CS186 Discussion 09

(Distributed Data, Data Science)

Matthew Deng

Distributed Data

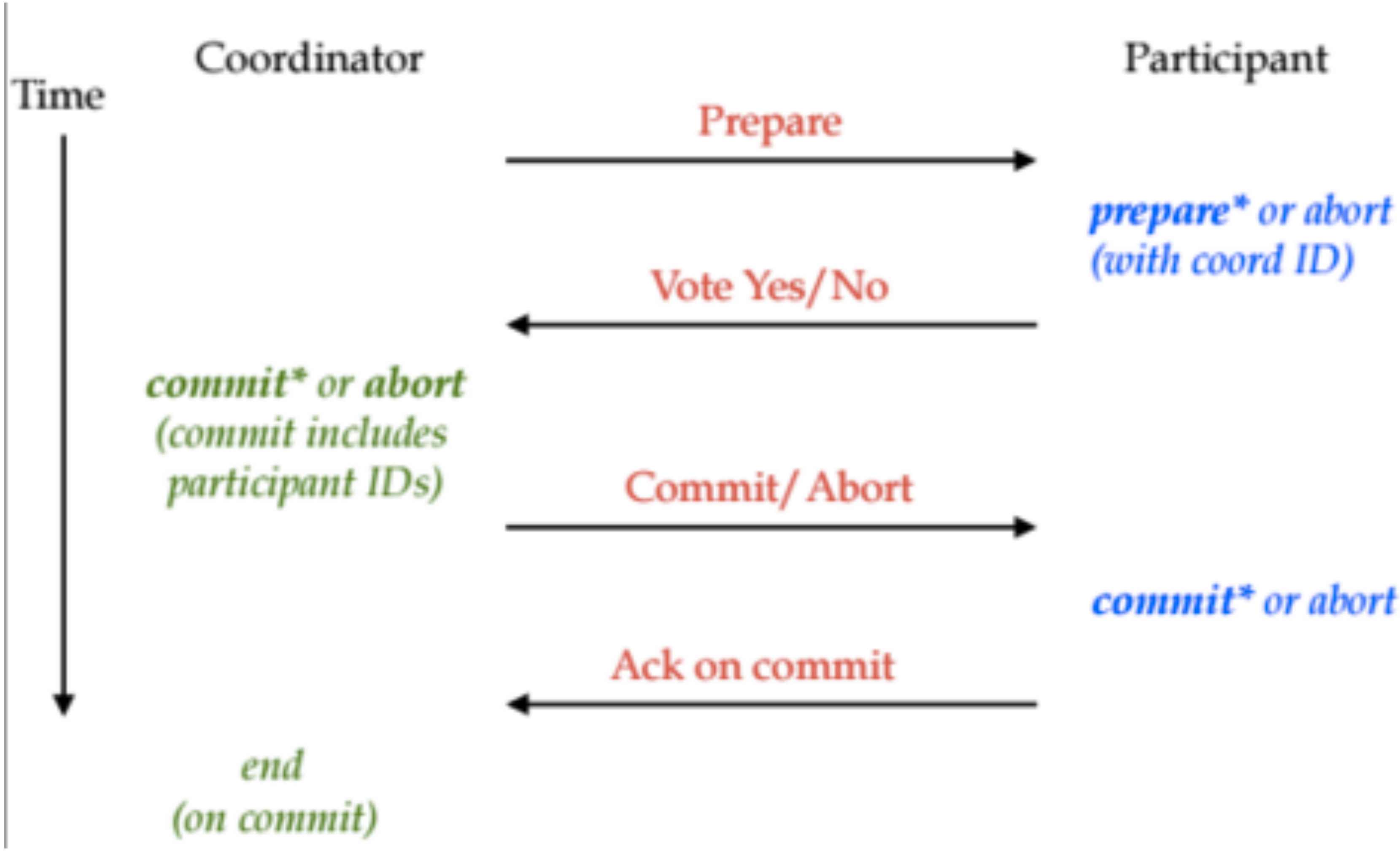
Partitioned Data

- Data is partitioned across nodes
- One copy of each record

2-Phase Commit

- Phase 1:PREPARE
  - Coordinator: Prepare
  - Participatnt: Yes/No
- Phase 2: COMMIT/ABORT
  - Coordinator: Commit/abort
  - Participant: Ack

2-Phase Commit



Coordinator

Participant

Time

Prepare

prepare* *or abort*
(with coord ID)

Vote Yes/No

commit* *or* ***abort***
(commit includes participant IDs)

Commit/ Abort

commit* *or abort*

Ack on commit

end
(on commit)

Replicated Data

- Increases availability
- Reduces latency
- Load balancing

Single-Master vs. Multi-Master

- Single-Master
  - Every data item has one master node
- Multi-Master
  - Anyone can write a data item

Quorums

- Replicate each item on N nodes
- Write to W nodes
- Read from R nodes

$$W+R > N$$

NoSQL

- Key/Value Stores/ “Document” Stores
- Replicated Data
- Multi-master

Eventual Consistency

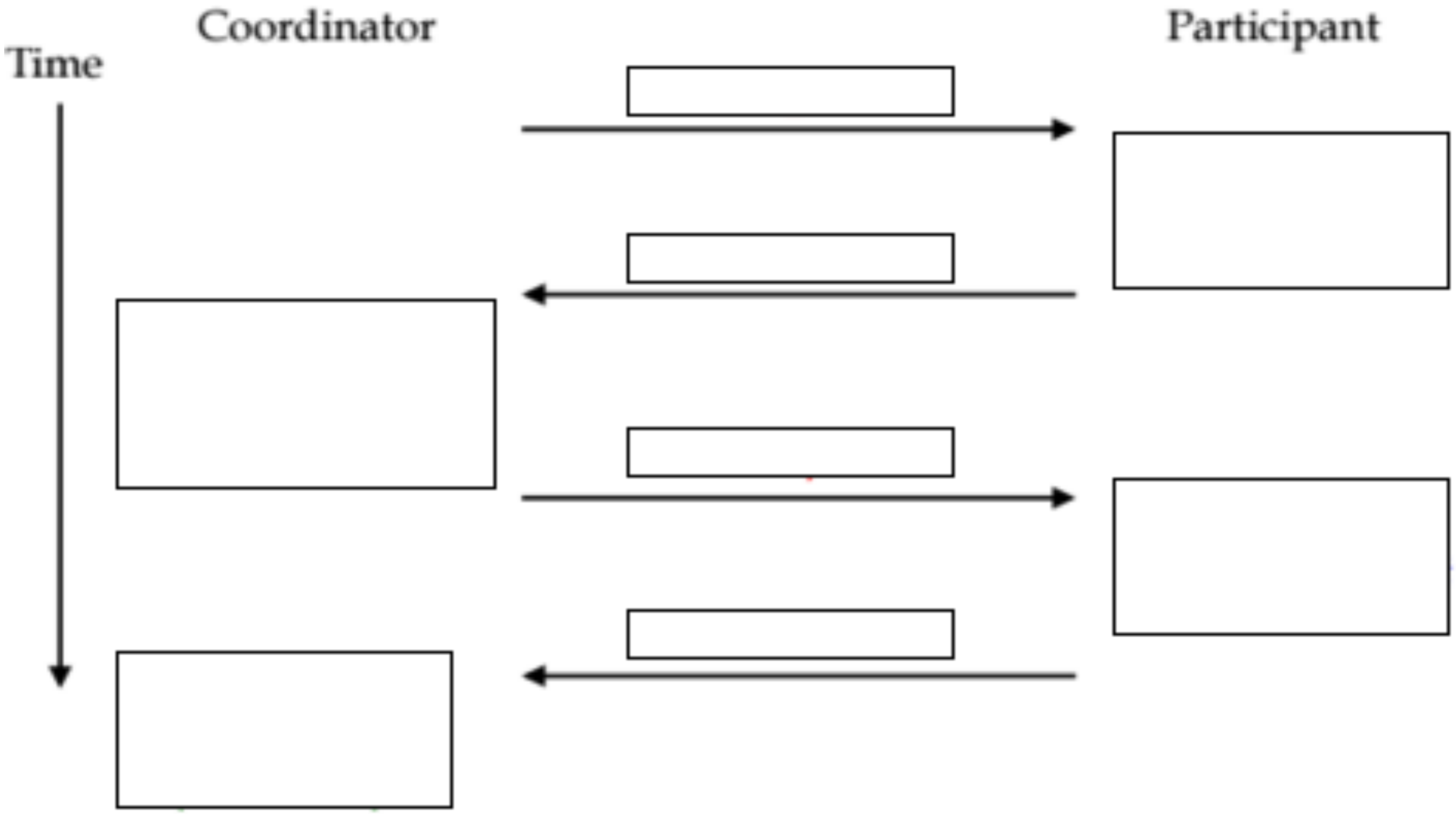
- Safety
  - – Nothing bad ever happens
- Liveness
  - – A good thing eventually happens

Monotonic Code

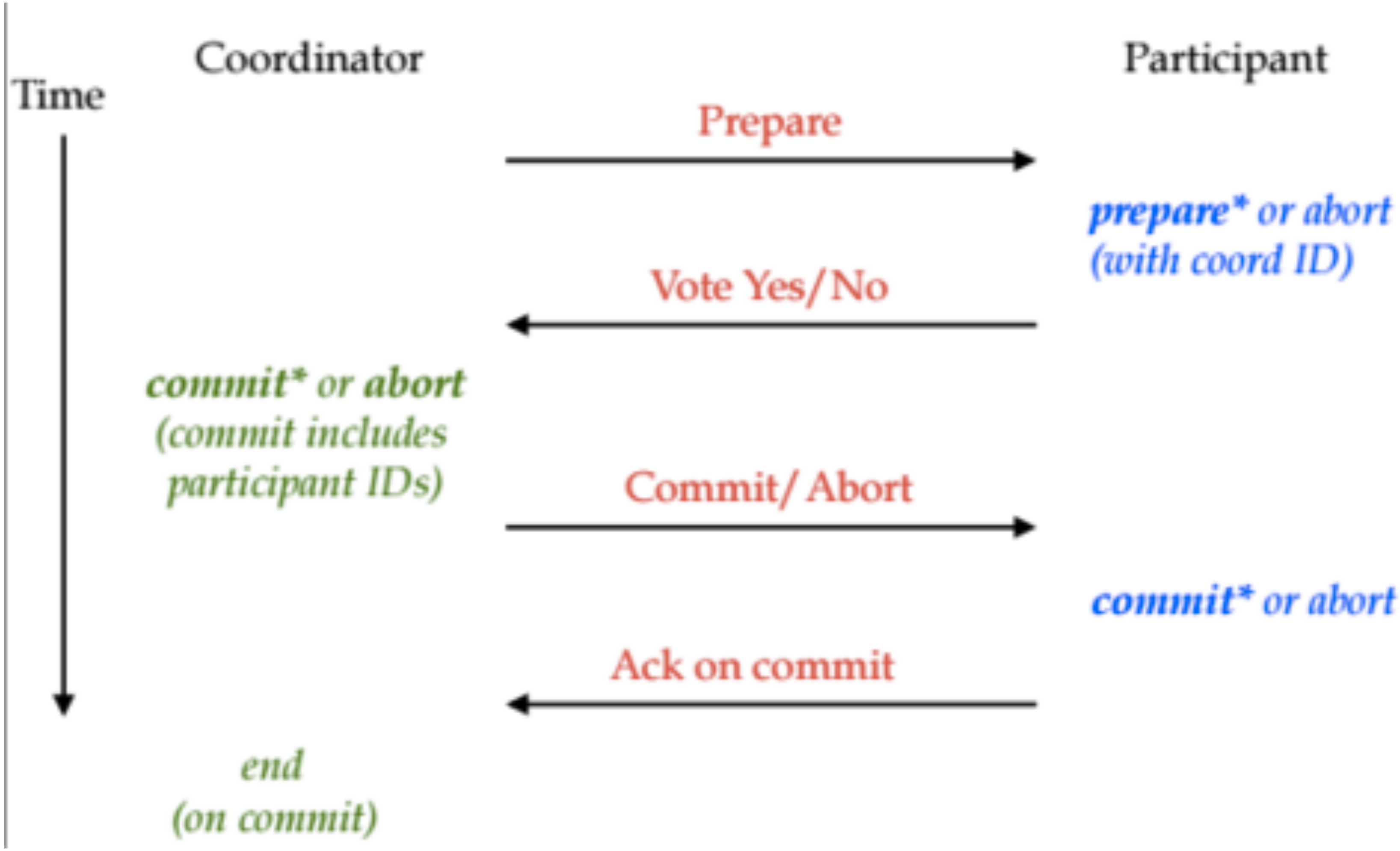
- Sets grow bigger
  - UNION
- Counters go up
  - MAX
- Booleans go from false to true
  - OR

2PC + ARIES Worksheet

2PC + ARIES Exercises



2PC + ARIES Exercises



2PC + ARIES Exercises

1. In a distributed commit protocol, what new log record types are needed to support Two-Phase Commit with ARIES?

2PC + ARIES Exercises

1. In a distributed commit protocol, what new log record types are needed to support Two-Phase Commit with ARIES?

   PREPARE

2PC + ARIES Exercises

2. What happens when the Coordinator crashes before all participants ACK and after logging COMMIT?

2PC + ARIES Exercises

2. What happens when the Coordinator crashes before all participants ACK and after logging COMMIT?

   1. Coordinator will restart
   2. Coordinator will check the last entry in its log - “COMMIT”
   3. Coordinator must periodically resend— because there may be other link or site failures in the system — a commit or abort message to each subordinate until we receive an ack.
   4. After we have received acks from all subordinates, we write an end log record for .

Eventual Consistency Worksheet

Eventual Consistency Exercises

1. You are designing a version of GitHub where file updates are stored in a geo-distributed NoSQL store, that is eventually consistent. Why might we choose NoSQL over a RDBMS for our application?

Eventual Consistency Exercises

1. You are designing a version of GitHub where file updates are stored in a geo-distributed NoSQL store, that is eventually consistent. Why might we choose NoSQL over a RDBMS for our application?

    - NoSQL allows for faster writer latency since updates are only ACKed by one node
    - Highly available since it is horizontal scaling
    - Con: consistency requirements need to be added at the application layer

Eventual Consistency Exercises

2. Say Alice and Bob are project partners. Alice makes a commit on top of the skeleton code on her computer and pushes her changes. Bob does the same from his computer. What do you think should happen when Alice/ Bob/their TA pulls their code?

Eventual Consistency Exercises

2. Say Alice and Bob are project partners. Alice makes a commit on top of the skeleton code on her computer and pushes her changes. Bob does the same from his computer. What do you think should happen when Alice/Bob/their TA pulls their code?

   - You could use a “last writer wins” (or first writer) → pulling may overwrite your commit history
   - When the distributed nodes are gossiping updates, they could merge together the code (using gits merging algorithm). This means pulling could retrieve code with merge conflicts from the server.
   - You could do 2PC at the application layer on each push, to give replica consistency guarantees.

Data Science

OTLP & OLAP

- Online Transaction Processing
- Online Analytics Processing

Data Warehouse

- Extract
  - Collect data from multiple sources
- Transform (Clean)
  - Data validation and filtering
  - Schema manipulation
  - Data normalization
- Load
  - Bulk load

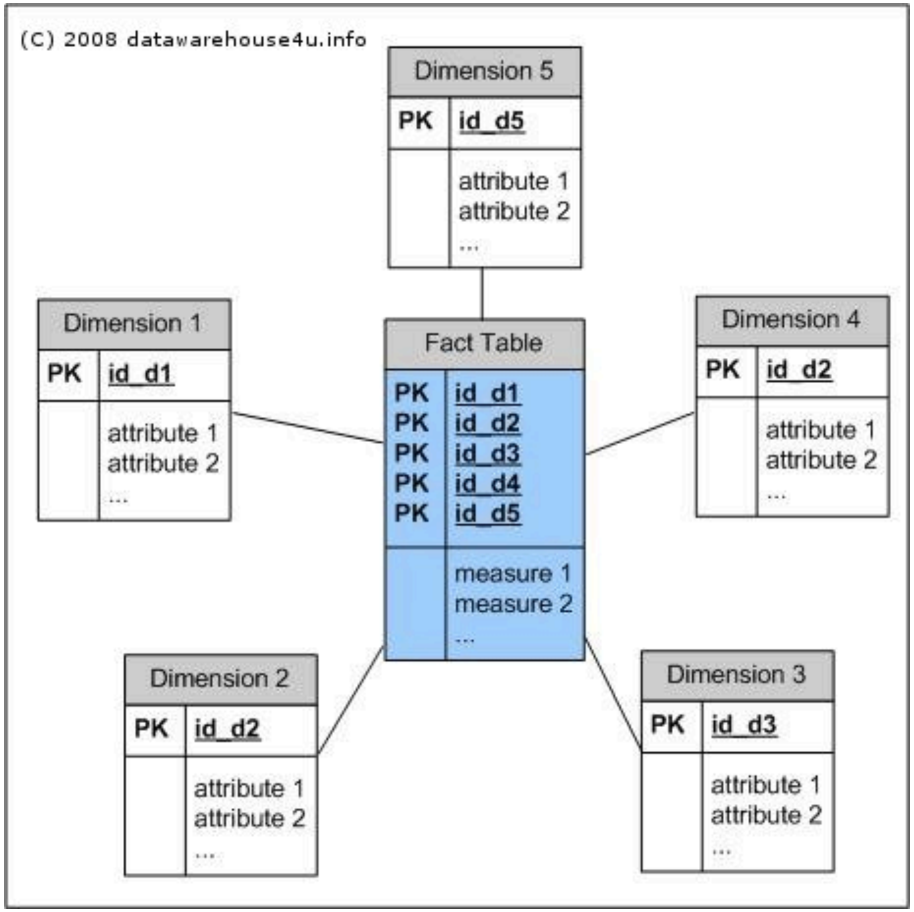
Data Lake

- Like data warehouse, but without ETL
  - Save in raw form

- Beware of data swamp
  - Dirty data

Multidimensional Data

- Multidimensional cube of data
- StarSchema



(C) 2008 datawarehouse4u.info

Dimension 5	
PK	id_d5
	attribute 1 attribute 2 ...

Dimension 1	
PK	id_d1
	attribute 1 attribute 2 ...

Fact Table	
PK PK PK PK PK	id_d1 id_d2 id_d3 id_d4 id_d5
	measure 1 measure 2 ...

Dimension 4	
PK	id_d2
	attribute 1 attribute 2 ...

Dimension 2	
PK	id_d2
	attribute 1 attribute 2 ...

Dimension 3	
PK	id_d3
	attribute 1 attribute 2 ...

Cross Tabulation

- Aggregate data across pair of dimensions
- GROUP BY
- Pivot Tables
- Cube Operator

X	Y	Value

	Y1	Y2	Total
X1			
X2			
X3			
Total			

OLAP Queries

- Slicing
  - Select a value for 1 dimension
- Dicing
  - Select a range of values for multiple dimensions
- Rollup
  - Aggregate along 1 dimension
- Drill-Down
  - De-aggregate along 1 dimension

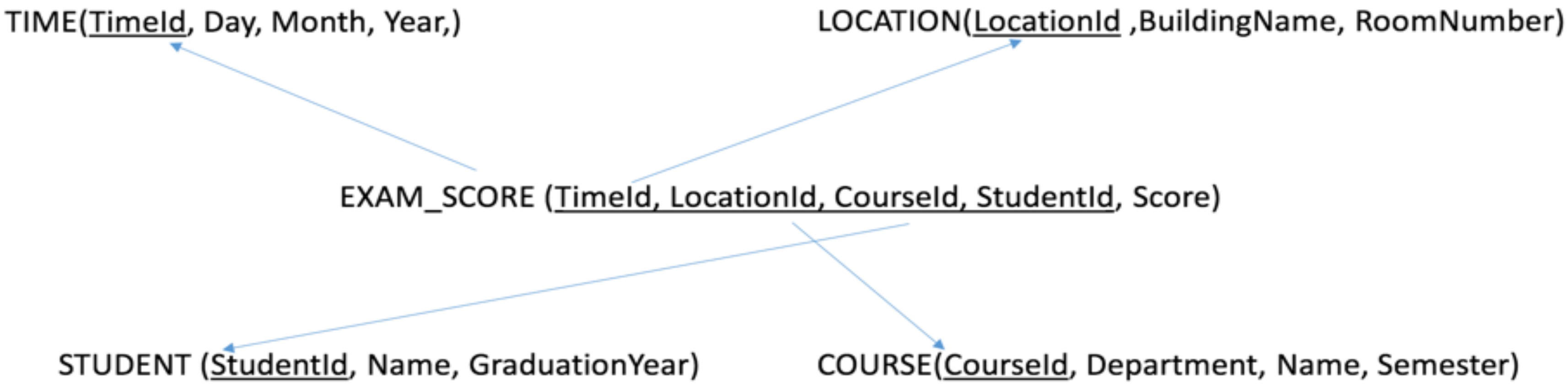
Data Science / Queries Worksheet

Data Science / Queries Exercises

1. Create a star schema for this data (be sure to include all fields). You may need to introduce new fields.

Data Science / Queries Exercises

1. Create a star schema for this data (be sure to include all fields). You may need to introduce new fields.



TIME(TimeId, Day, Month, Year,)

LOCATION(LocationId ,BuildingName, RoomNumber)

EXAM_SCORE (TimeId, LocationId, CourseId, StudentId, Score)

STUDENT (StudentId, Name, GraduationYear)

COURSE(CourseId, Department, Name, Semester)

Data Science / Queries Exercises

2a. Fill in our pivot table where the dimensions are Make and Year, and our aggregation is MEAN().

	Ferrari	Tesla	Total
2014			
2015			
Total			

Data Science / Queries Exercises

2a. Fill in our pivot table where the dimensions are Make and Year, and our aggregation is MEAN().

	Ferrari	Tesla	Total
2014	50	-	50
2015	85	80	82.5
Total	67.5	80	71.67

Data Science / Queries Exercises

2b. How many rows are in the output of this query?

```
SELECT Make, Year, Color, SUM(Sales) FROM Sales
 GROUP BY Make, Year, Color WITH CUBE;
```

Data Science / Queries Exercises

2b. How many rows are in the output of this query?

```
SELECT Make, Year, Color, SUM(Sales) FROM Sales
 GROUP BY Make, Year, Color WITH CUBE;
```

(Ferrari, Tesla, *) x (2014, 2015, *) x (Red, *)

3 * 3 * 2 = 18 rows

Data Science / Queries Exercises

2b. Fill in the rows:

Make	Year	Color	SUM(Sales)
Ferrari	*	Red	
Tesla	*	*	
*	*	*	

Data Science / Queries Exercises

2b. Fill in the rows:

Make	Year	Color	SUM(Sales)
Ferrari	*	Red	135
Tesla	*	*	80
*	*	*	215

Data Science / Queries Exercises

2c. Suppose we “drill down” (deaggregate) on Year, by quarter. How many rows are in the output ?

Data Science / Queries Exercises

2c. Suppose we “drill down” (deaggregate) on Year, by quarter. How many rows are in the output ?

3 rows * 4 quarters per row = 12

Data Science / Queries Exercises

2d. If we sold equal numbers of Teslas each quarter, what does the drill down by year look like for Tesla?

Make	Year	Color	Quarter	Sales
Tesla	2015	Red		

Data Science / Queries Exercises

2d. If we sold equal numbers of Teslas each quarter, what does the drill down by year look like for Tesla?

Make	**Year**	**Color**	**Quarter**	**Sales**
Tesla	2015	Red	Q1	20
Tesla	2015	Red	Q2	20
Tesla	2015	Red	Q3	20
Tesla	2015	Red	Q4	20