Updated 1/11/2023

Topic 1: Data models and query languages
Unit 1: SQL
Lecture 1

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

1/10/2023

Topic 1: Data Models and Query Languages

PRELIMINARY

- **Lecture 1 (Tue 1/10):** Course introduction / T1-U1 SQL / PostgreSQL setup / SQL Activities
- **Lecture 2 (Fri 1/13):** T1-U1 SQL
- **Lecture 3 (Tue 1/17):** T1-U1 SQL
- **Lecture 4 (Fri 1/20):** T1-U2 Logic & Relational Calculus
- **Lecture 5 (Tue 1/24):** T1-U1 Logic & Relational Calculus
- **Lecture 6 (Fri 1/27):** T1-U3 Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 1/31):** T1-U3 Relational Algebra & Codd's Theorem
- **Lecture 8 (Fri 2/3):** T1-U4 Datalog & Recursion
- **Lecture 9 (Tue 2/7):** T1-U4 Datalog & Recursion
- **Lecture 10 (Tue 2/10):** T1-U4 Datalog & Recursion

Pointers to relevant concepts & supplementary material:

- **Unit 1. SQL**: [SAMS'12], [CS 3200], [Cow'03] Ch3 & Ch5, [Complete'08] Ch6, [Silberschatz+'20] Ch3.8
- **Unit 2. Logic & Relational Calculus**: First-Order Logic (FOL), relational calculus (RC): [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'20] Ch27.1 & Ch27.2, [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4, [Barker-Plummer+'11] Ch11
- **Unit 3. Relational Algebra & Codd's Theorem**: Relational Algebra (RA), Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'20] Ch2.6, [Alice'95] Ch4.4 & Ch5.4
- **Unit 4. Datalog & Recursion**: Datalog, recursion, Stratified Datalog with negation, Datalog evaluation strategies, Stable Model semantics, Answer Set Programming (ASP): [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [G., Suciu'10]
- **Unit 5. Alternative Data Models**: NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Outline: T1-U1: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k
  - [Recursion: moved to T1-U4: Datalog]

Structured Query Language: SQL

- Influenced by relational calculus (= First Order Logic)
- SQL is a declarative query language
  - We say what we want to get
  - We don’t say how we should get it ("separation of concerns")

SQL: Declarative Programming

SQL

```
select (e.salary / (e.age - 18)) as comp
from employee as e
where e.name = "Jones"
```

Declarative Language: you say what you want without having to say how to do it.

Procedural Language: you have to specify exact steps to get the result.

SQL: was not the only Attempt

SQL

```
select (e.salary / (e.age - 18)) as comp
from employee as e
where e.name = "Jones"
```

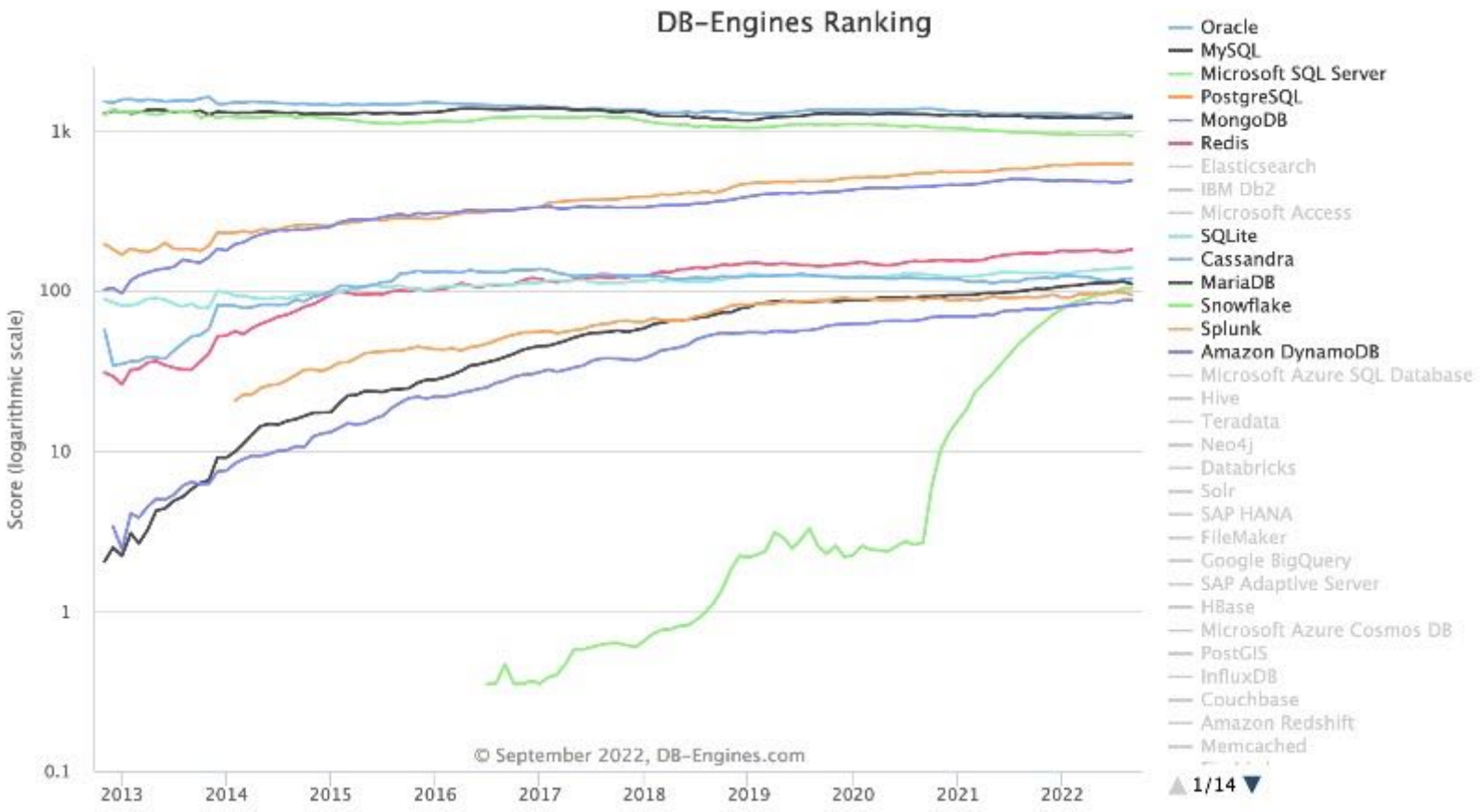
QUEL

```
range of e is employee
retrieve (comp = e.salary / (e.age - 18))
where e.name = "Jones"
```

Commercially not used anymore since ~1980

Source: http://en.wikipedia.org/wiki/QUEL_query_languages
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Why PostgreSQL instead of MariaDB (or MySQL)



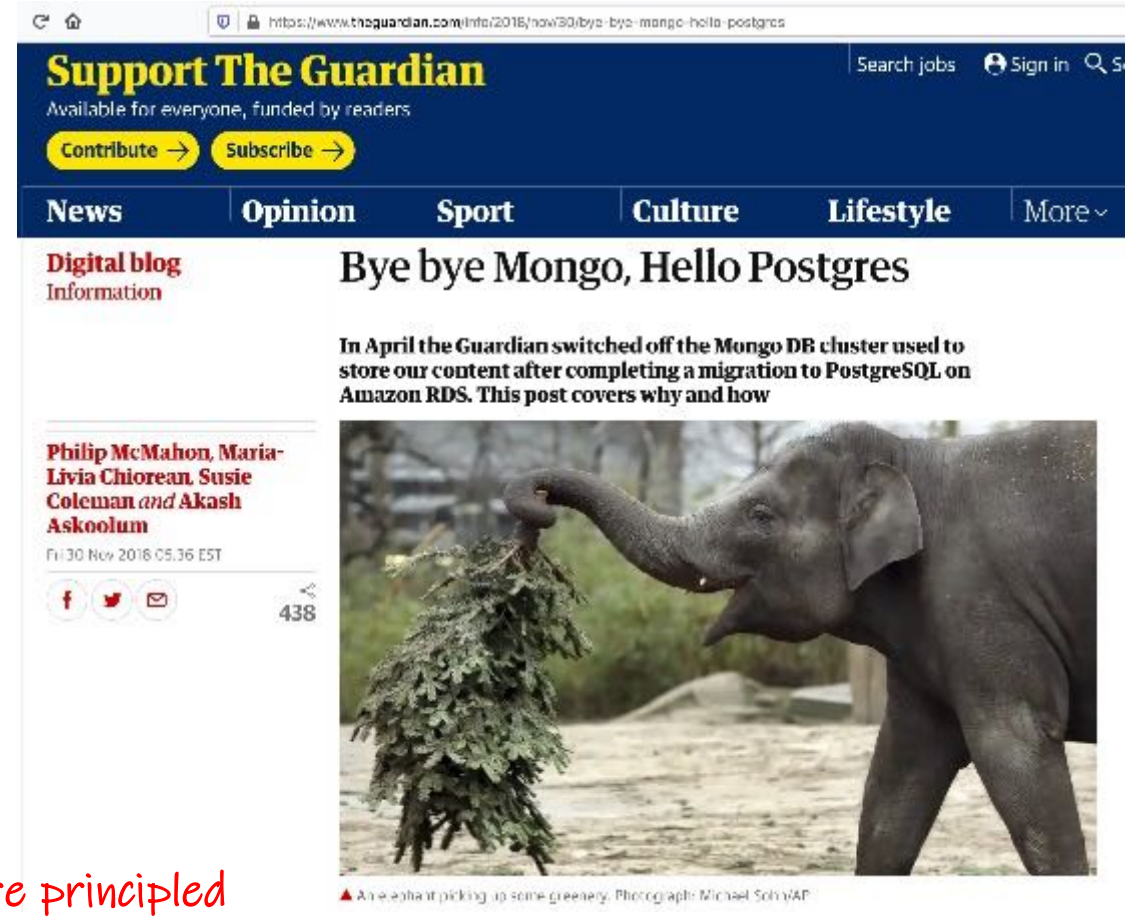
Source: https://db-engines.com/en/ranking_trend

Why PostgreSQL instead of MariaDB (or MySQL)

Although PostgreSQL has been around for a while, the relative decline of MySQL has made it a serious contender for the title of most used open source database. Since it works very similarly to MySQL, developers who prefer open source software are converting in droves.

Advantages

- By far, PostgreSQL's most mentioned advantage is the efficiency of its central algorithm, which means it outperforms many databases that are advertised as more advanced. This is especially useful if you are working with large datasets, for which I/O processes can otherwise become a bottleneck.
- It is also one of the most flexible open source databases around; you can write functions in a wide range of server-side languages: Python, Perl, Java, Ruby, C, and R.
- As one of the most commonly used open source databases, PostgreSQL's community support is some of the best around.



Support The Guardian
Available for everyone, funded by readers
Contribute → Subscribe →
Search jobs Sign in
News Opinion Sport Culture Lifestyle More

Digital blog
Information

Bye bye Mongo, Hello Postgres

In April the Guardian switched off the Mongo DB cluster used to store our content after completing a migration to PostgreSQL on Amazon RDS. This post covers why and how

Philip McMahon, Maria-Livia Chiorean, Susie Coleman and Akash Askoolum

Fri 30 Nov 2018 05.36 EST

438

I also prefer PostgreSQL over MySQL because it has a more principled interpretation of SQL (and a powerful EXPLAIN command)

Source: https://db-engines.com/en/ranking_trend

Source: https://www.theguardian.com/info/2018/nov/30/bye-bye-mongo-hello-postgres

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

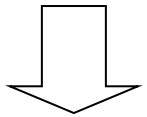
Simple SQL Query

Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```



?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Simple SQL Query

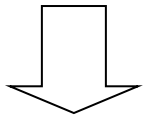
Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

302

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT pName, price
FROM   Product
WHERE  price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Selection
& Projection

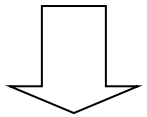
SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Selection vs. Projection

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Where does the selection happen?

?

Selection & Projection

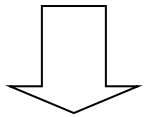
SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Selection vs. Projection

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

One **selects** certain entires=tuples (rows)
-> happens in the **WHERE** clause
-> acts like a **filter**

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Selection vs. Projection

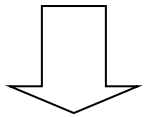
Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

One **projects** onto some attributes (columns)
-> happens in the **SELECT** clause

One **selects** certain entires=tuples (rows)
-> happens in the **WHERE** clause
-> acts like a **filter**

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Eliminating Duplicates


Our colorful hands represent "team exercises"

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT category
FROM   Product
```

⇨ ?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

Set vs. Bag semantics

Think of a dictionary: keys mapping to # of occurences

Gadgets : 2
Photography : 1
Houshold : 1

?

Category
Gadgets
Photography
Household

underlying set also called the "support" of the bag

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

Set vs. Bag semantics

Think of a dictionary: keys mapping to # of occurences

Gadgets : 2
Photography : 1
Houshold : 1

```
SELECT DISTINCT category
FROM   Product
```

Category
Gadgets
Photography
Household

underlying set also called the "support" of the bag

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Outline: T1-U1: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k
  - [Recursion: moved to T1-U4: Datalog]

Keys and Foreign Keys

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is here a key vs. a foreign key?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Keys and Foreign Keys

Key

Foreign key

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Key

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Keys and foreign keys are special cases of more general constraints. Which?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Keys and Foreign Keys

In the following, $R(U)$ denotes the schema of a relation with name R and set of attributes U.

Functional Dependencies

A *functional dependency (FD)* on relations of schema $R(U)$ is an expression of the form

$$R : X \to Y, \qquad (1)$$

where $X \subseteq U$ and $Y \subseteq U$ are subsets of R's attributes. Instance r of schema $R(U)$ is said to *satisfy* FD fd, denoted $r \models fd$, if whenever tuples $t_1 \in r$ and $t_2 \in r$ agree on all attributes in X, they also agree on all attributes in Y:

$$r \models fd \iff \text{for every } t_1, t_2 \in r \text{ if } \pi_X(t_1) = \pi_X(t_2) \text{ then } \pi_Y(t_1) = \pi_Y(t_2)$$

Here, $\pi_X(t)$ denotes the projection of tuple t on the attributes in X.

Key Dependencies

In the particular case when $Y = U$, a functional dependency of form (1) is called a *key dependency*, and the set of attributes X is a called a *key for R*.

Key

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Inclusion Dependencies

Functional and join dependencies and their special-case subclasses each pertain to single relations. The following class of dependencies can express connections between relations. An *inclusion dependency (IND)* on pairs of relations of schemas $R(U)$ and $S(V)$ (with R and S not necessarily distinct) is an expression of the form

$$R[X] \subseteq S[Y], \qquad (4)$$

where $X \subseteq U$ and $Y \subseteq V$. Inclusion dependencies are also known as *referential constraints*. Relations r and s of schemas $R(U)$, respectively $S(V)$ satisfy inclusion dependency id, denoted $r, s \models id$, if the projection of r on X is included in the projection of s on Y:

$$r, s \models id \iff \Pi_X(r) \subseteq \Pi_Y(s).$$

When R and S refer to the same relation name, then $r = s$ in the above definition of satisfaction.

Foreign key

Foreign Key Dependencies

In the particular case when Y is a key for relations of schema S (S: $Y \to V$), INDs of form (4) are called *foreign key dependencies*. Intuitively, in this case the projection on X of every tuple t in r contains the key of a tuple from the "foreign" table s.

Screenshots from: Deutsch. FOL Modeling of Integrity Constraints (Dependencies). Encyclopedia of Database Systems, 2018. https://doi.org/10.1007/978-1-4614-8265-9_980
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Keys and Foreign Keys

In the following, $R(U)$ denotes the schema of a relation with name R and set of attributes U.

Functional Dependencies
A *functional dependency (FD)* on relations of schema $R(U)$ is an expression of the form

$$R : X \to Y, \qquad (1)$$

where $X \subseteq U$ and $Y \subseteq U$ are subsets of R's attributes. Instance r of schema $R(U)$ is said to *satisfy* FD fd, denoted $r \models fd$, if whenever tuples $t_1 \in r$ and $t_2 \in r$ agree on all attributes in X, they also agree on all attributes in Y:

$$r \models fd \iff \text{for every } t_1, t_2 \in r \text{ if } \pi_X(t_1) = \pi_X(t_2) \text{ then } \pi_Y(t_1) = \pi_Y(t_2)$$

Here, $\pi_X(t)$ denotes the projection of tuple t on the attributes in X.

Key Dependencies
In the particular case when $Y = U$, a functional dependency of form (1) is called a *key dependency*, and the set of attributes X is a called a *key for R*.

R[X] functionally determines R[Y]:
Y = f(X)

R

…	X	Y	…
…	1	7	…
…	1	7	…
…	2	5	…
…	3	7	…

R[X] is included in S[Y]:
R[Y] ⊆ S[Y]

S

…	Y	…
…	1	…
…	2	…
…	2	…
…	3	…
…	4	…

Inclusion Dependencies
Functional and join dependencies and their special-case subclasses each pertain to single relations. The following class of dependencies can express connections between relations. An *inclusion dependency (IND)* on pairs of relations of schemas $R(U)$ and $S(V)$ (with R and S not necessarily distinct) is an expression of the form

$$R[X] \subseteq S[Y], \qquad (4)$$

where $X \subseteq U$ and $Y \subseteq V$. Inclusion dependencies are also known as *referential constraints*. Relations r and s of schemas $R(U)$, respectively $S(V)$ satisfy inclusion dependency id, denoted $r, s \models id$, if the projection of r on X is included in the projection of s on Y:

$$r, s \Big| = id \iff \Pi_X(r) \subseteq \Pi_Y(s).$$

When R and S refer to the same relation name, then $r = s$ in the above definition of satisfaction.

Foreign Key Dependencies
In the particular case when Y is a key for relations of schema S (S: $Y \to V$), INDs of form (4) are called *foreign key dependencies*. Intuitively, in this case the projection on X of every tuple t in r contains the key of a tuple from the "foreign" table s.

Screenshots from: Deutsch. FOL Modeling of Integrity Constraints (Dependencies). Encyclopedia of Database Systems, 2018. https://doi.org/10.1007/978-1-4614-8265-9_980
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

Foreign key: attribute in a relational table that matches a candidate key of another table

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

?

Foreign key: attribute in a relational table that matches a candidate key of another table

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

?

SuperTouch	$249.99	Computer	NewCom

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

tuple violates foreign key constraint

SuperTouch	$249.99	Computer	NewCom

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

tuple violates foreign key constraint

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

SuperTouch	$249.99	Computer	NewCom

```
Delete from Company
where CName = 'Canon';
```

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

SuperTouch	$249.99	Computer	NewCom

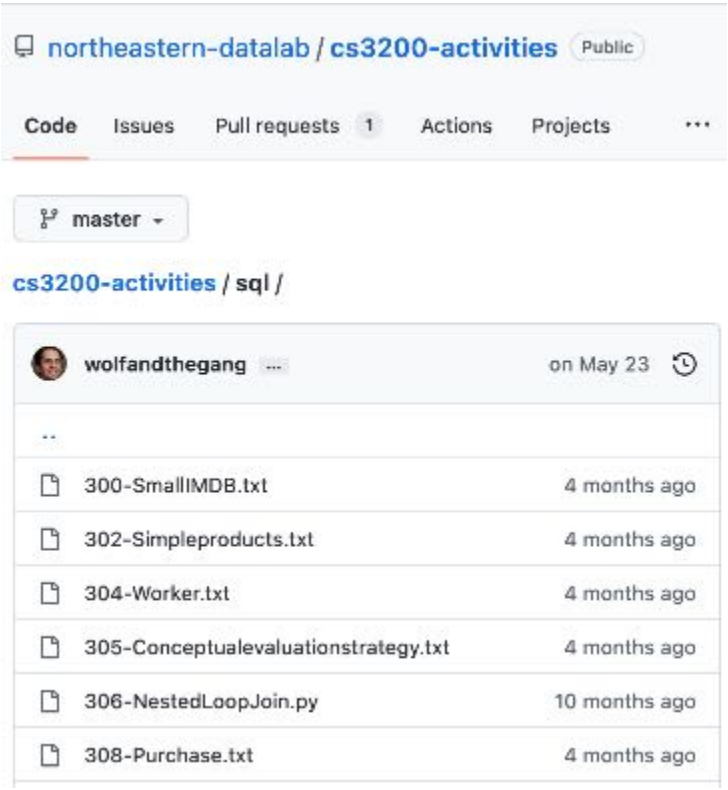
tuple violates foreign key constraint

```
Delete from Company
where CName = 'Canon';
```

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Schema specification in SQL




```
----------------------------
-- Create the tables
----------------------------

create table Company (
      CName char(20) PRIMARY KEY,
      StockPrice int,
      Country char(20) );

create table Product (
      PName char(20),
      Price decimal(9, 2),
      Category char(20),
      Manufacturer char(20),
PRIMARY KEY (PName),
FOREIGN KEY (Manufacturer) REFERENCES Company(CName) );



----------------------------
-- Populate the tables
----------------------------

insert into Company values ('GizmoWorks', 25, 'USA');
insert into Company values ('Canon', 65, 'Japan');
insert into Company values ('Hitachi', 15, 'Japan');

insert into Product values ('Gizmo', 19.99, 'Gadgets', 'GizmoWorks');
insert into Product values ('PowerGizmo', 29.99, 'Gadgets', 'GizmoWorks');
```

Example SQL code and database: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Outline: T1-U1: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k
  - [Recursion: moved to T1-U4: Datalog]

Joins

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Joins

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

```
SELECT   pName, price
FROM     Product, Company
WHERE    manufacturer=cName
   and   country='Japan'
   and   price <= 200
```

Join b/w Product and Company

PName	Price
SingleTouch	$149.99

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql