COSC344
Database Theory and Applications

Lecture 5
SQL - Data Definition Language

Learning Objectives of Lecture 5

- You should
  - understand the following relational operators
    - Cartesian product
    - Join
    - Natural join
    - Division
  - be able to represent a query using relational algebra expression
  - be able to use the CREATE TABLE command to define
    - Attributes and their data types
    - Primary key, candidate keys, foreign key
  - be able to insert and retrieve date and time data with a given format
- Source:
  - Text book: chapter 8.3, chapter 6.1-6.2
  - Oracle documentation

CARTESIAN PRODUCT

Q = R × S

Produce a new relation **Q** by combining every tuple from relation **R** with every tuple from relation **S**.

R

A	B
a1	b1
a2	b2
a3	b3

X

S

C	D
c1	d1
c2	d2

→

Q

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b2	c1	d1
a2	b2	c2	d2
a3	b3	c1	d1
a3	b3	c2	d2

- # attributes in **Q** = # attributes in **R** + # attributes in **S**
- # tuples in **Q** = # tuples in **R** × # tuples in **S**
- Meaningless unless used with other operations
- Also called CROSS PRODUCT
- **R** and **S** do not have to be union compatible

CARTESIAN PRODUCT Example

Retrieve the names for each female employee and her dependents.

FEMALE_EMPS <– $\sigma_{SEX='F'}$ (EMPLOYEE)

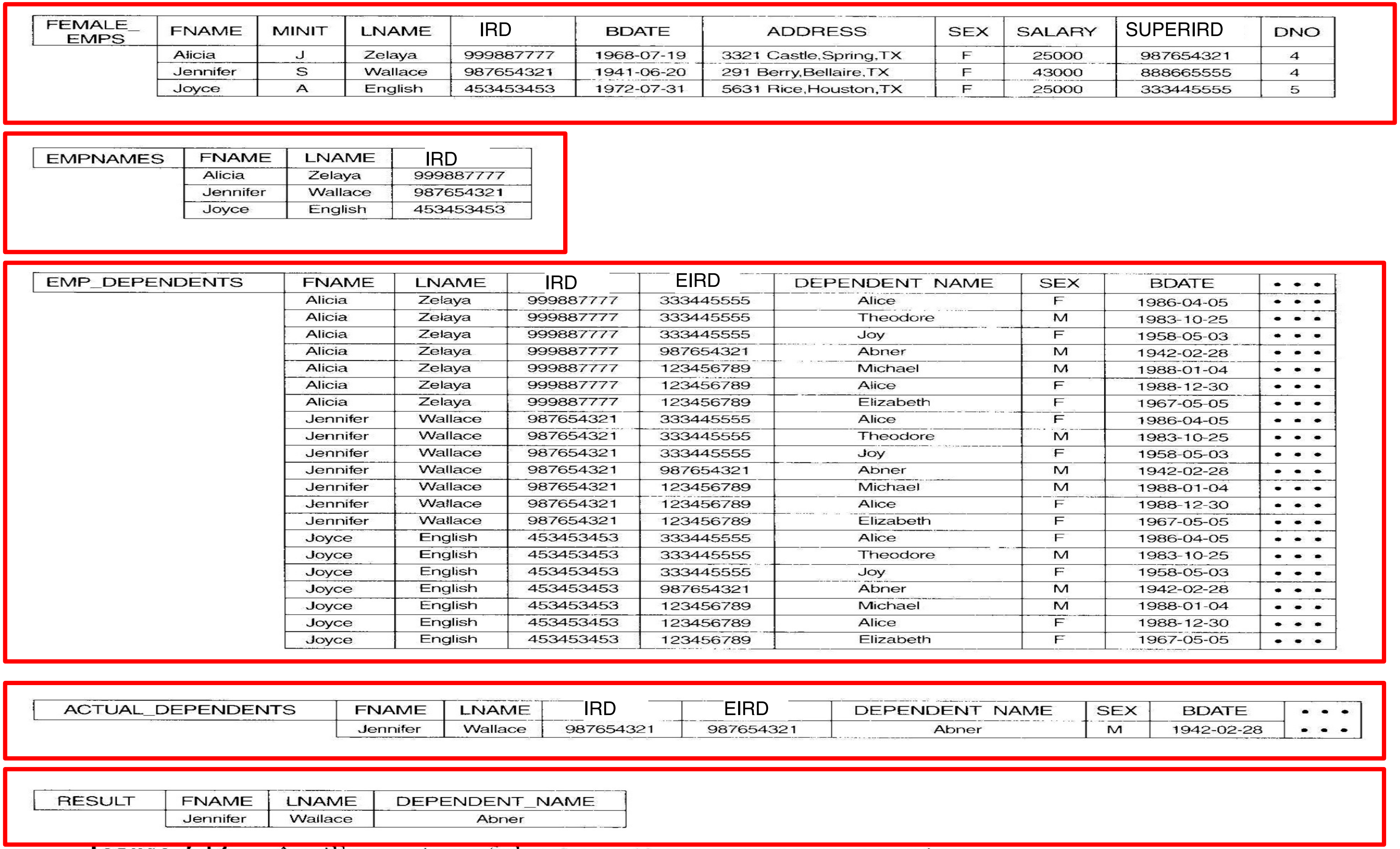
EMPNAMES <– $\Pi_{FNAME,\ LNAME,\ IRD}$ (FEMALE_EMPS)

EMP_DEPENDENS <– EMPNAMES × DEPENDENT

ACTUAL_DEPENDENS <– $\sigma_{IRD=EIRD}$ (EMP_DEPENDENS)

RESULT<– $\Pi_{FNAME,\ LNAME,\ DEPENDENT\_NAME}$ (ACTUAL_DEPENDENS)

CARTESIAN PRODUCT Example (cont'd)



FEMALE_EMPS	FNAME	MINIT	LNAME	IRD	BDATE	ADDRESS	SEX	SALARY	SUPERIRD	DNO
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle,Spring,TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
	Joyce	A	English	453453453	1972-07-31	5631 Rice,Houston,TX	F	25000	333445555	5

EMPNAMES	FNAME	LNAME	IRD
	Alicia	Zelaya	999887777
	Jennifer	Wallace	987654321
	Joyce	English	453453453

EMP_DEPENDENTS	FNAME	LNAME	IRD	EIRD	DEPENDENT_NAME	SEX	BDATE	• • •
	Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	• • •
	Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	• • •
	Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	• • •
	Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	• • •
	Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	• • •
	Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	• • •
	Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	• • •
	Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	• • •
	Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	• • •
	Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	• • •
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	• • •
	Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	• • •
	Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	• • •
	Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	• • •
	Joyce	English	453453453	333445555	Alice	F	1986-04-05	• • •
	Joyce	English	453453453	333445555	Theodore	M	1983-10-25	• • •
	Joyce	English	453453453	333445555	Joy	F	1958-05-03	• • •
	Joyce	English	453453453	987654321	Abner	M	1942-02-28	• • •
	Joyce	English	453453453	123456789	Michael	M	1988-01-04	• • •
	Joyce	English	453453453	123456789	Alice	F	1988-12-30	• • •
	Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	• • •

ACTUAL_DEPENDENTS	FNAME	LNAME	IRD	EIRD	DEPENDENT_NAME	SEX	BDATE	• • •
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	• • •

RESULT	FNAME	LNAME	DEPENDENT_NAME
	Jennifer	Wallace	Abner

Figure 7.12 An illustration of the CARTESIAN PRODUCT operation.

JOIN

$Q = R \bowtie_{\text{<join condition>}} S$

Produce a new relation **Q** that contains the combinations of tuples in **R** and **S** satisfying the join condition.

R

A	B
a1	b1
a2	b1
a3	b2
a4	b3

⋈ B = BB

S

BB	C
b1	c1
b2	c2

→

Q

A	B	BB	C
a1	b1	b1	c1
a2	b1	b1	c1
a3	b2	b2	c2

- The main difference from CARTESIAN PRODUCT is that only the combinations of tuples satisfying the join condition appear.
- The join condition can be a compound set of conditions using AND and OR.
- The usual comparison operators are used. When the comparison operator is =, it is called an **EQUIJOIN.**

JOIN (cont'd)

- In the earlier example illustrating CARTESIAN PRODUCT, the following operations

EMP_DEPENDENS <– EMPNAMES × DEPENDENT

ACTUAL_DEPENDENS <– $\sigma_{IRD=EIRD}$ (EMP_DEPENDENS)

can be replaced with a single JOIN operation

ACTUAL_DEPENDENS <– EMPNAMES $\bowtie_{IRD=EIRD}$ DEPENDENT

NATURAL JOIN

Q = R * S

Performs an Equijoin of the two relations **R** and **S** over all common attributes. One occurrence of each common attribute is eliminated from **Q**

R

A	B
a1	b1
a2	b1
a3	b2
a4	b3

*

S

B	C
b1	c1
b2	c2

→

Q

A	B	C
a1	b1	c1
a2	b1	c1
a3	b2	c2

- The standard definition requires that the two join attributes (or each pair of join attributes) have the same name in both relations.
- If not, a renaming operation is applied first.

DIVISION

Q = R ÷ S

Produce a relation **Q** in which each tuple must appear in **R** in combination with ***every*** tuple in **S**.

R

A	B
a1	b1
a2	b1
a1	b2
a4	b2

÷

S

B
b1
b2

→

Q

A
a1

- Can be expressed as a sequence of PROJECT, CARTESIAN PRODUCT and DIFFERENCE

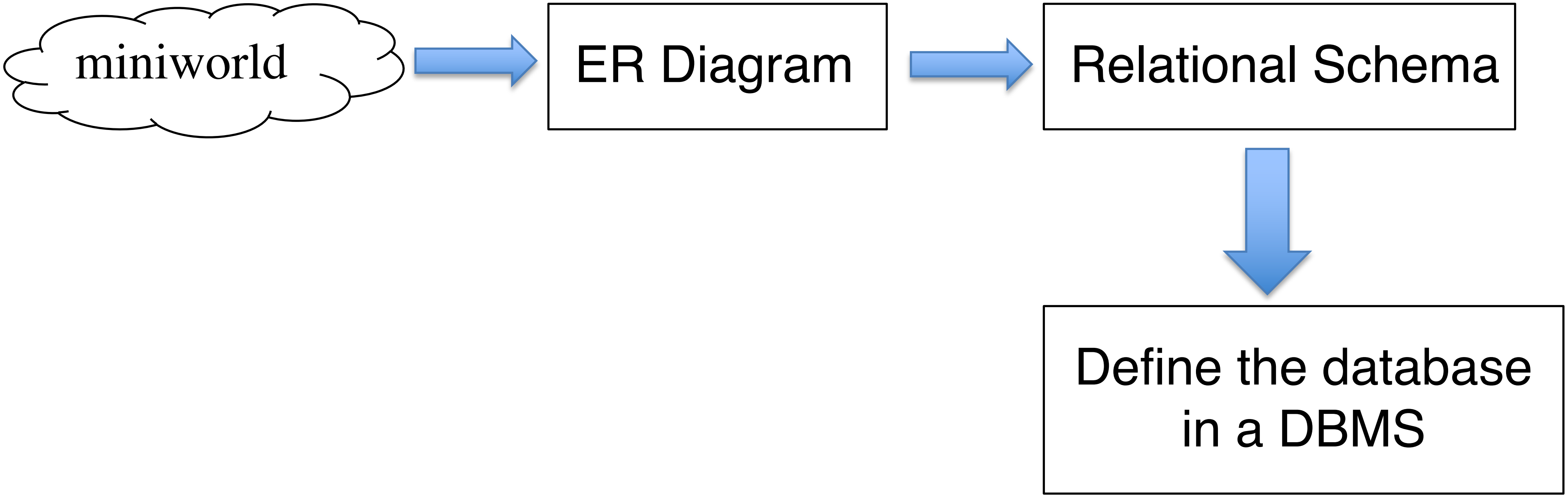
$$Q_1 \leftarrow \Pi_A (R)$$

$$Q_2 \leftarrow \Pi_A ((S \times Q_1) - R)$$

$$Q \leftarrow Q_1 - Q_2$$

A Complete Set of Relational Operators

- The six primitive relational operators are:
  - SELECT
  - PROJECT
  - UNION
  - DIFFERENCE
  - CARTESIAN PRODUCT
  - RENAME
- Any other original relational operators can be expressed as a sequence of operations using the above six operators.
- The relational algebra includes operators like JOIN and DIVISION as they are convenient to use in database applications.



miniworld

ER Diagram

Relational Schema

Define the database in a DBMS

SQL - History & Introduction

- Structured Query Language (SQL)
- Standard
  - ANSI & ISO
  - SQL (1986)
  - SQL2 (1992)
  - SQL3 (1999)
  - SQL - 2003
  - SQL - 2006
- Declarative
- Includes DDL and DML
- Other facilities
- Standard
  - Well almost

The CREATE Statement

- The main SQL command for data definition, which can create
  - Schema, table, view, assertions, triggers, …
- A simple CREATE table example

```
CREATE TABLE test
    (i  INT,
     r  NUMBER(6,2),
     s  VARCHAR2(20) );
```

- Convention on SQL-scripts
  - Case is unimportant except for text strings.
  - Using upper case letters for SQL keywords and lower case for other things.

Data Types

- Oracle Built-in data types
  - NUMBER(p, s): precision p∈[1,38], scale s∈ [-84, 127]
  - CHAR(size): fixed-length character data
  - VARCHAR2(size): variable-length character string
  - DATE
  - CLOB: a large object containing characters
  - BLOB: a binary large object
  - BFILE: contains a locator to a large binary file
  - …
- ANSI Data types
  - INTEGER/INT: equivalent to NUMBER(38)
  - CHARACTER(n): equivalent to CHAR(n)
  - …

Refer to Oracle Documentations for more detailed data types

The CREATE TABLE Command

- Used to create a new relation by
  - Giving it a name, specifying its attributes and data types
  - Specifying constraints: attributes, tuples, key and referential integrity
- CREATE TABLE syntax

```
CREATE TABLE tablename (
  column1  datatype  [constraint][,
  [CONSTRAINT [symbol]] PRIMARY KEY [index_type]
                (index_col_name,...)[index_option] ...

  [CONSTRAINT [symbol]] FOREIGN KEY
     [index_name] (index_col_name,...) reference_definition
  CHECK (expr)
 );
```

Attributes Constraints

- NOT NULL
  - Specify that attribute values are not allowed to be set to NULL
  - Always implicitly specified for attributes that are part of the primary key
- Attribute Defaults: DEFAULT <value>
  - The default value is included in any new tuple if an explicit value is not provide for that attribute
  - Example: Dno INT NOT NULL DEFAULT 1,
- CECHK clause
  - Restrict attribute or domain values following an attribute or domain definition
  - Oracle does not have a CREATE DOMAIN command
  - Example
    - Suppose *Dnumber is restricted to integer between 1 and 20*

Dnumber INT NOT NULL CHECK (Dnumber >0 and Dnumber <21),

Key and Entity Integrity Constraints

- PRIMARY KEY clause
  - The primary key is a single attribute

```
Dunmber    INT  PRIMARY KEY,
```

  - The primary key is a composite of multiple attributes

```
PRIMARY KEY (Eird, Pno),
```

- UNIQUE clause: specifies candidate keys

```
Pname  VARCHAR2(15)   NOT NULL   UNIQUE,
```

Referential Integrity Constraints

- FOREIGN KEY clause: specifies referencing integrity
  - Can be defined by following the attribute directly

```
dnum   INT   NOT NULL   REFERENCES   department(dnumber),
```

  - Can be defined using the FOREIGN KEY clause

```
FOREIGN KEY  (dnum)    REFERENCES   department(dnumber),
```

- Specify actions to deal with integrity violations
  - SET NULL, CASCADE, SET DEFAULT

```
FOREIGN KEY  (Superird)    REFERENCES  employee(ird)
                  ON DELETE  SET DEFAULT    ON UPDATE   CASCADE,
```

Giving Names to Constraints

- The names of all constraints within a particular schema must be unique.

- A constraint name is used to identify a particular constraint in case the constraint must be dropped later or replaced with another constraint.

```
CONSTRAINT  superird_cnst  REFERENCES employee(ird),
```

A Complex CREATE TABLE Example

```
CREATE TABLE EMPLOYEE
  (Fname VARCHAR2(10) NOT NULL,
   Minit CHAR,
   Lname VARCHAR2(50) NOT NULL,
   Ird   CHAR(9)     PRIMARY KEY,
   Bdate DATE,
   Address VARCHAR2(30),
   Sex   CHAR  CONSTRAINT sck  CHECK (sex IN (‘M’,’F’)),
   Salary DECIMAL(10,2),
   Superird  CHAR(9)
    CONSTRAINT superird_cnst REFERENCES EMPLOYGEE(ird) DISABLE,
   Dno INT NOT NULL DEFAULT 1  REFERENCES DEPARTMENT(Dnumber)
   );
```

Date and Time

- SQL specifies
  - DATE: Year, Month and Day in the form YYYY-MM-DD
  - TIME: Hour, Minute and Second in the form HH:MM:SS
  - TIMESTAMP: include both DATE and TIME plus a minimum of six positions for decimal fractions of seconds in the form YYYY-MM-DD HH:MI:SS[.ssssss]
- Oracle uses DATE for both date and time
  - DD - number of days in month (e.g., 24)
  - MM – numeric month (e.g., 07)
  - MON – abbreviated month name (e.g., JUL)
  - MONTH – full month name (e.g., JULY)
  - YY - last two digits of year (e.g., 12)
  - YYYY - all digits of year (e.g., 2012)
  - HH (or HH12) - hours of the day in 12 hour format
  - HH24 - hours of the day in 24 hour format
  - MI - minutes of the hour
  - SS - seconds of the minute
  - …

Date and Time in Oracle

- Use TO_DATE to specify the format
  - TO_DATE('string', 'format')
  - Must be used when entering dates
  - e.g, TO_DATE('20-03-2000','dd-mm-yyyy')

```
INSERT INTO x VALUES
(......... , ......., TO_DATE(‘27-Jul-2010’, ‘dd-mon-yyyy’));
```

- Use TO_CHAR to format the output of a date
  - TO_CHAR(attr, 'format')
  - TO_CHAR(bdate, 'dd/mon/yy')

```
SELECT TO_CHAR (bdate, 'dd/mon/yy’)
FROM  EMPLOYEE;
```

DROP/ALTER TABLE in Oracle

- DROP TABLE:

```
DROP TABLE table [CASCADE CONSTRAINTS];
```

  - CASCADE CONSTRAINTS: drop all referential integrity constraints that refer to primary and unique keys in the dropped table.

    DROP TABLE project;

    DROP TABLE employee cascade constraints;

- ALTER TABLE

```
ALTER  TABLE table

[ADD/DROP (column_element |

           constraint)]

[MODIFY (column_element)]

[ENABLE | DISABLE constraint]
```

- Add or drop a column
- Change a column definition
- Add or drop a column constraint

ALTER TABLE in Oracle

- Add a column

```
ALTER TABLE employee  ADD  Job  VARCHAR(12);
```

- Change a column definition

```
ALTER TABLE employee  MODIFY  Mgrird
                          SET DEFAULT ‘333445555’;
```

- Add a column constraint

```
ALTER TABLE employee ADD CONSTRAINT superird_cnst
FOREIGN KEY (Superird) REFERENCES employee(ird);
```

```
ALTER TABLE employee ENABLE CONSTRAINT superird_cnst;
```

Summary

- Relational algebra
  - Cartesian product
  - Join
  - Natural join
  - Division
- SQL - DDL
  - CREATE TABLE
  - Data types
  - Constraints
  - DROP TABLE
  - ALTER TABLE