Dr Vasa Curcin

School of Population Health
and Environmental Sciences /
Department of Informatics

Faculty of Natural and Mathematical Sciences

February 2021

KING'S College LONDON

4CCS1DBS – Database Systems

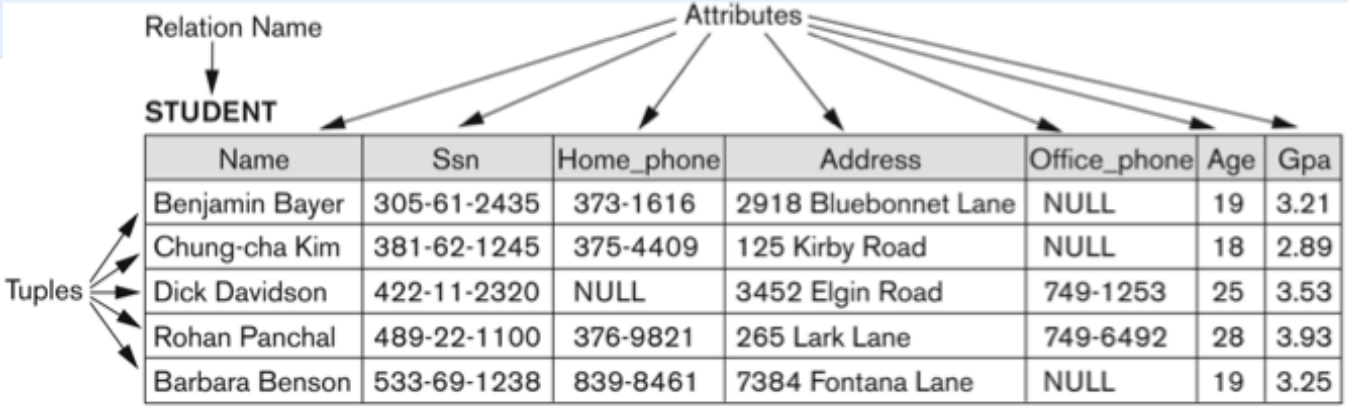
Week 4 – Structured Query Language (SQL)

Topic: Data Manipulation Language

Week 4 – Learning Outcomes

- Understand the basics of SQL
- Learn the schema definition operations
- Translate the constraints from relational data model to SQL
- Learn how to pose database queries using SQL

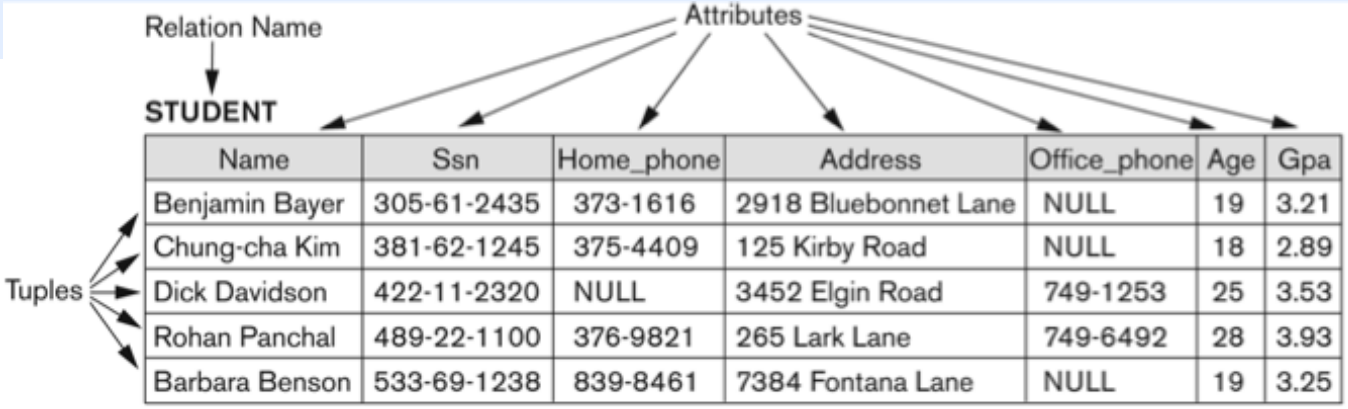
Review: Relational Model Informal Definitions



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows.**
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

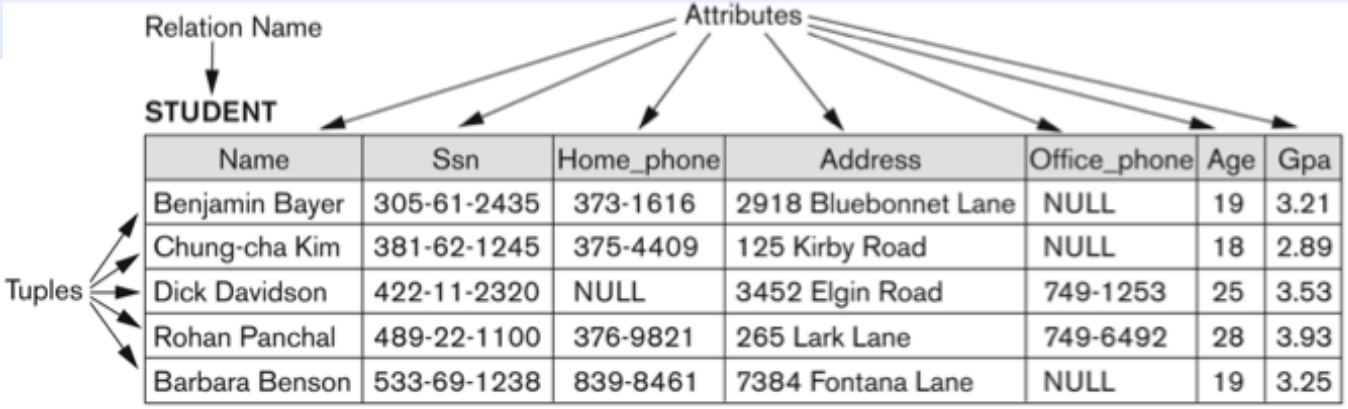
Review: Relational Model Informal Definitions



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows.**
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

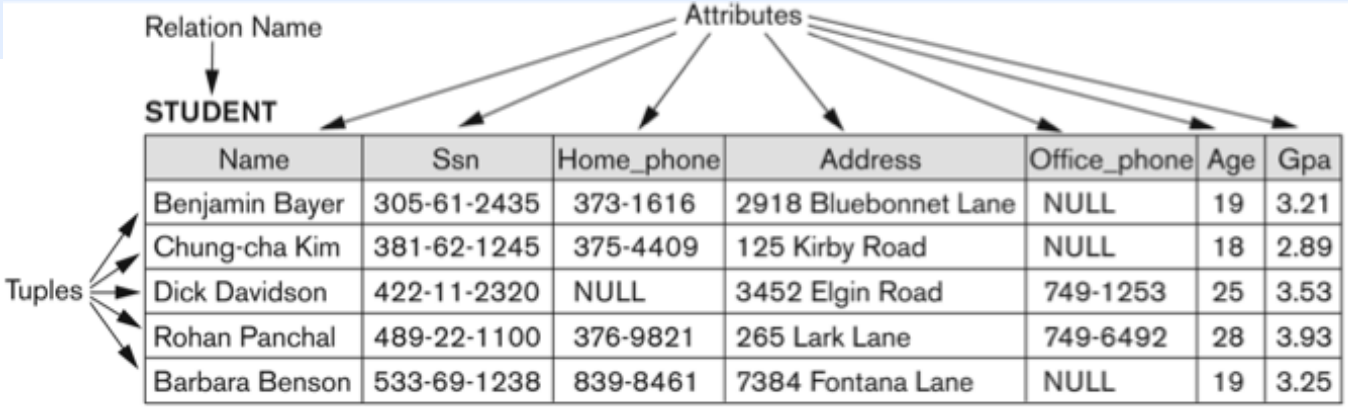
Review: Relational Model Informal Definitions



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows.**
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Review: Relational Model Informal Definitions



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows.**
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
    - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
    - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Review: Relational Integrity Constraints

- Constraints are **conditions** that must hold in **all** valid relation states.
- Three *main types* of constraints in the relational model:
  - **Entity integrity** constraints – Keys must not be null and unique
  - **Referential integrity** constraints -Two relations reference each other using foreign keys
- Additionally, **domain constraints** are implicit:
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)
- Constraints specific to the data domain that you are building a database for are **semantic integrity constraints**.
    - ☐ (e.g. grades must be < 100%)

Review: Entity Integrity

- **Entity Integrity:**
- The *primary key attributes* PK of each relation schema R in S cannot have **null** values in any tuple of r(R)
  - Primary key values are used to *identify* the individual tuples.
  - If PK has several attributes, **null** is not allowed in any of these attributes

- Note: Other attributes in R can be constraint to *not hold **null** values*. Even though they are not members of the primary key.

Review: Referential Integrity

- Involves **two** relations and the integrity of their relationship
- Used to maintain CONSISTENCY among tuples in two relations, by specifying a **relationship** among these tuples:
  - The **referencing relation** and the **referenced relation**

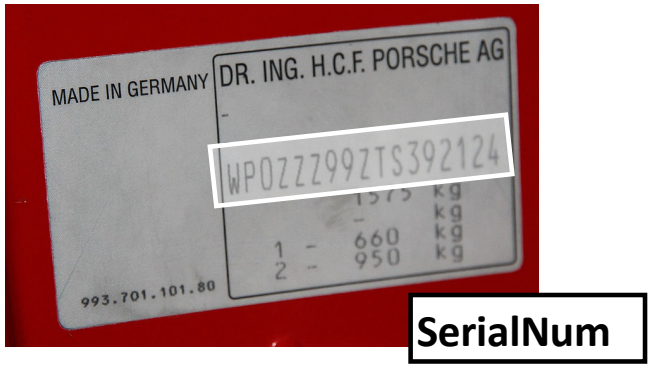
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the **primary key** attributes PK of the **referenced relation** R2.
  - A tuple *t1* in *R1* is said to **reference** a tuple *t2* in *R2* if t1[**FK**] = t2[**PK**].

Review: Key Constraints

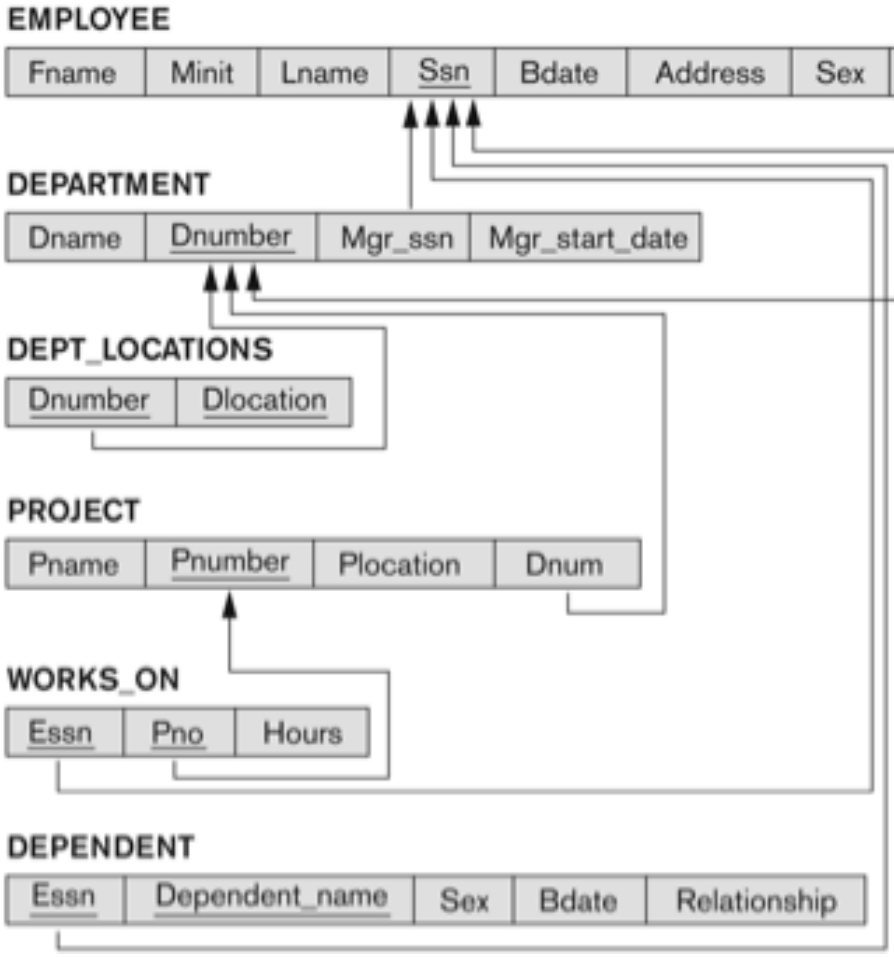
- **Superkey** of relation **R** - A set of attributes **SK** where:
  - No two tuples in any valid relation state **r(R)** will have same value for **SK**
  - Must hold in *any valid state* **r(R)**
  - Implicitly all of the attributes of a relation is a Superkey, e.g. The full row **{Name,SSN, etc etc}**
- **Key** of R:
  - A **"*minimal*"** superkey
  - A key **(K)** is a superkey where if you remove any attribute from K, results in a set of attributes that is not a superkey
  - All Keys are Superkeys. Not all Superkeys are Keys.

Review: Key Constraints

- **Example**: Consider the CAR relation schema (US):
  - CAR(State, RegNum, SerialNum, Make, Model, Year)
  - CAR has two possible keys:
    - Key 1 = {State, RegNum}
    - Key 2 = {SerialNum}
  - Both are also **superkeys** of CAR
  - {SerialNum, Make} is a **superkey** but *not* a **key**



Review: Referential Integrity in COMPANY database



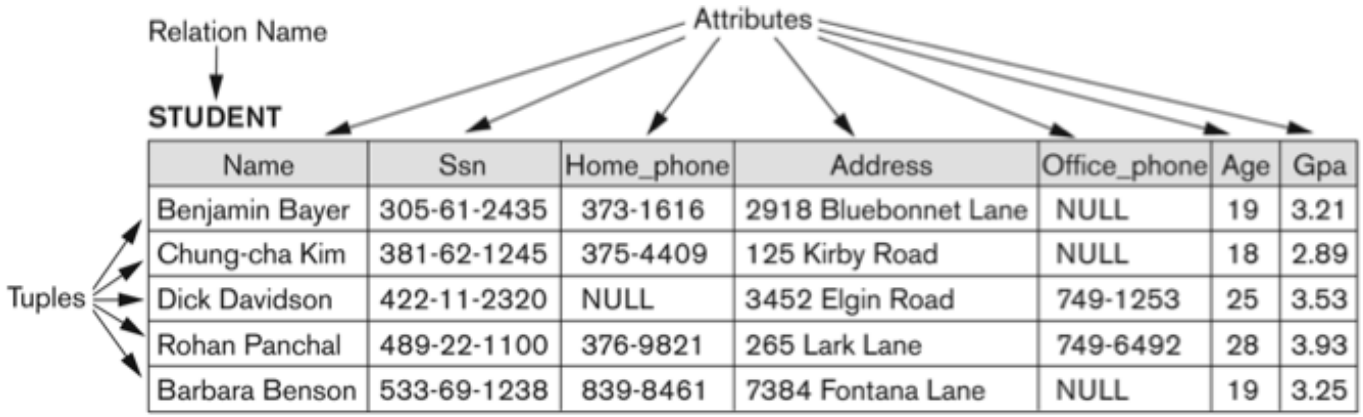
Key Points:

Arrows:
Point from **FK** to **PK**

KEYs:
For DEPENDENT:

- (ESSN,Dname) = PK
- ESSN = FK, referencing EMPLOYEE
- ESSN cannot be NULL

Review: Keys in COMPANY database



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Superkeys = All rows are distinct!

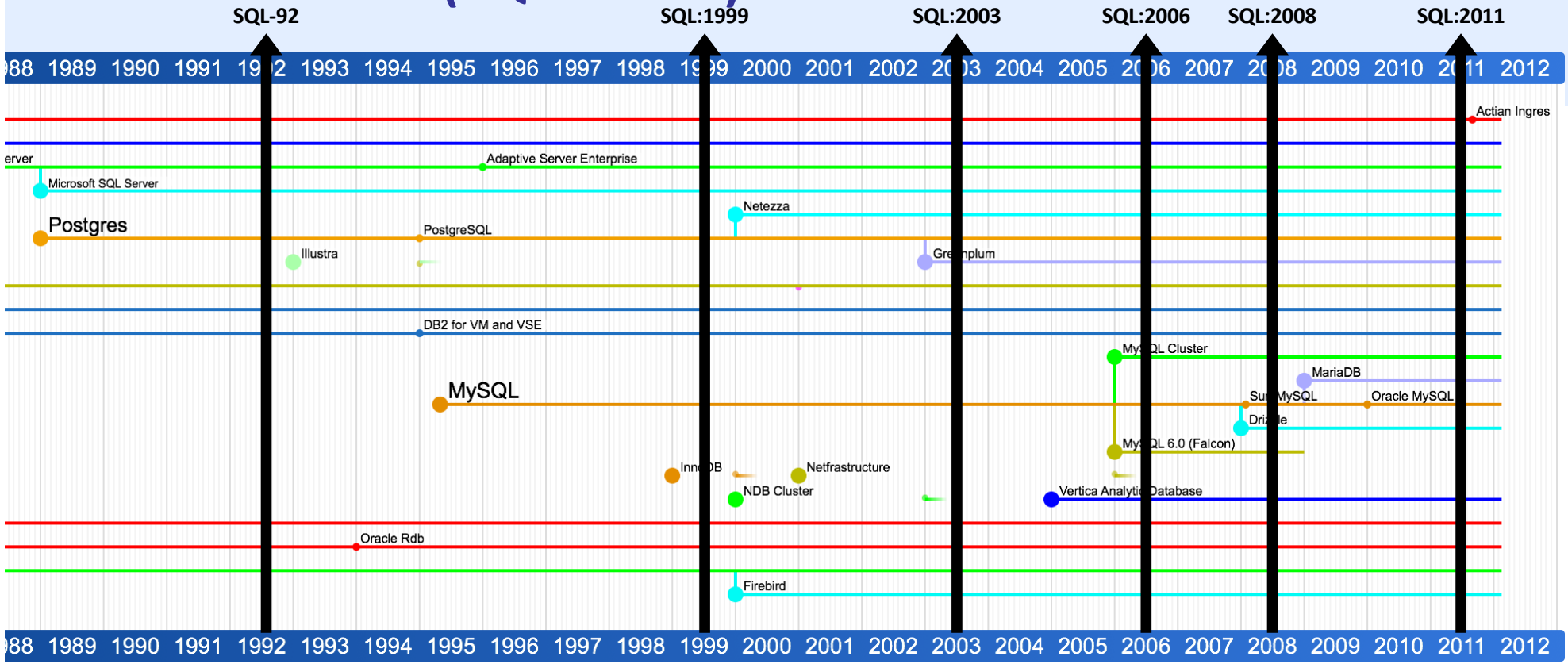
{Name,SSN,Home_phone,Address,Office_phone,Age,GPA}
{Name,SSN,Home_phone,Address,Office_phone,Age}
{Name,SSN,Home_phone,Address}
etc etc etc

Candidate key = a superkey which only has the minimal set of attributes needed to uniquely identify a row!

SQL: Structured Query Language

- Originally named SEQUAL (**S**tructured **E**nglish **QUE**ry Language)
- Designed and implemented by IBM Research as interface of experimental relational database SYSTEM R
  - High-level “non-navigational” language, ad-hoc queries
  - Support rapidly changing database environment
- Used for data definition as well as queries an updates (both DDL and DML)
- SQL can also specify **authorization** and security, define integrity **constraints**, define **views**, specify **transaction controls**

Timeline RDBMS (SQL 1973 -)



Source: https://en.wikipedia.org/wiki/User:Intgr/RDBMS_timeline

Important take aways:

- Core SQL concepts and Keywords most common in SQL compliant versions (backward compat.)
- Concept that SQL will have variation, and not to get stuck in a vendor-specific version.
- Learn how to adapt to a particular version.

Relational Model Terms compared to SQL Terms

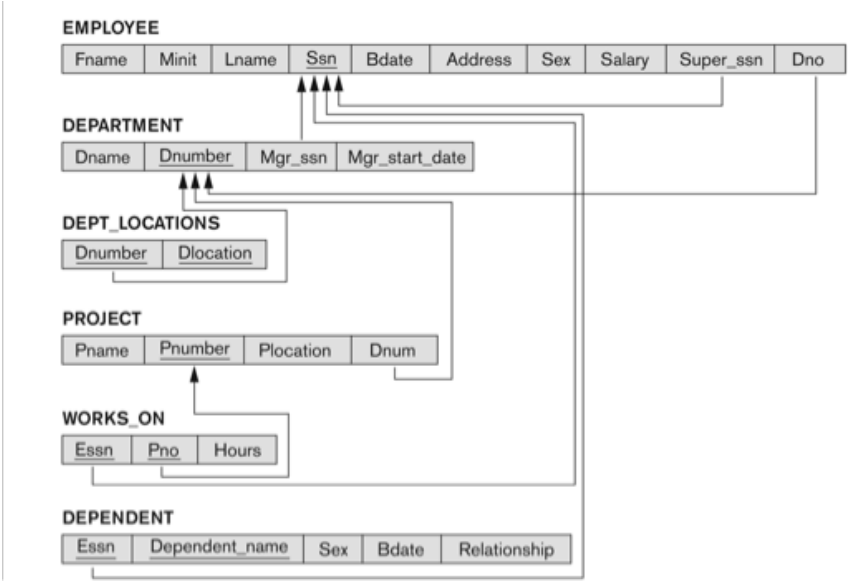
Formal Terms (Relational Model)	**Informal Terms (SQL)**
Relation	Table
Attribute	Column Header
Domain	All possible Column Values
Tuple	Row
Schema of a Relation	Table Definition
State of the Relation	Populated Table

Data Definition, Constraints and Schema Changes

- Data Definition Language (**DDL**) Commands
  - CREATE – create a description of the relations
  - DROP – delete the descriptions
  - ALTER – update the descriptions

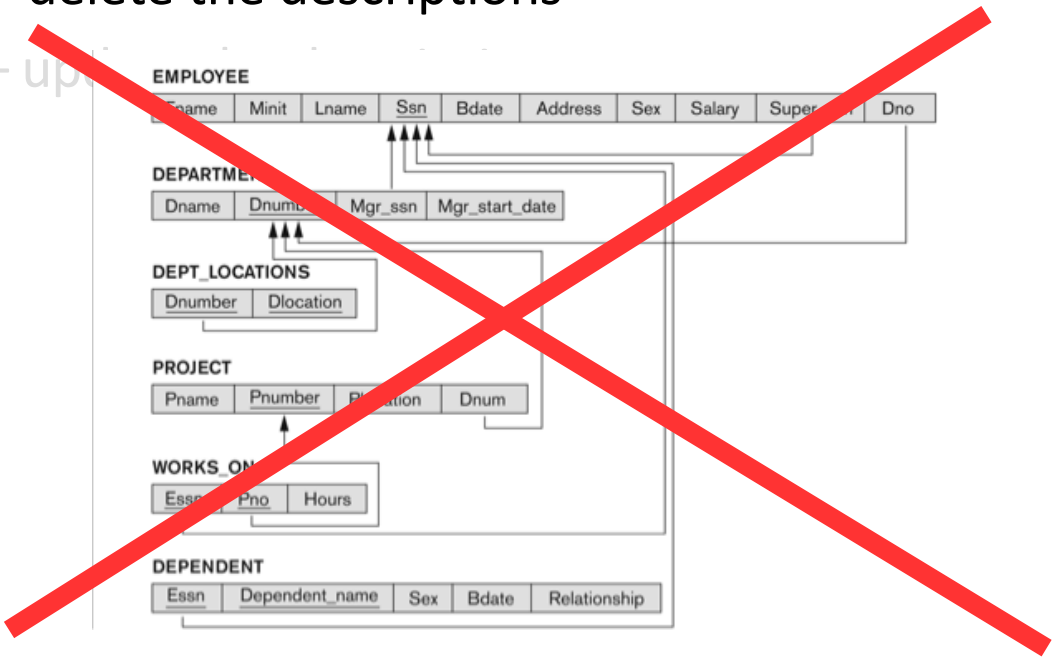
Data Definition, Constraints and Schema Changes

- Data Definition Language (**DDL**) Commands
  - CREATE – create a description of the relations
  - DROP – delete the descriptions
  - ALTER – update the descriptions



Data Definition, Constraints and Schema Changes

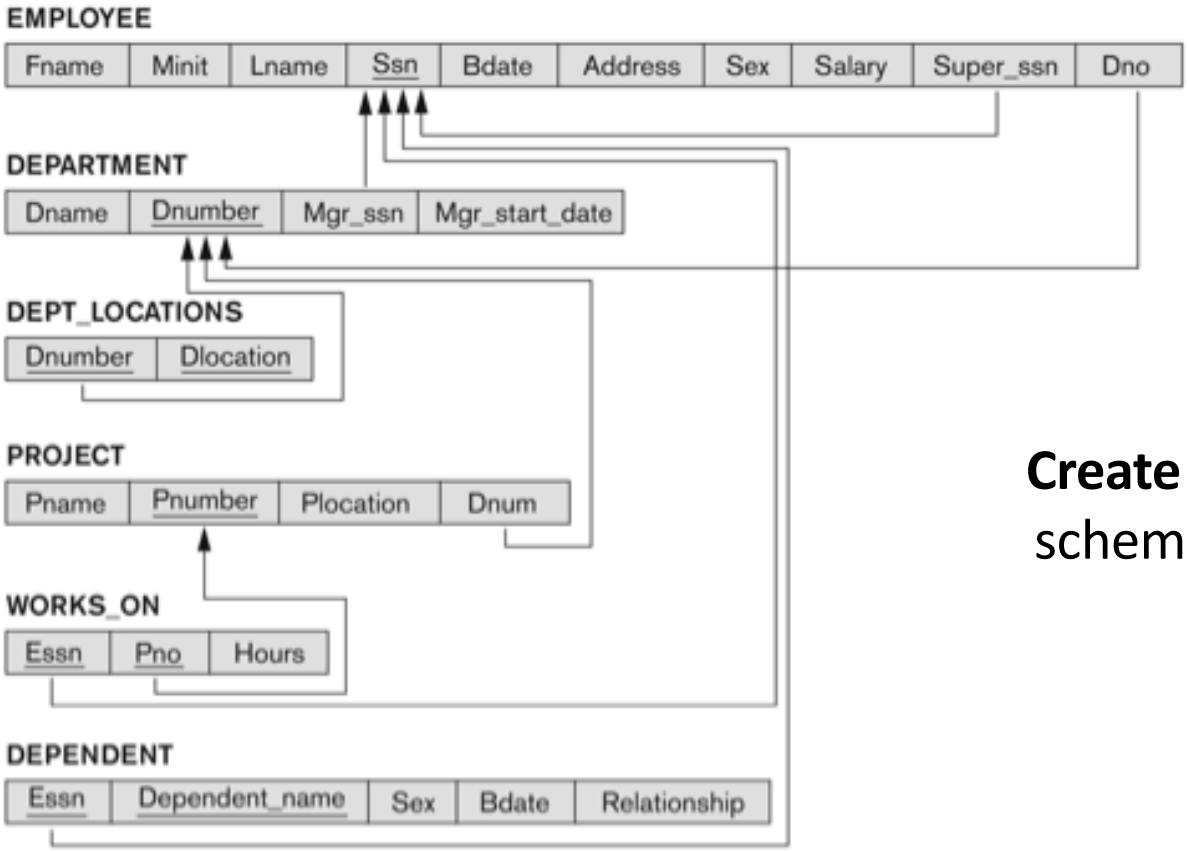
- Data Definition Language (**DDL**) Commands
  - CREATE – create a description of the relations
  - DROP – delete the descriptions
  - ALTER – up



Data Definition, Constraints and Schema Changes

- Data Definition Language (**DDL**) Commands
  - CREATE – create a description of the relations
  - DROP – delete the descriptions
  - ALTER – update the descriptions

COMPANY database



Create a schema

CREATE SCHEMA

- Specifies a new database schema by giving it a name.

```
CREATE SCHEMA COMPANY AUTHORIZATION JSMITH;
```

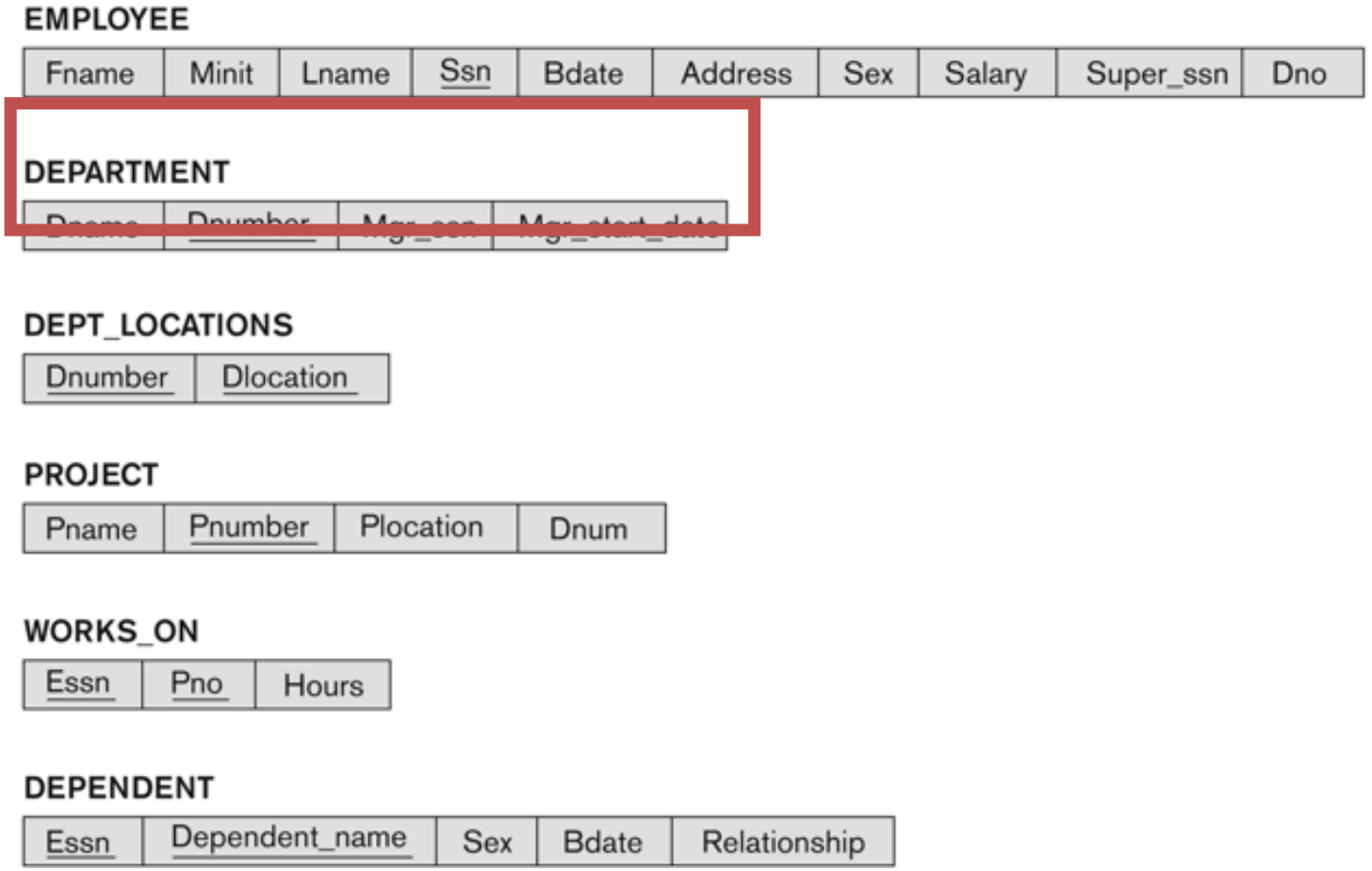
- Selects a schema to be defined:

```
USE COMPANY;
```

- Multiple schemas exist within a database, although some RDBMSs have schema and a database as synonymous concept.

- Comments:
  - `-- line comments are two dashes`

The COMPANY Database Schema



CREATE TABLE — Name of Relation

```
CREATE TABLE DEPARTMENT (
      DNAME       VARCHAR(10)     NOT NULL,
      DNUMBER     INTEGER         NOT NULL,
      MGRSSN      CHAR(9),
      MGRSTARTDATE    DATE
);
```

- Specifies a new base relation by giving it a name, and specifying each of it's attributes.

CREATE TABLE — Attributes

```
CREATE TABLE DEPARTMENT (
      DNAME      VARCHAR(10)     NOT NULL,
      DNUMBER    INTEGER         NOT NULL,
      MGRSSN     CHAR(9),
      MGRSTARTDATE   DATE
);
```

- In SQL attributes are ordered based on the order they are specified.

Attribute Names and Table Names Reserved Words in SQL

- **Vendor specific** … and can be a long list…
- How to deal with reserved words?
  - "I have an attribute named 'CREATE' or 'GROUP'…"
- Use back-ticks ` if you *really* want to use reserved words

```
`GROUP`   INTEGER   NOT NULL
```

Table 9.2 Keywords and Reserved Words in MySQL 5.5

ACCESSIBLE (R)	ACTION	ADD (R)
AFTER	AGAINST	AGGREGATE
ALGORITHM	ALL (R)	ALTER (R)
ANALYZE (R)	AND (R)	ANY
AS (R)	ASC (R)	ASCII
ASENSITIVE (R)	AT	AUTHORS
AUTOEXTEND_SIZE	AUTO_INCREMENT	AVG
AVG_ROW_LENGTH	BACKUP	BEFORE (R)
BEGIN	BETWEEN (R)	BIGINT (R)
BINARY (R)	BINLOG	BIT
BLOB (R)	BLOCK	BOOL
BOOLEAN	BOTH (R)	BTREE
BY (R)	BYTE	CACHE
CALL (R)	CASCADE (R)	CASCADED
CASE (R)	CATALOG_NAME	CHAIN
CHANGE (R)	CHANGED	CHAR (R)

Example: https://dev.mysql.com/doc/refman/5.5/en/keywords.html

CREATE TABLE — Initial Constraints

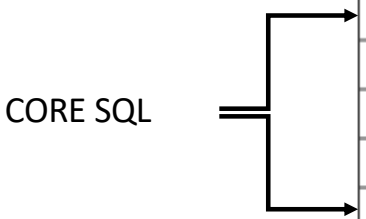
```
CREATE TABLE DEPARTMENT (
      DNAME       VARCHAR(10)    NOT NULL
      DNUMBER     INTEGER        NOT NULL,
      MGRSSN      CHAR(9),
      MGRSTARTDATE   DATE
);
```

- Attributes can have initial constraints defined, as in NOT NULL.

Attribute Data Types - Numerics - Integer

Example: MySQL Implementation of Integer Types

Type	Storage (Bytes)	Minimum Value (Signed/Unsigned)	Maximum Value (Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615



- **INT** or **INTEGER** are signed implicitly
- **UNSIGNED INTEGER** are unsigned
- **INT(n)** - denotes the number of *digits (i.e. INT(2) 0-99)*
- Size depends on implementation

Attribute Data Types - Numerics - Real Numbers

- **Approximate Value**:
  - FLOAT, REAL, DOUBLE
  - Can specify digit precision (DB does *rounding*)
  - FLOAT(**n**)
    - n - precision, as in ***number of bits used to store the mantissa of the float number in scientific notation***
  - *Example: FLOAT(24) holds a single precision floating point number*

- **Exact Value** (Fixed-Point Type)
  - DECIMAL(i,j) for exact formatted numbers
    - i - precision, ***total number of digits to store number***
    - j - scale, *after decimal*
  - *Example: DECIMAL(7,4) will look like -999.9999 when displayed*
  - More expensive to store, but more precise

Attribute Data Types - Character - String

- **CHAR(n)**, **CHARACTER(n)** — fixed length, right padded with spaces
- **VARCHAR(n)**, **CHAR VARYING(n)**, **CHARACTER VARYING(n)** — varying length
- **CLOB / TEXT** — CHARACTER LARGE OBJECT

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Attribute Data Types - Binary Data

- **BIT(n)** — fixed length
- **BIT VARYING(n)** — varying length
- **BLOB** — Binary Large Object. Megabyte, Gigabyte.

- Typically, people do not store files (images etc…) or documents in a RDBS.
- Stored on a File System with reference (URL, path)
- **PRO TIP**: Do *hash* your binary data and store that in the DB with the metadata. (Use “md5 hash” of your binary data)

Attribute Data Types - Boolean

- **BOOLEAN** - TRUE or FALSE
- Can implement with BIT(1) — 1 bit
- Implementations vary, examples:
  - MySQL uses a BIT(1), so 0 or 1 as False / True
  - PostgreSQL stores these any of these literals:

TRUE	FALSE
't'	'f'
'true'	'false'
'y'	'n'
'yes'	'no'
'on'	'off'
'1'	'0'

Any data type can also be NULL…

What is NULL?

- Attributes in SQL can have the value of **NULL** (unless of course they are specified with NOT NULL constraints)
- **NULL** means *Unknown data*, and **does not mean False**
- With Boolean values and Conditionals, there is a Three Value Logic System in SQL

p	*q*	*p* OR *q*	*p* AND *q*	*p* = *q*
True	True	True	True	True
True	False	True	False	False
True	Unknown	True	Unknown	Unknown
False	True	True	False	False
False	False	False	False	True
False	Unknown	Unknown	False	Unknown
Unknown	True	True	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown

p	NOT *p*
True	False
False	True
Unknown	Unknown

Attribute Data Types - Date and Time

- **DATE**— made up of year-month-day (“yyyy-mm-dd”)

  Feb 11 2016 -> 2016-11-02

- **TIME** — made of up hour:minute:second (“hh:mm:ss")

  15:43 and 21 seconds - 15:43:21

- **TIME(i)** — TIME plus i addition digits for fractions of second (hh:mm:ss:ii…i)
  - TIME(3) -> precision for milliseconds
- **DATETIME** / **TIMESTAMP** — both DATE and TIME components

Attribute Data Types – Interval data

- **INTERVAL** — relative time value as opposed to absolute
  - Can be DAY/TIME intervals of YEAR/MONTH intervals
  - Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

Examples of Interval Values that can be stored:

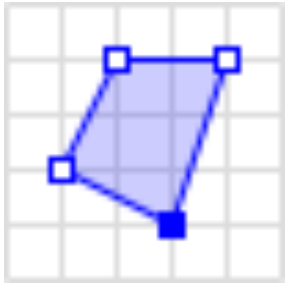
INTERVAL 1 DAY

INTERVAL 3 MONTH

INTERVAL 2 HOUR

Domain Specific / Complex Types

- Special Types:
  - CURRENCY, MONEY
- Spatial Types (GIS)
  - GEOMETRY type can store geometries
    - POINT(0 0)
    - LINESTRING(0 0,1 1,1 2)
    - POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))



- Enumerated Types: ENUM(“One”,”Two”,”Three”)
- Collection Types: SET, VALUE_MAP

Good to know about and be aware of this slide, but stick to the Core and Primitive Types that we covered in SQL for this course.

Attribute Data Types - CREATE DOMAIN

- CREATE DOMAIN — allows you to specify your own data type to use in the schema.

```
CREATE DOMAIN SSN AS CHAR(9);

CREATE TABLE DEPARTMENT ( DNAME    VARCHAR(10)        NOT NULL,
                          DNUMBER  INTEGER            NOT NULL,
                          MGRSSN   SSN,
                          MGRSTART DATE               DATE
                          );
```

- Need to CREATE DOMAIN before utilizing in CREATE TABLE

Specifying Integrity Constraints

- NOT NULL - for enforcing that attributes cannot take NULL values

```
DNUMBER INTEGER NOT NULL;
```

- DEFAULT <value>

```
MGRSSN CHAR(9) DEFAULT '123456789';
```

- AUTO_INCREMENT - for INTEGER types, helpful with IDs (usually starts with 1, but check implementation…)

```
DNUMBER INTEGER NOT NULL AUTO_INCREMENT;
```

Specifying Domain Constraints - CHECK

- CHECK clause — requires a valid conditional expression

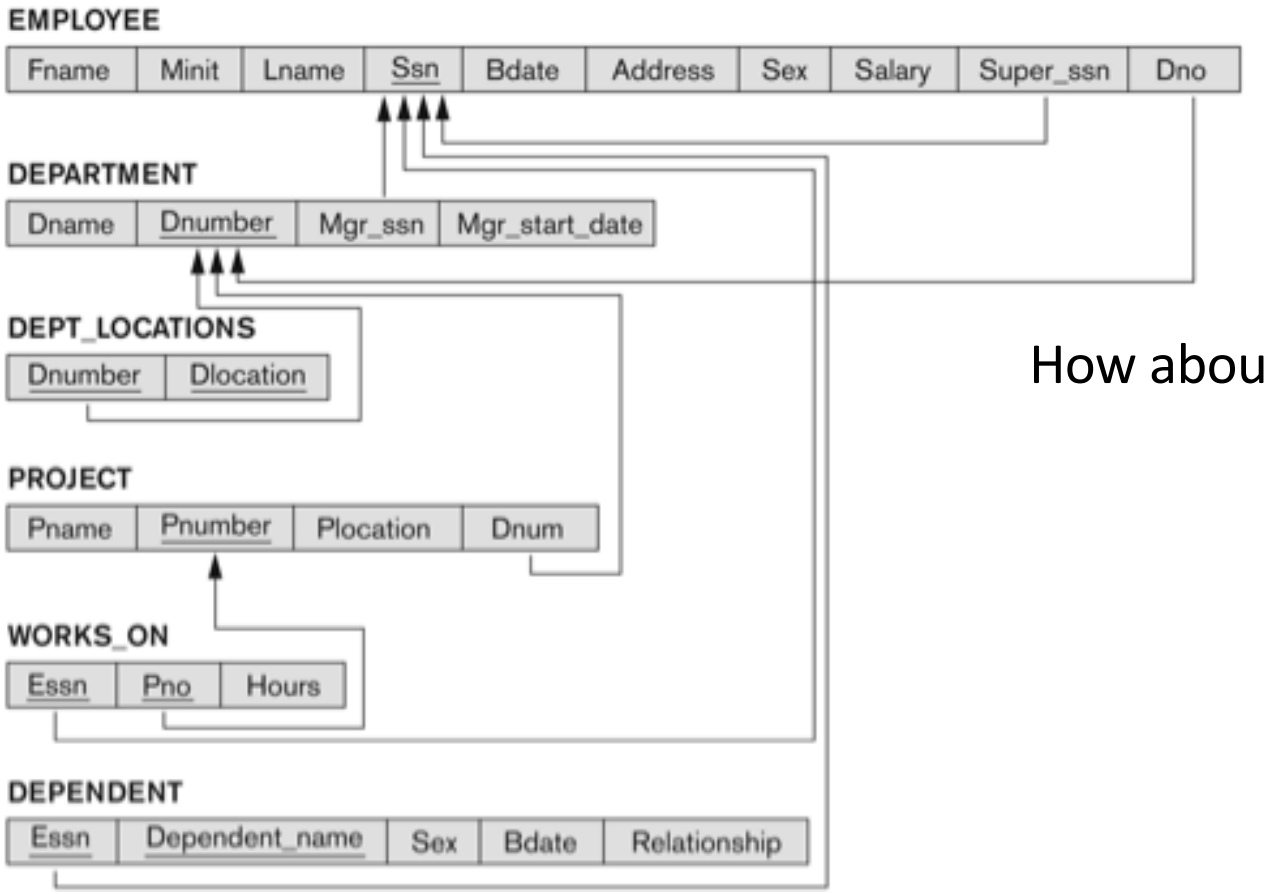
```
DNUMBER INT CHECK (DNUMBER >0 AND DNUMBER < 21);
```

- Use with CREATE DOMAIN (note the use of VALUE to reference the attribute name):

```
CREATE DOMAIN D_NUM AS INTEGER CHECK
(VALUE > 0 AND VALUE < 21);
```

- Check is unable to compare against other attributes/relations — need to use an ASSERTION (next lecture)…

Key/Referential Integrity Constraints for COMPANY database



How about keys?

Key and Referential Integrity Constraints

- PRIMARY KEY clause

```
DNUMBER INT PRIMARY KEY;
```

Don't need to state "NOT NULL" for PK, it is automatic!

- UNIQUE clause - for secondary/alternate keys

```
DNAME CHAR(9) UNIQUE;
```

- FOREIGN KEY clause — for referential integrity

```
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN);
```

CREATE TABLE — Integrity Constraints

```
CREATE TABLE DEPARTMENT (
        DNAME           VARCHAR(10)     NOT NULL,
        DNUMBER         INTEGER         NOT NULL,
        MGRSSN          CHAR(9),
        MGRSTARTDATE    DATE,
        PRIMARY KEY(DNUMBER),
        UNIQUE(DNAME),
        FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN)
);
```

- CREATE TABLE can specify *primary key attributes*, *secondary keys*, and *referential integrity constraints* (foreign keys) **after** the attributes.
- Key attributes specified via PRIMARY KEY and UNIQUE

CREATE TABLE — Composite Primary and Foreign Keys

```
CREATE TABLE DEPARTMENT (
        DNAME           VARCHAR(10)     UNIQUE NOT NULL,
        DNUMBER         INTEGER         NOT NULL,
        MGR_FNAME       CHAR(9),
        MGR_LNAME       CHAR(9),
        MGRSTARTDATE    DATE,
        PRIMARY KEY(DNUMBER, DNAME),
        FOREIGN KEY(MGR_FNAME, MGR_LNAME) REFERENCES EMPLOYEE FNAME, LNAME
);
```

- Composite PRIMARY or FOREIGN KEYS are specified after the attributes.

Referential Integrity Options

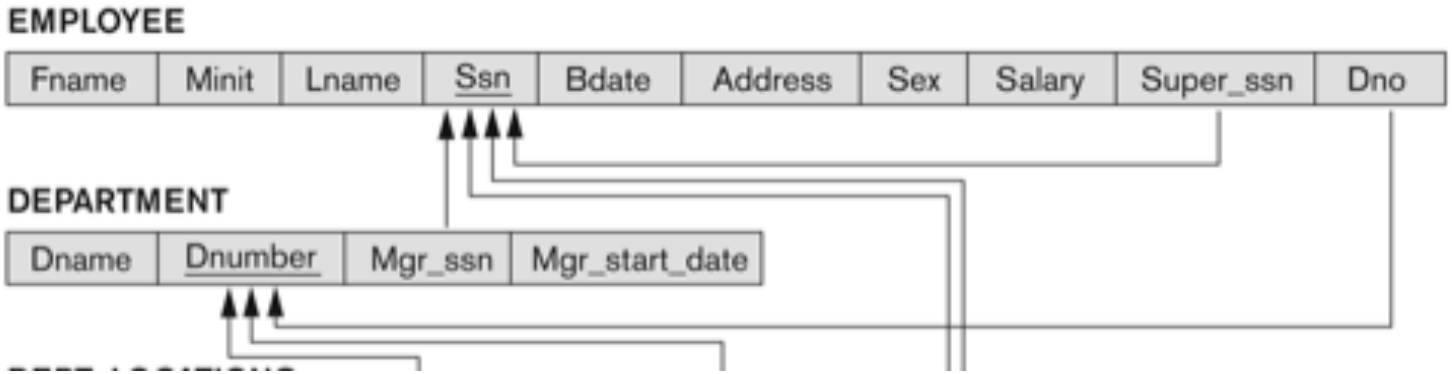
- A referential integrity constraint may be ***violated*** when tuples in the referenced tuple are updated/deleted.
- Default: ***reject*** the operation that violates constraint
- Or: specify a referential triggered action, options:

Events **(what occurs on Referenced Tuple with PK)**	**Triggered Action** ***(to happen on Referencing Tuple with FK)***
ON DELETE	RESTRICT
	CASCADE
ON UPDATE	SET NULL
	SET DEFAULT

- *Example*:
  - ON DELETE SET NULL
  - ON UPDATE CASCADE

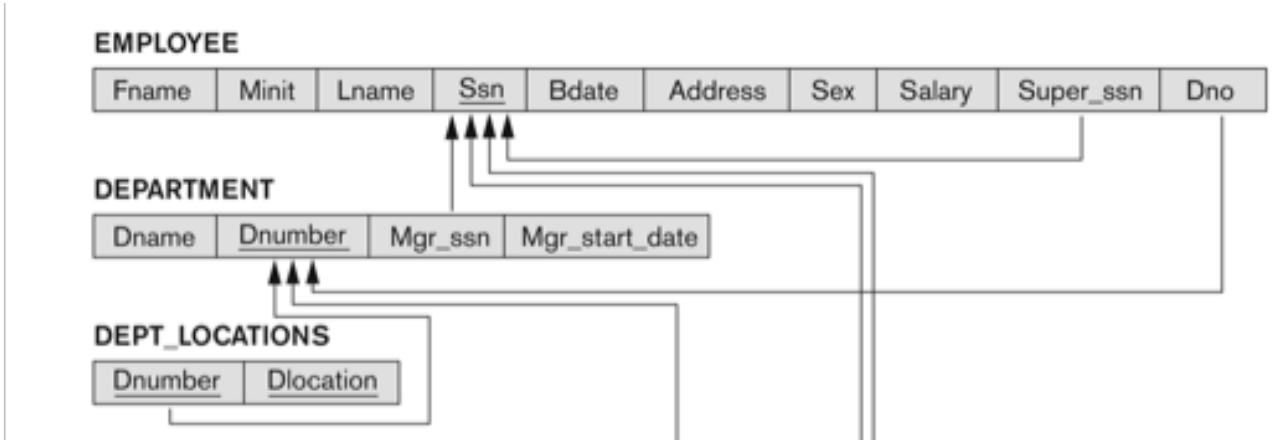
Referential Integrity Options: DEPARTMENT Example

```
CREATE TABLE DEPARTMENT (
...
        FOREIGN KEY(MGRSSN)REFERENCES EMPLOYEE(SSN)
        ON DELETE SET DEFAULT ON UPDATE CASCADE
...
);
```



Referential Integrity Options: EMPLOYEE Example

```
CREATE TABLE EMPLOYEE (
…
        FOREIGN KEY(DNO)REFERENCES DEPARTMENT(DNUMBER)ON DELETE SET
DEFAULT ON UPDATE CASCADE,
        FOREIGN KEY(SUPERSSN)REFERENCES EMPLOYEE(SSN)ON DELETE SET
NULL ON UPDATE CASCADE,
…
);
```



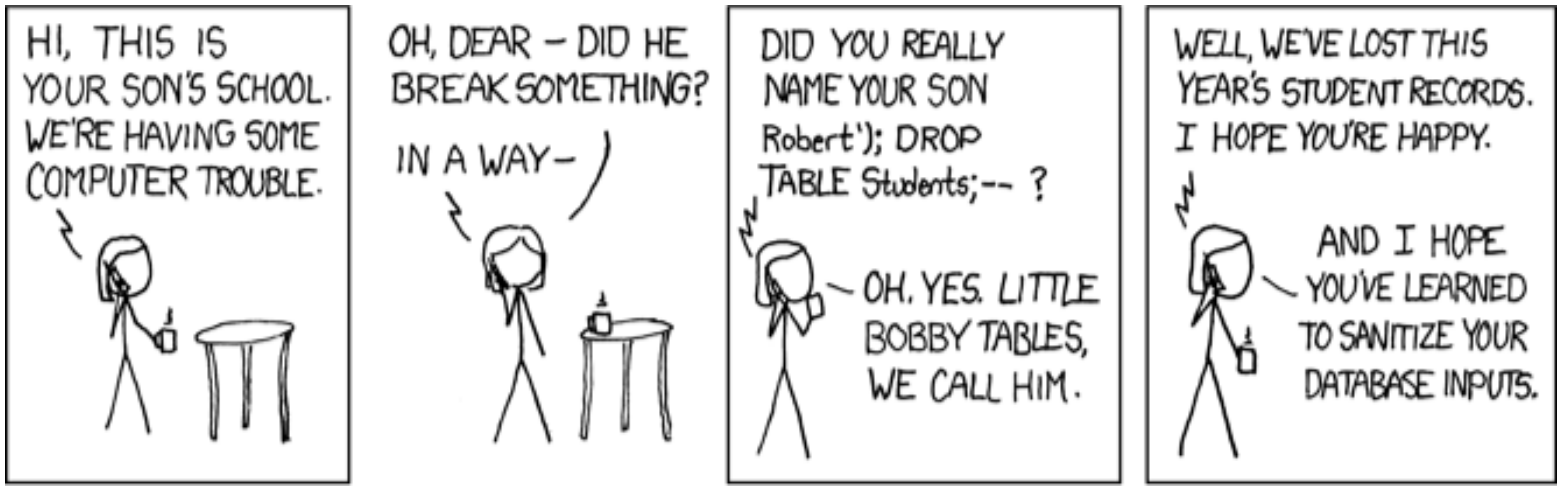
CREATE TABLE — Base Relations vs. Virtual Relations

- ***Base tables*** (or ***base relations***): Relations and tuples store as a file by DBMS
  -> Created through the CREATE TABLE statement

- Base tables are distinguished from ***virtual relations*** which may not correspond to an actual physical file
  -> Created through the CREATE VIEW statement
    - (More on Views next week)

- Remember: while attributes in CREATE TABLE are *ordered,* tuples (rows) are not considered ordered.

Changing a Database Schema



Exploits of a Mom - https://xkcd.com/327/

DROP TABLE

- Used to ***remove*** a relation (base table) and its definition
- Relation is unable to be used in queries, updates, or any commands since it's description no longer exists
- Example

```
DROP TABLE DEPENDENT;
```

- Table dropped if not referenced in any constraints

```
DROP TABLE DEPENDENT RESTRICT;
```

- All constraints that reference table are dropped along with table.

```
DROP TABLE DEPENDENT CASCADE;
```

DROP SCHEMA

- Used to remove the entire **schema**
- Similar distinction with RESTRICT vs. CASCADE as with DROP TABLE

- Dropped only if no elements in schema

```
DROP SCHEMA COMPANY RESTRICT;
```

- ***All*** tables, views, and constraints dropped (!!!)

```
DROP SCHEMA COMPANY CASCADE;
```

ALTER TABLE - ADD

- Used to add an attribute to one of the base relations
- New attribute will have NULLs in all existing tuples of relation

```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);
```

- Database users will need to UPDATE a value for the new JOB attribute for the existing Employees.
- Utilize a DEFAULT value

```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12) DEFAULT
'President';
```

- Specifying NOT NULL will require a DEFAULT value

ALTER TABLE - ADD constraints

- Depending on which order tables are created, circular Referential Integrity Constraints may need to be added later.
- Example, DEPARTMENT and EMPLOYEE reference each other:

```
ALTER TABLE EMPLOYEE ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT
  (Dnumber);

ALTER TABLE DEPARTMENT ADD FOREIGN KEY (MGRSSN) REFERENCES
EMPLOYEE (Ssn);
```

ALTER TABLE - DROP

- Can remove attributes (which removes data), although some RDBS do not allow removing columns

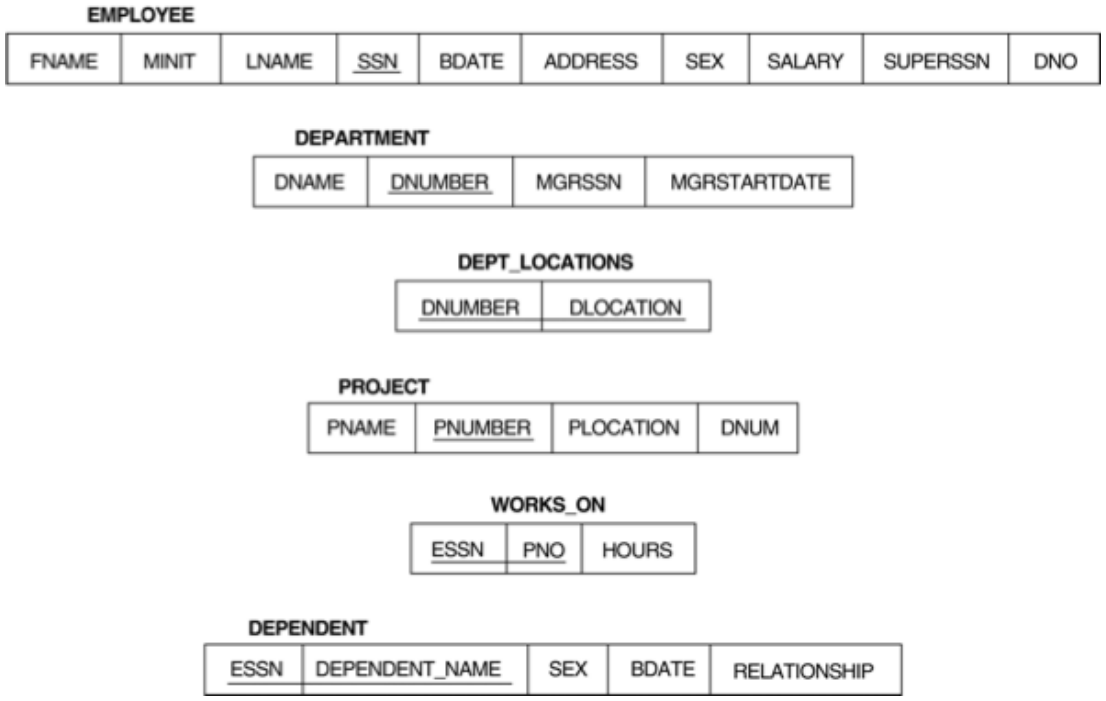
```
ALTER TABLE EMPLOYEE DROP JOB;
```

- Need to specify the FOREIGN KEY to DROP a constraint.

```
ALTER TABLE DEPARTMENT DROP FOREIGN KEY (MGRSSN);
```

COMPANY Relational Database Schema

We've talked about how to define a schema



COMPANY Populated Database (State)

Now how do we populate / retrieve from a database?

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Retrieval Queries in SQL — Bags vs. Sets

- A **bag** or **multi-set** is like a set, but an element may appear more than once.

{A, B, C, A} is a *bag*.

{A, B, C} is also a bag that also is a *set*.

- Bags also resemble lists, but order is irrelevant in a bag.

{A, B, A} = {B, A, A} as *bags*

However, **[A, B, A] =/= [B, A, A]** as *lists*

- SQL can enforce sets with Key Constraints and DISTINCT
- Ordered relations (lists) with ORDER BY

Retrieval Queries in SQL — SELECT

- Basic form of the SQL SELECT statement is called a *mapping* or a SELECT-FROM-WHERE *block*

SELECT <attribute list>
FROM <table list>
WHERE <condition>

- <attribute list> is a list of **attribute names** whose **values** are to be retrieved by the query
- <table list> is a list of the **relation names** required to process the query
- <condition> is a **conditional** (boolean) expression that identifies the tuples to be retrieved by the query

Retrieval Queries in SQL — WHERE conditions

WHERE <condition>

Basic Logical Operators in the <condition>

- = (equals)
- <, <=
- >, >=
- <>, != (not equals)
- AND
- OR
- NOT
- IS NULL, IS NOT NULL

WHERE DNO = 5

WHERE (DNO = 5) AND (PNUMBER > 1)

Simple SQL Queries and Relational Algebra

- Basic SQL queries correspond to using the following operations of the relational algebra
    - (discussed in weeks 6 and 7):
    - SELECT operator
    - PROJECT operator
    - JOIN (several types of operators)

- Rest of lecture's examples use the COMPANY database

Simple Query on One Relation

Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
SELECT BDATE, ADDRESS
FROM       EMPLOYEE
WHERE      FNAME='John' AND MINIT='B'
       AND LNAME='Smith'
```

- SELECT-clause specifies the ***projection attributes***
- WHERE-clause specifies the ***selection condition***

- Similar to the SELECT-PROJECT relation algebra operation
  - But, query result may contain duplicate tuples

Simple Query on One Relation

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

```
SELECT  BDATE, ADDRESS
FROM        EMPLOYEE
WHERE       FNAME='John' AND MINIT='B'
        AND LNAME='Smith'
```

Result

BDATE	ADDRESS
09/01/1965	731 Fondren, Houston TX

Simple Query with More than 1 Relation

Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT   FNAME, LNAME, ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNAME='Research' AND DNUMBER=DNO
```

- (DNAME='Research') is a ***selection condition*** *(*corresponds to the SELECT operation in relational algebra)
- (DNUMBER=DNO) is a **join condition** (corresponds to a JOIN operation in relational algebra, which will be discussed in weeks 6,7)

Simple Query with More than 1 Relation

```
SELECT   FNAME, LNAME, ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNAME='Research' AND DNUMBER=DNO
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

Result

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren, Houston TX
Franklin	Wong	638 Voss, Houston TX
Joyce	English	5631 Rice, Houston TX
Ramesh	Narayan	975 Fire Oak, Humble TX

The Effect of the Join Condition

What happens if the condition DNUMBER=DNO is omitted?

```
SELECT   FNAME, LNAME, ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNAME='Research' AND DNUMBER=DNO
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

The Effect of the Join Condition

```
SELECT   FNAME, LNAME, ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNAME='Research' AND DNUMBER=DNO
```

Result: All rows selected

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren, Houston TX
Franklin	Wong	638 Voss, Houston TX
Joyce	English	5631 Rice, Houston TX
Ramesh	Narayan	975 Fire Oak, Humble TX
James	Borg	450 Stone, Houston TX
Jennifer	Wallace	291 Berry, Bellaire TX
Ahmad	Jabbar	980 Dallas, Houston TX
Alicia	Zelaya	3321 Castle, Spring TX

SELECT FROM More Relations!

For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate

```
SELECT      PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM              PROJECT, DEPARTMENT, EMPLOYEE
WHERE             DNUM=DNUMBER AND MGRSSN=SSN
                            AND PLOCATION='Stafford'
```

Two Join Conditions:

- The join condition **DNUM=DNUMBER** relates a project to its controlling department
- The join condition **MGRSSN=SSN** relates the controlling department to the employee who manages that department

SELECT FROM More Relations!

```
SELECT PNUMBER, DNUM,
  LNAME, ADDRESS, BDATE
FROM  PROJECT, DEPARTMENT,
      EMPLOYEE
WHERE DNUM=DNUMBER AND
      MGRSSN=SSN AND
      PLOCATION='Stafford'
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

PNUMBER	DNUM	LNAME	ADDRESS	BDATE
10	4	Wallace	291 Berry, Bellaire TX	20/06/1941
30	4	Wallace	291 Berry, Bellaire TX	20/06/1941

Qualification of Relation Names

- In SQL, we can use the **same name** for multiple attributes as long as the attributes are in ***different relations***
- If two or more attributes in different relations have the same name, we need to specify them by the **relation name**
- We can *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name.

E.g. Unique attribute names

```
SELECT   EMPLOYEE.FNAME, EMPLOYEE.LNAME,
         EMPLOYEE.ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DEPARTMENT.DNAME='Research' AND
         DEPARTMENT.DNUMBER=EMPLOYEE.DNO
```

Aliases

- Some queries need to refer to the same relation twice
- *Aliases* can be given to the relation names

For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN=S.SSN
```

- Alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
- Think of E and S as two different *copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

Aliases

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

```
SELECT E.FNAME AS EMPLOYEE_FNAME ,
     E.LNAME  AS EMPLOYEE_LNAME ,
     S.FNAME AS SUPER_FNAME,
     S.LNAME AS SUPER_LNAME
FROM EMPLOYEE E, EMPLOYEE S
WHERE E.SUPERSSN=S.SSN
```

EMPLOYEE_FNAME	EMPLOYEE_LNAME	SUPER_FNAME	SUPER_LNAME
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Joyce	English	Franklin	Wong
Ramesh	Narayan	Franklin	Wong
Jennifer	Wallace	James	Borg
Ahmad	Jabbar	Jennifer	Wallace
Alicia	Zelaya	Jennifer	Wallace

Unspecified WHERE-clause

- A *missing WHERE-clause* indicates no condition — all tuples of the relations in the FROM-clause are selected
    - Equivalent to the condition WHERE TRUE

Retrieve the SSN values for all employees,

```
SELECT   SSN
FROM     EMPLOYEE
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Result

SSN
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

Cartesian product

- If more than one relation is specified in the FROM-clause and there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

```
SELECT    SSN, DNAME
FROM          EMPLOYEE, DEPARTMENT
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

Result

SSN	DNAME
888665555	Administration
888665555	Headquarters
888665555	Research
987654321	Administration
987654321	Headquarters
987654321	Research
987987987	Administration
987987987	Headquarters
987987987	Research
999887777	Administration

Be careful: Easy to get LARGE relations as a result!

Use of *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attrib*

```
SELECT      *
FROM        EMPLOYEE
WHERE       DNO=5
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Result

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09/01/1965	731 Fondren, Houston TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08/12/1965	638 Voss, Houston TX	M	40000	888665555	5
Joyce	A	English	453453453	31/07/1972	5631 Rice, Houston TX	F	25000	333445555	5
Ramesh	K	Narayan	666884444	15/09/1962	975 Fire Oak, Humble TX	M	38000	333445555	5

Use of DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used

```
SELECT ALL
  SALARY
FROM
     EMPLOYEE
```

SALARY
38000
43000
25000
25000

```
SELECT DISTINCT
  SALARY
FROM
     EMPLOYEE
```

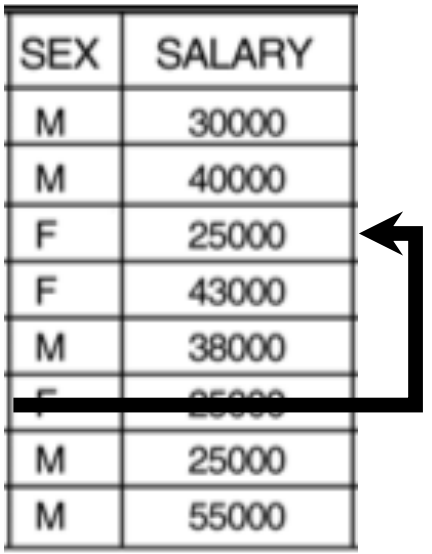
SALARY
38000
43000
25000

Use of DISTINCT

- Using DISTINCT with more than one attribute, creates a SET of the resulting tuples

```
SELECT DISTINCT SEX, SALARY
FROM      EMPLOYEE
```

Result: one duplicate row is removed

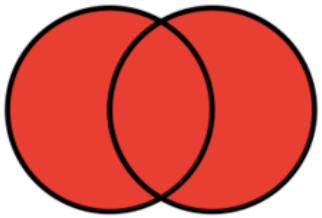


SEX	SALARY
M	30000
M	40000
F	25000
F	43000
M	38000
~~F~~	~~25000~~
M	25000
M	55000

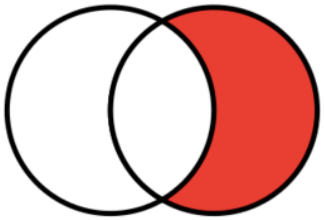
Set Operations

- SQL has directly incorporated some set operations

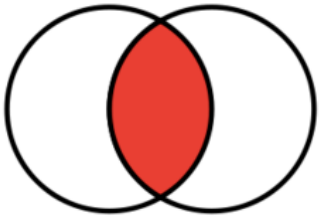
UNION
(set union)



MINUS/EXCEPT
(set difference)



INTERSECTION
(set intersection)



- Resulting relations of these set operations are sets of tuples — *duplicate tuples are eliminated from the result*
- Set operations apply only to *union compatible relations:*

1. Two relations must have the number of attributes
2. Each corresponding pair of attributes has the same domain.

Set Operations — UNION

Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
(SELECT  PNUMBER
 FROM    PROJECT, DEPARTMENT, EMPLOYEE
 WHERE   DNUM=DNUMBER AND MGRSSN=SSN
         AND LNAME='Smith')
UNION
(SELECT  PNUMBER
 FROM    PROJECT, WORKS_ON, EMPLOYEE
 WHERE   PNUMBER=PNO AND
         ESSN=SSN AND LNAME='Smith')
```

Substring Comparison / Pattern Matching

- The **LIKE** comparison operator is used to compare partial strings
- Two reserved characters are used:
  - '**%**' (or '*****' in some implementations) replaces an arbitrary number of characters
  - '**_**' replaces a single arbitrary character
- Usually use an escape-charater '\' to specify these reserve characters in your search string
  - `LIKE '%15\%%'`

Substring Comparison / Pattern Matching

Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX‘ in it.

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     ADDRESS LIKE '%Houston,TX%'
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Substring Comparison / Pattern Matching

Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX‘ in it.

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     ADDRESS LIKE '%Houston,TX%'
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Substring Comparison / Pattern Matching

Retrieve all employees who were born during the 1950s.

```
SELECT FNAME, LNAME
FROM   EMPLOYEE
WHERE  BDATE LIKE '__5_______'
```

'5' must be the third character of the string

(according to our format for date)

BDATE value is '__5_______', with each underscore as a place holder for a single arbitrary character.

- **LIKE** operator different from the formal relational model which considers each attribute value as atomic and indivisible

  In SQL, character string attribute values are not atomic

Substring Comparison / Pattern Matching

Retrieve all employees who were born during the 1950s.

```
SELECT FNAME, LNAME
FROM   EMPLOYEE
WHERE  BDATE LIKE '__5_______'
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	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
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Arithmetic Operations

- The standard arithmetic operators **'+', '-'. '*', and '/'** (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result

Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
SELECT FNAME, LNAME, 1.1*SALARY
FROM   EMPLOYEE, WORKS_ON, PROJECT
WHERE  SSN=ESSN AND PNO=PNUMBER AND
       PNAME='ProductX’
```

ORDER BY

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)

Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
SELECT  DNAME, LNAME, FNAME, PNAME
FROM    DEPARTMENT, EMPLOYEE,
        WORKS_ON, PROJECT
WHERE   DNUMBER=DNO AND SSN=ESSN
        AND PNO=PNUMBER
ORDER BY DNAME, LNAME
```

ORDER BY

- The default order is in **ascending** order of values
- We can specify the keyword **DESC** if we want a descending order; the keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

```
SELECT  DNAME, LNAME, FNAME, PNAME
FROM    DEPARTMENT, EMPLOYEE,
        WORKS_ON, PROJECT
WHERE   DNUMBER=DNO AND SSN=ESSN
        AND PNO=PNUMBER
ORDER BY DNAME DESC, LNAME ASC
```

Summary of SQL SELECT Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory.
- Here is what we have covered with SELECT:

```
SELECT   <attribute list>
FROM     <table list>
[WHERE <condition>]
[ORDER BY   <attribute list>]
```

- *Next week*, we will cover the rest of the components of the SELECT queries and look at more complex queries.

Summary of Important Take-aways

- Be able to **define** and **update** a schema in SQL
  - Using CREATE, ALTER, DROP
  - Primitive Data Types in SQL
  - Express Integrity Constraints
- Construct **SELECT** Queries
  - Using one or more relations
  - WHERE-clause and conditions
  - Simple arithmetic in queries
  - LIKE and sub-string queries
- Express Sets SQL using **DISTINCT**, and **UNION**
- Use **ORDER BY** to create ordered (lists) relations