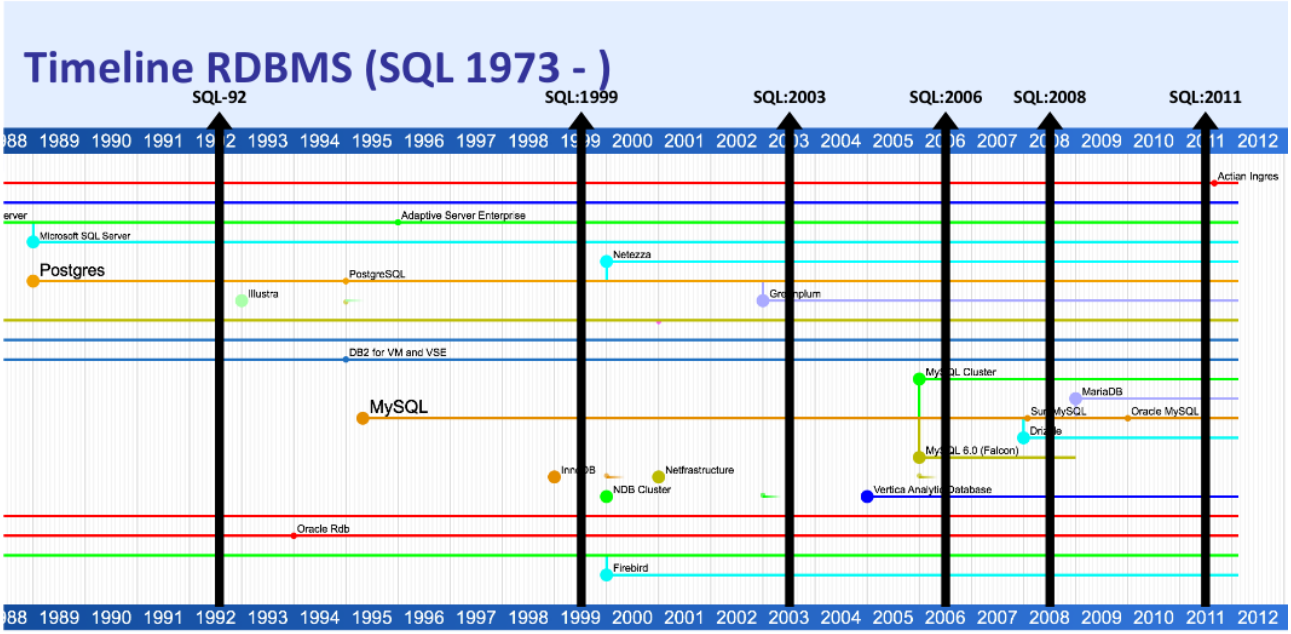
STRUCTURED QUERY LANGUAGE

This was originally named SEQUEL (Structured English QUEry Language)

- Designed and implemented by IBM Research as an 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 and updates (both DDL and DML)
- SQL can also specify authorization and security, define integrity constraints, define views, specify transaction controls.



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 AND 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

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 a synchronous concept.
- Comments:

```
--line comments are two dashes
```

So to create the table DEPARTMENT:

```
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 its attributes.
- In SQL, attributes are ordered based on the order they are specified.
- Attributes can have initial constraints defined, as in NOT NULL

ATTRIBUTE NAMES AND TABLE NAMES RESERVED WORDS IN SQL

- Vendor specific… and can be a long list…
- How to deal with reserved words? 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)

ATTRIBUTE DATA TYPES

INTEGER

- 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

Example: MySQL Implementation of Integer Types

CORE SQL

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

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: DECMIAL(7, 4) will look like -999.9999 when displayed

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

BINARY DATA

- BIT(n) – Fixed length
- BIT VARYING(n) – Varying length
- BLOB – Binary Large. Megabyte, Gigabyte.

Typically, people do not store files (images etc.) or documents in a RDBS. They are stored on a File System with reference.

PRO TIP: Do *hash* your binary data and store that in the DB with the metadata (Use “md5 hash” of your binary data).

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 as 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…

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

DATE AND TIME

- DATE – Made up of year-month-day (“yyyy-mm-dd”)
- TIME – Made up of hour:minute:second (“hh:mm:ss”)
- TIME(i) – TIME plus I addition digits for fractions of second
  - TIME(3) → precision for milliseconds
- DATETIME/TIMESTAMP – both DATE and TIME components

INTERVAL

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

COMPLEX TYPES

- Special types:
  - CURRENCY, MONEY
- Spatial Types (GIS)
  - GEOMETRY type can store gemoetries
    - 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

CREATING DOMAINS

- CREATE DOMAIN – Allows you to specify your own data type to use in schema

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

- 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
```

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

KEY AND REFERENTIAL INTEGRITY CONSTRAINTS

- PRIMARY KEY clause

```
DNUMBER INT PRIMARY KEY;
```

- UNIQUE clause – for secondary/alternate keys

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

- FOREIGN KEY clause – for referential integrity

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

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

COMPOSITE PRIMARY AND FOREIGN KEYS

Composite PRIMARY or FOREIGN KEYS are specified after the attributes

```
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
);
```

REFERENTIAL INTEGRITY OPTIONS

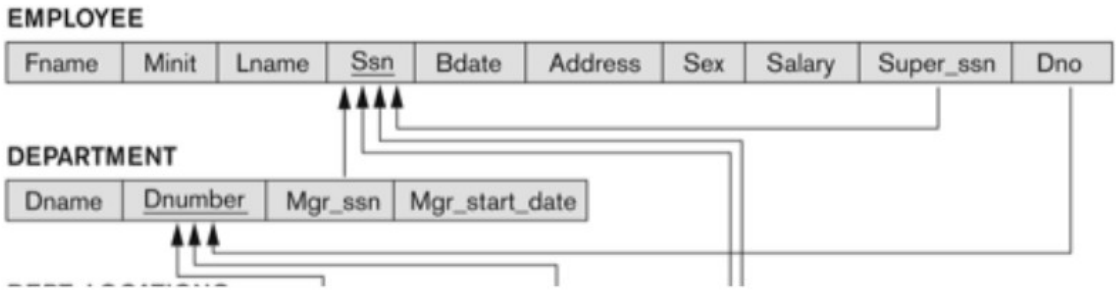
- A referential integrity constraint may be violated when tuples in the referenced tuple are updated/deleted.
- Default: reject the operation that violates constraint
- Or: Specify a referential triggered action, options:

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

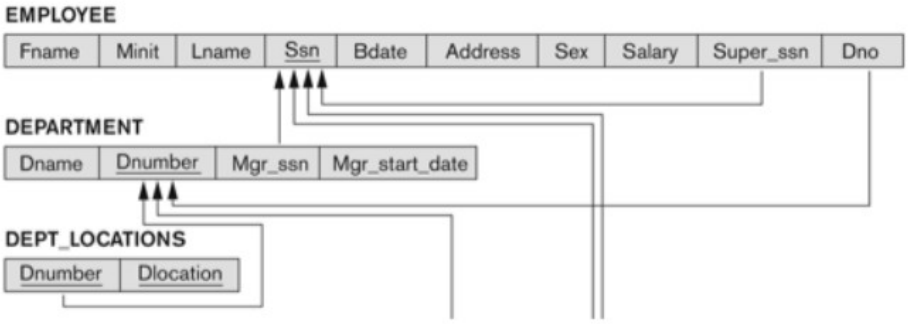
Example:

```
ON DELETE SET NULL
ON UPDATE CASCADE
```

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

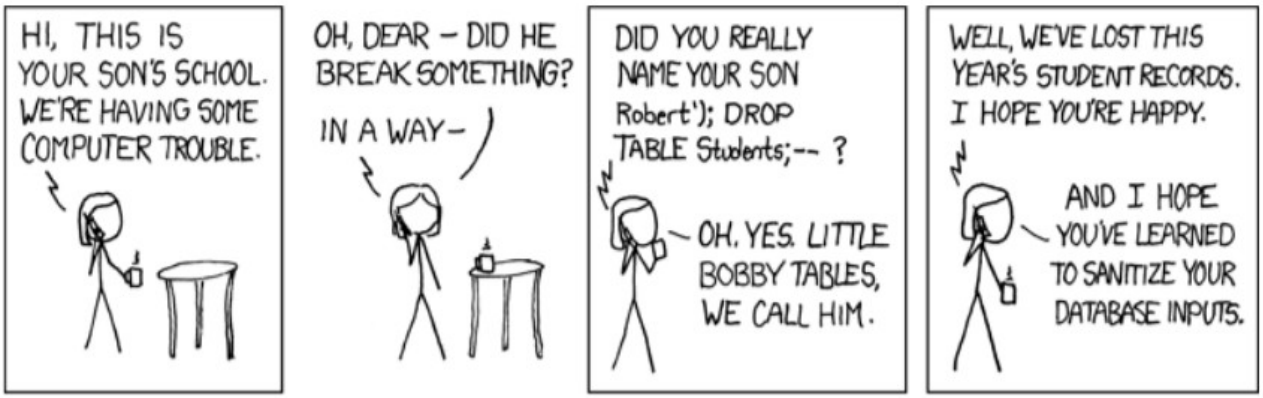


```
CREATE TABLE EMPLOYEE (
…
        FOREIGN KEY(DNO)REFERENCES DEPARTMENT(DNUMBER)ON DELETE SET
DEFAULT ON UPDATE CASCADE,
        FOREIGN KEY(SUPERSSN)REFERENCES EMPLOYEE(SSN)ON DELETE SET
NULL ON UPDATE CASCADE,
…
);
```



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



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 its 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 DEPENDENT 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.

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);
```

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);
```

DATA MANIPULATION LANGUAGE

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 a bag.

{A, B, C} is also a bag that is also 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

SELECT FROM WHERE

- 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

WHERE CONDITIONS

Basic Logical Operators in the <condition> : `=, <, <=, >, >=, <>, !=, 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
  - SELECT operator
  - PROJECT operator
  - JOIN

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"
```

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 ONE 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
```

```
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 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

All rows are 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 to a project to its controlling department.
- The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department.

```
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

DEPARTMENT

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

DEPT_LOCATIONS

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

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

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.

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

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

Result

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

USE OF *

To retrieve all values in tuples, a * is used.

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

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 values 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

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

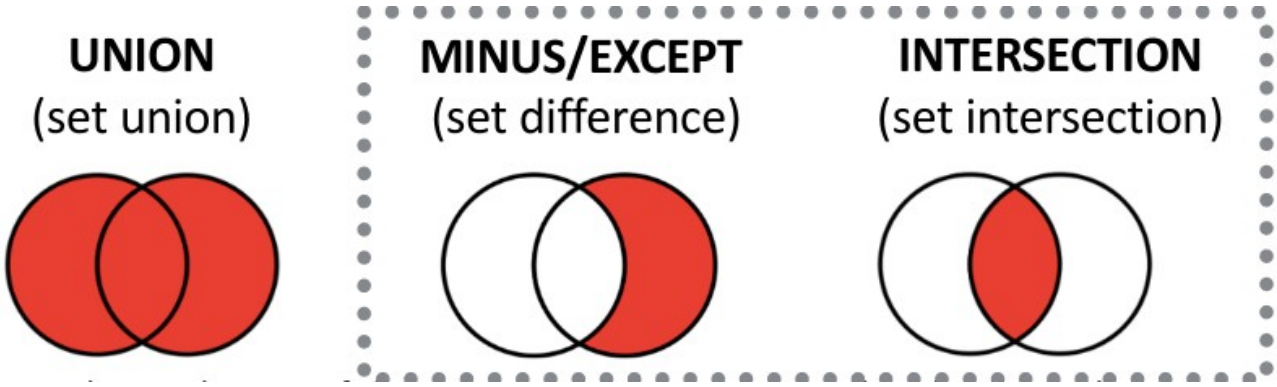
```
SELECT DISTINCT SEX, SALARY
FROM      EMPLOYEE
```

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



- 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:
  - Two relations must have the same number of attributes
  - Each corresponding pair of attributes has the same domain

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-character ‘\’ to specify these reserve characters in your search string
  - LIKE ‘%15\%%’

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

Retrieve all employees who were born during the 1950s

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE BDATE LIKE '_ _ 5 _ _ _ _ _ _'
```

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

```
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 / 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
```

- 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
```