Christmas Santa 2.0

Database Systems 308
Kayla Mesmain

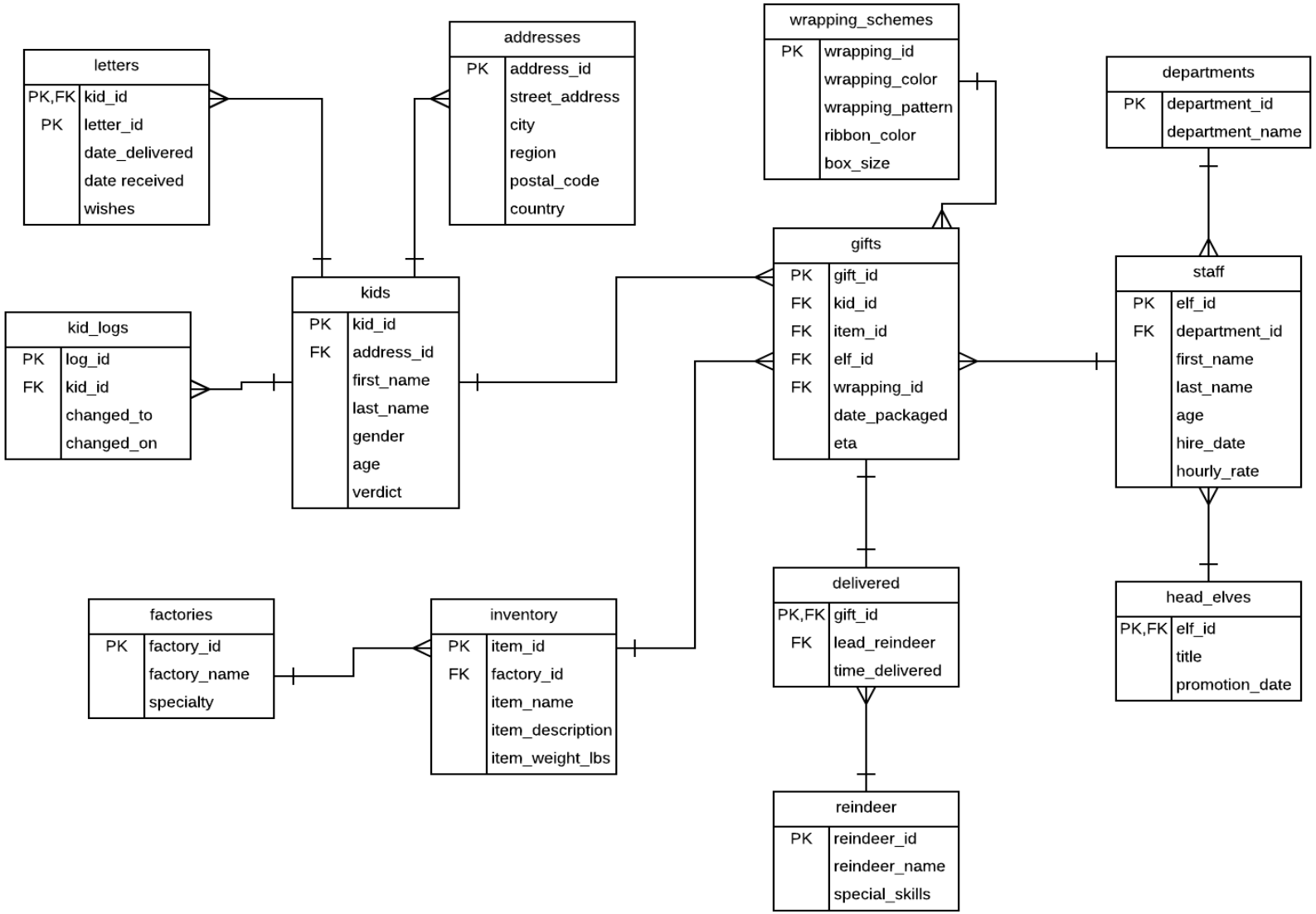
Table of contents

Executive Summary

The holidays are coming soon. Every year a child requests a gift from Santa. Santa can't operate the entire north pole on just pen and paper. Christmas Santa 2.0 database helps Santa and his workers have an organized and controlled Christmas.

The database is designed to show the process that it takes once a child writes a letter to the process in which the child receives the gift from Santa. This multi step involves storing information in a table to create a database. The entity relationship diagram includes all the tables of the process of receiving the gift. In the following document you will find the steps in order to make Christmas successful.

ER -DIAGRAM


letters
PK,FK kid_id
PK letter_id
date_delivered
date received
wishes
addresses
PK address_id
street_address
city
region
postal_code
country
wrapping_schemes
PK wrapping_id
wrapping_color
wrapping_pattern
ribbon_color
box_size
departments
PK department_id
department_name
kid_logs
PK log_id
FK kid_id
changed_to
changed_on
kids
PK kid_id
FK address_id
first_name
last_name
gender
age
verdict
gifts
PK gift_id
FK kid_id
FK item_id
FK elf_id
FK wrapping_id
date_packaged
eta
staff
PK elf_id
FK department_id
first_name
last_name
age
hire_date
hourly_rate
factories
PK factory_id
factory_name
specialty
inventory
PK item_id
FK factory_id
item_name
item_description
item_weight_lbs
delivered
PK,FK gift_id
FK lead_reindeer
time_delivered
head_elves
PK,FK elf_id
title
promotion_date
reindeer
PK reindeer_id
reindeer_name
special_skills

The kids table contains all the kids of the world. Each kid is given an ID which is connected to their address ID, name, gender, age, and Santa's Current Verdict.

kids Table

```
CREATE TABLE kids(
kid_id integer NOT NULL,
address_id integer NOT NULL,
first_name text,
last_name text,
gender text,
age integer,
verdict text,
CONSTRAINT kids_pkey PRIMARY KEY (kid_id),
CONSTRAINT kids_age_check CHECK (age < 22)),
CONSTRAINT kids_verdict_check CHECK (verdict =
'Naughty'::text OR verdict = 'Nice'::text);
```

	kid_id integer	address_id integer	first_name text	last_name text	gender text	age integer	verdict text
98	447889	325897	Buffy	Jefferson	Female	14	Nice
99	447943	325895	Lael	Jenkins	Female	19	Naughty
100	447824	325910	Ethan	Jennings	Male	2	Nice
101	447942	325903	Zelenia	Jones	Female	1	Nice
102	447901	325902	Alexandra	Jones	Female	1	Nice
103	447775	325867	Bert	Joseph	Male	12	Nice
104	447786	325929	Cairo	Joyce	Male	17	Naughty
105	447831	325865	Brock	Kaufman	Male	17	Nice
106	447819	325874	Amos	Kemp	Male	9	Naughty
107	447812	325893	Paki	Kidd	Male	8	Naughty
108	447884	325900	Brynn	Kramer	Female	10	Nice
109	447866	325838	Alan	Labouseur	Male	21	Nice
110	447857	325838	Kyle	Lambert	Male	21	Naughty
111	447886	325909	Lillian	Lawson	Female	18	Naughty
112	447802	325910	Eaton	Le	Male	17	Naughty

Functional Dependencies: kid_id→ address_id, first_name, last_name, gender, age, verdict

Constraints: kids_age_check→ Checks to see kid is under the age of 22

kids_verdict_check→ checks to see if verdict is equal to 'Naughty' or 'Nice'

The addresses Table contains all the addresses belonging to each of the children. Each address is given an ID and each address ID is linked to a Street Address, City, Region("State"), Postal code and Country anywhere in the World.

```
CREATE TABLE addresses(
address_id integer NOT NULL,
street_address text NOT NULL,
city text,
region text,
postal_code text,
country text NOT NULL,
CONSTRAINT addresses_pkey PRIMARY
KEY (address_id));
```

addresses Table

	address_id integer	street_address text	city text	region text	postal_code text	country text
1	325837	887-5208 Dolor, St.	Anjou	Quebec	44039-585	Madagascar
2	325838	P.O. Box 220, 3758 Libero. Avenue	Segni	Lazio	37635	Hungary
3	325839	P.O. Box 214, 2825 Integer St.	Carapicuíba	São Paulo	43703	Mayotte
4	325840	5655 Ac Av.	Elgin	Morayshire	13645	Belarus
5	325841	Ap #434-3217 Id, St.	Teruel	Aragón	WU3Z 7CP	Poland
6	325842	P.O. Box 688, 1947 Eget Avenue	Rzeszów	PK	24599	Aruba
7	325843	192-277 Semper Rd.	El Tabo	V	08-896	United Kingdom (Great Britain)
8	325844	Ap #898-814 Dapibus St.	Pointe-Claire	Quebec	30308	Christmas Island
9	325845	P.O. Box 740, 1621 Sed Rd.	Morwell	VIC	AZ1 8NW	Kyrgyzstan
10	325846	Ap #680-6434 Pellentesque. Ave	Portland	Maine	2801	South Sudan

Functional Dependencies: address_id→ street_address, city, region, postal_code, country

The letters Table stores the information from all the letters sent in by each kid. Kids can send in multiple letters but each letter will have its own unique ID and linked to each letter ID will be the date the letter was delivered, the date the letter was received, and lastly a description of the kids Christmas wishes.

letters Table

	kid_id integer	letter_id integer	date_delivered date	date_received date	wishes text
1	447833	665522	2017-09-28	2017-12-07	Bike
2	447820	665523	2017-09-25	2017-11-22	Xbox One X
3	447893	665524	2017-09-06	2017-10-22	PS4 Pro
4	447825	665525	2017-09-27	2017-11-09	Dog
5	447890	665526	2017-09-21	2017-11-24	Bike
6	447769	665527	2017-09-22	2017-11-06	iPhone X
7	447936	665528	2017-09-01	2017-11-25	Apple Watch
8	447959	665529	2017-09-10	2017-11-08	PS4 Pro
9	447881	665530	2017-09-28	2017-12-02	Dog
10	447909	665531	2017-09-21	2017-11-20	Bike
11	447839	665532	2017-09-22	2017-10-06	World Peace
12	447822	665533	2017-09-22	2017-11-29	Apple Watch
13	447773	665534	2017-09-23	2017-10-16	World Peace
14	447879	665535	2017-09-21	2017-12-16	Macbook Pro
15	447842	665536	2017-09-27	2017-11-17	Nintendo Switch

```
CREATE TABLE letters(
kid_id integer NOT NULL,
letter_id integer NOT NULL,
date_delivered date NOT NULL,
date_received date,
wishes text,
CONSTRAINT letters_pkey PRIMARY KEY (kid_id, letter_id),
CONSTRAINT letters_check CHECK (date_delivered <
date_received));
```

Functional Dependencies: kid_id, letter_id→ date_delivered, date_received, wishes
Constraints: letters_check → Checks to see if date received comes after date delivered

The kid logs table logs each and every time a kid switches from naughty to nice or vice versa in order to track their behavior over time. Each log is given an ID and connected to a kid_id, the behavior they changed to, as well as the time the change occurred.

kid_logs Table

```
CREATE TABLE kid_logs(
log_id integer NOT NULL,
kid_id integer NOT NULL,
changed_to text NOT NULL,
changed_on timestamp without time
zone NOT NULL,  CONSTRAINT
kid_logs_pkey PRIMARY KEY
(log_id));
```

	log_id integer	kid_id integer	changed_to text	changed_on timestamp without time zone
1	2	447866	Naughty	2017-11-30 00:32:11.379301
2	3	447866	Nice	2017-11-30 00:32:53.33635
3	4	447767	Naughty	2017-11-30 12:59:37.489802
4	5	447768	Naughty	2017-11-30 12:59:37.489802
5	6	447771	Naughty	2017-11-30 12:59:37.489802
6	7	447775	Naughty	2017-11-30 12:59:37.489802
7	8	447777	Naughty	2017-11-30 12:59:37.489802
8	9	447779	Naughty	2017-11-30 12:59:37.489802
9	10	447769	Nice	2017-11-30 12:59:37.489802
10	11	447773	Nice	2017-11-30 12:59:37.489802
11	12	447778	Nice	2017-11-30 12:59:37.489802
12	13	447783	Nice	2017-11-30 12:59:37.489802

Functional Dependencies: log_id → kid_id, changed_to, changed_on

The gifts table contains a list of all the packaged wrapped and ready for delivery by Santa. Each gift is given a unique gift ID and is linked with a kid ID to determine who its for, a item ID for the contents of the gift, a elf ID to see who prepared the gift, a wrapping ID to see what scheme was used to wrap the gift, the date it was packaged, and an estimated time of arrival for the gift.

gifts Table

```
CREATE TABLE gifts(
gift_id integer NOT NULL,
kid_id integer NOT NULL,
item_id integer NOT NULL,
elf_id integer NOT NULL,
wrapping_id integer NOT NULL,  date_packaged date,
eta date,
CONSTRAINT gifts_pkey PRIMARY KEY (gift_id),
CONSTRAINT gifts_check CHECK (date_packaged <= eta));
```

	gift_id integer	kid_id integer	item_id integer	elf_id integer	wrapping_id integer	date_packaged date	eta date
1	273756	447823	514396	100031	890029	2017-12-24	2017-12-24
2	273757	447861	514407	100005	890000	2017-12-23	2017-12-25
3	273758	447933	514407	100024	890021	2017-12-16	2017-12-24
4	273759	447938	514392	100119	890053	2017-12-18	2017-12-24
5	273760	447933	514404	100118	889981	2017-12-17	2017-12-25
6	273761	447828	514405	100119	890030	2017-12-06	2017-12-25
7	273762	447820	514408	100086	890036	2017-12-19	2017-12-25
8	273763	447938	514388	100189	890037	2017-12-13	2017-12-24
9	273764	447865	514403	100088	890021	2017-12-24	2017-12-24
10	273765	447882	514392	100174	889976	2017-12-16	2017-12-24
11	273766	447773	514393	100030	890042	2017-12-22	2017-12-24
12	273767	447957	514407	100183	890049	2017-12-15	2017-12-25

Functional Dependencies: gift_id→ kid_id, item_id, elf_id, wrapping_id, date_packaged, eta

Constraints: gifts_check→ Makes sure that date_packaged comes before or on estimated time of arrival

The wrapping_schemes Table holds all the different wrapping combinations elves can wrap presents in. Each Wrapping Scheme has a Unique ID which is linked to a wrapping paper color, a wrapping paper pattern, a ribbon color, and a box size.

wrapping_schemes Table

	wrapping_id integer	wrapping_color text	wrapping_pattern text	ribbon_color text	box_size text
1	889976	Blue	Gingerbread Men	Indigo	Small
2	889977	Indigo	Stripes	Pink	X-Large
3	889978	Pink	Jingle Bells	Pink	Medium
4	889979	Pink	Candy Canes	Black	X-Large
5	889980	Black	Santa	Black	Small
6	889981	Black	Snowflakes	Indigo	Large
7	889982	Yellow	Jingle Bells	Brown	Medium
8	889983	Indigo	SnowGlobes	Indigo	X-Large
9	889984	Pink	Snowmen	Pink	Medium
10	889985	Orange	Gingerbread Men	Yellow	Large
11	889986	Green	Jingle Bells	Red	Small
12	889987	White	Christmas Trees	Violet	Medium

```
CREATE TABLE wrapping_schemes(
wrapping_id integer NOT NULL,
wrapping_color text,
wrapping_pattern text,
ribbon_color text,
box_size text,
CONSTRAINT wrapping_schemes_pkey PRIMARY
KEY (wrapping_id));
```

Functional dependences: wrapping_id→ wrapping_color, wrapping_pattern, ribbon_color, box_size

The inventory Table is used to store all the different items made in Santa's workshop. Each item is unique has their own ID along with the factory ID in which it was made, the item name, a description of the item and lastly the weight of the item in lbs.

inventory Table

```
CREATE TABLE inventory(
item_id integer NOT NULL,
factory_id integer NOT NULL,
item_name text,
item_description text,
item_weight_lbs integer,
CONSTRAINT inventory_pkey PRIMARY
KEY (item_id));
```

	item_id integer	factory_id integer	item_name text	item_description text	item_weight_lbs integer
1	514388	663532	FIFA 18(Xbox One)	Video Game for Xbox One gaming system	1
2	514389	663532	FIFA 18(PS4)	Video Game for PS4 gaming system	1
3	514390	663522	Gun(water)	Water gun	2
4	514391	663522	Gun(nerf)	Gun that shoot foam bullets	5
5	514392	663523	Coloring book(Frozen)	Disney movie frozen color book	1
6	514393	663523	Paint(face)	Face and Skin paint	1
7	514394	663524	Blanket(Froggy)	Blanket with frogs on it	1
8	514395	663524	Play Pin(baby toy)	Child to play in safe area	10
9	514396	663525	LEGOS(42 packet)	building blocks	3
10	514397	663525	Toy hammer(building tool)	building tool set	2
11	514398	663526	Easy bake oven(kitchen)	Kitchen oven	9
12	514399	663526	Linda recipe book(baking)	Cookbook	1
13	514400	663527	teddy bear(stuff animal)	stuffed animal toy	2

Functional Dependencies: item_id→ factory_id, item_name, item_description, item_weight_lbs

The factories Table is used to store information about each different factory that produces items for inventory. Each factory has a unique ID that is linked to the factories name/category and also a description of items produced in each factory.

factories Table

	factory_id integer	factory_name text	speciality text
1	663522	Action	Nerf Guns, Action Figures, Super Soakers, etc.
2	663523	Arts	Crafts, Coloring Books, Paint, etc
3	663524	Baby	Learning Toys, Bottles, Blankets, etc.
4	663525	Building	Blocks, LEGOS, Toy Tools, etc
5	663526	Cooking	Play Kitchens, Easy Bake Oven, Cookbooks, etc.
6	663527	Dolls	Dress-Up, Stuffed Animals, American Girl, etc.
7	663528	Electronics	Tech Toys, Movies, Music, etc.
8	663529	Games	Puzzles, Board Games, Outdoor Games, etc.
9	663530	Outdoor	Trampoline, Mini-Pool, Basketball Hoops, etc
10	663531	Vehicles	Bikes, Control Cars, Trains, etc.
11	663532	Video	Video Games, Online Gift Cards, Controllers
12	663533	Pets	Rescue Dogs, Cats, etc. looking for a new home.
13	663534	Clothes	Dresses, Bags, Watches, etc.

```
CREATE TABLE factories(
factory_id integer NOT NULL,
factory_name text,
speciality text,
CONSTRAINT factories_pkey
PRIMARY KEY (factory_id));
```

Functional Dependences: factory_id→ factory_name, speciality

The staff Table is used to store information about the elves working at the North Pole. Each elf is given an ID and all ID's are linked to the department ID of where the elf is working, the first and last name of the elf, the elf's age, the date the elf was hired, and Lastly the elf's hourly rate.

staff Table

	elf_id integer	department_id integer	first_name text	last_name text	age integer	hire_date date	hourly_rate money
1	100000	382135	Price	Hightop	4624	1904-05-30	$0.61
2	100001	382139	Oren	Who	5395	1968-03-15	$91.61
3	100002	382136	Hunter	Evergreen	1219	1954-03-27	$37.26
4	100003	382141	Dalton	Smith	2284	1974-12-25	$2.21
5	100004	382140	Owen	Joy	505	1874-02-14	$51.54
6	100005	382135	Axel	Openslae	3081	1977-04-28	$79.92
7	100006	382143	Lawrence	Mary	6691	1926-05-20	$54.99
8	100007	382140	Ulric	Minstix	3969	1923-09-29	$17.89
9	100008	382134	Ivor	Maybottom	2582	2002-01-30	$2.16
10	100009	382145	Robert	Openslae	5409	1897-12-28	$25.03
11	100010	382138	Walker	Evergreen	5708	1908-12-24	$86.20
12	100011	382138	Vaughan	Jones	2762	1992-04-28	$19.45

```
CREATE TABLE staff(
elf_id integer NOT NULL,
department_id integer NOT NULL,
first_name text,
last_name text,
age integer,
hire_date date,
hourly_rate money,
CONSTRAINT staff_pkey PRIMARY KEY (elf_id));
```

Functional Dependencies: elf_id→ department_id, first_name, last_name, age, hire_date, hourly_rate

The departments table is used to list all the different departments the North Pole has. Each department has a Unique ID and each ID is associated with a specific department name.

	department_id integer	department_name text
1	382133	Production
2	382134	Research and Development
3	382135	Purchasing
4	382136	Marketing
5	382137	Human Resources
6	382138	Finance
7	382139	Customer Service
8	382140	Distribution
9	382141	Santa Care
10	382142	Legal
11	382143	IT Support
12	382144	Operations
13	382145	Administrative

departments Table

```
CREATE TABLE departments(
department_id integer NOT NULL,
department_name text,
CONSTRAINT departments_pkey PRIMARY KEY
(department_id));
```

Functional Dependencies: department_id→ department_name

The head elves Table is used keep track of all the head elves. Elves with more permissions/access in the system are called head elves and the head elves table associates a head elves ID with their job title and the date they were promoted.

head_elves Table

```
CREATE TABLE head_elves(
elf_id integer NOT NULL,
title text,
promotion_date timestamp without time
zone,
CONSTRAINT head_elves_pkey
PRIMARY KEY (elf_id));
```

	elf_id integer	title text	promotion_date date
1	100120	Chief Manufacturing Engineer	1999-01-23
2	100008	Department Manager	1994-11-17
3	100000	Resourcing Lead	2001-06-12
4	100002	Marketing Director	2000-01-01
5	100020	Marketing Manager	2005-04-21
6	100031	Chief Happiness Officer	2009-06-06
7	100033	Chief People Officer	2008-06-06
8	100111	Accounting Director	2010-03-05
9	100112	Accounting Manager	1999-06-21
10	100176	Department Manager	2003-03-03
11	100030	Chief Legal Analyist	1998-09-03

Functional Dependencies: elf_id→ title, promotion_date

The delivered Table is used to keep track of all the gifts that have already been delivered. Each gift ID within the delivered Table is associated with a reindeer ID which signifies which reindeer was leading the sleigh during the time of delivery as well as the time the gift was delivered.

delivered Table

	gift_id integer	lead_reindeer integer	time_delivered timestamp with time zone
1	273794	178636	2017-12-24 00:16:50-05
2	273816	178641	2017-12-25 00:37:54-05
3	273774	178640	2017-12-25 00:52:38-05
4	273775	178633	2017-12-24 04:52:28-05
5	273767	178636	2017-12-25 07:39:17-05
6	273809	178633	2017-12-24 22:14:55-05
7	273841	178633	2017-12-24 11:13:59-05
8	273792	178641	2017-12-25 04:23:41-05
9	273771	178638	2017-12-25 05:59:01-05
10	273778	178634	2017-12-25 06:31:06-05
11	273773	178635	2017-12-25 06:03:20-05
12	273802	178637	2017-12-24 14:31:27-05

```
CREATE TABLE delivered(
gift_id integer NOT NULL,
lead_reindeer integer NOT NULL,
time_delivered timestamp with time zone NOT NULL,
CONSTRAINT delivered_pkey PRIMARY KEY
(gift_id));
```

Functional Dependencies: gift_id → lead_reindeer, time_delivered

The reindeer Table is used to keep track of all the reindeer Santa uses to pull his sleigh. Each reindeer is given a unique ID that is associated with a reindeer name as well as a list of special skills linked to that specific reindeer.

```
CREATE TABLE reindeer(
reindeer_id integer NOT NULL,
reindeer_name text,
special_skills text,
CONSTRAINT reindeer_pkey PRIMARY
KEY (reindeer_id));
```

Reindeer Table

	reindeer_id integer	reindeer_name text	special_skills text
1	178633	Dasher	Speed
2	178634	Prancer	High Jumper
3	178635	Comet	Speed
4	178636	Vixen	None
5	178637	Dancer	Stamina
6	178638	Cupid	Navigator
7	178639	Donner	None
8	178640	Blitzen	Short Bursts of Speed
9	178641	Rudolph	Navigator, Shiny Nose

Functional Dependencies: reindeer_id→ reindeer_name, special_skill

kids_wishes View:

Retrieves information about kids who sent in a letter and his/her Christmas wish to Santa

```
create or replace view kids_wishes
  as
  SELECT k.kid_id,
            k.first_name,
            k.last_name,
            k.age, l.wishes
  FROM kids as k INNER JOIN letters as l
                        ON k.kid_id = l.kid_id
  GROUP BY k.kid_id, l.wishes
  ORDER by k.last_name;

Select * from kids_wishes;
```

	kid_id integer	first_name text	last_name text	age integer	wishes text
1	447962	Xena	Abbott	13	PS4 Pro
2	447959	Fredericka	Alford	15	PS4 Pro
3	447791	Byron	Anthony	10	Nintendo Switch
4	447859	Levi	Armstrong	2	Apple Watch
5	447930	Cleo	Arnold	12	Apple Watch
6	447892	Quintessa	Bell	13	Macbook Pro
7	447892	Quintessa	Bell	13	World Peace
8	447899	Gail	Bennett	17	Apple Watch
9	447822	Seth	Blanchard	12	Apple Watch
10	447846	Ezra	Bowers	8	iPhone X

item_gifted_amount View:

Retrieves information about an Item and the frequency an item is given as a gift to a kid.

```
create or replace view item_gifted_amount
  as
  SELECT i.item_id,
         i.item_name,
         i.item_description,
         COUNT(g.item_id) AS item_gifted_amount
  FROM inventory as i INNER JOIN gifts as g
                      ON i.item_id = g.item_id
  GROUP BY i.item_id
  ORDER by item_gifted_amount desc;
------------
Select * from item_gifted_amount;
```

	item_id integer	item_name text	item_description text	item_gifted_amount bigint
1	514391	Gun(nerf)	Gun that shoot foam bullets	12
2	514407	mini trampoline(trampoline)	outdoor trampoline	12
3	514411	Taby Cat(cat)	rescue cat	11
4	514395	Play Pin(baby toy)	Child to play in safe area	10
5	514392	Coloring book(Frozen)	Disney movie frozen color book	10
6	514390	Gun(water)	Water gun	10
7	514412	Forever 21(clothing)	dress	9
8	514399	Linda recipe book(baking)	Cookbook	9
9	514405	twister(game)	board game	9
10	514403	Beyonce(music)	music cd	9

not_sent_letters View:

Retrieves information about any kid who has not sent in a letter to Santa

```
CREATE or REPLACE view not_sent_letters
as
SELECT distinct k.kid_id,
                k.first_name,
                k.last_name,
                a.country
FROM kids as k, addresses as a
WHERE k.kid_id NOT IN
        (select l.kid_id from letters as l)
AND (a.address_id = k.address_id)
ORDER BY k.last_name;
----------
SELECT * FROM not_sent_letters;
```

	kid_id integer	first_name text	last_name text	country text
1	447795	Connor	Abbott	United Kingdom (Great Britain)
2	447867	Cecilia	Adkins	Afghanistan
3	447848	Zeus	Aguilar	Andorra
4	447945	Jaden	Alston	Puerto Rico
5	447782	Carl	Anderson	Ukraine
6	447926	Maite	Andrews	Belgium
7	447816	Arden	Ayers	Azerbaijan
8	447850	Patrick	Barr	Western Sahara
9	447964	Angelica	Bartlett	French Guiana
10	447801	Lars	Bass	Singapore

Stored Procedure: get_gift_info

This procedure takes in a child's first name and last name and retrieve that child's gift information. This procedure will also work with partial entries using the '%' symbol

```
CREATE OR REPLACE FUNCTION get_gift_info (TEXT, TEXT, REFCURSOR) RETURNS refcursor AS
$$
DECLARE
  input_one TEXT := $1;
  input_two TEXT := $2;
  resultSet REFCURSOR := $3;
BEGIN
  OPEN resultset FOR

    select k.first_name, k.last_name, i.item_name as Item, s.first_name as Packaged_By,
w.box_size as Box_Size
    from gifts as g, kids as k, inventory as i, staff as s, wrapping_schemes as w
    where
      (g.kid_id = k.kid_id) AND
      (g.item_id = i.item_id) AND
      (g.elf_id = s.elf_id) AND
      (g.wrapping_id = w.wrapping_id) AND
      k.first_name LIKE input_one AND
      k.last_name LIKE input_two
      order by k.last_name;

    return resultSet;
end;
$$
LANGUAGE plpgsql;
```

```
SELECT get_gift_info('Nora', 'Gentry', 'ref2');
FETCH ALL FROM ref2;
```

	first_name text	last_name text	item text	packaged_by text	box_size text
1	Nora	Gentry	Coloring book(Frozen)	Charlotte	Small
2	Nora	Gentry	Blanket(Froggy)	Rhonda	X-Large
3	Nora	Gentry	teddy bear(stuff animal)	Rigel	Medium
4	Nora	Gentry	mini trampoline(trampoline)	Yeo	Large
5	Nora	Gentry	Gun(water)	Jenna	Medium
6	Nora	Gentry	LEGOS(42 packet)	Alice	Medium
7	Nora	Gentry	Toy hammer(building tool)	Stacy	Small

```
--First Names that start with A
SELECT get_gift_info('A%', '%', 'ref3');
FETCH ALL FROM ref3;
```

	first_name text	last_name text	item text	packaged_by text	box_size text
1	Arden	Ayers	Linda recipe book(baking)	Yasir	Small
2	Aphrodite	Castro	German Shepard(dog)	September	X-Large
3	Armand	Farmer	Beyonce(music)	Ashton	Small
4	Armand	Farmer	Coloring book(Frozen)	Wayne	Medium
5	Ayanna	Nolan	Taby Cat(cat)	Len	Large
6	Anika	Ramirez	Ferrari(car)	Curran	X-Large
7	Anika	Ramirez	monster inc puzzle(puzzle)	Dane	Medium

Stored Procedure: get_elf_gift_sum

This procedure takes in a elf's first name and last name and retrieve's the number of gifts that elf has wrapped and prepared for delivery. This procedure will also work with partial entries using the '%' symbol

```
CREATE OR REPLACE FUNCTION get_elf_gift_sum (TEXT, TEXT, REFCURSOR)
RETURNS refcursor AS
$$
DECLARE
  input_one TEXT := $1;
  input_two TEXT := $2;
  resultSet REFCURSOR := $3;
BEGIN
  OPEN resultset FOR

    select s.first_name, s.last_name, COUNT(g.gift_id) as Packaged
    from staff as s
    inner join  gifts as g on s.elf_id = g.elf_id
    where
      s.first_name LIKE input_one AND
      s.last_name LIKE input_two
    group by s.first_name, s.last_name
    order by s.last_name;

    return resultSet;
end;
$$
LANGUAGE plpgsql;
```

```
SELECT get_elf_gift_sum('Nadine', 'Minstix', 'ref');
FETCH ALL FROM ref;
```

	first_name text	last_name text	packaged bigint
1	Nadine	Minstix	3

```
--Last Names that start with the letters 'Min'
SELECT get_elf_gift_sum('%', 'Min%', 'ref3');
FETCH ALL FROM ref3;
```

	first_name text	last_name text	packaged bigint
1	Anjolie	Minstix	2
2	Castor	Minstix	1
3	Cooper	Minstix	2
4	Daquan	Minstix	1
5	Erin	Minstix	3

Trigger: rate_changes

This trigger instantly updates a head elf's promotion date to the current date and time of the update if they are to receive another promotion. A promotion is triggered by a change in the head elf's hourly rate. ("The trigger will only occur if the new hourly_rate value is bigger than the old hourly_rate value")

Trigger Function

```
CREATE OR REPLACE FUNCTION hourly_rate_changes()
  RETURNS trigger AS
$BODY$
BEGIN
IF NEW.hourly_rate <> OLD.hourly_rate AND
    NEW.hourly_rate > OLD.hourly_rate
THEN
     UPDATE head_elves SET promotion_date = now() WHERE elf_id = OLD.elf_id;
END IF;

RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
```

Trigger

```
CREATE TRIGGER rate_changes
  BEFORE UPDATE
  ON staff
  FOR EACH ROW
  EXECUTE PROCEDURE hourly_rate_changes();
```

Testing Trigger: rate_changes

Before:

Staff Table:

	elf_id integer	department_id integer	first_name text	last_name text	age integer	hire_date date	hourly_rate money
1	100000	382135	Price	Hightop	4624	1904-05-30	$100.00
2	100001	382139	Oren	Who	5395	1968-03-15	$91.61
3	100002	382136	Hunter	Evergreen	1219	1954-03-27	$37.26
4	100003	382141	Dalton	Smith	2284	1974-12-25	$2.21
5	100004	382140	Owen	Joy	505	1874-02-14	$51.54

Head_elves Table:

	elf_id integer	title text	promotion_date timestamp without time zone
1	100000	Resourcing Lead	2017-12-01 19:40:39.642472
2	100002	Marketing Director	2000-01-01 00:00:00
3	100008	Department Manager	2017-12-02 21:09:53.888043
4	100020	Marketing Manager	2017-12-01 19:40:39.642472
5	100030	Chief Legal Analyist	1998-09-03 00:00:00

UPDATE staff **SET** hourly_rate = '$100.00' **WHERE** elf_id = 100002;

After:

	elf_id integer	department_id integer	first_name text	last_name text	age integer	hire_date date	hourly_rate money
1	100000	382135	Price	Hightop	4624	1904-05-30	$100.00
2	100001	382139	Oren	Who	5395	1968-03-15	$91.61
3	100002	382136	Hunter	Evergreen	1219	1954-03-27	$100.00
4	100003	382141	Dalton	Smith	2284	1974-12-25	$2.21
5	100004	382140	Owen	Joy	505	1874-02-14	$51.54

	elf_id integer	title text	promotion_date timestamp without time zone
1	100000	Resourcing Lead	2017-12-01 19:40:39.642472
2	100002	Marketing Director	2017-12-02 21:37:53.815298
3	100008	Department Manager	2017-12-02 21:09:53.888043
4	100020	Marketing Manager	2017-12-01 19:40:39.642472
5	100030	Chief Legal Analyist	1998-09-03 00:00:00

Trigger: verdict_changes

This trigger instantly keeps track of the frequency a kid switches from Naughty to Nice and vice versa. When an update is made to a specific kid's verdict in the kids table the trigger activates and inserts a new log with a Unique ID that is linked to the kids kid_id, the verdict they changed to ('Naughty' or 'Nice'), and the time of occurrence.

Trigger Function

```
CREATE OR REPLACE FUNCTION log_verdict_changes()
  RETURNS trigger AS
$BODY$
BEGIN
IF NEW.verdict <> OLD.verdict THEN
INSERT INTO kid_logs(kid_id,changed_to,changed_on)
VALUES(OLD.kid_id,NEW.verdict,now());
END IF;

RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
```

Trigger

```
CREATE TRIGGER verdict_changes
  BEFORE UPDATE
  ON kids
  FOR EACH ROW
  EXECUTE PROCEDURE public.log_verdict_changes();
```

Testing Trigger: verdict_changes

Kids Table:

Before:

	kid_id integer	address_id integer	first_name text	last_name text	gender text	age integer	verdict text
1	447766	325903	Walter	Simmons	Male	3	Naughty
2	447772	325879	Amery	Gallegos	Male	13	Naughty
3	447774	325887	Berk	Myers	Male	10	Naughty

Kid_logs Table:

	log_id integer	kid_id integer	changed_to text	changed_on timestamp without time zone
1	13	447767	Naughty	2017-12-01 19:40:39.642472
2	14	447768	Naughty	2017-12-01 19:40:39.642472
3	21	447769	Nice	2017-12-01 19:40:39.642472
4	15	447771	Naughty	2017-12-01 19:40:39.642472
5	22	447773	Nice	2017-12-01 19:40:39.642472

UPDATE kids SET verdict = 'Nice' WHERE kid_id = 447772;

After:

6	447771	325876	Davis	Dudley	Male	11	Naughty
7	447772	325879	Amery	Gallegos	Male	13	Nice
8	447773	325910	Perry	Cleveland	Male	2	Nice

	log_id integer	kid_id integer	changed_to text	changed_on timestamp without time zone
1	13	447767	Naughty	2017-12-01 19:40:39.642472
2	14	447768	Naughty	2017-12-01 19:40:39.642472
3	21	447769	Nice	2017-12-01 19:40:39.642472
4	15	447771	Naughty	2017-12-01 19:40:39.642472
5	25	447772	Nice	2017-12-02 22:14:58.147074

Reports on Christmas 17': Average Age

Average age of kids receiving gifts for Christmas 2017:

```
SELECT round(sum(k.age) / count(k.kid_id)) AS average_age
FROM kids as k, gifts as g
WHERE k.kid_id in (select kid_id from gifts) AND
        g.date_packaged > '2016-12-25';
```

	average_age double precision
1	11

Reports on Christmas 17': Number of Gifts

Number of Gifts delivered
for Christmas 2017:

```
SELECT count(gift_id) as gift_count
FROM gifts
WHERE date_packaged > '2016-12-25';
```

	gift_count bigint
1	194

Reports on Christmas 17': Total Weight

Total weight of Santa's Sleigh for Christmas 2017: *Not including the sleighs weight alone or Santa himself*

```
SELECT sum(i.item_weight_lbs) AS sled_weight_lbs
FROM gifts as g INNER JOIN inventory as i ON (g.item_id = i.item_id)
WHERE g.date_packaged > '2016-12-25';
```

	sled_weight_lbs bigint
1	794

Security: User Roles

```
create role santa;
create role head_elf;
create role elf;
```

Santa: He see's all and controls all

```
grant all on all tables in schema public to santa;
```

Head Elf: Total control over addresses, gifts, wrapping_schemes, inventory and delivered tables. Limited control over kids and letters table. Only viewing permitted on reindeer, factories, departments , and kid logs table.

```
grant SELECT on kid_logs to head_elf;
grant SELECT on departments to head_elf;
grant SELECT on factories to head_elf;
grant SELECT on reindeer to head_elf;
grant SELECT, INSERT on kids to head_elf;
grant SELECT, INSERT on letters to head_elf;
grant SELECT, INSERT, UPDATE, DELETE on addresses to head_elf;
grant SELECT, INSERT, UPDATE, DELETE on gifts to head_elf;
grant SELECT, INSERT, UPDATE, DELETE on wrapping_schemes to head_elf;
grant SELECT, INSERT, UPDATE, DELETE on delivered to head_elf;
grant SELECT, INSERT, UPDATE, DELETE on inventory to head_elf;
```

Elf: Limited control permitted on gifts and inventory table. The rest of the database is view only access.

```
grant SELECT on letters to elf;
grant SELECT on kid_logs to elf;
grant SELECT on kids to elf;
grant SELECT on addresses to elf;
grant SELECT on wrapping_schemes to elf;
grant SELECT on departments to elf;
grant SELECT on delivered to elf;
grant SELECT on factories to elf;
grant SELECT on reindeer to elf;
grant SELECT, INSERT, UPDATE on gifts to elf;
grant SELECT, INSERT, UPDATE on inventory to elf;
```

Implementation notes

- Children all over the world write a letter to Santa in the north pole. This letter is then decided if the child has been naught or nice. If the child is nice the elves will then assemble a gift in a factory for the child. After the gift is wrapped it will then be delivered at a specific time by Santa and his reindeers.

- The process that occurs at the north pole is very similar to a company. It is important that each piece of information is stored in the database. This will help prevent an error from occurring. The north pole will be organized when Christmas comes so that each child will receive their gift from Santa.

Known Problems

- A known problem is concerning the factories and the items table. There is no linkage between a specific item and the category it should be in. No check constraint exists to avoid the possible Human error when filling out a items data entry.
- In many of the tables date and time is affiliated with the unique ID's. However being at the north pole it's difficult to keep track of time due to the longitude lines all converging there. This resulted in a difficulty to measure time from all over the world.

Future Enhancements

1. The queries can implement an average of letters that children send to the north pole each year. With the knowledge of the average letters this will allow Santa in the future to prepare hiring more or less elves for Christmas.

2. In the future Santa can implement adults that still believe in Santa to receive a gift for Christmas. This will prevent a limitation of the age up to 21. This will involve more data to be stored and more columns to be added due to adults wanting different kinds of gifts.

	letter_id integer	first_name text	last_name text	age integer	verdict text	wishes text
1	665622	Alan	Labouseur	21	Nice	White Porsche