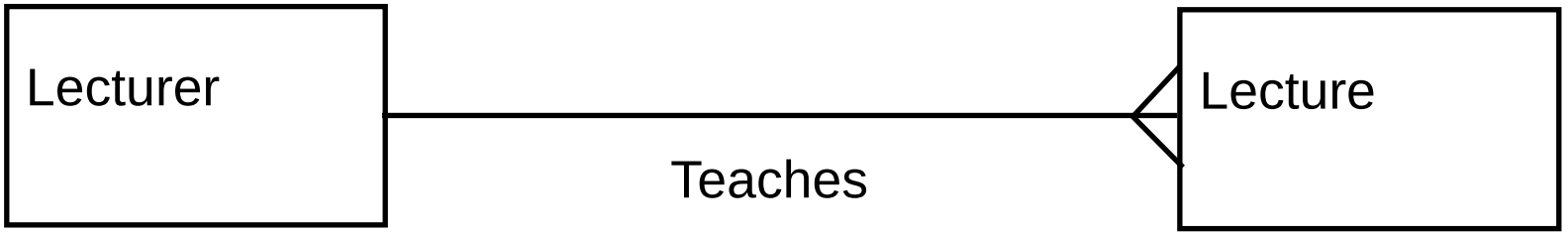


(The 'manys' always go on the link entity)

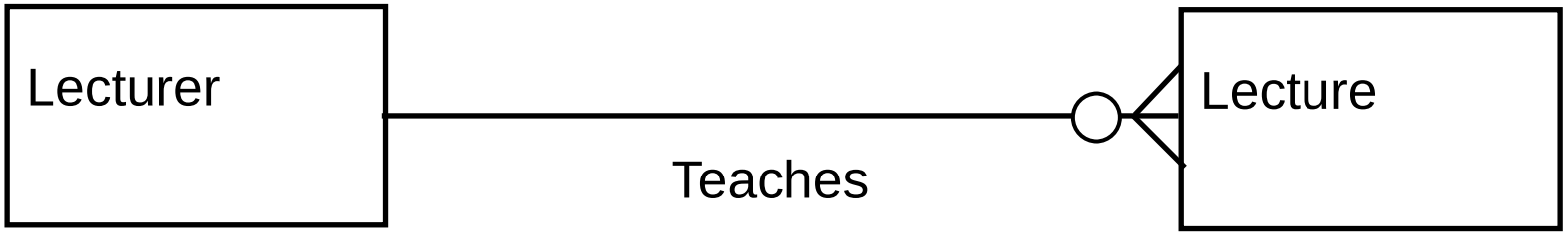
Optionality

- An entity can have
  - mandatory participation, or
  - optional participation
- in a relationship.

- A lecturer must teach one or more lectures.



A lecturer may teach one or more lectures, but may teach none.



School Computer Exercise

What is SQL?

- **S**tructured **Q**uery **L**anguage.
  Pronounced "S-Q-L" (or sometimes "sequel").

- Based on relational algebra

- The standard relational database language:
  - More than 100 RDBMS's support SQL.

- Different manufacturers use slightly different versions of SQL.
    - Microsoft Access SQL
    - Oracle SQL*Plus.
  - Manufacturers' dialects provide features on top of the standard

Data Definition Language Statements

Change the structure of the database tables

```
CREATE TABLE
DROP TABLE
ALTER TABLE tablename ADD COLUMN
ALTER TABLE tablename MODIFY COLUMN
ALTER TABLE tablename DROP COLUMN
```

Data Manipulation Language Statements

Change the contents of the database tables (the data)

```
INSERT INTO tablename VALUES
UPDATE tablename SET
DELETE FROM tablename

SELECT fieldname FROM tablename
```

Referential Integrity

```
CREATE TABLE room
        (
        room_no     CHAR(2),
        capacity    NUMERIC(2),
        PRIMARY KEY ( room_no ),
        );


CREATE TABLE employee
        (
        emp_no        CHAR(2),
        emp_name      CHAR(15),
        room_no       CHAR(2),
        PRIMARY KEY ( emp_no ),
        FOREIGN KEY ( room_no ) REFERENCES room (room_no)
        ON DELETE CASCADE
        );
```

SELECT Queries

The SELECT keyword is used to create queries that retrieve data from a database.

To retrieve all the data from a relation (all columns, all rows):

```
SELECT *
FROM tablename ;
```

SELECT clause lists the attributes

FROM clause lists the tables to be used in the query

Ordering the results of a query

In ASCENDING order :

SELECT ID, name
FROM student
ORDER BY name asc;

ID	Name
3	Bloggs
13	Chambers
11	Harrison
4	Johnson
1	Jones
12	Swift
5	Walker

In DESCENDING order:

SELECT ID, name
FROM student
ORDER BY name desc;

ID	Name
5	Walker
12	Swift
1	Jones
4	Johnson
11	Harrison
13	Chambers
3	Bloggs

Restriction Query

To link WHERE conditions using AND

```
SELECT *
FROM student
WHERE add2 = "Bournemouth"
AND course_id= "BIT";
```

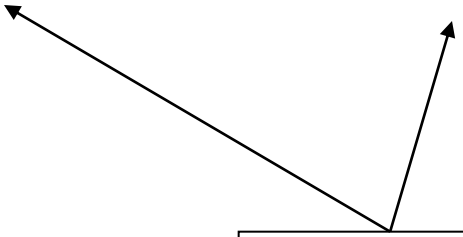
ID	Name	Add1	Add2	Pcode	Course_ID	Year	Dob	Mark	Gender
3	Bloggs	12 Alder Way	Bournemouth	BH15	BIT	1	05-Sep-80	40	M
11	Harrison	10 Daly Road	Bournemouth	BH7	BIT	2	24-Jul-70	85	F

Joining Tables

- The WHERE clause is also used to join tables together - linking the primary key and the foreign key:

```
SELECT name, course_name
FROM student, course
WHERE student.course_id = course.course_id;
```

Name	Course_Name
Jones	Internet Computing
Bloggs	Business
Johnson	Multi-Media
Walker	Internet Computing
Harrison	Business
Swift	Computing
Chambers	Computing



Two columns are labelled course_ID:
primary key of course and.
foreign key of student
So, refer to the columns by tablename.fieldname

Aggregate functions

- These functions operate on **a number of rows** to produce summary information, taking a column name as argument.

```
SELECT Avg(mark) AS AvgOfmark
FROM student;
```

AvgOfmark
55.9

```
SELECT Count(*) AS Females
FROM student
WHERE student.gender="F";
```

Females
5

- Other aggregate functions include: MAX, MIN, SUM
- Why can't you include other fields in the Select line?

Using GROUP BY with aggregate functions

- The GROUP BY clause is used to group selected rows and return a single row of summary information about each group.

```
SELECT Course_id, COUNT(*),
AVG(mark)
FROM student
GROUP BY course_id;
```

Course_ID	Expr1001	Expr1002
BIT	2	62.5
COMP	2	45
ICS	2	72
MCS	1	69

```
SELECT course_id, max(mark),
min(mark)
FROM student
GROUP BY course_id;
```

Course_ID	Expr1001	Expr1002
BIT	85	40
COMP	78	12
ICS	78	66
MCS	69	69

Joining Tables

It is possible to define queries that access more than one table. You are said to be making a join between two tables if you do this.

To join two tables together, there must be a common field – a primary key in one which is a foreign key in the other.

If there isn't, you will get a very unexpected result.

Cartesian Product

SELECT *
FROM emp, dept

will give you an output table consisting of every possible combination of rows from the two input tables.

This is called the Cartesian Product.

There are two main categories of join:

- Inner Joins
- Outer Joins (Left, Right, Full)

Inner Joins

An inner join is one in which a row is output only when there is at least one row in each of the input tables which matches the condition.

```
SELECT  *
FROM emp, dept
WHERE emp.deptno = dept.deptno;
```

Note:

- The FROM statement specifies both tables.
- The WHERE clause is the thing that joins the tables together (order is not important).
- How the table name and the dot is used in front of the field name in the WHERE clause, to avoid ambiguity.
- Due to the *, all fields from both tables are displayed, including both deptno fields.

A Join Between 3 Tables

```
SELECT employee.emp_name, room.room_no, telephone.extension
FROM employee, room, telephone
WHERE employee.room_no = room.room_no
AND room.room_no = telephone.room_no
AND capacity BETWEEN 1 AND 3;
```

You can create queries which join as many tables as you want.

If your queries start to get a bit unwieldy, it may be a sign that you need to rethink the relationships in your database.

Other Ways of Specifying a Join

Instead of using the WHERE clause, you can use the JOIN and ON clauses:

```
SELECT emp_no, emp_name, telephone.room_no, extension
FROM employee JOIN telephone
ON employee.room_no = telephone.room_no;
```

or

```
SELECT emp_no, emp_name, room_no, extension
FROM employee JOIN telephone
USING (room_no);
```

Outer Joins

An Outer join is one in which a row in one of input tables will produce a row in the output table even if there isn't a matching row in the other input table.

```
SELECT room_no, capacity, extension
FROM room LEFT OUTER JOIN telephone
USING (room_no);

SELECT room_no, capacity, extension
FROM room RIGHT OUTER JOIN telephone
USING (room_no);

SELECT room_no, capacity, extension
FROM room FULL OUTER JOIN telephone
USING (room_no);
```

Example Files

room

room_no	capacity
R1	5
R2	4
R3	1
R4	3

telephone

extension	location	room_no*
217	desk	R3
218	wall	R4
219	desk	R5
350	wall	R6

Inner

room_no	capacity	extension
R3	1	217
R4	3	218

Left Outer

room_no	capacity	extension
R1	5	NULL
R2	4	NULL
R3	1	217
R4	3	218

Right Outer

room_no	capacity	extension
R3	1	217
R4	3	218
R5	NULL	219
R6	NULL	350

Full Outer

room_no	capacity	extension
R1	5	NULL
R2	4	NULL
R3	1	217
R4	3	218
R5	NULL	219
R6	NULL	350